

Contents

Visual Studio SDK

Instalar Visual Studio SDK

Empezar a desarrollar extensiones de Visual Studio

Crear la primera extensión: Hello World

Novedades del SDK de Visual Studio 2019

- Novedades del SDK de Visual Studio 2019

- Extensiones cargadas automáticamente y sincrónicamente

Novedades del SDK de Visual Studio 2017

- Novedades del SDK de Visual Studio 2017

- Cambios importantes en la extensibilidad

- Compatibilidad con Ngen en VSIX v3

- Instalar fuera de la carpeta de extensiones

- Migrar el registro COM del depurador

- Preguntas más frecuentes

Migrar proyectos de extensibilidad a Visual Studio 2017

Extensiones de Roundtrip para Visual Studio

Protocolo de servidor de lenguaje

- Protocolo de servidor de lenguaje

- Adición de una extensión del protocolo de servidor de lenguaje

Abrir carpeta

- Abrir carpeta

- Áreas de trabajo

- Contextos de archivo

- Indización

- Servicios de lenguaje

- Build

Ampliación de menús y comandos

- Ampliación de menús y comandos

- Creación de una extensión con un comando de menú

- Adición de un menú a la barra de menús de Visual Studio
- Enlace de métodos abreviados de teclado a elementos de menú
- Adición de un submenú a un menú
- Adición de una lista Usados más recientemente a un submenú
- Creación de grupos reutilizables de botones
- Adición de iconos a comandos de menú
- Cambio del texto de un comando de menú
- Cambio de la apariencia de un comando
- Actualización de la interfaz de usuario
- Adaptación de comandos de menú

Adición de comandos a barras de herramientas

- Adición de comandos a barras de herramientas
- Adición de una barra de herramientas
- Adición de una barra de herramientas a una ventana de herramientas
- Adición de un comando a la barra de herramientas del Explorador de soluciones
- Adición de un controlador de menú a una barra de herramientas
- Adición dinámica de elementos de menú

Ampliación y personalización de ventanas de herramientas

- Ampliación y personalización de ventanas de herramientas
- Creación de una extensión con una ventana de herramientas
- Adición de una ventana de herramientas
- Registro de una ventana de herramientas
- Creación de una ventana de herramientas de varias instancias
- Apertura de una ventana de herramientas dinámica
- Adición de un menú contextual en una ventana de herramientas
- Adición de búsqueda a una ventana de herramientas
- Suscripción a un evento
- Ampliación de la ventana de salida
- Ampliación de las propiedades, la lista de tareas, la salida y las ventanas de opciones
- Creación y administración de cuadros de diálogo modales

Editor y extensiones de servicio de lenguaje

- Editor y extensiones de servicio de lenguaje

Creación de una extensión con una plantilla de elemento de editor

Ampliación del editor y los servicios de lenguaje

Ampliación del editor y los servicios de lenguaje

Introducción al servicio de lenguaje y las extensiones de editor

Dentro del editor

Managed Extensibility Framework en el editor

Servicio de lenguaje y puntos de extensión del editor

Importaciones del editor

Compatibilidad con EditorConfig

Tutorial: Vinculación de un tipo de contenido con una extensión de nombre de archivo

Tutorial: Creación de un glifo de margen

Tutorial: Personalización de la vista de texto

Tutorial: Creación de un elemento de gráfico de vista, comandos y configuración (guías de columnas)

Tutorial: Resaltado de texto

Tutorial: Esquemmatización

Tutorial: Visualización de llaves coincidentes

Tutorial: Visualización de información rápida sobre herramientas

Tutorial: Visualización de ayuda de signatura

Tutorial: Visualización de finalización de instrucciones

Tutorial: Implementación de fragmentos de código

Tutorial: Visualización de sugerencias de bombilla

Tutorial: Uso de un comando de shell con una extensión del editor

Tutorial: Uso de una tecla de método abreviado con una extensión del editor

Tutorial: Acceso al objeto DTE desde una extensión del editor

Creación de diseñadores y editores personalizados

Creación de diseñadores y editores personalizados

Tutorial: Creación de un editor personalizado

Tutorial: Adición de características a un editor personalizado

Inicialización de diseñador y configuración de metadatos

Compatibilidad de la característica Deshacer en los diseñadores

Colores de sintaxis en editores personalizados

Datos de documento y vista de documento en editores personalizados

Datos de documento y vista de documento en editores personalizados

Compatibilidad con vistas de varios documentos

Inserción simplificada

Asociación de vistas a datos de documentos

Administración de propietarios de bloqueo de documentos

Vistas de una sola pestaña y varias pestañas

Abrir editores específicos de proyecto

Abrir editores estándar

Abrir editores para documentos abiertos

Ampliación de proyectos

Ampliación de proyectos

Creación de un sistema de proyectos básico, parte 1

Creación de un sistema de proyectos básico, parte 2

Guardado de datos en archivos de proyecto

Comprobación de subtipos de un proyecto en tiempo de ejecución

Adición y eliminación de páginas de propiedades

Adición de un atributo a un elemento de proyecto

Conservación de la propiedad de un elemento de proyecto

Administración de proyectos de Windows universal

Ampliación de soluciones

Ampliación de soluciones

Ampliación del filtro del Explorador de soluciones

Administración de la carga de proyectos en una solución

Creación de plantillas de proyecto y de elemento personalizadas

Creación de plantillas de proyecto y de elemento personalizadas

Actualización de plantillas de proyecto y elemento para Visual Studio 2017 personalizadas

Solución de problemas de detección de plantillas

Usar asistentes con plantillas de proyectos

Referencia de esquema de plantillas de Visual Studio

Referencia de esquema de plantillas de Visual Studio

AppliesTo (Elemento, Plantillas de Visual Studio)

Assembly (Elemento, Plantillas de Visual Studio)

Assembly (Elemento, extensión del Asistente para plantillas de Visual Studio)

Atributo y elemento BuildOnLoad (plantillas de Visual Studio)

BuildProjectOnload (Elemento, Plantillas de Visual Studio)

CreateInPlace (Plantillas de Visual Studio)

CreateNewFolder (Elemento, Plantillas de Visual Studio)

CustomDataSignature (Elemento, Plantillas de Visual Studio)

CustomParameter (Elemento, Plantillas de Visual Studio)

CustomParameters (Elemento, Plantillas de Visual Studio)

DefaultName (Elemento, Plantillas de Visual Studio)

Description (Elemento, Plantillas de Visual Studio)

EnableEditOfLocationField (Elemento, Plantillas de Visual Studio)

EnableLocationBrowseButton (Elemento, Plantillas de Visual Studio)

Folder (Elemento, Plantillas de proyecto de Visual Studio)

FullClassName (Elemento, extensión del Asistente para plantillas de Visual Studio)

Hidden (Elemento, Plantillas de Visual Studio)

Icon (Elemento, Plantillas de Visual Studio)

LocationField (Elemento, Plantillas de proyecto de Visual Studio)

LocationFieldMRUPrefix (Elemento, Plantillas de Visual Studio)

MaxFrameworkVersion (Elemento, Plantillas de Visual Studio)

Name (Elemento, Plantillas de Visual Studio)

NumberOfParentCategoriesToRollUp (Plantillas de Visual Studio)

PreviewImage (Elemento, Plantillas de Visual Studio)

Project (Elemento, Plantillas de Visual Studio)

ProjectCollection (Elemento, Plantillas de Visual Studio)

ProjectItem (Elemento, Plantillas de proyecto de Visual Studio)

ProjectItem (Elemento, Plantillas de elementos de Visual Studio)

ProjectSubType (Elemento, Plantillas de Visual Studio)

ProjectTemplateLink (Elemento, Plantillas de Visual Studio)

ProjectType (Elemento, Plantillas de Visual Studio)

PromptForSaveOnCreation (Elemento, Plantillas de Visual Studio)

ProvideDefaultName (Elemento, Plantillas de Visual Studio)

[Reference \(Elemento, Plantillas de Visual Studio\)](#)

[References \(Elemento, Plantillas de Visual Studio\)](#)

[RequiredFrameworkVersion \(Elemento, Plantillas de Visual Studio\)](#)

[RequiredPlatformVersion \(Elemento, Plantillas de Visual Studio\)](#)

[SDKReference \(Elemento, Plantillas de Visual Studio\)](#)

[ShowByDefault \(Plantillas de Visual Studio\)](#)

[SolutionFolder \(Elemento, Plantillas de Visual Studio\)](#)

[SortOrder \(Elemento, Plantillas de Visual Studio\)](#)

[SupportsCodeSeparation \(Elemento, Plantillas de Visual Studio\)](#)

[SupportsLanguageDropDown \(Elemento, Plantillas de Visual Studio\)](#)

[SupportsMasterPage \(Elemento, Plantillas de Visual Studio\)](#)

[TargetPlatformName \(Elemento, Plantillas de Visual Studio\)](#)

[TemplateContent \(Elemento, Plantillas de Visual Studio\)](#)

[TemplateData \(Elemento, Plantillas de Visual Studio\)](#)

[TemplateGroupID \(Elemento, Plantillas de Visual Studio\)](#)

[TemplateID \(Elemento, Plantillas de Visual Studio\)](#)

[VSTemplate \(Elemento, Plantillas de Visual Studio\)](#)

[WizardData \(Elemento, Plantillas de Visual Studio\)](#)

[WizardExtension \(Elemento, Plantillas de Visual Studio\)](#)

[Referencia de esquema de manifiesto de plantillas de Visual Studio](#)

[Ampliación de configuración y opciones de usuario](#)

[Ampliación de configuración y opciones de usuario](#)

[Creación de una página de opciones](#)

[Creación de una categoría de configuración](#)

[Uso del almacén de configuración](#)

[Obtención de información de servicio desde el almacén de configuración](#)

[Escritura en el almacén de configuración de usuario](#)

[Ampliación de propiedades y la ventana de propiedades](#)

[Ampliación de propiedades y la ventana de propiedades](#)

[Exposición de las propiedades en la ventana de propiedades](#)

[Obtención de las propiedades del proyecto](#)

[Ampliación de otras partes de Visual Studio](#)

[Ampliación de otras partes de Visual Studio](#)

Creación de una extensión con un VSPackage

Creación de un control de cuadro de herramientas de WPF

Creación de un control de cuadro de herramientas de Windows Forms

Ampliación de la barra de estado

Creación de una página de inicio personalizada

- Creación de una página de inicio personalizada

- Tutorial: Adición de XAML personalizado a la página de inicio

- Adición de control de usuario a la página de inicio

- Adición de comandos de Visual Studio a una página de inicio

- Tutorial: Guardado de la configuración de usuario en una página de inicio

- Implementación de páginas de inicio personalizadas

Uso del registro de actividad

Extensibilidad de proyectos de Visual Studio C++

Instrucciones para la experiencia de usuario de Visual Studio

- Instrucciones para la experiencia de usuario de Visual Studio

- Conceptos básicos de UX

- Fuentes y formato

- Colores y estilos

- Colores y estilos

- Colores compartidos

- Referencia de valor de color

- Imágenes e iconos

- Imágenes e iconos

- Diccionario del lenguaje visual

- Menús y comandos

- Patrones de interacción

- Patrones de interacción

- Patrones de aplicación

- Patrones de control comunes

- Patrones compuestos

- Texto de UI y ayuda

- Diseño de cuadros de diálogo

- Notificaciones y progreso
- Animaciones
- Herramientas de evaluación
- Compatibilidad con reconocimiento de monitor
- Solución de problemas de DPI
- Catálogo y servicio de imágenes
- Uso y proporción de servicios
 - Uso y proporción de servicios
 - Obtener un servicio
 - Prestar un servicio
 - Prestar un servicio asincrónico de Visual Studio
 - Solucionar problemas de servicios
- Administración de VSPackages
 - Administración de VSPackages
 - Instancia experimental
 - Registro y anulación del registro de VSPackages
 - Carga de VSPackages
 - Administración de varios subprocessos en código administrado
 - Usar AsyncPackage para cargar VSPackages en segundo plano
 - Usar el contexto de interfaz de usuario basada en reglas para extensiones de Visual Studio
 - Diagnosticar el rendimiento de la extensión
 - Diagnosticar retrasos de la interfaz de usuario causados por las extensiones
 - Solución de problemas de VSPackages
- Envío de extensiones de Visual Studio
 - Envío de extensiones de Visual Studio
 - Plantilla de proyecto de VSIX
 - Introducción a la plantilla de proyecto de VSIX
 - Tutorial: Publicación de una extensión de Visual Studio
 - Tutorial: Publicación de una extensión de Visual Studio mediante la línea de comandos
 - Tutorial: Creación de un paquete de extensión
 - Agregar una dependencia a un paquete VSIX
 - Anatomía de un paquete VSIX

Anatomía de un paquete VSIX

Referencia de esquema 2.0 de la extensión VSIX

Estructura del archivo [Content_types].xml

Diseñador de manifiestos de VSIX

Localización de paquetes VSIX

Localización de paquetes VSIX

Referencia de esquema 2.0 del paquete de idioma de VSIX

Actualizar una extensión de Visual Studio

Preparación de las extensiones para la implementación de Windows Installer

Firma de paquetes VSIX

Private Galleries

Private Galleries

Crear una fuente Atom para una galería privada

Administrar una galería privada mediante la configuración del Registro

Compatibilidad con varias versiones de Visual Studio

Compatibilidad con varias versiones de Visual Studio

Elección entre VSPackages compartidos y con versiones

Registro de extensiones de nombre de archivo para implementaciones en paralelo

Registro de extensiones de nombre de archivo para implementaciones en paralelo

Acerca de las extensiones de nombre de archivo

Especificación de identificadores de archivo para extensiones de nombre de archivo

Registro de verbos para extensiones de nombre de archivo

Administración de asociaciones de archivos en paralelo

Localización de Visual Studio

Inicio de Visual Studio con DTE

Dentro de Visual Studio SDK

Dentro de Visual Studio SDK

Visual Studio Shell

Comandos, menús y barras de herramientas

Comandos, menús y barras de herramientas

Cómo VSPackages agrega elementos de la interfaz de usuario

Archivos de tabla de comandos de Visual Studio (.Vsct)

Archivos de tabla de comandos de Visual Studio (.Vsct)

Creación de archivos .Vsct

Diseñar archivos de tabla de comandos XML (.Vsct)

Crear un archivo .Vsct

Marcas de la línea de comandos del compilador de VSCT

Ubicación predeterminada de comando, grupo y barra de herramientas

Grupos, menús y comandos definidos por el IDE

Grupos, menús y comandos definidos por el IDE

GUID e identificadores de menús de Visual Studio

GUID e identificadores de barras de herramientas de Visual Studio

GUID e identificadores de comandos de Visual Studio

Diseño de comandos

Diseño de comandos

Implementación de comandos

Disponibilidad de los comandos

Algoritmo de enrutamiento de comandos

Orientaciones para la colocación de comandos

Optimización de comandos de menú y barra de herramientas

Puesta a disposición de comandos

Comandos y menús que usan ensamblados de interoperabilidad

Comandos y menús que usan ensamblados de interoperabilidad

Determinación del estado de los comandos mediante ensamblados de interoperabilidad

Contratos de comandos en los ensamblados de interoperabilidad

Registro de controladores de comandos de ensamblado de interoperabilidad

Uso de ensamblados de interoperabilidad de Visual Studio

Enrutamiento de comandos en VSPackages

VSPackages

VSPackages

Especificación de la ubicación del archivo de VSPackage en el Shell de VS

Recursos de VSPackages

Procedimientos recomendados para seguridad en VSPackages

Registro de VSPackages

Ventanas de documento

Ventanas de documento

Tabla de documentos en ejecución

Tabla de documentos en ejecución

Uso de RDT_ReadLock

Persistencia y tabla de documentos en ejecución

Carga de documentos retrasada

Extensibilidad de servicio de lenguaje heredado

Extensibilidad de servicio de lenguaje heredado

Migración de un servicio de lenguaje heredado

Fundamentos de servicio de lenguaje heredado

Desarrollo de un servicio de lenguaje heredado

Desarrollo de un servicio de lenguaje heredado

Modelo de un servicio de lenguaje heredado

Interfaces de servicio de lenguaje heredado

Intercepción de comandos del servicio de lenguaje heredado

Intercepción de comandos del servicio de lenguaje heredado

Comandos importantes para los filtros del servicio de lenguaje

Registro de un servicio de lenguaje heredado2

Lista de comprobación: creación de un servicio de lenguaje heredado

Características del servicio de lenguaje heredado2

Características del servicio de lenguaje heredado2

Colores de la sintaxis en un servicio de lenguaje heredado

Colores de la sintaxis en un servicio de lenguaje heredado

Implementación de colores de la sintaxis

Usar elementos coloreables integrados

Elementos coloreables personalizados

Formato automático en un servicio de lenguaje heredado

Información de parámetros en un servicio de lenguaje heredado1

Finalización de instrucciones en un servicio de lenguaje heredado

Esquematización y texto oculto en un servicio de lenguaje heredado

Esquematización y texto oculto en un servicio de lenguaje heredado

Compatibilidad de la esquematización en un servicio de lenguaje heredado

Provisión de compatibilidad con texto oculto en un servicio de lenguaje heredado

Provisión de compatibilidad con esquematización ampliada en un servicio de lenguaje heredado

Compatibilidad del servicio de lenguaje con la depuración

Implementación de un servicio de lenguaje heredado1

Implementación de un servicio de lenguaje heredado1

Información general del servicio de lenguaje heredado

Implementación de un servicio de lenguaje heredado2

Registro de un servicio de lenguaje heredado1

Escáner y analizador del servicio de lenguaje heredado

Tutorial: Creación de un servicio de lenguaje heredado

Tutorial: Obtención de una lista de fragmentos de código instalados (implementación heredada)

Características del servicio de lenguaje heredado1

Características del servicio de lenguaje heredado1

Coincidencia de llaves en un servicio de lenguaje heredado

Comentario de código en un servicio de lenguaje heredado

Propiedades de documento personalizadas en un servicio de lenguaje heredado

Esquematización en un servicio de lenguaje heredado

Cambio de formato de código en un servicio de lenguaje heredado

Compatibilidad con fragmentos de código en un servicio de lenguaje heredado

Información de parámetros en un servicio de lenguaje heredado2

Información rápida en un servicio de lenguaje heredado

Finalización de miembros en un servicio de lenguaje heredado

Finalización de palabras en un servicio de lenguaje heredado

Compatibilidad con la ventana Automático en un servicio de lenguaje heredado

Compatibilidad con la barra de navegación en un servicio de lenguaje heredado

Coloreado de sintaxis en un servicio de lenguaje heredado

Validación de puntos de interrupción en un servicio de lenguaje heredado

Compatibilidad con herramientas de exploración de símbolos

Compatibilidad con herramientas de exploración de símbolos

Registrar una biblioteca con el Administrador de objetos

Exponer listas de símbolos proporcionadas por la biblioteca al Administrador de objetos

Identificar símbolos en una biblioteca

Proyectos

Proyectos

Nueva generación de proyectos: aspectos técnicos, primera parte

Nueva generación de proyectos: aspectos técnicos, segunda parte

Tipos de proyecto

Tipos de proyecto

Conceptos básicos del tipo de proyecto

Creación de tipos de proyecto

Creación de tipos de proyecto

Decisiones de diseño del tipo de proyecto

Lista de comprobación: creación de nuevos tipos de proyecto

Creación de instancias de proyecto mediante generadores de proyecto

Uso de la plataforma de paquete administrado para implementar un tipo de proyecto (C#)

Registro de un tipo de proyecto

Persistencia de un proyecto

Uso de MSBuild

Adición de plantillas de proyecto y de elementos de proyecto

Adición de plantillas de proyecto y de elementos de proyecto

Contexto del proyecto

Prioridad del proyecto

Proyecto de archivos varios

Registro de plantillas para proyectos y elementos

Adición de elementos a los cuadros de diálogo Agregar nuevo elemento

Adición de directorios al cuadro de diálogo Nuevo proyecto

Adición de directorios al cuadro de diálogo Agregar nuevo elemento

Comandos definidos por el IDE para ampliar sistemas del proyecto

CATID para los objetos que se usan normalmente para ampliar proyectos

Apertura y guardado de elementos de proyecto

Apertura y guardado de elementos de proyecto

Visualización de archivos mediante el comando Abrir archivo

Visualización de archivos mediante el comando Abrir con

Guardado de un documento estándar

Guardado de un documento personalizado

Determinación del editor que abre un archivo en un proyecto

Administración de opciones de configuración

Administración de opciones de configuración

Información general de opciones de configuración

Páginas de propiedades

Configuración de soluciones

Objeto de configuración del proyecto

Configuración del proyecto para la compilación

Configuración del proyecto para administrar la implementación

Configuración del proyecto para la salida

Compatibilidad con control de código fuente

Compatibilidad con control de código fuente

Modelo para paquetes de control de código fuente

Decisiones de diseño del control de código fuente

Detalles de configuración del control de código fuente

Instrucciones de control de código fuente adicionales para proyectos y editores

Detalles de tiempo de ejecución del control de código fuente

Anidamiento de proyectos

Anidamiento de proyectos

Implementar proyectos anidados

Consideraciones para descargar y volver a cargar proyectos anidados

Compatibilidad del asistente con los proyectos anidados

Implementación del control de comandos para proyectos anidados

Filtrado del cuadro de diálogo AddItem para proyectos anidados

Actualización de proyectos

Arquitectura de tipos de proyecto

Arquitectura de tipos de proyecto

Elementos de un modelo de proyecto

Componentes principales del modelo de proyecto

Momento para la creación de tipos de proyecto

Jerarquías y selección

Jerarquías y selección

Jerarquías en Visual Studio

Selección y moneda en el IDE

Objetos de contexto de selección

Comentarios para el usuario

Subtipos de proyecto

Subtipos de proyecto

Diseño de subtipos de proyecto

Secuencia de inicialización de subtipos de proyecto

Propiedades y métodos ampliados por subtipos de proyecto

Conservación de datos en el archivo de proyecto de MSBuild

Interfaz de usuario de propiedades de proyecto

Ampliación del modelo de objetos del proyecto de base

Contribución al cuadro de diálogo Agregar nuevo elemento

Proyectos web

Proyectos web

Conceptos básicos del proyecto web

Compatibilidad del sitio web

Compatibilidad del sitio web

Plantillas de compatibilidad del sitio web

Atributos de compatibilidad del sitio web

Control de implementación especializada

Contribución al modelo de automatización

Contribución al modelo de automatización

Información general del modelo de automatización

Provisión de automatización de VSPackages

Exposición de objetos de proyecto

Modelado de proyectos

Exposición de eventos en Visual Studio SDK

Compatibilidad de automatización con páginas de opciones

Provisión de automatización para código

Provisión de automatización para Windows

Uso del modelo de automatización

Automatización para configuración y objetos SelectedItem

Soluciones

Soluciones

Archivo de la solución (.Sln)

Archivo de opciones de usuario de la solución (.Suo)

Extensión de propiedades

Extensión de propiedades

Información general sobre la ventana Propiedades

Directiva de plantilla y ventana Propiedades

Interfaces y campos de la ventana Propiedades

Lista de objetos de la ventana Propiedades

Botones de la ventana Propiedades

Cuadrícula para mostrar de Propiedades

Compatibilidad con las propiedades de proyecto y configuración

Opciones y páginas de opciones

Opciones y páginas de opciones

Creación de páginas de opciones

Compatibilidad con la configuración de usuario

Conceptos básicos del servicio

Conceptos básicos del servicio

Lista de servicios disponibles

Extensibilidad del depurador de Visual Studio

Extensibilidad del depurador de Visual Studio

Elección de una estrategia de implementación del motor de depuración

Ejemplos de depuración de Visual Studio

Creación de un motor de depuración personalizado

Creación de un motor de depuración personalizado

Registro de un motor de depuración personalizado

Habilitación de un programa que se va a depurar

Habilitación de un programa que se va a depurar

Obtención de un puerto

Registro del programa

Asociación al programa

Adjunto basado en el inicio

Envío de los eventos necesarios

Control de ejecución y evaluación de estado

Control de ejecución y evaluación de estado

Control de programas

Métodos relacionados con el punto de interrupción

Evaluación de la pila de llamadas

Evaluación de expresiones (SDK de depuración de Visual Studio)

Eventos de control

Envío de eventos

Envío de eventos

Orígenes de eventos (Visual Studio SDK)

Tipos de evento compatibles

Descripciones de eventos

Terminación y desasociación

Llamada a eventos del depurador

Llamada a eventos del depurador

Asociación a un programa y desasociación

Inicio del depurador

Terminación de un programa

Creación de un punto de interrupción

Cuando un punto de interrupción se enlaza o deja de estarlo

Errores de punto de interrupción

Llegada a un punto de interrupción

Eliminación de un punto de interrupción

Especificación de modo de interrupción

Escalonado en modo de interrupción

Evaluación de expresiones en el modo de interrupción

Control de excepciones (Visual Studio SDK)

Cómo: Depurar un motor de depuración personalizado

Introducción a la extensibilidad del depurador

Introducción a la extensibilidad del depurador

Guía básica para ampliar el depurador

Componentes del depurador

Componentes del depurador

Depurar paquete

Administrador de depuración del proceso

Administrador de depuración de sesión

Motor de depuración

Modos de funcionamiento

Evaluador de expresiones

Proveedor de símbolos

Visualizador de tipo y visor personalizado

Conceptos del depurador

Conceptos del depurador

Sesión de depuración

Servidores (Visual Studio SDK)

Proveedores de puertos

Puertos

Procesos

Nodos de programa

Programas

Subprocesos

Marcos de pila

Módulos

Puntos de interrupción (Visual Studio SDK)

Contextos de depurador

Contextos de depurador

Contexto de código

Posición de documento

Contexto de documento

Contexto de evaluación de expresiones

Tareas de depuración

Tareas de depuración

Problemas de seguridad

Inicio de un programa

Inicio de un programa

Notificación del puerto

Asociación tras el inicio

Asociación directa a un programa

Envío de eventos de inicio después de un inicio

Control de ejecución

Enlace de puntos de interrupción

Evaluación de expresiones

Visualización de datos

Escritura de un evaluador de expresiones de Common Language Runtime

Escritura de un evaluador de expresiones de Common Language Runtime

Common Language Runtime y evaluación de expresiones

Arquitectura del evaluador de expresiones

Arquitectura del evaluador de expresiones

Contexto de evaluación

Interfaces de evaluador de expresiones de clave

Registro de un evaluador de expresiones

Implementación de un evaluador de expresiones

Implementación de un evaluador de expresiones

Estrategia de implementación del evaluador de expresiones

Visualización de variables locales

Visualización de variables locales

Implementación de ejemplo de variables locales

Implementación de ejemplo de variables locales

Implementación de GetMethodProperty

Enumeración de variables locales

Obtención de propiedades locales

Obtención de valores locales

Evaluación de variables locales

Evaluación de una expresión de la ventana Inspección

Evaluación de una expresión de la ventana Inspección

Implementación de ejemplo de la evaluación de expresiones

Evaluación de una expresión de inspección

Cambio del valor de una variable local

Cambio del valor de una variable local

Implementación de ejemplo de cambio de valores

Implementación de visualizadores de tipo y visores personalizados

Implementación de un proveedor de puertos

Implementación de un proveedor de puertos

Implementación y registro de un proveedor de puertos

Interfaces de proveedor de puertos requeridas

Datos internos de extensiones paralelas para .NET Framework

Datos internos de extensiones paralelas para .NET Framework

Clase de tarea: miembros internos

Clase de tarea: miembros internos

m_action (Campo)

m_contingentProperties (Campo)

m_parent (Campo)

m_stateFlags (Campo)

m_stateObject (Campo)

m_taskId (Campo)

s_taskIdCounter (Campo)

TASK_STATE_CANCELED (Campo)

TASK_STATE_EXECUTED (Campo)

TASK_STATE_FAULTED (Campo)

TASK_STATE_RAN_TO_COMPLETION (Campo)

TASK_STATE_WAITING_ON_CHILDREN (Campo)

[SetNotificationForWaitCompletion \(Método\)](#)

[NotifyDebuggerOfWaitCompletion \(Método\)](#)

[TaskScheduler \(Clase\): miembros internos](#)

[TaskScheduler \(Clase\): miembros internos](#)

[GetScheduledTasksForDebugger \(Método\)](#)

[GetTaskSchedulersForDebugger \(Método\)](#)

[ContingentProperties \(Clase\): miembros internos](#)

[ContingentProperties \(Clase\): miembros internos](#)

[m_children \(Campo\)](#)

[AsyncTaskMethodBuilder \(Estructura\): miembros internos](#)

[AsyncTaskMethodBuilder \(Estructura\): miembros internos](#)

[AsyncTaskMethodBuilder.ObjectIdForDebugger \(Propiedad\)](#)

[AsyncTaskMethodBuilder.m_builder \(Campo\)](#)

[AsyncTaskMethodBuilder<TResult> \(Estructura\): miembros internos](#)

[AsyncTaskMethodBuilder<TResult> \(Estructura\): miembros internos](#)

[AsyncTaskMethodBuilder<TResult>.ObjectIdForDebugger \(Propiedad\)](#)

[AsyncTaskMethodBuilder<TResult>.m_task \(Campo\)](#)

[AsyncVoidMethodBuilder \(Estructura\): miembros internos](#)

[AsyncVoidMethodBuilder \(Estructura\): miembros internos](#)

[AsyncVoidMethodBuilder.ObjectIdForDebugger \(Propiedad\)](#)

[AsyncVoidMethodBuilder.m_objectIdForDebugger \(Campo\)](#)

[Control de código fuente](#)

[Control de código fuente](#)

[Conceptos básicos de la integración del control de código fuente](#)

[Información general de la integración del control de código fuente](#)

[Información general de la integración del control de código fuente](#)

[Novedades del control de código fuente](#)

[Creación de un complemento de control de código fuente](#)

[Creación de un complemento de control de código fuente](#)

[Introducción a los complementos de control de código fuente](#)

[Introducción a los complementos de control de código fuente](#)

[Instalar un complemento de control de código fuente](#)

Novedades de la API del complemento de control de código fuente, versión 1.3

Novedades de la API del complemento de control de código fuente, versión 1.2

Novedades de la API del complemento de control de código fuente, versión 1.2

Eliminación de archivos ~SAK

Aplicación de configuración en varias conexiones de proyecto

Creación de carpetas contenedoras primarias para soluciones

Comparación opcional de la carpeta de proyecto local con el almacén de control de código fuente

Eliminación de la información de control de código fuente de archivos .Proj y .Sln

Arquitectura de complemento de control de código fuente

Guía de pruebas para los complementos de control de código fuente

Guía de pruebas para los complementos de control de código fuente

Área de prueba 1: agregar a/abrir desde control de código fuente

Área de prueba 2: obtener de control de código fuente

Área de prueba 3: extraer del repositorio/cancelar extracción del repositorio

Área de prueba 4: insertar en el repositorio

Área de prueba 5: cambiar control de código fuente

Área de prueba 6: eliminar

Área de prueba 7: compartir

Área de prueba 8: cambio de complementos

Creación de un VSPackage de control de código fuente

Creación de un VSPackage de control de código fuente

Introducción a VSPackages de control de código fuente

Introducción a VSPackages de control de código fuente

Determinación de si se debe implementar un VSPackage de control de código fuente

Arquitectura de VSPackage de control de código fuente

Características de VSPackage de control de código fuente

Características de VSPackage de control de código fuente

Registro y selección (VSPackage de control de código fuente)

Editar/guardar consulta (VSPackage de control de código fuente)

Control de glifo (VSPackage de control de código fuente)

Interfaz de usuario personalizada (VSPackage de control de código fuente)

Elementos de diseño de VSPackage de control de código fuente

Elementos de diseño de VSPackage de control de código fuente

Estructura de VSPackage (VSPackage de control de código fuente)

Interfaces y servicios relacionados (VSPackage de control de código fuente)

Servicios proporcionados (VSPackage de control de código fuente)

Asistentes

Asistentes

Archivos de descripción del directorio de plantilla (.Vsdir)

Archivo de asistente (.Vsz)

Interfaz de asistente (IDTWizard)

Parámetros de contexto

Parámetros personalizados

Herramientas personalizadas

Herramientas personalizadas

Implementación de generadores de un solo archivo

Registro de generadores de un solo archivo

Exposición de tipos a diseñadores visuales

Utilidades VSSDK

Utilidades VSSDK

Utilidad RegPkg

Utilidad RegPkg

Solución de problemas del registro de paquete RegPkg

Utilidad CreatePkgDef

Utilidad CreateExpInstance

Herramientas de creación de temas de color

Herramientas de creación de temas de color

Editor de colores de VSIX

Compilador de colores de VSIX

Herramientas de servicio de imágenes

Herramientas de servicio de imágenes

[Manifest from Resources](#)

[Manifest to Code](#)

[Visor de la biblioteca de imágenes](#)

[Instalación de VSPackages con Windows Installer](#)

[Instalación de VSPackages con Windows Installer](#)

[Datos básicos de Windows Installer](#)

[Escenarios de instalación de VSPackage](#)

[Creación de un paquete de Windows Installer](#)

[Detección de requisitos del sistema](#)

[Administración de componentes](#)

[Selección del directorio de instalación de un VSPackage](#)

[Registro de VSPackage](#)

[Implementación de tipos de proyecto](#)

[Generar información del Registro para un instalador](#)

[Comandos que se deben ejecutar después de la instalación](#)

[Desinstalación de un VSPackage con Windows Installer](#)

[SDK del Visor de Ayuda de Microsoft](#)

[Soporte técnico para Visual Studio SDK](#)

[Referencia de Visual Studio SDK](#)

[Referencia de Visual Studio SDK](#)

[IntelliSenseHostFlags](#)

[Referencia del esquema XML de VSCT](#)

[Referencia del esquema XML de VSCT](#)

[CommandTable \(Elemento\)](#)

[Extern \(Elemento\)](#)

[Include \(Elemento\)](#)

[Define \(Elemento\)](#)

[Elemento Commands](#)

[Elemento Commands](#)

[Menus \(Elemento\)](#)

[Menus \(Elemento\)](#)

[Menu \(Elemento\)](#)

Elemento Groups

Elemento Groups

Elemento Group

Buttons (Elemento)

Buttons (Elemento)

Button (Elemento)

Bitmaps (Elemento)

Bitmaps (Elemento)

Bitmap (Elemento)

Combos (Elemento)

Combos (Elemento)

Combo (Elemento)

CommandPlacements (Elemento)

CommandPlacements (Elemento)

CommandPlacement (Elemento)

VisibilityConstraints (Elemento)

VisibilityConstraints (Elemento)

VisibilityItem (Elemento)

KeyBindings (Elemento)

KeyBindings (Elemento)

KeyBinding (Elemento)

UsedCommands (Elemento)

UsedCommands (Elemento)

UsedCommand (Elemento)

Elemento primario

Icon (Elemento)

Strings (Elemento)

Strings (Elemento)

ButtonText (Elemento)

CommandName (Elemento)

Command Flag (Elemento)

Symbols (Elemento)

[Symbols \(Elemento\)](#)

[GuidSymbol \(Elemento\)](#)

[GuidSymbol \(Elemento\)](#)

[IDSymbol \(Elemento\)](#)

[Atributos condicionales del esquema XML de VSCT](#)

[GUID y constantes de Visual Studio SDK](#)

[GUID y constantes de Visual Studio SDK](#)

[GUID de IDE](#)

[Constantes de IDE](#)

[Constantes de COM en código administrado](#)

[Complementos de control de código fuente](#)

[Complementos de control de código fuente](#)

[Funciones de API de complemento de control de código fuente](#)

[Funciones de API de complemento de control de código fuente](#)

[SccAdd \(Función\)](#)

[SccAddFilesFromSCC \(Función\)](#)

[SccAddFromScc \(Función\)](#)

[SccBackgroundGet \(Función\)](#)

[SccBeginBatch \(Función\)](#)

[SccCheckin \(Función\)](#)

[SccCheckout \(Función\)](#)

[SccCloseProject \(Función\)](#)

[SccCreateSubProject \(Función\)](#)

[SccDiff \(Función\)](#)

[SccDirDiff \(Función\)](#)

[SccDirQueryInfo \(Función\)](#)

[SccEndBatch \(Función\)](#)

[SccEnumChangedFiles \(Función\)](#)

[SccGet \(Función\)](#)

[SccGetCommandOptions \(Función\)](#)

[SccGetEvents \(Función\)](#)

[SccGetExtendedCapabilities \(Función\)](#)

SccGetParentProjectPath (Función)
SccGetProjPath (Función)
SccGetUserOption (Función)
SccGetVersion (Función)
SccHistory (Función)
SccInitialize (Función)
SccIsMultiCheckoutEnabled (Función)
SccOpenProject (Función)
SccPopulateDirList (Función)
SccPopulateList (Función)
SccProperties (Función)
SccQueryChanges (Función)
SccQueryInfo (Función)
SccRemove (Función)
SccRename (Función)
SccRunScc (Función)
SccSetOption (Función)
SccUncheckout (Función)
SccUninitialize (Función)
SccWillCreateSccFile (Función)

Funciones de devolución de llamada implementadas por el IDE

Funciones de devolución de llamada implementadas por el IDE

LPTEXTOUTPROC

POPLISTFUNC

QUERYCHANGESFUNC

POPDIRLISTFUNC

OPTNAMECHANGEPFN

Enumerators

Enumerators

Enumerador de código de comando

Enumerador de mensaje

Enumerador de código de estado de archivo

Enumerador de código de estado de directorio

Marcas de capacidad

Marcas de bits usadas por comandos específicos

Códigos de error

Cadenas usadas como claves para buscar un complemento de control de código fuente

Archivo MSSCCPRJ.SCC

Procedimientos recomendados para implementar un complemento de control de código fuente

Restricciones en las longitudes de cadena

Glosario del complemento de control de código fuente

Desactivar advertencias de compatibilidad para complementos de control de código fuente

Modificadores de la línea de comandos (Visual Studio SDK)

Modificadores de la línea de comandos (Visual Studio SDK)

Modificadores de la línea de comandos Devenv para el desarrollo de VSPackage

Adición de modificadores de la línea de comandos

Control de errores y valores devueltos

Objetos

Objetos

VSCodeWindow (Objeto)

VSCodeWindowManager (Objeto)

VSTextBuffer (Objeto)

VSTextView (Objeto)

Glosario de Visual Studio SDK

Creación de un kit de desarrollo de software

Creación de un kit de desarrollo de software

Comparación entre los SDK y los paquetes NuGet

Tutorial: Creación de un SDK con C# o Visual Basic

Tutorial: Creación de un SDK con C++

Tutorial: Creación de un SDK con JavaScript

Extensibilidad de la plataforma de compilación .NET ("Roslyn")

Extensibilidad de la plataforma de compilación .NET ("Roslyn")

[Introducción a los analizadores de Roslyn](#)

[Analizadores de Roslyn y biblioteca compatible con código para ImmutableArrays](#)

[Referencia de versión del paquete de Roslyn](#)

Visual Studio SDK

06/04/2020 • 5 minutes to read • [Edit Online](#)

El SDK de Visual Studio le ayuda a ampliar las características de Visual Studio o integrar nuevas características en Visual Studio. Puede distribuir las extensiones a otros usuarios, así como a Visual Studio Marketplace. Estas son algunas de las formas en que se puede ampliar Visual Studio:

- Agregar comandos, botones, menús y otros elementos de la interfaz de usuario al IDE
- Añadir ventanas de herramientas para nuevas funciones
- Amplíe IntelliSense para un lenguaje determinado o proporcione IntelliSense para nuevos lenguajes de programación
- Utilice bombillas para proporcionar sugerencias y sugerencias que ayuden a los desarrolladores a escribir mejor código
- Habilitar soporte para un nuevo idioma
- Agregar un tipo de proyecto personalizado
- Llega a millones de desarrolladores a través de Visual Studio Marketplace

Si nunca ha escrito una extensión de Visual Studio antes, debería encontrar más información acerca de estas características y en [Empezar a desarrollar extensiones](#) de Visual Studio .

Instalar Visual Studio SDK

El SDK de Visual Studio es una característica opcional en el programa de instalación de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Novedades del SDK de Visual Studio 2017

El SDK de Visual Studio tiene algunas características nuevas, como el formato VSIX v3, así como cambios importantes, que pueden requerir que actualice la extensión. Para obtener más información, vea [Novedades del SDK de Visual Studio 2017](#).

Directrices de experiencia de usuario de Visual Studio

Obtenga excelentes consejos para diseñar la interfaz de usuario para la extensión en las directrices de la experiencia del usuario de [Visual Studio](#).

También puede aprender a hacer que la extensión se vea muy bien en dispositivos de ppp altos con el artículo [Problemas de PPP de dirección](#).

Aproveche el [servicio y catálogo](#) de imágenes para una excelente gestión de imágenes y soporte para alta DPI y temas.

Buscar e instalar extensiones de Visual Studio existentes

Puede encontrar extensiones de Visual Studio en el cuadro de diálogo **Extensiones y actualizaciones** del menú **Herramientas**. Para obtener más información, vea [Buscar y usar extensiones](#) de Visual Studio . También puede encontrar extensiones en [Visual Studio Marketplace](#)

Referencia del SDK de Visual Studio

Puede encontrar la referencia de la API del SDK de Visual Studio en Referencia del SDK de [Visual Studio](#).

Ejemplos del SDK de Visual Studio

Puede encontrar ejemplos de código abierto de extensiones de SDK de VS en GitHub en [Visual Studio Samples](#). Este repositorio de GitHub contiene ejemplos que ilustran varias características extensibles en Visual Studio.

Otros recursos del SDK de Visual Studio

Si tiene preguntas sobre VSSDK o desea compartir sus experiencias desarrollando extensiones, puede usar el Foro de [extensibilidad](#) de Visual Studio o [extendVS Gitter Chatroom](#).

Puede encontrar más información en el [blog vsx Arcana](#) y una serie de blogs escritos por Microsoft MVPs:

- [Extensiones favoritas de Visual Studio](#)
- [Extensibilidad de Visual Studio](#)
- [Ampliación de Visual Studio](#)

Vea también

- [Crear una extensión con un comando de menú](#)
- [Cómo: Migrar proyectos de extensibilidad a Visual Studio 2017](#)
- [Preguntas frecuentes: Conversión de complementos a extensiones de VSPackage](#)
- [Administrar varios subprocesos en código administrado](#)
- [Extender menús y comandos](#)
- [Añadir comandos a las barras de herramientas](#)
- [Ampliar y personalizar las ventanas de herramientas](#)
- [Editor y extensiones de servicio de lenguaje](#)
- [Ampliar proyectos](#)
- [Ampliar la configuración y las opciones del usuario](#)
- [Crear plantillas de proyectos y elementos personalizadas](#)
- [Extender las propiedades y la ventana de propiedades](#)
- [Ampliar otras partes de Visual Studio](#)
- [Uso y prestación de servicios](#)
- [Administración de VSPackages](#)
- [Shell aislado de Visual Studio](#)
- [Enviar extensiones de Visual Studio](#)
- [Dentro de Visual Studio SDK](#)
- [Soporte técnico para Visual Studio SDK](#)
- [Referencia del SDK de Visual Studio](#)

Instalar Visual Studio SDK

06/04/2020 • 2 minutes to read • [Edit Online](#)

El SDK de Visual Studio (Kit de desarrollo de software) es una característica opcional en la instalación de Visual Studio. También puede instalar el SDK de VS más adelante.

Instale el SDK de Visual Studio como parte de una instalación de Visual Studio

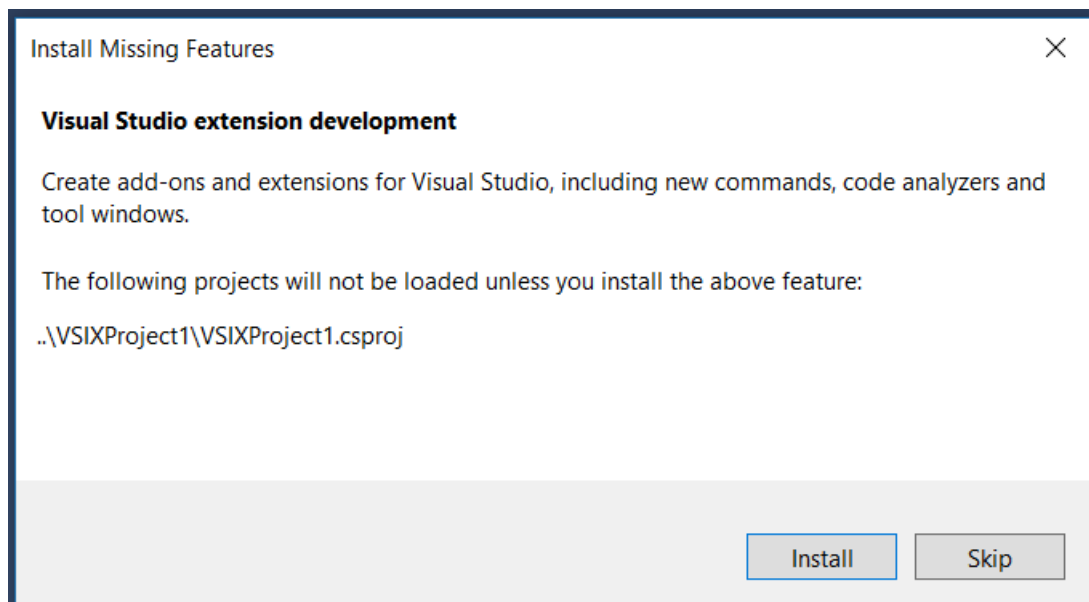
Para incluir el SDK de VS en la instalación de Visual Studio, instale la carga de trabajo de desarrollo de **extensiones** de Visual Studio en **Otros conjuntos de herramientas**. Esta carga de trabajo instalará el SDK de Visual Studio y los requisitos previos necesarios. Puede ajustar aún más la instalación seleccionando o anulando la selección de componentes en la vista **Resumen**.

Instale el SDK de Visual Studio después de instalar Visual Studio

Para instalar el SDK de Visual Studio después de completar la instalación de Visual Studio, vuelva a ejecutar el instalador de Visual Studio y seleccione la carga de trabajo de desarrollo de extensión de **Visual Studio**.

Instale el SDK de Visual Studio desde una solución

Si abre una solución con un proyecto de extensibilidad sin instalar primero el SDK de VS, se le pedirá un cuadro de diálogo **Instalar característica que falta** para instalar la carga de trabajo de desarrollo de extensión de Visual **Studio**:



Instale el SDK de Visual Studio desde la línea de comandos

Al igual que con cualquier carga de trabajo o componente de Visual Studio, también puede instalar la carga de trabajo de desarrollo de extensión de **Visual Studio** (ID: Microsoft.VisualStudio.Workload.VisualStudioExtension) desde la línea de comandos. Consulte [Usar parámetros de línea de comandos para instalar Visual Studio](#) para obtener más información sobre los modificadores de línea de comandos adecuados e instrucciones generales sobre cómo determinar los

identificadores de carga de trabajo o componentes.

Tenga en cuenta que debe usar el instalador de Visual Studio que coincida con la versión instalada de Visual Studio. Por ejemplo, si tiene Visual Studio Enterprise instalado en el equipo, debe ejecutar el instalador de Visual Studio Enterprise (*vs_enterprise.exe*).

Comienzo del desarrollo de extensiones de Visual Studio

06/04/2020 • 9 minutes to read • [Edit Online](#)

Si nunca ha escrito una extensión de Visual Studio antes, probablemente tenga algunas preguntas. Hemos enumerado algunos de los más comunes aquí. Si no ves la información que estás buscando, usa los botones de comentarios (**¿Fue útil esta página?** en la parte inferior de la pantalla) para preguntar lo que quieres.

NOTE

Este artículo se aplica a Visual Studio en Windows. Para Visual Studio para Mac, vea [Extender Visual Studio para Mac](#). Para Visual Studio Code, vea [Api de extensión de código de Visual Studio](#).

¿Qué software necesito para desarrollar extensiones de Visual Studio?

Debe instalar el SDK de Visual Studio además de Visual Studio para desarrollar extensiones de Visual Studio. Puede instalar el SDK de Visual Studio como parte de la instalación normal o puede instalarlo más adelante. Para obtener más información acerca de la instalación del SDK de Visual Studio, vea SDK de [Visual Studio](#).

¿Qué tipos de cosas puedo hacer con las extensiones de Visual Studio?

El cielo es el límite cuando se trata de imaginar diferentes extensiones de Visual Studio. Por supuesto, la mayoría de las extensiones tienen algo que ver con escribir código, pero ese no tiene que ser el caso. Estos son algunos ejemplos de los tipos de extensiones que puede crear:

- Compatibilidad con lenguajes que no se incluyen en Visual Studio, con coloración de sintaxis, IntelliSense y compatibilidad con compiladores y depuración
- Herramientas de productividad que amplían la experiencia principal del IDE con plantillas adicionales, refactorización de código, nuevos cuadros de diálogo o ventanas de herramientas
- Diseñadores específicos de dominio para escenarios como el diseño de datos o la compatibilidad con la nube

Para ver ejemplos de extensiones, consulte [Visual Studio Marketplace](#). Muchas extensiones son de código abierto y Marketplace incluye vínculos a su repositorio de GitHub.

¿Qué características de Visual Studio puedo ampliar?

En teoría, puede ampliar casi cualquier parte de Visual Studio: menús, barras de herramientas, comandos, ventanas, soluciones, proyectos, editores, etc.

En la práctica, hemos encontrado que las características que la mayoría de la gente quiere ampliar son comandos, menús y barras de herramientas, ventanas, IntelliSense y proyectos. Aquí hay enlaces a las secciones relevantes:

- [Ampliación de menús y comandos](#): agregue sus propios elementos a los menús y barras de herramientas de Visual Studio. Puede usarlos para iniciar la nueva funcionalidad de Visual Studio o sus propias aplicaciones auxiliares externas. También puede proporcionar accesos directos personalizados para los elementos de menú.
- [Ampliación y personalización](#) de ventanas de herramientas: amplíe las ventanas de herramientas existentes

o cree sus propias ventanas de herramientas. Por ejemplo, podría agregar nuevas propiedades a **Propiedades** o crear una nueva ventana de herramientas para agregar características adicionales.

- [Editor y extensiones](#) de servicio de lenguaje: agregue sus propias personalizaciones a IntelliSense proporcionado para los lenguajes de Visual Studio o cree compatibilidad con nuevos lenguajes de programación. Puede crear nuevas terminaciones de instrucciones, sugerencias y nuevas descripciones emergentes de QuickInfo. Con las bombillas, puede agregar sugerencias de refactorización y correcciones de código para admitir nuevos lenguajes de programación.
- [Ampliación de proyectos](#)
- [Ampliación de configuración y opciones de usuario](#)
- [Ampliación de propiedades y la ventana de propiedades](#)
- [Ampliación de otras partes de Visual Studio](#)
- [Shell aislado de Visual Studio](#)

¿Qué plantillas de proyecto proporciona VSSDK?

Los dos tipos principales de extensiones son VSPackages y extensiones MEF. En general, las extensiones de VSPackage se usan para las extensiones que usan o extienden comandos, ventanas de herramientas y proyectos. Las extensiones MEF se usan para ampliar o personalizar el editor de Visual Studio.

Para las extensiones de Visual C y Visual Basic, el VSSDK proporciona una plantilla de proyecto VSIX vacía que puede usar junto con las nuevas plantillas de elemento que crean comandos de menú, ventanas de herramientas y extensiones de editor. También puede usar esta plantilla para empaquetar plantillas de proyecto, fragmentos de código y otros artefactos para su distribución a otros usuarios.

Para C++, el asistente de VSPackage proporciona el código para agregar comandos de menú, ventanas de herramientas y editores personalizados.

La plantilla Shell aislado se usa para empaquetar una extensión en una versión del shell de Visual Studio que puede marcar y distribuir como propia. Los siguientes temas le muestran cómo empezar a utilizar cada tipo de extensión:

- Comandos de menú: [Creación de una extensión con un comando de menú](#)
- Ventanas de herramientas: [Creación de una extensión con una ventana de herramientas](#)
- Extensiones del editor: [Creación de una extensión con una plantilla](#) de elemento de editor
- VSPackages básicos: [crear una extensión con un VSPackage](#)
- Plantilla de proyecto VSIX: [Introducción a la plantilla](#) de proyecto VSIX

¿Cómo obtengo mi extensión para que se parezca a Visual Studio?

Obtenga excelentes consejos para diseñar la interfaz de usuario para la extensión en [Visual Studio User Experience Guidelines](#).

¿Dónde puedo encontrar ejemplos de código VSSDK?

Cada uno de los vínculos enumerados en la sección anterior tiene tutoriales paso a paso que muestran cómo implementar características específicas. También puede encontrar ejemplos de VSSDK de código abierto en GitHub en [Visual Studio Samples](#).

¿Cómo puedo distribuir mi extensión?

Puede instalar su extensión en otro equipo o enviarlo a sus amigos como un archivo .vsix, que se instala haciendo doble clic en él. Puede obtener más información sobre los paquetes VSIX en [Shipping Visual Studio Extensions](#).

También puede publicar la extensión en Visual Studio Marketplace, lo que la hace visible para un gran número de clientes de Visual Studio. Para obtener un ejemplo de empaquetado de una extensión en Marketplace, vea [Tutorial: publicación](#) de una extensión de Visual Studio . Para obtener más información acerca de lo que debe hacer para publicar en Marketplace, vea [Productos y extensiones para Visual Studio](#).

Vea también

- [Extender Visual Studio para Mac](#)
- [Ampliación del código de Visual Studio](#)

Crea tu primera extensión: Hello World

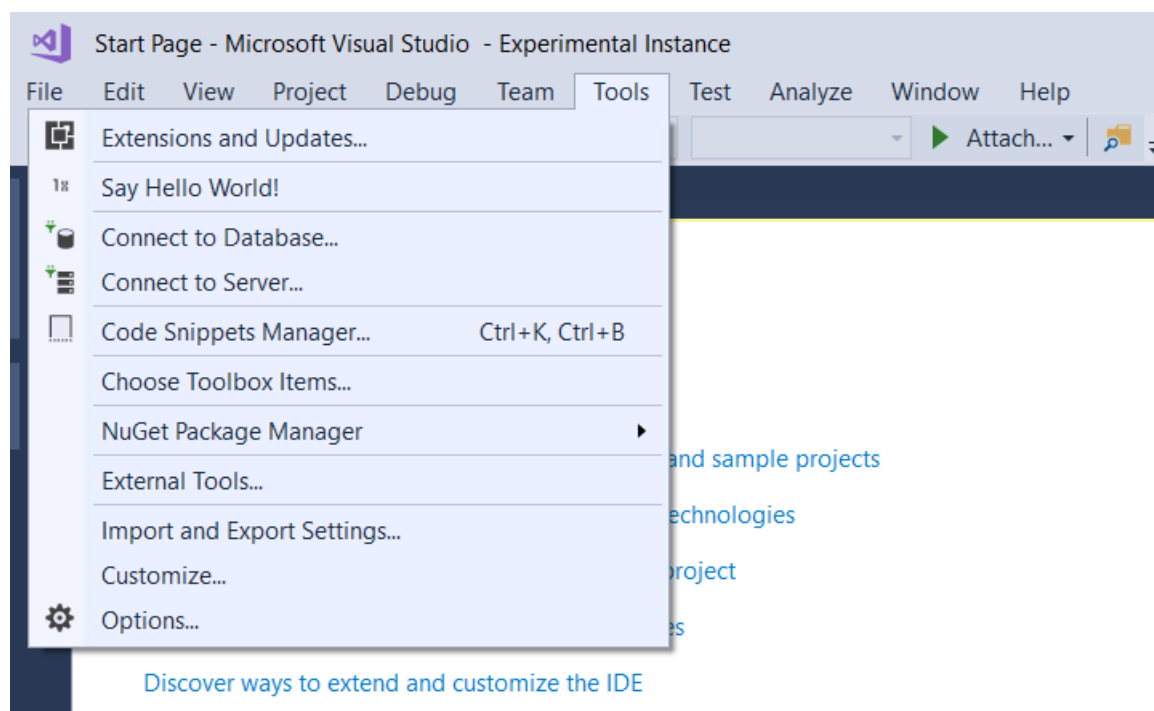
06/04/2020 • 7 minutes to read • [Edit Online](#)

Este ejemplo de Hello World le guía a través de la creación de la primera extensión para Visual Studio. En este tutorial se muestra cómo agregar un nuevo comando a Visual Studio.

En el proceso, aprenderá a:

- [Crear un proyecto de extensibilidad](#)
- [Agregar un comando personalizado](#)
- [Modificar el código fuente](#)
- [Ejecútelo.](#)

En este ejemplo, usará Visual C para agregar un botón de menú personalizado denominado "Say Hello World!" que se ve así:



NOTE

Este artículo se aplica a Visual Studio en Windows. Para Visual Studio para Mac, vea Tutorial de [extensibilidad en Visual Studio para Mac](#).

Prerrequisitos

Antes de empezar, asegúrese de que ha instalado la carga de trabajo de desarrollo de extensiones de **Visual Studio**, que incluye la plantilla VSIX que necesitará y código de ejemplo.

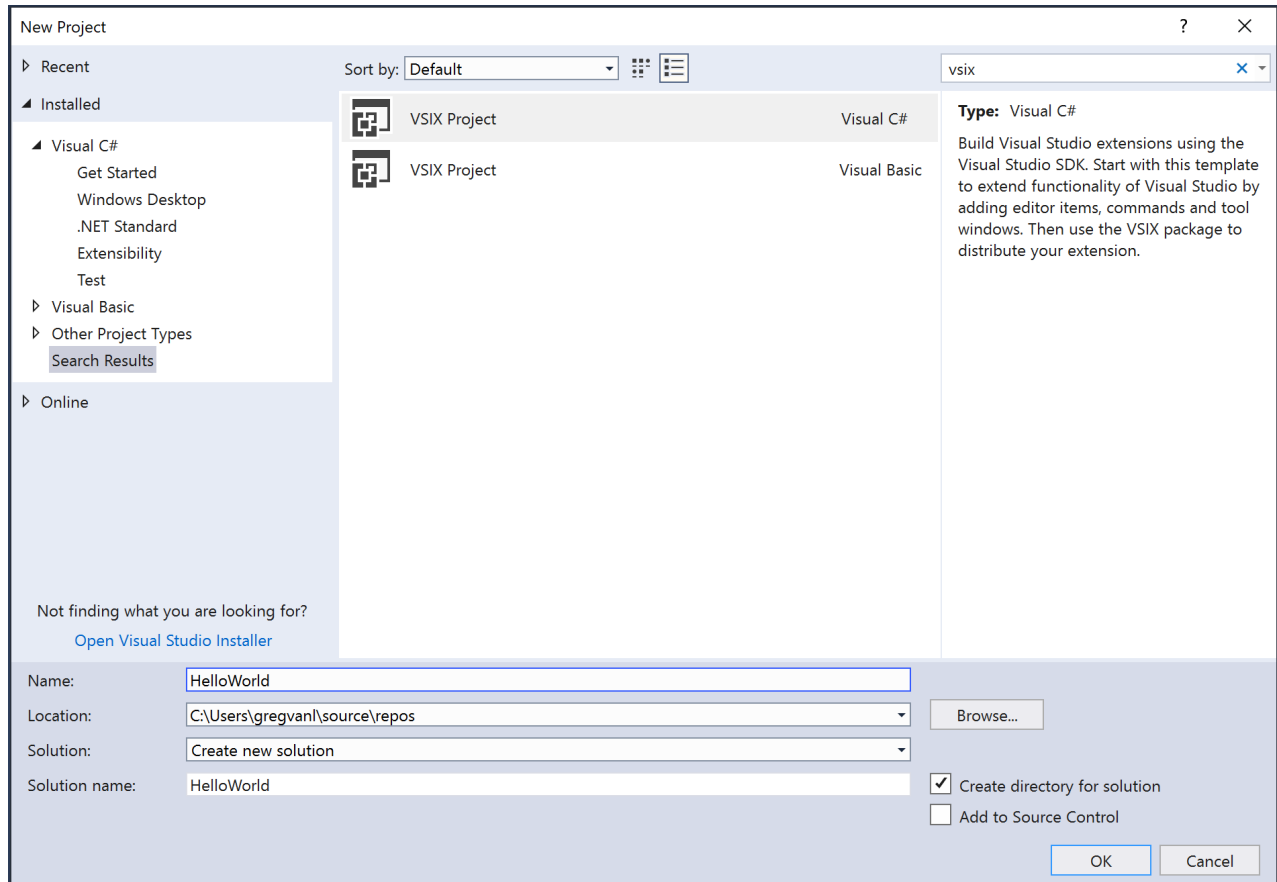
NOTE

Puede usar cualquier edición de Visual Studio (Community, Professional o Enterprise) para crear un proyecto de extensibilidad de Visual Studio.

Crear un proyecto de extensibilidad

Paso 1. En el menú **Archivo**, seleccione **Nuevo > Proyecto**.

Paso 2. En el cuadro de búsqueda de la parte superior derecha, escriba "vsix" y seleccione el proyecto Visual C. **VSIX**. Escriba "HelloWorld" para el **Nombre** en la parte inferior del cuadro de diálogo y seleccione **Aceptar**.



Ahora debería ver la página Introducción y algunos recursos de ejemplo.

Si necesita dejar este tutorial y volver a él, puede encontrar su nuevo proyecto HelloWorld en la página de inicio en la sección **Reciente**.

Paso 1. En el menú **Archivo**, seleccione **Nuevo > Proyecto**. Busque "vsix" y seleccione el **proyecto VSIX** de Visual C. y, a continuación, **Siguiente**.

Paso 2. Escriba "HelloWorld" para el nombre del **proyecto** y seleccione **Crear**.

×

Configure your new project

VSIX ProjectC#WindowsExtensions

Project name

HelloWorld

Location

C:\Users\gregvanl\source\repos

...

Solution name ⓘ

HelloWorld

☐ Place solution and project in the same directory

Back

Create

Ahora debería ver el proyecto HelloWorld en el Explorador de **soluciones**.

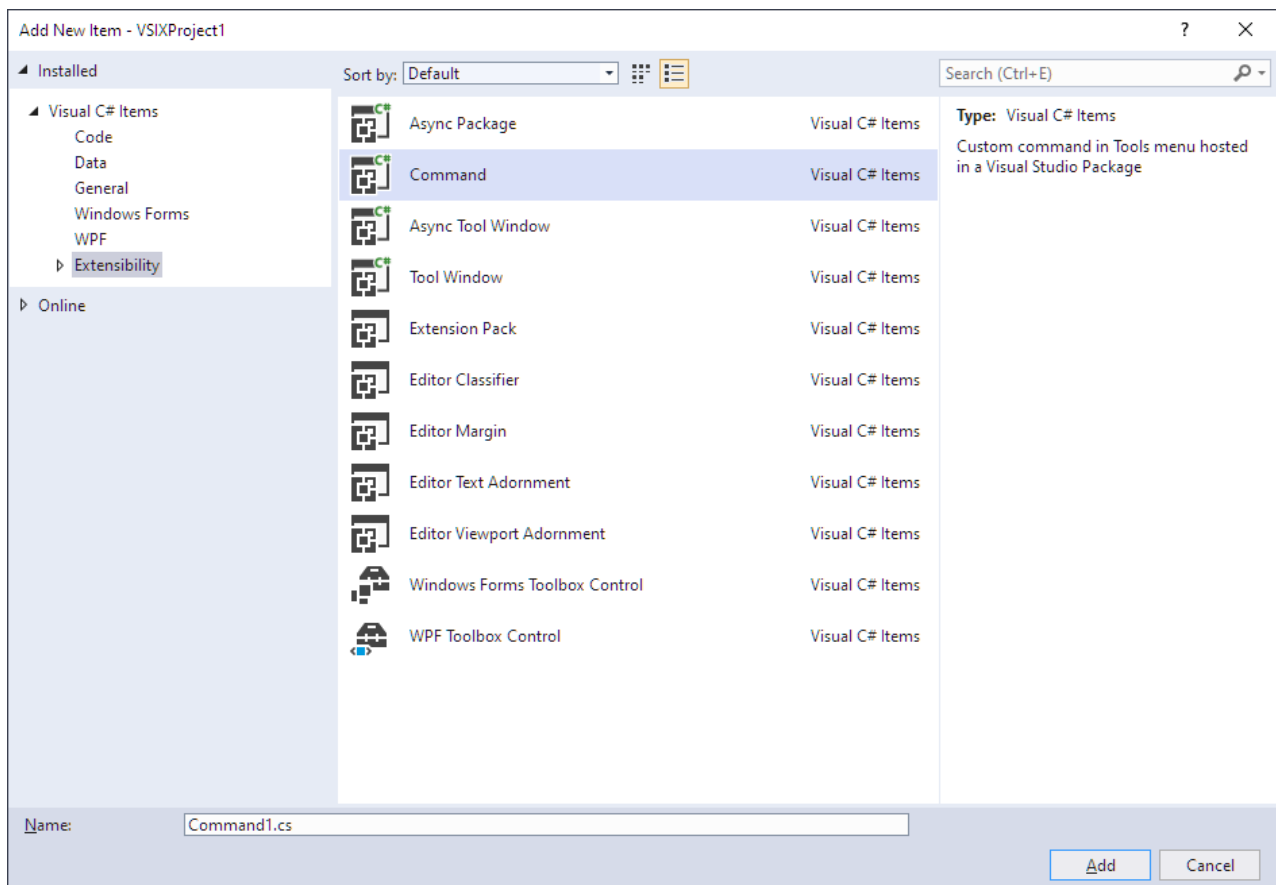
Agregar un comando personalizado

Paso 1. Si selecciona el archivo de manifiesto *.vsixmanifest*, puede ver qué opciones se pueden cambiar, como la descripción, el autor y la versión.

Paso 2. Haga clic con el botón derecho en el proyecto (no en la solución). En el menú contextual, seleccione **Agregar**, a continuación, **Nuevo elemento**.

Paso 3. Seleccione la sección **Extensibilidad** y, a continuación, elija **Comando**.

Paso 4. En el campo **Nombre** de la parte inferior, escriba un nombre de archivo como *Command.cs*.



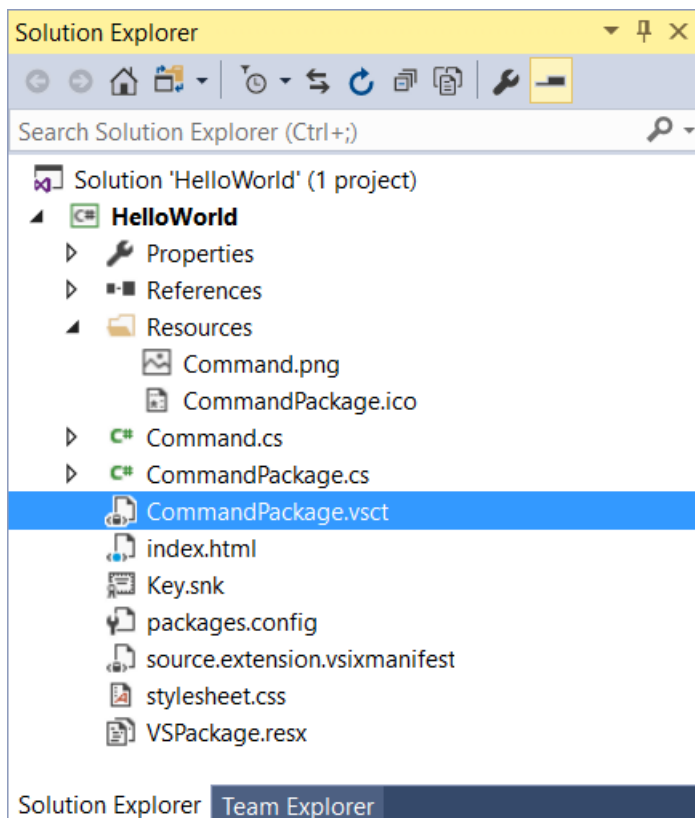
El nuevo archivo de comandos está visible en el Explorador de **soluciones**. En el nodo **Recursos**, encontrará otros archivos relacionados con el comando. Por ejemplo, si desea modificar la imagen, el archivo PNG está aquí.

Modificar el código fuente

En este punto, el comando y el texto del botón se generan automáticamente y no son muy interesantes. Puede modificar el archivo VSCT y el archivo CS si desea realizar cambios.

- El archivo VSCT es donde puede cambiar el nombre de los comandos, así como definir dónde van en el sistema de comandos de Visual Studio. A medida que explora el archivo VSCT, observará comentarios que explican lo que controla cada sección del código VSCT.
- El archivo CS es donde puede definir acciones, como el controlador de clics.

Paso 1. En el **Explorador de soluciones**, busque el archivo VSCT para el nuevo comando. En este caso, se llamará *CommandPackage.vsct*.



Paso 1. En el **Explorador** de soluciones , busque el archivo VSCT para el paquete VS de extensión. En este caso, se llamará *HelloWorldPackage.vscx*.

Paso 2. Cambie `ButtonText` el `Say Hello World!` parámetro a .

```
...
<Button guid="guidCommandPackageCmdSet" id="CommandId" priority="0x0100" type="Button">
  <Parent guid="guidCommandPackageCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <ButtonText>Say Hello World!</ButtonText>
  </Strings>
</Button>
...
```

Paso 3. Vuelva al **Explorador** de soluciones y busque el archivo *Command.cs*. En `Execute` el método, `message` cambie `string.Format(..)` `Hello World!` la cadena de a .

```
...
private void Execute(object sender, EventArgs e)
{
    ThreadHelper.ThrowIfNotOnUIThread();
    string message = "Hello World!";
    string title = "Command";

    // Show a message box to prove we were here
    VsShellUtilities.ShowMessageBox(
        this.ServiceProvider,
        message,
        title,
        OLEMSGICON.OLEMSGICON_INFO,
        OLEMSGBUTTON.OLEMSGBUTTON_OK,
        OLEMSGDEFBUTTON.OLEMSGDEFBUTTON_FIRST);
}
...
```

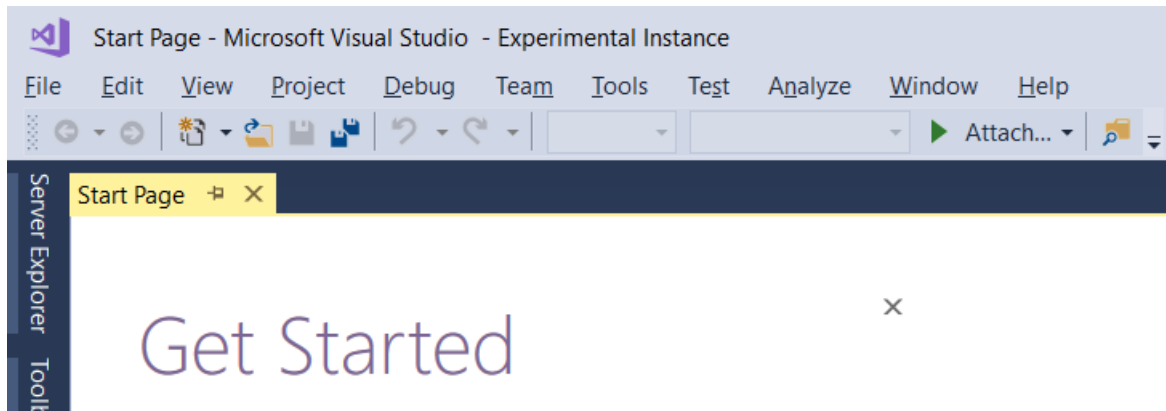
Asegúrese de guardar los cambios en cada archivo.

Ejecútelo.

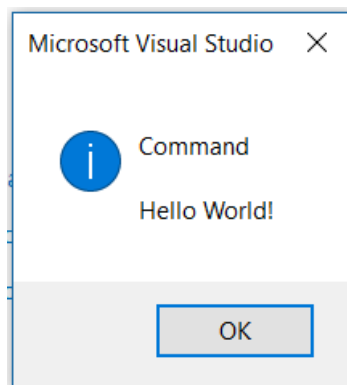
Ahora puede ejecutar el código fuente en la instancia experimental de Visual Studio.

Paso 1. Presione **F5** para ejecutar el comando **Iniciar depuración**. Este comando compila el proyecto e inicia el depurador, lanzando una nueva instancia de Visual Studio denominada **Instancia experimental**.

Verá las palabras **Instancia experimental** en la barra de título de Visual Studio.



Paso 2. En el menú **Herramientas** de la **Instancia Experimental**, haga clic en **Say Hello World!**.



Debería ver la salida de su nuevo comando personalizado, en este caso el cuadro de diálogo en el centro de la pantalla que le da el **Hello World!** de la directiva).

Pasos siguientes

Ahora que conoce los conceptos básicos de trabajar con la extensibilidad de Visual Studio, aquí es donde puede obtener más información:

- [Comience a desarrollar extensiones](#) de Visual Studio- Ejemplos, tutoriales. y la publicación de su extensión
- [Novedades del SDK de Visual Studio 2017](#): nuevas características de extensibilidad en Visual Studio 2017
- [Novedades del SDK de Visual Studio 2019](#): nuevas características de extensibilidad en Visual Studio 2019
- Dentro del SDK de [Visual Studio](#): aprenda los detalles de la extensibilidad de Visual Studio

Novedades del SDK de Visual Studio 2019

06/04/2020 • 3 minutes to read • [Edit Online](#)

El SDK de Visual Studio tiene las siguientes características nuevas y actualizadas para Visual Studio 2019.

Advertencia de extensiones cargadas automáticamente sincrónicamente

Los usuarios ahora verán una advertencia si alguna de sus extensiones instaladas se carga automáticamente al iniciarse. Puede obtener más información sobre la advertencia en [Extensiones cargadas automáticamente sincrónicamente](#).

SDK único y unificado de Visual Studio

Ahora puede obtener todos los activos del SDK de Visual Studio a través de un único paquete NuGet [Microsoft.VisualStudio.SDK](#).

Mejoras en el registro del editor

Desde su creación, Visual Studio ha admitido el registro de editor personalizado donde un editor puede declarar su afinidad por extensiones específicas (por ejemplo, .xaml y .rc), o que es adecuado para cualquier extensión (*). A partir de Visual Studio 2019 versión 16.1, ampliamos la compatibilidad con el registro del editor.

Nombres de archivo

Además de, o en lugar de, registrar la compatibilidad con una extensión de archivo `ProvideEditorFilename` determinada, un editor puede registrar que admite nombres de archivo específicos aplicando el nuevo atributo al paquete del editor.

Por ejemplo, un editor que admita `ProvideEditorExtension` todos los archivos .json aplicaría este atributo a su paquete:

```
[ProvideEditorExtension(typeof(MyEditor), ".json", MyEditor.Priority)]
```

A partir de 16.1, si MyEditor solo admite un par de `ProvideEditorFilename` archivos .json conocidos, en su lugar puede aplicar estos atributos a su paquete:

```
[ProvideEditorFilename(typeof(MyEditor), "particular.json", MyEditor.Priority)]  
[ProvideEditorFilename(typeof(MyEditor), "special.json", MyEditor.Priority)]
```

UIContexts

Un editor puede registrar uno o varios UIContexts que representan cuando está habilitado. UIContexts se registran aplicando una o `ProvideEditorUIContextAttribute` varias instancias del paquete que registra el editor.

Si un editor ha registrado UIContexts:

- Si al menos uno de sus UIContexts registrados está activo cuando se abre un archivo con la extensión dada, el editor se incluye en la búsqueda del editor.
- Si ninguno de los UIContexts registrados está activo, el editor no se incluye en la búsqueda del editor.

Si un editor no registra ningún `UIContexts`, siempre se incluye en la búsqueda del editor para esa extensión.

Por ejemplo, si un editor solo está disponible cuando un proyecto de C- está abierto, puede declarar esta afinidad aplicando un `ProvideEditorUIContext` atributo:

```
[ProvideEditorUIContext(typeof(MyEditor), KnownUIContexts.CSharpProjectContext)]
```


Extensiones cargadas automáticamente y sincrónicamente

06/04/2020 • 2 minutes to read • [Edit Online](#)

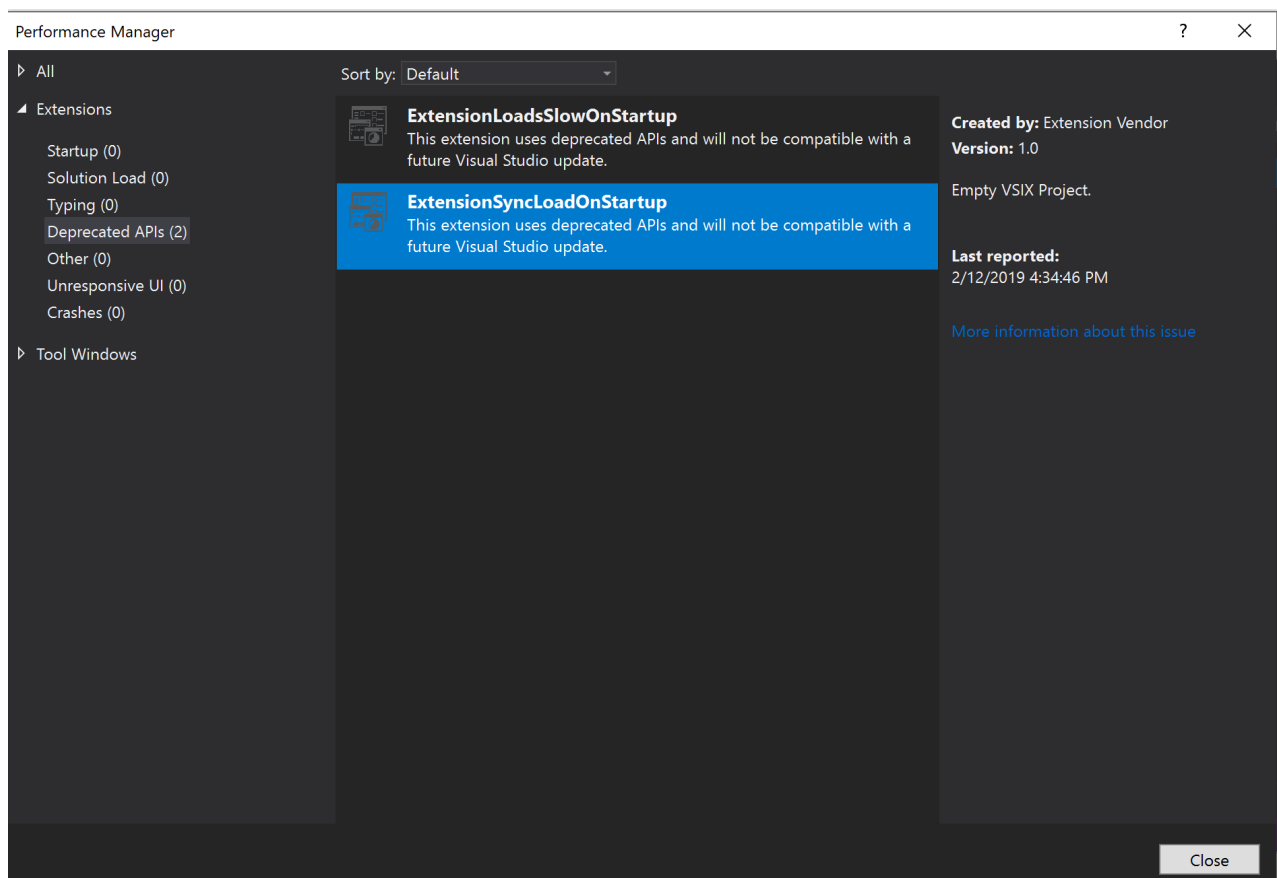
Las extensiones cargadas automáticamente sincrónicamente tienen un impacto negativo en el rendimiento de Visual Studio y se deben convertir para usar la carga automática asincrónica en su lugar. De forma predeterminada, Visual Studio 2019 bloquea los paquetes cargados automáticamente sincrónicamente desde cualquier extensión y notifica al usuario.

Visual Studio did not load one or more extensions that were using deprecated APIs. [Allow synchronous autoload](#) [Manage performance](#) [Don't show this message for current extensions](#)

Puede:

- Haga clic en **Permitir carga automática sincrónica** para permitir que las extensiones se carguen automáticamente. Para cambiar esta configuración en las opciones de Visual Studio, haga clic en Entorno, haga clic en Extensionesy, a continuación, active la casilla "Permitir la carga automática sincrónica de extensiones".
- Haga clic en **Administrar rendimiento** para abrir el cuadro de diálogo Administrador de [rendimiento](#) que muestra problemas de rendimiento con extensiones y ventanas de herramientas.
- Haga clic en **No mostrar este mensaje para las extensiones actuales** para descartar la notificación y evitar futuras notificaciones de extensiones instaladas existentes. Si agrega una nueva extensión que se carga automáticamente, esta notificación se mostrará de nuevo. Seguirá recibiendo notificaciones sobre otras características de Visual Studio.

Diálogo de Performance Manager



Performance Manager

Sort by: Default

Extensions

- Startup (0)
- Solution Load (0)
- Typing (0)
- Deprecated APIs (2)
- Other (0)
- Unresponsive UI (0)
- Crashes (0)

Tool Windows

ExtensionLoadsSlowOnStartup
This extension uses deprecated APIs and will not be compatible with a future Visual Studio update.

ExtensionSyncLoadOnStartup
This extension uses deprecated APIs and will not be compatible with a future Visual Studio update.

Created by: Extension Vendor
Version: 1.0
Empty VSIX Project.

Last reported:
2/12/2019 4:34:46 PM

[More information about this issue](#)

Close

Todas las extensiones que cargaron sincrónicamente los paquetes de las sesiones de usuario aparecen en la pestaña **API en desuso**.

- Haga clic en **Más información sobre este problema** para recopilar más información sobre las API en desuso.
- Póngase en contacto con sus proveedores de extensiones para conocer el progreso de la migración.

Especificar la configuración de carga automática sincrónica mediante la directiva de grupo

Los administradores pueden habilitar una directiva de grupo para permitir la carga automática sincrónica. Para ello, establezca una directiva basada en el Registro en la siguiente clave:

HKEY_LOCAL_MACHINE\SOFTWARE\Directivas\Microsoft\VisualStudio\SynchronousAutoload

Entrada: **Se permite**

Valor = (DWORD)

- **0** es autocarga sincrónica no permitida
- **1** es autocarga síncrona permitida

Autores de extensiones

Los autores de extensiones pueden encontrar instrucciones para migrar paquetes a la carga automática asincrónica en [Migrate to AsyncPackage](#).

Vea también

Para obtener más información acerca de la configuración de carga automática sincrónica en Visual Studio 2019, vea la página Comportamiento de [carga automática sincrónica](#).

Qué's Nuevo en el SDK de Visual Studio 2017

06/04/2020 • 2 minutes to read • [Edit Online](#)

El SDK de Visual Studio tiene las siguientes características nuevas y actualizadas para Visual Studio 2017.

Formato VSIX v3

Para admitir la nueva instalación ligera de Visual Studio 2017, el formato de manifiesto de extensión VSIX se ha actualizado a la versión 3 (VSIX v3).

El nuevo formato es compatible con:

- Declaración explícita de los requisitos previos que debe detectar e instalar VSIXInstaller.
- Ngen ensamblados en la instalación de la extensión.
- Instalación de activos fuera de la raíz de extensión habitual.

Para obtener más información sobre estos cambios, consulte los siguientes temas:

- [Cambios en la extensibilidad de Visual Studio 2017](#)
- [Compatibilidad con Ngen en VSIX v3](#)
- [Instalar fuera de la carpeta de extensiones](#)
- [Preguntas frecuentes sobre la extensibilidad de Visual Studio 2017](#)

Migrar proyecto de extensibilidad a Visual Studio 2017

Para obtener información sobre cómo actualizar los proyectos de extensibilidad y sus manifiestos VSIX a Visual Studio 2017, vea [Cómo: migrar proyectos de extensibilidad a Visual Studio 2017](#).

Plantillas personalizadas de proyectos y elementos

A partir de Visual Studio 2017, ya no se realizará el análisis de plantillas de proyecto y elemento personalizadas. En su lugar, la extensión debe proporcionar archivos de manifiesto de plantilla que describan la ubicación de instalación de estas plantillas. Puede usar Visual Studio 2017 para actualizar las extensiones VSIX. Si implementa la extensión mediante un MSI, debe generar los archivos de manifiesto de plantilla a mano. Para obtener más información, vea [Actualizar plantillas de proyecto y elemento personalizadas para Visual Studio 2017](#). El esquema de manifiesto de plantilla se documenta en la referencia de esquema de manifiesto de plantilla de [Visual Studio](#).

Directrices de rendimiento de la extensión actualizadas

Hay un nuevo artículo [Cómo: diagnosticar](#) el rendimiento de la extensión en [Administrar VSPackages](#) para mostrar cómo detectar y analizar el impacto de la extensión en los tiempos de carga de la solución y el inicio de Visual Studio.

Cambios en la extensibilidad de Visual Studio 2017

06/04/2020 • 11 minutes to read • [Edit Online](#)

Visual Studio 2017 proporciona una experiencia de instalación de [Visual Studio más rápida y ligera](#) que reduce el impacto de Visual Studio en los sistemas de usuario al tiempo que ofrece a los usuarios una mayor variedad de opciones sobre las cargas de trabajo y las características que se instalan. Para admitir estas mejoras, hemos realizado cambios en el modelo de extensibilidad, incluidos algunos cambios importantes. Este artículo describe los detalles técnicos de estos cambios y lo que se puede hacer para abordarlos.

NOTE

Parte de la información son detalles de implementación a un momento dado y se pueden cambiar más adelante.

Cambios que afectan al formato y la instalación de VSIX

Visual Studio 2017 introdujo el formato VSIX v3 (versión 3) para admitir la experiencia de instalación ligera.

Los cambios en el formato VSIX incluyen:

- Declaración de los requisitos previos de configuración. Para cumplir con la promesa de un Visual Studio ligero y de instalación rápida, el instalador ahora ofrece más opciones de configuración a los usuarios. Como resultado, para asegurarse de que las características y componentes requeridos por una extensión están instalados, las extensiones deben declarar sus dependencias.
 - El instalador de Visual Studio 2017 ofrece automáticamente adquirir e instalar los componentes necesarios para el usuario como parte de la instalación de la extensión.
 - También se advierte a los usuarios al intentar instalar una extensión que no se creó con el nuevo formato VSIX v3, incluso si se han marcado en su manifiesto como la versión de destino 15.0.
- Capacidades mejoradas para el formato VSIX. Para ofrecer una instalación de [bajo impacto](#) de Visual Studio que también admite instalaciones en paralelo, ya no guardamos la mayoría de los datos de configuración en el registro del sistema y hemos movido ensamblados específicos de Visual Studio fuera de la GAC. También aumentamos las capacidades del formato VSIX y el motor de instalación VSIX, lo que le permite utilizarlo en lugar de un MSI o EXE para instalar sus extensiones para algunos tipos de instalación.

Las nuevas capacidades incluyen:

- Registro en la instancia de Visual Studio especificada.
- Instalación fuera de la [carpeta de extensiones](#).
- Detección de arquitectura de procesador.
- Dependencia de paquetes de idioma separados por el idioma.
- Instalación con [soporte NGEN](#).

Crear una extensión para Visual Studio 2017

Las herramientas de diseñador para la creación del nuevo formato de manifiesto de VSIX v3 están disponibles en Visual Studio 2017. Consulte el documento adjunto [Cómo: migrar proyectos de extensibilidad a Visual Studio 2017](#) para obtener más información sobre el uso de las herramientas del diseñador o la realización de actualizaciones manuales del proyecto y el manifiesto para desarrollar extensiones VSIX v3.

Cambiar: ruta de acceso de datos de usuario de Visual Studio

Anteriormente, solo podía existir una instalación de cada versión principal de Visual Studio en cada equipo. Para admitir instalaciones en paralelo de Visual Studio 2017, pueden existir varias rutas de acceso de datos de usuario para Visual Studio en el equipo del usuario.

El código que se ejecuta dentro del proceso de Visual Studio debe actualizarse para usar el Administrador de configuración de Visual Studio. El código que se ejecuta fuera del proceso de Visual Studio puede encontrar la ruta de acceso del usuario de una instalación específica de Visual Studio [siguiendo las instrucciones aquí](#).

Cambiar: Caché global de ensamblados (GAC)

La mayoría de los ensamblados principales de Visual Studio ya no están instalados en la GAC. Se realizaron los siguientes cambios para que el código que se ejecuta en el proceso de Visual Studio pueda seguir encontrando los ensamblados necesarios en tiempo de ejecución.

NOTE

[INSTALLDIR] a continuación hace referencia al directorio raíz de instalación de Visual Studio. *VSIXInstaller.exe* rellenará automáticamente esto, pero para escribir código de implementación personalizado, lea [localizar Visual Studio](#).

- Ensamblados que solo se instalaron en la GAC:

Estos ensamblados ahora se instalan en *[INSTALLDIR], Common7,*IDE, *[INSTALLDIR], Common7, IDE, PublicAssemblies, o [INSTALLDIR], Common7, IDE, PrivateAssemblies*. Estas carpetas forman parte de las rutas de acceso de sondeo del proceso de Visual Studio.

- Ensamblados que se instalaron en una ruta de acceso sin sondeo y en la GAC:
 - La copia de la GAC se eliminó de la configuración.
 - Se ha agregado un archivo *.pkgdef* para especificar una entrada base de código para el ensamblado.

Por ejemplo:

```
[${RootKey}\RuntimeConfiguration\dependentAssembly\codeBase\{UniqueGUID}]
"name"="AssemblyName" "codeBase"="$PackageFolder$\AssemblyName.dll"
"publicKeyToken"="Public Key Token"
"culture"="neutral"
"version"=15.0.0.0
```

En tiempo de ejecución, el subsistema pkgdef de Visual Studio combina estas entradas en el archivo de configuración en tiempo `<codeBase>` de ejecución del proceso de Visual Studio (en *[VSAPPDATA]-devenv.exe.config*) como elementos. Esta es la forma recomendada de permitir que el proceso de Visual Studio encuentre el ensamblado, ya que evita buscar en las rutas de acceso de sondeo.

Reaccionando a este cambio de ruptura

- Si la extensión se ejecuta en el proceso de Visual Studio:
 - El código podrá encontrar los ensamblados principales de Visual Studio.
 - Considere la posibilidad de usar un archivo *.pkgdef* para especificar una ruta de acceso a los ensamblados si es necesario.
- Si la extensión se ejecuta fuera del proceso de Visual Studio:

Considere la posibilidad de buscar los ensamblados principales de Visual Studio en *[INSTALLDIR], Common7,*IDE, *[INSTALLDIR], Common7, IDE, PublicAssemblies, o [INSTALLDIR], Common7, IDE,*

Cambio: Reducir el impacto en el registro

Registro COM global

- Anteriormente, Visual Studio instalaba muchas claves del Registro en los subárboles HKEY_CLASSES_ROOT y HKEY_LOCAL_MACHINE para admitir el registro COM nativo. Para eliminar este impacto, Visual Studio ahora usa [activación sin registro para componentes COM](#).
- Como resultado, la mayoría de los archivos TLB / OLB / DLL en %ProgramFiles(x86)%-Archivos comunes-Microsoft Shared-MSEnv ya no se instalan de forma predeterminada por Visual Studio. Estos archivos ahora se instalan en [INSTALLDIR] con los manifiestos COM sin registro correspondientes utilizados por el proceso de host de Visual Studio.
- Como resultado, el código externo que se basa en el registro COM global para las interfaces COM de Visual Studio ya no encontrará estos registros. El código que se ejecuta dentro del proceso de Visual Studio no verá una diferencia.

Registro de Visual Studio

- Anteriormente, Visual Studio instalaba muchas claves del Registro en los **subárboles** HKEY_LOCAL_MACHINE y HKEY_CURRENT_USER del sistema en una clave específica de Visual Studio:
 - HKLM-Software-Microsoft-VisualStudio{Version}: claves del Registro creadas por los instaladores MSI y extensiones por equipo.
 - HKCU-Software-Microsoft-VisualStudio{Version}: claves del Registro creadas por Visual Studio para almacenar la configuración específica del usuario.
 - HKCU-Software-Microsoft-VisualStudio{Version-}_Config: Una copia de la clave HKLM de Visual Studio anterior, además de las claves del Registro combinadas de archivos .pkgdef por extensiones.
- Para reducir el impacto en el registro, Visual Studio ahora usa la función [RegLoadAppKey](#) para almacenar las claves del Registro en un archivo binario privado en [VSAPPDATA]-privateregistry.bin. Solo un número muy pequeño de claves específicas de Visual Studio permanecen en el registro del sistema.
- El código existente que se ejecuta dentro del proceso de Visual Studio no se ve afectado. Visual Studio redirigirá todas las operaciones del Registro bajo la clave específica de HKCU Visual Studio al registro privado. La lectura y escritura en otras ubicaciones del registro seguirá utilizando el registro del sistema.
- El código externo tendrá que cargar y leer desde este archivo para las entradas del Registro de Visual Studio.

Reaccionar a este cambio de ruptura

- El código externo también se debe convertir para usar la activación sin registro para los componentes COM.
- Los componentes externos pueden encontrar la ubicación de Visual Studio [siguiendo las instrucciones aquí](#).
- Se recomienda que los componentes externos usen el Administrador de [configuración externa](#) en lugar de leer y escribir directamente en las claves del Registro de Visual Studio.
- Compruebe si los componentes que utiliza la extensión pueden haber implementado otra técnica para el registro. Por ejemplo, las extensiones del depurador pueden aprovechar el nuevo registro COM de [archivo JSON msvsmon](#).

Compatibilidad con Ngen en VSIX v3

06/04/2020 • 3 minutes to read • [Edit Online](#)

Con Visual Studio 2017 y el nuevo formato de manifiesto de extensión VSIX v3 (versión 3), los desarrolladores de extensiones ahora pueden "ngen" sus ensamblados en el momento de la instalación.

A continuación se muestra un extracto de MSDN que explica lo que hace "ngen":

El generador de imágenes nativas (*Ngen.exe*) es una herramienta que mejora el rendimiento de las aplicaciones administradas. *Ngen.exe* crea imágenes nativas, que son archivos que contienen código de máquina específico del procesador compilado, y las instala en la caché de imágenes nativas del equipo local. El runtime puede usar imágenes nativas de la memoria caché en lugar de usar el compilador Just-In-Time (JIT) para compilar el ensamblado original.

de [Ngen.exe \(Generador de imágenes nativas\)](#)

Para "ngen" un ensamblaje, el VSIX debe instalarse "por instancia por máquina". Esto se puede habilitar marcando la casilla de `extension.vsixmanifest` verificación "todos los usuarios" en el diseñador:

- ☒ This VSIX is installed for all users (requires elevation on install)
- ☐ This VSIX is installed by Windows Installer

Cómo habilitar Ngen

Para habilitar ngen para un ensamblado, puede usar la ventana **Propiedades** en Visual Studio.

Hay 4 propiedades que se pueden establecer:

1. **Ngen** (Boolean): si es true, el instalador de Visual Studio "ngen" el ensamblado.
2. **Aplicación Ngen** (cadena): Ngen proporciona la oportunidad de utilizar el archivo *app.config* de una aplicación para resolver las dependencias del ensamblado. Este valor debe establecerse en una aplicación cuyo *app.config* desea usar (en relación con el directorio de instalación de Visual Studio).
3. **Ngen Architecture** (enum): la arquitectura para compilar de forma nativa el ensamblado. Las opciones son: a. No especificado b. X86 c. X64 d. Todas
4. **Ngen Priority** (entero entre 1 y 3) - El nivel de prioridad de Ngen se documenta en los niveles de [prioridad Ngen.exe](#).

Aquí hay un vistazo a la ventana **Propiedades** en acción:

Properties

ClassLibrary1 Reference Properties

Copy Local	True
Culture	
Description	
Embed Interop Types	False
File Type	Assembly
Identity	ClassLibrary1
Path	C:\Users\alexey\documents\visual studio 15\
Resolved	True
Runtime Version	
Strong Name	False
Version	0.0.0.0
Ngen	
Ngen	True
Ngen Application	vsn.exe
Ngen Architecture	X86
Ngen Priority	2
VSIX	
Install Root	MSBuild
Output Groups Included in VSIX	
Output Groups Included in VSIX (Local C	
Template Type	
VSIX Sub Path	MyCompany\MySDK\1.0

Ngen Priority

Specifies the ngen priority level.

Esto agregará metadatos a la referencia del proyecto dentro del archivo *.csproj* del proyecto VSIX:

```
<ProjectReference Include="..\ClassLibrary1\ClassLibrary1.csproj">
  <Project>{69a979f1-eba2-43e7-9346-0e56e803508b}</Project>
  <Name>ClassLibrary1</Name>
  <Ngen>True</Ngen>
  <NgenApplication>vsn.exe</NgenApplication>
  <NgenArchitecture>X86</NgenArchitecture>
  <NgenPriority>2</NgenPriority>
</ProjectReference>
```

NOTE

Puede editar el archivo *.csproj* directamente, si lo prefiere.

Información adicional

Los cambios del diseñador de propiedades se aplican a algo más que referencias de proyecto; También puede

establecer los metadatos de Ngen para los elementos dentro del proyecto (utilizando los mismos métodos descritos anteriormente) siempre que los elementos sean ensamblados .NET.

Instalar fuera de la carpeta de extensiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

A partir de Visual Studio 2017 y VSIX v3 (versión 3), los activos de extensión se pueden instalar fuera de la carpeta de extensiones. Actualmente, las siguientes ubicaciones están habilitadas como ubicaciones de instalación válidas (donde [INSTALLDIR] se asigna al directorio de instalación de la instancia de Visual Studio):

- [INSTALLDIR]-MSBuild
- [INSTALLDIR]-Xml-Schemas
- [INSTALLDIR]-Common7-IDE-PublicAssemblies
- [INSTALLDIR]-Licencias
- [INSTALLDIR]-Common7-IDE-ReferenceAssemblies
- [INSTALLDIR]-Common7-IDE-RemoteDebugger
- [INSTALLDIR]-Common7-IDE-VC-VCTargets (solo se admite para Visual Studio 2017; en desuso para Visual Studio 2019 y versiones posteriores)

NOTE

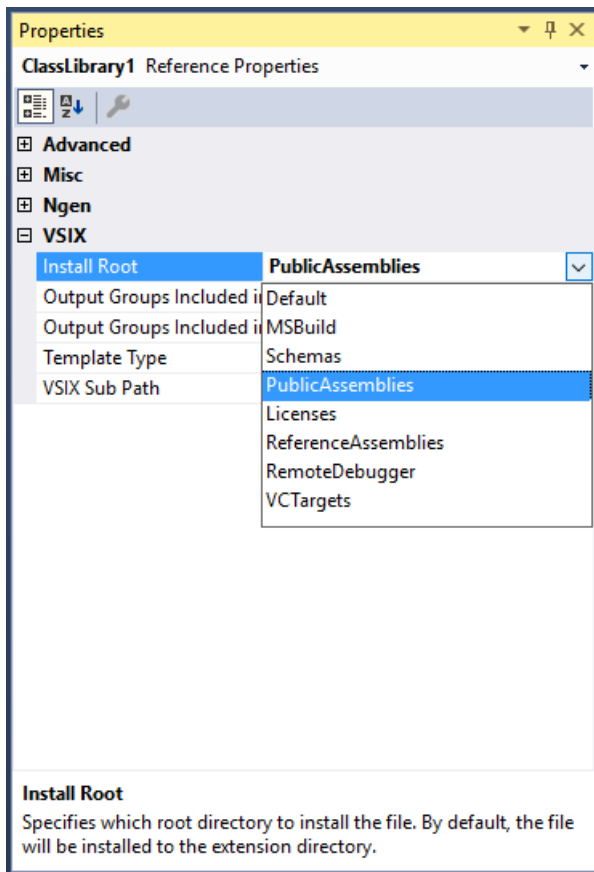
El formato VSIX no permite instalar fuera de la estructura de carpetas de instalación de Visual Studio.

Para admitir la instalación en estos directorios, el VSIX debe instalarse "por instancia por máquina". Esto se puede habilitar marcando la casilla de verificación "todos los usuarios" en el diseñador extension.vsixmanifest:

- ☒ This VSIX is installed for all users (requires elevation on install)
- ☐ This VSIX is installed by Windows Installer

Cómo configurar InstallRoot

Para establecer los directorios de instalación, puede usar la ventana **Propiedades** en Visual Studio. Por ejemplo, puede `InstallRoot` establecer la propiedad de una referencia de proyecto a una de las ubicaciones anteriores:



Esto agregará algunos metadatos `ProjectReference` a la propiedad correspondiente dentro del archivo .csproj del proyecto VSIX:

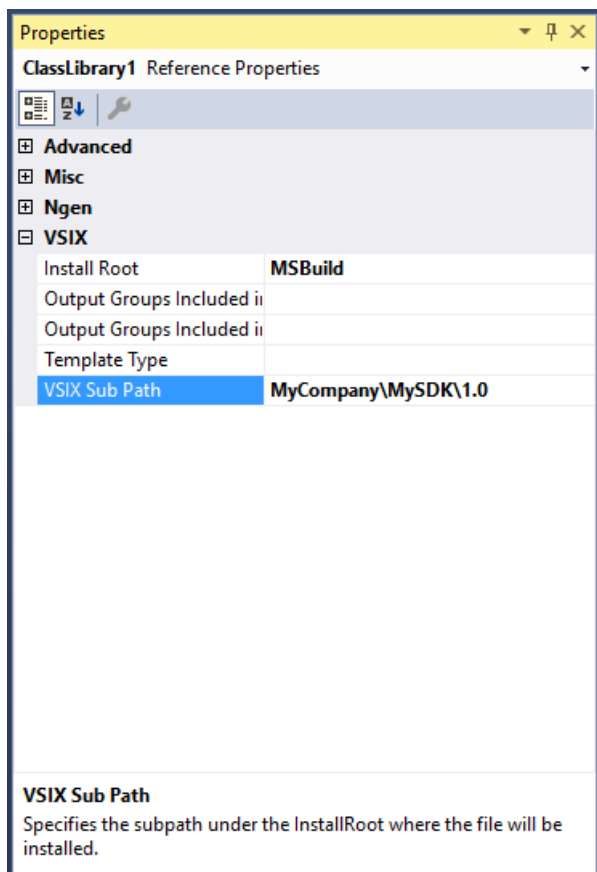
```
<ProjectReference Include="..\ClassLibrary1\ClassLibrary1.csproj">
  <Project>{69a979f1-eba2-43e7-9346-0e56e803508b}</Project>
  <Name>ClassLibrary1</Name>
  <InstallRoot>PublicAssemblies</InstallRoot>
</ProjectReference>
```

NOTE

Puede editar el archivo .csproj directamente, si lo prefiere.

Cómo establecer una subruta en InstallRoot

Si desea instalar en una subruta `InstallRoot` debajo de la , `VsixSubPath` puede hacerlo `InstallRoot` estableciendo la propiedad al igual que la propiedad. Por ejemplo, supongamos que queremos que la salida de nuestra referencia de proyecto se instale en '[INSTALLDIR], MSBuild, MyCompany, MySDK, 1.0'. Podemos hacerlo fácilmente con el diseñador de propiedades:



Los cambios .csproj correspondientes tendrán este aspecto:

```
<ProjectReference Include="..\ClassLibrary1\ClassLibrary1.csproj">
  <Project>{69a979f1-eba2-43e7-9346-0e56e803508b}</Project>
  <Name>ClassLibrary1</Name>
  <InstallRoot>MSBuild</InstallRoot>
  <VSIXSubPath>MyCompany\MySDK\1.0</VSIXSubPath>
</ProjectReference>
```

Información adicional

Los cambios del diseñador de propiedades se aplican a algo más que referencias de proyecto; también puede `InstallRoot` establecer los metadatos de los elementos dentro del proyecto (utilizando los mismos métodos descritos anteriormente).

Migrar el registro de la clase COM de depurador de 64 bits

02/12/2019 • 2 minutes to read • [Edit Online](#)

Para extensiones del depurador que COM de registrar las clases en HKEY_CLASSES_ROOT mediante regasm, regsvr32, o escribir directamente en el registro y se cargan en *msvsmon.exe* (el depurador remoto), ahora es posible que proporcionarle esta información registro de msvsmon sin necesidad de escribir en HKEY_CLASSES_ROOT. Esto afecta a los evaluadores de expresiones del depurador .NET heredados o motores de depuración que están configurados para cargar en el *msvsmon.exe* proceso.

msvsmon-comclass-def

Para usar esta técnica, agregue un *.msvsmon-comclass-def.json* archivo junto a *msvsmon* (*InstallDir: \Common7\IDE\Remote Debugger\x64**).

Este es un ejemplo de archivo msvsmon-comclass-def que registra uno administrado y una clase nativa:

FileName: *MyCompany.MyExample.msvsmon-comclass-def.json*

```
{
  "managedCOMClasses": [
    {
      "clsid": "{C9F48B25-36ED-4B22-8210-98BC5BE4D1E7}",
      "assemblyName": "MyCompany.MyAssembly",
      "className": "MyCompany.MyNamespace.MyClass"
    }
  ],
  "nativeCOMClasses": [
    {
      "clsid": "{42A476E9-8FA7-44D5-ADFE-216AD371EEC9}",
      "dllPath": "MyExample.dll"
    }
  ]
}
```

Preguntas más frecuentes sobre la extensibilidad de Visual Studio 2017

14/01/2020 • 11 minutes to read • [Edit Online](#)

Estas son algunas de las preguntas más frecuentes sobre la extensibilidad de Visual Studio 2017.

¿Cuál es el caso de compatibilidad con versiones anteriores de las extensiones?

El nuevo formato VSIX V3 es compatible con las versiones anteriores de VSIX V2, por lo que aún podrá tener un VSIX único con un solo identificador de VSIX que admita Visual Studio 2012 y versiones posteriores. El nuevo formato VSIX V3 no es compatible con Visual 2010 y versiones anteriores. Para admitir Visual Studio 2010 en adelante, tendrá que crear una extensión independiente (con un identificador VSIX independiente). Como Visual Studio 2010 es ahora un pequeño porcentaje de la base de usuarios, se recomienda usar el identificador de VSIX existente para la extensión que admite Visual Studio 2012 o posterior y asignar un nuevo identificador de VSIX a la versión de Visual Studio 2010.

¿Por qué es necesario declarar requisitos previos con Visual Studio 2017?

Visual Studio 2017 permite una instalación más rápida y más ligera de Visual Studio que ofrece a los usuarios un mayor control sobre las cargas de trabajo y los componentes que se instalan. Para asegurarse de que los componentes necesarios para la extensión están instalados, con VSIX V3 y Visual Studio 2017, debe declarar los componentes de los que depende su extensión como requisitos previos. Si alguno de los requisitos previos necesarios no está disponible en el equipo del usuario, el usuario no podrá ejecutar la extensión. Sin embargo, si Visual Studio detecta que un usuario está intentando instalar una extensión que requiere componentes que faltan, pedirá al usuario que instale los componentes necesarios. Si tiene componentes opcionales, no es necesario que los Enumere como requisitos previos, siempre que realice la detección de características adecuada en tiempo de ejecución. Todas las extensiones deben especificar el componente editor principal como dependencia

```
Microsoft.VisualStudio.Component.CoreEditor
```

Cuando se indica un requisito previo, ¿qué nivel de granularidad significa?

Los requisitos previos se declaran en el nivel de componente, es decir, las opciones individuales que se pueden seleccionar en la interfaz de usuario. No se declaran requisitos previos en cargas de trabajo o archivos dll individuales.

¿Dónde puedo encontrar una lista de identificadores de componentes para poder declarar dependencias?

Para encontrar la lista de identificadores de componente, consulte [identificadores de componente y carga de trabajo de Visual Studio 2017](#). Si no está seguro de qué componente contiene un archivo binario específico, puede descargar la [hoja de cálculo de asignación binaria de componentes](#) >. Para obtener más información sobre el uso de la hoja de cálculo, vea la sección **Buscar identificadores de componente** en [Cómo: migrar proyectos de extensibilidad a Visual Studio 2017](#).

Mi extensión requiere Visual Studio 15,3, ¿cómo puedo aplicar ese requisito?

Si la extensión requiere una versión específica de Visual Studio 2017, por ejemplo, depende de una característica publicada en 15,3, puede especificar el número de compilación en el `admitir` de VSIX. Por ejemplo, la versión 15,3 tiene un número de compilación de "15.0.26730.3". [Aquí](#) puede ver la asignación de versiones a los números de compilación. Tenga en cuenta que el uso del número de versión ' 15,3 ' no funcionará correctamente.

Si la extensión requiere 15,3 o superior, declararía la **versión de admitir** como [15.0.26730.3, 16,0):

```
<Installation>
  <InstallationTarget Id="Microsoft.VisualStudio.Community" Version="[15.0.26730.3, 16.0)" />
</Installation>
```

El VSIXInstaller detectará versiones anteriores de Visual Studio e informará al usuario de que se requiere una actualización posterior.

Sigo recibiendo un error al intentar cargar la extensión.

Es posible que esté usando la versión anterior del manifiesto de VSIX. Si la extensión está marcada como compatible con Visual Studio 2017, pero no usa el nuevo formato de manifiesto VSIX V3, no podrá cargarla.

Utilizo mi propio instalador. ¿Puedo continuar?

Con Visual Studio 2017, la instalación mínima de Visual Studio será mucho menor que las versiones anteriores.

Hemos mejorado el formato de manifiesto de VSIX para admitir los cambios necesarios para la instalación ligera. En la medida de lo posible, se recomienda proporcionar la extensión en formato VSIX V3. Es posible que necesite mantener el instalador existente para determinados escenarios, como si instala componentes fuera de Visual Studio.

Por ejemplo, si necesita MSBuild, debe especificarlo como requisito previo en el manifiesto de VSIX. Al instalar VSIX, el instalador se asegurará de que MSBuild esté disponible.

Después, el instalador invoca el VSIXInstaller para instalar un VSIX con los componentes de Visual Studio 2017.

¿Puede obtener más información sobre la migración?

Puede obtener más información en [Cómo: migrar proyectos de extensibilidad a Visual Studio 2017](#). Además, hay una excelente entrada de blog, [cambios en el programa de instalación de Visual Studio](#).

¿Cómo el registro de paquetes?

El registro de paquetes se realiza de la misma manera que en el pasado con las entradas del registro. El único cambio es que el registro se está desplazando ahora y debe declarar las entradas del registro desde un archivo `.pkgdef`.

Si utiliza el `PackageRegistrationAttribute`, esto debe ocurrir automáticamente.

¿Se necesita una nueva entrada de la galería para la versión 2017 de la extensión de Visual Studio?

No, no necesitará una nueva entrada en la galería de Visual Studio para el VSIX actualizado, siempre que los cambios sean totalmente compatibles con todas las versiones admitidas. Si la extensión no es compatible con versiones anteriores, se recomienda dividir la extensión en varios VSIXs (incluso si es temporal), cada uno con su

propio identificador de VSIX y su propia entrada en la galería.

¿Qué debo hacer con mi extensión que actualmente es compatible con Visual Studio 2010 y versiones posteriores?

Agregue compatibilidad con Visual Studio 2017 al VSIX actual y mantenga la compatibilidad con Visual Studio 2012 y versiones posteriores. Además, cree un nuevo VSIX (con el nuevo GUID y la nueva entrada de la galería) compatible con Visual Studio 2010.

¿Puedo compilar VSIX V3 con Visual Studio 2015?

Sí. Hay un paquete NuGet que tiene las herramientas y las tareas necesarias para compilar los manifiestos de formato VSIX V3 en Visual Studio 2015. Agregue una referencia al paquete de NuGet

`Microsoft.VisualStudio.Sdk.BuildTasks.14.0` a su proyecto de extensibilidad. También necesitará agregar el elemento `VsixType` que especifica `v3` al archivo de proyecto:

```
<VsixType>v3</VsixType>
```

¿Puedo ejecutar VSIXInstaller en modo silencioso?

Tendrá que pasar parámetros adicionales a VSIXInstaller ahora que hay varias instancias de Visual Studio 2017 instaladas.

```
vsixinstaller.exe /q /appidinstallpath:"c:\program files (x86)\Microsoft Visual  
Studio\2017\Enterprise\Common7\IDE\devenv.exe" /appidname:"Visual Studio" /logfile:<path to log file>  
/skuName:Enterprise /skuVersion:15.0.25810.0 "KendoUI.Mvc.VSPackage.vsix"
```

¿Por qué el VSIXInstaller ahora espera a que los procesos salgan antes de instalar VSIX?

Ahora, VSIXInstaller usa el motor de instalación de Visual Studio para instalar los requisitos previos definidos por VSIX. El motor de instalación requiere que se cierren todos los procesos relacionados con VS para poder actualizar la instalación de Visual Studio.

¿Puedo instalar ahora mis recursos de extensión en cualquier ubicación con VSIX V3?

No, la propiedad `InstallRoot` de VSIX V3 no permite instalar nada fuera de la estructura de carpetas de instalación de Visual Studio. Consulte el tema [instalación fuera de la carpeta de extensiones](#) para ver las ubicaciones admitidas.

Los componentes de extensión que no forman parte de Visual Studio son probablemente singletons en el equipo y se instalan una vez para todas las instancias de Visual Studio 2017.

La recomendación para ese escenario sería crear un MSI para los componentes que no son de Visual Studio y hacer que el MSI invoque el VSIXInstaller para instalar los componentes específicos de Visual Studio.

Cómo: Migrar proyectos de extensibilidad a Visual Studio 2017

06/04/2020 • 15 minutes to read • [Edit Online](#)

En este documento se explica cómo actualizar proyectos de extensibilidad a Visual Studio 2017. Además de describir cómo actualizar los archivos de proyecto, también describe cómo actualizar de la versión 2 del manifiesto de extensión (VSIX v2) al nuevo formato de manifiesto VSIX de la versión 3 (VSIX v3).

Instale Visual Studio 2017 con las cargas de trabajo necesarias

Asegúrese de que la instalación incluye las siguientes cargas de trabajo:

- Desarrollo de escritorio de .NET
- Desarrollo de extensiones de Visual Studio

Abrir solución VSIX en Visual Studio 2017

Todos los proyectos VSIX requerirán una actualización unidireccional de la versión principal a Visual Studio 2017.

El archivo de proyecto (por ejemplo *.csproj) se actualizará:

- MinimumVisualStudioVersion - ahora establecido en 15.0
- OldToolsVersion (si existe anteriormente) - ahora establecido en 14.0

Actualice el paquete NuGet Microsoft.VSSDK.BuildTools

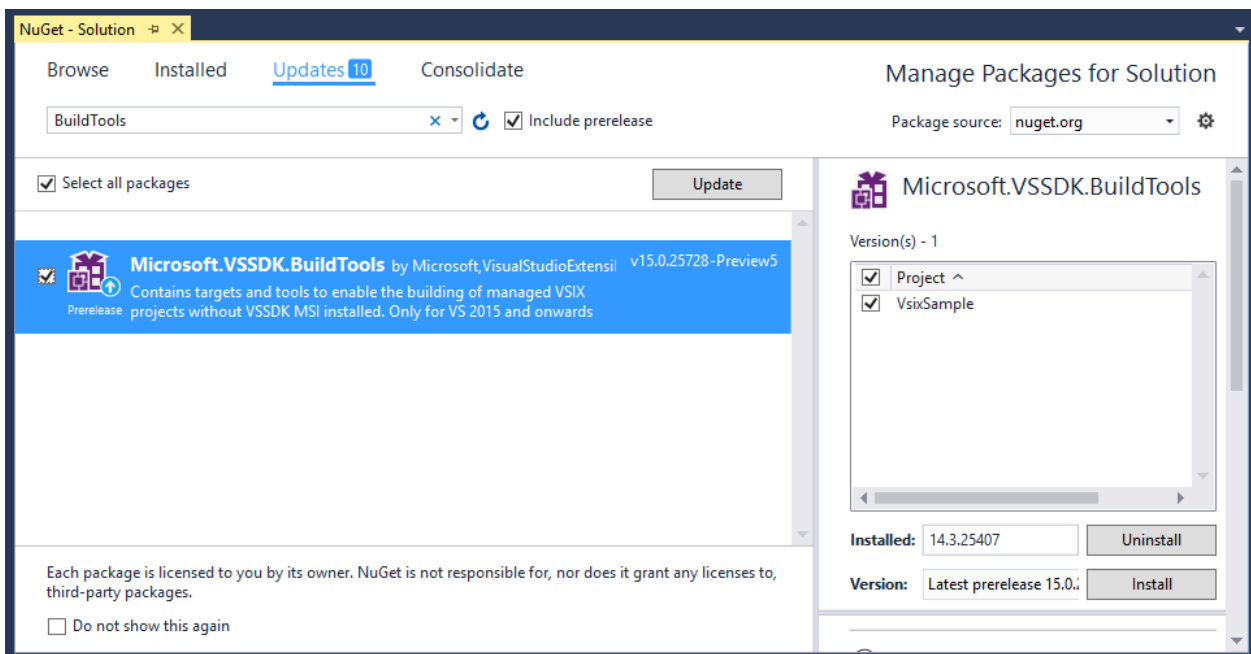
NOTE

Si la solución no hace referencia al paquete NuGet Microsoft.VSSDK.BuildTools, puede omitir este paso.

Para compilar la extensión en el nuevo formato VSIX v3 (versión 3), la solución tendrá que compilarse con las nuevas herramientas de compilación de VSSDK. Esto se instalará con Visual Studio 2017, pero la extensión VSIX v2 podría contener una referencia a una versión anterior a través de NuGet. Si es así, deberá instalar manualmente una actualización del paquete NuGet Microsoft.VSSDK.BuildTools para la solución.

Para actualizar las referencias de NuGet a Microsoft.VSSDK.BuildTools:

- Haga clic con el botón derecho en la solución y elija **Administrar paquetes NuGet para la solución**.
- Vaya a la pestaña **Actualizaciones**.
- Seleccione **Microsoft.VSSDK.BuildTools (última versión)**.
- Pulse **Actualizar**.



Realice cambios en el manifiesto de extensión VSIX

Para asegurarse de que la instalación del usuario de Visual Studio tiene todos los ensamblados necesarios para ejecutar la extensión, especifique todos los componentes o paquetes de requisitos previos en el archivo de manifiesto de extensión. Cuando un usuario intenta instalar la extensión, VSIXInstaller comprobará si todos los requisitos previos están instalados. Si faltan algunos, se le pedirá al usuario que instale los componentes que faltan como parte del proceso de instalación de la extensión.

NOTE

Como mínimo, todas las extensiones deben especificar el componente del editor principal de Visual Studio como requisito previo.

- Edite el archivo de manifiesto de extensión (normalmente denominado *source.extension.vsixmanifest*).
- Asegúrese que `InstallationTarget` incluye 15.0.
- Agregue los requisitos previos de instalación necesarios (como se muestra en el ejemplo siguiente).
 - Se recomienda especificar solo los identificadores de componente para los requisitos previos de instalación.
 - Consulte la sección al final de este documento para [obtener instrucciones sobre la identificación de id de componente](#).

Ejemplo:

```
<PackageManifest>
...
  <Prerequisites>
    <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[15.0,16.0]" />
    <Prerequisite Id="Microsoft.VisualStudio.Component.DiagnosticTools" Version="[15.0.25814.0,16.0]" />
    <Prerequisite Id="Microsoft.VisualStudio.Shell.12.0" Version="[12.0]" />
  </Prerequisites>
...
</PackageManifest>
```

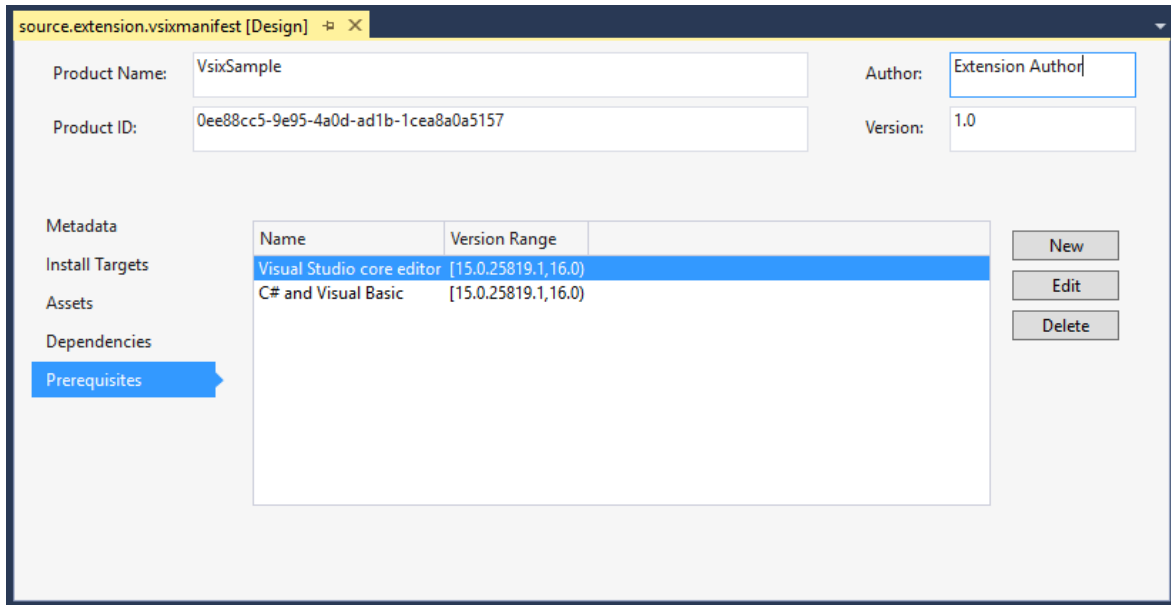
Opción: Utilice el diseñador para realizar cambios en el manifiesto de extensión VSIX

En lugar de editar directamente el XML de manifiesto, puede usar la nueva pestaña **Requisitos previos** del Diseñador de manifiestos para seleccionar los requisitos previos y el XML se actualizará automáticamente.

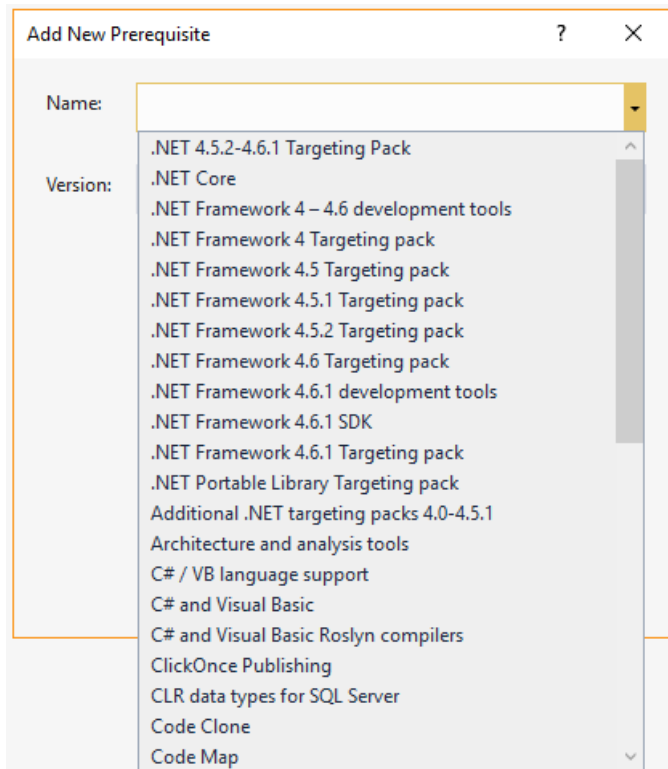
NOTE

El Diseñador de manifiestos solo le permitirá seleccionar Componentes (no Cargas de trabajo o Paquetes) que están instalados en la instancia actual de Visual Studio. Si necesita agregar un requisito previo para una carga de trabajo, paquete o componente que no está instalado actualmente, edite el manifiesto XML directamente.

- Abra el archivo *source.extension.vsixmanifest* [Design].
- Seleccione la pestaña **Requisitos previos** y pulse el botón **Nuevo**.



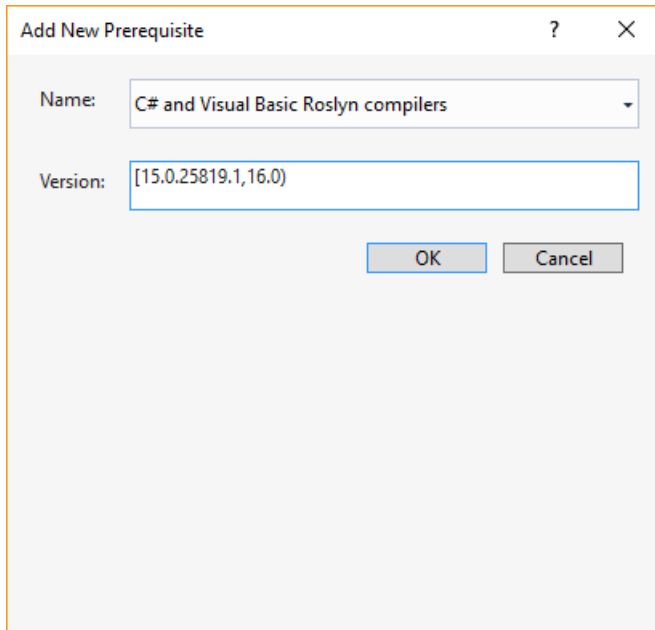
- Se abrirá la ventana **Agregar nuevo requisito previo**.



- Haga clic en el menú desplegable para **Nombre** y seleccione el requisito previo deseado.
- Actualice la versión si es necesario.

NOTE

El campo de versión se rellenará previamente con la versión del componente instalado actualmente, con un rango que abarca hasta (pero sin incluir) la siguiente versión principal del componente.



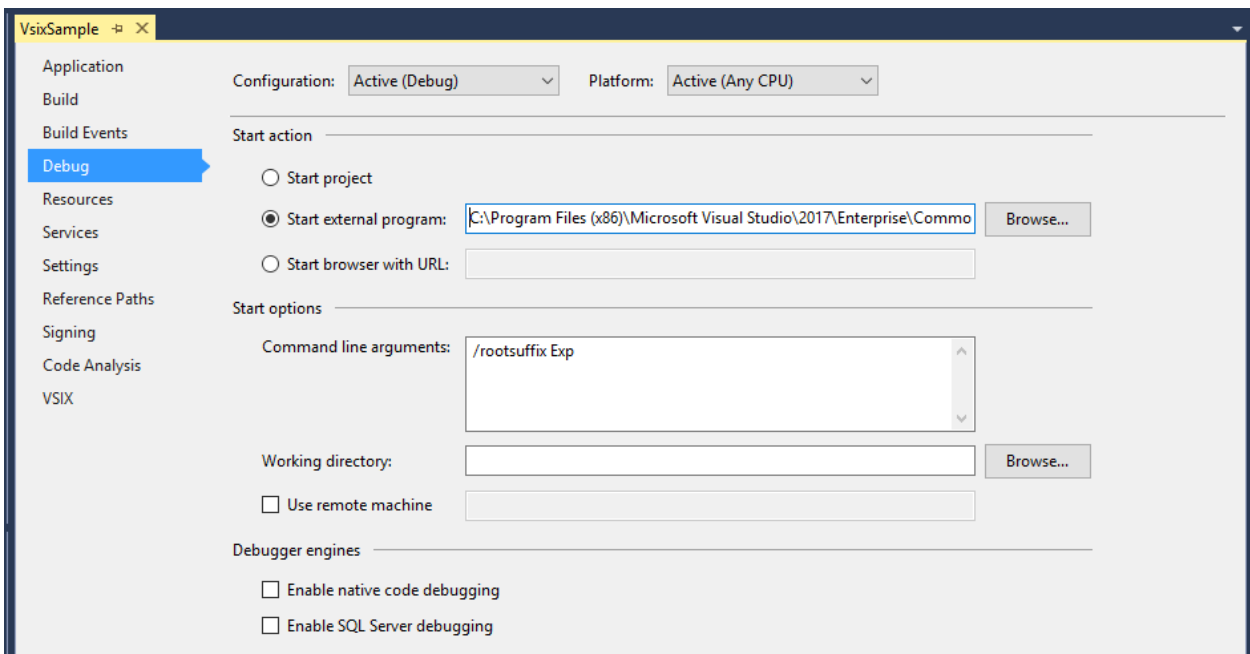
The dialog box is titled "Add New Prerequisite" and has a close button (X) in the top right corner. It contains two input fields: "Name:" with a dropdown menu showing "C# and Visual Basic Roslyn compilers", and "Version:" with a text box containing "[15.0.25819.1,16.0)". At the bottom right, there are two buttons: "OK" and "Cancel".

- Presione **Aceptar**.

Actualizar la configuración de depuración para el proyecto

Si desea depurar la extensión en una instancia experimental de Visual **Debug** > Studio, asegúrese de que la configuración del proyecto para la acción **Depurar inicio** tiene el valor Iniciar programa **externo**: establecido en el archivo *devenv.exe* de la instalación de Visual Studio 2017.

Podría tener el siguiente aspecto: C:-Archivos de programa (x86)-Microsoft Visual Studio-2017-Enterprise-Common7-IDE-devenv.exe



The screenshot shows the "Debug" configuration window in Visual Studio. The left sidebar has a "Debug" tab selected. The main area shows the "Start action" section with three radio buttons: "Start project", "Start external program:" (selected), and "Start browser with URL:". The "Start external program:" option has a text box containing "C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Commo" and a "Browse..." button. Below this is the "Start options" section with a "Command line arguments:" text box containing "/rootsuffix Exp" and a "Working directory:" text box with a "Browse..." button. At the bottom, there are checkboxes for "Use remote machine", "Enable native code debugging", and "Enable SQL Server debugging".

NOTE

La acción de inicio de depuración se almacena normalmente en el archivo `.csproj.user`. Este archivo se incluye normalmente en el archivo `.gitignore` y, por lo tanto, normalmente no se guarda con otros archivos de proyecto cuando se confirma en el control de código fuente. Por lo tanto, si ha extraído la solución fresca del control de código fuente, es probable que el proyecto no tenga ningún valor establecido para Iniciar acción. Los nuevos proyectos VSIX creados con Visual Studio 2017 tendrán un archivo `.csproj.user` creado con valores predeterminados que apuntan al directorio de instalación de Visual Studio actual. Sin embargo, si va a migrar una extensión VSIX v2, es probable que el archivo `.csproj.user` contenga referencias al directorio de instalación de la versión anterior de Visual Studio. Establecer el valor de la > acción Depurar inicio permitirá que se inicie la instancia experimental de Visual Studio correcta cuando intente depurar la extensión.

Compruebe que la extensión se compila correctamente (como VSIX v3)

- Compile el proyecto VSIX.
- Descomprima el VSIX generado.
 - De forma predeterminada, el archivo VSIX se encuentra dentro de `bin/Debug` o `bin/Release` como `[YourCustomExtension].vsix`.
 - Cambie el nombre de `.vsix` a `.zip` para ver fácilmente el contenido.
- Compruebe la existencia de tres archivos:
 - `extension.vsixmanifest`
 - `manifest.json`
 - `catalog.json`

Compruebe cuándo se instalan todos los requisitos previos necesarios

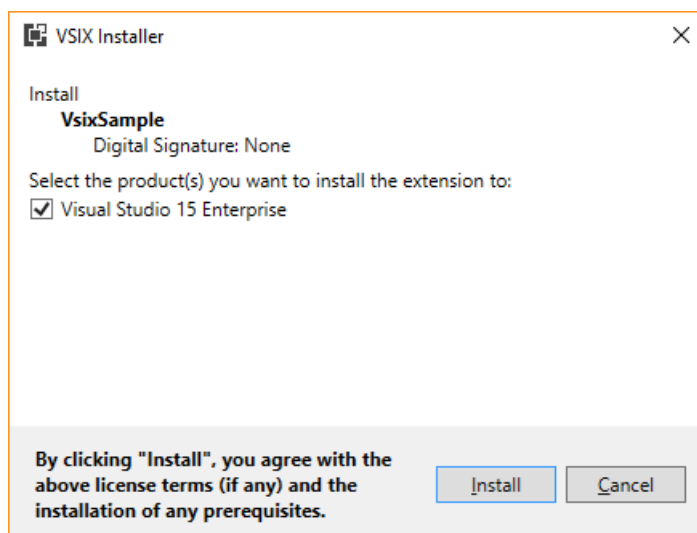
Compruebe que VSIX se instala correctamente en un equipo con todos los requisitos previos necesarios instalados.

NOTE

Antes de instalar cualquier extensión, cierre todas las instancias de Visual Studio.

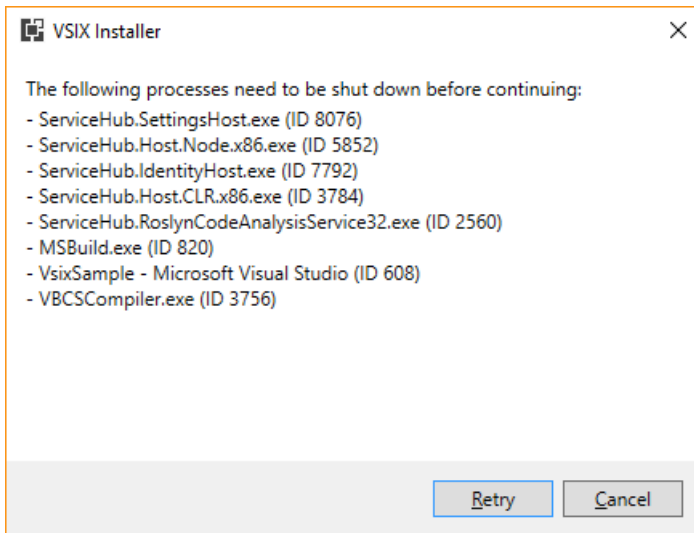
Intente instalar la extensión:

- En Visual Studio 2017



- Opcional: compruebe las versiones anteriores de Visual Studio.
 - Demuestra compatibilidad con versiones anteriores.
 - Debería funcionar para Visual Studio 2012, Visual Studio 2013, Visual Studio 2015.
- Opcional: Compruebe que VSIX Installer Version Checker ofrece una selección de versiones.
 - Incluye versiones anteriores de Visual Studio (si está instalado).
 - Incluye Visual Studio 2017.

Si Visual Studio se abrió recientemente, es posible que vea un cuadro de diálogo como este:



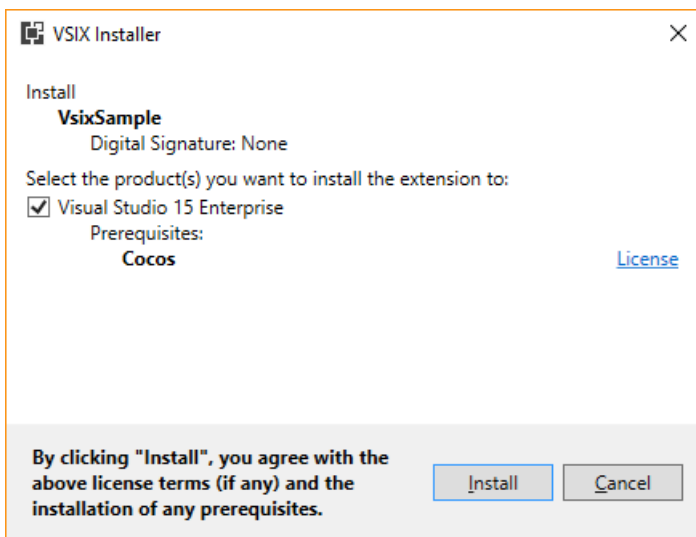
Espere a que los procesos se cierren o finalice manualmente las tareas. Puede encontrar los procesos por el nombre de la lista o con el PID que aparece entre paréntesis.

NOTE

Estos procesos no se apagarán automáticamente mientras se ejecuta una instancia de Visual Studio. Asegúrese de que ha apagado todas las instancias de Visual Studio en el equipo, incluidas las de otros usuarios, y, a continuación, continúe reintentando.

Compruebe cuando falte los requisitos previos requeridos

- Intente instalar la extensión en un equipo con Visual Studio 2017 que NO CONTenga todos los componentes definidos en los requisitos previos (arriba).
- Compruebe que la instalación identifica los componentes que faltan y los enumera como requisito previo en VSIXInstaller.
- Nota: La elevación será necesaria si es necesario instalar algún requisito previo con la extensión.



Decidir sobre los componentes

Al buscar las dependencias, encontrará que una dependencia podría asignarse a varios componentes. Para determinar qué dependencias debe especificar como requisito previo, le sugerimos que elija un componente que tenga una funcionalidad similar a la extensión y que también tenga en cuenta los usuarios y qué tipo de componentes probablemente tendrían instalados o no les importaría instalar. También le sugerimos que la creación de las extensiones de una manera en la que los requisitos previos necesarios satisfagan solo el mínimo que permitirá que se ejecute la extensión y para características adicionales las tenga inactivas si no se detectan determinados componentes.

Para proporcionar orientación adicional, hemos identificado algunos tipos de extensión comunes y sus requisitos previos sugeridos:

TIPO DE EXTENSIÓN	DISPLAY NAME (NOMBRE PARA MOSTRAR)	ID
Editor	Editor de núcleo de Visual Studio	Microsoft.VisualStudio.Component.CoreEditor
Roslyn	C# y Visual Basic	Microsoft.VisualStudio.Component.Roslyn.LanguageServices
WPF	Núcleo de carga de trabajo de escritorio administrado	Microsoft.VisualStudio.Component.ManagedDesktop.Core
Depurador	Depurador Just-In-Time	Microsoft.VisualStudio.Component.Debugger.JustInTime

Buscar iDescomponentes

La lista de componentes ordenados por producto de Visual Studio se encuentra en la carga de trabajo y los identificadores de componentes de [Visual Studio 2017](#). Utilice estos SDK de componentes para los ids de requisitos previos en el manifiesto.

Si no está seguro de qué componente contiene un binario específico, descargue la hoja de [cálculo de asignación binaria](#) > componente .

vs2017-ComponentBinaryMapping.xlsx

Hay cuatro columnas en la hoja de Excel: **Nombre de componente**, **ComponentId**, **Versión** y **Nombres binarios /de archivo**. Puede utilizar los filtros para buscar y encontrar componentes y binarios específicos.

Para todas las referencias, primero determine cuáles están en el componente del editor principal (Microsoft.VisualStudio.Component.CoreEditor). Como mínimo, requerimos que el componente del editor principal se especifique como requisito previo para todas las extensiones. De las referencias que quedan que no están en el editor principal, agregue filtros en la sección **Binarios / Nombres** de archivos para buscar componentes que tengan cualquiera de los subconjuntos de esas referencias.

Ejemplos:

- Si tiene una extensión de depurador y sabe que el proyecto tiene una referencia a *VSDebugEng.dll* y *VSDebug.dll*, haga clic en el botón de filtro en el encabezado **Binaries / Files Names** . Busque "VSDebugEng.dll" y seleccione *Aceptar*. A continuación, haga clic en el botón de filtro en el **binarios / archivos nombres** encabezado de nuevo y busque "VSDebug.dll". Active la casilla **Agregar selección actual para filtrar** y seleccione *Aceptar*. Ahora examine el Nombre de **componente** para encontrar el componente que esté más relacionado con el tipo de extensión. En este ejemplo, elegiría el depurador Just-In-Time y lo agregaría a su vsixmanifest.
- Si sabe que el proyecto se ocupa de los elementos del depurador, puede buscar en "depurador" en el cuadro de búsqueda de filtro para ver qué componentes contienen el depurador en su nombre.

Especificar una versión de Visual Studio 2017

Si la extensión requiere una versión específica de Visual Studio 2017, por ejemplo, depende de una característica publicada en 15.3, debe especificar el número de compilación en el destino de **instalación** de VSIX . Por ejemplo, la versión 15.3 tiene un número de compilación de '15.0.26730.3'. Puede ver la asignación de versiones para generar números [aquí](#). El uso del número de versión '15.3' no funcionará correctamente.

Si la extensión requiere 15.3 o superior, declararía la **versión InstallationTarget** como [15.0.26730.3, 16.0):

```
<Installation>
  <InstallationTarget Id="Microsoft.VisualStudio.Community" Version="[15.0.26730.3, 16.0)" />
</Installation>
```


Cómo: hacer que las extensiones sean compatibles con Visual Studio 2019/2017 y Visual Studio 2015

18/01/2020 • 14 minutes to read • [Edit Online](#)

En este documento se explica cómo realizar un recorrido de ida y vuelta de proyectos de extensibilidad entre Visual Studio 2015 y Visual Studio 2019 o Visual Studio 2017. Después de completar esta actualización, un proyecto podrá abrir, compilar, instalar y ejecutar en Visual Studio 2015 y Visual Studio 2019 o 2017. Como referencia, algunas extensiones que pueden llevarse a cabo entre Visual Studio 2015 y Visual Studio 2019 o 2017 se pueden encontrar en los [ejemplos de extensibilidad del SDK de vs.](#)

Si solo tiene previsto compilar en Visual Studio 2019/2017, pero quiere que el resultado de la extensión VSIX se ejecute en Visual Studio 2015 y Visual Studio 2019/2017, consulte el [documento de migración](#) de la extensión.

NOTE

Debido a los cambios en Visual Studio entre versiones, algunas cosas que funcionaban en una versión no funcionan en otro. Asegúrese de que las características a las que está intentando obtener acceso están disponibles en ambas versiones o que la extensión tendrá resultados inesperados.

A continuación se muestra un resumen de los pasos que debe completar en este documento para realizar un recorrido de ida y vuelta por una VSIX:

1. Importe los paquetes de NuGet correctos.
2. Actualizar manifiesto de extensión:
 - Destino de la instalación
 - Requisitos previos
3. Actualizar CSProj:
 - Actualice `<MinimumVisualStudioVersion>`.
 - Agregue la propiedad `<VsixType>`.
 - Agregue la propiedad de depuración `($DevEnvDir)` 3 veces.
 - Agregue condiciones para importar los destinos y las herramientas de compilación.
4. Compilar y probar

Configuración del entorno

En este documento se da por supuesto que tiene lo siguiente instalado en el equipo:

- Visual Studio 2015 con el SDK de VS instalado
- Visual Studio 2019 o 2017 con la carga de trabajo de extensibilidad instalada

Enfoque recomendado

Se recomienda encarecidamente iniciar esta actualización con Visual Studio 2015, en lugar de Visual Studio 2019 o 2017. La principal ventaja de desarrollar en Visual Studio 2015 es asegurarse de que no se hace referencia a ensamblados que no están disponibles en Visual Studio 2015. Si realiza el desarrollo en Visual Studio 2019 o 2017, existe un riesgo de que se pueda introducir una dependencia en un ensamblado que solo existe en Visual Studio 2019 o 2017.

Asegúrese de que no hay ninguna referencia a Project. JSON

Más adelante en este documento, se insertarán instrucciones de importación condicional en el archivo `*.csproj`. Esto no funcionará si las referencias de NuGet se almacenan en `Project.JSON`. Como tal, se recomienda que mueva todas las referencias de NuGet al archivo `packages.config`. Si el proyecto contiene un archivo `Project.JSON`:

- Tome nota de las referencias en `Project.JSON`.
- En el **Explorador de soluciones**, elimine el archivo `Project.JSON` del proyecto. Esto elimina el archivo `Project.JSON` y lo quita del proyecto.
- Vuelva a agregar las referencias de NuGet al proyecto:
 - Haga clic con el botón derecho en la **solución** y elija **administrar paquetes NuGet para la solución**.
 - Visual Studio crea automáticamente el archivo `packages.config`.

NOTE

Si el proyecto contenía paquetes EnvDTE, es posible que necesiten agregarse haciendo clic con el botón derecho en **referencias** seleccionando **Agregar referencia** y agregando la referencia adecuada. El uso de paquetes NuGet puede crear errores al intentar compilar el proyecto.

Agregar las herramientas de compilación adecuadas

Tenemos que asegurarnos de agregar herramientas de compilación que nos permitan compilar y depurar de forma adecuada. Microsoft ha creado un ensamblado para este llamado Microsoft.VisualStudio.SDK.BuildTasks.

Para compilar e implementar un VSIXv3 en Visual Studio 2015 y 2019/2017, necesitará los siguientes paquetes NuGet:

VERSION	HERRAMIENTAS INTEGRADAS
Visual Studio 2015	Microsoft.VisualStudio.Sdk.BuildTasks.14.0
Visual Studio 2019 o 2017	Microsoft.VSSDK.BuildTool

Para ello:

- Agregue el paquete NuGet Microsoft.VisualStudio.SDK.BuildTasks. 14.0 al proyecto.
- Si el proyecto no contiene Microsoft.VSSDK.BuildTools, agréguelo.
- Asegúrese de que la versión de Microsoft.VSSDK.BuildTools es 15. x o superior.

Actualizar manifiesto de extensión

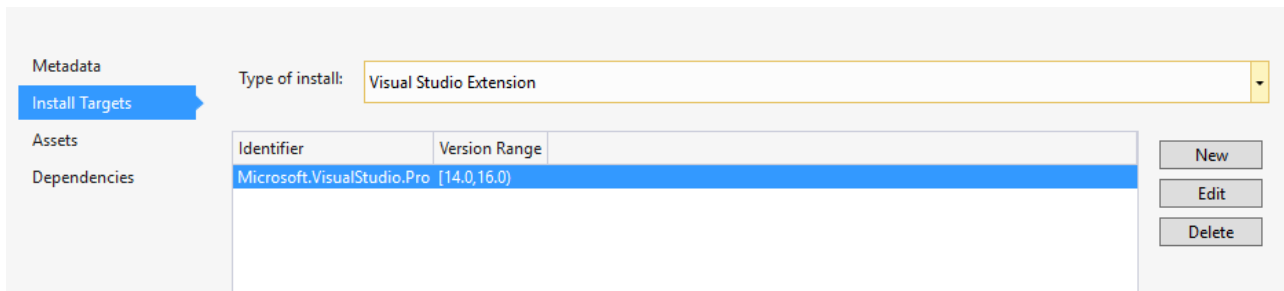
1. destinos de instalación

Es necesario indicar a Visual Studio qué versiones se van a usar para crear un VSIX. Normalmente, estas referencias son a la versión 14,0 (Visual Studio 2015), la versión 15,0 (Visual Studio 2017) o la versión 16,0 (Visual Studio 2019). En nuestro caso, queremos crear una VSIX que vaya a instalar una extensión para ambos, por lo que necesitamos tener como destino ambas versiones. Si desea que la VSIX se compile e instale en versiones anteriores a la 14,0, esto se puede hacer estableciendo el número de versión anterior. sin embargo, ya no se admiten la versión 10,0 y anteriores.

- Abra el archivo `source.Extension.vsixmanifest` en Visual Studio.
- Abra la pestaña **destinos de instalación**.
- Cambie el **intervalo de versiones** a `[14,0, 17,0). ' ['` Indica a Visual Studio que incluya 14,0 y todas las versiones posteriores. `) ']` Indica a Visual Studio que incluya todas las versiones hasta la versión 17,0, pero sin

incluirla.

- Guarde todos los cambios y cierre todas las instancias de Visual Studio.



2. agregar requisitos previos al archivo *Extension.vsixmanifest*

Necesitamos el editor principal de Visual Studio como requisito previo. Abra Visual Studio y use el diseñador de manifiestos actualizado para insertar los requisitos previos.

Para hacerlo manualmente:

- Navegue al directorio del proyecto en el explorador de archivos.
- Abra el archivo *Extension.vsixmanifest* con un editor de texto.
- Agregue la siguiente etiqueta:

```
<Prerequisites>
  <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[15.0,16.0]" DisplayName="Visual
Studio core editor" />
</Prerequisites>
```

- Guarde y cierre el archivo.

NOTE

Es posible que tenga que editar manualmente la versión de requisitos previos para asegurarse de que es compatible con todas las versiones de Visual Studio 2019 o 2017. Esto se debe a que el diseñador insertará la versión mínima como la versión actual de Visual Studio (por ejemplo, 15.0.26208.0). Sin embargo, dado que otros usuarios pueden tener una versión anterior, querrá editarlo manualmente en 15,0.

En este momento, el archivo de manifiesto debería tener un aspecto similar al siguiente:

```
<PackageManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011" xmlns:d="http://schemas.microsoft.com/developer/vsx-schema-2011" >
  <Metadata>
    <Identity Id="TypingSpeedMeter1" Version="1.0" Language="en-US" Publisher="Microsoft" />
    <DisplayName>Typing Speed</DisplayName>
    <Description>This is a sample viewport-relative adornment extension to the Visual Studio Editor that monitors the average speed that you are typing.
    <Icon>Adornment_small.png</Icon>
    <PreviewImage>Adornment_large.png</PreviewImage>
  </Metadata>
  <Installation>
    <InstallationTarget Id="Microsoft.VisualStudio.Pro" Version="[14.0,16.0]" />
  </Installation>
  <Dependencies>
    <Dependency Id="Microsoft.Framework.NDP" DisplayName="Microsoft .NET Framework" d:Source="Manual" Version="4.5" />
  </Dependencies>
  <Assets>
    <Asset Type="Microsoft.VisualStudio.MefComponent" d:Source="Project" d:ProjectName="%CurrentProject%" Path="%CurrentProject%" />
  </Assets>
  <Prerequisites>
    <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[15.0,16.0]" DisplayName="Visual Studio core editor" />
  </Prerequisites>
</PackageManifest>
```

Modificar el archivo de proyecto (de proyecto. csproj)

Se recomienda encarecidamente tener una referencia a un .csproj abierto mientras realiza este paso. Puede encontrar varios ejemplos [aquí](#). Seleccione cualquier ejemplo de extensibilidad, busque el archivo .csproj como referencia y ejecute los pasos siguientes:

- Navegue al directorio del proyecto en el **Explorador de archivos**.
- Abra el archivo de *proyecto.csproj* con un editor de texto.

1. actualizar el MinimumVisualStudioVersion

- Establezca la versión mínima de Visual Studio en `$(VisualStudioVersion)` y agregue una instrucción condicional para ella. Agregue estas etiquetas si no existen. Asegúrese de que las etiquetas se establecen como se indica a continuación:

```
<VisualStudioVersion Condition="'$(VisualStudioVersion)' == ''">14.0</VisualStudioVersion>
<MinimumVisualStudioVersion>$(VisualStudioVersion)</MinimumVisualStudioVersion>
```

2. Agregue la propiedad VsixType.

- Agregue la siguiente etiqueta `<VsixType>v3</VsixType>` a un grupo de propiedades.

NOTE

Se recomienda agregar esto debajo de la etiqueta `<OutputType></OutputType>`.

3. agregar las propiedades de depuración

- Agregue el siguiente grupo de propiedades:

```
<PropertyGroup>
  <StartAction>Program</StartAction>
  <StartProgram>$(DevEnvDir)devenv.exe</StartProgram>
  <StartArguments>/rootsuffix Exp</StartArguments>
</PropertyGroup>
```

- Elimine todas las instancias del siguiente ejemplo de código del archivo *.csproj* y los archivos *.csproj*. *User*:

```
<StartAction>Program</StartAction>
<StartProgram>$(ProgramFiles)\Microsoft Visual Studio 14.0\Common7\IDE\devenv.exe</StartProgram>
<StartArguments>/rootsuffix Exp</StartArguments>
```

4. agregar condiciones a las importaciones de herramientas de compilación

- Agregue instrucciones condicionales adicionales a las etiquetas de `<import>` que tengan una referencia de Microsoft.VSSDK.BuildTools. Inserte `'$(VisualStudioVersion)' != '14.0' And` al principio de la instrucción de condición. Estas instrucciones aparecerán en el encabezado y el pie de página del archivo *csproj*.

Por ejemplo:

```
<Import Project="packages\Microsoft.VSSDK.BuildTools.15.0.26201..." Condition="'$(VisualStudioVersion)' != '14.0'
And Exists(...)" />
```

- Agregue instrucciones condicionales adicionales a las etiquetas `<import>` que tengan un Microsoft.VisualStudio.SDK.BuildTasks. 14.0. Inserte `'$(VisualStudioVersion)' == '14.0' And` al principio de la instrucción de condición. Estas instrucciones aparecerán en el encabezado y el pie de página del archivo *csproj*.

Por ejemplo:

```
<Import Project="packages\Microsoft.VisualStudio.Sdk.BuildTasks.14.0.14.0..." Condition="'$(VisualStudioVersion)'
== '14.0' And Exists(...)" />
```

- Agregue instrucciones condicionales adicionales a las etiquetas de `<Error>` que tengan una referencia de Microsoft. VSSDK. BuildTools. Para ello, inserte `'$(VisualStudioVersion)' != '14.0' And Exists('packages\Microsoft.VSSDK.BuildTools.15.0.26201...' />` al principio de la instrucción de condición. Estas instrucciones aparecerán en el pie de página del archivo csproj.

Por ejemplo:

```
<Error Condition=" '$(VisualStudioVersion)' != '14.0' And  
Exists('packages\Microsoft.VSSDK.BuildTools.15.0.26201...' />
```

- Agregue instrucciones condicionales adicionales a las etiquetas `<Error>` que tengan un Microsoft. VisualStudio. SDK. BuildTasks. 14.0. Inserte `'$(VisualStudioVersion)' == '14.0' And` al principio de la instrucción de condición. Estas instrucciones aparecerán en el pie de página del archivo csproj.

Por ejemplo:

```
<Error Condition=" '$(VisualStudioVersion)' == '14.0' And  
Exists('packages\Microsoft.VisualStudio.Sdk.BuildTasks.14.0.14.0...' />
```

- Guarde el archivo csproj y ciérrelo.
 - Tenga en cuenta que si usa más de un proyecto en la solución, establezca este proyecto como proyecto de inicio mediante "establecer como proyecto de inicio" en el menú contextual del proyecto. Esto garantiza que Visual Studio vuelva a abrir este proyecto después de descargarlo.

Prueba de las instalaciones de la extensión en Visual Studio 2015 y Visual Studio 2019 o 2017

En este momento, el proyecto debe estar listo para compilar un VSIXv3 que se pueda instalar en Visual Studio 2015 y Visual Studio 2017.

- Abra el proyecto en Visual Studio 2015.
- Compile el proyecto y confirme en la salida que un VSIX se compila correctamente.
- Navegue hasta el directorio del proyecto.
- Abra la carpeta `\bin\debug`.
- Haga doble clic en el archivo VSIX e instale su extensión en Visual Studio 2015 y Visual Studio 2019/2017.
- Asegúrese de que puede ver la extensión en **herramientas > extensiones y actualizaciones** en la sección **instalado**.
- Intente ejecutar o usar la extensión para comprobar que funciona.

Extensibility-Samples > Code_Sweep > C# > VsPackage > bin > Debug

Search Debug

Name	Date modified	Type	Size
Microsoft.VisualStudio.Threading.dll	7/2/2015 5:19 PM	Application extens...	182 KB
Microsoft.VisualStudio.Threading	7/2/2015 5:19 PM	XML Document	330 KB
Microsoft.VisualStudio.Utilities.dll	8/5/2015 4:54 PM	Application extens...	342 KB
Microsoft.VisualStudio.Utilities	8/5/2015 4:54 PM	XML Document	213 KB
Microsoft.VisualStudio.Validation.dll	7/2/2015 12:21 PM	Application extens...	29 KB
Microsoft.VisualStudio.Validation	7/2/2015 12:21 PM	XML Document	28 KB
Microsoft.VisualStudio.VSHelp.dll	6/20/2016 2:45 PM	Application extens...	27 KB
sample_term_table	5/7/2017 8:07 PM	XML Document	3 KB
Scanner.dll	5/31/2017 4:40 PM	Application extens...	32 KB
Scanner.pdb	5/31/2017 4:40 PM	Program Debug D...	82 KB
stdole.dll	6/20/2016 2:45 PM	Application extens...	32 KB
Utilities.dll	5/31/2017 4:40 PM	Application extens...	13 KB
Utilities.pdb	5/31/2017 4:40 PM	Program Debug D...	30 KB
VSPackage.dll	5/31/2017 4:40 PM	Application extens...	87 KB
VSPackage.pdb	5/31/2017 4:40 PM	Program Debug D...	162 KB
VSPackage	5/31/2017 4:40 PM	PKGDEF File	2 KB
VSPackage	5/31/2017 4:40 PM	Microsoft Visual S...	163 KB

NOTE

Si el proyecto se bloquea con el mensaje que **abre el archivo**, fuerce el cierre de Visual Studio, vaya al directorio del proyecto, muestre las carpetas ocultas y elimine la carpeta . vs .

Protocolo de servidor de lenguaje

06/04/2020 • 16 minutes to read • [Edit Online](#)

¿Qué es el protocolo de servidor de idioma?

La compatibilidad con funciones de edición enriquecidas como las autocompletars de código fuente o **Ir a definición** para un lenguaje de programación en un editor o IDE es tradicionalmente muy difícil y consume mucho tiempo. Normalmente requiere escribir un modelo de dominio (un analizador, un analizador, un comprobador de tipos, un generador y más) en el lenguaje de programación del editor o IDE. Por ejemplo, el plugin CDT de Eclipse, que proporciona soporte para C/C++ en el IDE de Eclipse, está escrito en Java ya que el IDE de Eclipse está escrito en Java. Siguiendo este enfoque, significaría implementar un modelo de dominio de C/C++ en TypeScript para Visual Studio Code y un modelo de dominio independiente en C- para Visual Studio.

La creación de modelos de dominio específicos del lenguaje también es mucho más fácil si una herramienta de desarrollo puede reutilizar bibliotecas específicas del lenguaje existentes. Sin embargo, estas bibliotecas se implementan normalmente en el propio lenguaje de programación (por ejemplo, se implementan buenos modelos de dominio C/C++ en C/C++). Integrar una biblioteca C/C++ en un editor escrito en TypeScript es técnicamente posible pero difícil de hacer.

Servidores de idiomas

Otro enfoque es ejecutar la biblioteca en su propio proceso y utilizar la comunicación entre procesos para hablar con ella. Los mensajes enviados de ida y vuelta forman un protocolo. El protocolo de servidor de lenguaje (LSP) es el producto de estandarizar los mensajes intercambiados entre una herramienta de desarrollo y un proceso de servidor de lenguaje. El uso de servidores de lenguaje o demonios no es una idea nueva o novedosa. Editores como Vim y Emacs han estado haciendo esto durante algún tiempo para proporcionar soporte de autocompletado semántico. El objetivo del LSP era simplificar este tipo de integraciones y proporcionar un marco útil para exponer las características del lenguaje a una variedad de herramientas.

Tener un protocolo común permite la integración de características del lenguaje de programación en una herramienta de desarrollo con un mínimo alboroto mediante la reutilización de una implementación existente del modelo de dominio del lenguaje. Un back-end de servidor de lenguaje podría escribirse en PHP, Python o Java y el LSP permite integrarlo fácilmente en una variedad de herramientas. El protocolo funciona en un nivel común de abstracción para que una herramienta pueda ofrecer servicios de lenguaje enriquecido sin necesidad de comprender completamente los matices específicos del modelo de dominio subyacente.

Cómo comenzó el trabajo en el LSP

El LSP ha evolucionado con el tiempo y hoy está en la versión 3.0. Se inició cuando OmniSharp recogió el concepto de un servidor de lenguaje para proporcionar funciones de edición enriquecidas para C. Inicialmente, OmniSharp utilizaba el protocolo HTTP con una carga JSON y se ha integrado en varios editores, incluido [Visual Studio Code](#).

Al mismo tiempo, Microsoft comenzó a trabajar en un servidor de lenguaje TypeScript, con la idea de admitir TypeScript en editores como Emacs y Sublime Text. En esta implementación, un editor se comunica a través de stdin/stdout con el proceso de servidor TypeScript y utiliza una carga JSON inspirada en el protocolo de depurador V8 para solicitudes y respuestas. El servidor TypeScript se ha integrado en el plugin TypeScript Sublime y VS Code para la edición de TypeScript enriquecida.

Después de haber integrado dos servidores de lenguaje diferentes, el equipo de VS Code comenzó a explorar un protocolo de servidor de lenguaje común para editores e IDE. Un protocolo común permite a un proveedor de lenguaje crear un único servidor de idioma que pueden ser consumidos por diferentes IDE. Un consumidor de servidor de lenguaje solo tiene que implementar el lado cliente del protocolo una vez. Esto da como resultado una

situación de ganar-ganar tanto para el proveedor de idioma como para el consumidor de idiomas.

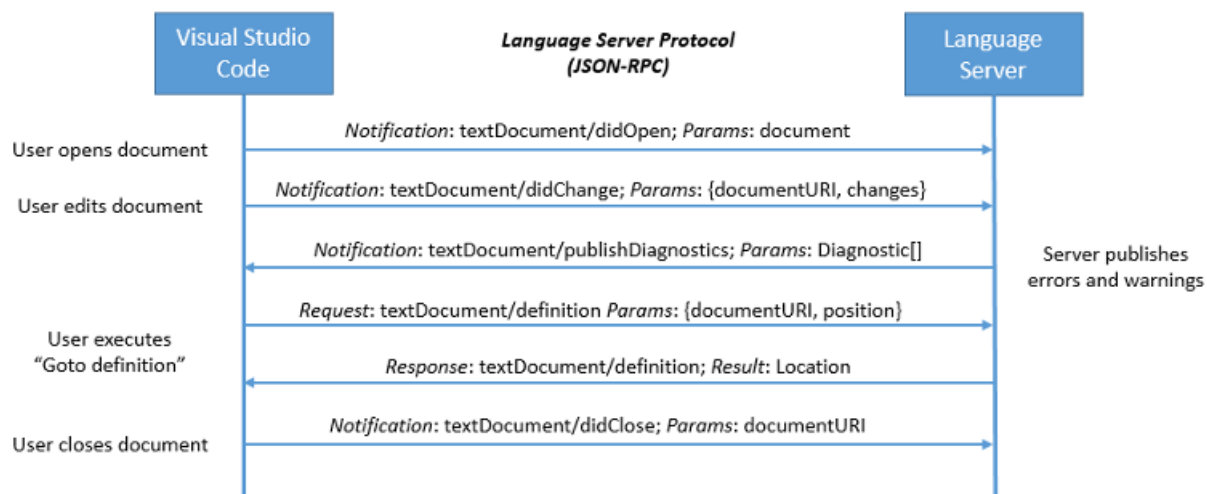
El protocolo de servidor de lenguaje se inició con el protocolo utilizado por el servidor TypeScript, expandiéndolo con más características de lenguaje inspiradas en la API de lenguaje de VS Code. El protocolo está respaldado por JSON-RPC para la invocación remota debido a su simplicidad y bibliotecas existentes.

El equipo de VS Code creó el prototipo del protocolo mediante la implementación de varios servidores de lenguaje linter que responden a las solicitudes de pelusa (escanear) un archivo y devolver un conjunto de advertencias y errores detectados. El objetivo era pelar un archivo a medida que el usuario edita en un documento, lo que significa que habrá muchas solicitudes de linting durante una sesión de editor. Tenía sentido mantener un servidor en funcionamiento para que no era necesario iniciar un nuevo proceso de linting para cada edición de usuario. Se implementaron varios servidores linter, incluidas las extensiones ESLint y TSLint de VS Code. Estos dos servidores linter se implementan en TypeScript/JavaScript y se ejecutan en Node.js. Comparten una biblioteca que implementa la parte de cliente y servidor del protocolo.

Cómo funciona el LSP

Un servidor de lenguaje se ejecuta en su propio proceso y herramientas como Visual Studio o VS Code se comunican con el servidor mediante el protocolo de lenguaje a través de JSON-RPC. Otra ventaja del servidor de lenguaje que funciona en un proceso dedicado es que se evitan los problemas de rendimiento relacionados con un único modelo de proceso. El canal de transporte real puede ser stdio, sockets, canalizaciones con nombre o ipc de nodo si tanto el cliente como el servidor están escritos en Node.js.

A continuación se muestra un ejemplo de cómo una herramienta y un servidor de lenguaje se comunican durante una sesión de edición de rutina:



- **El usuario abre un archivo (denominado documento) en la herramienta:** La herramienta notifica al servidor de idioma que un documento está abierto ('textDocument/didOpen'). A partir de ahora, la verdad sobre el contenido del documento ya no está en el sistema de archivos, sino que la herramienta lo guarda en la memoria.
- **El usuario realiza modificaciones:** La herramienta notifica al servidor sobre el cambio de documento ('textDocument/didChange') y la información semántica del programa es actualizada por el servidor de lenguaje. A medida que esto sucede, el servidor de lenguaje analiza esta información y notifica a la herramienta con los errores y advertencias detectados ('textDocument/publishDiagnostics').
- **El usuario ejecuta "Ir a definición" en un símbolo en el editor:** La herramienta envía una solicitud 'textDocument/definition' con dos parámetros: (1) el URI del documento y (2) la posición de texto desde donde se inició la solicitud Ir a definición al servidor. El servidor responde con el URI del documento y la posición de la definición del símbolo dentro del documento.
- **El usuario cierra el documento (archivo):** se envía una notificación 'textDocument/didClose' desde la

herramienta, informando al servidor de lenguaje que el documento ya no está en la memoria y que el contenido actual está ahora actualizado en el sistema de archivos.

Este ejemplo ilustra cómo el protocolo se comunica con el servidor de lenguaje en el nivel de características del editor como "Ir a definición", "Buscar todas las referencias". Los tipos de datos utilizados por el protocolo son editor o IDE 'tipos de datos' como el documento de texto abierto actualmente y la posición del cursor. Los tipos de datos no están en el nivel de un modelo de dominio de lenguaje de programación que normalmente proporcionaría árboles de sintaxis abstracta y símbolos del compilador (por ejemplo, tipos resueltos, espacios de nombres, ...). Esto simplifica significativamente el protocolo.

Ahora echemos un vistazo a la solicitud 'textDocument/definition' con más detalle. A continuación se muestran las cargas útiles que van entre la herramienta cliente y el servidor de idioma para la solicitud "Ir a definición" en un documento C++.

Esta es la petición:

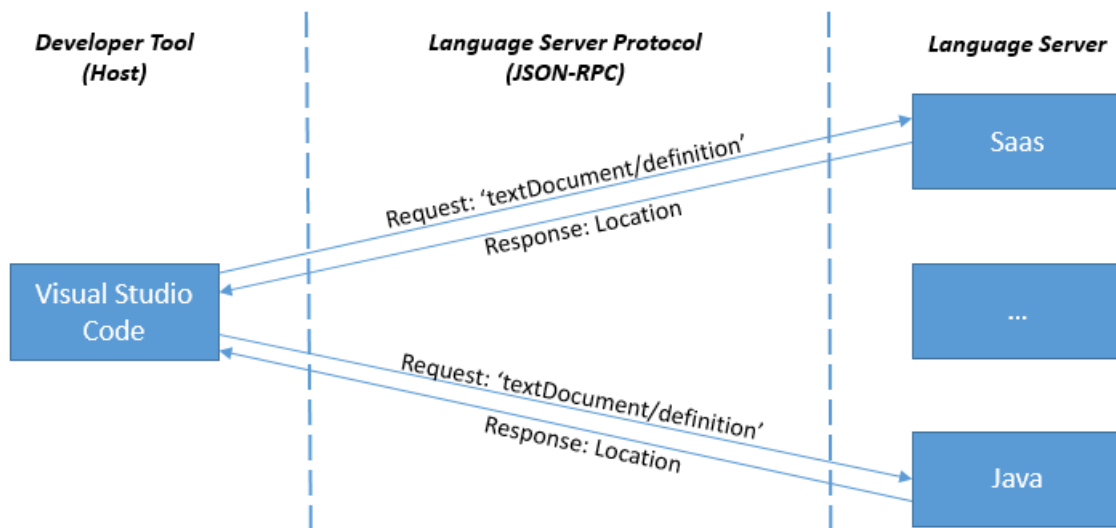
```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "textDocument/definition",
  "params": {
    "textDocument": {
      "uri": "file:///p%3A/mseng/VSCode/Playgrounds/cpp/use.cpp"
    },
    "position": {
      "line": 3,
      "character": 12
    }
  }
}
```

Esta es la respuesta:

```
{
  "jsonrpc": "2.0",
  "id": "1",
  "result": {
    "uri": "file:///p%3A/mseng/VSCode/Playgrounds/cpp/provide.cpp",
    "range": {
      "start": {
        "line": 0,
        "character": 4
      },
      "end": {
        "line": 0,
        "character": 11
      }
    }
  }
}
```

En retrospectiva, describir los tipos de datos en el nivel del editor en lugar de en el nivel del modelo de lenguaje de programación es una de las razones del éxito del protocolo de servidor de lenguaje. Es mucho más sencillo estandarizar un URI de documento de texto o una posición de cursor en comparación con la estandarización de un árbol de sintaxis abstracta y símbolos del compilador en diferentes lenguajes de programación.

Cuando un usuario está trabajando con diferentes idiomas, VS Code normalmente inicia un servidor de lenguaje para cada lenguaje de programación. El ejemplo siguiente muestra una sesión donde el usuario trabaja en archivos Java y SASS.



Capacidades

No todos los servidores de lenguaje pueden admitir todas las características definidas por el protocolo. Por lo tanto, el cliente y el servidor anuncian su conjunto de características compatibles a través de 'capacidades'. Por ejemplo, un servidor anuncia que puede controlar la solicitud 'textDocument/definition', pero es posible que no controle la solicitud 'workspace/symbol'. Del mismo modo, los clientes pueden anunciar que pueden proporcionar notificaciones de "a punto de guardar" antes de guardar un documento, de modo que un servidor pueda calcular las ediciones textuales para dar formato automáticamente al documento editado.

Integración de un servidor de idiomas

La integración real de un servidor de lenguaje en una herramienta determinada no está definida por el protocolo de servidor de lenguaje y se deja a los implementadores de la herramienta. Algunas herramientas integran servidores de lenguaje de forma genérica al tener una extensión que puede iniciar y hablar con cualquier tipo de servidor de lenguaje. Otros, como VS Code, crean una extensión personalizada por servidor de lenguaje, de modo que una extensión todavía pueda proporcionar algunas características de lenguaje personalizado.

Para simplificar la implementación de clientes y servidores de lenguaje, hay bibliotecas o SDK para las partes de cliente y servidor. Estas bibliotecas se proporcionan para diferentes idiomas. Por ejemplo, hay un [módulo npm](#) de cliente de lenguaje para facilitar la integración de un servidor de lenguaje en una extensión de código VS y otro [módulo npm](#) de servidor de idioma para escribir un servidor de lenguaje mediante Node.js. Esta es la [lista](#) actual de bibliotecas de soporte.

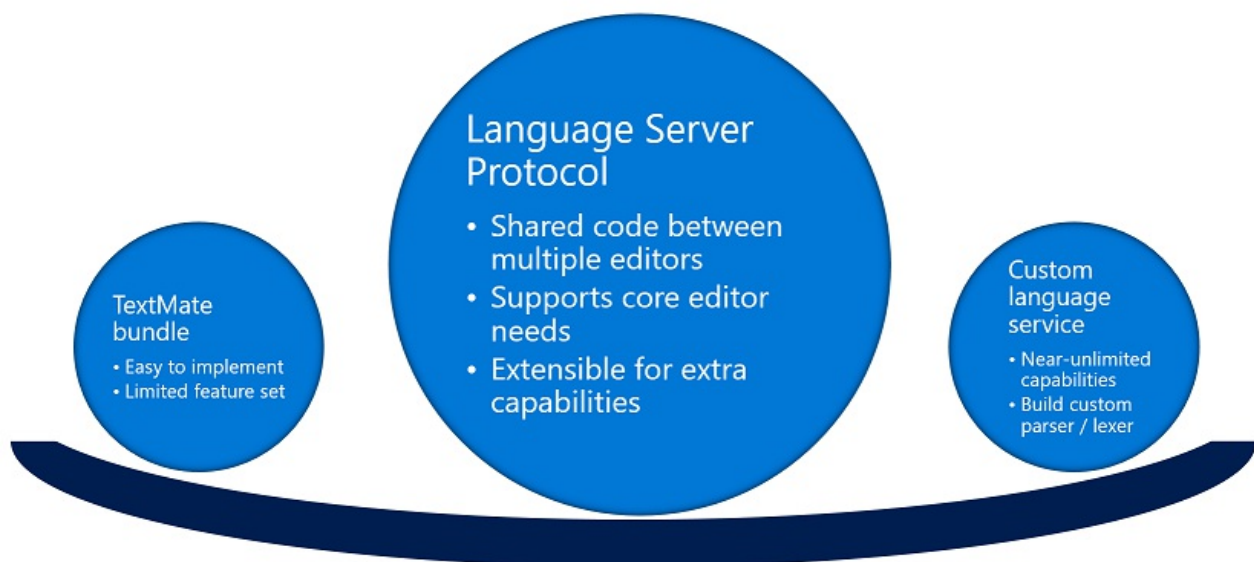
Uso del protocolo de servidor de lenguaje en Visual Studio

- [Agregar una extensión](#) de protocolo de servidor de lenguaje: obtenga información sobre cómo integrar un servidor de lenguaje en Visual Studio.

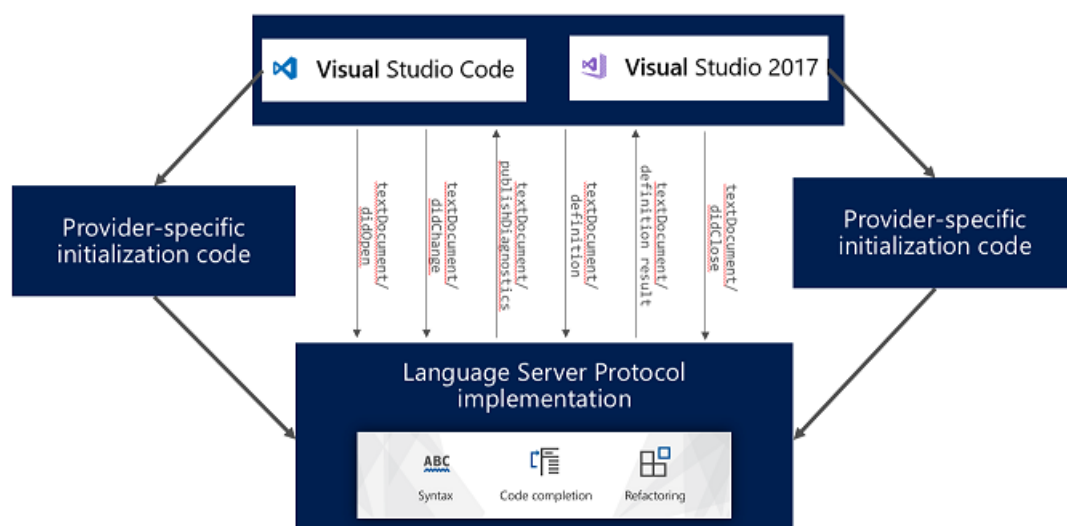
Adición de una extensión del protocolo de servidor de lenguaje

06/04/2020 • 28 minutes to read • [Edit Online](#)

El protocolo de servidor de lenguaje (LSP) es un protocolo común, en forma de JSON RPC v2.0, que se utiliza para proporcionar características de servicio de lenguaje a varios editores de código. Mediante el protocolo, los desarrolladores pueden escribir un único servidor de lenguaje para proporcionar características de servicio de lenguaje como IntelliSense, diagnóstico de errores, buscar todas las referencias, etc., a varios editores de código que admiten el LSP. Tradicionalmente, los servicios de lenguaje en Visual Studio se pueden agregar mediante el uso de archivos de gramática TextMate para proporcionar funcionalidades básicas, como resaltado de sintaxis o mediante la escritura de servicios de lenguaje personalizados que usan el conjunto completo de API de extensibilidad de Visual Studio para proporcionar datos más enriquecidos. Con la compatibilidad de Visual Studio con LSP, hay una tercera opción.



Protocolo de servidor de lenguaje



En este artículo se describe cómo crear una extensión de Visual Studio que usa un servidor de lenguaje basado en

LSP. Se supone que ya ha desarrollado un servidor de lenguaje basado en LSP y solo desea integrarlo en Visual Studio.

Para la compatibilidad con Visual Studio, los servidores de lenguaje pueden comunicarse con el cliente (Visual Studio) a través de cualquier mecanismo de transmisión basado en secuencias, por ejemplo:

- Flujos de entrada/salida estándar
- Canalizaciones con nombre
- Sockets (solo TCP)

La intención del LSP y la compatibilidad con él en Visual Studio es incorporar servicios de lenguaje que no forman parte del producto de Visual Studio. No está pensado para extender los servicios de lenguaje existentes (por lo que C) en Visual Studio. Para ampliar los idiomas existentes, consulte la guía de extensibilidad del servicio de lenguaje (por ejemplo, "[Roslyn](#)".NET Compiler Platform) o consulte [Extender el editor](#) y los servicios de lenguaje .

Para obtener más información sobre el protocolo en sí, consulte la documentación [aquí](#).

Para obtener más información sobre cómo crear un servidor de lenguaje de ejemplo o cómo integrar un servidor de lenguaje existente en Visual Studio Code, vea la documentación [aquí](#).

Características compatibles con Language Server Protocol

En las tablas siguientes se muestran qué características de LSP se admiten en Visual Studio:

MESSAGE	TIENE SOPORTE EN VISUAL STUDIO
initialize	sí
inicializado	sí
shutdown	sí
exit	sí
\$/cancelRequest	sí
window/showMessage	sí
window/showMessageRequest	sí
window/logMessage	sí
telemetría/evento	
cliente/registroCapability	
client/unregisterCapability	
workspace/didChangeConfiguration	sí
workspace/didChangeWatchedFiles	sí
espacio de trabajo/símbolo	sí

MESSAGE	TIENE SOPORTE EN VISUAL STUDIO
workspace/executeCommand	sí
workspace/applyEditor	sí
textDocument/publishDiagnostics	sí
textDocument/didOpen	sí
textDocument/didChange	sí
textDocument/willSave	
textDocument/willSaveWaitUntil	
textDocument/didSave	sí
textDocument/didClose	sí
textDocument/completion	sí
finalización/resolución	sí
textDocument/hover	sí
textDocument/signatureHelp	sí
textDocument/references	sí
textDocument/documentHighlight	sí
textDocument/documentSymbol	sí
textDocument/formatting	sí
textDocument/rangeFormatting	sí
textDocument/onTypeFormatting	
textDocument/definición	sí
textDocument/codeAction	sí
textDocument/codeLens	
codeLens/resolve	
textDocument/documentLink	
documentLink/resolve	

MESSAGE	TIENE SOPORTE EN VISUAL STUDIO
textDocument/rename	sí

Introducción

NOTE

A partir de Visual Studio 2017 versión 15.8, la compatibilidad con el protocolo de servidor de lenguaje común está integrada en Visual Studio. Si ha creado extensiones LSP con la versión preliminar de [Language Server Client VSIX](#), dejarán de funcionar una vez que actualice a la versión 15.8 o posterior. Tendrá que hacer lo siguiente para que las extensiones LSP vuelvan a funcionar:

1. Desinstale microsoft Visual Studio Language Server Protocol Preview VSIX.

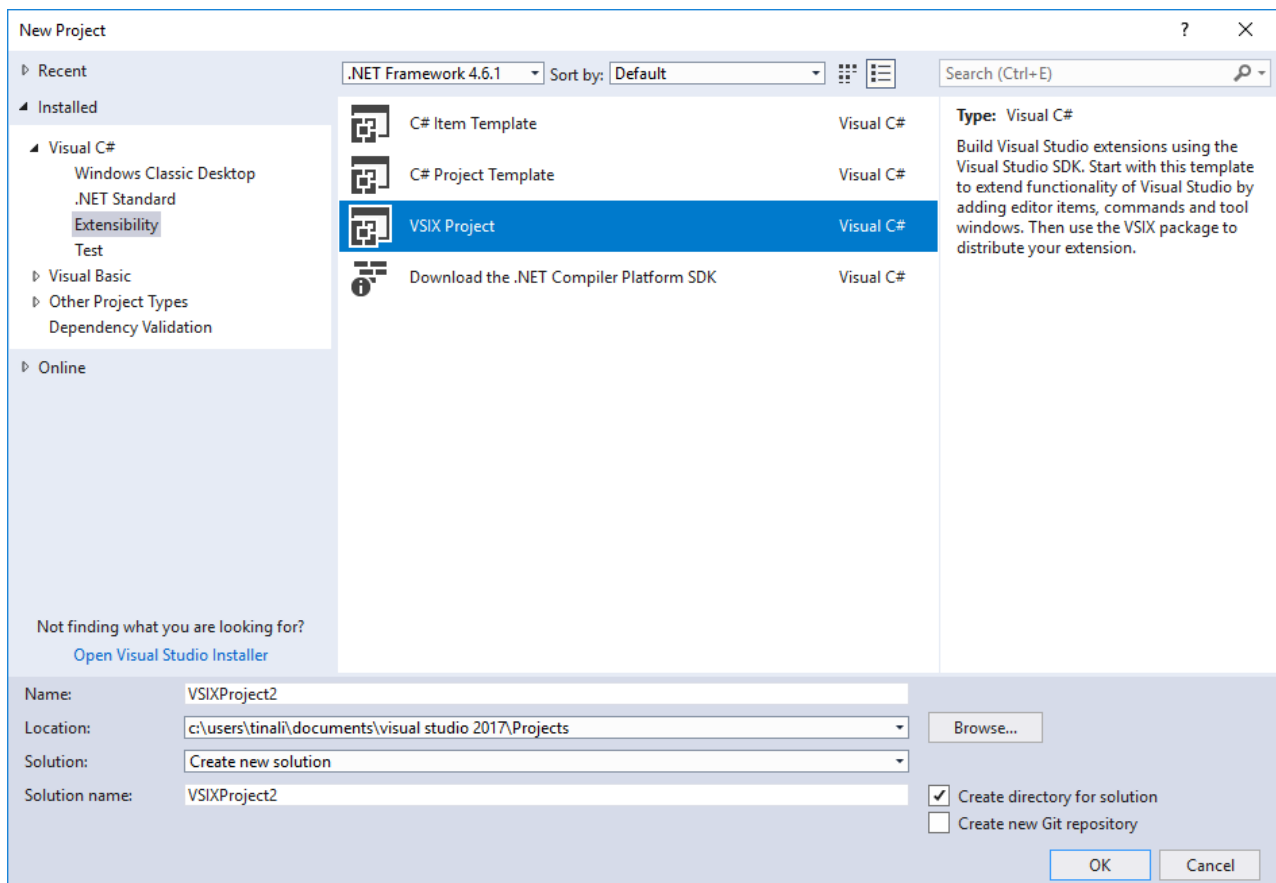
A partir de la versión 15.8, cada vez que se realiza una actualización en Visual Studio, la vista previa VSIX se detecta y quita automáticamente.

2. Actualice la referencia de Nuget a la última versión no preliminar para [paquetes LSP](#).
3. Quite la dependencia a Microsoft Visual Studio Language Server Protocol Preview VSIX en el manifiesto VSIX.
4. Asegúrese de que VSIX especifica Visual Studio 2017 versión 15.8 Preview 3 como el límite inferior para el destino de instalación.
5. Vuelva a compilar e implementar.

Crear un proyecto VSIX

Para crear una extensión de servicio de lenguaje mediante un servidor de lenguaje basado en LSP, primero asegúrese de que tiene la carga de trabajo de desarrollo de extensión de **Visual Studio** instalada para la instancia de VS.

A continuación, cree un nuevo proyecto VSIX navegando a **Archivo > Nuevo proyecto > Proyecto > de visual dextensibilidad > VSIX Proyecto**:



Servidor de idioma e instalación en tiempo de ejecución

De forma predeterminada, las extensiones creadas para admitir servidores de lenguaje basados en LSP en Visual Studio no contienen los propios servidores de lenguaje ni los tiempos de ejecución necesarios para ejecutarlos. Los desarrolladores de extensiones son responsables de distribuir los servidores de lenguaje y los tiempos de ejecución necesarios. Hay varias maneras de hacerlo:

- Los servidores de lenguaje se pueden incrustar en El VSIX como archivos de contenido.
- Cree un MSI para instalar el servidor de idioma y/o los tiempos de ejecución necesarios.
- Proporcione instrucciones sobre Marketplace para informar a los usuarios sobre cómo obtener tiempos de ejecución y servidores de idioma.

Archivos de gramática TextMate

El LSP no incluye especificaciones sobre cómo proporcionar coloración de texto para idiomas. Para proporcionar coloración personalizada para los lenguajes en Visual Studio, los desarrolladores de extensiones pueden usar un archivo de gramática TextMate. Para agregar archivos de tema o gramática TextMate personalizados, siga estos pasos:

1. Crea una carpeta llamada "Grammars" dentro de tu extensión (o puede ser el nombre que elijas).
2. Dentro de la carpeta *Gramáticas*, incluya los `*.tmlanguage*`, `*.plist*`, `*.tmtheme` o `*.json` que desee que proporcionen una coloración personalizada.

TIP

Un archivo `.tmtheme` define cómo se asignan los ámbitos a las clasificaciones de Visual Studio (denominadas claves de color). Para obtener instrucciones, puede hacer referencia al archivo `.tmtheme` global en el directorio `%ProgramFiles(x86)%-Microsoft Visual Studio\<>\<SKU>`.

3. Cree un archivo `.pkgdefy` agregue una línea similar a esta:

```
[$RootKey$\TextMate\Repositories]
"MyLang"="$PackageFolder$\Grammars"
```

4. Haga clic con el botón derecho en los archivos y seleccione **Propiedades**. Cambie la acción **Generar a Contenido** y cambie la propiedad **Incluir en VSIX** a **true**.

Después de completar los pasos anteriores, se agrega una carpeta *Grammars* al directorio *install* del paquete como un origen de repositorio denominado 'MyLang' ('MyLang' es solo un nombre para la desambiguación y puede ser cualquier cadena única). Todas las gramáticas (archivos *.tmlanguage*) y archivos de tema (archivos *.tmtheme*) en este directorio se recogen como potenciales y reemplazan a las gramáticas integradas proporcionadas con TextMate. Si las extensiones declaradas del archivo de gramática coinciden con la extensión del archivo que se está queabriendo, TextMate pisará el paso.

Crear un cliente de lenguaje simple

Interfaz principal - [ILanguageClient](#)

Después de crear el proyecto VSIX, agregue los siguientes paquetes NuGet al proyecto:

- [Microsoft.VisualStudio.LanguageServer.Client](#)

NOTE

Cuando se toma una dependencia del paquete NuGet después de completar los pasos anteriores, los paquetes *Newtonsoft.Json* y *StreamJsonRpc* también se agregan al proyecto. **No actualice estos paquetes a menos que esté seguro de que esas nuevas versiones se instalarán en la versión de Visual Studio** a la que se dirige la extensión. Los ensamblados no se incluirán en su VSIX; en su lugar, se recogerán en el directorio de instalación de Visual Studio. Si hace referencia a una versión más reciente de los ensamblados que a lo que está instalado en el equipo de un usuario, la extensión no funcionará.

A continuación, puede crear una nueva clase que implemente la interfaz [ILanguageClient](#), que es la interfaz principal necesaria para los clientes de lenguaje que se conectan a un servidor de lenguaje basado en LSP.

A continuación se muestra un ejemplo:


```

namespace MockLanguageExtension
{
    [ContentType("bar")]
    [Export(typeof(ILanguageClient))]
    public class BarLanguageClient : ILanguageClient
    {
        public string Name => "Bar Language Extension";

        public IEnumerable<string> ConfigurationSections => null;

        public object InitializationOptions => null;

        public IEnumerable<string> FilesToWatch => null;

        public event AsyncEventHandler<EventArgs> StartAsync;
        public event AsyncEventHandler<EventArgs> StopAsync;

        public async Task<Connection> ActivateAsync(Cancellation token)
        {
            await Task.Yield();

            ProcessStartInfo info = new ProcessStartInfo();
            info.FileName = Path.Combine(Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location),
"Server", @"MockLanguageServer.exe");
            info.Arguments = "bar";
            info.RedirectStandardInput = true;
            info.RedirectStandardOutput = true;
            info.UseShellExecute = false;
            info.CreateNoWindow = true;

            Process process = new Process();
            process.StartInfo = info;

            if (process.Start())
            {
                return new Connection(process.StandardOutput.BaseStream, process.StandardInput.BaseStream);
            }

            return null;
        }

        public async Task OnLoadedAsync()
        {
            await StartAsync.InvokeAsync(this, EventArgs.Empty);
        }

        public Task OnServerInitializeFailedAsync(Exception e)
        {
            return Task.CompletedTask;
        }

        public Task OnServerInitializedAsync()
        {
            return Task.CompletedTask;
        }
    }
}

```

Los métodos principales que deben implementarse son [OnLoadedAsync](#) y [ActivateAsync](#). [OnLoadedAsync](#) se llama cuando Visual Studio ha cargado la extensión y el servidor de idioma está listo para iniciarse. En este método, puede invocar el delegado [StartAsync](#) inmediatamente para indicar que se debe iniciar el servidor de lenguaje, o puede realizar lógica adicional e invocar [StartAsync](#) más adelante. **Para activar el servidor de idioma, debes llamar a StartAsync en algún momento.**

[ActivateAsync](#) es el método finalmente invocado llamando al delegado [StartAsync](#). Contiene la lógica para iniciar el

servidor de lenguaje y establecer conexión con él. Se debe devolver un objeto de conexión que contenga secuencias para escribir en el servidor y leer desde el servidor. Las excepciones que se producen aquí se detectan y se muestran al usuario a través de un mensaje infoBar en Visual Studio.

Activación

Una vez implementada la clase de cliente de lenguaje, deberá definir dos atributos para que defina cómo se cargará en Visual Studio y se activará:

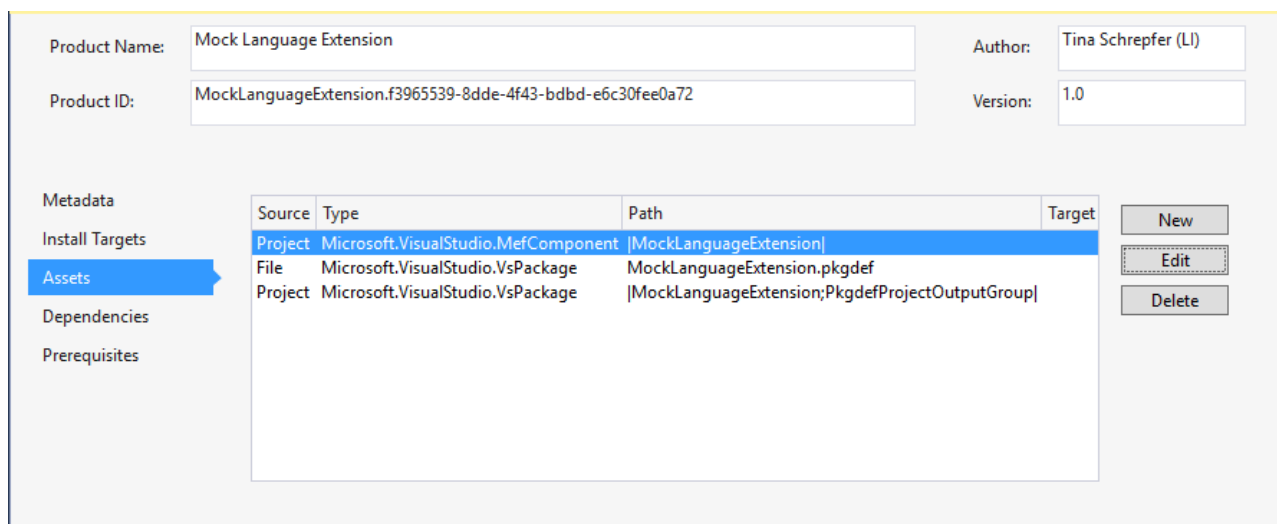
```
[Export(typeof(ILanguageClient))]  
[ContentType("bar")]
```

MEF

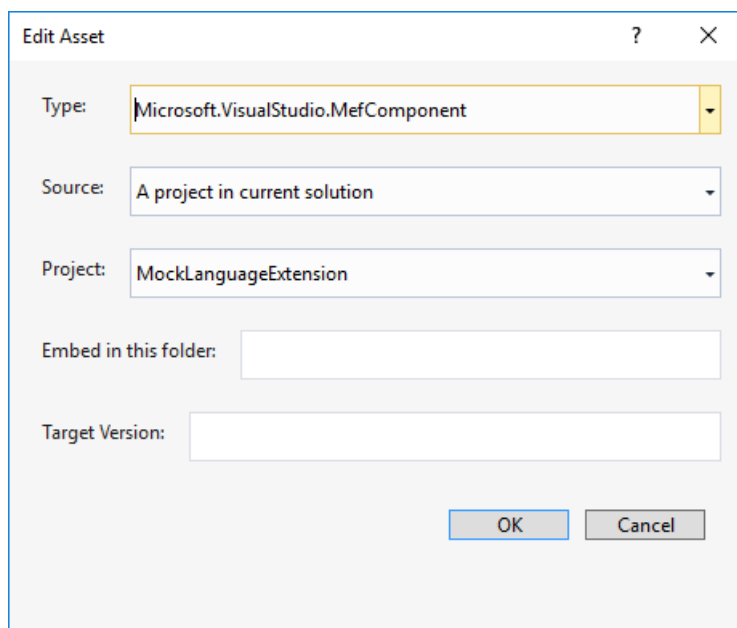
Visual Studio usa **MEF** (Managed Extensibility Framework) para administrar sus puntos de extensibilidad. El **Export** export atributo indica a Visual Studio que esta clase debe recogerse como un punto de extensión y cargar en el momento adecuado.

Para utilizar MEF, también debe definir MEF como un recurso en el manifiesto VSIX.

Abra el diseñador de manifiestos VSIX y vaya a la pestaña **Activos**:



Haga clic en **Nuevo** para crear un nuevo recurso:



- Tipo: Microsoft.VisualStudio.MefComponent

- **Fuente:** Un proyecto en la solución actual
- **Proyecto:** [Su proyecto]

Definición de tipo de contenido

Actualmente, la única manera de cargar la extensión de servidor de lenguaje basada en LSP es por tipo de contenido de archivo. Es decir, al definir la clase de cliente de lenguaje (que implementa [ILanguageClient](#)), deberá definir los tipos de archivos que, cuando se abren, harán que se cargue la extensión. Si no se abre ningún archivo que coincida con el tipo de contenido definido, la extensión no se cargará.

Esto se hace mediante la `ContentTypeDefinition` definición de una o más clases:

```
namespace MockLanguageExtension
{
    public class BarContentDefinition
    {
        [Export]
        [Name("bar")]
        [BaseDefinition(CodeRemoteContentDefinition.CodeRemoteContentTypeName)]
        internal static ContentTypeDefinition BarContentTypeDefinition;

        [Export]
        [FileExtension(".bar")]
        [ContentType("bar")]
        internal static FileExtensionToContentTypeDefinition BarFileExtensionDefinition;
    }
}
```

En el ejemplo anterior, se crea una definición de tipo de contenido para los archivos que terminan en la extensión de archivo `.bar`. La definición de tipo de contenido recibe el nombre "bar" y debe derivar de [CodeRemoteContentTypeName](#).

Después de agregar una definición de tipo de contenido, puede definir cuándo cargar la extensión de cliente de idioma en la clase de cliente de idioma:

```
[ContentType("bar")]
[Export(typeof(ILanguageClient))]
public class BarLanguageClient : ILanguageClient
{
}
```

Agregar compatibilidad con servidores de lenguaje LSP no requiere que implemente su propio sistema de proyecto en Visual Studio. Los clientes pueden abrir un único archivo o una carpeta en Visual Studio para empezar a usar el servicio de lenguaje. De hecho, la compatibilidad con servidores de lenguaje LSP está diseñada para funcionar solo en escenarios de carpetas/archivos abiertos. Si se implementa un sistema de proyecto personalizado, algunas características (como la configuración) no funcionarán.

Características avanzadas

Configuración

La compatibilidad con la configuración personalizada específica del servidor de idioma está disponible, pero todavía está en proceso de mejora. La configuración es específica de lo que admite el servidor de lenguaje y normalmente controla cómo el servidor de idioma emite los datos. Por ejemplo, un servidor de idioma puede tener una configuración para el número máximo de errores notificados. Los autores de extensiones definirían un valor predeterminado, que los usuarios pueden cambiar para proyectos específicos.

Siga estos pasos para agregar compatibilidad con la configuración a la extensión del servicio de lenguaje LSP:

1. Agregue un archivo JSON (por ejemplo, *MockLanguageExtensionSettings.json*) al proyecto que contenga la configuración y sus valores predeterminados. Por ejemplo:

```
{
  "foo.maxNumberOfProblems": -1
}
```

2. Haga clic con el botón derecho en el archivo JSON y seleccione **Propiedades**. Cambie la acción **Generar** a "Contenido" y la propiedad "Incluir en VSIX" a **true**.
3. Implemente `ConfigurationSections` y devuelva la lista de prefijos para la configuración definida en el archivo JSON (en Visual Studio Code, esto se asignaría al nombre de la sección de configuración en `package.json`):

```
public IEnumerable<string> ConfigurationSections
{
    get
    {
        yield return "foo";
    }
}
```

4. Agregue un archivo `.pkgdef` al proyecto (agregue un nuevo archivo de texto y cambie la extensión de archivo a `.pkgdef`). El archivo `pkgdef` debe contener esta información:

```
[$RootKey$\OpenFolder\Settings\VSWorkspaceSettings\[settings-name]]
@="$PackageFolder$\[settings-file-name].json"
```

Sample:

```
[$RootKey$\OpenFolder\Settings\VSWorkspaceSettings\MockLanguageExtension]
@="$PackageFolder$\MockLanguageExtensionSettings.json"
```

5. Haga clic con el botón derecho en el archivo `.pkgdef` y seleccione **Propiedades**. Cambie la acción **Generar** a **Contenido** y la propiedad **Incluir en VSIX** a **true**.
6. Abra el archivo *source.extension.vsixmanifest* y agregue un recurso en la pestaña **Asset**:

The screenshot shows the 'Edit Asset' dialog box with the following details:

- Type:** Microsoft.VisualStudio.VsPackage
- Source:** File on filesystem
- Path:** MockLanguageExtension.pkgdef
- Embed in this folder:** (empty text box)
- Buttons:** Browse..., OK, Cancel

- **Tipo:** Microsoft.VisualStudio.VsPackage

- **Fuente:** Archivo en el sistema de archivos
- **Ruta :**[Ruta de acceso al archivo *.pkgdef*]

Edición por parte del usuario de la configuración de un espacio de trabajo

1. El usuario abre un espacio de trabajo que contiene los archivos que posee el servidor.
2. El usuario agrega un archivo en la carpeta *.vs* denominado *VSWorkspaceSettings.json*.
3. El usuario agrega una línea al archivo *VSWorkspaceSettings.json* para una configuración que proporciona el servidor. Por ejemplo:

```
{
    "foo.maxNumberOfProblems": 10
}
```

Habilitar el seguimiento de diagnósticos

El seguimiento de diagnósticos se puede habilitar para generar todos los mensajes entre el cliente y el servidor, lo que puede ser útil al depurar problemas. Para habilitar el seguimiento de diagnóstico, haga lo siguiente:

1. Abra o cree el archivo de configuración del área de trabajo *VSWorkspaceSettings.json* (consulte "Edición de usuario de la configuración de un área de trabajo").
2. Agregue la siguiente línea en el archivo json de configuración:

```
{
    "foo.trace.server": "Off"
}
```

Hay tres valores posibles para la verbosidad de seguimiento:

- "Off": el rastreo se apagó por completo
- "Messages": seguimiento activado, pero solo se rastrean el nombre del método y el identificador de respuesta.
- "Verbose": seguimiento activado; se rastrea todo el mensaje rpc.

Cuando se activa el seguimiento, el contenido se escribe en un archivo en el directorio *%temp%-VisualStudio-LSP*. El registro sigue el formato de nomenclatura *[LanguageClientName]-[Datetime Stamp].log*. Actualmente, el seguimiento solo se puede habilitar para escenarios de carpetas abiertas. Abrir un único archivo para activar un servidor de lenguaje no tiene compatibilidad con el seguimiento de diagnósticos.

Mensajes personalizados

Existen API para facilitar el envío de mensajes y la recepción de mensajes desde el servidor de idioma que no forman parte del protocolo de servidor de idioma estándar. Para controlar los mensajes personalizados, implemente la interfaz *ILanguageClientCustomMessage* en la clase de cliente de lenguaje. La biblioteca *VS-StreamJsonRpc* se utiliza para transmitir mensajes personalizados entre el cliente de lenguaje y el servidor de lenguaje. Dado que la extensión de cliente de lenguaje LSP es igual que cualquier otra extensión de Visual Studio, puede decidir agregar características adicionales (que no son compatibles con el LSP) a Visual Studio (mediante otras API de Visual Studio) en la extensión a través de mensajes personalizados.

Recibir mensajes personalizados

Para recibir mensajes personalizados desde el servidor de lenguaje, implemente la propiedad *CustomMessageTarget* en *ILanguageClientCustomMessage* y devuelva un objeto que sepa cómo controlar los mensajes personalizados. Ejemplo a continuación:

```

internal class MockCustomLanguageClient : MockLanguageClient, ILanguageClientCustomMessage
{
    private JsonRpc customMessageRpc;

    public MockCustomLanguageClient() : base()
    {
        CustomMessageTarget = new CustomTarget();
    }

    public object CustomMessageTarget
    {
        get;
        set;
    }

    public class CustomTarget
    {
        public void OnCustomNotification(JToken arg)
        {
            // Provide logic on what happens OnCustomNotification is called from the language server
        }

        public string OnCustomRequest(string test)
        {
            // Provide logic on what happens OnCustomRequest is called from the language server
        }
    }
}

```

Enviar mensajes personalizados

Para enviar mensajes personalizados al servidor de lenguaje, implemente el método [AttachForCustomMessageAsync](#) en [ILanguageClientCustomMessage](#). Este método se invoca cuando se inicia el servidor de lenguaje y está listo para recibir mensajes. Un objeto [JsonRpc](#) se pasa como parámetro, que puede mantener para enviar mensajes al servidor de lenguaje mediante las API de [VS-StreamJsonRpc](#). Ejemplo a continuación:

```

internal class MockCustomLanguageClient : MockLanguageClient, ILanguageClientCustomMessage
{
    private JsonRpc customMessageRpc;

    public MockCustomLanguageClient() : base()
    {
        CustomMessageTarget = new CustomTarget();
    }

    public async Task AttachForCustomMessageAsync(JsonRpc rpc)
    {
        await Task.Yield();

        this.customMessageRpc = rpc;
    }

    public async Task SendServerCustomNotification(object arg)
    {
        await this.customMessageRpc.NotifyWithParameterObjectAsync("OnCustomNotification", arg);
    }

    public async Task<string> SendServerCustomMessage(string test)
    {
        return await this.customMessageRpc.InvokeAsync<string>("OnCustomRequest", test);
    }
}

```

Capa media

A veces, un desarrollador de extensiones puede interceptar los mensajes LSP enviados y recibidos desde el servidor de lenguaje. Por ejemplo, un desarrollador de extensiones puede modificar el parámetro de mensaje enviado para un mensaje LSP determinado o modificar los resultados devueltos desde el servidor de idioma para una característica LSP (por ejemplo, terminaciones). Cuando sea necesario, los desarrolladores de extensiones pueden usar la API `MiddleLayer` para interceptar mensajes LSP.

Cada mensaje LSP tiene su propia interfaz de capa media para la interceptación. Para interceptar un mensaje determinado, cree una clase que implemente la interfaz de capa intermedia para ese mensaje. A continuación, implemente la interfaz `ILanguageClientCustomMessage` en la clase de cliente de lenguaje y devuelva una instancia del objeto en la propiedad `MiddleLayer`. Ejemplo a continuación:

```
public class MockLanguageClient: ILanguageClient, ILanguageClientCustomMessage
{
    public object MiddleLayer => MiddleLayerProvider.Instance;

    private class MiddleLayerProvider : ILanguageClientWorkspaceSymbolProvider
    {
        internal readonly static MiddleLayerProvider Instance = new MiddleLayerProvider();

        private MiddleLayerProvider()
        {
        }

        public async Task<SymbolInformation[]> RequestWorkspaceSymbols(WorkspaceSymbolParams param,
Func<WorkspaceSymbolParams, Task<SymbolInformation[]>> sendRequest)
        {
            // Send along the request as given
            SymbolInformation[] symbols = await sendRequest(param);

            // Only return symbols that are "files"
            return symbols.Where(sym => string.Equals(new Uri(sym.Location.Uri).Scheme, "file",
StringComparison.OrdinalIgnoreCase)).ToArray();
        }
    }
}
```

La entidad de capa media todavía está en desarrollo y aún no está completa.

Extensión del servidor de lenguaje LSP de ejemplo

Para ver el código fuente de una extensión de ejemplo mediante la API de cliente LSP en Visual Studio, vea `VSSDK-Extensibility-Samples` [LSP sample](#).

Preguntas más frecuentes

Me gustaría crear un sistema de proyecto personalizado para complementar mi servidor de lenguaje LSP para proporcionar compatibilidad con características más ricas en Visual Studio, ¿cómo puedo hacerlo?

La compatibilidad con servidores de lenguaje basados en LSP en Visual Studio se basa en la característica de [carpeta abierta](#) y está diseñada para no requerir un sistema de proyecto personalizado. Puede crear su propio sistema de proyecto personalizado siguiendo las instrucciones [aquí](#), pero algunas características, como la configuración, pueden no funcionar. La lógica de inicialización predeterminada para los servidores de lenguaje LSP es pasar la ubicación de la carpeta raíz de la carpeta que se está abriendo actualmente, por lo que si utiliza un sistema de proyecto personalizado, es posible que deba proporcionar lógica personalizada durante la inicialización para asegurarse de que el servidor de idioma se puede iniciar correctamente.

¿Cómo agrego compatibilidad con el depurador?

Proporcionaremos soporte para el [protocolo de depuración común](#) en una versión futura.

Si ya hay un servicio de lenguaje compatible con VS instalado (por ejemplo, JavaScript), ¿puedo instalar una extensión de servidor de lenguaje LSP que ofrezca características adicionales (como linting)?

Sí, pero no todas las características funcionarán correctamente. El objetivo final de las extensiones de servidor de lenguaje LSP es habilitar servicios de lenguaje no admitidos de forma nativa por Visual Studio. Puede crear extensiones que ofrezcan compatibilidad adicional mediante servidores de lenguaje LSP, pero algunas características (como IntelliSense) no serán una experiencia fluida. En general, se recomienda utilizar extensiones de servidor de lenguaje LSP para proporcionar nuevas experiencias de idioma, no para extender las existentes.

¿Dónde publico mi servidor de lenguaje LSP VSIX completado?

Consulte las instrucciones de Marketplace [aquí](#).

Vea también

- [Agregar compatibilidad con otros lenguajes en el editor de Visual Studio](#)

Extensibilidad de carpeta abierta

02/12/2019 • 5 minutes to read • [Edit Online](#)

El [Abrir carpeta](#) característica permite a los usuarios abrir cualquier código base en Visual Studio sin necesidad de archivos de proyecto o solución. Abrir carpeta proporciona que los funciones que los usuarios esperan de Visual Studio, como:

- Búsqueda y la integración del explorador de soluciones
- Coloración de editor
- Ir a la navegación
- Buscar en la búsqueda de archivos

Cuando se usa con las cargas de trabajo como en el caso de desarrollo .NET y C++, también se muestra:

- Rich Intellisense
- Funcionalidad específica del lenguaje

Con Abrir carpeta, los autores de extensiones pueden crear características enriquecidas para cualquier lenguaje. Existen API para admitir la creación, depuración y búsqueda de símbolos para cualquier archivo de un usuario del código base. Los extensores actuales pueden actualizar sus características de Visual Studio existentes para entender el código sin el respaldo de una solución o proyectos.

Una API sin sistemas del proyecto

Históricamente, Visual Studio solo entiende los archivos en una solución y sus proyectos con los sistemas del proyecto. Un sistema de proyectos es responsable de las interacciones de usuario y la funcionalidad de un proyecto cargado. Entienda lo que su proyecto de archivos contiene la representación visual del contenido del proyecto, las dependencias en otros proyectos, y el archivo de proyecto de modificación de subyacente. Es a través de estas jerarquías y capacidades que otros componentes funcionan en nombre del usuario. No todos los códigos base también se representan en una estructura de proyecto y solución. Lenguajes de scripting y de código abierto escrito en C++ para Linux son buenos ejemplos. Con Abrir carpeta, Visual Studio proporciona a los usuarios una nueva forma de interactuar con su código fuente.

La API de carpeta abiertas están bajo el `Microsoft.VisualStudio.Workspace.*` espacio de nombres y están disponibles para que los extensores producir y consumir datos o acciones en torno a los archivos de carpeta abierta. Las extensiones pueden usar estas API para proporcionar la funcionalidad para muchas áreas, como:

- [Las áreas de trabajo](#) : la extensibilidad de Abrir carpeta de punto de inicio es el área de trabajo y sus API.
- [Contextos y las acciones de archivo](#) -proporcionada a través de los contextos de archivo de inteligencia de código específico de archivos.
- [Indexación](#) : recopilar y conservar los datos sobre las áreas de trabajo de la carpeta abierta.
- [Servicios de lenguaje](#) -integrar servicios de lenguaje en las áreas de trabajo de la carpeta abierta.
- [Compilar](#) -crear compatibilidad para las áreas de trabajo de la carpeta abierta.

Las características que usan los siguientes tipos debe adoptar nuevas API para admitir la carpeta abierta:

- `IVsHierarchy`
- `IVsProject`
- `DTE`

Comentarios, comentarios y problemas

Abra la carpeta y el `Microsoft.VisualStudio.Workspace.*` API están en desarrollo activo. Si ve un comportamiento inesperado, a continuación, consulte los problemas conocidos de la versión de interés. [Comunidad de desarrolladores](#) es el lugar recomendado para votar y crear cualquier problema. Para cada tipo de comentarios, se recomienda encarecidamente obtener una descripción detallada del problema. Incluir la versión de Visual Studio que está desarrollando, las API que está usando (lo que ha implementado y está interactuando con), el resultado esperado y el resultado real. Si es posible, incluir un volcado del proceso devenv.exe. Utilice el problema de GitHub de seguimiento para enviar comentarios sobre este y documentación relacionada.

Pasos siguientes

- [Las áreas de trabajo](#) -Obtenga información sobre la API de área de trabajo de carpeta abierta.

Áreas de trabajo

02/12/2019 • 12 minutes to read • [Edit Online](#)

Un área de trabajo es la forma en que Visual Studio representa una colección de archivos en [Abrir carpeta](#), y se representa mediante el [IWorkspace](#) tipo. Por sí mismo, el área de trabajo no entiende el contenido o las características relacionadas con los archivos dentro de la carpeta. En su lugar, proporciona un conjunto de API para las características y extensiones para producir y consumir datos que otras personas pueden actuar en general. Los productores se crean a través de la [Managed Extensibility Framework](#) (MEF) utilizando varios atributos de exportación.

Servicios y proveedores de área de trabajo

Servicios y proveedores de área de trabajo proporcionan los datos y funciones para reaccionar ante el contenido de un área de trabajo. Es posible que proporcionan información contextual del archivo, símbolos de archivos de origen, o crear una funcionalidad.

Use ambos conceptos un [patrón factory](#) y se importa a través de MEF, el área de trabajo. Implementan todos los atributos de exportación `IProviderMetadataBase` o `IWorkspaceServiceFactoryMetadata`, pero hay tipos concretos que deben usar las extensiones para los tipos exportados.

Una diferencia entre los proveedores y servicios es su relación con el área de trabajo. Un área de trabajo puede tener muchos proveedores de un tipo determinado, pero solo un servicio de un tipo determinado se crea por área de trabajo. Por ejemplo, un área de trabajo tiene muchos proveedores de escáner de archivos, pero el área de trabajo tiene solo un servicio de indización por área de trabajo.

Otra diferencia clave es el consumo de datos de proveedores y servicios. El área de trabajo es el punto de entrada para obtener los datos de los proveedores por dos motivos. En primer lugar, los proveedores normalmente tienen un reducido conjunto de datos crean. Los datos podrían ser los símbolos para un C# del origen de archivo o crear contextos de archivo para un *CMakeLists.txt* archivo. El área de trabajo hará coincidir la solicitud del consumidor a los proveedores cuyos metadatos se alinean con la solicitud. En segundo lugar, se permiten algunos escenarios para muchos proveedores para contribuir a una solicitud, mientras que otros escenarios de uso del proveedor con prioridad más alta.

En cambio, las extensiones de pueden obtener instancias de e interactuar directamente con los servicios de área de trabajo. Métodos de extensión en `IWorkspace` están disponibles para los servicios proporcionados por Visual Studio, como [GetFileWatcherService](#). La extensión puede ofrecer un servicio del área de trabajo de componentes dentro de la extensión o de otras extensiones que pueden consumir. Los consumidores deberían usar [GetServiceAsync](#) o un método de extensión se proporciona en el `IWorkspace` tipo.

WARNING

No crear servicios que entran en conflicto con Visual Studio. Puede provocar problemas inesperados.

Eliminación de cierre del área de trabajo

En el cierre de un área de trabajo, podrían necesitar extensores dispose pero asíncrona se llama a código. El [IAsyncDisposable](#) interfaz está disponible para facilitar la escritura de este código sencillo.

Tipos relacionados

- [IWorkspace](#) es la entidad central para un área de trabajo abierto, como una carpeta abierta.
- [IWorkspaceProviderFactory<T>](#) crea un proveedor por área de trabajo crea una instancia.
- [IWorkspaceServiceFactory](#) crea un servicio por área de trabajo crea una instancia.
- [IAsyncDisposable](#) se debe implementar en los proveedores y servicios que necesita para ejecutar código asíncrono durante la eliminación.
- [WorkspaceServiceHelper](#) Proporciona métodos auxiliares para tener acceso a servicios conocidos o servicios arbitrarios.

Configuración de área de trabajo

Áreas de trabajo tienen un [IWorkspaceSettingsManager](#) servicio simple pero eficaz control sobre un área de trabajo. Para obtener una introducción básica de configuración, consulte [personalización de compilación y depuración tareas](#).

Configuración para la mayoría `SettingsType` tipos son *.json* archivos, tales como *VSWorkspaceSettings.json* y *tasks.vs.json*.

La eficacia de la configuración de área de trabajo se centra en "ámbitos", que son simplemente las rutas de acceso del área de trabajo. Cuando un consumidor llama a [GetAggregatedSettings](#), se agregan todos los ámbitos que incluyen la ruta de acceso solicitada y el tipo de configuración. Prioridad de la agregación de ámbito es como sigue:

1. "Configuración local", que normalmente es la raíz del área de trabajo `.vs` directory.
2. La ruta de acceso solicitada propio.
3. El directorio principal de la ruta de acceso solicitado.
4. Todos los primarios más directorios hasta e incluida la raíz del área de trabajo.
5. "Global settings", que se encuentra en un directorio de usuario.

El resultado es una instancia de [IWorkspaceSettings](#). Este objeto contiene la configuración de un tipo determinado y se puede consultar para establecer los nombres de clave almacenado como `string`. El [GetProperty](#) métodos y [WorkspaceSettingsExtensions](#) métodos de extensión espera que el llamador saber el tipo del valor de configuración que se solicita. Como la mayoría de los archivos de configuración se almacenan como *.json* archivos, usará muchas invocaciones `string`, `bool`, `int` y las matrices de esos tipos. También se admiten los tipos de objeto. En esos casos, puede usar `IWorkspaceSettings` como el argumento de tipo. Por ejemplo:

```
{
  "intValue": 1,
  "stringValue": "s",
  "boolValue": true,
  "stringArray": [
    "s1",
    "s2"
  ],
  "nestedIWorkspaceSettings": {
    "nestedString": "ns"
  }
}
```

Suponiendo que esta configuración se encontraban en un usuario *VSWorkspaceSettings.json*, pueden tener acceso a los datos como:

```

using System.Collections.Generic;
using Microsoft.VisualStudio.Workspace;
using Microsoft.VisualStudio.Workspace.Settings;

private static void ReadSettings(IWorkspace workspace)
{
    IWorkspaceSettingsManager settingsManager = workspace.GetSettingsManager();
    IWorkspaceSettings settings = settingsManager.GetAggregatedSettings(SettingsTypes.Generic);

    // result == WorkspaceSettingsResult.Success
    WorkspaceSettingsResult result = settings.GetProperty("intValue", out int intValue);
    result = settings.GetProperty("stringValue", out string stringValue);
    result = settings.GetProperty("boolValue", out bool boolValue);
    result = settings.GetProperty("stringArray", out string[] stringArray);
    result = settings.GetProperty("nestedIWorkspaceSettings", out IWorkspaceSettings
nestedIWorkspaceSettings);
    result = nestedIWorkspaceSettings.GetProperty("nestedString", out string nestedString);

    // Extension method alternative using default values.
    int intValueOrDefault = settings.Property("intValue", /* default */ 42);

    // Missing key. result == WorkspaceSettingsResult.Undefined
    result = settings.GetProperty("missing", out string missing);

    // Wrong type for a key. result == WorkspaceSettingsResult.Error
    result = settings.GetProperty("intValue", out IWorkspaceSettings notSettings);

    // Special ability to union "stringArray" across all scopes.
    IEnumerable<string> allStringArray = settings.UnionPropertyArray<string>("stringArray");
}

```

NOTE

Esta configuración de las API no está relacionadas con las API disponibles en el `Microsoft.VisualStudio.Settings` espacio de nombres. Configuración de área de trabajo es independientes del host y usa archivos de configuración específica del área de trabajo o los proveedores de configuración dinámica.

Proporcionar configuración dinámica

Pueden proporcionar extensiones [IWorkspaceSettingsProviders](#). Estos proveedores en memoria permiten las extensiones agregar la configuración o reemplazar otras.

Exportar un `IWorkspaceSettingsProvider` es diferente de otros proveedores de área de trabajo. El generador no es `IWorkspaceProviderFactory` y no hay ningún tipo de atributo especial. En su lugar, implemente [IWorkspaceSettingsProviderFactory](#) y usar `[Export(typeof(IWorkspaceSettingsProviderFactory))]`.

```
// Common workspace provider factory pattern
[ExportFeatureProvider(some, args, to, export)]
internal class MyProviderFactory : IWorkspaceProviderFactory<IFeatureProvider>
{
    IFeatureProvider CreateProvider(IWorkspace workspace) => new Provider(workspace);
}

// IWorkspaceSettingsProvider pattern
[Export(typeof(IWorkspaceSettingsProviderFactory))]
internal class MySettingsProviderFactory : IWorkspaceSettingsProviderFactory
{
    // 100 is typically the value used by built-in settings providers. Lower value is higher priority.
    int Priority => 100;

    IWorkspaceSettingsProvider CreateSettingsProvider(IWorkspace workspace) => new
    MySettingsProvider(workspace);
}
```

TIP

Al implementar los métodos que devuelven `IWorkspaceSettingsSource` (como `IWorkspaceSettingsProvider.GetSingleSettings`), devolver una instancia de `IWorkspaceSettings` lugar `IWorkspaceSettingsSource`. `IWorkspaceSettings` proporciona más información que puede ser útil durante algunos agregaciones de configuración.

API relacionadas con la configuración

- [IWorkspaceSettingsManager](#) lee y agrega la configuración de área de trabajo.
- [GetSettingsManager](#) Obtiene el `IWorkspaceSettingsManager` para un área de trabajo.
- [GetAggregatedSettings](#) Obtiene la configuración de un determinado ámbito agregado a través de todos los ámbitos que se superponen.
- [IWorkspaceSettings](#) contiene la configuración para un ámbito determinado.

Área de trabajo sugerido recomendados

- Devolver los objetos de `IWorkspaceProviderFactory.CreateProvider` o API similares que recordar sus `Workspace` contexto cuando se creó. Interfaces de los proveedores se escriben espera que este objeto se mantiene en la creación.
- Guardar cachés específicas del área de trabajo o los valores de la ruta de acceso de "Configuración Local" del área de trabajo. Crear una ruta de acceso para el archivo mediante `Microsoft.VisualStudio.Workspace.WorkspaceHelper.MakeRootedUnderWorkingFolder` en Visual Studio 2017 versión 15.6 o posterior. Las versiones anteriores a la versión 15.6, use el siguiente fragmento de código:

```
using System.IO;
using Microsoft.VisualStudio.Workspace;
using Microsoft.VisualStudio.Workspace.Settings;

private static string MakeRootedUnderWorkingFolder(IWorkspace workspace, string relativePath)
{
    string workingFolder =
    workspace.GetSettingsManager().GetAggregatedSettings(SettingsTypes.WorkspaceControlSettings).Property<string>
    ("WorkingFolder");
    return Path.Combine(workingFolder, relativePath);
}
```

Eventos de la solución y cargar paquete automáticamente

Pueden implementar paquetes cargados `IVsSolutionEvents7` e invocar `IVsSolution.AdviseSolutionEvents`. Incluye eventos de apertura y cierre de una carpeta en Visual Studio.

Un contexto de interfaz de usuario puede utilizarse para cargar automáticamente el paquete. El valor es `4646B819-1AE0-4E79-97F4-8A8176FDD664`.

Solución de problemas

No se cargó correctamente el paquete `SourceExplorerPackage`

Extensibilidad de área de trabajo es principalmente basado en MEF y errores de composición hará que el paquete que hospeda la carpeta abierta para que no se pudo cargar. Por ejemplo, si una extensión exporta un tipo con

`ExportFileContextProviderAttribute`, pero sólo implementa el tipo

`IWorkspaceProviderFactory<IFileContextActionProvider>`, se producirá un error al intentar abrir una carpeta en

Visual Studio.

Detalles del error pueden encontrarse en

`%LOCALAPPDATA%\Microsoft\VisualStudio\15.0_id\ComponentModel\Cache\Microsoft.VisualStudio.Default.err.`

Resuelva los errores para los tipos implementados por la extensión.

Detalles del error pueden encontrarse en

`%LOCALAPPDATA%\Microsoft\VisualStudio\16.0_id\ComponentModel\Cache\Microsoft.VisualStudio.Default.err.`

Resuelva los errores para los tipos implementados por la extensión.

Pasos siguientes

- [Contextos de archivo](#) -capturadores de contexto de archivo proporcionan inteligencia de código para las áreas de trabajo de la carpeta abierta.
- [Indexación](#) -área de trabajo de indexación recopila y se conserva la información sobre el área de trabajo.

Contextos de archivo del área de trabajo

02/12/2019 • 8 minutes to read • [Edit Online](#)

Toda la información en [Abrir carpeta](#) las áreas de trabajo producidos por "archivo capturadores de contexto" que implementan el [IFileContextProvider](#) interfaz. Estas extensiones pueden buscar patrones en carpetas o archivos, leer los archivos de MSBuild y archivos MAKE, detectan las dependencias del paquete, etc. con el fin de acumular los conocimientos que necesitan para definir un contexto de archivo. Un contexto de archivo por sí solo no realiza ninguna acción, pero en su lugar, proporciona datos que otra extensión, a continuación, puede actuar en.

Cada [FileContext](#) tiene un [Guid](#) asociado que identifica el tipo de datos se lleva. Un área de trabajo usa esto [Guid](#) más adelante para que coincidan con las extensiones que consumen los datos a través de la [Context](#) propiedad. Un proveedor de contexto de archivo se exporta con metadatos que identifica qué contexto de archivo [Guid](#) s puede generar datos de.

Una vez definido, un contexto de archivo puede asociarse con cualquier número de archivos o carpetas en el área de trabajo. Un determinado archivo o carpeta puede asociarse con cualquier número de contextos de archivo. Es una relación varios a varios.

Los escenarios más comunes para los contextos de archivo están relacionados con la compilación, depuración y servicios de lenguaje. Para obtener más información, consulte los otros temas relacionados con estos escenarios.

Ciclo de vida del contexto de archivo

Los ciclos de vida para un [FileContext](#) son no deterministas. En cualquier momento, un componente de forma asíncrona puede solicitar para algún conjunto de tipos de contexto. Los proveedores que admiten un subconjunto de los tipos de contexto de solicitud que se va a consultar. El [IWorkspace](#) instancia actúa como intermediario entre los consumidores y proveedores a través de la [GetFileContextsAsync](#) método. Los consumidores pueden solicitar un contexto y realizar alguna acción a corto plazo en función del contexto, mientras que otros usuarios pueden solicitar un contexto y mantener una referencia de larga duración.

Es posible que realizan los cambios en los archivos que provocan un contexto de archivo a quedar obsoleto. El proveedor puede desencadenar un evento en el [FileContext](#) para notificar a los consumidores de las actualizaciones. Por ejemplo, si se proporciona un contexto de compilación para algunos archivos, pero ese contexto invalida un cambio en el disco, el productor original puede invocar el evento. Los consumidores que siguen haciendo referencia a ella [FileContext](#) , a continuación, puede volver a consultar un nuevo [FileContext](#) .

NOTE

No hay ningún modelo de inserción a los consumidores. Los consumidores no recibirá notificación de un proveedor nuevo [FileContext](#) después de su solicitud.

Cálculos del contexto de archivos costosas

Contenido del archivo de lectura de disco puede ser costoso, especialmente cuando se necesita resolver la relación entre los archivos de un proveedor. Por ejemplo, puede consultar un proveedor de contexto del archivo de algún archivo de código fuente, pero el contexto del archivo es dependiente de los metadatos de un archivo de proyecto. Analizar el archivo de proyecto o evaluarlo con MSBuild es costosa. En su lugar, puede exportar la extensión de un [IFileScanner](#) crear [FileDataValue](#) datos durante la indización de área de trabajo. Ahora cuando se le pida para contextos de archivo, el [IFileContextProvider](#) puede consultar rápidamente para los datos indizados. Para obtener más información sobre la indexación, consulte el [área de trabajo de indexación](#) tema.

WARNING

Tenga cuidado de otras formas de su `FileContext` podría ser costoso calcular. Algunos tipos de interacción de la interfaz de usuario son sincrónicos y se basan en un gran volumen de `FileContext` solicitudes. Algunos ejemplos son abrir una pestaña del editor y abrir un el **Explorador de soluciones** menú contextual. Estas acciones hacer que el contexto de compilación muchas solicitudes de tipo.

API relacionadas con el contexto de archivo

- `FileContext` contiene los datos y metadatos.
- `IFileContextProvider` y `IFileContextProvider<T>` crear el contexto del archivo.
- `ExportFileContextProviderAttribute` exporta un proveedor de contexto de archivo.
- `GetFileContextsAsync` se usa para los consumidores para obtener los contextos de archivo.
- `BuildContextTypes` define los tipos de contexto de compilación que consumirá la carpeta abierta.

Acciones de contexto de archivo

Mientras un `FileContext` sí es simplemente datos acerca de algunos archivos, un `IFileContextAction` es una manera de express y actuar sobre esos datos. `IFileContextAction` es flexible en su uso. Dos de los casos más comunes son compilar y depurar.

Informes de progreso

El `ExecuteAsync` se pasa al método `IProgress<IFileContextActionProgressUpdate>`, pero el argumento no debe usarse como ese tipo. `IFileContextActionProgressUpdate` es una interfaz vacía e invocar `IProgress<IFileContextActionProgressUpdate>.Report(IFileContextActionProgressUpdate)` podría producir `NotSupportedException`. En su lugar, el `IFileContextAction` debe convertir el argumento a otro tipo según sea necesario para el escenario.

Para obtener información sobre los tipos proporcionados por Visual Studio, consulte la documentación del escenario respectivo.

API relacionadas con la acción contextual de archivo

- `IFileContextAction` algunos comportamientos según se ejecuta un `FileContext`.
- `IFileContextActionProvider` crea instancias de `IFileContextAction`.
- `ExportFileContextActionProviderAttribute` exporta el tipo `IWorkspaceProviderFactory<IFileContextActionProvider>`.

Inspección de archivos

Un área de trabajo escucha las notificaciones de cambio y proporciona el `IFileWatcherService` a través de `GetFileWatcherService`. Archivos que coinciden con la configuración de "ExcludedItems" no producirá los eventos de notificación de archivos. Un umbral entre los eventos se usa para la reducción duplicado y simplificación de la notificación. Cuando necesite responder a un cambio de archivo, debe suscribirse a este servicio.

TIP

Un área de trabajo [servicio de indización](#) se suscribe a eventos de archivo de forma predeterminada. Archivo adiciones y modificaciones provocará relevantes `IFileScanner` eventos s que se invocará para los nuevos datos para ese archivo. Las eliminaciones de archivos quitarán los datos indizados. No es necesario suscribirse su `IFileScanner` para el servicio de Monitor de archivos.

Pasos siguientes

- [Indexación](#) -área de trabajo de indexación recopila y se conserva la información sobre el área de trabajo.
- [Las áreas de trabajo](#) -revisar los conceptos de área de trabajo y de almacenamiento de configuración.

Área de trabajo de indexación

02/12/2019 • 4 minutes to read • [Edit Online](#)

En una solución, los sistemas de proyectos son responsables de proporcionar funcionalidad de compilación, depuración, **GoTo** buscar símbolos y mucho más. Sistemas del proyecto pueden hacer este trabajo, porque entienden las capacidades de archivos dentro de un proyecto y la relación. Un [Abrir carpeta](#) área de trabajo necesita la misma información para proporcionar características del IDE enriquecidas también. La recopilación y el almacenamiento persistente de los datos es un proceso llamado el área de trabajo de indexación. Estos datos indizados pueden consultarse a través de un conjunto de API asincrónicas. Los extensores pueden participar en el proceso de indización proporcionando [IFileScanners](#) que sepa cómo controlar ciertos tipos de archivos.

Tipos de datos indizados

Hay tres tipos de datos que se indizan. Tenga en cuenta el tipo esperado de los escáneres de archivo diferente del tipo deserializado desde el índice.

DATOS	TIPO DE ESCÁNER DE ARCHIVO	TIPO DE RESULTADO DE CONSULTA DE ÍNDICE	TIPOS RELACIONADOS
Referencias	FileReferenceInfo	FileReferenceResult	FileReferenceInfoType
Símbolos	SymbolDefinition	SymbolDefinitionSearchResult	ISymbolService debe usarse en lugar de IIndexWorkspaceService para las consultas
Valores de datos	FileDataValue	FileDataResult<T>	

Consultar los datos indizados

Hay dos tipos asincrónicos disponibles para acceder a los datos persistentes. La primera es a través de [IIndexWorkspaceData](#). Proporciona acceso básico a un único archivo [FileReferenceResult](#) y [FileDataResult](#) datos y lo almacena en caché de resultados. El segundo es el [IIndexWorkspaceService](#) que no usa el almacenamiento en caché, pero permite más capacidades de consulta.

```
using Microsoft.VisualStudio.Workspace;
using Microsoft.VisualStudio.Workspace.Indexing;

private static IIndexWorkspaceData GetCachedIndexedData(IWorkspace workspace)
{
    // Gets access to indexed data wrapped in a cache.
    return workspace?.GetIndexWorkspaceDataService()?.CreateIndexWorkspaceData();
}

private static IIndexWorkspaceService GetDirectIndexedData(IWorkspace workspace)
{
    // Gets direct access to indexed data.
    // Can also be casted to IIndexWorkspaceService2.
    return workspace?.GetIndexWorkspaceService();
}
```

Participa en la indización

Área de trabajo de indexación aproximadamente sigue a la siguiente secuencia:

1. Detección y la persistencia de las entidades del sistema de archivos en el área de trabajo (solo en el análisis inicial de apertura).
2. Por cada archivo, el proveedor que coincida con la prioridad más alta se le pide que busque `FileReferenceInfo` S.
3. Por cada archivo, el proveedor que coincida con la prioridad más alta se le pide que busque `SymbolDefinition` S.
4. Por cada archivo, se le pedirán todos los proveedores `FileDataValue` S.

Las extensiones pueden exportar un escáner implementando `IWorkspaceProviderFactory<IFileScanner>` y exportar el tipo con `ExportFileScannerAttribute`. El `SupportedTypes` argumento de atributo debe ser uno o más valores de `FileScannerTypeConstants`. Para un escáner de ejemplo, consulte el SDK de VS [ejemplo](#).

WARNING

No se exporta un escáner de archivo que admite el `FileScannerTypeConstants.FileScannerContentType` tipo. Se utiliza Microsoft solamente para fines internos.

En situaciones avanzadas, una extensión dinámicamente puede admitir un conjunto arbitrario de tipos de archivo. En lugar de exportar MEF `IWorkspaceProviderFactory<IFileScanner>`, puede exportar una extensión `IWorkspaceProviderFactory<IFileScannerProvider>`. Cuando se inicia la indización, este tipo de generador se importados, creación de instancias y tienen su `GetSymbolScannersAsync` método invocado. `IFileScanner` las instancias que admiten un valor comprendido entre `FileScannerTpeConstants` se respetan, no solo los símbolos.

Pasos siguientes

- [Áreas de trabajo y servicios de lenguaje](#) -Obtenga información sobre cómo integrar servicios de lenguaje en un área de trabajo de la carpeta abierta.
- [Compilación de área de trabajo](#) -Abrir carpeta es compatible con sistemas como MSBuild y archivos MAKE de compilación.

Áreas de trabajo y servicios de lenguaje

02/12/2019 • 5 minutes to read • [Edit Online](#)

Pueden proporcionar servicios de lenguaje [Abrir carpeta](#) a los usuarios las mismas características de lenguaje enriquecido se utilizan para cuando se trabaja con soluciones y proyectos. Un servicio de lenguaje podría activarse automáticamente según la extensión de archivo o el contenido de un documento abierto, aunque este servicio de lenguaje "archivo dinámico" se limita a resaltado de sintaxis. Se necesita información adicional para proporcionar una experiencia más enriquecida cuando Editar y revisar el código fuente. Cada servicio de lenguaje tiene su propia API para la inicialización con estos datos extra contextuales para un documento. Normalmente, esto se administra mediante un sistema de proyectos, que está estrechamente asociado para el servicio de lenguaje y para el sistema de compilación.

Inicialización

En un [área de trabajo](#), servicios de lenguaje se inicializan mediante un [ILanguageServiceProvider](#) punto de extensión que se especializa solo en ese servicio de lenguaje y no sabe nada de la creación de la compilación. De este modo, un propietario del servicio de lenguaje puede mantener una sola carpeta abierta extensión independientemente de cuántas patrones existe dentro de carpetas y archivos para ejecutar su compilador durante la compilación (por ejemplo MSBuild, archivos MAKE, etcetera.). Cuando se cambian los archivos desde el que se creó un contexto de archivo en disco y se actualiza el contexto del archivo, el proveedor de servicio de lenguaje se notifica el conjunto actualizado de contextos de archivo. El proveedor de servicio de lenguaje, a continuación, puede actualizar su modelo.

Cuando se abre un documento en el editor, Visual Studio sólo considera lenguaje proveedores de servicios que requieren tipos de contexto de archivo para que se puede encontrar un proveedor de contexto de archivo coincidente. A continuación, pasa los contextos de archivo de los proveedores de búsqueda de coincidencias para el proveedor de servicios de idioma seleccionado a través de `ILanguageServiceProvider.InitializeAsync`. Lo que hace el proveedor de servicio de lenguaje con que los datos de contexto de archivo están un detalle de implementación del proveedor del servicio de lenguaje, pero la experiencia de usuario esperado es un servicio de lenguaje enriquecido para el abre el documento.

Uso de ILanguageServiceProvider

El servicio de lenguaje se notificará cuando se crea un contexto de archivo con un `ContextType` que coincide con uno de los `SupportedContextTypes` valores del servidor idioma de exportación el atributo.

Para admitir un servicio de lenguaje, tendrá una extensión:

- Un único `Guid`. Esto se utilizará para `SupportedContextTypes` argumentos de atributo y en un `FileContext` objeto.
- Contexto del archivo de lenguaje
 - Generador del proveedor
 - `ExportFileContextProviderAttribute` atributo con el anterior se genera de manera única `Guid` en `SupportedContextTypes`
 - Implementa `IWorkspaceProviderFactory<IFileContextProvider>`
 - Implementación de proveedor `IFileContextProvider.GetContextsForFileAsync`
 - Crear un nuevo `FileContext` con el `contextType` argumento de constructor como el generado de forma única `Guid`
 - Use la `Context` propiedad de la `FileContext` para proporcionar datos adicionales a la

`ILanguageServiceProvider`

- servicio de lenguaje
 - Generador del proveedor
 - `ExportLanguageServiceProvider` atributo con el anterior se genera de manera única `Guid` en `SupportedContextTypes`
 - Implementa `IWorkspaceProviderFactory<ILanguageServiceProvider>`
 - Proveedor
 - Implementa `ILanguageServiceProvider`
 - Use `ILanguageServiceProvider.InitializeAsync` para habilitar los servicios de lenguaje para los argumentos proporcionados cuando se abre un archivo
 - Use `ILanguageServiceProvider.UninitializeAsync` para deshabilitar los servicios de lenguaje para los argumentos proporcionados cuando se cierra un archivo

WARNING

El `ILanguageServiceProvider` métodos podrían ser invocados por el área de trabajo en el subprocesso principal. Considere la posibilidad de programar el trabajo en un subprocesso distinto para evitar la introducción de retrasos de la interfaz de usuario.

Protocolo de servidor de lenguaje

El `Microsoft.VisualStudio.Workspace.*` API no son la única forma de habilitar el servicio de lenguaje en la carpeta abierta. Otra opción es usar un servidor de lenguaje. Para obtener más información, lea sobre la [protocolo del servidor idioma](#).

Interfaces relacionadas

- [ILanguageServiceProvider](#) se invoca cuando un archivo de coincidencia de tipos de archivo está abierto o cerrado para su edición.

Pasos siguientes

- [Compilación de área de trabajo](#) -Abrir carpeta es compatible con sistemas como MSBuild y archivos MAKE de compilación.

Compilación de área de trabajo

02/12/2019 • 5 minutes to read • [Edit Online](#)

Soporte técnico de compilación [Abrir carpeta](#) escenarios requiere un objeto extender para proporcionar [indizada](#) y [contexto del archivo](#) datos para el [área de trabajo](#), como así como la acción de compilación se ejecute.

A continuación es un resumen de lo que tendrá la extensión.

Contexto de archivo de compilación

- Generador del proveedor
 - `ExportFileContextProviderAttribute` atributo con `supportedContextTypeGuids` como todos los aplicables `string` constantes de `BuildContextTypes`
 - Implementa `IWorkspaceProviderFactory<IFileContextProvider>`
 - Proveedor de contexto de archivo
 - Devolver un `FileContext` para cada compilación operación y se admite la configuración
 - `contextType` De [BuildContextTypes](#)
 - `context` implementa [IBuildConfigurationContext](#) con el `Configuration` propiedad como la configuración de compilación (por ejemplo `"Debug|x86"`, `"ret"`, o `null` si no es aplicable). Como alternativa, use una instancia de [BuildConfigurationContext](#). El valor de configuración **debe** coinciden con la configuración del valor de datos de archivos indizados.

Valor de datos de archivo de compilación indizada

- Generador del proveedor
 - `ExportFileScannerAttribute` atributo con `IReadOnlyCollection<FileDataValue>` como un tipo compatible
 - Implementa `IWorkspaceProviderFactory<IFileScanner>`
- Analizador de archivos en `ScanContentAsync<T>`
 - Devuelve los datos cuando `FileScannerTypeConstants.FileDataValuesType` es el argumento de tipo
 - Devuelve un valor de datos de archivo para cada configuración que se construye con:
 - `type` como `BuildConfigurationContext.ContextTypeGuid`
 - `context` como la configuración de compilación (por ejemplo `"Debug|x86"`, `"ret"`, o `null` si no es aplicable). Este valor **debe** coinciden con la configuración desde el contexto del archivo.

Acción de contexto de archivo de compilación

- Generador del proveedor
 - `ExportFileContextActionProvider` atributo con `supportedContextTypeGuids` como todos los aplicables `string` constantes de `BuildContextTypes`
 - Implementa `IWorkspaceProviderFactory<IFileContextActionProvider>`
- Proveedor de la acción en `IFileContextActionProvider.GetActionsAsync`
 - Devolver un `IFileContextAction` que coincida con la determinada `FileContext.ContextType` valor
- Acción contextual de archivo
 - Implementa `IFileContextAction` y [IVsCommandItem](#)
 - `CommandGroup` propiedad devuelve `16537f6e-cb14-44da-b087-d1387ce3bf57`

- o `CommandId` es `0x1000` para compilación, `0x1010` para volver a generar, o `0x1020` para limpiar

NOTE

Puesto que el `FileDataValue` necesario indexar, habrá cierta cantidad de tiempo entre abrir el área de trabajo y el punto en el que se analiza el archivo para la funcionalidad de compilación completa. El retraso se verá en el abrir primero de una carpeta porque no hay ningún índice previamente almacenado en caché.

Informar de los mensajes de una compilación

La compilación puede exponer información, advertencia y mensajes de error a los usuarios de dos maneras. La forma más sencilla es usar el [IBuildMessageService](#) y proporcionar un [BuildMessage](#), así:

```
using Microsoft.VisualStudio.Workspace;
using Microsoft.VisualStudio.Workspace.Build;

private static void OutputBuildMessage(IWorkspace workspace)
{
    IBuildMessageService buildMessageService = workspace.GetBuildMessageService();

    if (buildMessageService != null)
    {
        // Example error build message. See the documentation for BuildMessage for more information.
        var message = new BuildMessage()
        {
            Type = BuildMessage.TaskType.Error,
            Code = "MY1001",
            TaskText = "This is a sample error",
            ProjectFile = "buildfile.bld",
            File = "sourcefile.src"
            LogMessage = $"This is sample text that will only go to the Build output window pane.\n"

            // And any other properties to set
        };

        buildMessageService.ReportBuildMessages(new BuildMessage[] { message });
    }
}
```

`BuildMessage.Type` y `BuildMessage.LogMessage` controlan el comportamiento de donde se presenta información al usuario. Cualquier `BuildMessage.TaskType` otro valor distinto `None` generará un **lista de errores** entrada con los detalles proporcionados. `LogMessage` siempre se mostrarán en el **compilar** panel de la **salida** ventana de herramientas.

Como alternativa, las extensiones pueden interactuar directamente con el **lista de errores** o **compilar** panel. Hay un error en las versiones anteriores de Visual Studio 2017 versión 15.7 donde el `pszProjectUniqueName` argumento de [OutputTaskItemStringEx2](#) se omite.

WARNING

Los autores de llamadas de `IFileContextAction.ExecuteAsync` puede proporcionar implementaciones subyacentes arbitrarias para la `IProgress<IFileContextActionProgressUpdate>` argumento. No invocar nunca `IProgress<IFileContextActionProgressUpdate>.Report(IFileContextActionProgressUpdate)` directamente. Actualmente no hay ningún directrices generales para usar este argumento, pero estas directrices están sujetos a cambios.

API relacionadas con la compilación

- [IBuildConfigurationContext](#) Proporciona los detalles de configuración de compilación.
- [IBuildMessageService](#) muestra [BuildMessages](#) a los usuarios.

Tasks.VS.JSON y launch.vs.json

Para obtener información sobre la creación de un archivo tasks.vs.json o launch.vs.json, consulte [personalización de compilación y depuración tareas](#).

Pasos siguientes

- [Protocolo del servidor idioma](#) -Obtenga información sobre cómo integrar servidores de lenguaje en Visual Studio.

Extender menús y comandos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Los comandos son la forma de agregar acciones y procesos a Visual Studio. En la mayoría de los casos, los comandos se muestran en menús o barras de herramientas. La plantilla de proyecto VSPackage muestra cómo implementar un comando muy básico. Para una implementación ligeramente más larga pero básica, consulte [Crear una extensión con un comando](#) de menú.

Para obtener más información acerca de los comandos, menús y barras de herramientas de Visual Studio, vea [Comandos, menús y barras de herramientas](#).

Comandos, menús y barras de herramientas se definen en el archivo `.vsct` que forma parte de proyectos de VSPackage. Puede encontrar información sobre el IDE de Visual Studio y el archivo `.vsct` en [Cómo VSPackages agregan elementos](#) de interfaz de usuario.

En los temas siguientes se explica cómo agregar diferentes tipos de comandos, menús y barras de herramientas.

En esta sección

- [Agregue un menú a la barra](#) de menús de Visual Studio Explica cómo agregar un menú a la barra de menús de Visual Studio superior.
- [Enlazar métodos abreviados de teclado a elementos](#) de menú Explica cómo agregar un método abreviado de teclado (como CTRL + 3) a un elemento de menú.
- [Añadir un submenú a un menú](#) Explica cómo agregar un submenú al menú superior.
- [Agregue una lista utilizada más recientemente a un submenú](#) Explica cómo agregar una lista de uso más reciente.
- [Crear grupos reutilizables de botones](#) Describe cómo agrupar elementos de comando para que se puedan incluir en varios menús.
- [Añadir iconos a los comandos](#) del menú Describe cómo agregar un icono a un comando tanto en una barra de herramientas como en un menú.
- [Cambiar el texto de un comando de menú](#) Describe el uso `TextChanges` de la marca para permitir que un elemento de menú se cambie dinámicamente.
- [Cambiar la apariencia de un comando](#) Describe cómo habilitar o deshabilitar dinámicamente un comando.
- [Actualizar la interfaz de usuario](#) Describe cómo forzar una actualización de la interfaz de usuario para reflejar los cambios recientes.
- [Localizar comandos de menú](#) Explica cómo localizar comandos de menú.

Crear una extensión con un comando de menú

06/04/2020 • 9 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo crear una extensión con un comando de menú que inicia el Bloc de notas.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un comando de menú

1. Cree un proyecto VSIX denominado **FirstMenuCommand**. Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Cuando se abra el proyecto, agregue una plantilla de elemento de comando personalizada denominada **FirstCommand**. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C- >** y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a *FirstCommand.cs*.
3. Compile la solución y comience la depuración.

Aparece la instancia experimental de Visual Studio. Para obtener más información acerca de la instancia experimental, consulte [La instancia experimental](#).
4. En la instancia experimental, abra la ventana **Extensiones y actualizaciones de herramientas**. > Debería ver la extensión **FirstMenuCommand** aquí. (Si abre **Extensiones y actualizaciones** en la instancia de trabajo de Visual Studio, no verá **FirstMenuCommand**).
4. En la instancia experimental, abra la ventana **Extensions > Manage Extensions**. Debería ver la extensión **FirstMenuCommand** aquí. (Si abre **Administrar extensiones** en la instancia de trabajo de Visual Studio, no verá **FirstMenuCommand**).

Ahora vaya al menú **Herramientas** en la instancia experimental. Debería ver **Invocar comando FirstCommand**. En este punto, el comando muestra un cuadro de mensaje que dice **FirstCommandPackage Inside FirstMenuCommand.FirstCommand.MenuItemCallback()**. Veremos cómo iniciar realmente el Bloc de notas desde este comando en la siguiente sección.

Cambiar el controlador de comandos de menú

Ahora vamos a actualizar el controlador de comandos para iniciar el Bloc de notas.

1. Detenga la depuración y vuelva a la instancia de trabajo de Visual Studio. Abra el archivo *FirstCommand.cs* y agregue la siguiente instrucción using:

```
using System.Diagnostics;
```

2. Busque el constructor **FirstCommand** privado. Aquí es donde se enlaza el comando al servicio de

comandos y se especifica el controlador de comandos. Cambie el nombre del controlador de comandos a `StartNotepad`, como se indica a continuación:

```
private FirstCommand(AsyncPackage package, OleMenuCommandService commandService)
{
    this.package = package ?? throw new ArgumentNullException(nameof(package));
    commandService = commandService ?? throw new ArgumentNullException(nameof(commandService));

    CommandID menuCommandID = new CommandID(CommandSet, CommandId);
    // Change to StartNotepad handler.
    MenuCommand menuItem = new MenuCommand(this.StartNotepad, menuCommandID);
    commandService.AddCommand(menuItem);
}
```

3. Quite `MenuItemCallback` el método `StartNotepad` y agregue un método, que simplemente iniciará el Bloc de notas:

```
private void StartNotepad(object sender, EventArgs e)
{
    Process proc = new Process();
    proc.StartInfo.FileName = "notepad.exe";
    proc.Start();
}
```

4. Ahora pruébalo. Al iniciar la depuración del proyecto y hacer clic en **Herramientas > Invocar FirstCommand**, debería ver una instancia del Bloc de notas emergente.

Puede utilizar una instancia [Process](#) de la clase para ejecutar cualquier ejecutable, no solo el Bloc de notas. Pruébalo con `calc.exe`, por ejemplo.

Limpiar el entorno experimental

Si está desarrollando varias extensiones o simplemente explorando resultados con diferentes versiones del código de extensión, su entorno experimental puede dejar de funcionar como debería. En este caso, debe ejecutar el script de restablecimiento. Se denomina **Restablecer la instancia experimental** de Visual Studio y se incluye como parte del SDK de Visual Studio. Este script elimina todas las referencias a las extensiones del entorno experimental, por lo que puede empezar desde cero.

Puede llegar a este script de una de estas dos maneras:

1. En el escritorio, busque **Restablecer la instancia experimental** de Visual Studio .
2. Ejecute lo siguiente desde la línea de comandos:

```
<VSSDK installation>\VisualStudioIntegration\Tools\Bin\CreateExpInstance.exe /Reset /VSInstance=
<version> /RootSuffix=Exp && PAUSE
```

Implemente la extensión

Ahora que tienes tu extensión de herramienta funcionando como quieras, es hora de pensar en compartirla con tus amigos y colegas. Eso es fácil, siempre y cuando tengan Visual Studio 2015 instalado. Todo lo que tienes que hacer es enviarles el archivo `.vsix` que has creado. (Asegúrese de compilarlo en el modo de versión.)

Puede encontrar el archivo `.vsix` para esta extensión en el directorio bin *FirstMenuCommand*. Específicamente, suponiendo que usted ha construido la configuración de la versión, estará en:

<directorio de código>, FirstMenuCommand, FirstMenuCommand, bin, Release, FirstMenuCommand.vsix

Para instalar la extensión, su amigo debe cerrar todas las instancias abiertas de Visual Studio y, a continuación, haga doble clic en el archivo `.vsix`, que abre el **instalador de VSIX**. Los archivos se copian en el *directorio* `%LocalAppData%-Microsoft-VisualStudio<versión>-Extensions`.

Cuando tu amigo vuelva a aparecer Visual Studio, encontrará la extensión FirstMenuCommand en **Extensiones y actualizaciones de herramientas** > . Pueden ir a **Extensiones y actualizaciones** para desinstalar o deshabilitar la extensión, también.

Pasos siguientes

Este tutorial le ha mostrado solo una pequeña parte de lo que puede hacer con una extensión de Visual Studio. Esta es una breve lista de otras cosas (razonablemente fáciles) que puede hacer con las extensiones de Visual Studio:

1. Puede hacer muchas más cosas con un simple comando de menú:
 - a. Añadir su propio icono: [Añadir iconos a los comandos](#) de menú
 - b. Cambiar el texto del comando de menú: [Cambiar el texto de un comando](#) de menú
 - c. Añadir un acceso directo de menú a un comando: [Enlazar métodos abreviados](#) de teclado a elementos de menú
2. Añadir diferentes tipos de comandos, menús y barras de herramientas: [Extender menús y comandos](#)
3. Agregue ventanas de herramientas y amplíe las ventanas de herramientas integradas de Visual Studio: [Ampliar y personalizar ventanas](#) de herramientas
4. Agregar IntelliSense, sugerencias de código y otras características a los editores de código existentes: [Amplíe el editor y](#) los servicios de lenguaje
5. Agregar opciones y páginas de propiedades y configuración de usuario a la extensión: [Extender propiedades y la ventana Propiedad](#) y Extender la configuración y las opciones de [usuario](#)

[illegible]

Agregue un menú a la barra de menús de Visual Studio

06/04/2020 • 7 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo agregar un menú a la barra de menús del entorno de desarrollo integrado (IDE) de Visual Studio. La barra de menús IDE contiene categorías de menú como **Archivo**, **Editar**, **Ver**, **Ventana** y **Ayuda**.

Antes de agregar un nuevo menú a la barra de menús de Visual Studio, considere si los comandos deben colocarse dentro de un menú existente. Para obtener más información acerca de la colocación de [comandos](#), vea [Menús y comandos para Visual Studio](#).

Los menús se declaran en el archivo `.vsct` del proyecto. Para obtener más información acerca de los menús y archivos `.vsct`, vea [Comandos, menús y barras de herramientas](#).

Al completar este tutorial, puede crear un menú denominado **TestMenu** que contenga un comando.

NOTE

En VS 2019, los menús de nivel superior aportados por las extensiones se colocan en el menú **Extensiones**.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Cree un proyecto VSIX que tenga una plantilla de elemento de comando personalizada

1. Cree un proyecto `TopLevelMenu` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix". Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. Cuando se abra el proyecto, agregue una plantilla de elemento de comando personalizada denominada **TestCommand**. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Visual C/ Extensibilidad** y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a `TestCommand.cs`.

Crear un menú en la barra de menús IDE

1. En el **Explorador de soluciones**, abra `TestCommandPackage.vsct`.

Al final del archivo, hay <un nodo de <> Symbols que contiene varios nodos GuidSymbol>. En el nodo denominado `guidTestCommandPackageCmdSet`, agregue un nuevo símbolo, como se indica a continuación:

```
<IDSymbol name="TopLevelMenu" value="0x1021"/>
```

2. Cree un <nodo> de <menús vacío en <el nodo comandos>, justo antes de que> Grupos. En <el nodo de> <De menús, agregue un nodo de> Demenú, como se indica a continuación:

```
<Menus>
  <Menu guid="guidTestCommandPackageCmdSet" id="TopLevelMenu" priority="0x700" type="Menu">
    <Parent guid="guidSHLMainMenu"
      id="IDG_VS_MM_TOOLSADDINS" />
    <Strings>
      <ButtonText>TestMenu</ButtonText>
      <CommandName>TestMenu</CommandName>
    </Strings>
  </Menu>
</Menus>
```

Los `guid` `id` valores y los valores del menú especifican el conjunto de comandos y el menú específico del conjunto de comandos.

Los `guid` `id` valores y los valores del elemento primario colocan el menú en la sección de la barra de menús de Visual Studio que contiene los menús Herramientas y Complementos.

El valor `CommandName` de la cadena especifica que el texto debe aparecer en el elemento de menú.

3. En <la sección> <de grupos, <busque el> de grupo y cambie el elemento> padre para que apunte al menú que acabamos de agregar:

```
<Groups>
  <Group guid="guidTestCommandPackageCmdSet" id="MyMenuGroup" priority="0x0600">
    <Parent guid="guidTestCommandPackageCmdSet" id="TopLevelMenu"/>
  </Group>
</Groups>
```

Esto hace que el grupo forme parte del nuevo menú.

1. En el Explorador de soluciones, abra *TopLevelMenuPackage.vsct*.

Al final del archivo, hay <un nodo de <> Symbols que contiene varios nodos GuidSymbol>. En el nodo denominado guidTopLevelMenuPackageCmdSet, agregue un nuevo símbolo, como se indica a continuación:

```
<IDSymbol name="TopLevelMenu" value="0x1021"/>
```

2. Cree un <nodo> de <menús vacío en <el nodo comandos>, justo antes de que> Grupos. En <el nodo de> <De menús, agregue un nodo de> Demenú, como se indica a continuación:

```
<Menus>
  <Menu guid="guidTopLevelMenuPackageCmdSet" id="TopLevelMenu" priority="0x700" type="Menu">
    <Parent guid="guidSHLMainMenu"
      id="IDG_VS_MM_TOOLSADDINS" />
    <Strings>
      <ButtonText>TestMenu</ButtonText>
      <CommandName>TestMenu</CommandName>
    </Strings>
  </Menu>
</Menus>
```

Los `guid` `id` valores y los valores del menú especifican el conjunto de comandos y el menú específico del conjunto de comandos.

Los `guid` `id` valores y los valores del elemento primario colocan el menú en la sección de la barra de menús de Visual Studio que contiene los menús Herramientas y Complementos.

El valor `CommandName` de la cadena especifica que el texto debe aparecer en el elemento de menú.

3. En <la sección> <de grupos, <busque el> de grupo y cambie el elemento> padre para que apunte al menú que acabamos de agregar:

```
<Groups>
  <Group guid="guidTopLevelMenuPackageCmdSet" id="MyMenuGroup" priority="0x0600">
    <Parent guid="guidTopLevelMenuPackageCmdSet" id="TopLevelMenu"/>
  </Group>
</Groups>
```

Esto hace que el grupo forme parte del nuevo menú.

4. Busque la sección `Buttons`. Observe que Visual Studio la plantilla Package `Button` ha generado un `MyMenuGroup` elemento que tiene su elemento primario establecido en `.` Como resultado, este comando aparece en el menú.

Compile y pruebe la extensión

1. Compile la solución y comience la depuración. Debería aparecer una instancia de la instancia experimental.
2. La barra de menús de la instancia experimental debe contener un menú **TestMenu**.
2. El menú **Extensiones** de la instancia experimental debe contener un menú **TestMenu**.
3. En el menú **TestMenu**, haga clic en **Invocar comando** de prueba.

Debe aparecer un cuadro de mensaje y mostrar el mensaje "TestCommand Package Inside TopLevelMenu.TestCommand.MenuItemCallback()".

Vea también

- [Comandos, menús y barras de herramientas](#)

Enlazar métodos abreviados de teclado a elementos de menú

06/04/2020 • 6 minutes to read • [Edit Online](#)

Para enlazar un método abreviado de teclado a un comando de menú personalizado, simplemente agregue una entrada al archivo `.vsct` para el paquete. En este tema se explica cómo asignar un método abreviado de teclado a un botón personalizado, elemento de menú o comando de barra de herramientas, y cómo aplicar la asignación de teclado en el editor predeterminado o limitarla a un editor personalizado.

Para asignar métodos abreviados de teclado a elementos de menú de Visual Studio existentes, vea [Identificar y personalizar métodos abreviados de teclado](#).

Elija una combinación de teclas

Muchos métodos abreviados de teclado ya se usan en Visual Studio. No debe asignar el mismo acceso directo a más de un comando porque los enlaces duplicados son difíciles de detectar y también pueden provocar resultados impredecibles. Por lo tanto, es una buena idea comprobar la disponibilidad de un acceso directo antes de asignarlo.

Para verificar la disponibilidad de un método abreviado de teclado

1. En la ventana **Opciones** > de **herramientas**, > seleccione **Teclado**.
2. Asegúrese de que **Usar nuevo acceso directo en** está establecido en **Global**.
3. En el cuadro **Teclas de método abreviado de prensa**, escriba el método abreviado de teclado que desea utilizar.

Si el acceso directo ya se usa en Visual Studio, el cuadro **Acceso directo que se usa actualmente por** mostrará el comando al que llama actualmente el acceso directo.

4. Pruebe diferentes combinaciones de teclas hasta que encuentre una que no esté asignada.

NOTE

Los métodos abreviados de teclado que utilizan **Alt** pueden abrir un menú y no ejecutar directamente un comando. Por lo tanto, el cuadro **Acceso directo utilizado actualmente puede** estar en blanco al escribir un acceso directo que incluya **Alt**. Puede comprobar que el acceso directo no abre un menú cerrando el cuadro de diálogo **Opciones** y, a continuación, presionando las teclas.

En el procedimiento siguiente se supone que tiene un `VSPackage` existente con un comando de menú. Si necesita ayuda para hacerlo, eche un vistazo a [Crear una extensión con un comando de menú](#).

Para asignar un método abreviado de teclado a un comando

1. Abra el archivo `.vsct` del paquete.
2. Cree una `<KeyBindings>` sección `<Commands>` vacía después de si aún no está presente.

WARNING

Para obtener más información acerca de los enlaces de claves, vea [Enlace de claves](#).

En `<KeyBindings>` la sección, `<KeyBinding>` cree una entrada.

Establezca `guid` los `id` atributos y en los del comando que desea invocar.

Establezca `mod1` el atributo en **Control, Alto Mayús**.

La sección KeyBindings debería tener un aspecto similar al siguiente:

```
<KeyBindings>
  <KeyBinding guid="<name of command set>" id="<name of command id>"
    editor="guidVSStd97" key1="1" mod1="CONTROL"/>
</KeyBindings>
```

Si el método abreviado de teclado requiere `mod2` `key2` más de dos teclas, establezca los atributos y.

En la mayoría de las situaciones, **Shift** no debe utilizarse sin un segundo modificador porque al presionarlo ya hace que la mayoría de las teclas alfanuméricas escriban una letra mayúscula o un símbolo.

Los códigos de clave virtual le permiten acceder a claves especiales que no tienen un carácter asociado, por ejemplo, las teclas de función y la tecla **Retroceso**. Para obtener más información, consulte [Códigos de clave virtual](#).

Para que el comando esté disponible en `editor` el `guidVSStd97` editor de Visual Studio, establezca el atributo en .

Para que el comando solo esté disponible `editor` en un editor personalizado, establezca el Visual Studio atributo en el nombre del editor personalizado generado por la plantilla de paquete al crear el VSPackage que incluye el editor personalizado. Para encontrar el valor de `<Symbols>` nombre, `<GuidSymbol>` busque `name` en la `editorfactory` sección un nodo cuyo atributo termine en "." Este es el nombre del editor personalizado.

Ejemplo

En este ejemplo se enlaza el método `cmdidMyCommand` abreviado de `MyPackage` teclado **Ctrl+Alt+C** a un comando denominado en un paquete denominado .

```
<CommandTable>
. . .
<Commands>
. . .
</Commands>
<KeyBindings>
  <KeyBinding guid="guidMyPackageCmdSet" id="cmdidMyCommand"
    key1="C" mod1="CONTROL" mod2="ALT" editor="guidVSStd97" />
</KeyBindings>
. . .
</CommandTable>
```

Ejemplo

En este ejemplo se enlaza el método `cmdidBold` abreviado de `TestEditor` teclado **Ctrl+B** a un comando denominado en un proyecto denominado . El comando solo está disponible en el editor personalizado y no en otros editores.

```
<KeyBinding guid="guidVSStd97" id="cmdidBold" editor="guidTestEditorEditorFactory" key1="B" mod1="Control" />
```

Vea también

- [Ampliación de menús y comandos](#)

Añadir un submenú a un menú

06/04/2020 • 5 minutes to read • [Edit Online](#)

Este tutorial se basa en la demostración en [Agregar un menú a la barra de menús](#) de Visual Studio mostrando cómo agregar un submenú al menú **TestMenu**.

Un submenú es un menú secundario que aparece en otro menú. Un submenú se puede identificar mediante la flecha que sigue a su nombre. Al hacer clic en el nombre, se mostrará el submenú y sus comandos.

Este tutorial crea un submenú en un menú de la barra de menús de Visual Studio y coloca un nuevo comando en el submenú. El tutorial también implementa el nuevo comando.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Añadir un submenú a un menú

1. Siga los pasos descritos en [Agregar un menú a la barra de menús](#) de Visual Studio para crear el proyecto y el elemento de menú. En los pasos de este tutorial se supone que el nombre del proyecto VSIX es .
2. Abra *TestCommandPackage.vsct*. En `<Symbols>` la sección, `<IDSymbol>` agregue un elemento para el submenú, uno para el grupo `<GuidSymbol>` de submenú y otro para el comando, todo en el nodo denominado "guidTopLevelMenuCmdSet." Este es el mismo nodo `<IDSymbol>` que contiene el elemento para el menú de nivel superior.

```
<IDSymbol name="SubMenu" value="0x1100"/>
<IDSymbol name="SubMenuGroup" value="0x1150"/>
<IDSymbol name="cmdidTestSubCommand" value="0x0105"/>
```

3. Agregue el submenú recién `<Menus>` creado a la sección.

```
<Menu guid="guidTestCommandPackageCmdSet" id="SubMenu" priority="0x0100" type="Menu">
  <Parent guid="guidTestCommandPackageCmdSet" id="MyMenuGroup"/>
  <Strings>
    <ButtonText>Sub Menu</ButtonText>
    <CommandName>Sub Menu</CommandName>
  </Strings>
</Menu>
```

El par GUID/ID del elemento primario especifica el grupo de menús que se generó en [Agregar un menú a la barra de menús](#) de Visual Studio y es un elemento secundario del menú de nivel superior.

4. Agregue el grupo de menús `<Groups>` definido en el paso 2 a la sección y consítue en un elemento secundario del submenú.

```
<Group guid="guidTestCommandPackageCmdSet" id="SubMenuGroup" priority="0x0000">
  <Parent guid="guidTestCommandPackageCmdSet" id="SubMenu"/>
</Group>
```

5. Agregue un `<Button>` nuevo `<Buttons>` elemento a la sección para definir el comando creado en el paso 2 como un elemento en el submenú.

```
<Button guid="guidTestCommandPackageCmdSet" id="cmdidTestSubCommand" priority="0x0000" type="Button">
  <Parent guid="guidTestCommandPackageCmdSet" id="SubMenuGroup" />
  <Icon guid="guidImages" id="bmpPic2" />
  <Strings>
    <CommandName>cmdidTestSubCommand</CommandName>
    <ButtonText>Test Sub Command</ButtonText>
  </Strings>
</Button>
```

6. Compile la solución y comience la depuración. Debería ver la instancia experimental.
7. Haga clic en **TestMenu** para ver un nuevo submenú denominado **Submenú**. Haga clic en **Submenú** para abrir el submenú y ver un nuevo comando, **Probar subcomando**. Observe que al hacer clic en **Probar subcomando** no hace nada.

Añadir un comando

1. Abra *TestCommand.cs* y agregue el siguiente ID de comando después del ID de comando existente.

```
public const int cmdidTestSubCmd = 0x0105;
```

2. Agregue el subcomando. Busque el constructor de comandos. Agregue las siguientes líneas justo `AddCommand` después de la llamada al método.

```
CommandID subCommandID = new CommandID(CommandSet, cmdidTestSubCmd);
MenuCommand subItem = new MenuCommand(new EventHandler(SubItemCallback), subCommandID);
commandService.AddCommand(subItem);
```

El `SubItemCallback` controlador de comandos se definirá más adelante. El constructor ahora debería tener este aspecto:

```

private TestCommand(Package package)
{
    if (package == null)
    {
        throw new ArgumentNullException("package");
    }

    this.package = package;

    OleMenuCommandService commandService = this.ServiceProvider.GetService(typeof(IMenuCommandService))
as OleMenuCommandService;
    if (commandService != null)
    {
        var menuCommandID = new CommandID(CommandSet, CommandId);
        var menuItem = new MenuCommand(this.MenuItemCallback, menuCommandID);
        commandService.AddCommand(menuItem);

        CommandID subCommandID = new CommandID(CommandSet, cmdidTestSubCmd);
        MenuCommand subItem = new MenuCommand(new EventHandler(SubItemCallback), subCommandID);
        commandService.AddCommand(subItem);
    }
}

```

3. Agregue `SubItemCallback()`. Este es el método al que se llama cuando se hace clic en el nuevo comando en el submenú.

```

private void SubItemCallback(object sender, EventArgs e)
{
    IVsUIShell uiShell = (IVsUIShell)this.ServiceProvider.GetServiceAsync(typeof(SVsUIShell));
    Guid clsid = Guid.Empty;
    int result;
    uiShell.ShowMessageBox(
        0,
        ref clsid,
        "TestCommand",
        string.Format(CultureInfo.CurrentCulture,
            "Inside TestCommand.SubItemCallback()"),
        this.ToString(),
        string.Empty,
        0,
        OLEMSGBUTTON.OLEMSGBUTTON_OK,
        OLEMSGDEFBUTTON.OLEMSGDEFBUTTON_FIRST,
        OLEMSGICON.OLEMSGICON_INFO,
        0,
        out result);
}

```

4. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.
5. En el menú **TestMenu**, haga clic en **Submenú** y, a continuación, haga clic en **Probar subcomando**. Debe aparecer un cuadro de mensaje y mostrar el texto "Test Command Inside TestCommand.SubItemCallback()".

Vea también

- [Agregue un menú a la barra de menús de Visual Studio](#)
- [Comandos, menús y barras de herramientas](#)

Agregue una lista utilizada más recientemente a un submenú

06/04/2020 • 10 minutes to read • [Edit Online](#)

Este tutorial se basa en las demostraciones de [Agregar un submenú a un menú](#) muestra cómo agregar una lista dinámica a un submenú. La lista dinámica constituye la base para crear una lista de uso más reciente (MRU).

Una lista de menús dinámicos comienza con un marcador de posición en un menú. Cada vez que se muestra el menú, el entorno de desarrollo integrado (IDE) de Visual Studio solicita el `VSPackage` para todos los comandos que se deben mostrar en el marcador de posición. Una lista dinámica puede aparecer en cualquier parte de un menú. Sin embargo, las listas dinámicas normalmente se almacenan y se muestran por sí mismas en los submenús o en la parte inferior de los menús. Mediante el uso de estos patrones de diseño, se habilita la lista dinámica de comandos para expandir y contraer sin afectar a la posición de otros comandos en el menú. En este tutorial, la lista MRU dinámica se muestra en la parte inferior de un submenú existente, separado del resto del submenú por una línea.

Técnicamente, una lista dinámica también se puede aplicar a una barra de herramientas. Sin embargo, desaconsejamos ese uso porque una barra de herramientas debe permanecer sin cambios a menos que el usuario tome pasos específicos para cambiarlo.

Este tutorial crea una lista MRU de cuatro elementos que cambian su orden cada vez que se selecciona uno de ellos (el elemento seleccionado se mueve a la parte superior de la lista).

Para obtener más información acerca de los menús y archivos `.vsct`, vea [Comandos, menús y barras de herramientas](#).

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Creación de una extensión

- Siga los procedimientos descritos en [Agregar un submenú a un menú](#) para crear el submenú que se modifica en los procedimientos siguientes.

Los procedimientos de este tutorial suponen `TopLevelMenu` que el nombre del `VSPackage` es , que es el nombre que se usa en [Agregar un menú a la barra de menús](#) de Visual Studio .

Crear un comando de lista de elementos dinámicos

1. Abra `TestCommandPackage.vsct`.
2. En `Symbols` la sección, `GuidSymbol` en el nodo denominado `guidTestCommandPackageCmdSet`, agregue el símbolo para el `MRUListGroup` grupo y el `cmdidMRUList` comando, como se indica a continuación.

```
<IDSymbol name="MRUListGroup" value="0x1200"/>
<IDSymbol name="cmdidMRUList" value="0x0200"/>
```

3. En `Groups` la sección, agregue el grupo declarado después de las entradas de grupo existentes.

```
<Group guid="guidTestCommandPackageCmdSet" id="MRUListGroup"
    priority="0x0100">
    <Parent guid="guidTestCommandPackageCmdSet" id="SubMenu"/>
</Group>
```

- En `Buttons` la sección, agregue un nodo para representar el comando recién declarado, después de las entradas de botón existentes.

```
<Button guid="guidTestCommandPackageCmdSet" id="cmdidMRUList"
    type="Button" priority="0x0100">
    <Parent guid="guidTestCommandPackageCmdSet" id="MRUListGroup" />
    <CommandFlag>DynamicItemStart</CommandFlag>
    <Strings>
        <CommandName>cmdidMRUList</CommandName>
        <ButtonText>MRU Placeholder</ButtonText>
    </Strings>
</Button>
```

El `DynamicItemStart` indicador permite que el comando se genere dinámicamente.

- Compile el proyecto e inicie la depuración para probar la visualización del nuevo comando.

En el menú **TestMenu**, haga clic en el nuevo submenú, **Submenú**, para mostrar el nuevo comando, **Marcador de posición MRU**. Después de implementar una lista dinámica de comandos MRU en el siguiente procedimiento, esta etiqueta de comando se reemplazará por esa lista cada vez que se abra el submenú.

Llenar la lista mRU

- En `TestCommandPackageGuids.cs`, agregue las siguientes líneas después `TestCommandPackageGuids` de los ids de comando existentes en la definición de clase.

```
public const string guidTestCommandPackageCmdSet = "00000000-0000-0000-0000-00000000"; // get the GUID
from the .vsct file
public const uint cmdidMRUList = 0x200;
```

- En `TestCommand.cs` agregue la siguiente instrucción using.

```
using System.Collections;
```

- Agregue el código siguiente en el `TestCommand` constructor después de la última llamada `AddCommand`. El `InitMRUMenu` se definirá más adelante

```
this.InitMRUMenu(commandService);
```

- Agregue el código siguiente en la clase `TestCommand`. Este código inicializa la lista de cadenas que representan los elementos que se mostrarán en la lista MRU.


```

private int numMRUIItems = 4;
private int baseMRUID = (int)TestCommandPackageGuids.cmdidMRUList;
private ArrayList mruList;

private void InitializeMRUList()
{
    if (null == this.mruList)
    {
        this.mruList = new ArrayList();
        if (null != this.mruList)
        {
            for (int i = 0; i < this.numMRUIItems; i++)
            {
                this.mruList.Add(string.Format(CultureInfo.CurrentCulture,
                    "Item {0}", i + 1));
            }
        }
    }
}

```

5. Después `InitializeMRUList` del método, agregue el `InitMRUMenu` método. Esto inicializa los comandos del menú de lista MRU.

```

private void InitMRUMenu(OleMenuCommandService mcs)
{
    InitializeMRUList();
    for (int i = 0; i < this.numMRUIItems; i++)
    {
        var cmdID = new CommandID(
            new Guid(TestCommandPackageGuids.guidTestCommandPackageCmdSet), this.baseMRUID + i);
        var mc = new OleMenuCommand(
            new EventHandler(OnMRUExec), cmdID);
        mc.BeforeQueryStatus += new EventHandler(OnMRUQueryStatus);
        mcs.AddCommand(mc);
    }
}

```

Debe crear un objeto de comando de menú para cada elemento posible de la lista MRU. El IDE `OnMRUQueryStatus` llama al método para cada elemento de la lista MRU hasta que no hay más elementos. En el código administrado, la única manera de que el IDE sepa que no hay más elementos es crear primero todos los elementos posibles. Si lo desea, puede marcar elementos adicionales `mc.Visible = false;` como no visibles al principio mediante el uso después de crear el comando de menú. Estos elementos se pueden hacer `mc.Visible = true;` visibles más adelante mediante el `OnMRUQueryStatus` método.

6. Después `InitMRUMenu` del método, `OnMRUQueryStatus` agregue el siguiente método. Este es el controlador que establece el texto para cada elemento MRU.

```

private void OnMRUQueryStatus(object sender, EventArgs e)
{
    OleMenuCommand menuCommand = sender as OleMenuCommand;
    if (null != menuCommand)
    {
        int MRUItemIndex = menuCommand.CommandID.ID - this.baseMRUID;
        if (MRUItemIndex >= 0 && MRUItemIndex < this.mruList.Count)
        {
            menuCommand.Text = this.mruList[MRUItemIndex] as string;
        }
    }
}

```

- Después `OnMRUQueryStatus` del método, `OnMRUExec` agregue el siguiente método. Este es el controlador para seleccionar un elemento MRU. Este método mueve el elemento seleccionado a la parte superior de la lista y, a continuación, muestra el elemento seleccionado en un cuadro de mensaje.

```
private void OnMRUExec(object sender, EventArgs e)
{
    var menuCommand = sender as OleMenuCommand;
    if (null != menuCommand)
    {
        int MRUItemIndex = menuCommand.CommandID.ID - this.baseMRUID;
        if (MRUItemIndex >= 0 && MRUItemIndex < this.mruList.Count)
        {
            string selection = this.mruList[MRUItemIndex] as string;
            for (int i = MRUItemIndex; i > 0; i--)
            {
                this.mruList[i] = this.mruList[i - 1];
            }
            this.mruList[0] = selection;
            System.Windows.Forms.MessageBox.Show(
                string.Format(CultureInfo.CurrentCulture,
                    "Selected {0}", selection));
        }
    }
}
```

Prueba de la lista MRU

- Compile la solución y comience la depuración.
- En el menú **TestMenu**, haga clic en **Invocar TestCommand**. Al hacerlo, se muestra un cuadro de mensaje que indica que se ha seleccionado el comando.

NOTE

Este paso es necesario para forzar el VSPackage para cargar y mostrar correctamente la lista MRU. Si omite este paso, no se muestra la lista MRU.

- En el menú **Menú de prueba**, haga clic en **Submenú**. Se muestra una lista de cuatro elementos al final del submenú, debajo de un separador. Al hacer clic en **Elemento 3**, debe aparecer un cuadro de mensaje y mostrar el texto, Elemento **seleccionado 3**. (Si no se muestra la lista de cuatro elementos, asegúrese de haber seguido las instrucciones del paso anterior.)
- Vuelva a abrir el submenú. Observe que el **elemento 3** está ahora en la parte superior de la lista y los otros elementos se han empujado hacia abajo una posición. Vuelva a hacer clic en el **elemento 3** y observe que el cuadro de mensaje sigue mostrando Elemento seleccionado **3**, lo que indica que el texto se ha movido correctamente a la nueva posición junto con la etiqueta de comando.

Vea también

- [Adición dinámica de elementos de menú](#)

Crear grupos reutilizables de botones

06/04/2020 • 7 minutes to read • [Edit Online](#)

Un grupo de comandos es una colección de comandos que siempre aparecen juntos en un menú o barra de herramientas. Cualquier grupo de comandos se puede reutilizar asignándolo a diferentes menús primarios en la sección CommandPlacements del archivo .vsct.

Los grupos de comandos suelen contener botones, pero también pueden contener otros menús o cuadros combinados.

Para crear un grupo reutilizable de botones

1. Cree un proyecto `ReusableButtons` VSIX denominado . Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. Cuando se abra el proyecto, agregue una plantilla de elemento de comando personalizada denominada **ReusableCommand**. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C-** y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a *ReusableCommand.cs*.
3. En el archivo .vsct, vaya a la sección Símbolos y busque el elemento GuidSymbol que contiene grupos y comandos para el proyecto. Debe denominarse guidReusableCommandPackageCmdSet.
4. Agregue un IDSymbol para cada botón que agregará al grupo, como en el ejemplo siguiente.

```
<GuidSymbol name="guidReusableCommandPackageCmdSet" value="{7f383b2a-c6b9-4c1d-b4b8-a26dc5b60ca1}">
  <IDSymbol name="MyMenuGroup" value="0x1020" />
  <IDSymbol name="ReusableCommandId" value="0x0100" />
  <IDSymbol name="SecondReusableCommandId" value="0x0200" />
</GuidSymbol>
```

De forma predeterminada, la plantilla de elemento de comando crea un grupo denominado **MyMenuGroup** y un botón que tiene el nombre que proporcionó, junto con una entrada IDSymbol para cada uno.

5. En la sección Grupos, cree un elemento Group que tenga los mismos atributos GUID e ID que los especificados en la sección Símbolos. También puede utilizar un grupo existente o la entrada proporcionada por la plantilla de comando, como en el ejemplo siguiente. Este grupo aparece en el menú **Herramientas**

```
<Groups>
  <Group guid="guidReusableCommandPackageCmdSet" id="MyMenuGroup" priority="0x0600">
    <Parent guid="guidSHLMainMenu" id="IDM_VS_MENU_TOOLS"/>
  </Group>
</Groups>
```

Para crear un grupo de botones para reutilizarlos

1. Puede colocar un comando o menú en un grupo utilizando el grupo como elemento primario en la definición del comando o menú, o colocando el comando o menú en el grupo mediante la sección CommandPlacements.

En la sección Botones, defina un botón que tenga el grupo como elemento primario o use el botón proporcionado por la plantilla de paquete, como se muestra en el ejemplo siguiente.

```
<Button guid="guidReusableCommandPackageCmdSet" id="ReusableCommandId" priority="0x0100" type="Button">
  <Parent guid="guidReusableCommandPackageCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <ButtonText>Invoke ReusableCommand</ButtonText>
  </Strings>
</Button>
```

2. Si un botón debe aparecer en más de un grupo, cree una entrada para él en la sección CommandPlacements, que debe colocarse después de la sección Commands. Establezca los atributos GUID e ID del elemento CommandPlacement para que coincidan con los del botón que desea colocar y, a continuación, establezca el GUID y el identificador de su elemento Primario en los del grupo de destino, como se muestra en el ejemplo siguiente.

```
<CommandPlacements>
  <CommandPlacement guid="guidReusableCommandPackageCmdSet" id="SecondReusableCommandId"
priority="0x105">
    <Parent guid="guidReusableCommandPackageCmdSet" id="MyMenuGroup" />
  </CommandPlacement>
</CommandPlacements>
```

NOTE

El valor del campo Prioridad determina la posición del comando en el nuevo grupo de comandos. Las prioridades establecidas en el elemento CommandPlacement reemplazan las establecidas en la definición del elemento. Los comandos que tienen valores de prioridad más bajos se muestran antes que los comandos que tienen valores de prioridad más altos. Se permiten valores de prioridad duplicados, pero la posición relativa de los comandos que tienen el mismo valor de prioridad no se puede garantizar porque el orden en el que el comando **devenv /setup** crea la interfaz final del registro puede no ser coherente.

Para poner un grupo reutilizable de botones en un menú

1. Cree una entrada `CommandPlacements` en la sección. Establezca el GUID y `CommandPlacement` el identificador del elemento en los del grupo y establezca el GUID principal y el identificador en los de la ubicación de destino.

La sección CommandPlacements debe colocarse justo después de la sección Comandos:

```
<CommandTable>
...
  <Commands>... </Commands>
  <CommandPlacements>... </CommandPlacements>
...
</CommandTable>
```

Un grupo de comandos se puede incluir en más de un menú. El menú primario puede ser uno que ha Visual Studio creado, uno proporcionado por (como se describe en *ShellCmdDef.vsct* o *SharedCmdDef.vsct*), o uno que se define en otro VSPackage. El número de capas de elementos primarios es ilimitado Visual Studio siempre y cuando el menú primario se conecte finalmente a o a un menú contextual que se muestra mediante un VSPackage.

En el ejemplo siguiente se coloca el grupo en la barra de herramientas **del Explorador** de soluciones, a la derecha de los otros botones.

```
<CommandPlacements>
  <CommandPlacement guid="guidReusableCommandPackageCmdSet" id="MyMenuGroup" priority="0xF00">
    <Parent guid="guidSHLMainMenu" id="IDM_VS_TOOL_PROJWIN"/>
  </CommandPlacement>
</CommandPlacements>
```

```
<CommandPlacements>
  <CommandPlacement guid="guidButtonGroupCmdSet" id="MyMenuGroup"
    priority="0x605">
    <Parent guid="guidSHLMainMenu" id="IDM_VS_MENU_TOOLS" />
  </CommandPlacement>
</CommandPlacements>
```

Añadir iconos a los comandos del menú

06/04/2020 • 5 minutes to read • [Edit Online](#)

Los comandos pueden aparecer tanto en los menús como en las barras de herramientas. En las barras de herramientas, es común que un comando se muestre con solo un icono (para ahorrar espacio) mientras que en los menús suele aparecer un comando con un icono y texto.

Los iconos tienen 16 píxeles de ancho por 16 píxeles de alto y pueden tener una profundidad de color de 8 bits (256 colores) o una profundidad de color de 32 bits (color verdadero). Se prefieren los iconos de color de 32 bits. Los iconos normalmente se organizan en una sola fila horizontal en un único mapa de bits, aunque se permiten varios mapas de bits. Este mapa de bits se declara en el archivo `.vsct` junto con los iconos individuales disponibles en el mapa de bits. Consulte la referencia del [elemento Bitmaps](#) para obtener más detalles.

Añadir un icono a un comando

En el procedimiento siguiente se supone que tiene un proyecto de VSPackage existente con un comando de menú. Para saber cómo hacerlo, consulte [Crear una extensión con un comando de menú](#).

1. Cree un mapa de bits con una profundidad de color de 32 bits. Un icono siempre es de 16 x 16, por lo que este mapa de bits debe tener 16 píxeles de alto y un múltiplo de 16 píxeles de ancho.

Cada icono se coloca en el mapa de bits uno al lado del otro en una sola fila. Utilice el canal alfa para indicar los lugares de transparencia en cada icono.

Si utiliza una profundidad de color de 8 `RGB(255,0,255)` bits, utilice magenta, , como transparencia. Sin embargo, se prefieren los iconos de color de 32 bits.

2. Copie el archivo de icono en el directorio *Resources* del proyecto DE VSPackage. En el Explorador de **soluciones**, agregue el icono al proyecto. (Seleccione **Recursos**, en el menú contextual, haga clic en **Agregar**, luego en **Elemento existente** y seleccione el archivo de icono.)
3. Abra el archivo `.vsct` en el editor.
4. Agregue `<GuidSymbol>` un elemento con el nombre `testIcon`. Cree un GUID (**Tools** > **Create GUID**), a continuación, seleccione Registry Format (**Formato del Registro**) y haga clic en Copy (**Copiar**) y péguelo en el `value` atributo. El resultado debería tener este aspecto:

```
<!-- Create your own GUID -->
<GuidSymbol name="testIcon" value="{00000000-0000-0000-0000-0000}">
```

5. Agregue `<IDSymbol>` un para el icono. El `name` atributo es el identificador del `value` icono y indica su posición en la tira, si existe. Si solo hay un icono, agregue 1. El resultado debería tener este aspecto:

```
<!-- Create your own GUID -->
<GuidSymbol name="testIcon" value="{00000000-0000-0000-0000-0000}">
  <IDSymbol name="testIcon1" value="1" />
</GuidSymbol>
```

6. Cree `<Bitmap>` un `<Bitmaps>` en la sección del archivo `.vsct` para representar el mapa de bits que contiene los iconos.

- Establezca `guid` el valor en `<GuidSymbol>` el nombre del elemento que creó en el paso anterior.

- Establezca `href` el valor en la ruta de acceso relativa del archivo de mapa de bits (en este caso `Recursos\<nombre>` de archivo de icono).
- Establezca `usedList` el valor en el `IDSymbol` que creó anteriormente. Este atributo especifica una lista delimitada por comas de los iconos que se usarán en el `VSPackage`. Los iconos que no están en la lista se excluyen la compilación de formularios.

El bloque `Bitmap` debería tener este aspecto:

```
<Bitmap guid="testIcon" href="Resources\<icon file name>" usedList="testIcon1"/>
```

7. En el `<Button>` elemento existente, establezca el `Icon` elemento en los valores `GUIDSymbol` e `IDSymbol` que creó anteriormente. Este es un ejemplo de un elemento `Button` con esos valores:

```
<Button guid="guidAddIconCmdSet" id="cmdidMyCommand" priority="0x0100" type="Button">
  <Parent guid="guidAddIconCmdSet" id="MyMenuGroup" />
  <Icon guid="testIcon" id="testIcon1" />
  <Strings>
    <ButtonText>My Command name</ButtonText>
  </Strings>
</Button>
```

8. Pruebe su icono. Compile la solución y comience la depuración. En la instancia experimental, busque el comando. Debería mostrar el icono que agregó.

Vea también

- [Ampliación de menús y comandos](#)
- [Referencia de esquema XML de VSCT](#)

Cambiar el texto de un comando de menú

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los pasos siguientes muestran cómo cambiar la etiqueta `IMenuCommandService` de texto de un comando de menú mediante el servicio.

Cambiar una etiqueta de comando de menú con el `IMenuCommandService`

1. Cree un proyecto `MenuText` VSIX denominado con un comando de menú denominado `ChangeMenuText`. Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. En el archivo `.vsct`, agregue la `TextChanges` marca al comando de menú, como se muestra en el ejemplo siguiente.

```
<Button guid="guidChangeMenuTextPackageCmdSet" id="ChangeMenuTextId" priority="0x0100" type="Button">
  <Parent guid="guidChangeMenuTextPackageCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <CommandFlag>TextChanges</CommandFlag>
  <Strings>
    <ButtonText>Invoke ChangeMenuText</ButtonText>
  </Strings>
</Button>
```

3. En el archivo `ChangeMenuText.cs`, cree un controlador de eventos al que se llamará antes de que se muestre el comando de menú.

```
private void OnBeforeQueryStatus(object sender, EventArgs e)
{
    var myCommand = sender as OleMenuCommand;
    if (null != myCommand)
    {
        myCommand.Text = "New Text";
    }
}
```

También puede actualizar el estado del comando de `Visible` menú `Checked` en `Enabled` este `OleMenuCommand` método cambiando las propiedades `Visible`, `Checked` y `Enabled` en el objeto.

4. En el `ChangeMenuText` constructor, reemplace el código de inicialización y colocación de comandos original con código que crea un `OleMenuCommand` (en lugar de un `MenuCommand`) que representa el comando de menú, agrega el `BeforeQueryStatus` controlador de eventos y proporciona el comando de menú al servicio de comandos de menú.

Esto es lo que debería parecer:


```

private ChangeMenuText(Package package)
{
    if (package == null)
    {
        throw new ArgumentNullException(nameof(package));
    }

    this.package = package;

    OleMenuCommandService commandService = this.ServiceProvider.GetService(typeof(IMenuCommandService))
as OleMenuCommandService;
    if (commandService != null)
    {
        CommandID menuCommandID = new CommandID(MenuGroup, CommandId);
        EventHandler eventHandler = this.ShowMessageBox;
        OleMenuCommand menuItem = new OleMenuCommand(ShowMessageBox, menuCommandID);
        menuItem.BeforeQueryStatus +=
            new EventHandler(OnBeforeQueryStatus);
        commandService.AddCommand(menuItem);
    }
}

```

5. Compile la solución y comience la depuración. Aparece la instancia experimental de Visual Studio.
6. En el menú **Herramientas** debería ver un comando denominado **Invocar ChangeMenuText**.
7. Haga clic en el comando. Debería ver el cuadro de mensaje que anuncia que **MenuItemCallback** se ha llamado. Al descartar el cuadro de mensaje, debería ver que el nombre del comando en el menú Herramientas ahora es **Nuevo texto**.

Cambiar la apariencia de un comando

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede proporcionar comentarios al usuario cambiando la apariencia de un comando. Por ejemplo, es posible que desee que un comando tenga un aspecto diferente cuando no esté disponible. Puede hacer que los comandos estén disponibles o no disponibles, ocultarlos o mostrarlos, o marcarlos o desmarcarlos en el menú.

Para cambiar la apariencia de un comando, realice una de estas acciones:

- Especifique las marcas adecuadas en la definición de comando en el archivo de tabla de comandos.
- Utilice [OleMenuCommandService](#) el servicio.
- Implemente [IOleCommandTarget](#) la interfaz y modifique los objetos de comando sin procesar.

En los pasos siguientes se muestra cómo buscar y actualizar la apariencia de un comando mediante Managed Package Framework (MPF).

Para cambiar la apariencia de un comando de menú

1. Siga las instrucciones de [Cambiar el texto de un comando](#) de menú para crear un elemento de menú denominado `New Text`.
2. En el archivo *ChangeMenuText.cs*, agregue la siguiente instrucción using:

```
using System.Security.Permissions;
```

3. En el archivo *ChangeMenuTextPackageGuids.cs*, agregue la siguiente línea:

```
public const string guidChangeMenuTextPackageCmdSet= "00000000-0000-0000-0000-00000000"; // get the GUID from the .vsct file
```

4. En el archivo *ChangeMenuText.cs*, reemplace el código del método `ShowMessageBox` por lo siguiente:

```
private void ShowMessageBox(object sender, EventArgs e)
{
    var command = sender as OleMenuCommand;
    if (command.Text == "New Text")
        ChangeMyCommand(command.CommandID.ID, false);
}
```

5. Obtenga el comando que desea [OleMenuCommandService](#) actualizar desde el objeto y, a continuación, establezca las propiedades adecuadas en el objeto de comando. Por ejemplo, el método siguiente hace que el comando especificado de un conjunto de comandos de `VSPackage` esté disponible o no esté disponible. El código siguiente hace `New Text` que el elemento de menú denominado no esté disponible después de hacer clic en él.

```

public bool ChangeMyCommand(int cmdID, bool enableCmd)
{
    bool cmdUpdated = false;
    var mcs = this.ServiceProvider.GetService(typeof(IMenuCommandService))
        as OleMenuCommandService;
    var newCmdID = new CommandID(new Guid(ChangeMenuTextPackageGuids.guidChangeMenuTextPackageCmdSet),
cmdID);
    MenuCommand mc = mcs.FindCommand(newCmdID);
    if (mc != null)
    {
        mc.Enabled = enableCmd;
        cmdUpdated = true;
    }
    return cmdUpdated;
}

```

6. Compile la solución y comience la depuración. Debería aparecer la instancia experimental de Visual Studio.
7. En el menú **Herramientas**, haga clic en el comando **Invocar ChangeMenuText**. En este punto, el nombre del comando es **Invoke ChangeMenuText**, por lo que el controlador de comandos no llama a **ChangeMyCommand()**.
8. En el menú **Herramientas** ahora debería ver **Nuevo texto**. Haga clic en **Nuevo texto**. El comando ahora debería estar atenuado.

Vea también

- [Comandos, menús y barras de herramientas](#)
- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Ampliación de menús y comandos](#)
- [Tabla de comandos de Visual Studio \(. Vsct\) Archivos](#)

Actualización de la interfaz de usuario

06/04/2020 • 3 minutes to read • [Edit Online](#)

Después de implementar un comando, puede agregar código para actualizar la interfaz de usuario con el estado de los nuevos comandos.

En una aplicación Típica de Win32, el conjunto de comandos se puede sondear continuamente y el estado de los comandos individuales se puede ajustar a medida que el usuario los ve. Sin embargo, dado que el Visual Studio shell puede hospedar un número ilimitado de VSPackages, el sondeo extenso puede disminuir la capacidad de respuesta, especialmente el sondeo entre ensamblados de interoperabilidad entre código administrado y COM.

Para actualizar la interfaz de usuario

1. Realice uno de estos pasos:

- Llame al método [UpdateCommandUI](#) .

Una [IVsUIShell](#) interfaz se puede [SVsUIShell](#) obtener del servicio, de la siguiente manera.

```
void UpdateUI(Microsoft.VisualStudio.Shell.ServiceProvider sp)
{
    IVsUIShell vsShell = (IVsUIShell)sp.GetService(typeof(IVsUIShell));
    if (vsShell != null)
    {
        int hr = vsShell.UpdateCommandUI(0);
        Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(hr);
    }
}
```

Si el parámetro [UpdateCommandUI](#) de la `TRUE` es distinto de cero (), entonces la actualización se realiza de forma sincrónica e inmediatamente. Se recomienda pasar cero `FALSE` () para este parámetro para ayudar a mantener un buen rendimiento. Si desea evitar el almacenamiento `DontCache` en caché, aplique la marca al crear el comando en el archivo .vsct. Sin embargo, utilice la bandera con precaución o el rendimiento podría disminuir. Para obtener más información acerca de los indicadores de comando, consulte la documentación del [elemento Command Flag](#).

- En VSPackages que hospedan un control ActiveX mediante el modelo de activación en [UpdateUI](#) contexto en una ventana, podría ser más conveniente usar el método. El [UpdateCommandUI](#) método [IVsUIShell](#) en la [UpdateUI](#) interfaz [IOleInPlaceComponentUIManager](#) y el método en la interfaz son funcionalmente equivalentes. Ambos hacen que el entorno vuelva a consultar el estado de todos los comandos. Normalmente, una actualización no se realiza inmediatamente. En su lugar, una actualización se retrasa hasta el tiempo de inactividad. El shell almacena en caché el estado del comando para ayudar a mantener un buen rendimiento. Si desea evitar el almacenamiento `DontCache` en caché, aplique la marca al crear el comando en el archivo .vsct. Sin embargo, utilice el indicador con precaución porque el rendimiento podría disminuir.

Tenga en cuenta [IOleInPlaceComponentUIManager](#) que puede `QueryInterface` obtener la [IOleComponentUIManager](#) interfaz llamando al método [SOleComponentUIManager](#) en un objeto o obteniendo la interfaz del servicio.

Vea también

- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)

- Implementación

Localizar comandos de menú

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede proporcionar texto localizado para los comandos de menú y barra de herramientas mediante la creación de archivos `.vsct` localizados y archivos `.resx` localizados para el VSPackage y, a continuación, actualizar los archivos de proyecto para incorporar los cambios.

Para obtener información sobre cómo localizar la experiencia de instalación, consulte [Localizar paquetes VSIX](#).

Localizar nombres de comandos

En VSPackages, los comandos de menú y los botones de barra de herramientas se definen en el archivo `.vsct`.

1. En el **Explorador de soluciones**, cambie el nombre del archivo `.vsct` de `filename.vsct` a `filename.en-US.vsct`.
2. Haga una copia de `filename.en-US.vsct` para cada idioma localizado.

Asigne un nombre a cada nombre de *archivo de copia*. `Locale.vsct`, donde `.locale` es un nombre de referencia cultural determinado. Para obtener una lista de valores de nombre de referencia cultural, vea [Valores de configuración regional asignados por Microsoft](#).

Estos *nombre de archivo*. Los archivos `Locale.vsct` contendrán el texto del menú localizado del paquete.

3. Abra cada *nombre de archivo*. `Locale.vsct` para localizar el texto.
 - a. Modifique los valores del elemento `ButtonText` según corresponda para el idioma determinado.
 - b. Si va a proporcionar iconos localizados, modifique los valores de `mapa` de bits para que apunten a los archivos de destino.

En el ejemplo siguiente se muestra el texto de los botones en inglés y español de un comando para abrir una ventana de herramientas del Explorador de árboles genealógicos.

[`FamilyTree.en-US.vsct`]

```
<Button guid="guidLocalizedPackageCmdSet" id="cmdidFamilyTree" priority="0x0100" type="Button">
  <Parent guid="guidSHLMainMenu" id="IDG_VS_WNDO_OTRWNDWS1"/>
  <Icon guid="guidImages" id="bmpPic2" />
  <Strings>
    <CommandName>cmdidFamilyTree</CommandName>
    <ButtonText>Family Tree Explorer</ButtonText>
  </Strings>
</Button>
```

[`FamilyTree.es-ES.vsct`]

```
<Button guid="guidLocalizedPackageCmdSet" id="cmdidFamilyTree" priority="0x0100" type="Button">
  <Parent guid="guidSHLMainMenu" id="IDG_VS_WNDO_OTRWNDWS1"/>
  <Icon guid="guidImages" id="bmpPic2" />
  <Strings>
    <CommandName>cmdidFamilyTree</CommandName>
    <ButtonText>Explorar el arbol genealogico</ButtonText>
  </Strings>
</Button>
```

Localizar otros recursos de texto

Los recursos de texto distintos de los nombres de comando se definen en archivos de recursos (*.resx*).

1. Cambie el nombre de *VSPackage.resx* a *VSPackage.en-US.resx*.
2. Haga una copia del archivo *VSPackage.en-US.resx* para cada idioma localizado.

Asigne un nombre a cada copia de *VSPackage.Locale.resx*, donde *.Locale* es un nombre de referencia cultural determinado.

3. Cambie el nombre de *Resources.resx* a *Resources.en-US.resx*.
4. Haga una copia del archivo *Resources.en-US.resx* para cada idioma localizado.

Asigne un nombre a cada copia de *Recursos.Locale.resx*, donde *.Locale* es un nombre de referencia cultural determinado.

5. Abra cada archivo *.resx* para modificar los valores de cadena según corresponda para el idioma y la referencia cultural concretos. En el ejemplo siguiente se muestra la definición de recursos localizados para la barra de título de una ventana de herramientas.

[*Resources.en-US.resx*]

```
<data name="ToolWindowTitle" xml:space="preserve">
  <value>Family Tree Explorer</value>
</data>
```

[*Resources.es-ES.resx*]

```
<data name="ToolWindowTitle" xml:space="preserve">
  <value>Explorador del árbol genealógico</value>
</data>
```

Incorporar recursos localizados en el proyecto

Debe modificar el archivo *de assemblyinfo.cs* y el archivo de proyecto para incorporar los recursos localizados.

1. Desde el nodo **Propiedades** del Explorador de soluciones, abra *assemblyinfo.cs* o *assemblyinfo.vb* en el editor.
2. Agregue la siguiente entrada.

```
[assembly: NeutralResourcesLanguage("en-US", UltimateResourceFallbackLocation.Satellite)]
```

Esto establece el inglés de EE. UU. como idioma predeterminado.

3. Descargue el proyecto.
4. Abra el archivo de proyecto en el editor.
5. En el **Project** elemento raíz, agregue un **PropertyGroup** elemento con un **UICulture** elemento que coincida con el idioma predeterminado.

```
<PropertyGroup>
  <UICulture>en-US</UICulture>
</PropertyGroup>
```

Esto establece el inglés de EE. UU. como la referencia cultural de interfaz de usuario predeterminada para los controles de Windows Presentation Foundation (WPF).

6. Busque `ItemGroup` el elemento `EmbeddedResource` que contiene elementos.
7. En `EmbeddedResource` el elemento que llama a `VSPackage.en-US.resx`, reemplace el `ManifestResourceName` elemento por un `LogicalName` elemento establecido en `VSPackage.en-US.Resources` continuación:

```
<EmbeddedResource Include="VSPackage.en-US.resx">
  <MergeWithCTO>true</MergeWithCTO>
  <LogicalName>VSPackage.en-US.Resources</LogicalName>
</EmbeddedResource>
```

8. Para cada idioma localizado, `EmbeddedResource` `VsPackage.en-US` copie el elemento para `es-ES`, y establezca el atributo **Include** y el elemento **LogicalName** de la copia en la configuración regional de destino.
9. A cada `VSCTCompile` elemento localizado, agregue `ResourceName` `Menus.ctmenu` un elemento que apunte a `Menus.ctmenu`, como se muestra en el ejemplo siguiente:

```
<ItemGroup>
  <VSCTCompile Include="LocalizedPackage.es-ES.vsct">
    <ResourceName>Menus.ctmenu</ResourceName>
  </VSCTCompile>
</ItemGroup>
```

10. Guarde el archivo de proyecto y vuelva a cargar el proyecto.
11. Compile el proyecto.

Esto crea un ensamblado principal y ensamblados de recursos para cada idioma. Para obtener información sobre cómo localizar el proceso de implementación, consulte [Localizar paquetes VSIX](#)

Vea también

- [Extender menús y comandos](#)
- [Globalizar y localizar aplicaciones](#)

Añadir comandos a las barras de herramientas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una barra de herramientas es una franja horizontal o vertical que contiene botones enlazados a comandos. Puede agregar sus propios comandos a las barras de herramientas en muchas ventanas diferentes en Visual Studio: la ventana principal de Visual Studio, el Explorador de soluciones y sus propias ventanas de herramientas personalizadas.

Esta sección muestra una serie de cosas que puede hacer con los comandos en las barras de herramientas.

En esta sección

- [Añadir una barra de herramientas](#) Explica cómo agregar una barra de herramientas al conjunto de barras de herramientas en la ventana principal de Visual Studio.
- [Añadir una barra de herramientas a una ventana de herramientas](#) Explica cómo agregar una barra de herramientas a una ventana de herramientas.
- [Agregue un comando a la barra](#) de herramientas del Explorador de soluciones Explica cómo agregar un comando a la barra de herramientas en el Explorador de soluciones.
- [Agregar un controlador de menú](#) a una barra de herramientas Explica cómo agregar un controlador de menú a una barra de herramientas.
- [Añadir elementos](#) de menú de forma dinámica Explica cómo agregar elementos de menú dinámicamente en diferentes contextos.

Añadir una barra de herramientas

06/04/2020 • 4 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo agregar una barra de herramientas al IDE de Visual Studio.

Una barra de herramientas es una franja horizontal o vertical que contiene botones enlazados a comandos. Dependiendo de su implementación, una barra de herramientas en el IDE se puede cambiar de posición, acoplar en cualquier lado de la ventana principal del IDE o hacer que permanezca delante de otras ventanas.

Además, los usuarios pueden agregar comandos a una barra de herramientas o quitarlos de ella mediante el cuadro de diálogo **Personalizar**. Normalmente, las barras de herramientas de VSPackages son personalizables por el usuario. El IDE controla toda la personalización y el VSPackage responde a los comandos. El VSPackage no tiene que saber dónde se encuentra físicamente un comando.

Para obtener más información acerca de los menús, consulte [Comandos, menús y barras de herramientas](#).

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una extensión con una barra de herramientas

Cree un proyecto `IDEToolbar` VSIX denominado . Agregue una plantilla de elemento de comando de menú denominada **ToolbarTestCommand**. Para obtener información sobre cómo hacerlo, consulte [Crear una extensión con un comando de menú](#).

Crear una barra de herramientas para el IDE

1. En *ToolbarTestCommandPackage.vscf*, busque la sección Símbolos. En el guidSymbol elemento denominado guidToolbarTestCommandPackageCmdSet, agregue declaraciones para una barra de herramientas y un grupo de barra de herramientas, como se indica a continuación.

```
<IDSymbol name="Toolbar" value="0x1000" />
<IDSymbol name="ToolbarGroup" value="0x1050" />
```

2. En la parte superior de la sección Comandos, cree una sección Menús. Agregue un elemento Menu a la sección Menus para definir la barra de herramientas.

```
<Menus>
  <Menu guid="guidToolbarTestCommandPackageCmdSet" id="Toolbar"
    type="Toolbar" >
    <CommandFlag>DefaultDocked</CommandFlag>
    <Strings>
      <ButtonText>Test Toolbar</ButtonText>
      <CommandName>Test Toolbar</CommandName>
    </Strings>
  </Menu>
</Menus>
```

Las barras de herramientas no se pueden anidar como submenús. Por lo tanto, no es necesario asignar un grupo primario. Además, no es necesario establecer una prioridad, ya que el usuario puede mover barras de herramientas. Normalmente, la ubicación inicial de una barra de herramientas se define mediante programación, pero se conservan los cambios posteriores realizados por el usuario.

3. En la sección [Grupos](#), después de la entrada de grupo existente, defina un elemento [Group](#) que contenga los comandos de la barra de herramientas.

```
<Group guid="guidToolbarTestCommandPackageCmdSet" id="ToolbarGroup"
  priority="0x0000">
  <Parent guid="guidToolbarTestCommandPackageCmdSet" id="Toolbar"/>
</Group>
```

4. Haga que el botón aparezca en la barra de herramientas. En la sección Botones, reemplace el bloque Padre del botón a la barra de herramientas. El bloque Button resultante debería tener este aspecto:

```
<Button guid="guidToolbarTestCommandPackageCmdSet" id="ToolbarTestCommandId" priority="0x0100"
type="Button">
  <Parent guid="guidToolbarTestCommandPackageCmdSet" id="ToolbarGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <ButtonText>Invoke ToolbarTestCommand</ButtonText>
  </Strings>
</Button>
```

De forma predeterminada, si una barra de herramientas no tiene comandos, no aparece.

5. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.
6. Haga clic con el botón derecho en la barra de menús de Visual Studio para obtener la lista de barras de herramientas. Seleccione **Probar barra de herramientas**.
7. Ahora debería ver la barra de herramientas como un icono a la derecha del icono Buscar en archivos. Al hacer clic en el icono, debería ver un cuadro de mensaje que dice **ToolbarTestCommandPackage**. En el interior de `IDEToolbar.ToolbarTestCommand.MenuItemCallback()`.

Vea también

- [Comandos, menús y barras de herramientas](#)

Añadir una barra de herramientas a una ventana de herramientas

06/04/2020 • 5 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo agregar una barra de herramientas a una ventana de herramientas.

Una barra de herramientas es una franja horizontal o vertical que contiene botones enlazados a comandos. La longitud de una barra de herramientas en una ventana de herramientas es siempre la misma que la anchura o la altura de la ventana de herramientas, dependiendo de dónde esté acoplada la barra de herramientas.

A diferencia de las barras de herramientas del IDE, una barra de herramientas de una ventana de herramientas debe estar acoplada y no se puede mover ni personalizar. Si el `VSPackage` se escribe en código administrado, la barra de herramientas se puede acoplar en cualquier borde.

Para obtener más información sobre cómo agregar una barra de herramientas, consulte [Agregar una barra de herramientas](#).

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una barra de herramientas para una ventana de herramientas

1. Cree un proyecto `TWToolbar` VSIX denominado que tenga un comando de menú denominado `TWTestCommand` y una ventana de herramientas denominada `TestToolWindow`. Para obtener más información, consulte [Crear una extensión con un comando de menú](#) y [Crear una extensión con una ventana de herramientas](#). Debe agregar la plantilla de elemento de comando antes de agregar la plantilla de ventana de herramientas.
2. En `TWTestCommandPackage.vsct`, busque la sección `Symbols`. En el `GuidSymbol` nodo denominado `guidTWTestCommandPackageCmdSet` declarar una barra de herramientas y un grupo de barra de herramientas, como se indica a continuación.

```
<IDSymbol name="TWToolbar" value="0x1000" />
<IDSymbol name="TWToolbarGroup" value="0x1050" />
```

3. En la parte `Commands` superior de `Menus` la sección, cree una sección. Agregue `Menu` un elemento para definir la barra de herramientas.

```
<Menus>
  <Menu guid="guidTWTestCommandPackageCmdSet" id="TWToolbar" type="ToolWindowToolbar">
    <CommandFlag>DefaultDocked</CommandFlag>
    <Strings>
      <ButtonText>Test Toolbar</ButtonText>
      <CommandName>Test Toolbar</CommandName>
    </Strings>
  </Menu>
</Menus>
```

Las barras de herramientas no se pueden anidar como submenús. Por lo tanto, no es posible asignar un elemento primario. Además, no tiene que establecer una prioridad, ya que el usuario puede mover barras de herramientas. Normalmente, la ubicación inicial de una barra de herramientas se define mediante programación, pero se conservan los cambios posteriores realizados por el usuario.

4. En la sección Grupos, defina un grupo que contenga los comandos de la barra de herramientas.

```
<Group guid="guidTWTestCommandPackageCmdSet" id="TWToolbarGroup" priority="0x0000">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TWToolbar" />
</Group>
```

5. En la sección Botones, cambie el elemento primario del elemento Button existente al grupo de barra de herramientas para que se muestre la barra de herramientas.

```
<Button guid="guidTWTestCommandPackageCmdSet" id="TWTestCommandId" priority="0x0100" type="Button">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TWToolbarGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <ButtonText>Invoke TWTestCommand</ButtonText>
  </Strings>
</Button>
```

De forma predeterminada, si una barra de herramientas no tiene comandos, no aparece.

Dado que la nueva barra de herramientas no se agrega automáticamente a la ventana de herramientas, la barra de herramientas debe agregarse explícitamente. Esta técnica se analiza en la sección siguiente.

Agregue la barra de herramientas a la ventana de herramientas

1. En *TWTestCommandPackageGuids.cs* agregue las siguientes líneas.

```
public const string guidTWTestCommandPackageCmdSet = "00000000-0000-0000-0000-0000"; // get the GUID
from the .vsct file
public const int TWToolbar = 0x1000;
```

2. En *TestToolWindow.cs* agregue la siguiente instrucción using.

```
using System.ComponentModel.Design;
```

3. En el TestToolWindow constructor agregue la siguiente línea.

```
this.ToolBar = new CommandID(new Guid(TWTestCommandPackageGuids.guidTWTestCommandPackageCmdSet),
TWTestCommandPackageGuids.TWToolbar);
```

Pruebe la barra de herramientas en la ventana de herramientas

1. Compile la solución y comience la depuración. Debe aparecer la instancia experimental de Visual Studio.
2. En el menú **Ver / Otras ventanas**, haga clic en **Probar Ventana de herramienta** para mostrar la ventana de herramientas.

Debería ver una barra de herramientas (parece el icono predeterminado) en la parte superior izquierda de la ventana de herramientas, justo debajo del título.

3. En la barra de herramientas, haga clic en el icono para mostrar el mensaje **TWTestCommandPackage** Inside **TWToolbar.TWTestCommand.MenuItemCallback()**.

Vea también

- [Añadir una barra de herramientas](#)

Agregue un comando a la barra de herramientas del Explorador de soluciones

06/04/2020 • 7 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo agregar un botón a la barra de herramientas del **Explorador** de soluciones.

Cualquier comando de una barra de herramientas o menú se denomina botón en Visual Studio. Cuando se hace clic en el botón, se ejecuta el código del controlador de comandos. Normalmente, los comandos relacionados se agrupan para formar un grupo. Los menús o barras de herramientas actúan como contenedores para grupos. La prioridad determina el orden en el que aparecen los comandos individuales de un grupo en el menú o en la barra de herramientas. Puede evitar que se muestre un botón en la barra de herramientas o en el menú controlando su visibilidad. Un comando que aparece `<VisibilityConstraints>` en una sección del archivo `.vsct` solo aparece en el contexto asociado. La visibilidad no se puede aplicar a los grupos.

Para obtener más información acerca de los menús, comandos de barra de herramientas y archivos `.vsct`, vea [Comandos, menús y barras de herramientas](#).

NOTE

Utilice archivos de tabla de comandos XML (`.vsct`) en lugar de archivos de configuración de tabla de comandos (`.ctc`) para definir cómo aparecen los menús y comandos en los VSPackages. Para obtener más información, vea [Tabla de comandos de Visual Studio \(. Vsct\)](#).

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una extensión con un comando de menú

Cree un proyecto `SolutionToolbar` VSIX denominado . Agregue una plantilla de elemento de comando de menú denominada `ToolBarButton`. Para obtener información sobre cómo hacerlo, consulte [Crear una extensión con un comando de menú](#).

Agregue un botón a la barra de herramientas del Explorador de soluciones

En esta sección del tutorial se muestra cómo agregar un botón a la barra de herramientas del **Explorador** de soluciones. Cuando se hace clic en el botón, se ejecuta el código del método de devolución de llamada.

1. En el archivo `ToolBarButtonPackage.vsct`, vaya a la `<Symbols>` sección. El `<GuidSymbol>` nodo contiene el grupo de menús y el comando generados por la plantilla de paquete. Agregue `<IDSymbol>` un elemento a este nodo para declarar el grupo que contendrá el comando.

```
<IDSymbol name="SolutionToolbarGroup" value="0x0190"/>
```

2. En `<Groups>` la sección, después de la entrada de grupo existente, defina el nuevo grupo que declaró en el

paso anterior.

```
<Group guid="guidToolBarButtonPackageCmdSet"
  id="SolutionToolBarGroup" priority="0xF000">
  <Parent guid="guidSHLMainMenu" id="IDM_VS_TOOL_PROJWIN"/>
</Group>
```

Establecer el par GUID:ID primario `guidSHLMainMenu` y `IDM_VS_TOOL_PROJWIN` coloca este grupo en la barra de herramientas del **Explorador** de soluciones y establecer un valor de prioridad alta lo coloca después de los otros grupos de comandos.

- En `<Buttons>` la sección, cambie el identificador `<Button>` primario de la entrada generada para reflejar el grupo que definió en el paso anterior. El `<Button>` elemento modificado debe tener este aspecto:

```
<Button guid="guidToolBarButtonPackageCmdSet" id="ToolBarButtonId" priority="0x0100" type="Button">
  <Parent guid="guidToolBarButtonPackageCmdSet" id="SolutionToolBarGroup" />
  <Icon guid="guidImages" id="bmpPicStrikethrough" />
  <Strings>
    <ButtonText>Invoke ToolBarButton</ButtonText>
  </Strings>
</Button>
```

- Compile la solución y comience la depuración. Aparece la instancia experimental.

La barra de **herramientas del Explorador** de soluciones debe mostrar el nuevo botón de comando a la derecha de los botones existentes. El icono del botón es el tachado.

- Haga clic en el nuevo botón.

Se debe mostrar un cuadro de diálogo que tenga el mensaje **ToolBarButtonPackage Inside SolutionToolBar.ToolBarButton.MenuItemCallback()**.

Controlar la visibilidad de un botón

En esta sección del tutorial se muestra cómo controlar la visibilidad de un botón en una barra de herramientas. Al establecer un contexto en uno `<VisibilityConstraints>` o varios proyectos en la sección del archivo *SolutionToolBar.vsct*, se restringe un botón para que aparezca solo cuando un proyecto o proyectos están abiertos.

Para mostrar un botón cuando uno o más proyectos están abiertos

- En `<Buttons>` la sección de *ToolBarButtonPackage.vsct*, agregue `<Button>` dos indicadores de comando al elemento existente, entre las `<Strings>` etiquetas y. `<Icons>`

```
<CommandFlag>DefaultInvisible</CommandFlag>
<CommandFlag>DynamicVisibility</CommandFlag>
```

Los `DefaultInvisible` `DynamicVisibility` indicadores y deben establecerse `<VisibilityConstraints>` para que las entradas de la sección puedan surtir efecto.

- Cree `<VisibilityConstraints>` una sección `<VisibilityItem>` que tenga dos entradas. Coloque la nueva sección `</Commands>` justo después de la etiqueta de cierre.


```
<VisibilityConstraints>
  <VisibilityItem guid="guidToolBarButtonPackageCmdSet"
    id="ToolBarButtonId"
    context="UICONTEXT_SolutionHasSingleProject" />
  <VisibilityItem guid="guidToolBarButtonPackageCmdSet"
    id="ToolBarButtonId"
    context="UICONTEXT_SolutionHasMultipleProjects" />
</VisibilityConstraints>
```

Cada elemento de visibilidad representa una condición en la que se muestra el botón especificado. Para aplicar varias condiciones, debe crear varias entradas para el mismo botón.

3. Compile la solución y comience la depuración. Aparece la instancia experimental.

La barra de **herramientas del Explorador** de soluciones no contiene el botón de tachado.

4. Abra cualquier solución que contenga un proyecto.

El botón de tachado aparece en la barra de herramientas a la derecha de los botones existentes.

5. En el menú **Archivo**, haga clic en **Cerrar solución**. El botón desaparece de la barra de herramientas.

La visibilidad del botón Visual Studio se controla hasta que se carga el VSPackage. Después de cargar el VSPackage, el VSPackage controla la visibilidad del botón. Para obtener más información, vea [MenuCommands vs OleMenuCommands](#).

Vea también

- [Comandos, menús y barras de herramientas](#)

Agregar un controlador de menú a una barra de herramientas

06/04/2020 • 8 minutes to read • [Edit Online](#)

Este tutorial se basa en el agregar una barra de [herramientas a una ventana](#) de herramientas tutorial y muestra cómo agregar un controlador de menú a la barra de herramientas de la ventana de herramientas. Los pasos que se muestran aquí también se pueden aplicar a la barra de herramientas que se crea en el tutorial [Agregar una barra de herramientas](#).

Un controlador de menú es un control dividido. El lado izquierdo del controlador de menú muestra el último comando utilizado y puede ejecutarlo haciendo clic en él. El lado derecho del controlador de menú es una flecha que, al hacer clic, abre una lista de comandos adicionales. Al hacer clic en un comando de la lista, el comando se ejecuta y reemplaza el comando en el lado izquierdo del controlador de menú. De esta manera, el controlador de menú funciona como un botón de comando que siempre muestra el último comando utilizado de una lista.

Los controladores de menú pueden aparecer en los menús, pero se utilizan con mayor frecuencia en las barras de herramientas.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un controlador de menú

1. Siga los procedimientos descritos en [Agregar una barra de herramientas a una ventana](#) de herramientas para crear una ventana de herramientas que tenga una barra de herramientas.
2. En *TWTestCommandPackage.vsct*, vaya a la sección Símbolos. En el guidSymbol elemento denominado **guidTWTestCommandPackageCmdSet**, declare el controlador de menú, grupo de controladores de menú y tres elementos de menú.

```
<IDSymbol name="TestMenuController" value="0x1300" /><IDSymbol name="TestMenuControllerGroup" value="0x1060" /><IDSymbol name="cmdidMCItem1" value="0x0130" /><IDSymbol name="cmdidMCItem2" value="0x0131" /><IDSymbol name="cmdidMCItem3" value="0x0132" />
```

3. En la sección Menús, después de la última entrada de menú, defina el controlador de menú como un menú.

```
<Menu guid="guidTWTestCommandPackageCmdSet" id="TestMenuController" priority="0x0100" type="MenuController">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TWToolbarGroup" />
  <CommandFlag>IconAndText</CommandFlag>
  <CommandFlag>TextChanges</CommandFlag>
  <CommandFlag>TextIsAnchorCommand</CommandFlag>
  <Strings>
    <ButtonText>Test Menu Controller</ButtonText>
    <CommandName>Test Menu Controller</CommandName>
  </Strings>
</Menu>
```

Las `TextChanged` `TextIsAnchorCommand` marcas y deben incluirse para permitir que el controlador de menú refleje el último comando seleccionado.

- En la sección Grupos, después de la última entrada de grupo, agregue el grupo de controladores de menú.

```
<Group guid="guidTWTestCommandPackageCmdSet" id="TestMenuControllerGroup" priority="0x000">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TestMenuController" />
</Group>
```

Al establecer el controlador de menú como el principal, los comandos colocados en este grupo aparecen en el controlador de menú. Se `priority` omite el atributo, que lo establece en el valor predeterminado de 0, porque es el único grupo en el controlador de menú.

- En la sección Botones, después de la última entrada de botón, agregue un elemento Button para cada uno de los elementos de menú.

```
<Button guid="guidTWTestCommandPackageCmdSet" id="cmdidMCItem1" priority="0x0000" type="Button">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TestMenuControllerGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <CommandFlag>IconAndText</CommandFlag>
  <Strings>
    <ButtonText>MC Item 1</ButtonText>
    <CommandName>MC Item 1</CommandName>
  </Strings>
</Button>
<Button guid="guidTWTestCommandPackageCmdSet" id="cmdidMCItem2" priority="0x0100" type="Button">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TestMenuControllerGroup" />
  <Icon guid="guidImages" id="bmpPic2" />
  <CommandFlag>IconAndText</CommandFlag>
  <Strings>
    <ButtonText>MC Item 2</ButtonText>
    <CommandName>MC Item 2</CommandName>
  </Strings>
</Button>
<Button guid="guidTWTestCommandPackageCmdSet" id="cmdidMCItem3" priority="0x0200" type="Button">
  <Parent guid="guidTWTestCommandPackageCmdSet" id="TestMenuControllerGroup" />
  <Icon guid="guidImages" id="bmpPicSearch" />
  <CommandFlag>IconAndText</CommandFlag>
  <Strings>
    <ButtonText>MC Item 3</ButtonText>
    <CommandName>MC Item 3</CommandName>
  </Strings>
</Button>
```

- En este punto, puede mirar el controlador de menú. Compile la solución y comience la depuración. Debería ver la instancia experimental.
 - En el menú **Ver / Otras ventanas**, abra **Test ToolWindow**.
 - El controlador de menú aparece en la barra de herramientas de la ventana de herramientas.
 - Haga clic en la flecha en el lado derecho del controlador de menú para ver los tres comandos posibles.

Tenga en cuenta que al hacer clic en un comando, el título del controlador de menú cambia para mostrar ese comando. En la siguiente sección, agregaremos el código para activar estos comandos.

Implementar los comandos del controlador de menú

- En *TWTestCommandPackageGuids.cs*, agregue los iDE de comando para los tres elementos de menú después de los ides de comando existentes.

```
public const int cmdidMCItem1 = 0x130;
public const int cmdidMCItem2 = 0x131;
public const int cmdidMCItem3 = 0x132;
```

2. En *TWTestCommand.cs*, agregue el siguiente `TwTestCommand` código en la parte superior de la clase.

```
private int currentMCCommand; // The currently selected menu controller command
```

3. En el `TwTestCommand` constructor, después `AddCommand` de la última llamada al método, agregue código para enrutar los eventos de cada comando a través de los mismos controladores.

```
for (int i = TWTestCommandPackageGuids.cmdidMCItem1; i <=
    TWTestCommandPackageGuids.cmdidMCItem3; i++)
{
    CommandID cmdID = new
    CommandID(new Guid(TWTestCommandPackageGuids.guidTWTestCommandPackageCmdSet), i);
    OleMenuCommand mc = new OleMenuCommand(new
        EventHandler(OnMCItemClicked), cmdID);
    mc.BeforeQueryStatus += new EventHandler(OnMCItemQueryStatus);
    commandService.AddCommand(mc);
    // The first item is, by default, checked.
    if (TWTestCommandPackageGuids.cmdidMCItem1 == i)
    {
        mc.Checked = true;
        this.currentMCCommand = i;
    }
}
```

4. Agregue un controlador de eventos a la **clase** `TwTestCommand` para marcar el comando seleccionado como activado.

```
private void OnMCItemQueryStatus(object sender, EventArgs e)
{
    OleMenuCommand mc = sender as OleMenuCommand;
    if (null != mc)
    {
        mc.Checked = (mc.CommandID.ID == this.currentMCCommand);
    }
}
```

5. Agregue un controlador de eventos que muestre un cuadro de mensajes cuando el usuario seleccione un comando en el controlador de menú:

```

private void OnMCItemClicked(object sender, EventArgs e)
{
    OleMenuCommand mc = sender as OleMenuCommand;
    if (null != mc)
    {
        string selection;
        switch (mc.CommandID.ID)
        {
            case c.cmdidMCItem1:
                selection = "Menu controller Item 1";
                break;

            case TWTestCommandPackageGuids.cmdidMCItem2:
                selection = "Menu controller Item 2";
                break;

            case TWTestCommandPackageGuids.cmdidMCItem3:
                selection = "Menu controller Item 3";
                break;

            default:
                selection = "Unknown command";
                break;
        }
        this.currentMCCommand = mc.CommandID.ID;

        IVsUIShell uiShell =
            (IVsUIShell) ServiceProvider.GetService(typeof(SVsUIShell));
        Guid clsid = Guid.Empty;
        int result;
        uiShell.ShowMessageBox(
            0,
            ref clsid,
            "Test Tool Window Toolbar Package",
            string.Format(CultureInfo.CurrentCulture,
                "You selected {0}", selection),
            string.Empty,
            0,
            OLEMSGBUTTON.OLEMSGBUTTON_OK,
            OLEMSGDEFBUTTON.OLEMSGDEFBUTTON_FIRST,
            OLEMSGICON.OLEMSGICON_INFO,
            0,
            out result);
    }
}

```

Probar el controlador de menú

1. Compile la solución y comience la depuración. Debería ver la instancia experimental.
2. Abra la ventana **Probar ToolWindow** en el menú **Ver / Otras ventanas**.

El controlador de menú aparece en la barra de herramientas de la ventana de herramientas y muestra **EL elemento 1 de MC**.

3. Haga clic en el botón del controlador de menú a la izquierda de la flecha.

Debería ver tres elementos, el primero de los cuales está seleccionado y tiene un cuadro de resaltado alrededor de su icono. Haga clic en **MC Elemento 3**.

Aparece un cuadro de diálogo con el mensaje **Seleccionado Elemento 3** del controlador de menú. Observe que el mensaje corresponde al texto del botón del controlador de menú. El botón del controlador de menú ahora muestra **el elemento 3 de MC**.

Vea también

- [Adición de una barra de herramientas a una ventana de herramientas](#)
- [Adición de una barra de herramientas](#)

Añadir elementos de menú de forma dinámica

06/04/2020 • 17 minutes to read • [Edit Online](#)

Puede agregar elementos de menú en `DynamicItemStart` tiempo de ejecución especificando el indicador de comando en una definición de botón de marcador de posición en el archivo de tabla de comandos de Visual Studio (`.vsct`) y, a continuación, definiendo (en código) el número de elementos de menú para mostrar y controlar los comandos. Cuando se carga el `VSPackage`, el marcador de posición se reemplaza por los elementos de menú dinámico.

Visual Studio usa listas dinámicas en la lista **Usado más recientemente** (MRU), que muestra los nombres de los documentos que se han abierto recientemente, y la lista de **Windows**, que muestra los nombres de las ventanas que están abiertas actualmente. El `DynamicItemStart` indicador en una definición de comando especifica que el comando es un marcador de posición hasta que se abre el `VSPackage`. Cuando se abre el `VSPackage`, el marcador de posición se reemplaza por 0 o más comandos que se crean en tiempo de ejecución y se agregan a la lista dinámica. Es posible que no pueda ver la posición en el menú donde aparece la lista dinámica hasta que se abre el `VSPackage`. Para rellenar la lista dinámica, Visual Studio pide al `VSPackage` que busque un comando con un identificador cuyos primeros caracteres son los mismos que el identificador del marcador de posición. Cuando Visual Studio encuentra un comando coincidente, agrega el nombre del comando a la lista dinámica. A continuación, incrementa el ID y busca otro comando coincidente para agregar a la lista dinámica hasta que no haya más comandos dinámicos.

En este tutorial se muestra cómo establecer el proyecto de inicio en una solución de Visual Studio con un comando en la barra de herramientas del **Explorador** de soluciones. Utiliza un controlador de menú que tiene una lista desplegable dinámica de los proyectos en la solución activa. Para evitar que este comando aparezca cuando no hay ninguna solución abierta o cuando la solución abierta solo tiene un proyecto, el `VSPackage` se carga solo cuando una solución tiene varios proyectos.

Para obtener más información acerca de los archivos `.vsct`, vea Archivos de tabla de [comandos de Visual Studio \(.vsct\)](#).

Crear una extensión con un comando de menú

1. Cree un proyecto `DynamicMenuItems` VSIX denominado .
2. Cuando se abra el proyecto, agregue una plantilla de elemento de comando personalizada y asíse, **así saber, esCladynamicMenu**. Para obtener más información, consulte [Crear una extensión con un comando de menú](#).

Configuración de los elementos en el archivo `.vsct`

Para crear un controlador de menú con elementos de menú dinámicos en una barra de herramientas, especifique los siguientes elementos:

- Dos grupos de comandos, uno que contiene el controlador de menú y otro que contiene los elementos de menú en la lista desplegable
- Un elemento de menú de tipo `MenuController`
- Dos botones, uno que actúa como marcador de posición para los elementos de menú y otro que proporciona el icono y la información sobre herramientas en la barra de herramientas.

1. En `DynamicMenuPackage.vsct`, defina los elementos de comando. Vaya a la sección Símbolos y reemplace

los elementos IDSymbol en el bloque **guidDynamicMenuPackageCmdSet** GuidSymbol. Debe definir elementos IDSymbol para los dos grupos, el controlador de menú, el comando de marcador de posición y el comando de anclaje.

```
<GuidSymbol name="guidDynamicMenuPackageCmdSet" value="{ your GUID here }">
  <IDSymbol name="MyToolBarItemGroup" value="0x1020" />
  <IDSymbol name="MyMenuControllerGroup" value="0x1025" />
  <IDSymbol name="MyMenuController" value="0x1030"/>
  <IDSymbol name="cmdidMyAnchorCommand" value="0x0103" />
  <!-- NOTE: The following command expands at run time to some number of ids.
  Try not to place command ids after it (e.g. 0x0105, 0x0106).
  If you must add a command id after it, make the gap very large (e.g. 0x200) -->
  <IDSymbol name="cmdidMyDynamicStartCommand" value="0x0104" />
</GuidSymbol>
```

2. En la sección Grupos, elimine los grupos existentes y agregue los dos grupos que acaba de definir:

```
<Groups>
  <!-- The group that adds the MenuController on the Solution Explorer toolbar.
  The 0x4000 priority adds this group after the group that contains the
  Preview Selected Items button, which is normally at the far right of the toolbar. -->
  <Group guid="guidDynamicMenuPackageCmdSet" id="MyToolBarItemGroup" priority="0x4000" >
    <Parent guid="guidSHLMainMenu" id="IDM_VS_TOOL_PROJWIN" />
  </Group>
  <!-- The group for the items on the MenuController drop-down. It is added to the MenuController
  submenu. -->
  <Group guid="guidDynamicMenuPackageCmdSet" id="MyMenuControllerGroup" priority="0x4000" >
    <Parent guid="guidDynamicMenuPackageCmdSet" id="MyMenuController" />
  </Group>
</Groups>
```

Agregue el MenuController. Establezca el indicador de comando DynamicVisibility, ya que no siempre está visible. El ButtonText no se muestra.

```
<Menus>
  <!-- The MenuController to display on the Solution Explorer toolbar.
  Place it in the ToolBarItemGroup.-->
  <Menu guid="guidDynamicMenuPackageCmdSet" id="MyMenuController" priority="0x1000"
  type="MenuController">
    <Parent guid="guidDynamicMenuPackageCmdSet" id="MyToolBarItemGroup" />
    <CommandFlag>DynamicVisibility</CommandFlag>
    <Strings>
      <ButtonText></ButtonText>
    </Strings>
  </Menu>
</Menus>
```

3. Agregue dos botones, uno como marcador de posición para los elementos de menú dinámico y otro como delimitador para el MenuController.

El elemento primario del botón de marcador de posición es **MyMenuControllerGroup**. Agregue los indicadores de comando DynamicItemStart, DynamicVisibility y TextChanges al botón de marcador de posición. El ButtonText no se muestra.

El botón de anclaje mantiene el icono y el texto de información sobre herramientas. El elemento primario del botón de anclaje también es **MyMenuControllerGroup**. Agregue el indicador de comando NoShowOnMenuController para asegurarse de que el botón no aparece realmente en la lista desplegable del controlador de menú y el indicador de comando FixMenuController para convertirlo en el delimitador permanente.


```

<!-- The placeholder for the dynamic items that expand to N items at run time. -->
<Buttons>
  <Button guid="guidDynamicMenuPackageCmdSet" id="cmdidMyDynamicStartCommand" priority="0x1000" >
    <Parent guid="guidDynamicMenuPackageCmdSet" id="MyMenuControllerGroup" />
    <CommandFlag>DynamicItemStart</CommandFlag>
    <CommandFlag>DynamicVisibility</CommandFlag>
    <CommandFlag>TextChanges</CommandFlag>
    <!-- This text does not appear. -->
    <Strings>
      <ButtonText>Project</ButtonText>
    </Strings>
  </Button>

  <!-- The anchor item to supply the icon/tooltip for the MenuController -->
  <Button guid="guidDynamicMenuPackageCmdSet" id="cmdidMyAnchorCommand" priority="0x0000" >
    <Parent guid="guidDynamicMenuPackageCmdSet" id="MyMenuControllerGroup" />
    <!-- This is the icon that appears on the Solution Explorer toolbar. -->
    <Icon guid="guidImages" id="bmpPicArrows"/>
    <!-- Do not show on the menu controller's drop down list-->
    <CommandFlag>NoShowOnMenuController</CommandFlag>
    <!-- Become the permanent anchor item for the menu controller -->
    <CommandFlag>FixMenuController</CommandFlag>
    <!-- The text that appears in the tooltip.-->
    <Strings>
      <ButtonText>Set Startup Project</ButtonText>
    </Strings>
  </Button>
</Buttons>

```

4. Agregue un icono al proyecto (en la carpeta *Resources*) y, a continuación, agregue la referencia a él en el archivo *.vsct*. En este tutorial, usamos el icono Flechas que se incluye en la plantilla de proyecto.
5. Agregue una sección *VisibilityConstraints* fuera de la sección *Comandos* justo antes de la sección *Símbolos*. (Puede recibir una advertencia si la agrega después de *Símbolos*.) En esta sección se asegura de que el controlador de menú solo aparece cuando se carga una solución con varios proyectos.

```

<VisibilityConstraints>
  <!--Make the MenuController show up only when there is a solution with more than one project loaded-->
  <VisibilityItem guid="guidDynamicMenuPackageCmdSet" id="MyMenuController"
context="UICONTEXT_SolutionHasMultipleProjects"/>
</VisibilityConstraints>

```

Implementar el comando de menú dinámico

Cree una clase de comando de [OleMenuCommand](#) menú dinámico que herede de [OleMenuCommand](#). En esta implementación, el constructor especifica un predicado que se usará para los comandos coincidentes. Debe invalidar [DynamicItemMatch](#) el método para utilizar [MatchedCommandId](#) este predicado para establecer la propiedad, que identifica el comando que se va a invocar.

1. Cree un nuevo archivo de clase de C# denominado *DynamicItemMenuCommand.cs*
[OleMenuCommand](#) agregue una clase denominada **DynamicItemMenuCommand** que herede de:

```

class DynamicItemMenuCommand : OleMenuCommand
{
    // ...
}

```

2. Agregue lo siguiente mediante directivas:

```
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;
using System.ComponentModel.Design;
```

3. Agregue un campo privado para almacenar el predicado de coincidencia:

```
private Predicate<int> matches;
```

4. Agregue un constructor que [OleMenuCommand](#) herede del constructor y [BeforeQueryStatus](#) especifique un controlador de comandos y un controlador. Agregue un predicado para hacer coincidir el comando:

```
public DynamicItemMenuCommand(CommandID rootId, Predicate<int> matches, EventHandler invokeHandler,
    EventHandler beforeQueryStatusHandler)
    : base(invokeHandler, null /*changeHandler*/, beforeQueryStatusHandler, rootId)
{
    if (matches == null)
    {
        throw new ArgumentNullException("matches");
    }

    this.matches = matches;
}
```

5. Invalide [DynamicItemMatch](#) el método para que llame [MatchedCommandId](#) al predicado de coincidencias y establezca la propiedad:

```
public override bool DynamicItemMatch(int cmdId)
{
    // Call the supplied predicate to test whether the given cmdId is a match.
    // If it is, store the command id in MatchedCommandId
    // for use by any BeforeQueryStatus handlers, and then return that it is a match.
    // Otherwise clear any previously stored matched cmdId and return that it is not a match.
    if (this.matches(cmdId))
    {
        this.MatchedCommandId = cmdId;
        return true;
    }

    this.MatchedCommandId = 0;
    return false;
}
```

Añadir el comando

El `DynamicMenu` constructor es donde se configuran los comandos de menú, incluidos los menús dinámicos y los elementos de menú.

1. En *DynamicMenuPackage.cs*, agregue el GUID del conjunto de comandos y el identificador de comando:

```
public const string guidDynamicMenuPackageCmdSet = "00000000-0000-0000-0000-00000000"; // get the GUID
from the .vsct file
public const uint cmdidMyCommand = 0x104;
```

2. En el archivo *DynamicMenu.cs*, agregue las siguientes directivas using:

```
using EnvDTE;
using EnvDTE80;
using System.ComponentModel.Design;
```

3. En `DynamicMenu` la clase, agregue un campo privado `dte2`.

```
private DTE2 dte2;
```

4. Agregue un campo `rootItemId` privado:

```
private int rootItemId = 0;
```

5. En el `DynamicMenu` constructor, agregue el comando menu. En la siguiente sección definiremos el `BeforeQueryStatus` controlador de comandos, el controlador de eventos y el predicado de coincidencia.

```
private DynamicMenu(Package package)
{
    if (package == null)
    {
        throw new ArgumentNullException(nameof(package));
    }

    this.package = package;

    OleMenuCommandService commandService = this.ServiceProvider.GetService(typeof(IMenuCommandService))
as OleMenuCommandService;
    if (commandService != null)
    {
        // Add the DynamicItemMenuCommand for the expansion of the root item into N items at run time.
        CommandID dynamicItemRootId = new CommandID(new
Guid(DynamicMenuPackageGuids.guidDynamicMenuPackageCmdSet),
(int)DynamicMenuPackageGuids.cmdidMyCommand);
        DynamicItemMenuCommand dynamicMenuCommand = new DynamicItemMenuCommand(dynamicItemRootId,
            IsValidDynamicItem,
            OnInvokedDynamicItem,
            OnBeforeQueryStatusDynamicItem);
        commandService.AddCommand(dynamicMenuCommand);
    }

    dte2 = (DTE2)this.ServiceProvider.GetService(typeof(DTE));
}
```

Implementar los controladores

Para implementar elementos de menú dinámicos en un controlador de menú, debe controlar el comando cuando se hace clic en un elemento dinámico. También debe implementar la lógica que establece el estado del elemento de menú. Agregue los controladores `DynamicMenu` a la clase.

1. Para implementar el comando Establecer proyecto de inicio, agregue el controlador de eventos `OnInvokedDynamicItem`. Busca el proyecto cuyo nombre es el mismo que el texto del comando que se ha invocado y [StartupProjects](#) lo establece como el proyecto de inicio estableciendo su ruta de acceso absoluta en la propiedad.

```
private void OnInvokedDynamicItem(object sender, EventArgs args)
{
    DynamicItemMenuCommand invokedCommand = (DynamicItemMenuCommand)sender;
    // If the command is already checked, we don't need to do anything
    if (invokedCommand.Checked)
        return;

    // Find the project that corresponds to the command text and set it as the startup project
    var projects = dte2.Solution.Projects;
    foreach (Project proj in projects)
    {
        if (invokedCommand.Text.Equals(proj.Name))
        {
            dte2.Solution.SolutionBuild.StartupProjects = proj.FullName;
            return;
        }
    }
}
```

2. Agregue `OnBeforeQueryStatusDynamicItem` al controlador de eventos. Este es el controlador `QueryStatus` al que se llama antes de un evento. Determina si el elemento de menú es un elemento "real", es decir, no el elemento de marcador de posición, y si el elemento ya está comprobado (lo que significa que el proyecto ya está establecido como el proyecto de inicio).

```
private void OnBeforeQueryStatusDynamicItem(object sender, EventArgs args)
{
    DynamicItemMenuCommand matchedCommand = (DynamicItemMenuCommand)sender;
    matchedCommand.Enabled = true;
    matchedCommand.Visible = true;

    // Find out whether the command ID is 0, which is the ID of the root item.
    // If it is the root item, it matches the constructed DynamicItemMenuCommand,
    // and IsValidDynamicItem won't be called.
    bool isRootItem = (matchedCommand.MatchedCommandId == 0);

    // The index is set to 1 rather than 0 because the Solution.Projects collection is 1-based.
    int indexForDisplay = (isRootItem ? 1 : (matchedCommand.MatchedCommandId - (int)
DynamicMenuPackageGuids.cmdidMyCommand) + 1);

    matchedCommand.Text = dte2.Solution.Projects.Item(indexForDisplay).Name;

    Array startupProjects = (Array)dte2.Solution.SolutionBuild.StartupProjects;
    string startupProject =
System.IO.Path.GetFileNameWithoutExtension((string)startupProjects.GetValue(0));

    // Check the command if it isn't checked already selected
    matchedCommand.Checked = (matchedCommand.Text == startupProject);

    // Clear the ID because we are done with this item.
    matchedCommand.MatchedCommandId = 0;
}
```

Implemente el predicado de coincidencia de identificador de comando

Ahora implemente el predicado de coincidencia. Necesitamos determinar dos cosas: en primer lugar, si el identificador de comando es válido (es mayor o igual que el identificador de comando declarado) y, en segundo lugar, si especifica un proyecto posible (es menor que el número de proyectos de la solución).

```
private bool IsValidDynamicItem(int commandId)
{
    // The match is valid if the command ID is >= the id of our root dynamic start item
    // and the command ID minus the ID of our root dynamic start item
    // is less than or equal to the number of projects in the solution.
    return (commandId >= (int)DynamicMenuPackageGuids.cmdidMyCommand) && ((commandId -
(int)DynamicMenuPackageGuids.cmdidMyCommand) < dte2.Solution.Projects.Count);
}
```

Establezca el VSPackage para cargar solo cuando una solución tiene varios proyectos

Dado que el comando **Establecer proyecto** de inicio no tiene sentido a menos que la solución activa tenga más de un proyecto, puede establecer el VSPackage para que se cargue automáticamente solo en ese caso. Se [ProvideAutoLoadAttribute](#) usa junto con [SolutionHasMultipleProjects](#) el contexto de la interfaz de usuario. En el archivo *DynamicMenuPackage.cs* agregue los siguientes atributos a la clase *DynamicMenuPackage*:

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)]
[ProvideMenuResource("Menus.ctmenu", 1)]
[ProvideAutoLoad(UIContextGuids.SolutionHasMultipleProjects)]
[Guid(DynamicMenuPackage.PackageGuidString)]
public sealed class DynamicMenuItemsPackage : Package
{
}
```

Pruebe el comando set startup project

Ahora puede probar el código.

1. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.
2. En la instancia experimental, abra una solución que tenga más de un proyecto.

Debería ver el icono de flecha en la barra de herramientas del Explorador de **soluciones**. Al expandirlo, deben aparecer elementos de menú que representan los diferentes proyectos de la solución.

3. Al comprobar uno de los proyectos, se convierte en el proyecto de inicio.
4. Al cerrar la solución o abrir una solución que solo tiene un proyecto, el icono de la barra de herramientas debe desaparecer.

Vea también

- [Comandos, menús y barras de herramientas](#)
- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Ampliar y personalizar las ventanas de herramientas

06/04/2020 • 5 minutes to read • [Edit Online](#)

Visual Studio proporciona varios tipos diferentes de ventanas, por ejemplo, ventanas de herramientas, ventanas de documento y ventanas de cuadro de diálogo. Otras ventanas, como la ventana **Propiedades**, la ventana **Salida** y la ventana **Lista** de tareas, son tipos de ventanas de herramientas.

Ventanas de herramientas

Las ventanas de herramientas de Visual Studio suelen ser ventanas de solo lectura que no están basadas en archivos. En esto se diferencian de las ventanas de documento, que muestran archivos en modo de lectura y escritura. Las ventanas **Cuadro de herramientas**, **Explorador de soluciones**, **Propiedades** y el **Explorador web** son ejemplos de ventanas de herramientas.

Para saber cómo crear una ventana de herramientas sencilla, consulte [Agregar una ventana](#) de herramientas .

Para registrar una ventana de herramientas con Visual Studio, vea [Registrar una ventana](#) de herramientas .

Las ventanas de herramientas son de instancia única de forma predeterminada, lo que significa que solo una instancia de la ventana de herramientas puede estar abierta a la vez. Cuando se abre una ventana de herramientas de instancia única, permanece abierta hasta que se cierra el IDE. Al cerrar una ventana de herramientas de instancia única, solo cambia su visibilidad. También puede crear ventanas de herramientas de varias instancias, de forma que varias instancias de la ventana puedan abrir simultáneamente. Consulte [Crear una ventana de herramientas de varias instancias](#) para obtener más información.

Las ventanas de herramientas pueden ser *dinámicas*, lo que significa que son visibles siempre que se aplica su contexto de interfaz de usuario relacionado. El uso de la visibilidad automática puede reducir la cantidad de ventanas en el IDE. Para obtener más información, consulte [Abrir una ventana de herramientas dinámicas](#).

Las ventanas de herramientas se pueden acoplar, ser flotantes o con pestañas en el marco del documento. El IDE proporciona el marco de ventana de herramientas y se usa para controlar el tamaño, la ubicación, el estado de acoplamiento y otras propiedades persistentes. El panel de la ventana de herramientas muestra el contenido. La ubicación y tamaño predeterminados se aplican solo cuando la ventana de herramientas se abre por primera vez; después se guarda el estado de la ventana de herramientas.

Los paneles de la ventana de herramientas pueden hospedar controles de usuario WPF y admiten barras de herramientas. Puede invalidar la propiedad `Window` para devolver el identificador del control hospedado.

Puede agregar muchas características diferentes a las ventanas de herramientas. Por ejemplo, puede agregar una barra de herramientas: Agregar una barra de [herramientas a una ventana](#) de herramientas o un menú contextual: Agregar un menú contextual en una ventana de [herramientas](#). Puede agregar un control de búsqueda que le permite buscar elementos dentro de la ventana de herramientas: [Agregar búsqueda a una ventana](#) de herramientas .

Puede suscribirse a eventos de ventana de herramientas: [Suscríbase a un evento](#).

Ampliar las ventanas de herramientas existentes

Puede agregar información sobre la ventana de herramientas a una nueva página **Opciones** y una nueva configuración en la página **Propiedades**, escribir en las ventanas **Lista de tareas** y **Salida**. Para obtener más información, vea [Extender las ventanas Propiedades, Lista de tareas, Salida y Opciones](#).

Cuadros de diálogo modales

En una extensión de Visual Studio debe crear [Microsoft.VisualStudio.PlatformUI.DialogWindow](#) cuadros de diálogo modales derivando de , lo que le permite controlarlos y el resto de la interfaz de usuario. Para obtener más información, consulte [Crear y administrar cuadros](#) de diálogo modales .

Vea también

- [Crear una extensión con una ventana de herramientas](#)
- [Ampliar proyectos](#)
- [Ampliar las soluciones](#)

Crear una extensión con una ventana de herramientas

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este procedimiento, aprenderá a usar la plantilla de proyecto VSIX y la plantilla de elemento De la ventana de herramientas **personalizada** para crear una extensión con una ventana de herramientas.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear una ventana de herramientas

1. Cree un proyecto VSIX denominado **FirstWindow**. Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Cuando se abra el proyecto, agregue una plantilla de elemento de ventana de herramientas denominada **MyWindow**. En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento** , vaya a **Extensibilidad de Visual C- >** y seleccione **Ventana de herramientas personalizadas**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de la ventana de herramientas a *MyWindow.cs*.
3. Compile la solución y comience la depuración.

Aparece la instancia experimental de Visual Studio. Para obtener más información acerca de la instancia experimental, consulte [La instancia experimental](#).

4. En la instancia experimental, vaya a **Ver > otras ventanas**.

Debería ver un elemento de menú para **MyWindow**. Haga clic en él.

Debería ver una ventana de herramientas con el título **MyWindow** y un botón que dice **Click Me!**.

Añadir una ventana de herramientas

06/04/2020 • 12 minutes to read • [Edit Online](#)

En este tutorial aprenderá a crear una ventana de herramientas e integrarla en Visual Studio de las siguientes maneras:

- Agregue un control a la ventana de herramientas.
- Agregue una barra de herramientas a una ventana de herramientas.
- Agregue un comando a la barra de herramientas.
- Implemente los comandos.
- Establezca la posición predeterminada para la ventana de herramientas.

Prerrequisitos

El SDK de Visual Studio se incluye como una característica opcional en la instalación de Visual Studio. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una ventana de herramientas

1. Cree un proyecto denominado **FirstToolWin** con la plantilla VSIX y agregue una plantilla de elemento de ventana de herramientas personalizada denominada **FirstToolWindow**.

NOTE

Para obtener más información sobre cómo crear una extensión con una ventana de herramientas, consulte [Crear una extensión con una ventana de herramientas](#).

Añadir un control a la ventana de herramientas

1. Quite el control predeterminado. Abra *FirstToolWindowControl.xaml* y elimine el **Click Me!**.
2. En el cuadro de **herramientas**, expanda el **todos los controles WPF** sección y arrastre el **elemento multimedia** control a la **FirstToolWindowControl** formulario. Seleccione el control y, en la ventana **Propiedades**, asigne a este elemento el nombre **mediaElement1**.

Agregue una barra de herramientas a la ventana de herramientas

Al agregar una barra de herramientas de la siguiente manera, garantiza que sus degradados y colores son coherentes con el resto del IDE.

1. En el **Explorador de soluciones**, abra *FirstToolWindowPackage.vsct*. El archivo *.vsct* define los elementos de la interfaz gráfica de usuario (GUI) en la ventana de herramientas mediante XML.
2. En `<Symbols>` la sección, `<GuidSymbol>` busque `name` el `guidFirstToolWindowPackageCmdSet` nodo cuyo atributo es `.` Agregue los `<IDSymbol>` dos elementos `<IDSymbol>` siguientes a la lista de elementos de este nodo para definir una barra de herramientas y un grupo de barras de herramientas.

```
<IDSymbol name="ToolbarID" value="0x1000" />
<IDSymbol name="ToolbarGroupID" value="0x1001" />
```

3. Justo encima de `<Buttons>` `<Menus>` la sección, cree una sección que se asemeje a esto:

```
<Menus>
  <Menu guid="guidFirstToolWindowPackageCmdSet" id="ToolbarID" priority="0x0000"
type="ToolWindowToolbar">
  <Parent guid="guidFirstToolWindowPackageCmdSet" id="ToolbarID" />
  <Strings>
    <ButtonText>Tool Window Toolbar</ButtonText>
    <CommandName>Tool Window Toolbar</CommandName>
  </Strings>
</Menu>
</Menus>
```

Hay varios tipos diferentes de menú. Este menú es una barra de herramientas `type` en una ventana de herramientas, definida por su atributo. La `guid` `id` configuración y la configuración componen el ID completo de la barra de herramientas. Normalmente, `<Parent>` el de un menú es el grupo contenedor. Sin embargo, una barra de herramientas se define como su propio elemento primario. Por lo tanto, se `<Menu>` utiliza `<Parent>` el mismo identificador para los elementos y. El `priority` atributo es simplemente '0'.

4. Las barras de herramientas se asemejan a los menús de muchas maneras. Por ejemplo, al igual que un menú puede tener grupos de comandos, las barras de herramientas también pueden tener grupos. (En los menús, los grupos de comandos están separados por líneas horizontales. En las barras de herramientas, los grupos no están separados por divisores visuales.)

Agregue `<Groups>` una sección `<Group>` que contenga un elemento. Esto define el grupo cuyo `<Symbols>` identificador declaró en la sección. Agregue `<Groups>` la sección `<Menus>` justo después de la sección.

```
<Groups>
  <Group guid="guidFirstToolWindowPackageCmdSet" id="ToolbarGroupID" priority="0x0000">
    <Parent guid="guidFirstToolWindowPackageCmdSet" id="ToolbarID" />
  </Group>
</Groups>
```

Al establecer el GUID principal y el identificador en el GUID y el identificador de la barra de herramientas, agregue el grupo a la barra de herramientas.

Agregue un comando a la barra de herramientas

Agregue un comando a la barra de herramientas, que se muestra como un botón.

1. En `<Symbols>` la sección, declare los siguientes elementos `IDSymbol` justo después de las declaraciones de grupo de barra de herramientas y barra de herramientas.

```
<IDSymbol name="cmdidWindowsMedia" value="0x0100" />
<IDSymbol name="cmdidWindowsMediaOpen" value="0x132" />
```

2. Agregue un elemento `<Buttons>` `Button` dentro de la sección. Este elemento aparecerá en la barra de herramientas en la ventana de herramientas, con un icono de **búsqueda** (lupa).

```
<Button guid="guidFirstToolWindowPackageCmdSet" id="cmdidWindowsMediaOpen" priority="0x0101"
type="Button">
  <Parent guid="guidFirstToolWindowPackageCmdSet" id="ToolbarGroupID"/>
  <Icon guid="guidImages" id="bmpPicSearch" />
  <Strings>
    <CommandName>cmdidWindowsMediaOpen</CommandName>
    <ButtonText>Load File</ButtonText>
  </Strings>
</Button>
```

3. Abra *FirstToolWindowCommand.cs* y agregue las siguientes líneas en la clase justo después de los campos existentes.

```
public const string guidFirstToolWindowPackageCmdSet = "00000000-0000-0000-0000-0000"; // get the GUID
from the .vsct file
public const uint cmdidWindowsMedia = 0x100;
public const int cmdidWindowsMediaOpen = 0x132;
public const int ToolbarID = 0x1000;
```

Esto hace que los comandos estén disponibles en el código.

Agregue una propiedad MediaPlayer a FirstToolWindowControl

Desde los controladores de eventos para los controles de barra de herramientas, el código debe ser capaz de tener acceso a la Media Player control, que es un elemento secundario de la FirstToolWindowControl clase.

En el **Explorador** de soluciones , haga clic con el botón secundario en *FirstToolWindowControl.xaml*, haga clic en **Ver código** y agregue el código siguiente a la clase FirstToolWindowControl .

```
public System.Windows.Controls.MediaElement MediaPlayer
{
    get { return mediaElement1; }
}
```

Crear una instancia de la ventana de herramientas y la barra de herramientas

Agregue una barra de herramientas y un comando de menú que invoque el cuadro de diálogo **Abrir archivo** y reproduzca el archivo multimedia seleccionado.

1. Abra *FirstToolWindow.cs* y `using` agregue las siguientes directivas:

```
using System.ComponentModel.Design;
using System.Windows.Forms;
using Microsoft.VisualStudio.Shell.Interop;
```

2. Dentro de la FirstToolWindow clase, agregue una referencia pública a la FirstToolWindowControl control.

```
public FirstToolWindowControl control;
```

3. Al final del constructor, establezca esta variable de control en el control recién creado.

```
control = new FirstToolWindowControl();
base.Content = control;
```

4. Cree una instancia de la barra de herramientas dentro del constructor.

```
this.ToolBar = new CommandID(new Guid(FirstToolWindowCommand.guidFirstToolWindowPackageCmdSet),
    FirstToolWindowCommand.ToolbarID);
this.ToolBarLocation = (int)VSTWT_LOCATION.VSTWT_TOP;
```

5. En este punto, el FirstToolWindow constructor debe tener este aspecto:

```
public FirstToolWindow() : base(null)
{
    this.Caption = "FirstToolWindow";
    this.BitmapResourceID = 301;
    this.BitmapIndex = 1;
    control = new FirstToolWindowControl();
    base.Content = control;
    this.ToolBar = new CommandID(new Guid(FirstToolWindowCommand.guidFirstToolWindowPackageCmdSet),
        FirstToolWindowCommand.ToolbarID);
    this.ToolBarLocation = (int)VSTWT_LOCATION.VSTWT_TOP;
}
```

6. Agregue el comando de menú a la barra de herramientas. En la clase FirstToolWindowCommand.cs, agregue la siguiente directiva using:

```
using System.Windows.Forms;
```

7. En la clase FirstToolWindowCommand, agregue el código siguiente al final del método ShowToolWindow(). El comando ButtonHandler se implementará en la siguiente sección.

```
// Create the handles for the toolbar command.
var mcs = this.ServiceProvider.GetService(typeof(IMenuCommandService)) as OleMenuCommandService;
var toolbarbtnCmdID = new CommandID(new Guid(FirstToolWindowCommand.guidFirstToolWindowPackageCmdSet),
    FirstToolWindowCommand.cmdidWindowsMediaOpen);
var menuItem = new MenuCommand(new EventHandler(
    ButtonHandler), toolbarbtnCmdID);
mcs.AddCommand(menuItem);
```

Para implementar un comando de menú en la ventana de herramientas

1. En la clase FirstToolWindowCommand, agregue un método ButtonHandler que invoque el cuadro de diálogo **Abrir archivo**. Cuando se ha seleccionado un archivo, reproduce el archivo multimedia.
2. En la clase FirstToolWindowCommand, agregue una referencia privada a la ventana FirstToolWindow que se crea en el método FindToolWindow().

```
private FirstToolWindow window;
```

3. Cambie el método ShowToolWindow() para establecer la ventana que definió anteriormente (para que el controlador de comandos ButtonHandler pueda tener acceso al control de ventana. Aquí está el método showToolWindow() completo.

```
private void ShowToolWindow(object sender, EventArgs e)
{
    window = (FirstToolWindow) this.package.FindToolWindow(typeof(FirstToolWindow), 0, true);
    if ((null == window) || (null == window.Frame))
    {
        throw new NotSupportedException("Cannot create tool window");
    }

    IVsWindowFrame windowFrame = (IVsWindowFrame)window.Frame;
    Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(windowFrame.Show());

    var mcs = this.ServiceProvider.GetService(typeof(IMenuCommandService)) as OleMenuCommandService;
    var toolbarbtnCmdID = new CommandID(new
    Guid(FirstToolWindowCommandguidFirstToolWindowPackageCmdSet),
        FirstToolWindowCommand.cmdidWindowsMediaOpen);
    var menuItem = new MenuCommand(new EventHandler(
        ButtonHandler), toolbarbtnCmdID);
    mcs.AddCommand(menuItem);
}
```

4. Agregue el Método ButtonHandler. Crea un OpenFileDialog para que el usuario especifique el archivo multimedia que se va a reproducir y, a continuación, reproduce el archivo seleccionado.

```
private void ButtonHandler(object sender, EventArgs arguments)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    DialogResult result = openFileDialog.ShowDialog();
    if (result == DialogResult.OK)
    {
        window.control.MediaPlayer.Source = new System.Uri(openFileDialog.FileName);
    }
}
```

Establezca la posición predeterminada para la ventana de herramientas

A continuación, especifique una ubicación predeterminada en el IDE para la ventana de herramientas. La información de configuración de la ventana de herramientas se encuentra en el archivo *FirstToolWindowPackage.cs*.

1. En *FirstToolWindowPackage.cs*, [ProvideToolWindowAttribute](#) busque `FirstToolWindowPackage` el atributo de la clase, que pasa el tipo `FirstToolWindow` al constructor. Para especificar una posición predeterminada, debe agregar más parámetros al ejemplo siguiente del constructor.

```
[ProvideToolWindow(typeof(FirstToolWindow),
    Style = Microsoft.VisualStudio.Shell.VsDockStyle.Tabbed,
    Window = "3ae79031-e1bc-11d0-8f78-00a0c9110057")]
```

El primer parámetro `style` con nombre `Tabbed` es y su valor es `,` lo que significa que la ventana será una pestaña en una ventana existente. El parámetro especifica la `Window` posición de acoplamiento, n este caso, el GUID del Explorador de **soluciones**.

NOTE

Para obtener más información acerca de los [vsWindowType](#)tipos de ventanas en el IDE, vea .

Pruebe la ventana de herramientas

1. Presione **F5** para abrir una nueva instancia de la compilación experimental de Visual Studio.
2. En el menú **Ver**, seleccione **Otras ventanas** y, a continuación, haga clic en **Primera ventana de herramientas**.

La ventana de herramientas del reproductor multimedia debe abrirse en la misma posición que el **Explorador** de soluciones. Si sigue apareciendo en la misma posición que antes, restablezca el diseño de la ventana (**Ventana / Restablecer diseño** de ventana).

3. Haga clic en el botón (tiene el icono **Buscar**) en la ventana de herramientas. Seleccione un archivo de sonido o vídeo compatible, por ejemplo, *C:-windows-media-chimes.wavy*, a continuación, pulse **Abrir**.

Deberías oír el sonido de la campana.

Vea también

- [Comandos, menús y barras de herramientas](#)

Registrar una ventana de herramientas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede registrar las ventanas de herramientas utilizando [ProvideToolWindowAttribute](#) y [ProvideToolWindowVisibilityAttribute](#).

Ejemplo

```
[ProvideToolWindow(typeof(PersistedWindowPane), Style = MsVsShell.VsDockStyle.Tabbed, Window = "3ae79031-e1bc-11d0-8f78-00a0c9110057")]
[ProvideToolWindow(typeof(DynamicWindowPane), PositionX=250, PositionY=250, Width=160, Height=180, Transient=true)]
[ProvideToolWindowVisibility(typeof(DynamicWindowPane), /*UICONTEXT_SolutionExists*/"f1536ef8-92ec-443c-9ed7-fdadf150da82")]
[ProvideMenuResource(1000, 1)]
[PackageRegistration(UseManagedResourcesOnly = true)]
[Guid("01069CDD-95CE-4620-AC21-DDFF6C57F012")]
public class PackageToolWindow : Package
{
```

En el código [ProvideToolWindowAttribute](#) anterior, `PersistedWindowPane` las `DynamicWindowPane` ventanas de herramientas y las de Visual Studio registran. La ventana de herramientas persistentes está acoplada y tabulada con el **Explorador** de soluciones y la ventana dinámica tiene una posición y un tamaño iniciales predeterminados. La ventana dinámica se hace transitoria, lo que indica que no se crea en el inicio. Esto escribe `DontForceCreate` un valor `ToolWindows` en la clave en el registro del sistema. Para obtener más información, consulte [Configuración de visualización](#) de la ventana de herramientas .

Crear una ventana de herramientas de varias instancias

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede programar una ventana de herramientas para que varias instancias de la misma se puedan abrir simultáneamente. De forma predeterminada, las ventanas de herramientas solo pueden tener una instancia abierta.

Cuando se utiliza una ventana de herramientas de varias instancias, puede mostrar varias fuentes de información relacionadas al mismo tiempo. Por ejemplo, podría colocar un [TextBox](#) control de varias líneas en una ventana de herramientas de varias instancias para que varios fragmentos de código estén disponibles simultáneamente durante una sesión de programación. Además, por ejemplo, [DataGrid](#) podría colocar un control y un cuadro de lista desplegable en una ventana de herramientas de varias instancias para que se pueda realizar un seguimiento simultánea de varios orígenes de datos en tiempo real.

Crear una ventana de herramientas básica (de instancia única)

1. Cree un proyecto denominado **MultiInstanceToolWindow** con la plantilla VSIX y agregue una plantilla de elemento de ventana de herramientas personalizada denominada **MIToolWindow**.

NOTE

Para obtener más información sobre cómo crear una extensión con una ventana de herramientas, consulte [Crear una extensión con una ventana de herramientas](#).

Hacer una ventana de herramientas multiinstancia

1. Abra *MIToolWindowPackage.cs* el archivo *MIToolWindowPackage.cs* `ProvideToolWindow` y busque el atributo. y `MultiInstances=true` el parámetro, como se muestra en el ejemplo siguiente:

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)] // Info on this package
for Help/About
[ProvideMenuResource("Menus.ctmenu", 1)]
[ProvideToolWindow(typeof(MultiInstanceToolWindow.MIToolWindow), MultiInstances = true)]
[Guid(MIToolWindowPackage.PackageGuidString)]
public sealed class MIToolWindowPackage : Package
{. . .}
```

2. En el *archivo MIToolWindowCommand.cs*, busque el `ShowToolWinds()` método. En este método, [FindToolWindow](#) llame al `create` método `false` y establezca su marca para que iterará `id` a través de las instancias de ventana de herramientas existentes hasta que se encuentre una disponible.
3. Para crear una instancia de [FindToolWindow](#) ventana de `id` herramientas, llame `create` al `true` método y establezca su valor en un valor disponible y su marca en .

De forma predeterminada, `id` el valor [FindToolWindow](#) del `0` parámetro del método es . Este valor crea una ventana de herramientas de instancia única. Para que se hospede más de una `id` instancia, cada instancia debe tener su propio archivo .

4. Llame [Show](#) al método [IVsWindowFrame](#) en el objeto [Frame](#) devuelto por la propiedad de la instancia de la ventana de herramientas.
5. De forma `ShowToolWindow` predeterminada, el método creado por la plantilla de elemento de la ventana de herramientas crea una ventana de herramientas de instancia única. En el ejemplo siguiente `ShowToolWindow` se muestra cómo modificar el método para crear varias instancias.

```
private void ShowToolWindow(object sender, EventArgs e)
{
    for (int i = 0; i < 10; i++)
    {
        ToolWindowPane window = this.package.FindToolWindow(typeof(MIToolWindow), i, false);
        if (window == null)
        {
            // Create the window with the first free ID.
            window = (ToolWindowPane)this.package.FindToolWindow(typeof(MIToolWindow), i, true);
            if ((null == window) || (null == window.Frame))
            {
                throw new NotSupportedException("Cannot create tool window");
            }

            IVsWindowFrame windowFrame = (IVsWindowFrame)window.Frame;
            Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(windowFrame.Show());
            break;
        }
    }
}
```

Abrir una ventana de herramientas dinámica

06/04/2020 • 3 minutes to read • [Edit Online](#)

Las ventanas de herramientas normalmente se abren desde un comando de un menú o un método abreviado de teclado equivalente. A veces, sin embargo, es posible que necesite una ventana de herramientas que se abre siempre que se aplica un contexto de interfaz de usuario específico y se cierra cuando el contexto de la interfaz de usuario ya no se aplica. Estos tipos de ventanas de herramientas se denominan *dinámicas* o *autovisibles*.

NOTE

Para obtener una lista de contextos de interfaz de usuario predefinidos, vea [VSConstants.UICONTEXT](#).

Si desea abrir una ventana de herramientas dinámica según el inicio y es [IVsPackageDynamicToolOwnerEx](#) posible que se produzca [QueryShowTool](#) un error en la creación, debe implementar la interfaz y probar las condiciones de error en el método. Para que el shell sepa que tiene una ventana de herramientas dinámica que

`SupportsDynamicToolOwner` se debe abrir al inicio, debe agregar el valor (establecido en 1) al registro del paquete. Este valor no forma [PackageRegistrationAttribute](#) parte del estándar, por lo que debe crear un atributo personalizado para agregarlo. Para obtener más información acerca de los atributos personalizados, vea [Usar un atributo de registro personalizado para registrar una extensión](#).

Se [FindToolWindow](#) utiliza para abrir una ventana de herramientas. La ventana de herramientas se crea según sea necesario.

NOTE

El usuario puede cerrar una ventana de herramientas dinámica. Si desea crear un comando de menú para que el usuario pueda volver a abrir la ventana de herramientas, el comando de menú debe habilitarse en el mismo contexto de interfaz de usuario que abre la ventana de herramientas y deshabilitarse de lo contrario.

Para abrir una ventana de herramientas dinámica

1. Cree un proyecto VSIX denominado **DynamicToolWindow** y agregue una plantilla de elemento de ventana de herramientas denominada *DynamicWindowPane.cs*. Para obtener más información, consulte [Crear una extensión con una ventana de herramientas](#).
2. En el *DynamicWindowPanePackage.cs* archivo, busque el `DynamicWindowPanePackage` declaración. Agregue [ProvideToolWindowAttribute](#) los [ProvideToolWindowVisibilityAttribute](#) atributos y para registrar la ventana de herramientas.

```
[ProvideToolWindow(typeof(DynamicWindowPane))]  
[ProvideToolWindowVisibility(typeof(DynamicWindowPane), VSConstants.UICONTEXT.SolutionExists_string)]  
[PackageRegistration(UseManagedResourcesOnly = true)]  
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)] // Info on this package for  
Help/About  
[ProvideMenuResource("Menus.ctmenu", 1)]  
[ProvideToolWindow(typeof(DynamicToolWindow.DynamicWindowPane))]  
[Guid(DynamicWindowPanePackage.PackageGuidString)]  
public sealed class DynamicWindowPanePackage : Package  
{  
    . . .  
}
```

Los atributos anteriores registran la ventana de herramientas denominada `DynamicWindowPane` como una

ventana transitoria que no se conserva cuando Visual Studio se cierra y se vuelve a abrir.

DynamicWindowPane se [SolutionExists_string](#) abre siempre que se aplica y se cierra de lo contrario.

3. Compile la solución y comience la depuración. Debería aparecer la instancia experimental. No debería ver la ventana de herramientas.
4. Abra un proyecto en la instancia experimental. Debería aparecer la ventana de herramientas.

Añadir un menú contextual en una ventana de herramientas

06/04/2020 • 10 minutes to read • [Edit Online](#)

Este tutorial coloca un menú contextual en una ventana de herramientas. Un menú contextual es un menú contextual que aparece cuando un usuario hace clic con el botón derecho en un botón, cuadro de texto o fondo de ventana. Los comandos de un menú contextual se comportan igual que los comandos de otros menús o barras de herramientas. Para admitir un menú contextual, estibre en el archivo `.vscty` y muéselo en respuesta al clic derecho del ratón.

Una ventana de herramientas consta de un control de usuario `ToolWindowPaneWPF` en una clase de ventana de herramientas personalizada que hereda de `ToolWindowPaneWPF`.

En este tutorial se muestra cómo crear un menú contextual como un menú de Visual Studio, declarando elementos de menú en el archivo `.vscty`, a continuación, mediante Managed Package Framework para implementarlos en la clase que define la ventana de herramientas. Este enfoque facilita el acceso a los comandos de Visual Studio, los elementos de interfaz de usuario y el modelo de objetos de automatización.

Como alternativa, si el menú contextual no tendrá acceso `ContextMenu` a la funcionalidad de Visual Studio, puede usar la propiedad de un elemento XAML en el control de usuario. Para obtener más información, vea [ContextMenu](#).

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Cree el paquete de menú contextual de la ventana de herramientas

1. Cree un proyecto `TWShortcutMenu` VSIX con el nombre y agregue una plantilla de ventana de herramientas denominada `ShortcutMenu`. Para obtener más información sobre cómo crear una ventana de herramientas, consulte [Crear una extensión con una ventana de herramientas](#).

Especificar el menú contextual

Un menú contextual como el que se muestra en este tutorial permite al usuario seleccionar de una lista de colores que se utilizan para rellenar el fondo de la ventana de herramientas.

1. En `ShortcutMenuPackage.vsct`, busque en el `guidSymbol` elemento denominado `guidShortcutMenuPackageCmdSet` declare el menú contextual, el grupo de menú contextual y las opciones de menú. El `guidsymbol` elemento ahora debe tener este aspecto:

```
<GuidSymbol name="guidShortcutMenuPackageCmdSet" value="{00000000-0000-0000-0000-0000}"> // your GUID here
  <IDSymbol name="ShortcutMenuCommandId" value="0x0100" />
  <IDSymbol name="ColorMenu" value="0x1000"/>
  <IDSymbol name="ColorGroup" value="0x1100"/>
  <IDSymbol name="cmdidRed" value="0x102"/>
  <IDSymbol name="cmdidYellow" value="0x103"/>
  <IDSymbol name="cmdidBlue" value="0x104"/>
</GuidSymbol>
```

2. Justo antes del elemento Buttons, cree un elemento Menus y, a continuación, defina el menú contextual en él.

```
<Menus>
  <Menu guid="guidShortcutMenuPackageCmdSet" id="ColorMenu" type="Context">
    <Strings>
      <ButtonText>Color change</ButtonText>
      <CommandName>ColorChange</CommandName>
    </Strings>
  </Menu>
</Menus>
```

Un menú contextual no tiene un elemento primario porque no forma parte de un menú o barra de herramientas.

3. Cree un elemento Groups con un elemento Group que contenga los elementos de menú contextual y asocie el grupo al menú contextual.

```
<Groups>
  <Group guid="guidShortcutMenuPackageCmdSet" id="ColorGroup">
    <Parent guid="guidShortcutMenuPackageCmdSet" id="ColorMenu"/>
  </Group>
</Groups>
```

4. En el elemento Botones, defina los comandos individuales que aparecerán en el menú contextual. El elemento Buttons debe tener este aspecto:

```
<Buttons>
  <Button guid="guidShortcutMenuPackageCmdSet" id="ShortcutMenuCommandId" priority="0x0100"
type="Button">
    <Parent guid="guidSHLMainMenu" id="IDG_VS_WNDO_OTRWNDWS1"/>
    <Icon guid="guidImages" id="bmpPic1" />
    <Strings>
      <ButtonText>ShortcutMenu</ButtonText>
    </Strings>
  </Button>

  <Button guid="guidShortcutMenuPackageCmdSet" id="cmdidRed" priority="1" type="Button">
    <Parent guid="guidShortcutMenuPackageCmdSet" id="ColorGroup" />
    <Strings>
      <ButtonText>Red</ButtonText>
    </Strings>
  </Button>

  <Button guid="guidShortcutMenuPackageCmdSet" id="cmdidYellow" priority="3" type="Button">
    <Parent guid="guidShortcutMenuPackageCmdSet" id="ColorGroup" />
    <Strings>
      <ButtonText>Yellow</ButtonText>
    </Strings>
  </Button>

  <Button guid="guidShortcutMenuPackageCmdSet" id="cmdidBlue" priority="5" type="Button">
    <Parent guid="guidShortcutMenuPackageCmdSet" id="ColorGroup" />
    <Strings>
      <ButtonText>Blue</ButtonText>
    </Strings>
  </Button>
</Buttons>
```

5. En *ShortcutMenuCommand.cs*, agregue las definiciones para el GUID del conjunto de comandos, el menú contextual y los elementos de menú.

```
public const string guidShortcutMenuPackageCmdSet = "00000000-0000-0000-0000-00000000"; // your GUID
will differ
public const int ColorMenu = 0x1000;
public const int cmdidRed = 0x102;
public const int cmdidYellow = 0x103;
public const int cmdidBlue = 0x104;
```

Estos son los mismos IDs de comando que se definen en la sección Símbolos del archivo *ShortcutMenuPackage.vsct*. El grupo de contexto no se incluye aquí porque solo es necesario en el archivo *.vsct*.

Implementación del menú contextual

Esta sección implementa el menú contextual y sus comandos.

1. En *ShortcutMenu.cs*, la ventana de herramientas puede obtener el servicio de comandos de menú, pero el control que contiene no puede. Los pasos siguientes muestran cómo hacer que el servicio de comandos de menú esté disponible para el control de usuario.
2. En *ShortcutMenu.cs*, agregue las siguientes directivas using:

```
using Microsoft.VisualStudio.Shell;
using System.ComponentModel.Design;
```

3. Invalide el método `Initialize()` de la ventana de herramientas para obtener el servicio de comandos de menú y agregar el control, pasando el servicio de comandos de menú al constructor:

```
protected override void Initialize()
{
    var commandService = (OleMenuCommandService)GetService(typeof(IMenuCommandService));
    Content = new ShortcutMenuControl(commandService);
}
```

4. En el constructor de la ventana de la herramienta *ShortcutMenu*, quite la línea que agrega el control. El constructor ahora debería tener este aspecto:

```
public ShortcutMenu() : base(null)
{
    this.Caption = "ShortcutMenu";
    this.BitmapResourceID = 301;
    this.BitmapIndex = 1;
}
```

5. En *ShortcutMenuControl.xaml.cs*, agregue un campo privado para el servicio de comandos de menú y cambie el constructor de control para tomar el servicio de comandos de menú. A continuación, utilice el servicio de comandos de menú para agregar los comandos de menú contextual. El *ShortcutMenuControl* constructor ahora debe parecerse al código siguiente. El controlador de comandos se definirá más adelante.

```

public ShortcutMenuControl(OleMenuCommandService service)
{
    this.InitializeComponent();
    commandService = service;

    if (null != commandService)
    {
        // Create an alias for the command set guid.
        Guid guid = new Guid(ShortcutMenuCommand.guidShortcutMenuPackageCmdSet);

        // Create the command IDs.
        var red = new CommandID(guid, ShortcutMenuCommand.cmdidRed);
        var yellow = new CommandID(guid, ShortcutMenuCommand.cmdidYellow);
        var blue = new CommandID(guid, ShortcutMenuCommand.cmdidBlue);

        // Add a command for each command ID.
        commandService.AddCommand(new MenuCommand(ChangeColor, red));
        commandService.AddCommand(new MenuCommand(ChangeColor, yellow));
        commandService.AddCommand(new MenuCommand(ChangeColor, blue));
    }
}

```

6. En *ShortcutMenuControl.xaml*, [MouseRightButtonDown](#) agregue un [UserControl](#) evento al elemento de nivel superior. El archivo XAML ahora debería tener este aspecto:

```

<UserControl x:Class="TWShortcutMenu.ShortcutMenuControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    Background="{DynamicResource VsBrush.Window}"
    Foreground="{DynamicResource VsBrush.WindowText}"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300"
    Name="MyToolWindow"
    MouseRightButtonDown="MyToolWindow_MouseRightButtonDown">
    <Grid>
        <StackPanel Orientation="Vertical">
            <TextBlock Margin="10" HorizontalAlignment="Center">ShortcutMenu</TextBlock>
        </StackPanel>
    </Grid>
</UserControl>

```

7. En *ShortcutMenuControl.xaml.cs*, agregue un código auxiliar para el controlador de eventos.

```

private void MyToolWindow_MouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    . . .
}

```

8. Agregue las siguientes directivas using al mismo archivo:

```

using Microsoft.VisualStudio.Shell;
using System.ComponentModel.Design;
using System;
using System.Windows.Input;
using System.Windows.Media;

```

9. Implemente `MyToolWindowMouseRightButtonDown` el evento de la siguiente manera.

```
private void MyToolWindow_MouseRightButtonDown(object sender, MouseButtonEventArgs e)
{
    if (null != commandService)
    {
        CommandID menuID = new CommandID(
            new Guid(ShortcutMenuCommand.guidShortcutMenuPackageCmdSet),
            ShortcutMenuCommand.ColorMenu);
        Point p = this.PointToScreen(e.GetPosition(this));
        commandService.ShowContextMenu(menuID, (int)p.X, (int)p.Y);
    }
}
```

Esto crea [CommandID](#) un objeto para el menú contextual, identifica la ubicación del clic del [ShowContextMenu](#) mouse y abre el menú contextual en esa ubicación mediante el método.

10. Implemente el controlador de comandos.

```
private void ChangeColor(object sender, EventArgs e)
{
    var mc = sender as MenuCommand;

    switch (mc.CommandID.ID)
    {
        case ShortcutMenuCommand.cmdidRed:
            MyToolWindow.Background = Brushes.Red;
            break;
        case ShortcutMenuCommand.cmdidYellow:
            MyToolWindow.Background = Brushes.Yellow;
            break;
        case ShortcutMenuCommand.cmdidBlue:
            MyToolWindow.Background = Brushes.Blue;
            break;
    }
}
```

En este caso, solo un método controla los [CommandID](#) eventos de todos los elementos de menú identificando el y estableciendo el color de fondo en consecuencia. Si los elementos de menú hubieran contenido comandos no relacionados, habría creado un controlador de eventos independiente para cada comando.

Pruebe las características de la ventana de herramientas

1. Compile la solución y comience la depuración. Aparece la instancia experimental.
2. En la instancia experimental, haga clic en **Ver / Otras ventanas**, a continuación, haga clic en **Menú acceso directo**. Esto debería mostrar la ventana de herramientas.
3. Haga clic con el botón derecho en el cuerpo de la ventana de herramientas. Se debe mostrar un menú contextual que tenga una lista de colores.
4. Haga clic en un color en el menú contextual. El color de fondo de la ventana de herramientas debe cambiarse al color seleccionado.

Vea también

- [Comandos, menús y barras de herramientas](#)
- [Uso y prestación de servicios](#)

Añadir búsqueda a una ventana de herramientas

06/04/2020 • 20 minutes to read • [Edit Online](#)

Al crear o actualizar una ventana de herramientas en la extensión, puede agregar la misma funcionalidad de búsqueda que aparece en otro lugar de Visual Studio. Esta funcionalidad incluye las siguientes características:

- Un cuadro de búsqueda que siempre se encuentra en un área personalizada de la barra de herramientas.
- Un indicador de progreso que está superpuesto en el propio cuadro de búsqueda.
- La capacidad de mostrar los resultados tan pronto como introduzca cada carácter (búsqueda instantánea) o solo después de elegir la tecla **Intro** (búsqueda a petición).
- Una lista que muestra los términos para los que ha buscado más recientemente.
- La capacidad de filtrar las búsquedas por campos o aspectos específicos de los destinos de búsqueda.

Siguiendo este tutorial, aprenderá a realizar las siguientes tareas:

1. Crear un proyecto de VSPackage.
2. Cree una ventana de herramientas que contenga un UserControl con un cuadro de texto de solo lectura.
3. Agregue un cuadro de búsqueda a la ventana de herramientas.
4. Agregue la implementación de búsqueda.
5. Habilite la búsqueda instantánea y la visualización de una barra de progreso.
6. Agregue una opción **Coincidir mayúsculas y minúsculas**.
7. Agregue un filtro **Buscar solo líneas uniformes**.

Para crear un proyecto de VSIX

1. Cree un proyecto `TestToolWindowSearch` VSIX denominado con una ventana de herramientas denominada **TestSearch**. Si necesita ayuda para hacerlo, consulte [Creación de una extensión con una ventana de herramientas](#).

Para crear una ventana de herramientas

1. En `TestToolWindowSearch` el proyecto, abra el archivo `TestSearchControl.xaml`.
2. Reemplace el `<StackPanel>` bloque existente por el siguiente bloque, `TextBox` que `UserControl` agrega un solo lectura al bloque de la ventana de herramientas.

```
<StackPanel Orientation="Vertical">
    <TextBox Name="resultsTextBox" Height="800.0"
        Width="800.0"
        IsReadOnly="True">
    </TextBox>
</StackPanel>
```

3. En el archivo `TestSearchControl.xaml.cs`, agregue la siguiente directiva using:

```
using System.Text;
```

4. Quite `button1_Click()` el método.

En la clase **TestSearchControl**, agregue el código siguiente.

Este código agrega [TextBox](#) una propiedad pública denominada **SearchResultsTextBox** y una propiedad de cadena pública denominada **SearchContent**. En el constructor, **SearchResultsTextBox** se establece en el cuadro de texto y **SearchContent** se inicializa en un conjunto de cadenas delimitado por nueva línea. El contenido del cuadro de texto también se inicializa en el conjunto de cadenas.

```
public partial class MyControl : UserControl
{
    public TextBox SearchResultsTextBox { get; set; }
    public string SearchContent { get; set; }

    public MyControl()
    {
        InitializeComponent();

        this.SearchResultsTextBox = resultsTextBox;
        this.SearchContent = BuildContent();

        this.SearchResultsTextBox.Text = this.SearchContent;
    }

    private string BuildContent()
    {
        StringBuilder sb = new StringBuilder();
        sb.AppendLine("1 go");
        sb.AppendLine("2 good");
        sb.AppendLine("3 Go");
        sb.AppendLine("4 Good");
        sb.AppendLine("5 goodbye");
        sb.AppendLine("6 Goodbye");

        return sb.ToString();
    }
}
```

```

Partial Public Class MyControl
    Inherits System.Windows.Controls.UserControl

    Public Property SearchResultsTextBox As System.Windows.Controls.TextBox
    Public Property SearchContent As String

    Public Sub New()

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        Me.SearchResultsTextBox = resultsTextBox
        Me.SearchContent = buildContent

        Me.SearchResultsTextBox.Text = Me.SearchContent
    End Sub

    Private Function BuildContent() As String
        Dim sb As New System.Text.StringBuilder()

        sb.AppendLine("1 go")
        sb.AppendLine("2 good")
        sb.AppendLine("3 Go")
        sb.AppendLine("4 Good")
        sb.AppendLine("5 goodbye")
        sb.AppendLine("6 Goodbye")

        Return sb.ToString()
    End Function
End Class

```

5. Compile la solución y comience la depuración. Aparece la instancia experimental de Visual Studio.
6. En la barra de menús, elija **Ver** > otras **Pruebas de Windows**. >

Aparece la ventana de herramientas, pero el control de búsqueda aún no aparece.

Para añadir un cuadro de búsqueda a la ventana de herramientas

1. En el *archivo TestSearch.cs*, agregue `TestSearch` el código siguiente a la clase. El código reemplaza `SearchEnabled` la propiedad para `true` que el descriptor de acceso get devuelva .

Para habilitar la búsqueda, `SearchEnabled` debe invalidar la propiedad. La `ToolWindowPane` clase `IVsWindowSearch` implementa y proporciona una implementación predeterminada que no habilita la búsqueda.

```

public override bool SearchEnabled
{
    get { return true; }
}

```

2. Compile la solución y comience la depuración. Aparece la instancia experimental.
3. En la instancia experimental de Visual Studio, abra **TestSearch**.

En la parte superior de la ventana de herramientas, aparece un control de búsqueda con una marca de agua **de búsqueda** y un icono de lupa. Sin embargo, la búsqueda aún no funciona porque el proceso de búsqueda no se ha implementado.

Para agregar la implementación de búsqueda

Al habilitar la [ToolWindowPane](#) búsqueda en un , como en el procedimiento anterior, la ventana de herramientas crea un host de búsqueda. Este host configura y administra los procesos de búsqueda, que siempre se producen en un subproceso en segundo plano. Dado [ToolWindowPane](#) que la clase administra la creación del host de búsqueda y la configuración de la búsqueda, solo necesita crear una tarea de búsqueda y proporcionar el método de búsqueda. El proceso de búsqueda se produce en un subproceso en segundo plano y las llamadas al control de ventana de herramientas se producen en el subproceso de interfaz de usuario. Por lo tanto, debe usar el [método ThreadHelper.Invoke*](#) para administrar las llamadas que realice al tratar con el control.

1. En el archivo *TestSearch.cs*, agregue las siguientes `using` directivas:

```
using System;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using System.Text;
using System.Windows.Controls;
using Microsoft.Internal.VisualStudio.PlatformUI;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.PlatformUI;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;
```

2. En `TestSearch` la clase, agregue el código siguiente, que realiza las siguientes acciones:

- Reemplaza el [CreateSearch](#) método para crear una tarea de búsqueda.
- Reemplaza el [ClearSearch](#) método para restaurar el estado del cuadro de texto. Se llama a este método cuando un usuario cancela una tarea de búsqueda y cuando un usuario establece o desestablece opciones o filtros. Ambos [CreateSearch](#) [ClearSearch](#) y se llaman en el subproceso de interfaz de usuario. Por lo tanto, no es necesario tener acceso al cuadro de texto mediante el [ThreadHelper.Invoke*](#) método.
- Crea una clase con `TestSearchTask` el nombre [VsSearchTask](#) que hereda de [IVsSearchTask](#), que proporciona una implementación predeterminada de .

En `TestSearchTask` , el constructor establece un campo privado que hace referencia a la ventana de herramientas. Para proporcionar el método de [OnStartSearch](#) [OnStopSearch](#) búsqueda, invalide los métodos y. El [OnStartSearch](#) método es donde se implementa el proceso de búsqueda. Este proceso incluye realizar la búsqueda, mostrar los resultados de la búsqueda en el cuadro de texto y llamar a la implementación de la clase base de este método para informar de que la búsqueda se ha completado.

```
public override IVsSearchTask CreateSearch(uint dwCookie, IVsSearchQuery pSearchQuery,
IVsSearchCallback pSearchCallback)
{
    if (pSearchQuery == null || pSearchCallback == null)
        return null;
    return new TestSearchTask(dwCookie, pSearchQuery, pSearchCallback, this);
}

public override void ClearSearch()
{
    TestSearchControl control = (TestSearchControl)this.Content;
    control.SearchResultsTextBox.Text = control.SearchContent;
}

internal class TestSearchTask : VsSearchTask
{
    private TestSearch m_toolWindow;
```

```

    public TestSearchTask(uint dwCookie, IVsSearchQuery pSearchQuery, IVsSearchCallback
pSearchCallback, TestSearch toolwindow)
        : base(dwCookie, pSearchQuery, pSearchCallback)
    {
        m_toolWindow = toolwindow;
    }

    protected override void OnStartSearch()
    {
        // Use the original content of the text box as the target of the search.
        var separator = new string[] { Environment.NewLine };
        TestSearchControl control = (TestSearchControl)m_toolWindow.Content;
        string[] contentArr = control.SearchContent.Split(separator, StringSplitOptions.None);

        // Get the search option.
        bool matchCase = false;
        // matchCase = m_toolWindow.MatchCaseOption.Value;

        // Set variables that are used in the finally block.
        StringBuilder sb = new StringBuilder("");
        uint resultCount = 0;
        this.ErrorCode = VSConstants.S_OK;

        try
        {
            string searchString = this.SearchQuery.SearchString;

            // Determine the results.
            uint progress = 0;
            foreach (string line in contentArr)
            {
                if (matchCase == true)
                {
                    if (line.Contains(searchString))
                    {
                        sb.AppendLine(line);
                        resultCount++;
                    }
                }
                else
                {
                    if (line.ToLower().Contains(searchString.ToLower()))
                    {
                        sb.AppendLine(line);
                        resultCount++;
                    }
                }

                // SearchCallback.ReportProgress(this, progress++,
                (uint)contentArr.GetLength(0));

                // Uncomment the following line to demonstrate the progress bar.
                // System.Threading.Thread.Sleep(100);
            }
        }
        catch (Exception e)
        {
            this.ErrorCode = VSConstants.E_FAIL;
        }
        finally
        {
            ThreadHelper.Generic.Invoke(() =>
            { ((TextBox)((TestSearchControl)m_toolWindow.Content).SearchResultsTextBox).Text =
sb.ToString(); });

            this.SearchResults = resultCount;
        }
    }

```

```

        // Call the implementation of this method in the base class.
        // This sets the task status to complete and reports task completion.
        base.OnStartSearch();
    }

    protected override void OnStopSearch()
    {
        this.SearchResults = 0;
    }
}

```

3. Pruebe la implementación de la búsqueda realizando los pasos siguientes:

- a. Vuelva a generar el proyecto e inicie la depuración.
- b. En la instancia experimental de Visual Studio, vuelva a abrir la ventana de herramientas, escriba texto de búsqueda en la ventana de búsqueda y haga clic en **ENTRAR**.

Deberían aparecer los resultados correctos.

Para personalizar el comportamiento de búsqueda

Al cambiar la configuración de búsqueda, puede realizar una variedad de cambios en cómo aparece el control de búsqueda y cómo se lleva a cabo la búsqueda. Por ejemplo, puede cambiar la marca de agua (el texto predeterminado que aparece en el cuadro de búsqueda), el ancho mínimo y máximo del control de búsqueda y si desea mostrar una barra de progreso. También puede cambiar el punto en el que los resultados de búsqueda comienzan a aparecer (a petición o búsqueda instantánea) y si desea mostrar una lista de términos para los que ha buscado recientemente. Puede encontrar la lista completa [SearchSettingsDataSource](#) de ajustes en la clase.

1. En el archivo TestSearch.cs*, agregue el `TestSearch` código siguiente a la clase. Este código permite la búsqueda instantánea en lugar de la búsqueda bajo demanda (lo que significa que el usuario no tiene que hacer clic en **ENTRAR**). El código reemplaza `ProvideSearchSettings` el `TestSearch` método de la clase, que es necesario para cambiar la configuración predeterminada.

```

public override void ProvideSearchSettings(IVsUIDataSource pSearchSettings)
{
    Utilities.SetValue(pSearchSettings,
        SearchSettingsDataSource.SearchStartTypeProperty.Name,
        (uint)VSSEARCHSTARTTYPE.SST_INSTANT);}

```

2. Pruebe la nueva configuración reconstruyendo la solución y reiniciando el depurador.

Los resultados de la búsqueda aparecen cada vez que se introduce un carácter en el cuadro de búsqueda.

3. En `ProvideSearchSettings` el método, agregue la siguiente línea, que permite la visualización de una barra de progreso.

```

public override void ProvideSearchSettings(IVsUIDataSource pSearchSettings)
{
    Utilities.SetValue(pSearchSettings,
        SearchSettingsDataSource.SearchStartTypeProperty.Name,
        (uint)VSSEARCHSTARTTYPE.SST_INSTANT);
    Utilities.SetValue(pSearchSettings,
        SearchSettingsDataSource.SearchProgressTypeProperty.Name,
        (uint)VSSEARCHPROGRESSTYPE.SPT_DETERMINATE);
}

```

Para que aparezca la barra de progreso, se debe notificar el progreso. Para informar del progreso, quite el `OnStartSearch` comentario `TestSearchTask` del siguiente código en el método de la clase:

```
SearchCallback.ReportProgress(this, progress++, (uint)contentArr.GetLength(0));
```

4. Para ralentizar el procesamiento lo suficiente como para que `OnStartSearch` la barra `TestSearchTask` de progreso esté visible, quite el comentario de la siguiente línea en el método de la clase:

```
System.Threading.Thread.Sleep(100);
```

5. Pruebe la nueva configuración reconstruyendo la solución y empezando a depurar.

La barra de progreso aparece en la ventana de búsqueda (como una línea azul debajo del cuadro de texto de búsqueda) cada vez que realiza una búsqueda.

Para permitir a los usuarios refinar sus búsquedas

Puede permitir a los usuarios refinar sus búsquedas mediante opciones como **Coincidir mayúsculas y minúsculas** o **Coincidir con palabra completa**. Las opciones pueden ser booleanas, que aparecen como casillas de verificación, o comandos, que aparecen como botones. Para este tutorial, creará una opción booleana.

1. En el *archivo TestSearch.cs*, agregue `TestSearch` el código siguiente a la clase. El código reemplaza `SearchOptionsEnum` el método, lo que permite que la implementación de búsqueda detecte si una opción determinada está activada o desactivada. El código `SearchOptionsEnum` de agrega una opción [IVsEnumWindowSearchOptions](#) para que coincida con mayúsculas y minúsculas con un enumerador. La opción de coincidir con el `MatchCaseOption` caso también está disponible como la propiedad.

```
private IVsEnumWindowSearchOptions m_optionsEnum;
public override IVsEnumWindowSearchOptions SearchOptionsEnum
{
    get
    {
        if (m_optionsEnum == null)
        {
            List<IVsWindowSearchOption> list = new List<IVsWindowSearchOption>();

            list.Add(this.MatchCaseOption);

            m_optionsEnum = new WindowSearchOptionEnumerator(list) as IVsEnumWindowSearchOptions;
        }
        return m_optionsEnum;
    }
}

private WindowSearchBooleanOption m_matchCaseOption;
public WindowSearchBooleanOption MatchCaseOption
{
    get
    {
        if (m_matchCaseOption == null)
        {
            m_matchCaseOption = new WindowSearchBooleanOption("Match case", "Match case", false);
        }
        return m_matchCaseOption;
    }
}
```

2. En `TestSearchTask` la clase, quite el `OnStartSearch` comentario de la siguiente línea en el método:

```
matchCase = m_toolWindow.MatchCaseOption.Value;
```

3. Pruebe la opción:

- Compile la solución y comience la depuración. Aparece la instancia experimental.
- En la ventana de herramientas, elija la flecha hacia abajo en el lado derecho del cuadro de texto.

Aparece la casilla de verificación **Coincidir mayúsculas y minúsculas**.

- Active la casilla de verificación **Coincidir mayúsculas y minúsculas** y, a continuación, realice algunas búsquedas.

Para añadir un filtro de búsqueda

Puede agregar filtros de búsqueda que permitan a los usuarios refinar el conjunto de destinos de búsqueda. Por ejemplo, puede filtrar archivos en el Explorador de archivos por las fechas en las que se modificaron más recientemente y sus extensiones de nombre de archivo. En este tutorial, agregará un filtro solo para líneas uniformes. Cuando el usuario elige ese filtro, el host de búsqueda agrega las cadenas que especifique a la consulta de búsqueda. A continuación, puede identificar estas cadenas dentro del método de búsqueda y filtrar los destinos de búsqueda en consecuencia.

- En el *archivo TestSearch.cs*, agregue `TestSearch` el código siguiente a la clase. El código `SearchFiltersEnum` se implementa `WindowSearchSimpleFilter` agregando un que especifica filtrar los resultados de la búsqueda para que solo aparezcan líneas par.

```
public override IVsEnumWindowSearchFilters SearchFiltersEnum
{
    get
    {
        List<IVsWindowSearchFilter> list = new List<IVsWindowSearchFilter>();
        list.Add(new WindowSearchSimpleFilter("Search even lines only", "Search even lines only",
        "lines", "even"));
        return new WindowSearchFilterEnumerator(list) as IVsEnumWindowSearchFilters;
    }
}
```

Ahora el control de `Search even lines only` búsqueda muestra el filtro de búsqueda. Cuando el usuario elige el `lines:"even"` filtro, la cadena aparece en el cuadro de búsqueda. Pueden aparecer otros criterios de búsqueda al mismo tiempo que el filtro. Las cadenas de búsqueda pueden aparecer antes del filtro, después del filtro o ambas.

- En el *archivo TestSearch.cs*, agregue `TestSearchTask` los métodos siguientes `TestSearch` a la clase, que se encuentra en la clase. Estos métodos `OnStartSearch` admiten el método, que modificará en el paso siguiente.


```

private string RemoveFromString(string origString, string stringToRemove)
{
    int index = origString.IndexOf(stringToRemove);
    if (index == -1)
        return origString;
    else
        return (origString.Substring(0, index) + origString.Substring(index +
stringToRemove.Length)).Trim();
}

private string[] GetEvenItems(string[] contentArr)
{
    int length = contentArr.Length / 2;
    string[] evenContentArr = new string[length];

    int indexB = 0;
    for (int index = 1; index < contentArr.Length; index += 2)
    {
        evenContentArr[indexB] = contentArr[index];
        indexB++;
    }

    return evenContentArr;
}

```

3. En `TestSearchTask` la clase, `OnStartSearch` actualice el método con el código siguiente. Este cambio actualiza el código para admitir el filtro.

```

protected override void OnStartSearch()
{
    // Use the original content of the text box as the target of the search.
    var separator = new string[] { Environment.NewLine };
    string[] contentArr = ((TestSearchControl)m_toolWindow.Content).SearchContent.Split(separator,
StringSplitOptions.None);

    // Get the search option.
    bool matchCase = false;
    matchCase = m_toolWindow.MatchCaseOption.Value;

    // Set variables that are used in the finally block.
    StringBuilder sb = new StringBuilder("");
    uint resultCount = 0;
    this.ErrorCode = VSConstants.S_OK;

    try
    {
        string searchString = this.SearchQuery.SearchString;

        // If the search string contains the filter string, filter the content array.
        string filterString = "lines:\"even\"";

        if (this.SearchQuery.SearchString.Contains(filterString))
        {
            // Retain only the even items in the array.
            contentArr = GetEvenItems(contentArr);

            // Remove 'lines:"even"' from the search string.
            searchString = RemoveFromString(searchString, filterString);
        }

        // Determine the results.
        uint progress = 0;
        foreach (string line in contentArr)
        {
            if (matchCase == true)
            {

```

```

        if (line.Contains(searchString))
        {
            sb.AppendLine(line);
            resultCount++;
        }
    }
    else
    {
        if (line.ToLower().Contains(searchString.ToLower()))
        {
            sb.AppendLine(line);
            resultCount++;
        }
    }

    SearchCallback.ReportProgress(this, progress++, (uint)contentArr.GetLength(0));

    // Uncomment the following line to demonstrate the progress bar.
    // System.Threading.Thread.Sleep(100);
}
}
catch (Exception e)
{
    this.ErrorCode = VSConstants.E_FAIL;
}
finally
{
    ThreadHelper.Generic.Invoke(() =>
    { ((TextBox)((TestSearchControl)m_toolWindow.Content).SearchResultsTextBox).Text =
sb.ToString(); });

    this.SearchResults = resultCount;
}

// Call the implementation of this method in the base class.
// This sets the task status to complete and reports task completion.
base.OnStartSearch();
}

```

4. Pruebe el código.
5. Compile la solución y comience la depuración. En la instancia experimental de Visual Studio, abra la ventana de herramientas y, a continuación, elija la flecha abajo en el control de búsqueda.

Aparece la casilla de verificación **Coincidir mayúsculas y minúsculas** y el filtro **Buscar solo líneas uniformes**.

6. Elija el filtro.

El cuadro de búsqueda contiene **líneas:"par"** y aparecen los siguientes resultados:

2 buenos

4 Bueno

6 Adiós

7. Eliminar `lines:"even"` del cuadro de búsqueda, active la casilla ☐ **Coincidir mayúsculas y minúsculas** y, a continuación, escriba en el cuadro de búsqueda.

Aparecen los siguientes resultados:

1 ir

2 buenos

5 adiós

8. Elija la X en el lado derecho del cuadro de búsqueda.

La búsqueda se borra y aparece el contenido original. Sin embargo, la casilla de verificación **Coincidir mayúsculas y minúsculas** sigue activada.

Suscripción a un evento

06/04/2020 • 4 minutes to read • [Edit Online](#)

En este tutorial se explica cómo crear una ventana de herramientas que responde a eventos en una tabla de documentos en ejecución (RDT). Una ventana de herramientas hospeda [IVsRunningDocTableEvents](#) un control de usuario que implementa . El [AdviseRunningDocTableEvents](#) método conecta la interfaz con los eventos.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Suscribirse a eventos RDT

Para crear una extensión con una ventana de herramientas

1. Cree un proyecto denominado **RDTE Explorer** mediante la plantilla VSIX y agregue una plantilla de elemento de ventana de herramientas personalizada denominada **RDTE Explorer Window**.

Para obtener más información sobre cómo crear una extensión con una ventana de herramientas, consulte [Creación de una extensión con una ventana](#) de herramientas .

Para suscribirse a eventos rdt

1. Abra el archivo **RDTE Explorer Window Control.xaml** y **button1** elimine el botón denominado . Agregue [ListBox](#) un control y acepte el nombre predeterminado. El elemento Grid debe tener este aspecto:

```
<Grid>
    <StackPanel Orientation="Vertical" Margin="-10,10,10,0">
        <TextBlock Margin="10" HorizontalAlignment="Center">RDTE Explorer Window</TextBlock>
        <ListBox x:Name="listBox" Height="100" />
    </StackPanel>
</Grid>
```

2. Abra el archivo **RDTE Explorer Window.cs** en la vista de código. Agregue las siguientes directivas using al inicio del archivo.

```
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;
```

3. Modifique **RDTE Explorer Window** la clase para que, además de derivar de la [ToolWindowPane](#) clase, implemente la [IVsRunningDocTableEvents](#) interfaz.

```
public class RDTE Explorer Window : ToolWindowPane, IVsRunningDocTableEvents
{
    . . .
}
```

4. Implemente [IVsRunningDocTableEvents](#).

- Implemente la interfaz. Coloque el cursor en el nombre [IVsRunningDocTableEvents](#). Debería ver una bombilla en el margen izquierdo. Haga clic en la flecha abajo a la derecha de la bombilla y seleccione **Implementar interfaz**.

5. En cada método en la `throw new NotImplementedException();` interfaz, substituya la línea con esto:

```
return VSConstants.S_OK;
```

6. Agregue un campo de cookie a la clase `RDTEplorerWindow`.

```
private uint rdtCookie;
```

Esto contiene la cookie que [AdviseRunningDocTableEvents](#) devuelve el método.

7. Invalide el método `Initialize()` de `RDTEplorerWindow` para registrarse para eventos RDT. Siempre debe obtener servicios en el `ToolWindowPane Initialize()` método, no en el constructor.

```
protected override void Initialize()
{
    IVsRunningDocumentTable rdt = (IVsRunningDocumentTable)
    this.GetService(typeof(SVsRunningDocumentTable));
    rdt.AdviseRunningDocTableEvents(this, out rdtCookie);
}
```

Se [SVsRunningDocumentTable](#) llama al servicio [IVsRunningDocumentTable](#) para obtener una interfaz. El [AdviseRunningDocTableEvents](#) método conecta eventos RDT a [IVsRunningDocTableEvents](#) un objeto que implementa, en este caso, un objeto `RDTEplorer`.

8. Actualice el método `Dispose()` de `RDTEplorerWindow`.

```
protected override void Dispose(bool disposing)
{
    // Release the RDT cookie.
    IVsRunningDocumentTable rdt = (IVsRunningDocumentTable)
    Package.GetGlobalService(typeof(SVsRunningDocumentTable));
    rdt.UnadviseRunningDocTableEvents(rdtCookie);

    base.Dispose(disposing);
}
```

El [UnadviseRunningDocTableEvents](#) método elimina la `RDTEplorer` conexión entre y la notificación de eventos RDT.

9. Agregue la siguiente línea al [OnBeforeLastDocumentUnlock](#) cuerpo del `return` controlador, justo antes de la instrucción.

```
public int OnBeforeLastDocumentUnlock(uint docCookie, uint dwRDTLockType, uint dwReadLocksRemaining,
uint dwEditLocksRemaining)
{
    ((RDTEplorerWindowControl)this.Content).listBox.Items.Add("Entering OnBeforeLastDocumentUnlock");
    return VSConstants.S_OK;
}
```

10. Agregue una línea similar al [OnAfterFirstDocumentLock](#) cuerpo del controlador y a otros eventos que desee ver en el cuadro de lista.

```
public int OnAfterFirstDocumentLock(uint docCookie, uint dwRDTLockType, uint dwReadLocksRemaining, uint dwEditLocksRemaining)
{
    ((RDTEExplorerWindowControl)this.Content).listBox.Items.Add("Entering OnAfterFirstDocumentLock");
    return VSConstants.S_OK;
}
```

11. Compile la solución y comience la depuración. Aparece la instancia experimental de Visual Studio.

12. Abra la **ventana RDTEExplorerWindow** (Ver / Otras ventanas / RDTEExplorerWindow).

La ventana **RDTEExplorerWindow** se abre con una lista de eventos vacía.

13. Abra o cree una solución.

A `OnBeforeLastDocument` `OnAfterFirstDocument` medida que se desencadenan los eventos, la notificación de cada evento aparece en la lista de eventos.

Ampliar la ventana Salida

06/04/2020 • 7 minutes to read • [Edit Online](#)

La ventana **Salida** es un conjunto de paneles de texto de lectura y escritura. Visual Studio tiene estos paneles integrados: **Compilar**, en el que los Visual Studio proyectos comunican mensajes sobre compilaciones y **General**, en los que se comunican mensajes sobre el IDE. Los proyectos obtienen **Build** una referencia [IVsBuildableProjectCfg](#) al panel Compilar automáticamente a través de [SVsGeneralOutputWindowPane](#) los métodos de interfaz y Visual Studio ofrece acceso directo al panel **General** a través del servicio. Además de los paneles integrados, puede crear y administrar sus propios paneles personalizados.

Puede controlar **Output** la ventana Salida [IVsOutputWindow](#) [IVsOutputWindowPane](#) directamente a través de las interfaces y. La [IVsOutputWindow](#) interfaz, que [SVsOutputWindow](#) ofrece el servicio, define métodos para crear, recuperar y destruir paneles de ventana de **salida**. La [IVsOutputWindowPane](#) interfaz define métodos para mostrar paneles, ocultar paneles y manipular su texto. Una forma alternativa de **Output** controlar el [OutputWindow](#) salida [OutputWindowPane](#) ventana es a través de la y objetos en el modelo de objetos de automatización de Visual Studio. Estos objetos encapsulan casi toda [IVsOutputWindow](#) [IVsOutputWindowPane](#) la funcionalidad de las interfaces. Además, [OutputWindow](#) [OutputWindowPane](#) los objetos y agregan alguna funcionalidad de nivel superior para facilitar la enumeración de los paneles de la ventana **Salida** y la recuperación de texto de los paneles.

Cree una extensión que use el panel Salida

Puede crear una extensión que ejerza diferentes aspectos del panel Salida.

1. Cree un proyecto `TestOutput` VSIX denominado con un comando de menú denominado **TestOutput**. Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. Agregue las siguientes referencias:
 - a. EnvDTE
 - b. EnvDTE80
3. En *TestOutput.cs*, agregue la siguiente instrucción using:

```
using EnvDTE;  
using EnvDTE80;
```

4. En *TestOutput.cs*, `ShowMessageBox` elimine el método. Agregue el siguiente código auxiliar de método:

```
private void OutputCommandHandler(object sender, EventArgs e)  
{  
}  
}
```

5. En el `TestOutput` constructor, cambie el controlador de comandos a `OutputCommandHandler`. Aquí está la parte que agrega los comandos:

```
OleMenuCommandService commandService = this.ServiceProvider.GetService(typeof(IMenuCommandService)) as
OleMenuCommandService;
if (commandService != null)
{
    CommandID menuCommandID = new CommandID(MenuGroup, CommandId);
    EventHandler eventHandler = OutputCommandHandler;
    MenuCommand menuItem = new MenuCommand(eventHandler, menuCommandID);
    commandService.AddCommand(menuItem);
}
```

6. Las secciones siguientes tienen diferentes métodos que muestran diferentes formas de tratar con el panel Salida. Puede llamar a estos métodos al cuerpo del `OutputCommandHandler()` método. Por ejemplo, el código `CreatePane()` siguiente agrega el método especificado en la sección siguiente.

```
private void OutputCommandHandler(object sender, EventArgs e)
{
    CreatePane(new Guid(), "Created Pane", true, false);
}
```

Crear una ventana de salida con IVsOutputWindow

En este ejemplo se muestra cómo crear [IVsOutputWindow](#) un nuevo panel de ventana de **salida** mediante la interfaz.

```
void CreatePane(Guid paneGuid, string title,
    bool visible, bool clearWithSolution)
{
    IVsOutputWindow output =
        (IVsOutputWindow)GetService(typeof(SVsOutputWindow));
    IVsOutputWindowPane pane;

    // Create a new pane.
    output.CreatePane(
        ref paneGuid,
        title,
        Convert.ToInt32(visible),
        Convert.ToInt32(clearWithSolution));

    // Retrieve the new pane.
    output.GetPane(ref paneGuid, out pane);

    pane.OutputString("This is the Created Pane \n");
}
```

Si agrega este método a la extensión dada en la sección anterior, al hacer clic en el comando **Invocar TestOutput** debería ver la ventana **Salida** con un encabezado que dice **Mostrar salida desde: CreatedPane** y las palabras **Este es el panel Creado** en el propio panel.

Crear una ventana de salida con OutputWindow

En este ejemplo se **Output** muestra cómo crear [OutputWindow](#) un panel de ventana Salida mediante el objeto.


```

void CreatePane(string title)
{
    DTE2 dte = (DTE2)GetService(typeof(DTE));
    OutputWindowPanes panes =
        dte.ToolWindows.OutputWindow.OutputWindowPanes;

    try
    {
        // If the pane exists already, write to it.
        panes.Item(title);
    }
    catch (ArgumentException)
    {
        // Create a new pane and write to it.
        panes.Add(title);
    }
}

```

Aunque [OutputWindowPanes](#) la colección le permite recuperar un panel de la ventana **Salida** por su título, no se garantiza que los títulos del panel sean únicos. Cuando dude de la unicidad [GetPane](#) de un título, use el método para recuperar el panel correcto por su GUID.

Si agrega este método a la extensión dada en la sección anterior, al hacer clic en el comando **Invocar TestOutput** debería ver la ventana Salida con un encabezado que dice **Mostrar salida desde: DTEPane** y las palabras **Añadido panel DTE** en el propio panel.

Eliminar una ventana de salida

En este ejemplo se muestra cómo eliminar un panel de ventana **Salida**.

```

void DeletePane(Guid paneGuid)
{
    IVsOutputWindow output =
        (IVsOutputWindow)ServiceProvider.GetService(typeof(SVsOutputWindow));

    IVsOutputWindowPane pane;
    output.GetPane(ref paneGuid, out pane);

    if (pane == null)
    {
        CreatePane(paneGuid, "New Pane\n", true, true);
    }
    else
    {
        output.DeletePane(ref paneGuid);
    }
}

```

Si agrega este método a la extensión dada en la sección anterior, al hacer clic en el comando **Invocar TestOutput** debería ver la ventana Salida con un encabezado que dice **Mostrar salida desde: Nuevo panel** y las palabras **Añadido panel creado** en el propio panel. Si vuelve a hacer clic en el comando **Invocar TestOutput**, se eliminará el panel.

Obtenga el panel General de la ventana Salida

En este ejemplo se muestra cómo obtener el panel **General** integrado de la ventana **Salida**.

```
IVsOutputWindowPane GetGeneralPane()
{
    return (IVsOutputWindowPane)GetService(
        typeof(SVsGeneralOutputWindowPane));
}
```

Si agrega este método a la extensión dada en la sección anterior, al hacer clic en el comando **Invocar TestOutput** debería ver que la ventana **Salida** muestra las palabras del panel **General encontrado** en el panel.

Obtenga el panel Compilar de la ventana Salida

En este ejemplo se muestra cómo buscar el panel **Compilar** y escribir en él. Dado que el panel **Compilar** no está activado de forma predeterminada, también lo activa.

```
void OutputTaskItemStringExExample(string buildMessage)
{
    EnvDTE80.DTE2 dte = (EnvDTE80.DTE2)ServiceProvider.GetService(typeof(EnvDTE.DTE));

    EnvDTE.OutputWindowPanes panes =
        dte.ToolWindows.OutputWindow.OutputWindowPanes;
    foreach (EnvDTE.OutputWindowPane pane in panes)
    {
        if (pane.Name.Contains("Build"))
        {
            pane.OutputString(buildMessage + "\n");
            pane.Activate();
            return;
        }
    }
}
```

Amplíe las ventanas Propiedades, Lista de tareas, Salida y Opciones

06/04/2020 • 20 minutes to read • [Edit Online](#)

Puede tener acceso a cualquier ventana de herramientas en Visual Studio. En este tutorial se muestra cómo integrar información sobre la ventana de herramientas en una nueva página **Opciones** y una nueva configuración en la página **Propiedades**, así como cómo escribir en las ventanas **Lista** de tareas y **Salida**.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear una extensión con una ventana de herramientas

1. Cree un proyecto denominado **ToDoList** con la plantilla VSIX y agregue una plantilla de elemento de ventana de herramientas personalizada denominada **ToDoWindow**.

NOTE

Para obtener más información sobre cómo crear una extensión con una ventana de herramientas, consulte [Crear una extensión con una ventana](#) de herramientas .

Configurar la ventana de herramientas

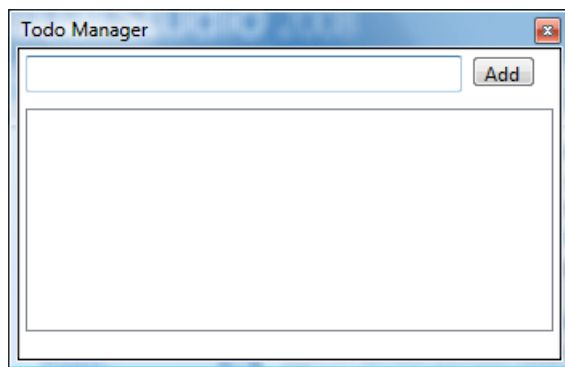
Agregue un cuadro de texto en el que escribir un nuevo elemento de tareas pendientes, un botón para agregar el nuevo elemento a la lista y un listbox para mostrar los elementos de la lista.

1. En *ToDoWindow.xaml*, elimine los controles Button, TextBox y StackPanel de UserControl.

NOTE

Esto no elimina el controlador de eventos `button1_Click`, que reutilizará en un paso posterior.

2. En la sección **Todos los controles WPF** del Cuadro de **herramientas**, arrastre un control **Canvas** a la cuadrícula.
3. Arrastre un cuadro de **texto**, un **botón** y un **cuadro de** lista al lienzo. Organice los elementos para que el cuadro de texto y el botón están en el mismo nivel y el ListBox rellena el resto de la ventana debajo de ellos, como en la imagen siguiente.



- En el panel XAML, busque Button y establezca su content propiedad **Add**. Vuelva a conectar el controlador de eventos `Click="button1_Click"` button al control Button agregando un atributo. El bloque Canvas debería tener este aspecto:

```
<Canvas HorizontalAlignment="Left" Width="306">
    <TextBox x:Name="textBox" HorizontalAlignment="Left" Height="23" Margin="10,10,0,0"
    TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="208"/>
    <Button x:Name="button" Content="Add" HorizontalAlignment="Left" Margin="236,13,0,0"
    VerticalAlignment="Top" Width="48" Click="button1_Click"/>
    <ListBox x:Name="listBox" HorizontalAlignment="Left" Height="222" Margin="10,56,0,0"
    VerticalAlignment="Top" Width="274"/>
</Canvas>
```

Personalizar el constructor

- En el archivo *TodoWindowControl.xaml.cs*, agregue la siguiente directiva using:

```
using System;
```

- Agregue una referencia pública a *TodoWindow* y haga que el *TodoWindowControl* constructor tome un *TodoWindow* parámetro. El código debe tener este aspecto:

```
public TodoWindow parent;

public TodoWindowControl(TodoWindow window)
{
    InitializeComponent();
    parent = window;
}
```

- En *TodoWindow.cs*, cambie *TodoWindowControl* constructor para incluir el *TodoWindow* parámetro. El código debe tener este aspecto:

```
public TodoWindow() : base(null)
{
    this.Caption = "TodoWindow";
    this.BitmapResourceID = 301;
    this.BitmapIndex = 1;

    this.Content = new TodoWindowControl(this);
}
```

Crear una página Opciones

Puede proporcionar una página en el cuadro de diálogo **Opciones** para que los usuarios puedan cambiar la configuración de la ventana de herramientas. Crear una página Options requiere una clase que describa las

opciones y una entrada en el *archivo* `TodoListPackage.cs` o `TodoListPackage.vb`.

1. Agregue una clase denominada `ToolsOptions.cs`. Haga `ToolsOptions` que la `DialogPage` clase herede de .

```
class ToolsOptions : DialogPage
{
}
```

2. Agregue la siguiente directiva de uso:

```
using Microsoft.VisualStudio.Shell;
```

3. La página Opciones de este tutorial solo proporciona una opción denominada DaysAhead. Agregue un campo privado denominado **daysAhead** y `ToolsOptions` una propiedad denominada **DaysAhead** a la clase:

```
private double daysAhead;

public double DaysAhead
{
    get { return daysAhead; }
    set { daysAhead = value; }
}
```

Ahora debe hacer que el proyecto tenga en cuenta esta página Opciones.

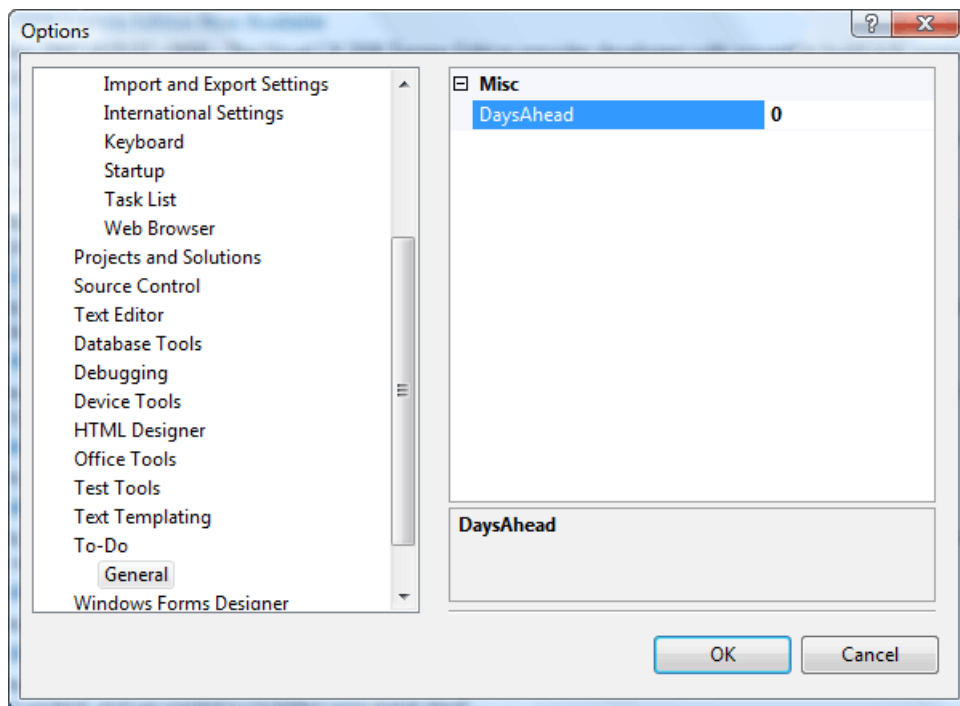
Haga que la página Opciones esté disponible para los usuarios

1. En `TodoWindowPackage.cs`, `ProvideOptionPageAttribute` agregue `TodoWindowPackage` a la clase:

```
[ProvideOptionPage(typeof(ToolsOptions), "ToDo", "General", 101, 106, true)]
```

2. El primer parámetro para el `ProvideOptionPage` constructor `ToolsOptions` es el tipo de la clase , que creó anteriormente. El segundo parámetro, "ToDo", es el nombre de la categoría en el cuadro de diálogo **Opciones**. El tercer parámetro, "General", es el nombre de la subcategoría del cuadro de diálogo **Opciones** donde estará disponible la página Opciones. Los dos parámetros siguientes son identificadores de recursos para cadenas; el primero es el nombre de la categoría, y el segundo es el nombre de la subcategoría. El parámetro final determina si se puede acceder a esta página mediante la automatización.

Cuando un usuario abre la página Opciones, debe ser similar a la siguiente imagen.



Observe la categoría **To-Do** y la subcategoría **General**.

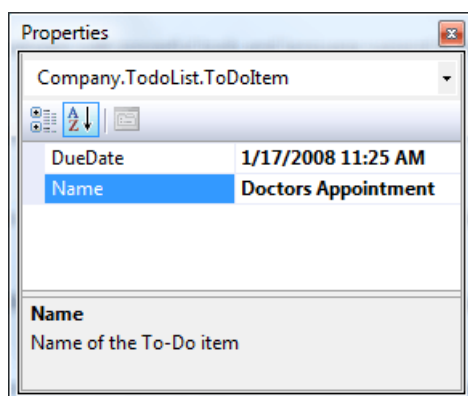
Poner los datos a disposición de la ventana Propiedades

Puede hacer que la información de la `ToDoItem` lista de tareas pendientes esté disponible mediante la creación de una clase denominada que almacena información sobre los elementos individuales en la lista Tareas pendientes.

1. Agregue una clase denominada `ToDoItem.cs`.

Cuando la ventana de herramientas está disponible para los usuarios, los elementos de la `ListBox` se representarán por `ToDoItems`. Cuando el usuario selecciona uno de estos elementos en el `ListBox`, la ventana **Propiedades** mostrará información sobre el elemento.

Para que los datos estén disponibles en la ventana **Propiedades**, convierta los datos en propiedades públicas que tengan dos atributos especiales `Description` y `Category`. `Description` es el texto que aparece en la parte inferior de la ventana **Propiedades**. `Category` determina dónde debe aparecer la propiedad cuando se muestra la ventana **Propiedades** en la vista **Clasificado**. En la siguiente imagen, la ventana **Propiedades** se encuentra en la vista **Clasificado**, la propiedad **Nombre** de la categoría Campos de torsión está seleccionada y la descripción de la propiedad **Nombre** se muestra en la parte inferior de la ventana.



2. Agregue las siguientes directivas using el archivo `ToDoItem.cs`.

```
using System.ComponentModel;
using System.Windows.Forms;
using Microsoft.VisualStudio.Shell.Interop;
```

3. Agregue `public` el modificador de acceso a la declaración de clase.

```
public class TodoItem
{
}
```

Agregue las dos `Name` `DueDate` propiedades y . Haremos lo `UpdateList()` y `CheckForErrors()` más tarde.

```
public class TodoItem
{
    private TodoWindowControl parent;
    private string name;
    [Description("Name of the ToDo item")]
    [Category("ToDo Fields")]
    public string Name
    {
        get { return name; }
        set
        {
            name = value;
            parent.UpdateList(this);
        }
    }

    private DateTime dueDate;
    [Description("Due date of the ToDo item")]
    [Category("ToDo Fields")]
    public DateTime DueDate
    {
        get { return dueDate; }
        set
        {
            dueDate = value;
            parent.UpdateList(this);
            parent.CheckForErrors();
        }
    }
}
```

4. Agregue una referencia privada al control de usuario. Agregue un constructor que tome el control de usuario y el nombre de este elemento ToDo. Para encontrar el `daysAhead` valor de , obtiene la propiedad de página Opciones.

```

private TodoWindowControl parent;

public TodoItem(TodoWindowControl control, string itemName)
{
    parent = control;
    name = itemName;
    dueDate = DateTime.Now;

    double daysAhead = 0;
    IVsPackage package = parent.parent.Package as IVsPackage;
    if (package != null)
    {
        object obj;
        package.GetAutomationObject("ToDo.General", out obj);

        ToolsOptions options = obj as ToolsOptions;
        if (options != null)
        {
            daysAhead = options.DaysAhead;
        }
    }

    dueDate = dueDate.AddDays(daysAhead);
}

```

5. Dado que las `TodoItem` instancias de la clase se almacenarán `ToString` en el ListBox `ToString` y el ListBox llamará a la función, debe sobrecargar la función. Agregue el código siguiente a *TodoItem.cs*, después del constructor y antes del final de la clase.

```

public override string ToString()
{
    return name + " Due: " + dueDate.ToShortDateString();
}

```

6. En *TodoWindowControl.xaml.cs*, agregue métodos de código auxiliar a `TodoWindowControl` la clase para los `CheckForError` métodos y. `UpdateList` Colóquelos después de `ProcessDialogChar` y antes del final del archivo.

```

public void CheckForErrors()
{
}
public void UpdateList(TodoItem item)
{
}

```

El `CheckForError` método llamará a un método que tenga el mismo nombre en el objeto primario y ese método comprobará si se han producido errores y los controlará correctamente. El `UpdateList` método actualizará el ListBox en el control primario; se llama al `Name` método `DueDate` cuando cambian las propiedades y de esta clase. Se implementarán más adelante.

Integrar en la ventana Propiedades

Ahora escriba el código que administra el ListBox, que se vinculará a la **ventana Propiedades**.

Debe cambiar el controlador de clic de botón para leer el cuadro de texto, crear un `TodoItem` y lo agrega a la ListBox.

1. Reemplace la `button1_Click` función existente con código que crea un nuevo `TodoItem` y lo agrega a

ListBox. Llama `TrackSelection()` a , que se definirá más adelante.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    if (textBox.Text.Length > 0)
    {
        var item = new TodoItem(this, textBox.Text);
        listBox.Items.Add(item);
        TrackSelection();
        CheckForErrors();
    }
}
```

2. En la vista Diseño, seleccione el control ListBox. En la ventana **Propiedades**, haga clic en el botón **Controladores** de eventos y busque el evento **SelectionChanged**. Rellene el cuadro de texto con **listBox_SelectionChanged**. Al hacerlo, se agrega un código auxiliar para un SelectionChanged controlador y se asigna al evento.
3. Implemente el método `TrackSelection()`. Puesto que necesitará obtener [SVsUIShell STrackSelection](#) los servicios, [GetService](#) necesita hacer que el accesible por el TodoWindowControl. Agregue el siguiente método a la clase `TodoWindow` :

```
internal object GetVsService(Type service)
{
    return GetService(service);
}
```

4. Agregue las siguientes directivas using a *TodoWindowControl.xaml.cs*:

```
using System.Runtime.InteropServices;
using Microsoft.VisualStudio.Shell.Interop;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Shell;
```

5. Rellene el selectionChanged controlador de la siguiente manera:

```
private void listBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    TrackSelection();
}
```

6. Ahora, rellene la función TrackSelection, que proporcionará integración con la ventana **Propiedades**. Esta función se llama cuando el usuario agrega un elemento a la ListBox o hace clic en un elemento en el ListBox. Agrega el contenido de la ListBox a un SelectionContainer y pasa el [OnSelectChange](#) SelectionContainer al controlador de eventos de la ventana **Properties**. El servicio TrackSelection realiza un seguimiento de los objetos seleccionados en la interfaz de usuario (UI) y muestra sus propiedades

```

private SelectionContainer mySelContainer;
private System.Collections.ArrayList mySelItems;
private IVsWindowFrame frame = null;

private void TrackSelection()
{
    if (frame == null)
    {
        var shell = parent.GetVsService(typeof(SVsUIShell)) as IVsUIShell;
        if (shell != null)
        {
            var guidPropertyBrowser = new
                Guid(ToolWindowGuids.PropertyBrowser);
            shell.FindToolWindow((uint)__VSFINDTOOLWIN.FTW_fForceCreate,
                ref guidPropertyBrowser, out frame);
        }
    }
    if (frame != null)
    {
        frame.Show();
    }
    if (mySelContainer == null)
    {
        mySelContainer = new SelectionContainer();
    }

    mySelItems = new System.Collections.ArrayList();

    var selected = listBox.SelectedItem as TodoItem;
    if (selected != null)
    {
        mySelItems.Add(selected);
    }

    mySelContainer.SelectedObjects = mySelItems;

    ITrackSelection track = parent.GetVsService(typeof(STrackSelection))
        as ITrackSelection;
    if (track != null)
    {
        track.OnSelectChange(mySelContainer);
    }
}

```

Ahora que tiene una clase que puede usar la ventana **Propiedades**, puede integrar la ventana **Propiedades** con la ventana de herramientas. Cuando el usuario hace clic en un elemento en el ListBox en la ventana de herramientas, la ventana **Propiedades** debe actualizarse en consecuencia. De forma similar, cuando el usuario cambia un elemento de tareas pendientes en el **propiedades** ventana, el elemento asociado debe actualizarse.

7. Ahora, agregue el resto del código de la función UpdateList en *TodoWindowControl.xaml.cs*. Debe quitar y volver a agregar el TodoItem modificado desde el ListBox.

```

public void UpdateList(TodoItem item)
{
    var index = listBox.SelectedIndex;
    listBox.Items.RemoveAt(index);
    listBox.Items.Insert(index, item);
    listBox.SelectedItem = index;
}

```

8. Pruebe el código. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.

9. Abra la página **Opciones de herramientas**. > Debería ver la categoría Tareas pendientes en el panel izquierdo. Las categorías se enumeran alfabéticamente, así que mire bajo las Ts.
10. En la página Opciones de `DaysAhead` **Todo**, debería ver la propiedad establecida en **0**. Cámbielo a **2**.
11. En el menú **Ver / Otras ventanas**, abra **TodoWindow**. Escriba **EndDate** en el cuadro de texto y haga clic en **Agregar**.
12. En el cuadro de lista debería ver una fecha dos días después de hoy.

Agregue texto a la ventana Salida y elementos a la Lista de tareas

Para la lista de **tareas**, cree un nuevo objeto de tipo Task y, a continuación, agregue ese objeto Task a la **lista** de tareas llamando a su `Add` método. Para escribir en el **salida** `GetPane` ventana, llame a su método para `OutputString` obtener un objeto de panel y, a continuación, llame al método del objeto de panel.

1. En `TodoWindowControl.xaml.cs`, `button1_Click` en el método, agregue código para obtener el panel **General** de la salida ventana (que es el valor predeterminado) y escribir en él. Este es el aspecto que debe tener el método:

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox.Text.Length > 0)
    {
        var item = new TodoItem(this, textBox.Text);
        listBox.Items.Add(item);

        var outputWindow = parent.GetVsService(
            typeof(SVsOutputWindow)) as IVsOutputWindow;
        IVsOutputWindowPane pane;
        Guid guidGeneralPane = VSConstants.GUID_OutWindowGeneralPane;
        outputWindow.GetPane(ref guidGeneralPane, out pane);
        if (pane != null)
        {
            pane.OutputString(string.Format(
                "To Do item created: {0}\r\n",
                item.ToString()));
        }
        TrackSelection();
        CheckForErrors();
    }
}
```

2. Para agregar elementos a la lista de tareas, necesita a para agregar una clase anidada a la `TodoWindowControl` clase. La clase anidada debe `TaskProvider` derivar de `.` Agregue el código siguiente al `TodoWindowControl` final de la clase.

```
[Guid("72de1eAD-a00c-4f57-bff7-57edb162d0be")]
public class TodoWindowTaskProvider : TaskProvider
{
    public TodoWindowTaskProvider(IServiceProvider sp)
        : base(sp)
    {
    }
}
```

3. A continuación, agregue `TodoTaskProvider` una `CreateProvider()` referencia `TodoWindowControl` privada y un método a la clase. El código debe tener este aspecto:

```
private TodoWindowTaskProvider taskProvider;
private void CreateProvider()
{
    if (taskProvider == null)
    {
        taskProvider = new TodoWindowTaskProvider(parent);
        taskProvider.ProviderName = "To Do";
    }
}
```

4. Agregar `ClearError()`, que borra la `ReportError()` lista de tareas y , que agrega `TodoWindowControl` una entrada a la lista de tareas, a la clase.

```
private void ClearError()
{
    CreateProvider();
    taskProvider.Tasks.Clear();
}
private void ReportError(string p)
{
    CreateProvider();
    var errorTask = new Task();
    errorTask.CanDelete = false;
    errorTask.Category = TaskCategory.Comments;
    errorTask.Text = p;

    taskProvider.Tasks.Add(errorTask);

    taskProvider.Show();

    var taskList = parent.GetVsService(typeof(SVsTaskList))
        as IVsTaskList2;
    if (taskList == null)
    {
        return;
    }

    var guidProvider = typeof(TodoWindowTaskProvider).GUID;
    taskList.SetActiveProvider(ref guidProvider);
}
```

5. Ahora implemente `CheckForErrors` el método, como se indica a continuación.

```
public void CheckForErrors()
{
    foreach (TodoItem item in listBox.Items)
    {
        if (item.DueDate < DateTime.Now)
        {
            ReportError("To Do Item is out of date: "
                + item.ToString());
        }
    }
}
```

Prueba

1. Compile la solución y comience la depuración. Aparece la instancia experimental.
2. Abra **TodoWindow** (**Ver > otras ventanas > TodoWindow**).

3. Escriba algo en el cuadro de texto y, a continuación, haga clic en **Agregar**.

Una fecha de vencimiento 2 días después de hoy se agrega al cuadro de lista. No se generan errores y la **lista de tareas (Ver > lista de tareas)** no debe tener entradas.

4. Ahora cambie la configuración en **la > página Opciones > de herramientas para hacer** de 2 a 0.

5. Escriba algo más en **TodoWindow** y, a continuación, haga clic en **Agregar** de nuevo. Esto desencadena un error y también una entrada en la **lista de tareas**.

A medida que agrega elementos, la fecha inicial se establece en ahora más 2 días.

6. En el menú **Ver**, haga clic en **Salida** para abrir la ventana **Salida**.

Tenga en cuenta que cada vez que agrega un elemento, se muestra un mensaje en el panel Lista de **tareas**.

7. Haga clic en uno de los elementos del cuadro de lista.

La ventana **Propiedades** muestra las dos propiedades del elemento.

8. Cambie una de las propiedades y, a continuación, pulse **Intro**.

El elemento se actualiza en el ListBox.

Crear y administrar cuadros de diálogo modales

06/04/2020 • 4 minutes to read • [Edit Online](#)

Al crear un cuadro de diálogo modal dentro de Visual Studio, debe asegurarse de que la ventana primaria del cuadro de diálogo está deshabilitada mientras se muestra el cuadro de diálogo y, a continuación, volver a habilitar la ventana primaria después de cerrar el cuadro de diálogo. Si no lo hace, puede recibir el error: Microsoft Visual Studio no se puede cerrar porque un cuadro de *diálogo modal está activo*. *Cierre el cuadro de diálogo activo e inténtelo de nuevo*.

Hay dos maneras de hacer esto. La forma recomendada, si tiene un cuadro de [DialogWindow](#) diálogo WPF, [ShowModal](#) es derivarlo de y, a continuación, llamar para mostrar el cuadro de diálogo. Si lo hace, no es necesario administrar el estado modal de la ventana primaria.

Si el cuadro de diálogo no es WPF, o por [DialogWindow](#) alguna otra razón no puede derivar [GetDialogOwnerHwnd](#) la clase de cuadro de [EnableModeless](#) diálogo de , a continuación, debe obtener el elemento primario del cuadro de diálogo llamando y administrar el estado modal usted mismo, llamando al método con un parámetro de 0 (false) antes de mostrar el cuadro de diálogo y llamar al método de nuevo con un parámetro de 1 (true) después de cerrar el cuadro de diálogo.

Crear un cuadro de diálogo derivado de DialogWindow

1. Cree un proyecto VSIX denominado **OpenDialogTest** y agregue un comando de menú denominado **OpenDialog**. Para obtener más información sobre cómo hacerlo, vea [Crear una extensión con un comando](#) de menú .
2. Para usar [DialogWindow](#) la clase, debe agregar referencias a los ensamblados siguientes (en la pestaña Marco del cuadro de diálogo **Agregar referencia**):

- *PresentationCore*
- *PresentationFramework*
- *WindowsBase*
- *System.Xaml*

3. En *OpenDialog.cs*, `using` agregue la siguiente instrucción:

```
using Microsoft.VisualStudio.PlatformUI;
```

4. Declare una `TestDialogWindow` clase denominada [DialogWindow](#) que derive de:

```
class TestDialogWindow : DialogWindow
{
    . . .
}
```

5. Para minimizar y maximizar el cuadro [HasMaximizeButton](#) de [HasMinimizeButton](#) diálogo, establezca y en true:

```
internal TestDialogWindow()
{
    this.HasMaximizeButton = true;
    this.HasMinimizeButton = true;
}
```

6. En `OpenDialog.ShowMessageBox` el método, reemplace el código existente por el siguiente:

```
TestDialogWindow testDialog = new TestDialogWindow();
testDialog.ShowDialog();
```

7. Compile y ejecute la aplicación. Debería aparecer la instancia experimental de Visual Studio. En el menú **Herramientas** de la instancia experimental debería ver un comando denominado **Invocar OpenDialog**. Al hacer clic en este comando, debería ver la ventana de diálogo. Usted debe ser capaz de minimizar y maximizar la ventana.

Crear y administrar un cuadro de diálogo no derivado de DialogWindow

1. Para este procedimiento, puede usar la solución **OpenDialogTest** que creó en el procedimiento anterior, con las mismas referencias de ensamblado.
2. Agregue las `using` siguientes declaraciones:

```
using System.Windows;
using Microsoft.Internal.VisualStudio.PlatformUI;
```

3. Cree una `TestDialogWindow2` clase denominada **Window** que se derive de:

```
class TestDialogWindow2 : Window
{. . .}
```

4. Agregue una referencia **IVsUIShell** privada a:

```
private IVsUIShell shell;
```

5. Agregue un constructor que **IVsUIShell** establezca la referencia en:

```
public TestDialogWindow2(IVsUIShell uiShell)
{
    shell = uiShell;
}
```

6. En `OpenDialog.ShowMessageBox` el método, reemplace el código existente por el siguiente:

```
IVsUIShell uiShell = (IVsUIShell)ServiceProvider.GetService(typeof(SVsUIShell));

TestDialogWindow2 testDialog2 = new TestDialogWindow2(uiShell);
//get the owner of this dialog
IntPtr hwnd;
uiShell.GetDialogOwnerHwnd(out hwnd);
testDialog2.WindowStartupLocation = System.Windows.WindowStartupLocation.CenterOwner;
uiShell.EnableModeless(0);
try
{
    WindowHelper.ShowModal(testDialog2, hwnd);
}
finally
{
    // This will take place after the window is closed.
    uiShell.EnableModeless(1);
}
```

7. Compile y ejecute la aplicación. En el menú **Herramientas** debería ver un comando denominado **Invocar OpenDialog**. Al hacer clic en este comando, debería ver la ventana de diálogo.

Editor y extensiones de servicio de lenguaje

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede ampliar la mayoría de las características del editor de código de Visual Studio. El editor se basa en Windows Presentation Foundation (WPF) y se escribe en código administrado. Aunque este diseño difiere de los diseños de versiones anteriores de Visual Studio, proporciona la mayoría de las mismas características. Para ampliar el editor, utilice Managed Extensibility Framework (MEF).

El SDK de Visual Studio proporciona adaptadores conocidos como *compatibilidades* para admitir VSPackages que se escribieron para versiones anteriores. Sin embargo, si tiene un VSPackage existente, se recomienda actualizarlo a la nueva tecnología para obtener un mejor rendimiento y confiabilidad.

Temas relacionados

TITLE	DESCRIPCIÓN
Crear una extensión con una plantilla de elemento de editor	Introducción al uso de las plantillas de elementos de Editor.
Ampliar el editor y los servicios de idiomas	Enlaces a documentos que introducen el diseño y las características del editor principal y muestran cómo ampliarlo.
Interfaces heredadas en el editor	Vínculos a documentos que explican cómo acceder al editor principal desde el código existente.
Crear editores y diseñadores personalizados	Vínculos a documentos que explican cómo crear editores personalizados.
Extensibilidad del servicio de idiomas heredado	Vínculos a documentos que describen cómo integrar lenguajes de programación en Visual Studio.
Managed Extensibility Framework (MEF)	Presenta Managed Extensibility Framework (MEF).
Windows Presentation Foundation	Presenta Windows Presentation Foundation (WPF).

Crear una extensión con una plantilla de elemento de editor

06/04/2020 • 7 minutes to read • [Edit Online](#)

Puede usar plantillas de elementos que se incluyen en el SDK de Visual Studio para crear extensiones de editor básicas que agreguen clasificadores, adornos y márgenes al editor. Las plantillas de elemento de editor están disponibles para proyectos de Visual C o VSIX de Visual Basic.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una extensión clasificadora

La plantilla de elemento Clasificador de editor crea un clasificador de editor que colorea el texto adecuado (en este caso, todo) en cualquier archivo de texto.

1. En el cuadro de diálogo **Nuevo proyecto**, expanda **Visual C** o **Visual Basic** y, a continuación, haga clic en **Extensibilidad**. En el panel **Plantillas**, seleccione **Proyecto VSIX**. En el cuadro **Nombre**, escriba `TestClassifier`. Haga clic en **OK**.
2. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. Vaya al nodo **Extensibilidad** de Visual C- y seleccione **Clasificador de editor**. Deje el nombre de archivo predeterminado (`EditorClassifier1.cs`).
3. Hay cuatro archivos de código, como se indica a continuación:
 - `EditorClassifier1.cs` contiene `EditorClassifier1` la clase.
 - `EditorClassifier1ClassificationDefinition.cs` contiene `EditorClassifier1ClassificationDefinition` la clase.
 - `EditorClassifier1Format.cs` contiene `EditorClassifier1Format` la clase.
 - `EditorClassifier1Provider.cs` contiene `EditorClassifier1Provider` la clase.
4. Compile la solución y comience la depuración. Aparece la instancia experimental de Visual Studio.

Si abre un archivo de texto, todo el texto se subraya con un fondo violeta.

Crear una extensión de adorno relativo al texto

La plantilla Adornación de texto del editor crea un adorno relativo al texto que decora todas las instancias del carácter de texto 'a' mediante un cuadro que tiene un contorno rojo y un fondo azul. Es relativo al texto porque el cuadro siempre se superpone a los caracteres 'a', incluso cuando se mueven o se reformatean.

1. En el cuadro de diálogo **Nuevo proyecto**, expanda **Visual C** o **Visual Basic** y, a continuación, haga clic en **Extensibilidad**. En el panel **Plantillas**, seleccione **Proyecto VSIX**. En el cuadro **Nombre**, escriba `TestAdornment`. Haga clic en **OK**.
2. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione

Agregar > nuevo elemento. Vaya al nodo **Extensibilidad** de Visual C- y seleccione **Editor Text Adornment**. Deje el nombre de archivo predeterminado (*TextAdornment1.cs/vb*).

3. Hay dos archivos de código, como se indica a continuación:

- *TextAdornment1.cs* contiene `TextAdornment1` la clase.
- *TextAdornment1TextViewCreationListener.cs* contiene `TextAdornment1TextViewCreationListener` la clase.

4. Compile la solución y comience la depuración. Aparece la instancia experimental. Si abre un archivo de texto, todos los caracteres 'a' del texto se describen en rojo sobre un fondo azul.

Crear una extensión de adorno relativo a la ventana gráfica

La plantilla Adornación de ventana gráfica del editor crea un adorno relativo a la ventana gráfica que agrega un cuadro violeta que tiene un contorno rojo en la esquina superior derecha de la ventana gráfica.

NOTE

La **ventana gráfica** es el área de la vista de texto que se muestra actualmente.

Para crear una extensión de adorno de ventana gráfica mediante la plantilla Adornar ventana gráfica del Editor

1. En el cuadro de diálogo **Nuevo proyecto**, expanda **Visual C** o **Visual Basic** y, a continuación, haga clic en **Extensibilidad**. En el panel **Plantillas**, seleccione **Proyecto VSIX**. En el cuadro **Nombre**, escriba `ViewportAdornment`. Haga clic en **OK**.
2. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. Vaya al nodo **Extensibilidad** de Visual C- y seleccione **Editor Viewport Adornment**. Deje el nombre de archivo predeterminado (*ViewportAdornment1.cs/vb*).
3. Hay dos archivos de código, como se indica a continuación:
 - *ViewportAdornment1.cs* contiene `ViewportAdornment1` la clase.
 - *ViewportAdornment1TextViewCreationListener.cs* contiene `ViewportAdornment1TextViewCreationListener` la clase
4. Compile la solución y comience la depuración. Aparece la instancia experimental. Si crea un nuevo archivo de texto, se muestra un cuadro violeta que tiene un contorno rojo en la esquina superior derecha de la ventana gráfica.

Crear una extensión de margen

La plantilla Margen del editor crea un margen verde que aparece junto con las palabras **¡Hola mundo!* debajo de la barra de desplazamiento horizontal.

Para crear una extensión de margen mediante la plantilla Margen de editor

1. En el cuadro de diálogo **Nuevo proyecto**, expanda **Visual C** o **Visual Basic** y, a continuación, haga clic en **Extensibilidad**. En el panel **Plantillas**, seleccione **Proyecto VSIX**. En el cuadro **Nombre**, escriba `MarginExtension`. Haga clic en **OK**.
2. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. Vaya al nodo **Extensibilidad** de Visual C- y seleccione **Margen del editor**. Deje el nombre de archivo predeterminado (*EditorMargin1.cs/vb*).

3. Hay dos archivos de código, como se indica a continuación:

- *EditorMargin1.cs* contiene `EditorMargin1` la clase.
- *EditorMargin1Factory.cs* contiene `EditorMargin1Factory` la clase.

4. Compile este proyecto e inicie la depuración. Aparece la instancia experimental. Si abre un archivo de texto, se muestra un margen verde que tiene las palabras **Hello EditorMargin1** debajo de la barra de desplazamiento horizontal.

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Ampliar el editor y los servicios de idiomas

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede agregar características de servicio de lenguaje (como IntelliSense) a su propio editor y ampliar la mayoría de las características del editor de código de Visual Studio. Para obtener una lista completa de lo que puede ampliar, consulte Servicio de lenguaje y puntos de extensión del editor.

La mayoría de las características del editor se amplían mediante Managed Extensibility Framework (MEF). Por ejemplo, si la característica del editor que desea ampliar es la coloración de sintaxis, puede escribir una parte del *componente* MEF que defina las clasificaciones para las que desea colorear diferente y cómo desea que se manejen. El editor también admite varias extensiones de la misma característica.

La capa de presentación del editor se basa en Windows Presentation Framework (WPF). WPF proporciona una biblioteca de gráficos para el formato de texto flexible y también proporciona visualizaciones como gráficos y animaciones.

El SDK de Visual Studio proporciona adaptadores conocidos como *compatibilidades* para admitir VSPackages que se escribieron para versiones anteriores. Sin embargo, si tiene un VSPackage existente, se recomienda actualizarlo a la nueva tecnología para obtener un mejor rendimiento y confiabilidad.

Temas relacionados

TITLE	DESCRIPCIÓN
Comience con el servicio de lenguaje y las extensiones del editor	Explica cómo crear una extensión para el editor.
Dentro del editor	Describe la estructura general del editor y enumera algunas de sus características.
Managed Extensibility Framework en el editor	Explica cómo usar Managed Extensibility Framework (MEF) con el editor.
Servicio de lenguaje y puntos de extensión del editor	Enumera los puntos de extensión del editor. Los puntos de extensión representan las características del editor que se pueden ampliar.
Tutorial: Crear un adorno de vista, comandos y configuración (guías de columna)	Recorre y explica la creación de un adorno de vista que dibuja líneas guía de columna para ayudarle a mantener el código en un determinado ancho de pantalla. También muestra la configuración de lectura y escritura, así como la declaración e implementación de comandos que se pueden invocar desde la ventana de comandos.
Importaciones de editores	Enumera los servicios que puede importar una extensión.
Adaptar el código heredado al editor	Explica diferentes formas de adaptar el código heredado (anterior a Visual Studio 2010) para ampliar el editor.
Migrar un servicio de lenguaje heredado	Explica cómo migrar un servicio de lenguaje basado en VSPackage.

TITLE	DESCRIPCIÓN
Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo	Muestra cómo vincular un tipo de contenido a una extensión de nombre de archivo.
Tutorial: Crear un glifo de margen	Muestra cómo agregar un icono a un margen.
Tutorial: Resaltar texto	Muestra cómo utilizar <i>etiquetas</i> para resaltar texto.
Tutorial: Agregar esquematización	Muestra cómo agregar esquematización para tipos específicos de llaves.
Tutorial: Mostrar llaves coincidentes	Muestra cómo resaltar llaves coincidentes.
Tutorial: Mostrar información sobre herramientas quickInfo	Muestra cómo mostrar ventanas emergentes de QuickInfo que describen elementos de código como propiedades, métodos y eventos.
Tutorial: Mostrar la ayuda de firma	Muestra cómo mostrar ventanas emergentes que proporcionan información sobre el número y los tipos de parámetros de una firma.
Tutorial: Mostrar la finalización de instrucciones	Muestra cómo implementar la finalización de instrucciones.
Tutorial: Implementar fragmentos de código	Muestra cómo implementar la expansión de fragmento de código.
Tutorial: Mostrar sugerencias de bombillas	Muestra cómo mostrar bombillas para sugerencias de código.
Tutorial: Utilice un comando shell con una extensión de editor	Muestra cómo asociar un comando de menú en un VSPackage con un componente MEF.
Tutorial: Use una tecla de método abreviado con una extensión de editor	Muestra cómo asociar un acceso directo de menú en un VSPackage con un componente MEF.
Managed Extensibility Framework (MEF)	Proporciona información sobre Managed Extensibility Framework (MEF).
Windows Presentation Foundation	Proporciona información acerca de Windows Presentation Foundation (WPF).

Referencia

El editor de Visual Studio incluye los siguientes espacios de nombres.

[Microsoft.VisualStudio.Language.Intellisense](#)

[Microsoft.VisualStudio.Language.StandardClassification](#)

[Microsoft.VisualStudio.Editor](#)

[Microsoft.VisualStudio.Text](#)

[Microsoft.VisualStudio.Text.Adornments](#)

[Microsoft.VisualStudio.Text.Classification](#)

Microsoft.VisualStudio.Text.Differencing

Microsoft.VisualStudio.Text.Document

Microsoft.VisualStudio.Text.Editor

Microsoft.VisualStudio.Text.Editor.OptionsExtensionMethods

Microsoft.VisualStudio.Text.Formatting

Microsoft.VisualStudio.Text.IncrementalSearch

Microsoft.VisualStudio.Text.Operations

Microsoft.VisualStudio.Text.Outlining

Microsoft.VisualStudio.Text.Projection

Microsoft.VisualStudio.Text.Tagging

Microsoft.VisualStudio.Utilities

Comience con el servicio de lenguaje y las extensiones del editor

06/04/2020 • 8 minutes to read • [Edit Online](#)

Puede usar extensiones de editor para agregar características de servicio de lenguaje como esquematización, coincidencia de llaves, IntelliSense y bombillas a su propio lenguaje de programación o a cualquier tipo de contenido. También puede personalizar la apariencia y el comportamiento del editor de Visual Studio, por ejemplo, el color del texto, los márgenes, los adornos y otros elementos visuales. También puede definir su propio tipo de contenido y especificar la apariencia y el comportamiento de las vistas de texto en las que aparece el contenido.

Para empezar a escribir extensiones de editor, use las plantillas de proyecto del editor que se instalan como parte del SDK de Visual Studio. El SDK de Visual Studio es un conjunto descargable de herramientas que facilitan el desarrollo de extensiones de Visual Studio, ya sea mediante VSPackages o mediante Managed Extensibility Framework (MEF).

NOTE

Para obtener más información acerca del SDK de Visual Studio, vea SDK de [Visual Studio](#).

Le recomendamos que obtenga información sobre los siguientes conceptos y tecnologías antes de escribir sus propias extensiones de editor.

Windows Presentation Foundation (WPF) y las extensiones del editor

La interfaz de usuario (UI) del editor de Visual Studio se implementa mediante Windows Presentation Foundation (WPF). WPFWPF proporciona una experiencia visual enriquecida y un modelo de programación coherente que separa los aspectos visuales del código de la lógica empresarial. Puede usar muchos elementos y características de WPFWPF al crear extensiones de editor. Para obtener más información, vea [Windows Presentation Foundation](#).

El marco de extensibilidad administrada (MEF) y las extensiones del editor

El editor de Visual Studio usa Managed Extensibility Framework (MEF) para administrar sus componentes y extensiones. El MEF también permite a los desarrolladores crear más fácilmente extensiones para una aplicación host como Visual Studio. En este marco, se define una extensión según un contrato MEF y se exporta como una parte de componente MEF. La aplicación host administra los componentes findingthem, registering them y asegurándose de que se aplican al contexto correcto.

NOTE

Para obtener más información sobre el MEF en el editor, vea [Managed Extensibility Framework en el editor](#).

Puntos de extensión y extensiones del editor de Visual Studio

Los puntos de extensión del editor son componentes MEF que puede personalizar y ampliar. En algunos casos, se amplía el punto de extensión implementando una interfaz y exportándola junto con los metadatos correctos. En otros casos, solo se declara una extensión y se exporta como un tipo determinado.

Los siguientes son algunos de los tipos básicos de extensiones de editor:

- Márgenes y barras de desplazamiento
- Etiquetas
- Adornos
- Opciones
- IntelliSense

Para obtener más información acerca de los puntos de extensión del editor, vea [Servicio de lenguaje y puntos de extensión del editor](#).

Implementación de extensiones de editor

En Visual Studio, implementar extensiones de editor agregando un archivo de metadatos denominado *source.extension.vsixmanifest* a la solución, compilar la solución y, a continuación, agregar una copia de los archivos binarios y el manifiesto en una carpeta que es conocido por Visual Studio. El archivo de manifiesto define los hechos básicos sobre la extensión (por ejemplo, nombre, autor, versión y tipo de contenido). Para obtener más información sobre el archivo de manifiesto VSIX y cómo implementar extensiones, vea [Enviar extensiones](#) de Visual Studio.

Al instalar una extensión en un equipo, incluya los archivos binarios y el manifiesto en una subcarpeta de carpeta que Visual Studio conoce.

WARNING

No tiene que preocuparse por los detalles de los manifiestos y las ubicaciones de implementación si usa una de las plantillas de extensibilidad del editor que se incluyen en Visual Studio. Las plantillas contienen todo lo necesario para registrar e implementar una extensión.

Ejecutar extensiones en la instancia experimental

Puede aislar la versión de trabajo de Visual Studio mientras desarrolla una extensión implementándola en la siguiente carpeta experimental (en Windows Vista y Windows 7):

```
•%LOCALAPPDATA%-VisualStudio-10.0Exp-Extensiones-Empresa-ExtensionID--ExtensionID---ExtensionID-----  
-----||
```

donde *%LOCALAPPDATA%* es el nombre del usuario que ha iniciado sesión, *Company* es el nombre de la empresa propietaria de la extensión y *ExtensionID* es el identificador de la extensión.

Al implementar una extensión en la ubicación experimental, se ejecuta en modo de depuración. Se inicia una segunda instancia de Visual Studio y se denomina **Microsoft Visual Studio - Experimental Instance**.

Administración de extensiones

Las extensiones de Visual Studio se enumeran en **Extensiones y actualizaciones** (en el menú **Herramientas**). Si está probando una extensión en la instancia experimental, aparece en **Extensiones y actualizaciones** en la instancia experimental, pero no aparece en la instancia de desarrollo.

Para obtener más información, vea [Buscar y usar extensiones](#) de Visual Studio.

Usar plantillas para crear extensiones de editor

Puede usar plantillas de editor para crear extensiones MEF que personalicen clasificadores, adornos y márgenes. Hay plantillas para proyectos de Visual Basic y de C. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).

También puede usar la plantilla De proyecto VSIX para crear extensiones. Esta plantilla proporciona solo los elementos necesarios para implementar cualquier tipo de extensión e incluir el archivo *source.extension.vsixmanifest*, las referencias de ensamblado necesarias y un archivo de proyecto que incluya las tareas de compilación que permiten implementar la extensión. Para obtener más información, consulte [Plantilla de proyecto VSIX](#).

También puede crear componentes MEF de editor a partir de una extensión de paquete de Visual Studio. Consulte los siguientes tutoriales para obtener más información:

- [Tutorial: Uso de un comando shell con una extensión de editor](#)
- [Tutorial: Uso de una tecla de método abreviado con una extensión de editor](#)

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Dentro del editor

06/04/2020 • 46 minutes to read • [Edit Online](#)

El editor se compone de varios subsistemas diferentes, que están diseñados para mantener el modelo de texto del editor separado de la vista de texto y la interfaz de usuario.

Estas secciones describen diferentes aspectos del editor:

- [Resumen de los subsistemas](#)
- [El modelo de texto](#)
- [La vista de texto](#)

Estas secciones describen las características del editor:

- [Etiquetas y clasificadores](#)
- [Adornos](#)
- [Proyección](#)
- [esquematizar](#)
- [Fijaciones de ratón](#)
- [Operaciones del editor](#)
- [IntelliSense](#)

Resumen de los subsistemas

Subsistema de modelos de texto

El subsistema de modelo de texto es responsable de representar el texto y permitir su manipulación. El subsistema de [ITextBuffer](#) modelo de texto contiene la interfaz, que describe la secuencia de caracteres que debe mostrar el editor. Este texto se puede modificar, rastrear y manipular de muchas maneras. El modelo de texto también proporciona tipos para los siguientes aspectos:

- Un servicio que asocia texto con archivos y administra la lectura y escritura en el sistema de archivos.
- Un servicio de diferenciación que encuentra las diferencias mínimas entre dos secuencias de objetos.
- Un sistema para describir el texto en un búfer en términos de subconjuntos del texto en otros búferes.

El subsistema de modelo de texto está libre de conceptos de interfaz de usuario (UI). Por ejemplo, no es responsable del formato del texto o del diseño del texto, y no tiene conocimiento de los adornos visuales que pueden estar asociados con el texto.

Los tipos públicos del subsistema de modelo de texto se encuentran en *Microsoft.VisualStudio.Text.Data.dll* y *Microsoft.VisualStudio.CoreUtility.dll*, que dependen únicamente de la biblioteca de clases base de .NET Framework y Managed Extensibility Framework (MEF).

Subsistema de vista de texto

El subsistema de vista de texto es responsable de dar formato y mostrar texto. Los tipos de este subsistema se dividen en dos capas, dependiendo de si los tipos dependen de Windows Presentation Foundation (WPF). Los tipos más [ITextView](#) [IWpfTextView](#) importantes son y , que controlan el conjunto de líneas de texto que se van a mostrar,

así como el intercalador, la selección y las facilidades para adornar el texto mediante elementos de interfaz de usuario de WPF. Este subsistema también proporciona márgenes alrededor del área de visualización del texto. Estos márgenes se pueden ampliar y pueden contener diferentes tipos de contenido y efectos visuales. Ejemplos de márgenes son las visualizaciones de números de línea y las barras de desplazamiento.

Los tipos públicos del subsistema de vista de texto se encuentran en *Microsoft.VisualStudio.Text.UI.dll* y *Microsoft.VisualStudio.Text.UI.Wpf.dll*. El primer ensamblado contiene los elementos independientes de la plataforma y el segundo contiene los elementos específicos de WPF.

Subsistema de clasificación

El subsistema de clasificación es responsable de determinar las propiedades de fuente para el texto. Un clasificador divide el texto en diferentes clases, por ejemplo, "palabra clave" o "comentario". El mapa de formato de clasificación relaciona estas clases con las propiedades de fuente reales, por ejemplo, "Blue Consolas 10 pt". Esta información la utiliza la vista de texto cuando da formato y representa texto. El etiquetado, que se describe con más detalle más adelante en este tema, permite que los datos se asocien con intervalos de texto.

Los tipos públicos del subsistema de clasificación están contenidos en *Microsoft.VisualStudio.Text.Logic.dll* y interactúan con los aspectos visuales de la clasificación, que se encuentran en *Microsoft.VisualStudio.Text.UI.Wpf.dll*.

Subsistema de operaciones

El subsistema de operaciones define el comportamiento del editor. Proporciona la implementación para los comandos del editor de Visual Studio y el sistema de deshacer.

Una mirada más cercana al modelo de texto y a la vista de texto

El modelo de texto

El subsistema de modelo de texto consta de diferentes agrupaciones de tipos de texto. Estos incluyen el búfer de texto, las instantáneas de texto y los intervalos de texto.

Búferes de texto e instantáneas de texto

La [ITextBuffer](#) interfaz representa una secuencia de caracteres Unicode que se codifican mediante `String` UTF-16, que es la codificación utilizada por el tipo en .NET Framework. Un búfer de texto se puede conservar como un documento del sistema de archivos, pero esto no es necesario.

Se [ITextBufferFactoryService](#) utiliza para crear un búfer de texto vacío o un búfer [TextReader](#) de texto que se inicializa desde una cadena o desde . El búfer de texto se puede conservar [ITextDocument](#) en el sistema de archivos como un archivo .

Cualquier subprocesso puede editar el búfer de texto hasta [TakeThreadOwnership](#) que un subprocesso tome la propiedad del búfer de texto llamando a . Después de eso, solo ese subprocesso puede realizar ediciones.

Un búfer de texto puede pasar por muchas versiones durante su vida útil. Se genera una nueva versión cada vez que se [ITextSnapshot](#) edita el búfer y un inmutable representa el contenido de esa versión del búfer. Dado que las instantáneas de texto son inmutables, puede tener acceso a una instantánea de texto en cualquier subprocesso, sin restricciones, incluso si el búfer de texto que representa sigue cambiando.

Instantáneas de texto y líneas de instantáneas de texto

Puede ver el contenido de una instantánea de texto como una secuencia de caracteres o como una secuencia de líneas. Los caracteres y las líneas se indexan a partir de cero. Una instantánea de texto vacía contiene cero caracteres y una línea vacía. Una línea está delimitada por cualquier secuencia de caracteres de salto de línea Unicode válida o por el principio o el final del búfer. Los caracteres de salto de línea se representan explícitamente en la instantánea de texto y los saltos de línea de una instantánea de texto no tienen que ser todos iguales.

NOTE

Para obtener más información acerca de los caracteres de salto de línea en el editor de Visual Studio, vea [codificaciones y saltos de línea](#).

Una línea de texto se [ITextSnapshotLine](#) representa mediante un objeto, que se puede obtener de una instantánea de texto para un número de línea determinado o para una posición de carácter determinada.

SnapshotPoints, SnapshotSpans y NormalizedSnapshotSpanCollections

A [SnapshotPoint](#) representa una posición de carácter en una instantánea. Se garantiza que la posición se encuentra entre cero y la longitud de la instantánea. A [SnapshotSpan](#) representa un intervalo de texto en una instantánea. Se garantiza que su posición final se encuentra entre cero y la longitud de la instantánea. Consta [NormalizedSnapshotSpanCollection](#) de un [SnapshotSpan](#) conjunto de objetos de la misma instantánea.

Spans y NormalizedSpanCollections

A [Span](#) representa un intervalo que se puede aplicar a un intervalo de texto en una instantánea de texto. Las posiciones de las instantáneas se basan en cero, por lo que los intervalos pueden comenzar en cualquier posición, incluido cero. La `End` propiedad de un intervalo es `Start` igual a `Length` la suma de su propiedad y su propiedad. A `Span` no incluye el carácter indizado por la `End` propiedad. Por ejemplo, un intervalo que tiene `Start` 5 y `Length` 3 tiene `End` 8, e incluye los caracteres en las posiciones 5, 6 y 7. La notación para este intervalo es [5..8).

Dos intervalos se intersecan si tienen alguna posición en común, incluida la posición Final. Por lo tanto, la intersección de [3, 5) y [2, 7) es [3, 5) y la intersección de [3, 5) y [5, 7) es [5, 5). (Tenga en cuenta que [5, 5) es un intervalo vacío.)

Dos intervalos se superponen si tienen posiciones en común, excepto la posición Final. Un intervalo vacío nunca se superpone a ningún otro intervalo y la superposición de dos intervalos nunca está vacía.

A [NormalizedSpanCollection](#) es una lista de intervalos en el orden de las propiedades de inicio de los intervalos. En la lista, se combinan los intervalos superpuestos o colindantes. Por ejemplo, dado el conjunto de intervalos [5..9), [0..1), [3..6) y [9..10), la lista normalizada de intervalos es [0..1), [3..10).

ITextEdit, TextVersion y notificaciones de cambio de texto

El contenido de un búfer de [ITextEdit](#) texto se puede cambiar mediante un objeto. La creación de un objeto `CreateEdit()` de [ITextBuffer](#) este tipo (mediante uno de los métodos de) inicia una transacción de texto que consta de ediciones de texto. Cada edición es un reemplazo de algún intervalo de texto en el búfer por una cadena. Las coordenadas y el contenido de cada edición se expresan en relación con la instantánea del búfer cuando se inició la transacción. El [ITextEdit](#) objeto ajusta las coordenadas de las ediciones que se ven afectadas por otras ediciones en la misma transacción.

Por ejemplo, considere un búfer de texto que contenga esta cadena:

abcdefghij

Aplique una transacción que contenga dos ediciones, una edición que reemplace el `x` intervalo en [2..4) utilizando el carácter `y` y una segunda edición que reemplace el intervalo en [6..9) utilizando el carácter `.` El resultado es este buffer:

abXefYj

Las coordenadas de la segunda edición se calcularon con respecto al contenido del búfer al principio de la transacción, antes de aplicar la primera edición.

Los cambios en el [ITextEdit](#) búfer surten `Apply()` efecto cuando se confirma el objeto llamando a su método. Si

hubo al menos una edición `ITextVersion` no vacía, `ITextSnapshot` se crea una `Changed` nueva, se crea una nueva y se genera un evento. Cada versión de texto tiene una instantánea de texto diferente. Una instantánea de texto representa el estado completo del búfer de texto después de una transacción de edición, pero una versión de texto describe solo los cambios de una instantánea a la siguiente. En general, las instantáneas de texto están diseñadas para usarse una vez y, a continuación, se descartan, mientras que las versiones de texto deben permanecer vivas durante algún tiempo.

Una versión `INormalizedTextChangeCollection` de texto contiene un archivo. Esta colección describe los cambios que, cuando se aplican a la instantánea, producen la instantánea posterior. Cada `ITextChange` parte de la colección contiene la posición del carácter del cambio, la cadena reemplazada y la cadena de reemplazo. La cadena reemplazada está vacía para una inserción básica y la cadena de reemplazo está vacía para una eliminación básica. La colección normalizada siempre `null` es para la versión más reciente del búfer de texto.

Solo `ITextEdit` se puede crear una instancia de un objeto para un búfer de texto en cualquier momento y todas las ediciones de texto se deben realizar en el subproceso que posee el búfer de texto (si se ha reclamado la propiedad). Una edición de texto `Cancel` se puede `Dispose` abandonar llamando a su método o a su método.

`ITextBuffer` también `Insert()` proporciona `Delete()`, `Replace()`, y métodos `ITextEdit` que se asemejan a los que se encuentran en la interfaz. Llamar a estos tiene el `ITextEdit` mismo efecto que crear un objeto, realizar la llamada similar y, a continuación, aplicar la edición.

Puntos de seguimiento y intervalos de seguimiento

Un `ITrackingPoint` representa una posición de carácter en un búfer de texto. Si el búfer se edita de una manera que hace que la posición del carácter cambie, el punto de seguimiento cambia con él. Por ejemplo, si un punto de seguimiento hace referencia a la posición 10 en un búfer y se insertan cinco caracteres al principio del búfer, el punto de seguimiento hace referencia a la posición 15. Si una inserción se produce precisamente en la posición `PointTrackingMode` indicada por el `Positive` `Negative` punto de seguimiento, su comportamiento viene determinado por su `Mode`, que puede ser `Positive` o `Negative`. Si el modo de seguimiento es positivo, el punto de seguimiento hace referencia al mismo carácter, que ahora está al final de la inserción. Si el modo de seguimiento es negativo, el punto de seguimiento hace referencia al primer carácter insertado en la posición original. Si se elimina el carácter en la posición representada por un punto de seguimiento, el punto de seguimiento cambia al primer carácter que sigue al intervalo eliminado. Por ejemplo, si un punto de seguimiento hace referencia al carácter en la posición 5 y se eliminan los caracteres de las posiciones 3 a 6, el punto de seguimiento hace referencia al carácter en la posición 3.

Un `ITrackingSpan` representa un rango de caracteres en lugar de una sola posición. Su comportamiento está `SpanTrackingMode` determinado por su `Mode`. Si el modo de seguimiento de intervalo es `SpanTrackingMode.EdgeInclusive`, el intervalo de seguimiento crece para incorporar texto insertado en sus bordes. Si el modo de seguimiento de intervalo es `SpanTrackingMode.EdgeExclusive`, el intervalo de seguimiento no incorpora texto insertado en sus bordes. Sin embargo, si el modo de seguimiento de intervalo es `SpanTrackingMode.EdgePositive`, una inserción inserta la posición actual hacia el inicio y si el modo de seguimiento de intervalo es `SpanTrackingMode.EdgeNegative`, una inserción empuja la posición actual hacia el final.

Puede obtener la posición de un punto de seguimiento o el intervalo de un intervalo de seguimiento para cualquier instantánea del búfer de texto al que pertenecen. Los puntos de seguimiento y los intervalos de seguimiento se pueden hacer referencia de forma segura desde cualquier subproceso.

Tipos de contenido

Los tipos de contenido son un mecanismo para definir diferentes tipos de contenido. Un tipo de contenido puede ser un tipo de archivo como "text", "code" o "binary", o un tipo de tecnología como "xml", "vb" o "c-". Por ejemplo, la palabra "usar" es una palabra clave tanto en C- como en Visual Basic, pero no en otros lenguajes de programación. Por lo tanto, la definición de esta palabra clave se limitaría a los tipos de contenido "c" y "vb".

Los tipos de contenido se utilizan como filtro para adornos y otros elementos del editor. Muchas características del editor y puntos de extensión se definen por tipo de contenido. Por ejemplo, el color de texto es diferente para los archivos de texto sin formato, los archivos XML y los archivos de código fuente de Visual Basic. Los búferes de

texto generalmente se asignan un tipo de contenido cuando se crean y el tipo de contenido de un búfer de texto se puede cambiar.

Los tipos de contenido pueden heredar varios de otros tipos de contenido. Le [ContentTypeDefinition](#) permite especificar varios tipos base como los elementos primarios de un tipo de contenido determinado.

Los desarrolladores pueden definir sus propios [IContentTypeRegistryService](#) tipos de contenido y registrarlos mediante el archivo . Muchas características del editor se pueden definir con [ContentTypeAttribute](#) respecto a un tipo de contenido específico mediante el uso de la . Por ejemplo, los márgenes del editor, los adornos y los controladores del mouse se pueden definir para que se apliquen solo a los editores que muestran determinados tipos de contenido.

La vista de texto

La parte de vista del patrón de controlador de vista de modelo (MVC) define la vista de texto, el formato de la vista, los elementos gráficos como la barra de desplazamiento y el icono de intercalación. Todos los elementos de presentación del editor de Visual Studio se basan en WPF.

Vistas de texto

La [ITextView](#) interfaz es una representación independiente de la plataforma de una vista de texto. Se utiliza principalmente para mostrar documentos de texto en una ventana, pero también se puede utilizar para otros fines, por ejemplo, en una información sobre herramientas.

La vista de texto hace referencia a diferentes tipos de búferes de texto. La [TextViewModel](#) propiedad hace [ITextViewModel](#) referencia a un objeto que apunta a estos tres búferes de texto diferentes: el búfer de datos, que es el búfer de nivel de datos superior, el búfer de edición, en el que se produce la edición, y el búfer visual, que es el búfer que se muestra en la vista de texto.

El texto se formatea en función de los clasificadores que se adjuntan al búfer de texto subyacente y se adorna mediante los proveedores de adornos que se adjuntan a la propia vista de texto.

El sistema de coordenadas de vista de texto

El sistema de coordenadas de vista de texto especifica posiciones en la vista de texto. En este sistema de coordenadas, el valor x 0.0 corresponde al borde izquierdo del texto que se muestra y el valor y 0.0 corresponde al borde superior del texto que se muestra. La coordenada x aumenta de izquierda a derecha, y la coordenada y aumenta de arriba a abajo.

Una ventana gráfica (la parte del texto visible en la ventana de texto) no se puede desplazar de la misma manera que se desplaza verticalmente. Una ventana gráfica se desplaza horizontalmente cambiando su coordenada izquierda para que se mueva con respecto a la superficie de dibujo. Sin embargo, una ventana gráfica solo se puede desplazar verticalmente cambiando el texto representado, lo que hace que se genere un [LayoutChanged](#) evento.

Las distancias en el sistema de coordenadas corresponden a píxeles lógicos. Si la superficie de representación de texto se muestra sin una transformación de escala, una unidad del sistema de coordenadas de representación de texto corresponde a un píxel de la pantalla.

Márgenes

La [ITextViewMargin](#) interfaz representa un margen y permite controlar la visibilidad del margen y su tamaño. Hay cuatro márgenes predefinidos, que se denominan "Arriba", "Izquierda", "Derecha" y "Inferior" y están unidos al borde superior, inferior, izquierdo o derecho de una vista. Estos márgenes son contenedores en los que se pueden colocar otros márgenes. La interfaz define métodos que devuelven el tamaño del margen y la visibilidad de un margen. Los márgenes son elementos visuales que proporcionan información adicional sobre la vista de texto a la que están asociados. Por ejemplo, el margen de número de línea muestra números de línea para la vista de texto. El margen de glifo muestra elementos de interfaz de usuario.

La [IWpfTextViewMarginProvider](#) interfaz controla la creación y colocación de márgenes. Los márgenes se pueden ordenar con respecto a otros márgenes. Los márgenes de mayor prioridad se encuentran más cerca de la vista de

texto. Por ejemplo, si hay dos márgenes izquierdos, el margen A y el margen B, y el margen B tiene una prioridad menor que el margen A, el margen B aparece a la izquierda del margen A.

El host de vista de texto

La [IWpfTextViewHost](#) interfaz contiene la vista de texto y las decoraciones colindantes que acompañan a la vista, por ejemplo, las barras de desplazamiento. El host de vista de texto también contiene márgenes que se adjuntan a un borde de la vista.

Texto con formato

El texto que se muestra en una [ITextViewLine](#) vista de texto se compone de objetos. Cada línea de vista de texto corresponde a una línea de texto en la vista de texto. Las líneas largas en el búfer de texto subyacente pueden oscurecerse parcialmente (si el ajuste de palabras no está habilitado) o dividirse en varias líneas de vista de texto. La [ITextViewLine](#) interfaz contiene métodos y propiedades para la asignación entre coordenadas y caracteres, y para los adornos que pueden estar asociados a la línea.

[ITextViewLine](#) los objetos se [IFormattedLineSource](#) crean mediante una interfaz. Si solo le preocupa el texto que se muestra actualmente en la vista, puede omitir el origen de formato. Si está interesado en el formato de texto que no se muestra en la vista (por ejemplo, para admitir un corte y pegado de texto enriquecido), puede usarlo para dar [IFormattedLineSource](#) formato al texto en un búfer de texto.

La vista de [ITextViewSnapshotLine](#) texto da formato de uno en uno.

Características del editor

Las características del editor están diseñadas para que la definición de la característica sea independiente de su implementación. El editor incluye estas características:

- Etiquetas y clasificadores
- Adornos
- Proyección
- esquematizar
- Enlaces de ratón y teclas
- Operaciones y primitivos
- IntelliSense

Etiquetas y clasificadores

Las etiquetas son marcadores asociados a un intervalo de texto. Se pueden presentar de diferentes maneras, por ejemplo, mediante el uso de coloración de texto, subrayados, gráficos o ventanas emergentes. Los clasificadores son un tipo de etiqueta.

Otros tipos de [TextMarkerTag](#) etiquetas son [OutliningRegionTag](#) para resaltar texto, para esquematización y [ErrorTag](#) para errores de compilación.

Tipos de clasificación

Una [IClassificationType](#) interfaz representa una clase de equivalencia, que es una categoría abstracta de texto. Los tipos de clasificación pueden heredar varios de otros tipos de clasificación. Por ejemplo, las clasificaciones de lenguaje de programación pueden incluir "palabra clave", "comentario" e "identificador", que heredan de "código". Los tipos de clasificación de lenguaje natural pueden incluir "nombre", "verbo" y "adjetivo", que heredan del "lenguaje natural".

Classifications

Una clasificación es una instancia de un tipo de clasificación determinado, normalmente en un intervalo de texto. A [ClassificationSpan](#) se utiliza para representar una clasificación. Un intervalo de clasificación se puede considerar

como una etiqueta que cubre un intervalo de texto determinado y le dice al sistema que este intervalo de texto es de un tipo de clasificación determinado.

Clasificadores

Un [IClassifier](#) es un mecanismo que divide el texto en un conjunto de clasificaciones. Los clasificadores deben definirse para tipos de contenido específicos y crear instancias para búferes de texto específicos. Los clientes deben implementar [IClassifier](#) para participar en la clasificación de texto.

Agregadores clasificadores

Un agregador de clasificador es un mecanismo que combina todos los clasificadores para un búfer de texto en un solo conjunto de clasificaciones. Por ejemplo, tanto un clasificador de C- como un clasificador de idioma inglés podrían crear clasificaciones sobre un comentario en un archivo de C. Considere este comentario:

```
// This method produces a classifier
```

Un clasificador de C- podría etiquetar todo el intervalo como un comentario, y el clasificador de idioma inglés podría clasificar "produce" como un "verbo" y "método" como un "nombre". El agregador produce un conjunto de clasificaciones no superpuestas y el tipo del conjunto se basa en todas las contribuciones.

Un agregador clasificador también es un clasificador porque divide el texto en un conjunto de clasificaciones. El agregador clasificador también garantiza que no haya clasificaciones superpuestas y que las clasificaciones estén ordenadas. Los clasificadores individuales son libres de devolver cualquier conjunto de clasificaciones, en cualquier orden, y superponerse de cualquier manera.

Formato de clasificación y coloración del texto

El formato de texto es un ejemplo de una característica que se basa en la clasificación de texto. La capa de vista de texto lo utiliza para determinar la visualización del texto en una aplicación. El área de formato de texto depende de WPF/WPF, pero la definición lógica de clasificaciones no lo hace.

Un formato de clasificación es un conjunto de propiedades de formato para un tipo de clasificación específico. Estos formatos heredan del formato del elemento primario del tipo de clasificación.

Un [IClassificationFormatMap](#) es un mapa de un tipo de clasificación a un conjunto de propiedades de formato de texto. La implementación del mapa de formato en el editor controla todas las exportaciones de formatos de clasificación.

Adornos

Los adornos son efectos gráficos que no están directamente relacionados con la fuente y el color de los caracteres de la vista de texto. Por ejemplo, el subrayado de ondulación roja que se utiliza para marcar código no compilado en muchos lenguajes de programación es un adorno incrustado y las descripciones emergentes son adornos emergentes. Los adornos se [UIElement](#) derivan [ITag](#) implementan. Dos tipos especializados de etiqueta [SpaceNegotiatingAdornmentTag](#) de adorno son el , para los adornos [ErrorTag](#) que ocupan el mismo espacio que el texto en una vista, y el , para el subrayado ondulado.

Los adornos incrustados son gráficos que forman parte de la vista de texto con formato. Se organizan en diferentes capas de orden Z. Hay tres capas integradas, como se indica a continuación: texto, el intercalador y la selección. Sin embargo, los desarrolladores pueden definir más capas y ponerlas en orden con respecto a la otra. Los tres tipos de adornos incrustados son adornos relativos al texto (que se mueven cuando se mueve el texto y se eliminan cuando se elimina el texto), adornos relativos a la vista (que tienen que ver con partes que no son de texto de la vista) y adornos controlados por el propietario (el desarrollador debe administrar su ubicación).

Los adornos emergentes son gráficos que aparecen en una pequeña ventana sobre la vista de texto, por ejemplo, información sobre herramientas.

Proyección

La proyección es una técnica para construir un tipo diferente de búfer de texto que en realidad no almacena texto,

sino que combina texto de otros búferes de texto. Por ejemplo, un búfer de proyección se puede utilizar para concatenar el texto de otros dos búferes y presentar el resultado como si estuviera en un solo búfer, o para ocultar partes del texto en un búfer. Un búfer de proyección puede actuar como búfer de origen en otro búfer de proyección. Se puede construir un conjunto de búferes relacionados por proyección para reorganizar el texto de muchas maneras diferentes. (Este conjunto también se conoce como *gráfico de búfer*.) La característica de esquematización de texto de Visual Studio se implementa mediante un búfer de proyección para ocultar el texto contraído y el editor de Visual Studio para ASP.NET páginas usa la proyección para admitir lenguajes incrustados como Visual Basic y C.

Un [IProjectionBuffer](#) se crea [IProjectionBufferFactoryService](#) mediante . Un búfer de proyección se [ITrackingSpan](#) representa mediante una secuencia ordenada de objetos que se conocen como intervalos de *origen*. El contenido de estos intervalos se presenta como una secuencia de caracteres. Los búferes de texto desde los que se extraen los intervalos de origen se denominan búferes de *origen*. Los clientes de un búfer de proyección no tienen que ser conscientes de que difiere de un búfer de texto normal.

El búfer de proyección escucha eventos de cambio de texto en los búferes de origen. Cuando cambia el texto de un intervalo de origen, el búfer de proyección asigna las coordenadas de texto modificadas a sus propias coordenadas y genera los eventos de cambio de texto adecuados. Por ejemplo, considere los búferes de origen A y B que tienen estos contenidos:

```
A: ABCDE
B: vwxyz
```

Si el búfer de proyección P se forma a partir de dos intervalos de texto, uno que tiene todo el búfer A y el otro que tiene todo el búfer B, Entonces P tiene el siguiente contenido:

```
P: ABCDEvwxyz
```

Si la `xy` subcadena se elimina del búfer B, el búfer P genera un evento que indica que se eliminaron los caracteres de las posiciones 7 y 8.

El búfer de proyección también se puede editar directamente. Propaga las ediciones a los búferes de origen adecuados. Por ejemplo, si se inserta una cadena en el búfer P en la posición 6 (la posición original del carácter "v"), la inserción se propaga al búfer B en la posición 1.

Hay restricciones en los intervalos de origen que contribuyen a un búfer de proyección. Los intervalos de origen no pueden superponerse; una ubicación en un búfer de proyección no se puede asignar a más de una ubicación en cualquier búfer de origen y una ubicación en un búfer de origen no se puede asignar a más de una ubicación en un búfer de proyección. No se permiten circularidades en la relación de búfer de origen.

Los eventos se generan cuando cambia el conjunto de búferes de origen para un búfer de proyección y cuando cambia el conjunto de intervalos de origen. Un búfer de elision es un tipo especial de búfer de proyección. Se utiliza principalmente para esquematización y para operaciones que expanden y contraen bloques de texto. Un búfer de elision se basa en un solo búfer de origen y los intervalos en el búfer de elision se deben ordenar de la misma manera que se ordenan en el búfer de origen.

El gráfico de búfer

La [IBufferGraph](#) interfaz permite la asignación a través de un gráfico de búferes de proyección. Todos los búferes de texto y los búferes de proyección se recopilan en un gráfico acíclico dirigido, al igual que el árbol de sintaxis abstracta generado por un compilador de lenguaje. El gráfico se define mediante el búfer superior, que puede ser cualquier búfer de texto. El gráfico de búfer puede asignar desde un punto del búfer superior a un punto de un búfer de origen o de un intervalo en el búfer superior a un conjunto de intervalos en un búfer de origen. De forma similar, puede asignar un punto o un intervalo de un búfer de origen a un punto del búfer superior. Los gráficos de búfer [IBufferGraphFactoryService](#) se crean mediante el archivo .

Eventos y búferes de proyección

Cuando se modifica un búfer de proyección, las modificaciones se envían desde el búfer de proyección a los búferes que dependen de él. Después de modificar todos los búferes, se generan eventos de cambio de búfer, empezando por el búfer más profundo.

esquematar

Esquema es la capacidad de expandir o contraer diferentes bloques de texto en una vista de texto. La esquematización se [ITag](#) define como una especie de , de la misma manera que se definen los adornos. A [OutliningRegionTag](#) es una etiqueta que define una región de texto que se puede expandir o contraer. Para usar la esquematización, debe importar el [IOutliningManagerService](#) para obtener un [IOutliningManager](#) archivo . El administrador de esquematización enumera, contrae y expande los [ICollapsible](#) diferentes bloques, que se representan como objetos, y genera eventos en consecuencia.

Fijaciones de ratón

Los enlaces de ratón vinculan los movimientos del ratón a diferentes comandos. Los enlaces de mouse [IMouseProcessorProvider](#) se definen mediante un [IKeyProcessorProvider](#) archivo , y los enlaces de clave se definen mediante un archivo . La [IWpfTextViewHost](#) instancia automática de todos los enlaces y los conecta a eventos del mouse en la vista.

La [IMouseProcessor](#) interfaz contiene controladores de eventos previos y posteriores al proceso para diferentes eventos del mouse. Para controlar uno de los eventos, puede [MouseProcessorBase](#) invalidar algunos de los métodos de .

Operaciones del editor

Las operaciones del editor se pueden utilizar para automatizar la interacción con el editor, con fines de scripting u otros fines. Puede importar [IEditorOperationsFactoryService](#) las operaciones de [ITextView](#) acceso a un archivo . A continuación, puede utilizar estos objetos para modificar la selección, desplazar la vista o mover el símbolo de intercalación a diferentes partes de la vista.

IntelliSense

IntelliSense admite la finalización de instrucciones, la ayuda de firma (también conocida como información de parámetros), información rápida y bombillas.

La finalización de instrucciones proporciona listas emergentes de posibles terminaciones para nombres de método, elementos XML y otros elementos de codificación o marcado. En general, un gesto de usuario invoca una sesión de finalización. La sesión muestra la lista de posibles terminaciones y el usuario puede seleccionar una o descartar la lista. El [ICompletionBroker](#) es responsable de crear [ICompletionSession](#) y desencadenar el archivo . El [ICompletionSource](#) proceso [CompletionSet](#) calcula los elementos de finalización de la sesión.

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)
- [Importaciones de editores](#)

Managed Extensibility Framework en el editor

06/04/2020 • 5 minutes to read • [Edit Online](#)

El editor se crea mediante componentes de Managed Extensibility Framework (MEF). Puede crear sus propios componentes MEF para ampliar el editor, y el código también puede consumir componentes del editor.

Visión general del marco de extensibilidad administrada

MeF es una biblioteca de .NET que le permite agregar y modificar características de una aplicación o componente que sigue el modelo de programación MEF. El editor de Visual Studio puede proporcionar y consumir componentes MEF.

El MEF se encuentra en el ensamblado *System.ComponentModel.Composition.dll* de .NET Framework versión 4.

Para obtener más información acerca de MEF, vea [Managed Extensibility Framework \(MEF\)](#).

Piezas de componentes y contenedores de composición

Una parte de componente es una clase o un miembro de una clase que puede realizar una (o ambas) de las siguientes acciones:

- Consumir otro componente
- Ser consumido por otro componente

Por ejemplo, considere una aplicación de compras que tiene un componente de entrada de pedido que depende de los datos de disponibilidad del producto proporcionados por un componente de inventario de almacén. En términos MEF, la parte de inventario puede *exportar* datos de disponibilidad de productos y la parte de entrada de pedido puede *importar* los datos. La parte de entrada de pedido y la pieza de inventario no tienen que conocerse entre sí; el contenedor de *composición* (proporcionado por la aplicación anfitriona) es responsable de mantener el conjunto de exportaciones y resolver las exportaciones e importaciones.

El contenedor [CompositionContainer](#) de composición, , suele ser propiedad del host. El contenedor de composición mantiene un *catálogo* de componentes exportados.

Exportar e importar componentes

Puede exportar cualquier funcionalidad, siempre que se implemente como una clase pública o un miembro público de una clase (propiedad o método). No es necesario derivar la [ComposablePart](#) parte del componente de . En su lugar, [ExportAttribute](#) debe agregar un atributo a la clase o miembro de clase que desea exportar. Este atributo especifica el *contrato* por el que otra parte del componente puede importar la funcionalidad.

El contrato de exportación

Define [ExportAttribute](#) la entidad (clase, interfaz o estructura) que se está exportando. Normalmente, el atributo export toma un parámetro que especifica el tipo de la exportación.

```
[Export(typeof(ContentTypeDefinition))]  
class TestContentTypeDefinition : ContentTypeDefinition { }
```

De forma [ExportAttribute](#) predeterminada, el atributo define un contrato que es el tipo de la clase de exportación.

```
[Export]
[Name("Structure")]
[Order(After = "Selection", Before = "Text")]
class TestAdornmentLayerDefinition : AdornmentLayerDefinition { }
```

En el ejemplo, `[Export]` el atributo `[Export(typeof(TestAdornmentLayerDefinition))]` predeterminado es equivalente a .

También puede exportar una propiedad o un método, como se muestra en el ejemplo siguiente.

```
[Export]
[Name("Scarlet")]
[Order(After = "Selection", Before = "Text")]
public AdornmentLayerDefinition scarletLayerDefinition;
```

Importar una exportación MEF

Cuando desee consumir una exportación MEF, debe conocer el contrato (normalmente el [ImportAttribute](#) tipo) por el que se exportó y agregar un atributo que tenga ese valor. De forma predeterminada, el atributo `import` toma un parámetro, que es el tipo de la clase que modifica. Las siguientes líneas de [IClassificationTypeRegistryService](#) código importan el tipo.

```
[Import]
internal IClassificationTypeRegistryService ClassificationRegistry;
```

Obtenga la funcionalidad del editor a partir de una pieza de componente MEF

Si el código existente es una pieza de componente MEF, puede utilizar metadatos MEF para consumir partes de componentes del editor.

Para consumir la funcionalidad del editor desde una pieza de componente MEF

1. Agregue referencias a *System.Composition.ComponentModel.dll*, que se encuentra en la caché global de ensamblados (GAC) y a los ensamblados del editor.
2. Agregue las directivas `using` relevantes.

```
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text;
```

3. Agregue `[Import]` el atributo a la interfaz de servicio, como se indica a continuación.

```
[Import]
ITextBufferFactoryService textBufferService;
```

4. Cuando haya obtenido el servicio, puede consumir cualquiera de sus componentes.
5. Cuando haya compilado el ensamblado, colóquelo en **.. Carpeta de componentes** de la instalación de Visual Studio.

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Servicio de lenguaje y puntos de extensión del editor

06/04/2020 • 27 minutes to read • [Edit Online](#)

El editor proporciona puntos de extensión que puede ampliar como componentes de Managed Extensibility Framework (MEF), incluidas la mayoría de las características del servicio de lenguaje. Estas son las principales categorías de puntos de extensión:

- Tipos de contenido
- Tipos de clasificación y formatos de clasificación
- Márgenes y barras de desplazamiento
- Etiquetas
- Adornos
- Procesadores de ratón
- Manipuladores de caídas
- Opciones
- IntelliSense

Ampliar los tipos de contenido

Los tipos de contenido son las definiciones de los tipos de texto controlados por el editor, por ejemplo, "texto", "código" o "CSharp". Para definir un nuevo tipo de contenido, [ContentTypeDefinition](#) declare una variable del tipo y asigne un nombre único al nuevo tipo. Para registrar el tipo de contenido con el editor, expórtelo junto con los siguientes atributos:

- [NameAttribute](#) es el nombre del tipo de contenido.
- [BaseDefinitionAttribute](#) es el nombre del tipo de contenido del que se deriva este tipo de contenido. Un tipo de contenido puede heredar de varios otros tipos de contenido.

Dado [ContentTypeDefinition](#) que la clase está sellada, puede exportarla sin ningún parámetro de tipo.

En el ejemplo siguiente se muestran los atributos de exportación en una definición de tipo de contenido.

```
[Export]
[Name("test")]
[BaseDefinition("code")]
[BaseDefinition("projection")]
internal static ContentTypeDefinition TestContentTypeDefinition;
```

Los tipos de contenido se pueden basar en cero o más tipos de contenido preexistentes. Estos son los tipos integrados:

- Cualquiera: el tipo de contenido básico. Padre de todos los demás tipos de contenido.
- Texto: el tipo básico para el contenido sin proyección. Hereda de "cualquiera".

- Texto sin formato: para texto sin código. Hereda de "texto".
- Código: para código de todo tipo. Hereda de "texto".
- Inerte: excluye el texto de cualquier tipo de manipulación. El texto de este tipo de contenido nunca tendrá ninguna extensión aplicada.
- Proyección: para el contenido de los búferes de proyección. Hereda de "cualquiera".
- Intellisense: para el contenido de IntelliSense. Hereda de "texto".
- Sighelp: ayuda de firma. Hereda de "intellisense".
- Sighelp-doc: documentación de ayuda de firma. Hereda de "intellisense".

Estos son algunos de los tipos de contenido definidos por Visual Studio y algunos de los lenguajes que se hospedan en Visual Studio:

- Básico
- C/C++
- ConsoleOutput
- CSharp
- CSS
- Enc
- FindResults
- F#
- HTML
- JScript
- XAML
- XML

Para detectar la lista de tipos [IContentTypeRegistryService](#) de contenido disponibles, importe el archivo , que mantiene la colección de tipos de contenido para el editor. El código siguiente importa este servicio como una propiedad.

```
[Import]
internal IContentTypeRegistryService ContentTypeRegistryService { get; set; }
```

Para asociar un tipo de contenido [FileExtensionToContentTypeDefinition](#) a una extensión de nombre de archivo, utilice .

NOTE

En Visual Studio, las extensiones [ProvideLanguageExtensionAttribute](#) de nombre de archivo se registran mediante el uso de un paquete de servicio de lenguaje. El [FileExtensionToContentTypeDefinition](#) asocia un tipo de contenido MEF con una extensión de nombre de archivo que se ha registrado de esta manera.

Para exportar la extensión de nombre de archivo a la definición de tipo de contenido, debe incluir los siguientes atributos:

- [FileExtensionAttribute](#): especifica la extensión de nombre de archivo.
- [ContentTypeAttribute](#): especifica el tipo de contenido.

Dado [FileExtensionToContentTypeDefinition](#) que la clase está sellada, puede exportarla sin ningún parámetro de tipo.

En el ejemplo siguiente se muestran los atributos de exportación de una extensión de nombre de archivo a una definición de tipo de contenido.

```
[Export]
[FileExtension(".test")]
[ContentType("test")]
internal static FileExtensionToContentTypeDefinition TestFileExtensionDefinition;
```

El [IFileExtensionRegistryService](#) administra las asociaciones entre las extensiones de nombre de archivo y los tipos de contenido.

Ampliar los tipos de clasificación y los formatos de clasificación

Puede utilizar tipos de clasificación para definir los tipos de texto para los que desea proporcionar un control diferente (por ejemplo, colorear el texto "palabra clave" en azul y el texto "comentario" verde). Defina un nuevo tipo de clasificación [ClassificationTypeDefinition](#) declarando una variable de tipo y dándole un nombre único.

Para registrar el tipo de clasificación con el editor, expórtelo junto con los siguientes atributos:

- [NameAttribute](#): el nombre del tipo de clasificación.
- [BaseDefinitionAttribute](#): el nombre del tipo de clasificación del que hereda este tipo de clasificación. Todos los tipos de clasificación heredan de "texto" y un tipo de clasificación puede heredar de varios otros tipos de clasificación.

Dado [ClassificationTypeDefinition](#) que la clase está sellada, puede exportarla sin ningún parámetro de tipo.

En el ejemplo siguiente se muestran los atributos de exportación en una definición de tipo de clasificación.

```
[Export]
[Name("csharp.test")]
[BaseDefinition("test")]
internal static ClassificationTypeDefinition CSharpTestDefinition;
```

El [IStandardClassificationService](#) proporciona acceso a clasificaciones estándar. Los tipos de clasificación integrados incluyen los siguientes:

- "text"
- "lenguaje natural" (deriva de "texto")
- "lenguaje formal" (deriva de "texto")
- "string" (deriva de "literal")
- "carácter" (deriva de "literal")
- "numérico" (deriva de "literal")

Un conjunto de tipos [ErrorTypeDefinition](#) de error diferentes heredan de . Incluyen los siguientes tipos de

error:

- "error de sintaxis"
- "error del compilador"
- "otro error"
- "Advertencia"

Para detectar la lista de tipos [IClassificationTypeRegistryService](#) de clasificación disponibles, importe el archivo , que mantiene la colección de tipos de clasificación para el editor. El código siguiente importa este servicio como una propiedad.

```
[Import]
internal IClassificationTypeRegistryService ClassificationTypeRegistryService { get; set; }
```

Puede definir una definición de formato de clasificación para el nuevo tipo de clasificación. Derive una [ClassificationFormatDefinition](#) clase de y [EditorFormatDefinition](#) expórtela con el tipo, junto con los siguientes atributos:

- [NameAttribute](#): el nombre del formato.
- [DisplayNameAttribute](#): el nombre para mostrar del formato.
- [UserVisibleAttribute](#): especifica si el formato aparece en la página **Fuentes y colores** del cuadro de diálogo **Opciones**.
- [OrderAttribute](#): la prioridad del formato. Los valores [Priority](#) válidos son de .
- [ClassificationTypeAttribute](#): el nombre del tipo de clasificación al que se asigna este formato.

En el ejemplo siguiente se muestran los atributos de exportación en una definición de formato de clasificación.

```
[Export(typeof(EditorFormatDefinition))]
[ClassificationType(ClassificationTypeNames = "test")]
[Name("test")]
[DisplayName("Test")]
[UserVisible(true)]
[Order(After = Priority.Default, Before = Priority.High)]
internal sealed class TestFormat : ClassificationFormatDefinition
```

Para descubrir la lista de [IEditorFormatMapService](#) formatos disponibles, importe el archivo , que mantiene la colección de formatos para el editor. El código siguiente importa este servicio como una propiedad.

```
[Import]
internal IEditorFormatMapService FormatMapService { get; set; }
```

Extender márgenes y barras de desplazamiento

Los márgenes y las barras de desplazamiento son los elementos principales de la vista del editor, además de la propia vista de texto. Puede proporcionar cualquier número de márgenes además de los márgenes estándar que aparecen alrededor de la vista de texto.

Implemente [IWpfTextViewMargin](#) una interfaz para definir un margen. También debe implementar [IWpfTextViewMarginProvider](#) la interfaz para crear el margen.

Para registrar el proveedor de márgenes en el editor, debe exportar el proveedor junto con los siguientes atributos:

- **NameAttribute**: el nombre del margen.
- **OrderAttribute**: el orden en el que aparece el margen, en relación con los demás márgenes.

Estos son los márgenes incorporados:

- "Barra de desplazamiento horizontal wpf"
- "Barra de desplazamiento vertical wpf"
- "Margen de número de línea wpf"

Los márgenes horizontales que `After="Wpf Horizontal Scrollbar"` tienen un atributo de orden se muestran debajo del `Before = "Wpf Horizontal Scrollbar"` margen integrado y los márgenes horizontales que tienen un atributo de orden se muestran por encima del margen integrado. Los márgenes verticales derecho `After="Wpf Vertical Scrollbar"` que tienen un atributo order de se muestran a la derecha de la barra de desplazamiento. Los márgenes verticales izquierdos `After="Wpf Line Number Margin"` que tienen un atributo de orden aparecen a la izquierda del margen del número de línea (si está visible).

- **MarginContainerAttribute**: el tipo de margen (izquierda, derecha, superior o inferior).
- **ContentTypeAttribute**: el tipo de contenido (por ejemplo, "texto" o "código") para el que su margen es válido.

En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de margen para un margen que aparece a la derecha del margen del número de línea.

```
[Export(typeof(IWpfTextViewMarginProvider))]  
[Name("TestMargin")]  
[Order(Before = "Wpf Line Number Margin")]  
[MarginContainer(PredefinedMarginNames.Left)]  
[ContentType("text")]
```

Extender etiquetas

Las etiquetas son una forma de asociar datos con diferentes tipos de texto. En muchos casos, los datos asociados se muestran como un efecto visual, pero no todas las etiquetas tienen una presentación visual. Puede definir su propio tipo **ITag** de etiqueta implementando . También debe **ITagger<T>** implementar para proporcionar las etiquetas para un conjunto **ITaggerProvider** determinado de intervalos de texto y un para proporcionar el etiquetador. Debe exportar el proveedor de etiquetas junto con los siguientes atributos:

- **ContentTypeAttribute**: el tipo de contenido (por ejemplo, "texto" o "código") para el que su etiqueta es válida.
- **TagTypeAttribute**: el tipo de etiqueta.

En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de etiqueta.

<CodeContentPlaceholder> 8 Los siguientes tipos de etiqueta están integrados:

- **ClassificationTag**: asociado **IClassificationType** a un archivo .
- **ErrorTag**: asociado con tipos de error.
- **TextMarkerTag**: asociado con un adorno.

NOTE

Para obtener un [TextMarkerTag](#) ejemplo de un , vea la definición [HighlightWordTag](#) en [Tutorial: Resaltar texto](#).

- [OutliningRegionTag](#): asociado a regiones que se pueden expandir o contraer en la esquematización.
- [SpaceNegotiatingAdornmentTag](#): define el espacio que ocupa un adorno en una vista de texto. Para obtener más información acerca de los adornos de negociación de espacio, consulte la sección siguiente.
- [IntraTextAdornmentTag](#): proporciona espaciado y tamaño automáticos para el adorno.

Para buscar y utilizar etiquetas para búferes [IBufferTagAggregatorFactoryService](#) y vistas, [ITagAggregator<T>](#) importe el [IViewTagAggregatorFactoryService](#) , que le proporciona un tipo solicitado. El código siguiente importa este servicio como una propiedad.

```
[Import]
internal IViewTagAggregatorFactoryService ViewTagAggregatorFactoryService { get; set; }
```

Etiquetas y MarkerFormatDefinitions

Puede ampliar [MarkerFormatDefinition](#) la clase para definir la apariencia de una etiqueta. Debe exportar la clase [EditorFormatDefinition](#)(como a) con los siguientes atributos:

- [NameAttribute](#): el nombre utilizado para hacer referencia a este formato
- [UserVisibleAttribute](#): esto hace que el formato aparezca en la interfaz de usuario

En el constructor, se define el nombre para mostrar y la apariencia de la etiqueta. [BackgroundColor](#) define el color [ForegroundColor](#) de relleno y define el color del borde. Es [DisplayName](#) el nombre localizable de la definición de formato.

A continuación se muestra un ejemplo de una definición de formato:

```
[Export(typeof(EditorFormatDefinition))]
[Name("MarkerFormatDefinition/HighlightWordFormatDefinition")]
[UserVisible(true)]
internal class HighlightWordFormatDefinition : MarkerFormatDefinition
{
    public HighlightWordFormatDefinition()
    {
        this.BackgroundColor = Colors.LightBlue;
        this.ForegroundColor = Colors.DarkBlue;
        this.DisplayName = "Highlight Word";
        this.ZOrder = 5;
    }
}
```

Para aplicar esta definición de formato a una etiqueta, haga referencia al nombre establecido en el atributo name de la clase (no al nombre para mostrar).

NOTE

Para obtener un [MarkerFormatDefinition](#) ejemplo de un , vea la clase [HighlightWordFormatDefinition](#) en [Tutorial: Resaltar texto](#).

Extender adornos

Los adornos definen efectos visuales que se pueden agregar al texto que se muestra en una vista de texto o a la propia vista de texto. Puede definir su propio adorno [UIElement](#) como cualquier tipo de archivo.

En la clase de adorno, debe declarar un [AdornmentLayerDefinition](#) archivo. Para registrar la capa de adorno, expórtela junto con los siguientes atributos:

- [NameAttribute](#): el nombre del adorno.
- [OrderAttribute](#): el orden del adorno con respecto a otras capas de adorno. La [PredefinedAdornmentLayers](#) clase define cuatro capas predeterminadas: Selección, Esquema, Intercalación y Texto.

En el ejemplo siguiente se muestran los atributos de exportación en una definición de capa de adorno.

```
[Export]
[Name("TestEmbeddedAdornment")]
[Order(After = PredefinedAdornmentLayers.Selection, Before = PredefinedAdornmentLayers.Text)]
internal AdornmentLayerDefinition testLayerDefinition;
```

Debe crear una segunda clase [IWpfTextViewCreationListener](#) que [TextViewCreated](#) implemente y controle su evento creando instancias del adorno. Debe exportar esta clase junto con los siguientes atributos:

- [ContentTypeAttribute](#): el tipo de contenido (por ejemplo, "texto" o "código") para el que el adorno es válido.
- [TextViewRoleAttribute](#): el tipo de vista de texto para la que este adorno es válido. La [PredefinedTextViewRoles](#) clase tiene el conjunto de roles de vista de texto predefinidos. Por ejemplo, [Document](#) se utiliza principalmente para vistas de texto de archivos. [Interactive](#) se utiliza para las vistas de texto que un usuario puede editar o navegar mediante un ratón y un teclado. Ejemplos [Interactive](#) de vistas son la vista de texto del editor y la ventana **Salida**.

En el ejemplo siguiente se muestran los atributos de exportación en el proveedor de adornos.

```
[Export(typeof(IWpfTextViewCreationListener))]
[ContentType("csharp")]
[TextViewRole(PredefinedTextViewRoles.Document)]
internal sealed class TestAdornmentProvider : IWpfTextViewCreationListener
```

Un adorno de negociación de espacio es aquel que ocupa el espacio al mismo nivel que el texto. Para crear este tipo de adorno, debe definir [SpaceNegotiatingAdornmentTag](#) una clase de etiqueta que herede de [AdornmentTag](#), que define la cantidad de espacio que ocupa el adorno.

Al igual que con todos los adornos, debe exportar la definición de capa de adorno.

```
[Export]
[Name("TestAdornment")]
[Order(After = DefaultAdornmentLayers.Text)]
internal AdornmentLayerDefinition testAdornmentLayer;
```

Para crear una instancia del adorno de negociación [ITaggerProvider](#) de espacio, debe crear [IWpfTextViewCreationListener](#) una clase que implemente, además de la clase que implementa (como con otros tipos de adornos).

Para registrar el proveedor de etiquetas, debe exportarlo junto con los siguientes atributos:

- [ContentTypeAttribute](#): el tipo de contenido (por ejemplo, "texto" o "código") para el que su adorno es válido.

- [TextViewRoleAttribute](#): el tipo de vista de texto para la que esta etiqueta o adorno es válido. La [PredefinedTextViewRoles](#) clase tiene el conjunto de roles de vista de texto predefinidos. Por ejemplo, [Document](#) se utiliza principalmente para vistas de texto de archivos. [Interactive](#) se utiliza para las vistas de texto que un usuario puede editar o navegar mediante un ratón y un teclado. Ejemplos [Interactive](#) de vistas son la vista de texto del editor y la ventana [Salida](#).
- [TagTypeAttribute](#): el tipo de etiqueta o adorno que ha definido. Debe agregar un [TagTypeAttribute](#) [SpaceNegotiatingAdornmentTag](#)segundo para .

En el ejemplo siguiente se muestran los atributos de exportación en el proveedor de etiqueta para una etiqueta de adorno de negociación de espacio.

```
[Export(typeof(ITaggerProvider))]
[ContentType("text")]
[TextViewRole(PredefinedTextViewRoles.Document)]
[TagType(typeof(SpaceNegotiatingAdornmentTag))]
[TagType(typeof(TestSpaceNegotiatingTag))]
internal sealed class TestTaggerProvider : ITaggerProvider
```

Ampliación de los procesadores de ratón

Puede agregar un control especial para la entrada del ratón. Cree una clase que [MouseProcessorBase](#) herede y reemplace los eventos del mouse para la entrada que desea controlar. También debe [IMouseProcessorProvider](#) implementar en una segunda clase [ContentTypeAttribute](#) y exportarlo junto con el que especifica el tipo de contenido (por ejemplo, "texto" o "código") para el que el controlador del mouse es válido.

En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de procesador de mouse.

```
[Export(typeof(IMouseProcessorProvider))]
[Name("test mouse processor")]
[ContentType("text")]
[TextViewRole(PredefinedTextViewRoles.Interactive)]
internal sealed class TestMouseProcessorProvider : IMouseProcessorProvider
```

Extender los controladores de caída

Puede personalizar el comportamiento de los controladores de colocación para [IDropHandler](#) tipos específicos de [IDropHandlerProvider](#) texto mediante la creación de una clase que implementa y una segunda clase que implementa para crear el controlador de colocación. Debe exportar el controlador de colocación junto con los siguientes atributos:

- [DropFormatAttribute](#): el formato de texto para el que este controlador de colocación es válido. Los siguientes formatos se gestionan en orden de prioridad de mayor a menor:
 1. Cualquier formato personalizado
 2. FileDrop
 3. EnhancedMetafile
 4. WaveAudio
 5. Riff
 6. Dif
 7. Configuración regional

8. Paleta
9. PenData
10. Serializable
11. SymbolicLink
12. Xaml
13. XamlPackage
14. Tiff
15. Bitmap
16. Dib
17. MetafilePicture
18. CSV
19. System.String
20. Formato HTML
21. UnicodeText
22. OEMText
23. Texto

- [NameAttribute](#): el nombre del controlador de colocación.
- [OrderAttribute](#): el orden del controlador de colocación antes o después del controlador de colocación predeterminado. El controlador de colocación predeterminado para Visual Studio se denomina "DefaultFileDropHandler".

En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de controlador de colocación.

```
[Export(typeof(IDropHandlerProvider))]  
[DropFormat("Text")]  
[Name("TestDropHandler")]  
[Order(Before="DefaultFileDropHandler")]  
internal class TestDropHandlerProvider : IDropHandlerProvider
```

Ampliación de las opciones del editor

Puede definir opciones para que sean válidas solo en un ámbito determinado, por ejemplo, en una vista de texto. El editor proporciona este conjunto de opciones predefinidas: opciones de editor, opciones de vista y opciones de vista de Windows Presentation Foundation (WPF). Estas opciones se [DefaultOptions](#) pueden encontrar [DefaultTextViewOptions](#) en , , y .

Para agregar una nueva opción, derive una clase de una de estas clases de definición de opciones:

- [EditorOptionDefinition<T>](#)
- [ViewOptionDefinition<T>](#)
- [WpfViewOptionDefinition<T>](#)

En el ejemplo siguiente se muestra cómo exportar una definición de opción que tiene un valor booleano.

```
[Export(typeof(EditorOptionDefinition))]  
internal sealed class TestOption : EditorOptionDefinition<bool>
```

Ampliar IntelliSense

IntelliSense es un término general para un grupo de características que proporcionan información sobre el texto estructurado y la finalización de instrucciones para él. Estas características incluyen la finalización de instrucciones, la ayuda de firma, la información rápida y las bombillas. La finalización de instrucciones ayuda a los usuarios a escribir correctamente una palabra clave de idioma o un nombre de miembro. La ayuda de firma muestra la firma o las firmas del método que el usuario acaba de escribir. Información rápida muestra una firma completa para un tipo o nombre de miembro cuando el mouse se apoya en él. La bombilla proporciona acciones adicionales para determinados identificadores en determinados contextos, por ejemplo, cambiar el nombre de todas las apariciones de una variable después de cambiar el nombre de una aparición.

El diseño de una característica de IntelliSense es muy similar en todos los casos:

- Un *intermediario* de IntelliSense es responsable del proceso general.
- Una *sesión* de IntelliSense representa la secuencia de eventos entre la activación del presentador y la confirmación o cancelación de la selección. La sesión se desencadena normalmente por algún gesto del usuario.
- Un *controlador* IntelliSense es responsable de decidir cuándo debe iniciarse y finalizar la sesión. También decide cuándo se debe confirmar la información y cuándo se debe cancelar la sesión.
- Un *origen* de IntelliSense proporciona el contenido y decide la mejor coincidencia.
- Un *presentador* de IntelliSense es responsable de mostrar el contenido.

En la mayoría de los casos, se recomienda proporcionar al menos un origen y un controlador. También puede proporcionar un presentador si desea personalizar la pantalla.

Implementar un origen de IntelliSense

Para personalizar un origen, debe implementar una (o más) de las siguientes interfaces de origen:

- [ICompletionSource](#)
- [IQuickInfoSource](#)
- [ISignatureHelpSource](#)
- [ISuggestedActionsSource](#)

IMPORTANT

[ISmartTagSource](#) ha quedado en desuso a favor de [ISuggestedActionsSource](#).

Además, debe implementar un proveedor del mismo tipo:

- [ICompletionSourceProvider](#)
- [IQuickInfoSourceProvider](#)
- [ISignatureHelpSourceProvider](#)
- [ISuggestedActionsSourceProvider](#)

IMPORTANT

[ISmartTagSourceProvider](#) ha quedado en desuso a favor de [ISuggestedActionsSourceProvider](#).

Debe exportar el proveedor junto con los siguientes atributos:

- [NameAttribute](#): el nombre de la fuente.
- [ContentTypeAttribute](#): el tipo de contenido (por ejemplo, "texto" o "código") al que se aplica la fuente.
- [OrderAttribute](#): el orden en el que debe aparecer la fuente (con respecto a otras fuentes).
- En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de origen de finalización.

```
Export(typeof(ICompletionSourceProvider))
[Name(" Test Statement Completion Provider")]
[Order(Before = "default")]
[ContentType("text")]
internal class TestCompletionSourceProvider : ICompletionSourceProvider
```

Para obtener más información acerca de la implementación de orígenes de IntelliSense, consulte los siguientes tutoriales:

- [Tutorial: Mostrar información sobre herramientas quickInfo](#)
- [Tutorial: Mostrar la Ayuda de firma](#)
- [Tutorial: Mostrar la finalización de instrucciones](#)

Implementar un controlador IntelliSense

Para personalizar un controlador, [IIntellisenseController](#) debe implementar la interfaz. Además, debe implementar un proveedor de controlador junto con los siguientes atributos:

- [NameAttribute](#): el nombre del controlador.
- [ContentTypeAttribute](#): el tipo de contenido (por ejemplo, "texto" o "código") al que se aplica el controlador.
- [OrderAttribute](#): el orden en el que debe aparecer el controlador (con respecto a otros controladores).

En el ejemplo siguiente se muestran los atributos de exportación en un proveedor de controlador de finalización.

```
Export(typeof(IIntellisenseControllerProvider))
[Name(" Test Controller Provider")]
[Order(Before = "default")]
[ContentType("text")]
internal class TestIntellisenseControllerProvider : IIntellisenseControllerProvider
```

Para obtener más información acerca del uso de controladores IntelliSense, consulte los siguientes tutoriales:

- [Tutorial: Mostrar información sobre herramientas quickInfo](#)

Importaciones de editores

06/04/2020 • 4 minutes to read • [Edit Online](#)

Puede importar una serie de servicios de editor, fábricas y intermediarios que proporcionan a su extensión diferentes tipos de acceso al editor principal. Por ejemplo, puede [ITextStructureNavigatorSelectorService](#) importar el para [ITextStructureNavigator](#) proporcionarle un para un tipo de contenido determinado. (Este navegador le permite realizar diferentes tipos de búsquedas en un búfer de texto.)

Para usar una importación de editor, impórtela como un campo o propiedad de una clase que exporta un componente de Managed Extensibility Framework.

NOTE

Para obtener más información acerca de Managed Extensibility Framework, vea [Managed Extensibility Framework \(MEF\)](#).

Sintaxis de importación

En el ejemplo siguiente se muestra cómo importar el servicio de fábrica de opciones del editor.

```
[Import]
internal IEditorOptionsFactoryService EditorOptions { get; set; }
```

Si desea importar el servicio como un campo y no `null` como una propiedad, debe establecerlo en la declaración para evitar las advertencias del compilador sobre no asignar a una variable:

```
[Import]
internal IEditorOptionsFactoryService m_editorOptions = null;
```

Para obtener más ejemplos de uso de importaciones, consulte los siguientes tutoriales:

- [Tutorial: Crear un glifo de margen](#)
- [Tutorial: Personalizar la vista de texto](#)
- [Tutorial: Resaltar texto](#)
- [Tutorial: Mostrar información sobre herramientas quickInfo](#)
- [Tutorial: Mostrar la Ayuda de firma](#)
- [Tutorial: Mostrar la finalización de instrucciones](#)
- [Tutorial: Mostrar sugerencias de bombillas](#)

Importar el proveedor de servicios

También puede importar [SVsServiceProvider](#) un (que se encuentra en el ensamblado `Microsoft.VisualStudio.Shell.Immutable.10.0`) de la misma manera para obtener acceso a los servicios de Visual Studio:

```
[Import]
internal SVsServiceProvider ServiceProvider = null;
```

Consulte [Tutorial: Acceda al objeto DTE desde una extensión](#) de editor para obtener más información.

Servicios

Los servicios de editor son generalmente entidades únicas que proporcionan un servicio y se comparten entre varios componentes.

IMPORTAR	PROPORCIONA
IFileExtensionRegistryService	La relación entre IContentType las extensiones de archivo y los objetos.
IContentTypeRegistryService	La colección de objetos IContentType .
IVsFontsAndColorsInformationService	IVsFontsAndColorsInformation Objetos.
IVsEditorAdaptersFactoryService	Muchos objetos de adaptador de editor: IVsCodeWindow IVsTextBuffer IVsTextBufferCoordinator IVsTextView
IIncrementalSearchFactoryService	Un IIncrementalSearch objeto para una vista de texto determinada.
ITextBufferFactoryService	Un valor de tipo ITextBuffer .
ITextDocumentFactoryService	Un valor de tipo ITextDocument .
IDifferenceService	Una IDifferenceCollection<T> de las diferencias.
IHierarchicalStringDifferenceService	Una IHierarchicalDifferenceCollection de las diferencias.
IProjectionBufferFactoryService	Un IProjectionBuffer o IElisionBuffer un archivo .
IBufferGraphFactoryService	Un IBufferGraph para un ITextBuffer conjunto de objetos.
IClassifierAggregatorService	Un IClassifier for ITextBuffer a .
IViewClassifierAggregatorService	Un IClassifier for ITextView a .
IClassificationFormatMapService	Un IClassificationFormatMap for ITextView a .
IEditorFormatMapService	Un IEditorFormatMap for ITextView a .
IClassificationTypeRegistryService	Mantiene la colección de IClassificationType objetos.

IMPORTAR	PROPORCIONA
IBufferTagAggregatorFactoryService	Un ITagAggregator<T> para un búfer de texto.
IViewTagAggregatorFactoryService	Un ITagAggregator<T> para una vista de texto.
IEditorOptionsFactoryService	El IEditorOptions para el ámbito especificado.
IScrollMapFactoryService	Un IScrollMap para una vista de texto.
ISmartIndentationService	Un ISmartIndent for ITextView .
ISmartIndentationService	Obtiene la sangría automática ISmartIndentProvider a través de los objetos.
ITextEditorFactoryService	Administra IWpfTextViewHost el IWpfTextView for a .
IFormattedTextSourceFactoryService	Un valor de tipo IFormattedLineSource .
IRtfBuilderService	Genera texto con formato RTF a partir de un conjunto de intervalos de instantáneas.
ITextAndAdornmentSequencerFactoryService	Un ITextAndAdornmentSequencer for ITextView an .
ITextParagraphPropertiesFactoryService	A TextParagraphProperties para dar formato a las líneas de texto en una vista.
IEditorOperationsFactoryService	Un IEditorOperations objeto ITextView para un archivo .
ITextSearchService	Busca una instantánea de texto.
ITextStructureNavigatorSelectorService	Un ITextStructureNavigator for ITextBuffer IContentType ean by .
IOutliningManagerService	Un IOutliningManager para una vista de texto.
IGlyphService	Un conjunto estándar de glifos.
IIntellisenseSessionStackMapService	Un IIntellisenseSessionStack for ITextView a .
IWpfKeyboardTrackingService	Realiza un seguimiento del manejo del teclado.
IStandardClassificationService	Objetos estándar. IClassificationType
ITextUndoHistoryRegistry	Mantiene la relación entre los ITextUndoHistory búferes de texto y los objetos.

Otras importaciones

Las fábricas y los intermediarios de proveedores son generalmente entidades que pueden tener varias instancias en varios componentes.

IMPORTAR	PROPORCIONA
IErrorProviderFactory	A SimpleTagger<T> de ErrorTag tipo) para el búfer especificado.
ITextMarkerProviderFactory	Un marcador de texto SimpleTagger<T> (a de tipo TextMarkerTag).
IToolTipProviderFactory	Un IToolTipProvider para ITextView un determinado .
ICompletionBroker	Un valor de tipo ICompletionSession .
IQuickInfoBroker	Un valor de tipo IQuickInfoSession .
ISignatureHelpBroker	Un valor de tipo ISignatureHelpSession .

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Compatibilidad con EditorConfig para su servicio de lenguaje

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los archivos [EditorConfig](#) le permiten describir opciones comunes del editor de texto, como el tamaño de sangría, por proyecto. Para obtener más información sobre la compatibilidad de Visual Studio con los archivos EditorConfig, vea [Crear configuración de editor portátil mediante EditorConfig](#).

En la mayoría de los casos, cuando se implementa un servicio de lenguaje de Visual Studio, no es necesario llevar a cabo ninguna tarea adicional para admitir las propiedades universales de EditorConfig. El editor principal detecta automáticamente y lee el archivo .editorconfig cuando los usuarios abren archivos, y establece el búfer de texto y las opciones de visualización adecuados. Sin embargo, para ediciones como pestañas y espacios, algunos servicios de idioma optan por usar una opción de vista de texto contextual adecuada en lugar de usar la configuración global. En estos casos, el servicio de lenguaje debe actualizarse para que admita archivos EditorConfig.

A continuación se muestran los cambios necesarios para actualizar un servicio de lenguaje para admitir archivos EditorConfig, reemplazando una opción *global específica del idioma* por una opción *contextual*:

Estilo de sangría

OPCIONES ESPECÍFICAS DEL IDIOMA	OPCIONES CONTEXTUALES
Microsoft.VisualStudio.TextManager.Interop.LANGPREFERENCE S.InsertTabs Microsoft.VisualStudio.Package.LanguagePreferences.InsertTab s	!textBufferOptions.GetOptionValue(DefaultOptions.ConvertTab sToSpacesOptionId) !textView.Options.GetOptionValue(DefaultOptions.ConvertTab sToSpacesOptionId)

Tamaño de sangría

OPCIONES ESPECÍFICAS DEL IDIOMA	OPCIONES CONTEXTUALES
Microsoft.VisualStudio.TextManager.Interop.LANGPREFERENCE S.uIndentSize Microsoft.VisualStudio.Package.LanguagePreferences.InsertTab s.IndentSize	textBufferOptions.GetOptionValue(DefaultOptions.IndentSizeO ptionId) textView.Options.GetOptionValue(DefaultOptions.IndentSizeO ptionId)

Tamaño de tabulación

OPCIONES ESPECÍFICAS DEL IDIOMA	OPCIONES CONTEXTUALES
Microsoft.VisualStudio.TextManager.Interop.LANGPREFERENCE S.uTabSize Microsoft.VisualStudio.Package.LanguagePreferences.InsertTab s.TabSize	textBufferOptions.GetOptionValue(DefaultOptions.TabSizeOpti onId) textView.Options.GetOptionValue(DefaultOptions.TabSizeOptio nId)

Vea también

- [Crear ajustes de editor portátiles con EditorConfig](#)
- [Ampliación del editor y los servicios de idiomas](#)

Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede definir su propio tipo de contenido y vincular una extensión de nombre de archivo mediante las extensiones de Managed Extensibility Framework (MEF) del editor. En algunos casos, la extensión de nombre de archivo ya está definida por un servicio de lenguaje. Pero, para usarlo con MEF, debe vincularlo a un tipo de contenido.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `ContentTypeTest` nombre a la solución .
2. En el archivo `source.extension.vsixmanifest`, vaya a la pestaña **Assets** y establezca el campo **Type** en **Microsoft.VisualStudio.MefComponent**, el campo **Source** en **A project in the solution** actualy el campo **Project** en el nombre del proyecto.

Definir el tipo de contenido

1. Agregue un archivo de clase y asígnele el nombre `FileAndContentTypes` .
2. Agregue referencias a los siguientes ensamblados:
 - a. `System.ComponentModel.Composition`
 - b. `Microsoft.VisualStudio.Text.Classification`
 - c. `Microsoft.VisualStudio.CoreUtility`
3. Agregue las `using` siguientes directivas.

```
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text.Classification;
using Microsoft.VisualStudio.Utilities;
```

4. Declare una clase estática que contenga las definiciones.

```
internal static class FileAndContentTypeDefinitions
{
    . . .
}
```

5. En esta clase, `ContentTypeDefinition` exporte un "hid" con nombre y declare su definición base como "text".

```
internal static class FileAndContentTypeDefinitions
{
    [Export]
    [Name("hid")]
    [BaseDefinition("text")]
    internal static ContentTypeDefinition hidingContentTypeDefinition;
}
```

Vincular una extensión de nombre de archivo a un tipo de contenido

- Para asignar este tipo de contenido a [FileExtensionToContentTypeDefinition](#) una extensión de nombre de archivo, exporte a que tenga la extensión *.hid* y el tipo de contenido "hid".

```
internal static class FileAndContentTypeDefinitions
{
    [Export]
    [Name("hid")]
    [BaseDefinition("text")]
    internal static ContentTypeDefinition hidingContentTypeDefinition;

    [Export]
    [FileExtension(".hid")]
    [ContentType("hid")]
    internal static FileExtensionToContentTypeDefinition hiddenFileExtensionDefinition;
}
```

Agregar el tipo de contenido a una exportación de editor

1. Cree una extensión de editor. Por ejemplo, puede utilizar la extensión de glifo de margen que se describe en [Tutorial: Crear un glifo de margen](#).
2. Agregue la clase que definió en este procedimiento.
3. Cuando exporte la clase [ContentTypeAttribute](#) de extensión, agregue un tipo "hid" a ella.

```
[Export]
[ContentType("hid")]
```

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Tutorial: Crear un glifo de margen

06/04/2020 • 8 minutes to read • [Edit Online](#)

Puede personalizar la apariencia de los márgenes del editor mediante extensiones de editor personalizadas. Este tutorial coloca un glifo personalizado en el margen del indicador cada vez que aparece la palabra "todo" en un comentario de código.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `TodoGlyphTest` nombre a la solución .
2. Agregue un elemento de proyecto Clasificador de editor. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.

Definir el glifo

Defina un glifo [IGlyphFactory](#) ejecutando la interfaz.

Para definir el glifo

1. Agregue un archivo de clase y asígnele el nombre `TodoGlyphFactory` .
2. Agregue el código siguiente mediante declaraciones.

```
using System.ComponentModel.Composition;
using System.Windows;
using System.Windows.Shapes;
using System.Windows.Media;
using System.Windows.Controls;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Formatting;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Utilities;
```

```
Imports System.ComponentModel.Composition
Imports System.Windows
Imports System.Windows.Shapes
Imports System.Windows.Media
Imports System.Windows.Controls
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Text.Formatting
Imports Microsoft.VisualStudio.Text.Tagging
Imports Microsoft.VisualStudio.Utilities
```

3. Agregue una `TodoGlyphFactory` clase denominada `IGlyphFactory` que implemente .

```
internal class TodoGlyphFactory : IGlyphFactory
```

```
Friend Class TodoGlyphFactory  
Implements IGlyphFactory
```

4. Agregue un campo privado que defina las dimensiones del glifo.

```
const double m_glyphSize = 16.0;
```

```
Const m_glyphSize As Double = 16.0
```

5. Implementar `GenerateGlyph` mediante la definición del elemento de interfaz de usuario (UI) glifo. `TodoTag` se define más adelante en este tutorial.

```
public UIElement GenerateGlyph(IWpfTextViewLine line, IGlyphTag tag)  
{  
    // Ensure we can draw a glyph for this marker.  
    if (tag == null || !(tag is TodoTag))  
    {  
        return null;  
    }  
  
    System.Windows.Shapes.Ellipse ellipse = new Ellipse();  
    ellipse.Fill = Brushes.LightBlue;  
    ellipse.StrokeThickness = 2;  
    ellipse.Stroke = Brushes.DarkBlue;  
    ellipse.Height = m_glyphSize;  
    ellipse.Width = m_glyphSize;  
  
    return ellipse;  
}
```

```
Public Function GenerateGlyph(ByVal line As IWpfTextViewLine, ByVal tag As IGlyphTag) As  
System.Windows.UIElement Implements IGlyphFactory.GenerateGlyph  
    ' Ensure we can draw a glyph for this marker.  
    If tag Is Nothing OrElse Not (TypeOf tag Is TodoTag) Then  
        Return Nothing  
    End If  
  
    Dim ellipse As Ellipse = New Ellipse()  
    ellipse.Fill = Brushes.LightBlue  
    ellipse.StrokeThickness = 2  
    ellipse.Stroke = Brushes.DarkBlue  
    ellipse.Height = m_glyphSize  
    ellipse.Width = m_glyphSize  
  
    Return ellipse  
  
End Function
```

6. Agregue una `TodoGlyphFactoryProvider` clase denominada `IGlyphFactoryProvider` que implemente . Exporte esta `NameAttribute` clase con un de `OrderAttribute` "TodoGlyph", `ContentTypeAttribute` un de After VsTextMarker, un de "código" y un `TagTypeAttribute` de TodoTag.

```
[Export(typeof(IGlyphFactoryProvider))]
[Name("TodoGlyph")]
[Order(After = "VsTextMarker")]
[ContentType("code")]
[TagType(typeof(TodoTag))]
internal sealed class TodoGlyphFactoryProvider : IGlyphFactoryProvider
```

```
<Export(GetType(IGlyphFactoryProvider), Name("TodoGlyph"), Order(After:="VsTextMarker"),
ContentType("code"), TagType(GetType(TodoTag)))>
Friend NotInheritable Class TodoGlyphFactoryProvider
    Implements IGlyphFactoryProvider
```

7. Implemente [GetGlyphFactory](#) el método creando una `TodoGlyphFactory` instancia de .

```
public IGlyphFactory GetGlyphFactory(IWpfTextView view, IWpfTextViewMargin margin)
{
    return new TodoGlyphFactory();
}
```

```
Public Function GetGlyphFactory(ByVal view As IWpfTextView, ByVal margin As IWpfTextViewMargin) As
IGlyphFactory Implements IGlyphFactoryProvider.GetGlyphFactory
    Return New TodoGlyphFactory()
End Function
```

Definir una etiqueta Todo y un tagger

Defina la relación entre el elemento de interfaz de usuario que definió en los pasos anteriores y el margen del indicador. Cree un tipo de etiqueta y un tagger y expórtelo mediante un proveedor de etiquetas.

Para definir una etiqueta de todo y un tagger

1. Agregue un nuevo archivo de clase `TodoTagger` al proyecto y asílo.
2. Agregue las importaciones siguientes.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Classification;
using Microsoft.VisualStudio.Utilities;
```

```
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Tagging
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Text.Classification
Imports Microsoft.VisualStudio.Utilities
```

3. Agregue una clase denominada `TodoTag` .

```
internal class TodoTag : IGlyphTag
```

```
Friend Class TodoTag
    Implements IGlyphTag

    Public Sub New()
        MyBase.New()
    End Sub
End Class
```

4. Modifique la `TodoTagger` clase denominada `ITagger<T>` que `TodoTag` implementa de tipo .

```
internal class TodoTagger : ITagger<TodoTag>
```

```
Friend Class TodoTagger
    Implements ITagger(Of TodoTag)
```

5. A `TodoTagger` la clase, agregue `IClassifier` campos privados para un y para que el texto se encuentre en los intervalos de clasificación.

```
private IClassifier m_classifier;
private const string m_searchText = "todo";
```

```
Private m_classifier As IClassifier
Private Const m_searchText As String = "todo"
```

6. Agregue un constructor que establezca el clasificador.

```
internal TodoTagger(IClassifier classifier)
{
    m_classifier = classifier;
}
```

```
Friend Sub New(ByVal classifier As IClassifier)
    m_classifier = classifier
End Sub
```

7. Implemente `GetTags` el método buscando todos los intervalos de clasificación cuyos nombres incluyen la palabra "comentario" y cuyo texto incluye el texto de búsqueda. Cada vez que se encuentra el `TagSpan<T>` texto `TodoTag` de búsqueda, devolver un nuevo tipo .

```

IEnumerable<ITagSpan<TodoTag>> ITagger<TodoTag>.GetTags(NormalizedSnapshotSpanCollection spans)
{
    foreach (SnapshotSpan span in spans)
    {
        //look at each classification span \
        foreach (ClassificationSpan classification in m_classifier.GetClassificationSpans(span))
        {
            //if the classification is a comment
            if (classification.ClassificationType.Classification.ToLower().Contains("comment"))
            {
                //if the word "todo" is in the comment,
                //create a new TodoTag TagSpan
                int index = classification.Span.GetText().ToLower().IndexOf(m_searchText);
                if (index != -1)
                {
                    yield return new TagSpan<TodoTag>(new SnapshotSpan(classification.Span.Start +
index, m_searchText.Length), new TodoTag());
                }
            }
        }
    }
}

```

```

Private Function GetTags(ByVal spans As NormalizedSnapshotSpanCollection) As IEnumerable(Of ITagSpan(Of
TodoTag)) Implements ITagger(Of TodoTag).GetTags
    Dim list As List(Of ITagSpan(Of TodoTag))
    list = New List(Of ITagSpan(Of TodoTag))()

    For Each span As SnapshotSpan In spans
        'look at each classification span \
        For Each classification As ClassificationSpan In m_classifier.GetClassificationSpans(span)
            'if the classification is a comment
            If classification.ClassificationType.Classification.ToLower().Contains("comment") Then
                'if the word "todo" is in the comment,
                'create a new TodoTag TagSpan
                Dim index As Integer = classification.Span.GetText().ToLower().IndexOf(m_searchText)
                If index <> -1 Then
                    list.Add(New TagSpan(Of TodoTag)(New SnapshotSpan(classification.Span.Start +
index, m_searchText.Length), New TodoTag()))
                End If
            End If
        Next classification
    Next span

    Return list
End Function

```

8. Declarar `TagsChanged` un evento.

```
public event EventHandler<SnapshotSpanEventArgs> TagsChanged;
```

```

Public Event TagsChanged(ByVal sender As Object, ByVal e As
Microsoft.VisualStudio.Text.SnapshotSpanEventArgs) Implements
Microsoft.VisualStudio.Text.Tagging.ITagger(Of TodoTag).TagsChanged

```

9. Agregue una `TodoTaggerProvider` clase denominada `ITaggerProvider` que implemente y `ContentTypeAttribute` expórtela `TagTypeAttribute` con un "código" y un `de TodoTag`.

```
[Export(typeof(ITaggerProvider))]
[ContentType("code")]
[TagType(typeof(TodoTag))]
class TodoTaggerProvider : ITaggerProvider
```

```
<Export(GetType(ITaggerProvider)), ContentType("code"), TagType(GetType(TodoTag))>
Friend Class TodoTaggerProvider
    Implements ITaggerProvider
```

10. Importe [IClassifierAggregatorService](#) el archivo .

```
[Import]
internal IClassifierAggregatorService AggregatorService;
```

```
<Import()>
Friend AggregatorService As IClassifierAggregatorService
```

11. Implemente [CreateTagger](#) el método creando una `TodoTagger` instancia de .

```
public ITagger<T> CreateTagger<T>(ITextBuffer buffer) where T : ITag
{
    if (buffer == null)
    {
        throw new ArgumentNullException("buffer");
    }

    return new TodoTagger(AggregatorService.GetClassifier(buffer)) as ITagger<T>;
}
```

```
Public Function CreateTagger(Of T As Microsoft.VisualStudio.Text.Tagging.ITag)(ByVal buffer As
Microsoft.VisualStudio.Text.ITextBuffer) As Microsoft.VisualStudio.Text.Tagging.ITagger(Of T)
Implements Microsoft.VisualStudio.Text.Tagging.ITaggerProvider.CreateTagger
    If buffer Is Nothing Then
        Throw New ArgumentNullException("buffer")
    End If

    Return TryCast(New TodoTagger(AggregatorService.GetClassifier(buffer)), ITagger(Of T))
End Function
```

Compile y pruebe el código

Para probar este código, compile la solución `TodoGlyphTest` y ejecútela en la instancia experimental.

Para compilar y probar la solución `TodoGlyphTest`

1. Compile la solución.
2. Ejecute el proyecto presionando **F5**. Se inicia una segunda instancia de Visual Studio.
3. Asegúrese de que se muestra el margen del indicador. (En el menú **Herramientas**, haga clic en **Opciones**. En la página Editor de **texto**, asegúrese de que **el margen del indicador** está seleccionado.)
4. Abra un archivo de código que tenga comentarios. Agregue la palabra "todo" a una de las secciones de comentarios.
5. Un círculo azul claro con un contorno azul oscuro aparece en el margen del indicador a la izquierda de la

ventana de código.

Tutorial: Personalizar la vista de texto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede personalizar una vista de texto modificando cualquiera de las siguientes propiedades en su mapa de formato editor:

- Margen del indicador
- Intercalación de inserción
- Sobrescribir el cuidado
- Texto seleccionado
- Texto seleccionado inactivo (es decir, texto seleccionado que ha perdido el foco)
- Espacio en blanco visible

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un proyecto MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto**, seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `ViewPropertyTest` nombre a la solución.
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.

Definir el tipo de contenido

1. Agregue un archivo de clase y asígnele el nombre `ViewPropertyModifier`.
2. Agregue las siguientes directivas `using`:

```
using System;
using System.Collections;
using System.Windows;
using System.Windows.Media;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text.Classification;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Utilities;
```



```
Imports System
Imports System.Collections
Imports System.Windows
Imports System.Windows.Media
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Text.Classification
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Utilities
```

3. Declare una `TextViewCreationListener` clase denominada `IWpfTextViewCreationListener` que herede de .
Exporte esta clase con los siguientes atributos:

- `ContentTypeAttribute` para especificar el tipo de contenido al que se aplica este agente de escucha.
- `TextViewRoleAttribute` para especificar el rol de este agente de escucha.

```
[Export(typeof(IWpfTextViewCreationListener))]
[ContentType("text")]
[TextViewRole(PredefinedTextViewRoles.Document)]
internal class TextViewCreationListener : IWpfTextViewCreationListener
```

```
<Export(GetType(IWpfTextViewCreationListener)), ContentType("text"),
TextViewRole(PredefinedTextViewRoles.Document)>
Friend Class TextViewCreationListener
    Implements IWpfTextViewCreationListener
```

4. En esta clase, `IEditorFormatMapService` importe el archivo .

```
[Import]
internal IEditorFormatMapService FormatMapService = null;
```

```
<Import()>
Friend FormatMapService As IEditorFormatMapService = Nothing
```

Cambiar las propiedades de la vista

1. Configure el `TextViewCreated` método para que las propiedades de la vista se cambien cuando se abra la vista. Para realizar el cambio, `ResourceDictionary` primero busque el que corresponde al aspecto de la vista que desea encontrar. A continuación, cambie la propiedad adecuada en el diccionario de recursos y establezca las propiedades. Lote las llamadas al `SetProperties BeginBatchUpdate` método mediante una llamada al `EndBatchUpdate` método antes de establecer las propiedades y, a continuación, el después de establecer las propiedades.

```
public void TextViewCreated(IWpfTextView textView)
{
    IEditorFormatMap formatMap = FormatMapService.GetEditorFormatMap(textView);

    ResourceDictionary regularCaretProperties = formatMap.GetProperties("Caret");
    ResourceDictionary overwriteCaretProperties = formatMap.GetProperties("Overwrite Caret");
    ResourceDictionary indicatorMargin = formatMap.GetProperties("Indicator Margin");
    ResourceDictionary visibleWhitespace = formatMap.GetProperties("Visible Whitespace");
    ResourceDictionary selectedText = formatMap.GetProperties("Selected Text");
    ResourceDictionary inactiveSelectedText = formatMap.GetProperties("Inactive Selected Text");

    formatMap.BeginBatchUpdate();

    regularCaretProperties[EditorFormatDefinition.ForegroundBrushId] = Brushes.Magenta;
    formatMap.SetProperties("Caret", regularCaretProperties);

    overwriteCaretProperties[EditorFormatDefinition.ForegroundBrushId] = Brushes.Turquoise;
    formatMap.SetProperties("Overwrite Caret", overwriteCaretProperties);

    indicatorMargin[EditorFormatDefinition.BackgroundColorId] = Colors.LightGreen;
    formatMap.SetProperties("Indicator Margin", indicatorMargin);

    visibleWhitespace[EditorFormatDefinition.ForegroundColorId] = Colors.Red;
    formatMap.SetProperties("Visible Whitespace", visibleWhitespace);

    selectedText[EditorFormatDefinition.BackgroundBrushId] = Brushes.LightPink;
    formatMap.SetProperties("Selected Text", selectedText);

    inactiveSelectedText[EditorFormatDefinition.BackgroundBrushId] = Brushes.DeepPink;
    formatMap.SetProperties("Inactive Selected Text", inactiveSelectedText);

    formatMap.EndBatchUpdate();
}
```

```

Public Sub TextViewCreated(ByVal textView As IWPfTextView) Implements
IWpfTextViewCreationListener.TextViewCreated

    Dim formatMap As IEditorFormatMap = FormatMapService.GetEditorFormatMap(textView)

    Dim regularCaretProperties As ResourceDictionary = formatMap.GetProperties("Caret")
    Dim overwriteCaretProperties As ResourceDictionary = formatMap.GetProperties("Overwrite Caret")
    Dim indicatorMargin As ResourceDictionary = formatMap.GetProperties("Indicator Margin")
    Dim visibleWhitespace As ResourceDictionary = formatMap.GetProperties("Visible Whitespace")
    Dim selectedText As ResourceDictionary = formatMap.GetProperties("Selected Text")
    Dim inactiveSelectedText As ResourceDictionary = formatMap.GetProperties("Inactive Selected Text")

    formatMap.BeginBatchUpdate()

    regularCaretProperties(EditorFormatDefinition.ForegroundBrushId) = Brushes.Magenta
    formatMap.SetProperties("Caret", regularCaretProperties)

    overwriteCaretProperties(EditorFormatDefinition.ForegroundBrushId) = Brushes.Turquoise
    formatMap.SetProperties("Overwrite Caret", overwriteCaretProperties)

    indicatorMargin(EditorFormatDefinition.BackgroundColorId) = Colors.LightGreen
    formatMap.SetProperties("Indicator Margin", indicatorMargin)

    visibleWhitespace(EditorFormatDefinition.ForegroundColorId) = Colors.Red
    formatMap.SetProperties("Visible Whitespace", visibleWhitespace)

    selectedText(EditorFormatDefinition.BackgroundBrushId) = Brushes.LightPink
    formatMap.SetProperties("Selected Text", selectedText)

    inactiveSelectedText(EditorFormatDefinition.BackgroundBrushId) = Brushes.DeepPink
    formatMap.SetProperties("Inactive Selected Text", inactiveSelectedText)

    formatMap.EndBatchUpdate()
End Sub

```

Compile y pruebe el código

1. Compile la solución.

Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.

2. Cree un archivo de texto y escriba algún texto.

- El intercalador de inserción debe ser magenta y el intercalador de sobrescritura debe ser de color turquesa.
- El margen del indicador (a la izquierda de la vista de texto) debe ser de color verde claro.

3. Seleccione el texto que escribió. El color del texto seleccionado debe ser de color rosa claro.

4. Mientras el texto está seleccionado, haga clic en cualquier lugar fuera de la ventana de texto. El color del texto seleccionado debe ser de color rosa oscuro.

5. Active el espacio en blanco visible. (En el menú **Edición**, seleccione **Avanzado** y, a continuación, haga clic en **Ver espacio en blanco**). Escriba algunas pestañas en el texto. Se deben mostrar las flechas rojas que representan las pestañas.

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)

Tutorial: Crear un adorno de vista, comandos y configuración (guías de columna)

06/04/2020 • 55 minutes to read • [Edit Online](#)

Puede ampliar el editor de texto/código de Visual Studio con comandos y efectos de vista. En este artículo se muestra cómo empezar a usar una característica de extensión popular, guías de columna. Las guías de columna son líneas visualmente claras dibujadas en la vista del editor de texto para ayudarle a administrar el código a anchos de columna específicos. Específicamente, el código con formato puede ser importante para los ejemplos que incluya en documentos, entradas de blog o informes de errores.

En este tutorial realizará lo siguiente:

- Crear un proyecto VSIX
- Añadir un adorno de vista de editor
- Agregar soporte para guardar y obtener ajustes (dónde dibujar guías de columna y su color)
- Añadir comandos (añadir/eliminar guías de columna, cambiar su color)
- Coloque los comandos en el menú Editar y en los menús contextuales del documento de texto
- Agregue compatibilidad para invocar los comandos desde la ventana de comandos de Visual Studio

Puede probar una versión de la característica de guías de columna con esta [extensión](#) de Galería de Visual Studio.

NOTE

En este tutorial, pegará una gran cantidad de código en algunos archivos generados por plantillas de extensión de Visual Studio. Pero, pronto este tutorial hará referencia a una solución completa en GitHub con otros ejemplos de extensión. El código completado es ligeramente diferente en que tiene iconos de comandos reales en lugar de usar iconos generictemplate.

Introducción

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Configuración de la solución

En primer lugar, cree un proyecto VSIX, agregue un adorno de vista de editor y, a continuación, agregue un comando (que agrega un VSPackage para poseer el comando). La arquitectura básica es la siguiente:

- Tiene un agente de escucha `ColumnGuideAdornment` de creación de vista de texto que crea un objeto por vista. Este objeto escucha eventos sobre el cambio de vista o la configuración cambiar, actualizar o volver a dibujar guías de columna según sea necesario.
- Hay un `GuidesSettingsManager` que controla la lectura y escritura desde el almacenamiento de configuración de Visual Studio. El administrador de configuración también tiene operaciones para actualizar la configuración que admite los comandos de usuario (agregar columna, quitar columna, cambiar color).

- Hay un paquete VSIP que es necesario si tiene comandos de usuario, pero es solo código reutilizable el que inicializa el objeto de implementación de comandos.
- Hay un `ColumnGuideCommands` objeto que ejecuta los comandos de usuario y enlaza los controladores de comandos para los comandos declarados en el archivo `.vsct`.

VSIX. Utilice el comando **File | New ...** para crear un proyecto. Elija el nodo **Extensibilidad** en C- en el panel de navegación izquierdo y elija **VSIX Project** en el panel derecho. Escriba el nombre **ColumnGuides** y elija **OK** para crear el proyecto.

Ver adorno. Presione el botón derecho del puntero en el nodo del proyecto en el Explorador de soluciones. Elija el comando **Agregar | nuevo elemento ...** para agregar un nuevo elemento de adorno de vista. Elija **Extensibilidad | Editor** en el panel de navegación izquierdo y elija Editor Viewport **Adornment** en el panel derecho. Escriba el nombre **ColumnGuideAdornment** como el nombre del elemento y elija **Agregar** para agregarlo.

Puede ver que esta plantilla de elemento ha agregado dos archivos al proyecto (así como referencias, etc.): **ColumnGuideAdornment.cs** y **ColumnGuideAdornmentTextViewCreationListener.cs**. Las plantillas dibujan un rectángulo púrpura en la vista. En la sección siguiente, se cambian un par de líneas en el agente de escucha de creación de vistas y se reemplaza el contenido de **ColumnGuideAdornment.cs**.

Comandos. En el Explorador de soluciones, presione el botón derecho del puntero en el nodo del proyecto. Elija el comando **Agregar | nuevo elemento ...** para agregar un nuevo elemento de adorno de vista. Elija **Extensibilidad | VSPackage** en el panel de navegación izquierdo y elija **Comando personalizado** en el panel derecho. Escriba el nombre **ColumnGuideCommands** como nombre del elemento y elija **Add**. Además de varias referencias, agregar los comandos y el paquete también **ColumnGuideCommands.cs**, **ColumnGuideCommandsPackage.csy** **ColumnGuideCommandsPackage.vsct**. En la siguiente sección, reemplazará el contenido del primer y último archivo para definir e implementar los comandos.

Configurar el agente de escucha de creación de vistas de texto

Abra *ColumnGuideAdornmentTextViewCreationListener.cs* en el editor. Este código implementa un controlador para cada vez que Visual Studio crea vistas de texto. Hay atributos que controlan cuándo se llama al controlador en función de las características de la vista.

El código también debe declarar una capa de adorno. Cuando el editor actualiza las vistas, obtiene las capas de adorno de la vista y de eso se obtienen los elementos de adorno. Puede declarar el orden de la capa en relación con otros con atributos. Reemplace la siguiente línea:

```
[Order(After = PredefinedAdornmentLayers.Caret)]
```

con estas dos líneas:

```
[Order(Before = PredefinedAdornmentLayers.Text)]
[TextViewRole(PredefinedTextViewRoles.Document)]
```

La línea reemplazada se encuentra en un grupo de atributos que declaran una capa de adorno. La primera línea que ha cambiado solo cambia cuando aparecen las líneas guía de columna. Dibujar las líneas "antes" del texto en la vista significa que aparecen detrás o debajo del texto. La segunda línea declara que los adornos de guía de columna son aplicables a las entidades de texto que se ajustan a la noción de un documento, pero podría declarar el adorno, por ejemplo, para que solo funcione para texto editable. Hay más información en el servicio de lenguaje y los [puntos](#) de extensión del editor

Implementar el administrador de configuración

Sustituya el contenido del *GuidesSettingsManager.cs* por el siguiente código (explicado a continuación):

```
using Microsoft.VisualStudio.Settings;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Settings;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Media;

namespace ColumnGuides
{
    internal static class GuidesSettingsManager
    {
        // Because my code is always called from the UI thred, this succeeds.
        internal static SettingsManager VsManagedSettingsManager =
            new ShellSettingsManager(ServiceProvider.GlobalProvider);

        private const int _maxGuides = 5;
        private const string _collectionSettingsName = "Text Editor";
        private const string _settingName = "Guides";
        // 1000 seems reasonable since primary scenario is long lines of code
        private const int _maxColumn = 1000;

        static internal bool AddGuideline(int column)
        {
            if (!IsValidColumn(column))
                throw new ArgumentOutOfRangeException(
                    "column",
                    "The parameter must be between 1 and " + _maxGuides.ToString());
            var offsets = GuidesSettingsManager.GetColumnOffsets();
            if (offsets.Count() >= _maxGuides)
                return false;
            // Check for duplicates
            if (offsets.Contains(column))
                return false;
            offsets.Add(column);
            WriteSettings(GuidesSettingsManager.GuidelinesColor, offsets);
            return true;
        }

        static internal bool RemoveGuideline(int column)
        {
            if (!IsValidColumn(column))
                throw new ArgumentOutOfRangeException(
                    "column", "The parameter must be between 1 and 10,000");
            var columns = GuidesSettingsManager.GetColumnOffsets();
            if (!columns.Remove(column))
            {
                // Not present. Allow user to remove the last column
                // even if they're not on the right column.
                if (columns.Count != 1)
                    return false;

                columns.Clear();
            }
            WriteSettings(GuidesSettingsManager.GuidelinesColor, columns);
            return true;
        }

        static internal bool CanAddGuideline(int column)
        {
            if (!IsValidColumn(column))
                return false;
        }
    }
}
```

```

        var offsets = GetColumnOffsets();
        if (offsets.Count >= _maxGuides)
            return false;
        return ! offsets.Contains(column);
    }

    static internal bool CanRemoveGuideline(int column)
    {
        if (! IsValidColumn(column))
            return false;
        // Allow user to remove the last guideline regardless of the column.
        // Okay to call count, we limit the number of guides.
        var offsets = GuidesSettingsManager.GetColumnOffsets();
        return offsets.Contains(column) || offsets.Count() == 1;
    }

    static internal void RemoveAllGuidelines()
    {
        WriteSettings(GuidesSettingsManager.GuidelinesColor, new int[0]);
    }

    private static bool IsValidColumn(int column)
    {
        // zero is allowed (per user request)
        return 0 <= column && column <= _maxColumn;
    }

    // This has format "RGB(<int>, <int>, <int>) <int> <int>...".
    // There can be any number of ints following the RGB part,
    // and each int is a column (char offset into line) where to draw.
    static private string _guidelinesConfiguration;
    static private string GuidelinesConfiguration
    {
        get
        {
            if (_guidelinesConfiguration == null)
            {
                _guidelinesConfiguration =
                    GetUserSettingsString(
                        GuidesSettingsManager._collectionSettingsName,
                        GuidesSettingsManager._settingName)
                    .Trim();
            }
            return _guidelinesConfiguration;
        }

        set
        {
            if (value != _guidelinesConfiguration)
            {
                _guidelinesConfiguration = value;
                WriteUserSettingsString(
                    GuidesSettingsManager._collectionSettingsName,
                    GuidesSettingsManager._settingName, value);
                // Notify ColumnGuideAdornments to update adornments in views.
                var handler = GuidesSettingsManager.SettingsChanged;
                if (handler != null)
                    handler();
            }
        }
    }

    internal static string GetUserSettingsString(string collection, string setting)
    {
        var store = GuidesSettingsManager
            .VsManagedSettingsManager
            .GetReadOnlySettingsStore(SettingsScope.UserSettings);
        return store.GetString(collection, setting, "RGB(255,0,0) 80");
    }

```

```

internal static void WriteUserSettingsString(string key, string propertyName,
                                           string value)
{
    var store = GuidesSettingsManager
        .VsManagedSettingsManager
        .GetWritableSettingsStore(SettingsScope.UserSettings);
    store.CreateCollection(key);
    store.SetString(key, propertyName, value);
}

// Persists settings and sets property with side effect of signaling
// ColumnGuideAdornments to update.
static private void WriteSettings(Color color, IEnumerable<int> columns)
{
    string value = ComposeSettingsString(color, columns);
    GuidelinesConfiguration = value;
}

private static string ComposeSettingsString(Color color,
                                           IEnumerable<int> columns)
{
    StringBuilder sb = new StringBuilder();
    sb.AppendFormat("RGB({0},{1},{2})", color.R, color.G, color.B);
    IEnumerator<int> columnsEnumerator = columns.GetEnumerator();
    if (columnsEnumerator.MoveNext())
    {
        sb.AppendFormat(" {0}", columnsEnumerator.Current);
        while (columnsEnumerator.MoveNext())
        {
            sb.AppendFormat(", {0}", columnsEnumerator.Current);
        }
    }
    return sb.ToString();
}

// Parse a color out of a string that begins like "RGB(255,0,0)"
static internal Color GuidelinesColor
{
    get
    {
        string config = GuidelinesConfiguration;
        if (!String.IsNullOrEmpty(config) && config.StartsWith("RGB("))
        {
            int lastParen = config.IndexOf(')');
            if (lastParen > 4)
            {
                string[] rgbs = config.Substring(4, lastParen - 4).Split(',');

                if (rgbs.Length >= 3)
                {
                    byte r, g, b;
                    if (byte.TryParse(rgbs[0], out r) &&
                        byte.TryParse(rgbs[1], out g) &&
                        byte.TryParse(rgbs[2], out b))
                    {
                        return Color.FromRgb(r, g, b);
                    }
                }
            }
        }
        return Colors.DarkRed;
    }

    set
    {
        WriteSettings(value, GetColumnOffsets());
    }
}

```



```

// Parse a list of integer values out of a string that looks like
// "RGB(255,0,0) 1, 5, 10, 80"
static internal List<int> GetColumnOffsets()
{
    var result = new List<int>();
    string settings = GuidesSettingsManager.GuidelinesConfiguration;
    if (String.IsNullOrEmpty(settings))
        return new List<int>();

    if (!settings.StartsWith("RGB("))
        return new List<int>();

    int lastParen = settings.IndexOf('');
    if (lastParen <= 4)
        return new List<int>();

    string[] columns = settings.Substring(lastParen + 1).Split(',');

    int columnCount = 0;
    foreach (string columnText in columns)
    {
        int column = -1;
        // VS 2008 gallery extension didn't allow zero, so per user request ...
        if (int.TryParse(columnText, out column) && column >= 0)
        {
            columnCount++;
            result.Add(column);
            if (columnCount >= _maxGuides)
                break;
        }
    }
    return result;
}

// Delegate and Event to fire when settings change so that ColumnGuideAdornments
// can update. We need nothing special in this event since the settings manager
// is statically available.
//
internal delegate void SettingsChangedHandler();
static internal event SettingsChangedHandler SettingsChanged;
}
}

```

La mayor parte de este código crea y analiza el formato de configuración: "<RGB(<int>, int>, int>) int>, int>, ...". Los enteros al final son las columnas basadas en uno donde desea guías de columna. La extensión de guías de columna captura todos sus valores en una cadena de valor de configuración única.

Hay algunas partes del código que vale la pena resaltar. La siguiente línea de código obtiene el contenedor administrado de Visual Studio para el almacenamiento de configuración. En su mayor parte, esto se abstrae sobre el registro de Windows, pero esta API es independiente del mecanismo de almacenamiento.

```

internal static SettingsManager VsManagedSettingsManager =
    new ShellSettingsManager(ServiceProvider.GlobalProvider);

```

El almacenamiento de configuración de Visual Studio usa un identificador de categoría y un identificador de configuración para identificar de forma única todas las configuraciones:

```

private const string _collectionSettingsName = "Text Editor";
private const string _settingName = "Guides";

```

No es necesario `"Text Editor"` utilizarse como nombre de categoría. Puedes elegir lo que quieras.

Las primeras funciones son los puntos de entrada para cambiar la configuración. Comprueban las restricciones de alto nivel, como el número máximo de guías permitidas. A continuación, `WriteSettings` llama a `GuidelinesConfiguration`, que compone una cadena de configuración y establece la propiedad `SettingsChanged`. Establecer esta propiedad guarda el valor de configuración en el almacén `ColumnGuideAdornment` de configuración de Visual Studio y desencadena el evento para actualizar todos los objetos, cada uno asociado a una vista de texto.

Hay un par de funciones de punto de entrada, como `CanAddGuideline`, que se utilizan para implementar comandos que cambian la configuración. Cuando Visual Studio muestra menús, consulta las implementaciones de comandos para ver si el comando está habilitado actualmente, cuál es su nombre, etc. A continuación puede ver cómo conectar estos puntos de entrada para las implementaciones de comandos. Para obtener más información sobre los [comandos](#), consulte [Extender menús y comandos](#).

Implementar la clase `ColumnGuideAdornment`

Se crea una instancia de la clase para cada vista de texto que puede tener adornos. Esta clase escucha eventos sobre el cambio de vista o el cambio de configuración, y las guías de columna de actualización o redibujo según sea necesario.

Sustituya el contenido del `ColumnGuideAdornment.cs` por el siguiente código (explicado a continuación):

```
using System;
using System.Windows.Media;
using Microsoft.VisualStudio.Text.Editor;
using System.Collections.Generic;
using System.Windows.Shapes;
using Microsoft.VisualStudio.Text.Formatting;
using System.Windows;

namespace ColumnGuides
{
    /// <summary>
    /// Adornment class, one instance per text view that draws a guides on the viewport
    /// </summary>
    internal sealed class ColumnGuideAdornment
    {
        private const double _lineThickness = 1.0;
        private IList<Line> _guidelines;
        private IWpfTextView _view;
        private double _baseIndentation;
        private double _columnWidth;

        /// <summary>
        /// Creates editor column guidelines
        /// </summary>
        /// <param name="view">The <see cref="IWpfTextView"/> upon
        /// which the adornment will be drawn</param>
        public ColumnGuideAdornment(IWpfTextView view)
        {
            _view = view;
            _guidelines = CreateGuidelines();
            GuidesSettingsManager.SettingsChanged +=
                new GuidesSettingsManager.SettingsChangedHandler(SettingsChanged);
            view.LayoutChanged +=
                new EventHandler<TextViewLayoutChangedEventArgs>(OnViewLayoutChanged);
            _view.Closed += new EventHandler(OnViewClosed);
        }

        void SettingsChanged()
        {
            _guidelines = CreateGuidelines();
            UpdatePositions();
        }
    }
}
```

```

        AddGuidelinesToAdornmentLayer();
    }

    void OnViewClosed(object sender, EventArgs e)
    {
        _view.LayoutChanged -= OnViewLayoutChanged;
        _view.Closed -= OnViewClosed;
        GuidesSettingsManager.SettingsChanged -= SettingsChanged;
    }

    private bool _firstLayoutDone;

    void OnViewLayoutChanged(object sender, TextViewLayoutChangedEventArgs e)
    {
        bool fUpdatePositions = false;

        IFormattedLineSource lineSource = _view.FormattedLineSource;
        if (lineSource == null)
        {
            return;
        }
        if (_columnWidth != lineSource.ColumnWidth)
        {
            _columnWidth = lineSource.ColumnWidth;
            fUpdatePositions = true;
        }
        if (_baseIndentation != lineSource.BaseIndentation)
        {
            _baseIndentation = lineSource.BaseIndentation;
            fUpdatePositions = true;
        }
        if (fUpdatePositions ||
            e.VerticalTranslation ||
            e.NewViewState.ViewportTop != e.OldViewState.ViewportTop ||
            e.NewViewState.ViewportBottom != e.OldViewState.ViewportBottom)
        {
            UpdatePositions();
        }
        if (!_firstLayoutDone)
        {
            AddGuidelinesToAdornmentLayer();
            _firstLayoutDone = true;
        }
    }

    private static IList<Line> CreateGuidelines()
    {
        Brush lineBrush = new SolidColorBrush(GuidesSettingsManager.GuidelinesColor);
        DoubleCollection dashArray = new DoubleCollection(new double[] { 1.0, 3.0 });
        IList<Line> result = new List<Line>();
        foreach (int column in GuidesSettingsManager.GetColumnOffsets())
        {
            Line line = new Line()
            {
                // Use the DataContext slot as a cookie to hold the column
                DataContext = column,
                Stroke = lineBrush,
                StrokeThickness = _lineThickness,
                StrokeDashArray = dashArray
            };
            result.Add(line);
        }
        return result;
    }

    void UpdatePositions()
    {
        foreach (Line line in _guidelines)
        {

```

```

        int column = (int)line.DataContext;
        line.X2 = _baseIndentation + 0.5 + column * _columnWidth;
        line.X1 = line.X2;
        line.Y1 = _view.ViewportTop;
        line.Y2 = _view.ViewportBottom;
    }
}

void AddGuidelinesToAdornmentLayer()
{
    // Grab a reference to the adornment layer that this adornment
    // should be added to
    // Must match exported name in ColumnGuideAdornmentTextViewCreationListener
    IAdornmentLayer adornmentLayer =
        _view.GetAdornmentLayer("ColumnGuideAdornment");
    if (adornmentLayer == null)
        return;
    adornmentLayer.RemoveAllAdornments();
    // Add the guidelines to the adornment layer and make them relative
    // to the viewport
    foreach (UIElement element in _guidelines)
        adornmentLayer.AddAdornment(AdornmentPositioningBehavior.OwnerControlled,
                                     null, null, element, null);
}
}
}

```

Las instancias de esta clase `IWpfTextView` se mantienen `Line` en la lista asociada y una lista de objetos dibujados en la vista.

El constructor (al que se llama desde `ColumnGuideAdornmentTextViewCreationListener` el `Line` momento en que Visual Studio crea nuevas vistas) crea los objetos de guía de columna. El constructor también agrega controladores `SettingsChanged` para `GuidesSettingsManager` el evento (definido en) y los eventos `LayoutChanged` de vista y `Closed` .

El `LayoutChanged` evento se desencadena debido a varios tipos de cambios en la vista, incluido cuando Visual Studio crea la vista. El `OnViewLayoutChanged` controlador `AddGuidelinesToAdornmentLayer` llama a ejecutar. El código `OnViewLayoutChanged` en determina si necesita actualizar las posiciones de línea en función de cambios como cambios de tamaño de fuente, canalones de vista, desplazamiento horizontal, etc. El código `UpdatePositions` de las líneas guía se dibuja entre caracteres o justo después de la columna de texto que se encuentra en el desplazamiento de caracteres especificado en la línea de texto.

Cada vez `SettingsChanged` que la configuración `Line` cambia la función simplemente vuelve a crear todos los objetos con cualquiera que sea la nueva configuración. Después de establecer las posiciones de `Line` línea, `ColumnGuideAdornment` el código elimina todos los objetos anteriores de la capa de adorno y agrega los nuevos.

Defina los comandos, menús y ubicaciones de menús

Puede haber mucho para declarar comandos y menús, colocar grupos de comandos o menús en varios otros menús y conectar controladores de comandos. En este tutorial se destaca cómo funcionan los comandos en esta extensión, pero para obtener información más detallada, consulte [Extender menús y comandos](#).

Introducción al código

La extensión Guías de columnas muestra la declaración de un grupo de comandos que pertenecen juntos (agregar columna, quitar columna, cambiar el color de línea) y, a continuación, colocar ese grupo en un submenú del menú contextual del editor. La extensión Guías de columnas también agrega los comandos al menú **Principal de edición**, pero los mantiene invisibles, que se describen como un patrón común a continuación.

Hay tres partes en la implementación de comandos: `ColumnGuideCommandsPackage.cs`,

ColumnGuideCommandsPackage.vsct y ColumnGuideCommands.cs. El código generado por las plantillas coloca un comando en el menú **Herramientas** que muestra un cuadro de diálogo como implementación. Puede ver cómo se implementa en los archivos *.vsct* y *ColumnGuideCommands.cs*, ya que es sencillo. Reemplace el código de estos archivos a continuación.

El código del paquete contiene declaraciones reutilizables necesarias para que Visual Studio detecte que la extensión ofrece comandos y para encontrar dónde colocar los comandos. Cuando se inicializa el paquete, crea una instancia de la clase de implementación de comandos. Para obtener más información acerca de los paquetes relacionados con comandos, consulte [Extender menús y comandos](#).

Un patrón de comandos común

Los comandos de la extensión Guías de columnas son un ejemplo de un patrón muy común en Visual Studio. Coloca comandos relacionados en un grupo y coloca ese grupo `<CommandFlag>CommandWellOnly</CommandFlag>` en un menú principal, a menudo con " " establecido para hacer que el comando sea invisible. Colocar comandos en los menús principales (como **Editar**) les da nombres bonitos (como **Edit.AddColumnGuide**), que son útiles para buscar comandos al volver a asignar enlaces de teclas en **Opciones** de herramientas. También es útil para obtener la finalización al invocar comandos desde la ventana de **comandos**.

A continuación, agregue el grupo de comandos a los menús contextuales o submenús donde espera que el usuario utilice los comandos. Visual Studio `CommandWellOnly` trata como una marca de invisibilidad solo para los menús principales. Al colocar el mismo grupo de comandos en un menú contextual o submenú, los comandos son visibles.

Como parte del patrón común, la extensión Guías de columnas crea un segundo grupo que contiene un único submenú. El submenú a su vez contiene el primer grupo con los comandos de guía de cuatro columnas. El segundo grupo que contiene el submenú es el activo reutilizable que se coloca en varios menús contextuales, que coloca un submenú en esos menús contextuales.

El archivo .vsct

El archivo *.vsct* declara los comandos y dónde van, junto con iconos, etc. Reemplace el contenido del archivo *.vsct* por el siguiente código (explicado a continuación):

```
<?xml version="1.0" encoding="utf-8"?>
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- This is the file that defines the actual layout and type of the commands.
  It is divided in different sections (e.g. command definition, command
  placement, ...), with each defining a specific set of properties.
  See the comment before each section for more details about how to
  use it. -->

  <!-- The VSCT compiler (the tool that translates this file into the binary
  format that VisualStudio will consume) has the ability to run a preprocessor
  on the vsct file; this preprocessor is (usually) the C++ preprocessor, so
  it is possible to define includes and macros with the same syntax used
  in C++ files. Using this ability of the compiler here, we include some files
  defining some of the constants that we will use inside the file. -->

  <!--This is the file that defines the IDs for all the commands exposed by
  VisualStudio. -->
  <Extern href="stdidcmd.h"/>

  <!--This header contains the command ids for the menus provided by the shell. -->
  <Extern href="vsshllids.h"/>

  <!--The Commands section is where commands, menus, and menu groups are defined.
  This section uses a Guid to identify the package that provides the command
  defined inside it. -->
  <Commands package="guidColumnGuideCommandsPkg">
    <!-- Inside this section we have different sub-sections: one for the menus, another
    for the menu groups, one for the buttons (the actual commands), one for the combos
```

and the last one for the bitmaps used. Each element is identified by a command id that is a unique pair of guid and numeric identifier; the guid part of the identifier is usually called "command set" and is used to group different command inside a logically related group; your package should define its own command set in order to avoid collisions with command ids defined by other packages. -->

<!-- In this section you can define new menu groups. A menu group is a container for other menus or buttons (commands); from a visual point of view you can see the group as the part of a menu contained between two lines. The parent of a group must be a menu. -->

<Groups>

<!-- The main group is parented to the edit menu. All the buttons within the group have the "CommandWellOnly" flag, so they're actually invisible, but it means they get canonical names that begin with "Edit". Using placements, the group is also placed in the GuidesSubMenu group. -->

<!-- The priority 0xB801 is chosen so it goes just after

IDG_VS_EDIT_COMMANDWELL -->

<Group guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup"
priority="0xB801">

<Parent guid="guidSHLMainMenu" id="IDM_VS_MENU_EDIT" />

</Group>

<!-- Group for holding the "Guidelines" sub-menu anchor (the item on the menu that drops the sub menu). The group is parented to the context menu for code windows. That takes care of most editors, but it's also placed in a couple of other windows using Placements -->

<Group guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
priority="0x0600">

<Parent guid="guidSHLMainMenu" id="IDM_VS_CTXT_CODEWIN" />

</Group>

</Groups>

<Menus>

<Menu guid="guidColumnGuidesCommandSet" id="GuidesSubMenu" priority="0x1000"
type="Menu">

<Parent guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup" />

<Strings>

<ButtonText>&Column Guides</ButtonText>

</Strings>

</Menu>

</Menus>

<!--Buttons section. -->

<!--This section defines the elements the user can interact with, like a menu command or a button or combo box in a toolbar. -->

<Buttons>

<!--To define a menu group you have to specify its ID, the parent menu and its display priority.

The command is visible and enabled by default. If you need to change the visibility, status, etc, you can use the CommandFlag node.

You can add more than one CommandFlag node e.g.:

<CommandFlag>DefaultInvisible</CommandFlag>

<CommandFlag>DynamicVisibility</CommandFlag>

If you do not want an image next to your command, remove the Icon node or set it to <Icon guid="guidOfficeIcon" id="msotcidNoIcon" /> -->

<Button guid="guidColumnGuidesCommandSet" id="cmdidAddColumnGuide"
priority="0x0100" type="Button">

<Parent guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup" />

<Icon guid="guidImages" id="bmpPicAddGuide" />

<CommandFlag>CommandWellOnly</CommandFlag>

<CommandFlag>AllowParams</CommandFlag>

<Strings>

<ButtonText>&Add Column Guide</ButtonText>

</Strings>

</Button>

```

<Button guid="guidColumnGuidesCommandSet" id="cmdidRemoveColumnGuide"
    priority="0x0101" type="Button">
    <Parent guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup" />
    <Icon guid="guidImages" id="bmpPicRemoveGuide" />
    <CommandFlag>CommandWellOnly</CommandFlag>
    <CommandFlag>AllowParams</CommandFlag>
    <Strings>
        <ButtonText>&Remove Column Guide</ButtonText>
    </Strings>
</Button>

<Button guid="guidColumnGuidesCommandSet" id="cmdidChooseGuideColor"
    priority="0x0103" type="Button">
    <Parent guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup" />
    <Icon guid="guidImages" id="bmpPicChooseColor" />
    <CommandFlag>CommandWellOnly</CommandFlag>
    <Strings>
        <ButtonText>Column Guide &Color...</ButtonText>
    </Strings>
</Button>

<Button guid="guidColumnGuidesCommandSet" id="cmdidRemoveAllColumnGuides"
    priority="0x0102" type="Button">
    <Parent guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup" />
    <CommandFlag>CommandWellOnly</CommandFlag>
    <Strings>
        <ButtonText>Remove A&ll Columns</ButtonText>
    </Strings>
</Button>
</Buttons>

<!--The bitmaps section is used to define the bitmaps that are used for the
    commands.-->
<Bitmaps>
    <!-- The bitmap id is defined in a way that is a little bit different from the
        others:
        the declaration starts with a guid for the bitmap strip, then there is the
        resource id of the bitmap strip containing the bitmaps and then there are
        the numeric ids of the elements used inside a button definition. An important
        aspect of this declaration is that the element id
        must be the actual index (1-based) of the bitmap inside the bitmap strip. -->
    <Bitmap guid="guidImages" href="Resources\ColumnGuideCommands.png"
        usedList="bmpPicAddGuide, bmpPicRemoveGuide, bmpPicChooseColor" />
</Bitmaps>

</Commands>

<CommandPlacements>

    <!-- Define secondary placements for our groups -->

    <!-- Place the group containing the three commands in the sub-menu -->
    <CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup"
        priority="0x0100">
        <Parent guid="guidColumnGuidesCommandSet" id="GuidesSubMenu" />
    </CommandPlacement>

    <!-- The HTML editor context menu, for some reason, redefines its own groups
        so we need to place a copy of our context menu there too. -->
    <CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
        priority="0x1001">
        <Parent guid="CMDSETID_HtmEdGrp" id="IDMX_HTM_SOURCE_HTML" />
    </CommandPlacement>

    <!-- The HTML context menu in Dev12 changed. -->
    <CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
        priority="0x1001">
        <Parent guid="CMDSETID_HtmEdGrp_Dev12" id="IDMX_HTM_SOURCE_HTML_Dev12" />
    </CommandPlacement>

```

```

</CommandPlacements>

<!-- Similarly for Script -->
<CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
    priority="0x1001">
    <Parent guid="CMDSETID_HtmEdGrp" id="IDMX_HTM_SOURCE_SCRIPT" />
</CommandPlacement>

<!-- Similarly for ASPX -->
<CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
    priority="0x1001">
    <Parent guid="CMDSETID_HtmEdGrp" id="IDMX_HTM_SOURCE_ASPX" />
</CommandPlacement>

<!-- Similarly for the XAML editor context menu -->
<CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
    priority="0x0600">
    <Parent guid="guidXamlUiCmds" id="IDM_XAML_EDITOR" />
</CommandPlacement>

</CommandPlacements>

<!-- This defines the identifiers and their values used above to index resources
    and specify commands. -->
<Symbols>
    <!-- This is the package guid. -->
    <GuidSymbol name="guidColumnGuideCommandsPkg"
        value="{e914e5de-0851-4904-b361-1a3a9d449704}" />

    <!-- This is the guid used to group the menu commands together -->
    <GuidSymbol name="guidColumnGuidesCommandSet"
        value="{c2bc0047-8bfa-4e5a-b5dc-45af8c274d8e}">
        <IDSymbol name="GuidesContextMenuGroup" value="0x1020" />
        <IDSymbol name="GuidesMenuItemsGroup" value="0x1021" />
        <IDSymbol name="GuidesSubMenu" value="0x1022" />
        <IDSymbol name="cmdidAddColumnGuide" value="0x0100" />
        <IDSymbol name="cmdidRemoveColumnGuide" value="0x0101" />
        <IDSymbol name="cmdidChooseGuideColor" value="0x0102" />
        <IDSymbol name="cmdidRemoveAllColumnGuides" value="0x0103" />
    </GuidSymbol>

    <GuidSymbol name="guidImages" value="{2C99F852-587C-43AF-AA2D-F605DE2E46EF}">
        <IDSymbol name="bmpPicAddGuide" value="1" />
        <IDSymbol name="bmpPicRemoveGuide" value="2" />
        <IDSymbol name="bmpPicChooseColor" value="3" />
    </GuidSymbol>

    <GuidSymbol name="CMDSETID_HtmEdGrp_Dev12"
        value="{78F03954-2FB8-4087-8CE7-59D71710B3BB}">
        <IDSymbol name="IDMX_HTM_SOURCE_HTML_Dev12" value="0x1" />
    </GuidSymbol>

    <GuidSymbol name="CMDSETID_HtmEdGrp" value="{d7e8c5e1-bdb8-11d0-9c88-0000f8040a53}">
        <IDSymbol name="IDMX_HTM_SOURCE_HTML" value="0x33" />
        <IDSymbol name="IDMX_HTM_SOURCE_SCRIPT" value="0x34" />
        <IDSymbol name="IDMX_HTM_SOURCE_ASPX" value="0x35" />
    </GuidSymbol>

    <GuidSymbol name="guidXamlUiCmds" value="{4c87b692-1202-46aa-b64c-ef01faec53da}">
        <IDSymbol name="IDM_XAML_EDITOR" value="0x103" />
    </GuidSymbol>
</Symbols>

</CommandTable>

```

GUIDS. Para que Visual Studio busque los controladores de comandos e invoquelos, debe asegurarse de que el GUID del paquete declarado en el archivo *ColumnGuideCommandsPackage.cs* (generado a partir de la plantilla de

elemento de proyecto) coincide con el GUID del paquete declarado en el archivo `.vsct` (copiado desde arriba). Si reutiliza este código de ejemplo, debe asegurarse de que tiene un GUID diferente para no entrar en conflicto con otra persona que pueda haber copiado este código.

Busque esta línea en `ColumnGuideCommandsPackage.cs` y copie el GUID entre comillas:

```
public const string PackageGuidIdString = "ef726849-5447-4f73-8de5-01b9e930f7cd";
```

A continuación, pegue el GUID en el archivo `.vsct` para que tenga la siguiente línea en las `Symbols` declaraciones:

```
<GuidSymbol name="guidColumnGuideCommandsPkg"
            value="{ef726849-5447-4f73-8de5-01b9e930f7cd}" />
```

Los GUID para el conjunto de comandos y el archivo de imagen de mapa de bits también deben ser únicos para las extensiones:

```
<GuidSymbol name="guidColumnGuidesCommandSet"
            value="{c2bc0047-8bfa-4e5a-b5dc-45af8c274d8e}">
<GuidSymbol name="guidImages" value="{2C99F852-587C-43AF-AA2D-F605DE2E46EF}">
```

Sin embargo, no es necesario cambiar el conjunto de comandos y los GUID de imagen de mapa de bits en este tutorial para que el código funcione. El GUID del conjunto de comandos debe coincidir con la declaración en el archivo `ColumnGuideCommands.cs`, pero también reemplaza el contenido de ese archivo; por lo tanto, los GUID coincidirán.

Otros GUID del archivo `.vsct` identifican los menús preexistentes a los que se agregan los comandos de guía de columna, por lo que nunca cambian.

Secciones de archivo. El *archivo .vsct* tiene tres secciones externas: comandos, ubicaciones y símbolos. La sección de comandos define grupos de comandos, menús, botones o elementos de menú, y mapas de bits para iconos. La sección de ubicaciones declara dónde van los grupos en los menús o ubicaciones adicionales en los menús preexistentes. La sección `symbols` declara los identificadores utilizados en otra parte del archivo `.vsct`, lo que hace que el código `.vsct` sea más legible que tener GUID y números hexadecimales en todas partes.

Sección Comandos, definiciones de grupos. La sección de comandos define primero los grupos de comandos. Los grupos de comandos son comandos que se ven en los menús con ligeras líneas grises que separan los grupos. Un grupo también puede rellenar un submenú completo, como en este ejemplo, y no verá las líneas de separación grises en este caso. Los archivos `.vsct` declaran `GuidesMenuItemsGroup` dos grupos, `IDM_VS_MENU_EDIT` el que se parentea en el `IDM_VS_CTXT_CODEWIN` (el menú principal de **edición**) y el `GuidesContextMenuGroup` que se parentea en el (menú contextual del editor de código).

La segunda declaración `0x0600` de grupo tiene prioridad:

```
<Group guid="guidColumnGuidesCommandSet" id="GuidesContextMenuGroup"
        priority="0x0600">
```

La idea es poner el submenú de guías de columna al final de cualquier menú contextual al que se agregue el submenú. Pero, usted no debe asumir que usted sabe mejor y forzar el `0xFFFF` submenú para ser siempre el último mediante el uso de una prioridad de . Tienes que experimentar con el número para ver dónde se encuentra tu submenú en los menús contextuales donde lo colocas. En este `0x0600` caso, es lo suficientemente alto como para ponerlo al final de los menús hasta donde se puede ver, pero deja espacio para que otra persona diseñe su extensión para que sea inferior a la extensión de las guías de columna si eso es deseable.

Sección Comandos, definición de menú. A continuación, la sección `GuidesSubMenu` de comandos

`GuidesContextMenuGroup` define el submenú , parented a la . Es `GuidesContextMenuGroup` el grupo que se agrega a todos los menús contextuales relevantes. En la sección de ubicaciones, el código coloca el grupo con los comandos de guía de cuatro columnas en este submenú.

Sección Comandos, definiciones de botones. A continuación, la sección de comandos define los elementos de menú o botones que son los comandos de guías de cuatro columnas. `CommandWellOnly` , discutido anteriormente, significa que los comandos son invisibles cuando se colocan en un menú principal. Dos de las declaraciones de botón de elemento `AllowParams` de menú (agregar guía y quitar guía) también tienen una marca:

```
<CommandFlag>AllowParams</CommandFlag>
```

Esta marca permite, junto con tener ubicaciones de menú principal, el comando para recibir argumentos cuando Visual Studio invoca el controlador de comandos. Si el usuario ejecuta el comando desde la ventana de comandos, el argumento se pasa al controlador de comandos en los argumentos de evento.

Secciones de comandos, definiciones de mapas de bits . Por último, la sección de comandos declara los mapas de bits o iconos utilizados para los comandos. Esta sección es una declaración simple que identifica el recurso del proyecto y enumera los índices basados en uno de los iconos usados. La sección symbols del archivo `.vsct` declara los valores de los identificadores utilizados como índices. En este tutorial se usa la franja de mapa de bits proporcionada con la plantilla de elemento de comando personalizado agregada al proyecto.

Sección Ubicaciones. Después de la sección de comandos es la sección de ubicaciones. El primero es donde el código agrega el primer grupo discutido anteriormente que contiene los comandos de guía de cuatro columnas al submenú donde aparecen los comandos:

```
<CommandPlacement guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup"
    priority="0x0100">
    <Parent guid="guidColumnGuidesCommandSet" id="GuidesSubMenu" />
</CommandPlacement>
```

Todas las demás ubicaciones agregan `GuidesContextMenuGroup` `GuidesSubMenu` el (que contiene el) a otros menús contextuales del editor. Cuando el código `GuidesContextMenuGroup` declaró el , se engendrará del menú contextual del editor de código. Es por eso que no ve una ubicación para el menú contextual del editor de código.

Sección Símbolos. Como se indicó anteriormente, la sección de símbolos declara los identificadores utilizados en otra parte del archivo `.vsct`, lo que hace que el código `.vsct` sea más legible que tener GUID y números hexadecimales en todas partes. Los puntos importantes de esta sección son que el GUID del paquete debe coincidir con la declaración en la clase de paquete. Además, el GUID del conjunto de comandos debe coincidir con la declaración en la clase de implementación de comandos.

Implementar los comandos

El archivo `ColumnGuideCommands.cs` implementa los comandos y enlaza los controladores. Cuando Visual Studio carga el paquete y lo `Initialize` inicializa, el paquete a su vez llama a la clase de implementación de comandos. La inicialización de comandos simplemente crea una instancia de la clase y el constructor enlaza todos los controladores de comandos.

Sustituya el contenido del archivo `ColumnGuideCommands.cs` por el siguiente código (explicado a continuación):

```
using System;
using System.ComponentModel.Design;
using System.Globalization;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;
using Microsoft.VisualStudio.TextManager.Interop;
using Microsoft.VisualStudio.Text.Editor;
```

```

using Microsoft.VisualStudio;

namespace ColumnGuides
{
    /// <summary>
    /// Command handler
    /// </summary>
    internal sealed class ColumnGuideCommands
    {
        const int cmdidAddColumnGuide = 0x0100;
        const int cmdidRemoveColumnGuide = 0x0101;
        const int cmdidChooseGuideColor = 0x0102;
        const int cmdidRemoveAllColumnGuides = 0x0103;

        /// <summary>
        /// Command menu group (command set GUID).
        /// </summary>
        static readonly Guid CommandSet =
            new Guid("c2bc0047-8bfa-4e5a-b5dc-45af8c274d8e");

        /// <summary>
        /// VS Package that provides this command, not null.
        /// </summary>
        private readonly Package package;

        OleMenuCommand _addGuidelineCommand;
        OleMenuCommand _removeGuidelineCommand;

        /// <summary>
        /// Initializes the singleton instance of the command.
        /// </summary>
        /// <param name="package">Owner package, not null.</param>
        public static void Initialize(Package package)
        {
            Instance = new ColumnGuideCommands(package);
        }

        /// <summary>
        /// Gets the instance of the command.
        /// </summary>
        public static ColumnGuideCommands Instance
        {
            get;
            private set;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="ColumnGuideCommands"/> class.
        /// Adds our command handlers for menu (commands must exist in the command
        /// table file)
        /// </summary>
        /// <param name="package">Owner package, not null.</param>
        private ColumnGuideCommands(Package package)
        {
            if (package == null)
            {
                throw new ArgumentNullException("package");
            }

            this.package = package;

            // Add our command handlers for menu (commands must exist in the .vsct file)

            OleMenuCommandService commandService =
                this.ServiceProvider.GetService(typeof(IMenuCommandService))
                    as OleMenuCommandService;
            if (commandService != null)
            {

```

```

        // Add guide
        _addGuidelineCommand =
            new OleMenuCommand(AddColumnGuideExecuted, null,
                               AddColumnGuideBeforeQueryStatus,
                               new CommandID(ColumnGuideCommands.CommandSet,
                                              cmdidAddColumnGuide));
        _addGuidelineCommand.ParametersDescription = "<column>";
        commandService.AddCommand(_addGuidelineCommand);
        // Remove guide
        _removeGuidelineCommand =
            new OleMenuCommand(RemoveColumnGuideExecuted, null,
                               RemoveColumnGuideBeforeQueryStatus,
                               new CommandID(ColumnGuideCommands.CommandSet,
                                              cmdidRemoveColumnGuide));
        _removeGuidelineCommand.ParametersDescription = "<column>";
        commandService.AddCommand(_removeGuidelineCommand);
        // Choose color
        commandService.AddCommand(
            new MenuCommand(ChooseGuideColorExecuted,
                            new CommandID(ColumnGuideCommands.CommandSet,
                                           cmdidChooseGuideColor)));

        // Remove all
        commandService.AddCommand(
            new MenuCommand(RemoveAllGuidelinesExecuted,
                            new CommandID(ColumnGuideCommands.CommandSet,
                                           cmdidRemoveAllColumnGuides)));
    }
}

/// <summary>
/// Gets the service provider from the owner package.
/// </summary>
private IServiceProvider ServiceProvider
{
    get
    {
        return this.package;
    }
}

private void AddColumnGuideBeforeQueryStatus(object sender, EventArgs e)
{
    int currentColumn = GetCurrentEditorColumn();
    _addGuidelineCommand.Enabled =
        GuidesSettingsManager.CanAddGuideline(currentColumn);
}

private void RemoveColumnGuideBeforeQueryStatus(object sender, EventArgs e)
{
    int currentColumn = GetCurrentEditorColumn();
    _removeGuidelineCommand.Enabled =
        GuidesSettingsManager.CanRemoveGuideline(currentColumn);
}

private int GetCurrentEditorColumn()
{
    IVsTextView view = GetActiveTextView();
    if (view == null)
    {
        return -1;
    }

    try
    {
        IWpfTextView textView = GetTextViewFromVsTextView(view);
        int column = GetCaretColumn(textView);

        // Note: GetCaretColumn returns 0-based positions. Guidelines are 1-based
        // ...

```

```

        // positions.
        // However, do not subtract one here since the caret is positioned to the
        // left of
        // the given column and the guidelines are positioned to the right. We
        // want the
        // guideline to line up with the current caret position. e.g. When the
        // caret is
        // at position 1 (zero-based), the status bar says column 2. We want to
        // add a
        // guideline for column 1 since that will place the guideline where the
        // caret is.
        return column;
    }
    catch (InvalidOperationException)
    {
        return -1;
    }
}

/// <summary>
/// Find the active text view (if any) in the active document.
/// </summary>
/// <returns>The IVsTextView of the active view, or null if there is no active
/// document or the
/// active view in the active document is not a text view.</returns>
private IVsTextView GetActiveTextView()
{
    IVsMonitorSelection selection =
        this.ServiceProvider.GetService(typeof(IVsMonitorSelection))
            as IVsMonitorSelection;

    object frameObj = null;
    ErrorHandler.ThrowOnFailure(
        selection.GetCurrentElementValue(
            (uint)VSConstants.VSSELELEMID.SEID_DocumentFrame, out frameObj));

    IVsWindowFrame frame = frameObj as IVsWindowFrame;
    if (frame == null)
    {
        return null;
    }

    return GetActiveView(frame);
}

private static IVsTextView GetActiveView(IVsWindowFrame windowFrame)
{
    if (windowFrame == null)
    {
        throw new ArgumentException("windowFrame");
    }

    object pvar;
    ErrorHandler.ThrowOnFailure(
        windowFrame.GetProperty((int)__VSFPROPID.VSFPROPID_DocView, out pvar));

    IVsTextView textView = pvar as IVsTextView;
    if (textView == null)
    {
        IVsCodeWindow codeWin = pvar as IVsCodeWindow;
        if (codeWin != null)
        {
            ErrorHandler.ThrowOnFailure(codeWin.GetLastActiveView(out textView));
        }
    }
    return textView;
}

private static IWpfTextView GetTextViewFromVsTextView(IVsTextView view)
{

```

```

        if (view == null)
        {
            throw new ArgumentNullException("view");
        }

        IVsUserData userData = view as IVsUserData;
        if (userData == null)
        {
            throw new InvalidOperationException();
        }

        object objTextViewHost;
        if (VSConstants.S_OK
            != userData.GetData(Microsoft.VisualStudio
                                .Editor
                                .DefGuidList.guidIWpfTextViewHost,
                                out objTextViewHost))
        {
            throw new InvalidOperationException();
        }

        IWpfTextViewHost textViewHost = objTextViewHost as IWpfTextViewHost;
        if (textViewHost == null)
        {
            throw new InvalidOperationException();
        }

        return textViewHost.TextView;
    }

    /// <summary>
    /// Given an IWpfTextView, find the position of the caret and report its column
    /// number. The column number is 0-based
    /// </summary>
    /// <param name="textView">The text view containing the caret</param>
    /// <returns>The column number of the caret's position. When the caret is at the
    /// leftmost column, the return value is zero.</returns>
    private static int GetCaretColumn(IWpfTextView textView)
    {
        // This is the code the editor uses to populate the status bar.
        Microsoft.VisualStudio.Text.Formatting.ITextViewLine caretViewLine =
            textView.Caret.ContainingTextViewLine;
        double columnWidth = textView.FormattedLineSource.ColumnWidth;
        return (int)(Math.Round((textView.Caret.Left - caretViewLine.Left)
                                / columnWidth));
    }

    /// <summary>
    /// Determine the applicable column number for an add or remove command.
    /// The column is parsed from command arguments, if present. Otherwise
    /// the current position of the caret is used to determine the column.
    /// </summary>
    /// <param name="e">Event args passed to the command handler.</param>
    /// <returns>The column number. May be negative to indicate the column number is
    /// unavailable.</returns>
    /// <exception cref="ArgumentException">The column number parsed from event args
    /// was not a valid integer.</exception>
    private int GetApplicableColumn(EventArgs e)
    {
        var inValue = ((OleMenuCmdEventArgs)e).InValue as string;
        if (!string.IsNullOrEmpty(inValue))
        {
            int column;
            if (!int.TryParse(inValue, out column) || column < 0)
                throw new ArgumentException("Invalid column");
            return column;
        }
    }

```

```

        return GetCurrentEditorColumn();
    }

    /// <summary>
    /// This function is the callback used to execute a command when the a menu item
    /// is clicked. See the Initialize method to see how the menu item is associated
    /// to this function using the OleMenuCommandService service and the MenuCommand
    /// class.
    /// </summary>
    private void AddColumnGuideExecuted(object sender, EventArgs e)
    {
        int column = GetApplicableColumn(e);
        if (column >= 0)
        {
            GuidesSettingsManager.AddGuideline(column);
        }
    }

    private void RemoveColumnGuideExecuted(object sender, EventArgs e)
    {
        int column = GetApplicableColumn(e);
        if (column >= 0)
        {
            GuidesSettingsManager.RemoveGuideline(column);
        }
    }

    private void RemoveAllGuidelinesExecuted(object sender, EventArgs e)
    {
        GuidesSettingsManager.RemoveAllGuidelines();
    }

    private void ChooseGuideColorExecuted(object sender, EventArgs e)
    {
        System.Windows.Media.Color color = GuidesSettingsManager.GuidelinesColor;

        using (System.Windows.Forms.ColorDialog picker =
            new System.Windows.Forms.ColorDialog())
        {
            picker.Color = System.Drawing.Color.FromArgb(255, color.R, color.G,
                color.B);
            if (picker.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                GuidesSettingsManager.GuidelinesColor =
                    System.Windows.Media.Color.FromRgb(picker.Color.R,
                        picker.Color.G,
                        picker.Color.B);
            }
        }
    }
}

```

Corregir referencias. Te falta una referencia en este punto. Presione el botón derecho del puntero en el referencias nodo en el Explorador de soluciones. Elija el comando **Add** El cuadro de diálogo **Agregar referencia** tiene un cuadro de búsqueda en la esquina superior derecha. Introduzca "editor" (sin comillas dobles). Elija el **Microsoft.VisualStudio.Editor** elemento (debe marcar la casilla a la izquierda del elemento, no sólo seleccione el elemento) y elija **Aceptar** para agregar la referencia.

Inicialización. Cuando se inicializa la clase `Initialize` de paquete, llama a la clase de implementación de comandos. La `ColumnGuideCommands` inicialización crea una instancia de la clase y guarda la instancia de clase y la referencia de paquete en los miembros de clase.

Echemos un vistazo a uno de los enlaces del controlador de comandos desde el constructor de clase:

```
_addGuidelineCommand =  
    new OleMenuCommand(AddColumnGuideExecuted, null,  
        AddColumnGuideBeforeQueryStatus,  
        new CommandID(ColumnGuideCommands.CommandSet,  
            cmdidAddColumnGuide));
```

Cree un `OleMenuCommand` archivo. Visual Studio usa el sistema de comandos de Microsoft Office. Los argumentos clave al crear `OleMenuCommand` instancias de `AddColumnGuideExecuted` la función que implementa el comando (`AddColumnGuideBeforeQueryStatus`), la función a la que se llama cuando Visual Studio muestra un menú con el comando () y el identificador de comando. Visual studio llama a la función de estado de consulta antes de mostrar un comando en un menú para que el comando pueda hacerse invisible o atenuado para una visualización determinada del menú (por ejemplo, deshabilitar **Copiar** si no hay selección), cambiar su icono o incluso cambiar su nombre (por ejemplo, desde Agregar algo para quitar algo), etc. El identificador de comando debe coincidir con un identificador de comando declarado en el archivo `.vsct`. Las cadenas para el conjunto de comandos y el comando de adición de guías de columna deben coincidir entre el archivo `.vsct` y el `ColumnGuideCommands.cs`.

La siguiente línea proporciona asistencia para cuando los usuarios invocan el comando a través de la ventana de comandos (explicada a continuación):

```
_addGuidelineCommand.ParametersDescription = "<column>";
```

Estado de la consulta. El estado `AddColumnGuideBeforeQueryStatus` de `RemoveColumnGuideBeforeQueryStatus` la consulta funciona y comprueba algunas opciones de configuración (como el número máximo de guías o la columna max) o si hay una guía de columna que se va a quitar. Habilitan los comandos si las condiciones son correctas. Las funciones de estado de consulta deben ser eficaces porque se ejecutan cada vez que Visual Studio muestra un menú y para cada comando del menú.

AddColumnGuideExecuted (función). La parte interesante de agregar una guía es averiguar la vista actual del editor y la ubicación del intercalación. En primer lugar, esta función llama `GetApplicableColumn` a , que comprueba si hay un argumento proporcionado por el usuario en los argumentos de evento del controlador de comandos y, si no hay ninguno, la función comprueba la vista del editor:

```
private int GetApplicableColumn(EventArgs e)  
{  
    var inValue = ((OleMenuCmdEventArgs)e).InValue as string;  
    if (!string.IsNullOrEmpty(inValue))  
    {  
        int column;  
        if (!int.TryParse(inValue, out column) || column < 0)  
            throw new ArgumentException("Invalid column");  
        return column;  
    }  
  
    return GetCurrentEditorColumn();  
}
```

`GetCurrentEditorColumn` tiene que cavar un `IWpfTextView` poco para obtener una vista del código. Si realiza `GetActiveTextView` un `GetActiveView` seguimiento `GetTextViewFromVsTextView` a través de , , y , puede ver cómo hacerlo. El código siguiente es el código relevante abstraído, comenzando con la selección actual, luego obtener `IVsTextView` el marco `IVsUserData` de la selección, a continuación, obtener `docView` del marco como un , a continuación, obtener un de la `IVsTextView`, a continuación, obtener un host de vista y, finalmente, el `IWpfTextView`:


```

IVsMonitorSelection selection =
    this.ServiceProvider.GetService(typeof(IVsMonitorSelection))
        as IVsMonitorSelection;
object frameObj = null;

ErrorHandler.ThrowOnFailure(selection.GetCurrentElementValue(
    (uint)VSConstants.VSSELELEMID.SEID_DocumentFrame,
    out frameObj));

IVsWindowFrame frame = frameObj as IVsWindowFrame;
if (frame == null)
    <<do nothing>>;

...
object pvar;
ErrorHandler.ThrowOnFailure(frame.GetProperty((int)__VSFPROPID.VSFPROPID_DocView,
    out pvar));

IVsTextView textView = pvar as IVsTextView;
if (textView == null)
{
    IVsCodeWindow codeWin = pvar as IVsCodeWindow;
    if (codeWin != null)
    {
        ErrorHandler.ThrowOnFailure(codeWin.GetLastActiveView(out textView));
    }
}

...
if (textView == null)
    <<do nothing>>

IVsUserData userData = textView as IVsUserData;
if (userData == null)
    <<do nothing>>

object objTextViewHost;
if (VSConstants.S_OK
    != userData.GetData(Microsoft.VisualStudio.Editor.DefGuidList
        .guidIWpfTextViewHost,
        out objTextViewHost))
{
    <<do nothing>>
}

IWpfTextViewHost textViewHost = objTextViewHost as IWpfTextViewHost;
if (textViewHost == null)
    <<do nothing>>

IWpfTextView textView = textViewHost.TextView;

```

Una vez que tenga un `IWpfTextView`, puede obtener la columna donde se encuentra el nombre de intercalación:

```

private static int GetCaretColumn(IWpfTextView textView)
{
    // This is the code the editor uses to populate the status bar.
    Microsoft.VisualStudio.Text.Formatting.ITextViewLine caretViewLine =
        textView.Caret.ContainingTextViewLine;
    double columnWidth = textView.FormattedLineSource.ColumnWidth;
    return (int)(Math.Round((textView.Caret.Left - caretViewLine.Left)
        / columnWidth));
}

```

Con la columna actual en la mano donde el usuario hace clic, el código solo llama al administrador de configuración para agregar o quitar la columna. El administrador de configuración activa `ColumnGuideAdornment` el evento en el que escuchan todos los objetos. Cuando se desencadena el evento, estos objetos actualizan sus vistas de texto asociadas con la nueva configuración de la guía de columnas.

Invocar comando desde la ventana de comandos

El ejemplo de guías de columna permite a los usuarios invocar dos comandos desde la ventana de comandos como una forma de extensibilidad. Si utiliza el comando **Ver | otra ventana de comandos de Windows |**, puede ver la ventana de comandos. Puede interactuar con la ventana de comandos escribiendo "edit.", y con el nombre del comando completion y proporcionando el argumento 120, tiene el siguiente resultado:

```
> Edit.AddColumnGuide 120
>
```

Las partes del ejemplo que habilitan este comportamiento se encuentran `ColumnGuideCommands` en las declaraciones de archivo `.vsct`, el constructor de clase cuando enlaza controladores de comandos y las implementaciones del controlador de comandos que comprueban los argumentos de evento.

Ha visto `<CommandFlag>CommandWellOnly</CommandFlag>` " " en el archivo `.vsct`, así como las ubicaciones en el menú principal **Editar** aunque los comandos no se muestran en la interfaz de usuario del menú **Editar**. Tenerlos en el menú **principal Editar** les da nombres como **Edit.AddColumnGuide**. La declaración del grupo de comandos que contiene los cuatro comandos colocó el grupo en el menú **Editar** directamente:

```
<Group guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup"
  priority="0xB801">
  <Parent guid="guidSHLMainMenu" id="IDM_VS_MENU_EDIT" />
</Group>
```

La sección de botones más tarde declaró los comandos `CommandWellOnly` para mantenerlos invisibles en el menú principal y los declaró con: `AllowParams`

```
<Button guid="guidColumnGuidesCommandSet" id="cmdidAddColumnGuide"
  priority="0x0100" type="Button">
  <Parent guid="guidColumnGuidesCommandSet" id="GuidesMenuItemsGroup" />
  <Icon guid="guidImages" id="bmpPicAddGuide" />
  <CommandFlag>CommandWellOnly</CommandFlag>
  <CommandFlag>AllowParams</CommandFlag>
```

Ha visto el controlador de `ColumnGuideCommands` comandos enlazar código en el constructor de clase proporcionado una descripción del parámetro permitido:

```
_addGuidelineCommand.ParametersDescription = "<column>";
```

Ha visto `GetApplicableColumn` la `OleMenuCmdEventArgs` función comprueba si hay un valor antes de comprobar la vista del editor para una columna actual:

```
private int GetApplicableColumn(EventArgs e)
{
    var inValue = ((OleMenuCmdEventArgs)e).InValue as string;
    if (!string.IsNullOrEmpty(inValue))
    {
        int column;
        if (!int.TryParse(inValue, out column) || column < 0)
            throw new ArgumentException("Invalid column");
        return column;
    }
}
```

Pruebe su extensión

Ahora puede presionar **F5** para ejecutar la extensión Guías de columnas. Abra un archivo de texto y utilice el menú contextual del editor para agregar líneas guía, eliminarlas y cambiar su color. Haga clic en el texto (no en el espacio en blanco pasó el final de la línea) para agregar una guía de columna o el editor lo agrega a la última columna de la línea. Si utiliza la ventana de comandos e invoca los comandos con un argumento, puede agregar guías de columna en cualquier lugar.

Si desea probar diferentes ubicaciones de comandos, cambiar nombres, cambiar iconos, etc., y tiene algún problema con Visual Studio que muestra el código más reciente en los menús, puede restablecer el subárbol experimental en el que está depurando. Para abrir el menú Inicio de **Windows**, escriba "reset". Busque y ejecute el comando **Reset the Next Visual Studio Experimental Instance**. Este comando limpia el subárbol del registro experimental de todos los componentes de extensión. No limpia la configuración de los componentes, por lo que las guías que tenía al apagar el subárbol experimental de Visual Studio siguen ahí cuando el código lee el almacén de configuración en el próximo inicio.

Proyecto de código terminado

Pronto habrá un proyecto de GitHub de ejemplos de extensibilidad de Visual Studio y el proyecto completado estará allí. Este artículo se actualizará para que apunte allí cuando eso suceda. El proyecto de ejemplo completado puede tener diferentes guids y tendrá una tira de mapas de bits diferente para los iconos de comando.

Puede probar una versión de la característica de guías de columna con esta [extensión](#) de Galería de Visual Studio.

Vea también

- [Dentro del editor](#)
- [Ampliar el editor y los servicios de idiomas](#)
- [Servicio de lenguaje y puntos de extensión del editor](#)
- [Extender menús y comandos](#)
- [Añadir un submenú a un menú](#)
- [Crear una extensión con una plantilla de elemento de editor](#)

Tutorial: Resaltar texto

06/04/2020 • 11 minutes to read • [Edit Online](#)

Puede agregar diferentes efectos visuales al editor mediante la creación de componentes de Managed Extensibility Framework (MEF). En este tutorial se muestra cómo resaltar cada aparición de la palabra actual en un archivo de texto. Si una palabra se produce más de una vez en un archivo de texto y coloca el intercalador en una ocurrencia, se resalta cada aparición.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un proyecto MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto**, seleccione **Visual C/Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `HighlightWordTest` nombre a la solución.
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.

Definir un TextMarkerTag

El primer paso para resaltar texto [TextMarkerTag](#) es subclase y definir su apariencia.

Para definir un TextMarkerTag y un MarkerFormatDefinition

1. Agregue un archivo de clase y asímócle el nombre `HighlightWordTag`.
2. Agregue las siguientes referencias:
 - a. `Microsoft.VisualStudio.CoreUtility`
 - b. `Microsoft.VisualStudio.Text.Data`
 - c. `Microsoft.VisualStudio.Text.Logic`
 - d. `Microsoft.VisualStudio.Text.UI`
 - e. `Microsoft.VisualStudio.Text.UI.Wpf`
 - f. `System.ComponentModel.Composition`
 - g. `Presentation.Core`
 - h. `Presentation.Framework`
3. Importe los siguientes espacios de nombres.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using System.Linq;
using System.Threading;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Classification;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Operations;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Utilities;
using System.Windows.Media;
```

4. Cree una clase que **TextMarkerTag** herede **HighlightWordTag** de ella y asíánte .

```
internal class HighlightWordTag : TextMarkerTag
{
}
}
```

5. Cree una segunda clase **MarkerFormatDefinition** que herede **HighlightWordFormatDefinition** de , y asímóquela. Para utilizar esta definición de formato para la etiqueta, debe exportarla con los siguientes atributos:

- **NameAttribute**: las etiquetas lo utilizan para hacer referencia a este formato
- **UserVisibleAttribute**: esto hace que el formato aparezca en la interfaz de usuario

```
[Export(typeof(EditorFormatDefinition))]
[Name("MarkerFormatDefinition/HighlightWordFormatDefinition")]
[UserVisible(true)]
internal class HighlightWordFormatDefinition : MarkerFormatDefinition
{
}
}
```

6. En el constructor de **HighlightWordFormatDefinition**, defina su nombre para mostrar y apariencia. La propiedad **Background** define el color de relleno, mientras que la propiedad **Foreground** define el color del borde.

```
public HighlightWordFormatDefinition()
{
    this.BackgroundColor = Colors.LightBlue;
    this.ForegroundColor = Colors.DarkBlue;
    this.DisplayName = "Highlight Word";
    this.ZOrder = 5;
}
```

7. En el constructor de **HighlightWordTag**, pase el nombre de la definición de formato que ha creado.

```
public HighlightWordTag() : base("MarkerFormatDefinition/HighlightWordFormatDefinition") { }
```

Implementar un **ITagger**

El siguiente paso es **ITagger<T>** implementar la interfaz. Esta interfaz asigna, a un búfer de texto determinado,

etiquetas que proporcionan resaltado de texto y otros efectos visuales.

Para implementar un tagger

1. Cree una clase `ITagger<T>` que `HighlightWordTag` implemente de `HighlightWordTagger` tipo y asimismo.

```
internal class HighlightWordTagger : ITagger<HighlightWordTag>
{
}
}
```

2. Agregue los siguientes campos y propiedades privados a la clase:

- Un `ITextView`, que corresponde a la vista de texto actual.
- Un `ITextBuffer`, que corresponde al búfer de texto que subyace a la vista de texto.
- Un `ITextSearchService`, que se utiliza para buscar texto.
- Un `ITextStructureNavigator`, que tiene métodos para navegar dentro de intervalos de texto.
- A `NormalizedSnapshotSpanCollection`, que contiene el conjunto de palabras que se resaltará.
- A `SnapshotSpan`, que corresponde a la palabra actual.
- A `SnapshotPoint`, que corresponde a la posición actual del intercalador.
- Un objeto de bloqueo.

```
ITextView View { get; set; }
ITextBuffer SourceBuffer { get; set; }
ITextSearchService TextSearchService { get; set; }
ITextStructureNavigator TextStructureNavigator { get; set; }
NormalizedSnapshotSpanCollection WordSpans { get; set; }
SnapshotSpan? CurrentWord { get; set; }
SnapshotPoint RequestedPoint { get; set; }
object updateLock = new object();
```

3. Agregue un constructor que inicialice las propiedades `LayoutChanged` `PositionChanged` enumeradas anteriormente y agregue y controladores de eventos.

```
public HighlightWordTagger(ITextView view, ITextBuffer sourceBuffer, ITextSearchService
textSearchService,
ITextStructureNavigator textStructureNavigator)
{
    this.View = view;
    this.SourceBuffer = sourceBuffer;
    this.TextSearchService = textSearchService;
    this.TextStructureNavigator = textStructureNavigator;
    this.WordSpans = new NormalizedSnapshotSpanCollection();
    this.CurrentWord = null;
    this.View.Caret.PositionChanged += CaretPositionChanged;
    this.View.LayoutChanged += ViewLayoutChanged;
}
```

4. Los controladores de `UpdateAtCaretPosition` eventos llaman al método.

```

void ViewLayoutChanged(object sender, TextViewLayoutChangedEventArgs e)
{
    // If a new snapshot wasn't generated, then skip this layout
    if (e.NewSnapshot != e.OldSnapshot)
    {
        UpdateAtCaretPosition(View.Caret.Position);
    }
}

void CaretPositionChanged(object sender, CaretPositionChangedEventArgs e)
{
    UpdateAtCaretPosition(e.NewPosition);
}

```

5. También debe agregar `TagsChanged` un evento al que llama el método `update`.

```

public event EventHandler<SnapshotSpanEventArgs> TagsChanged;

```

```

Public Event TagsChanged(ByVal sender As Object, ByVal e As SnapshotSpanEventArgs) _
    Implements ITagger(Of HighlightWordTag).TagsChanged

```

6. El `UpdateAtCaretPosition()` método busca cada palabra en el búfer de texto que es idéntica a [SnapshotSpan](#) la palabra donde se coloca el cursor y construye una lista de objetos que corresponden a las apariciones de la palabra. A continuación, llama `SynchronousUpdate` a `TagsChanged`, que genera el evento.

```

void UpdateAtCaretPosition(CaretPosition caretPosition)
{
    SnapshotPoint? point = caretPosition.Point.GetPoint(SourceBuffer, caretPosition.Affinity);

    if (!point.HasValue)
        return;

    // If the new caret position is still within the current word (and on the same snapshot), we don't
    need to check it
    if (CurrentWord.HasValue
        && CurrentWord.Value.Snapshot == View.TextSnapshot
        && point.Value >= CurrentWord.Value.Start
        && point.Value <= CurrentWord.Value.End)
    {
        return;
    }

    RequestedPoint = point.Value;
    UpdateWordAdornments();
}

void UpdateWordAdornments()
{
    SnapshotPoint currentRequest = RequestedPoint;
    List<SnapshotSpan> wordSpans = new List<SnapshotSpan>();
    //Find all words in the buffer like the one the caret is on
    TextExtent word = TextStructureNavigator.GetExtentOfWord(currentRequest);
    bool foundWord = true;
    //If we've selected something not worth highlighting, we might have missed a "word" by a little
    bit
    if (!WordExtentIsValid(currentRequest, word))
    {
        //Before we retry, make sure it is worthwhile
        if (word.Span.Start != currentRequest
            || currentRequest == currentRequest.GetContainingLine().Start
            || currentRequest == currentRequest.GetContainingLine().End)
        {
            return;
        }
    }
}

```

```

        || char.IsWhiteSpace((currentRequest - 1).GetChar()))
    {
        foundWord = false;
    }
    else
    {
        // Try again, one character previous.
        //If the caret is at the end of a word, pick up the word.
        word = TextStructureNavigator.GetExtentOfWord(currentRequest - 1);

        //If the word still isn't valid, we're done
        if (!WordExtentIsValid(currentRequest, word))
            foundWord = false;
    }
}

if (!foundWord)
{
    //If we couldn't find a word, clear out the existing markers
    SynchronousUpdate(currentRequest, new NormalizedSnapshotSpanCollection(), null);
    return;
}

SnapshotSpan currentWord = word.Span;
//If this is the current word, and the caret moved within a word, we're done.
if (CurrentWord.HasValue && currentWord == CurrentWord)
    return;

//Find the new spans
FindData findData = new FindData(currentWord.GetText(), currentWord.Snapshot);
findData.FindOptions = FindOptions.WholeWord | FindOptions.MatchCase;

wordSpans.AddRange(TextSearchService.FindAll(findData));

//If another change hasn't happened, do a real update
if (currentRequest == RequestedPoint)
    SynchronousUpdate(currentRequest, new NormalizedSnapshotSpanCollection(wordSpans),
currentWord);
}
static bool WordExtentIsValid(SnapshotPoint currentRequest, TextExtent word)
{
    return word.IsSignificant
        && currentRequest.Snapshot.GetText(word.Span).Any(c => char.IsLetter(c));
}

```

7. El `SynchronousUpdate` realiza una actualización `WordSpans` sincrónica en el `CurrentWord` propiedades y provoca el `TagsChanged` evento.


```

void SynchronousUpdate(SnapshotPoint currentRequest, NormalizedSnapshotSpanCollection newSpans,
SnapshotSpan? newCurrentWord)
{
    lock (updateLock)
    {
        if (currentRequest != RequestedPoint)
            return;

        WordSpans = newSpans;
        CurrentWord = newCurrentWord;

        var tempEvent = TagsChanged;
        if (tempEvent != null)
            tempEvent(this, new SnapshotSpanEventArgs(new SnapshotSpan(SourceBuffer.CurrentSnapshot,
0, SourceBuffer.CurrentSnapshot.Length)));
    }
}

```

8. Debe implementar [GetTags](#) el método. Este método toma [SnapshotSpan](#) una colección de objetos y devuelve una enumeración de intervalos de etiquetas.

Implemente este método como un iterador de rendimiento, que permite la evaluación diferida (es decir, la evaluación del conjunto solo cuando se tiene acceso a elementos individuales) de las etiquetas. En Visual Basic, agregue las etiquetas a una lista y devuelva la lista.

Aquí el método [TagSpan<T>](#) devuelve un objeto [TextMarkerTag](#) que tiene un "azul", que proporciona un fondo azul.

```

public IEnumerable<ITagSpan<HighlightWordTag>> GetTags(NormalizedSnapshotSpanCollection spans)
{
    if (CurrentWord == null)
        yield break;

    // Hold on to a "snapshot" of the word spans and current word, so that we maintain the same
    // collection throughout
    SnapshotSpan currentWord = CurrentWord.Value;
    NormalizedSnapshotSpanCollection wordSpans = WordSpans;

    if (spans.Count == 0 || wordSpans.Count == 0)
        yield break;

    // If the requested snapshot isn't the same as the one our words are on, translate our spans to
    the expected snapshot
    if (spans[0].Snapshot != wordSpans[0].Snapshot)
    {
        wordSpans = new NormalizedSnapshotSpanCollection(
            wordSpans.Select(span => span.TranslateTo(spans[0].Snapshot,
                SpanTrackingMode.EdgeExclusive)));

        currentWord = currentWord.TranslateTo(spans[0].Snapshot, SpanTrackingMode.EdgeExclusive);
    }

    // First, yield back the word the cursor is under (if it overlaps)
    // Note that we'll yield back the same word again in the wordspans collection;
    // the duplication here is expected.
    if (spans.OverlapsWith(new NormalizedSnapshotSpanCollection(currentWord)))
        yield return new TagSpan<HighlightWordTag>(currentWord, new HighlightWordTag());

    // Second, yield all the other words in the file
    foreach (SnapshotSpan span in NormalizedSnapshotSpanCollection.Overlap(spans, wordSpans))
    {
        yield return new TagSpan<HighlightWordTag>(span, new HighlightWordTag());
    }
}

```

Crear un proveedor de Tagger

Para crear el etiquetador, [IViewTaggerProvider](#) debe implementar un archivo. Esta clase es una parte del componente MEF, por lo que debe establecer los atributos correctos para que se reconozca esta extensión.

NOTE

Para obtener más información acerca de MEF, vea [Managed Extensibility Framework \(MEF\)](#).

Para crear un proveedor de etiquetas

1. Cree una `HighlightWordTaggerProvider` clase denominada [IViewTaggerProvider](#) que implemente y [ContentTypeAttribute](#) expórtela [TagTypeAttribute](#) `TextMarkerTag` con un de "texto" y un archivo.

```

[Export(typeof(IViewTaggerProvider))]
[ContentType("text")]
[TagType(typeof(TextMarkerTag))]
internal class HighlightWordTaggerProvider : IViewTaggerProvider
{ }

```

2. Debe importar dos servicios [ITextSearchService](#) de [ITextStructureNavigatorSelectorService](#) editor, el y el, para crear instancias del etiquetador.

```
[Import]
internal ITextSearchService TextSearchService { get; set; }

[Import]
internal ITextStructureNavigatorSelectorService TextStructureNavigatorSelector { get; set; }
```

3. Implemente [CreateTagger](#) el método para `HighlightWordTagger` devolver una instancia de .

```
public ITagger<T> CreateTagger<T>(ITextView textView, ITextBuffer buffer) where T : ITag
{
    //provide highlighting only on the top buffer
    if (textView.TextBuffer != buffer)
        return null;

    ITextStructureNavigator textStructureNavigator =
        TextStructureNavigatorSelector.GetTextStructureNavigator(buffer);

    return new HighlightWordTagger(textView, buffer, TextSearchService, textStructureNavigator) as
    ITagger<T>;
}
```

Compile y pruebe el código

Para probar este código, compile la solución `HighlightWordTest` y ejecútela en la instancia experimental.

Para compilar y probar la solución `HighlightWordTest`

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Cree un archivo de texto y escriba algún texto en el que se repitan las palabras, por ejemplo, "hola hola hola".
4. Coloque el cursor en una de las apariciones de "hola". Cada ocurrencia debe resaltarse en azul.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Esquematización

06/04/2020 • 16 minutes to read • [Edit Online](#)

Configure características basadas en lenguaje, como la esquematización, definiendo los tipos de regiones de texto que desea expandir o contraer. Puede definir regiones en el contexto de un servicio de lenguaje, o definir su propia extensión de nombre de archivo y tipo de contenido y aplicar la definición de región solo a ese tipo, o aplicar las definiciones de región a un tipo de contenido existente (por ejemplo, "texto"). En este tutorial se muestra cómo definir y mostrar regiones de esquematización.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un proyecto de Managed Extensibility Framework (MEF)

Para crear un nuevo proyecto de MEF

1. Crear un proyecto de VSIX. Asigne a la solución el nombre `OutlineRegionTest`.
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte Crear una extensión con una plantilla de elemento de [editor](#).
3. Elimine los archivos de clase existentes.

Implementar un etiquetado de esquematización

Las regiones de esquematización `OutliningRegionTag` están marcadas por un tipo de etiqueta (). Esta etiqueta proporciona el comportamiento de esquematización estándar. La región delineada se puede expandir o contraer. La región delineada está marcada `**+**` por un signo Más ()- si está contraída o un signo Menos () si se expande y la región expandida está demarcada por una línea vertical.

Los pasos siguientes muestran cómo definir un etiquetador que crea regiones de esquematización para todas las regiones delimitadas por los corchetes ([,]).

Para implementar un tagger de esquematización

1. Agregue un archivo de clase y asígnele el nombre `OutliningTagger`.
2. Importe los siguientes espacios de nombres.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text.Outlining;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Utilities;
using Microsoft.VisualStudio.Text;
```

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Text.Outlining
Imports Microsoft.VisualStudio.Text.Tagging
Imports Microsoft.VisualStudio.Utilities
Imports Microsoft.VisualStudio.Text
```

3. Cree una `OutliningTagger` clase denominada y `ITagger<T>` haga que implemente:

```
internal sealed class OutliningTagger : ITagger<IOutliningRegionTag>
```

```
Friend NotInheritable Class OutliningTagger
    Implements ITagger(Of IOutliningRegionTag)
```

4. Agregue algunos campos para realizar un seguimiento del búfer de texto y la instantánea y para acumular los conjuntos de líneas que se deben etiquetar como regiones de esquematización. Este código incluye una lista de regiones objetos (que se definirán más adelante) que representan las regiones de esquematización.

```
string startHide = "[";    //the characters that start the outlining region
string endHide = "]";      //the characters that end the outlining region
string ellipsis = "...";   //the characters that are displayed when the region is collapsed
string hoverText = "hover text"; //the contents of the tooltip for the collapsed span
ITextBuffer buffer;
ITextSnapshot snapshot;
List<Region> regions;
```

```
'the characters that start the outlining region
Private startHide As String = "["
'the characters that end the outlining region
Private endHide As String = "]"
'the characters that are displayed when the region is collapsed
Private ellipsis As String = "..."
'the contents of the tooltip for the collapsed span
Private hoverText As String = "hover text"
Private buffer As ITextBuffer
Private snapshot As ITextSnapshot
Private regions As List(Of Region)
```

5. Agregue un constructor de etiqueta que inicialice los campos, analice el `Changed` búfer y agregue un controlador de eventos al evento.

```
public OutliningTagger(ITextBuffer buffer)
{
    this.buffer = buffer;
    this.snapshot = buffer.CurrentSnapshot;
    this.regions = new List<Region>();
    this.ReParse();
    this.buffer.Changed += BufferChanged;
}
```

```

Public Sub New(ByVal buffer As ITextBuffer)
    Me.buffer = buffer
    Me.snapshot = buffer.CurrentSnapshot
    Me.regions = New List(Of Region)()
    Me.ReParse()
    AddHandler Me.buffer.Changed, AddressOf BufferChanged
End Sub

```

6. Implemente [GetTags](#) el método, que crea una instancia de los intervalos de etiquetas. En este ejemplo se supone [NormalizedSpanCollection](#) que los intervalos de los pasados al método son contiguos, aunque no siempre puede ser el caso. Este método crea una instancia de un nuevo intervalo de etiquetas para cada una de las regiones de esquematización.

```

public IEnumerable<ITagSpan<IOutliningRegionTag>> GetTags(NormalizedSnapshotSpanCollection spans)
{
    if (spans.Count == 0)
        yield break;
    List<Region> currentRegions = this.regions;
    ITextSnapshot currentSnapshot = this.snapshot;
    SnapshotSpan entire = new SnapshotSpan(spans[0].Start, spans[spans.Count - 1].End).TranslateTo(currentSnapshot, SpanTrackingMode.EdgeExclusive);
    int startLineNumber = entire.Start.GetContainingLine().LineNumber;
    int endLineNumber = entire.End.GetContainingLine().LineNumber;
    foreach (var region in currentRegions)
    {
        if (region.StartLine <= endLineNumber &&
            region.EndLine >= startLineNumber)
        {
            var startLine = currentSnapshot.GetLineFromLineNumber(region.StartLine);
            var endLine = currentSnapshot.GetLineFromLineNumber(region.EndLine);

            //the region starts at the beginning of the "[", and goes until the *end* of the line that
            contains the "]".
            yield return new TagSpan<IOutliningRegionTag>(
                new SnapshotSpan(startLine.Start + region.StartOffset,
                    endLine.End),
                new OutliningRegionTag(false, false, ellipsis, hoverText));
        }
    }
}

```

```

Public Function GetTags(ByVal spans As NormalizedSnapshotSpanCollection) As IEnumerable(Of ITagSpan(Of
IOutliningRegionTag)) Implements ITagger(Of
Microsoft.VisualStudio.Text.Tagging.IOutliningRegionTag).GetTags
    If spans.Count = 0 Then
        Return Nothing
        Exit Function
    End If
    Dim currentRegions As List(Of Region) = Me.regions
    Dim currentSnapshot As ITextSnapshot = Me.snapshot
    Dim entire As SnapshotSpan = New SnapshotSpan(spans(0).Start, spans(spans.Count - 1).
[End]).TranslateTo(currentSnapshot, SpanTrackingMode.EdgeExclusive)
    Dim startLineNumber As Integer = entire.Start.GetContainingLine().LineNumber
    Dim endLineNumber As Integer = entire.[End].GetContainingLine().LineNumber

    Dim list As List(Of ITagSpan(Of IOutliningRegionTag))
    list = New List(Of ITagSpan(Of IOutliningRegionTag))()

    For Each region In currentRegions
        If region.StartLine <= endLineNumber AndAlso region.EndLine >= startLineNumber Then
            Dim startLine = currentSnapshot.GetLineFromLineNumber(region.StartLine)
            Dim endLine = currentSnapshot.GetLineFromLineNumber(region.EndLine)

            'the region starts at the beginning of the "[", and goes until the *end* of the line that
contains the "]".
            list.Add(New TagSpan(Of IOutliningRegionTag)(New SnapshotSpan(startLine.Start +
region.StartOffset, endLine.End),
                New OutliningRegionTag(False, False, ellipsis, hoverText)))
        End If
    Next
    Return list
End Function

```

7. Declare `TagsChanged` un controlador de eventos.

```
public event EventHandler<SnapshotSpanEventArgs> TagsChanged;
```

```
Public Event TagsChanged As EventHandler(Of SnapshotSpanEventArgs) Implements ITagger(Of
IOutliningRegionTag).TagsChanged
```

8. Agregue `BufferChanged` un controlador de [Changed](#) eventos que responda a eventos analizando el búfer de texto.

```

void BufferChanged(object sender, TextContentChangedEventArgs e)
{
    // If this isn't the most up-to-date version of the buffer, then ignore it for now (we'll
eventually get another change event).
    if (e.After != buffer.CurrentSnapshot)
        return;
    this.ReParse();
}

```

```

Private Sub BufferChanged(ByVal sender As Object, ByVal e As TextContentChangedEventArgs)
    ' If this isn't the most up-to-date version of the buffer, then ignore it for now (we'll eventually
get another change event).
    If e.After IsNot buffer.CurrentSnapshot Then
        Exit Sub
    End If
    Me.ReParse()
End Sub

```

9. Agregue un método que analice el búfer. El ejemplo que se muestra aquí es solo para la ilustración. Analiza sincrónicamente el búfer en regiones de esquematización anidadas.

```
void ReParse()
{
    ITextSnapshot newSnapshot = buffer.CurrentSnapshot;
    List<Region> newRegions = new List<Region>();

    //keep the current (deepest) partial region, which will have
    // references to any parent partial regions.
    PartialRegion currentRegion = null;

    foreach (var line in newSnapshot.Lines)
    {
        int regionStart = -1;
        string text = line.GetText();

        //lines that contain a "[" denote the start of a new region.
        if ((regionStart = text.IndexOf(startHide, StringComparison.Ordinal)) != -1)
        {
            int currentLevel = (currentRegion != null) ? currentRegion.Level : 1;
            int newLevel;
            if (!TryGetLevel(text, regionStart, out newLevel))
                newLevel = currentLevel + 1;

            //levels are the same and we have an existing region;
            //end the current region and start the next
            if (currentLevel == newLevel && currentRegion != null)
            {
                newRegions.Add(new Region()
                {
                    Level = currentRegion.Level,
                    StartLine = currentRegion.StartLine,
                    StartOffset = currentRegion.StartOffset,
                    EndLine = line.LineNumber
                });

                currentRegion = new PartialRegion()
                {
                    Level = newLevel,
                    StartLine = line.LineNumber,
                    StartOffset = regionStart,
                    PartialParent = currentRegion.PartialParent
                };
            }
            //this is a new (sub)region
            else
            {
                currentRegion = new PartialRegion()
                {
                    Level = newLevel,
                    StartLine = line.LineNumber,
                    StartOffset = regionStart,
                    PartialParent = currentRegion
                };
            }
        }
        //lines that contain "]" denote the end of a region
        else if ((regionStart = text.IndexOf(endHide, StringComparison.Ordinal)) != -1)
        {
            int currentLevel = (currentRegion != null) ? currentRegion.Level : 1;
            int closingLevel;
            if (!TryGetLevel(text, regionStart, out closingLevel))
                closingLevel = currentLevel;

            //the regions match
            if (currentRegion != null &&
```



```

        currentLevel == closingLevel)
    {
        newRegions.Add(new Region()
        {
            Level = currentLevel,
            StartLine = currentRegion.StartLine,
            StartOffset = currentRegion.StartOffset,
            EndLine = line.LineNumber
        });

        currentRegion = currentRegion.PartialParent;
    }
}

//determine the changed span, and send a changed event with the new spans
List<Span> oldSpans =
    new List<Span>(this.regions.Select(r => AsSnapshotSpan(r, this.snapshot)
        .TranslateTo(newSnapshot, SpanTrackingMode.EdgeExclusive)
        .Span));
List<Span> newSpans =
    new List<Span>(newRegions.Select(r => AsSnapshotSpan(r, newSnapshot).Span));

NormalizedSpanCollection oldSpanCollection = new NormalizedSpanCollection(oldSpans);
NormalizedSpanCollection newSpanCollection = new NormalizedSpanCollection(newSpans);

//the changed regions are regions that appear in one set or the other, but not both.
NormalizedSpanCollection removed =
    NormalizedSpanCollection.Difference(oldSpanCollection, newSpanCollection);

int changeStart = int.MaxValue;
int changeEnd = -1;

if (removed.Count > 0)
{
    changeStart = removed[0].Start;
    changeEnd = removed[removed.Count - 1].End;
}

if (newSpans.Count > 0)
{
    changeStart = Math.Min(changeStart, newSpans[0].Start);
    changeEnd = Math.Max(changeEnd, newSpans[newSpans.Count - 1].End);
}

this.snapshot = newSnapshot;
this.regions = newRegions;

if (changeStart <= changeEnd)
{
    ITextSnapshot snap = this.snapshot;
    if (this.TagsChanged != null)
        this.TagsChanged(this, new SnapshotSpanEventArgs(
            new SnapshotSpan(this.snapshot, Span.FromBounds(changeStart, changeEnd))));
}
}

```

```

Private Sub Reparse()
    Dim newSnapshot As ITextSnapshot = buffer.CurrentSnapshot
    Dim newRegions As New List(Of Region)()

    'keep the current (deepest) partial region, which will have
    ' references to any parent partial regions.
    Dim currentRegion As PartialRegion = Nothing

    For Each line In newSnapshot.Lines
        Dim regionStart As Integer = -1
        Dim text As String = line.GetText()

```

```

Dim text As String = line.GetText()

'lines that contain a "[" denote the start of a new region.
If text.IndexOf(startHide, StringComparison.Ordinal) <> -1 Then
    regionStart = text.IndexOf(startHide, StringComparison.Ordinal)
    Dim currentLevel As Integer = If((currentRegion IsNot Nothing), currentRegion.Level, 1)
    Dim newLevel As Integer
    If Not TryGetLevel(text, regionStart, newLevel) Then
        newLevel = currentLevel + 1
    End If

    'levels are the same and we have an existing region;
    'end the current region and start the next
    If currentLevel = newLevel AndAlso currentRegion IsNot Nothing Then
        Dim newRegion = New Region()
        newRegion.Level = currentRegion.Level
        newRegion.StartLine = currentRegion.StartLine
        newRegion.StartOffset = currentRegion.StartOffset
        newRegion.EndLine = line.LineNumber
        newRegions.Add(newRegion)

        currentRegion = New PartialRegion()
        currentRegion.Level = newLevel
        currentRegion.StartLine = line.LineNumber
        currentRegion.StartOffset = regionStart
        currentRegion.PartialParent = currentRegion.PartialParent
    Else
        'this is a new (sub)region
        currentRegion = New PartialRegion()
        currentRegion.Level = newLevel
        currentRegion.StartLine = line.LineNumber
        currentRegion.StartOffset = regionStart
        currentRegion.PartialParent = currentRegion
    End If
    'lines that contain "]" denote the end of a region
ElseIf (text.IndexOf(endHide, StringComparison.Ordinal)) <> -1 Then
    regionStart = text.IndexOf(endHide, StringComparison.Ordinal)
    Dim currentLevel As Integer = If((currentRegion IsNot Nothing), currentRegion.Level, 1)
    Dim closingLevel As Integer
    If Not TryGetLevel(text, regionStart, closingLevel) Then
        closingLevel = currentLevel
    End If

    'the regions match
    If currentRegion IsNot Nothing AndAlso currentLevel = closingLevel Then
        Dim newRegion As Region
        newRegion = New Region()
        newRegion.Level = currentLevel
        newRegion.StartLine = currentRegion.StartLine
        newRegion.StartOffset = currentRegion.StartOffset
        newRegion.EndLine = line.LineNumber
        newRegions.Add(newRegion)

        currentRegion = currentRegion.PartialParent
    End If
End If
Next

'determine the changed span, and send a changed event with the new spans
Dim oldSpans As New List(Of Span)(Me.regions.[Select](Function(r) AsSnapshotSpan(r, Me.snapshot).TranslateTo(newSnapshot, SpanTrackingMode.EdgeExclusive).Span))
Dim newSpans As New List(Of Span)(newRegions.[Select](Function(r) AsSnapshotSpan(r, newSnapshot).Span))

Dim oldSpanCollection As New NormalizedSpanCollection(oldSpans)
Dim newSpanCollection As New NormalizedSpanCollection(newSpans)

'the changed regions are regions that appear in one set or the other, but not both.
Dim removed As NormalizedSpanCollection = NormalizedSpanCollection.Difference(oldSpanCollection,

```

```

newSpanCollection)

Dim changeStart As Integer = Integer.MaxValue
Dim changeEnd As Integer = -1

If removed.Count > 0 Then
    changeStart = removed(0).Start
    changeEnd = removed(removed.Count - 1).[End]
End If

If newSpans.Count > 0 Then
    changeStart = Math.Min(changeStart, newSpans(0).Start)
    changeEnd = Math.Max(changeEnd, newSpans(newSpans.Count - 1).[End])
End If

Me.snapshot = newSnapshot
Me.regions = newRegions

If changeStart <= changeEnd Then
    Dim snap As ITextSnapshot = Me.snapshot
    RaiseEvent TagsChanged(Me, New SnapshotSpanEventArgs(New SnapshotSpan(Me.snapshot,
Span.FromBounds(changeStart, changeEnd))))
End If
End Sub

```

10. El siguiente método auxiliar obtiene un entero que representa el nivel de la esquematización, de forma que 1 es el par de llaves más a la izquierda.

```

static bool TryGetLevel(string text, int startIndex, out int level)
{
    level = -1;
    if (text.Length > startIndex + 3)
    {
        if (int.TryParse(text.Substring(startIndex + 1), out level))
            return true;
    }

    return false;
}

```

```

Private Shared Function TryGetLevel(ByVal text As String, ByVal startIndex As Integer, ByRef level As
Integer) As Boolean
    level = -1
    If text.Length > startIndex + 3 Then
        If Integer.TryParse(text.Substring(startIndex + 1), level) Then
            Return True
        End If
    End If

    Return False
End Function

```

11. El siguiente método auxiliar traduce una región (definida más adelante en este artículo) en un SnapshotSpan.

```

static SnapshotSpan AsSnapshotSpan(Region region, ITextSnapshot snapshot)
{
    var startLine = snapshot.GetLineFromLineNumber(region.StartLine);
    var endLine = (region.StartLine == region.EndLine) ? startLine
        : snapshot.GetLineFromLineNumber(region.EndLine);
    return new SnapshotSpan(startLine.Start + region.StartOffset, endLine.End);
}

```

```

Private Shared Function AsSnapshotSpan(ByVal region As Region, ByVal snapshot As ITextSnapshot) As
    SnapshotSpan
    Dim startLine = snapshot.GetLineFromLineNumber(region.StartLine)
    Dim endLine = If((region.StartLine = region.EndLine), startLine,
        snapshot.GetLineFromLineNumber(region.EndLine))
    Return New SnapshotSpan(startLine.Start + region.StartOffset, endLine.[End])
End Function

```

12. El código siguiente es solo para la ilustración. Define una clase PartialRegion que contiene el número de línea y el desplazamiento del inicio de una región de esquematización y una referencia a la región primaria (si existe). Este código permite al analizador configurar regiones de esquematización anidadas. Una clase Region derivada contiene una referencia al número de línea del final de una región de esquematización.

```

class PartialRegion
{
    public int StartLine { get; set; }
    public int StartOffset { get; set; }
    public int Level { get; set; }
    public PartialRegion PartialParent { get; set; }
}

class Region : PartialRegion
{
    public int EndLine { get; set; }
}

```

```

Private Class PartialRegion
    Private _StartLine As Integer
    Public Property StartLine() As Integer
        Get
            Return _StartLine
        End Get
        Set(ByVal value As Integer)
            _StartLine = value
        End Set
    End Property
    Private _StartOffset As Integer
    Public Property StartOffset() As Integer
        Get
            Return _StartOffset
        End Get
        Set(ByVal value As Integer)
            _StartOffset = value
        End Set
    End Property
    Private _Level As Integer
    Public Property Level() As Integer
        Get
            Return _Level
        End Get
        Set(ByVal value As Integer)
            _Level = value
        End Set
    End Property
    Private _PartialParent As PartialRegion
    Public Property PartialParent() As PartialRegion
        Get
            Return _PartialParent
        End Get
        Set(ByVal value As PartialRegion)
            _PartialParent = value
        End Set
    End Property
End Class

Private Class Region
    Inherits PartialRegion
    Private _EndLine As Integer
    Public Property EndLine() As Integer
        Get
            Return _EndLine
        End Get
        Set(ByVal value As Integer)
            _EndLine = value
        End Set
    End Property
End Class

```

Implementar un proveedor de etiquetas

Exporte un proveedor de etiquetas para su etiqueta. El proveedor de `OutliningTagger` etiquetado crea un búfer para un búfer del `OutliningTagger` tipo de contenido "text" o, de lo contrario, devuelve un si el búfer ya tiene uno.

Para implementar un proveedor de etiquetas

1. Cree una `OutliningTaggerProvider` clase denominada `ITaggerProvider` que implemente y expórtela con los atributos `ContentType` y `TagType`.

```
[Export(typeof(ITaggerProvider))]
[TagType(typeof(IOutliningRegionTag))]
[ContentType("text")]
internal sealed class OutliningTaggerProvider : ITaggerProvider
```

```
<Export(GetType(ITaggerProvider))> _
<TagType(GetType(IOutliningRegionTag))> _
<ContentType("text")> _
Friend NotInheritable Class OutliningTaggerProvider
    Implements ITaggerProvider
```

2. Implemente [CreateTagger](#) el método `OutliningTagger` agregando un a las propiedades del búfer.

```
public ITagger<T> CreateTagger<T>(ITextBuffer buffer) where T : ITag
{
    //create a single tagger for each buffer.
    Func<ITagger<T>> sc = delegate() { return new OutliningTagger(buffer) as ITagger<T>; };
    return buffer.Properties.GetOrCreateSingletonProperty<ITagger<T>>(sc);
}
```

```
Public Function CreateTagger(Of T As ITag)(ByVal buffer As ITextBuffer) As ITagger(Of T) Implements
ITaggerProvider.CreateTagger
    'create a single tagger for each buffer.
    Dim sc As Func(Of ITagger(Of T)) = Function() TryCast(New OutliningTagger(buffer), ITagger(Of T))
    Return buffer.Properties.GetOrCreateSingletonProperty(Of ITagger(Of T))(sc)
End Function
```

Compile y pruebe el código

Para probar este código, compile la solución `OutlineRegionTest` y ejecútela en la instancia experimental.

Para compilar y probar la solución `OutlineRegionTest`

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Crear un archivo de texto Escriba algún texto que incluya tanto los corchetes de apertura como los corchetes de cierre.

```
[
    Hello
]
```

4. Debe haber una región de esquematización que incluya ambos corchetes. Debería poder hacer clic en el signo menos a la izquierda del corchete abierto para contraer la región de esquematización. Cuando la región está contraída, el símbolo de puntos suspensivos (...) debe aparecer a la izquierda de la región contraída y debe aparecer una ventana emergente que contiene el **texto de desplazamiento** del texto al mover el puntero sobre los puntos suspensivos.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Mostrar llaves coincidentes

06/04/2020 • 15 minutes to read • [Edit Online](#)

Implemente características basadas en lenguaje, como la coincidencia de llaves definiendo las llaves que desea que coincidan y agregando una etiqueta de marcador de texto a las llaves coincidentes cuando el intercalador está en una de las llaves. Puede definir llaves en el contexto de un idioma, definir su propia extensión de nombre de archivo y tipo de contenido, y aplicar las etiquetas solo a ese tipo o aplicar las etiquetas a un tipo de contenido existente (por ejemplo, "texto"). En el siguiente tutorial se muestra cómo aplicar etiquetas de coincidencia de llaves al tipo de contenido "text".

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un proyecto de Managed Extensibility Framework (MEF)

Para crear un nuevo proyecto de MEF

1. Cree un proyecto de clasificador de editor. Asigne a la solución el nombre `BraceMatchingTest`.
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.

Implementar un tagger de coincidencia de llaves

Para obtener un efecto de resaltado de llaves similar al que se usa en [TextMarkerTag](#) Visual Studio, puede implementar un etiquetador de tipo `ITagger`. El código siguiente muestra cómo definir el etiquetador para pares de llaves en cualquier nivel de anidamiento. En este ejemplo, los pares `{}` de llaves de `[]` y se definen en el constructor de etiqueta, pero en una implementación de lenguaje completo, los pares de llaves relevantes se definirían en la especificación de lenguaje.

Para implementar un tagger de coincidencia de llaves

1. Agregue un archivo de clase y asímócelo `BraceMatching`.
2. Importe los siguientes espacios de nombres.

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Utilities;
```

```
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Text.Tagging
Imports Microsoft.VisualStudio.Utilities
```

3. Defina una `BraceMatchingTagger` clase que `ITagger<T>` herede de tipo `TextMarkerTag`.

```
internal class BraceMatchingTagger : ITagger<TextMarkerTag>
```

```
Friend Class BraceMatchingTagger
    Implements ITagger(Of TextMarkerTag)
```

4. Agregue propiedades para la vista de texto, el búfer de origen, el punto de instantánea actual y también un conjunto de pares de llaves.

```
ITextView View { get; set; }
ITextBuffer SourceBuffer { get; set; }
SnapshotPoint? CurrentChar { get; set; }
private Dictionary<char, char> m_braceList;
```

```
Private _View As ITextView
Private Property View() As ITextView
    Get
        Return _View
    End Get
    Set(ByVal value As ITextView)
        _View = value
    End Set
End Property
Private _SourceBuffer As ITextBuffer
Private Property SourceBuffer() As ITextBuffer
    Get
        Return _SourceBuffer
    End Get
    Set(ByVal value As ITextBuffer)
        _SourceBuffer = value
    End Set
End Property
Private _CurrentChar As System.Nullable(Of SnapshotPoint)
Private Property CurrentChar() As System.Nullable(Of SnapshotPoint)
    Get
        Return _CurrentChar
    End Get
    Set(ByVal value As System.Nullable(Of SnapshotPoint))
        _CurrentChar = value
    End Set
End Property
Private m_braceList As Dictionary(Of Char, Char)
```

5. En el constructor de etiqueta, establezca las propiedades `PositionChanged` `LayoutChanged` y suscríbase a los eventos de cambio de vista y . En este ejemplo, para fines ilustrativos, los pares coincidentes también se definen en el constructor.


```

internal BraceMatchingTagger(ITextView view, ITextBuffer sourceBuffer)
{
    //here the keys are the open braces, and the values are the close braces
    m_braceList = new Dictionary<char, char>();
    m_braceList.Add('{', '}');
    m_braceList.Add('[', ']');
    m_braceList.Add('(', ')');
    this.View = view;
    this.SourceBuffer = sourceBuffer;
    this.CurrentChar = null;

    this.View.Caret.PositionChanged += CaretPositionChanged;
    this.View.LayoutChanged += ViewLayoutChanged;
}

```

```

Friend Sub New(ByVal view As ITextView, ByVal sourceBuffer As ITextBuffer)
    'here the keys are the open braces, and the values are the close braces
    m_braceList = New Dictionary(Of Char, Char)()
    m_braceList.Add("{", "}")
    m_braceList.Add("[", "]")
    m_braceList.Add("(", ")")
    Me.View = view
    Me.SourceBuffer = sourceBuffer
    Me.CurrentChar = Nothing

    AddHandler Me.View.Caret.PositionChanged, AddressOf Me.CaretPositionChanged
    AddHandler Me.View.LayoutChanged, AddressOf Me.ViewLayoutChanged
End Sub

```

6. Como parte `ITagger<T>` de la implementación, declare un `TagsChanged` eventos.

```

public event EventHandler<SnapshotSpanEventArgs> TagsChanged;

```

```

Public Event TagsChanged As EventHandler(Of SnapshotSpanEventArgs) _
    Implements ITagger(Of TextMarkerTag).TagsChanged

```

7. Los controladores de eventos actualizan la `CurrentChar` posición actual del intercalador de la propiedad y generan el `TagsChanged` eventos.

```

void ViewLayoutChanged(object sender, TextViewLayoutChangedEventArgs e)
{
    if (e.NewSnapshot != e.OldSnapshot) //make sure that there has really been a change
    {
        UpdateAtCaretPosition(View.Caret.Position);
    }
}

void CaretPositionChanged(object sender, CaretPositionChangedEventArgs e)
{
    UpdateAtCaretPosition(e.NewPosition);
}

void UpdateAtCaretPosition(CaretPosition caretPosition)
{
    CurrentChar = caretPosition.Point.GetPoint(SourceBuffer, caretPosition.Affinity);

    if (!CurrentChar.HasValue)
        return;

    var tempEvent = TagsChanged;
    if (tempEvent != null)
        tempEvent(this, new SnapshotSpanEventArgs(new SnapshotSpan(SourceBuffer.CurrentSnapshot, 0,
            SourceBuffer.CurrentSnapshot.Length)));
}

```

```

Private Sub ViewLayoutChanged(ByVal sender As Object, ByVal e As TextViewLayoutChangedEventArgs)
    If e.NewSnapshot IsNot e.OldSnapshot Then
        'make sure that there has really been a change
        UpdateAtCaretPosition(View.Caret.Position)
    End If
End Sub

Private Sub CaretPositionChanged(ByVal sender As Object, ByVal e As CaretPositionChangedEventArgs)
    UpdateAtCaretPosition(e.NewPosition)
End Sub

Private Sub UpdateAtCaretPosition(ByVal caretPosition As CaretPosition)
    CurrentChar = caretPosition.Point.GetPoint(SourceBuffer, caretPosition.Affinity)

    If Not CurrentChar.HasValue Then
        Exit Sub
    End If

    RaiseEvent TagsChanged(Me, New SnapshotSpanEventArgs(New SnapshotSpan(SourceBuffer.CurrentSnapshot,
    0, SourceBuffer.CurrentSnapshot.Length)))
End Sub

```

8. Implemente [GetTags](#) el método para que coincida con las llaves cuando el carácter actual es una llave abierta o cuando el carácter anterior es una llave estrecha, como en Visual Studio. Cuando se encuentra la coincidencia, este método crea dos etiquetas, una para la llave abierta y otra para la llave de cierre.

```

public IEnumerable<ITagSpan<TextMarkerTag>> GetTags(NormalizedSnapshotSpanCollection spans)
{
    if (spans.Count == 0)    //there is no content in the buffer
        yield break;

    //don't do anything if the current SnapshotPoint is not initialized or at the end of the buffer
    if (!CurrentChar.HasValue || CurrentChar.Value.Position >= CurrentChar.Value.Snapshot.Length)
        yield break;

    //hold on to a snapshot of the current character
    SnapshotPoint currentChar = CurrentChar.Value;

    //if the requested snapshot isn't the same as the one the brace is on, translate our spans to the
    expected snapshot
    if (spans[0].Snapshot != currentChar.Snapshot)
    {
        currentChar = currentChar.TranslateTo(spans[0].Snapshot, PointTrackingMode.Positive);
    }

    //get the current char and the previous char
    char currentText = currentChar.GetChar();
    SnapshotPoint lastChar = currentChar == 0 ? currentChar : currentChar - 1; //if currentChar is 0
    (beginning of buffer), don't move it back
    char lastText = lastChar.GetChar();
    SnapshotSpan pairSpan = new SnapshotSpan();

    if (m_braceList.ContainsKey(currentText))    //the key is the open brace
    {
        char closeChar;
        m_braceList.TryGetValue(currentText, out closeChar);
        if (BraceMatchingTagger.FindMatchingCloseChar(currentChar, currentText, closeChar,
            View.TextViewLines.Count, out pairSpan) == true)
        {
            yield return new TagSpan<TextMarkerTag>(new SnapshotSpan(currentChar, 1), new
            TextMarkerTag("blue"));
            yield return new TagSpan<TextMarkerTag>(pairSpan, new TextMarkerTag("blue"));
        }
    }
    else if (m_braceList.ContainsValue(lastText))    //the value is the close brace, which is the
    *previous* character
    {
        var open = from n in m_braceList
                    where n.Value.Equals(lastText)
                    select n.Key;
        if (BraceMatchingTagger.FindMatchingOpenChar(lastChar, (char)open.ElementAt<char>(0), lastText,
            View.TextViewLines.Count, out pairSpan) == true)
        {
            yield return new TagSpan<TextMarkerTag>(new SnapshotSpan(lastChar, 1), new
            TextMarkerTag("blue"));
            yield return new TagSpan<TextMarkerTag>(pairSpan, new TextMarkerTag("blue"));
        }
    }
}

```

```

Public Function GetTags(ByVal spans As NormalizedSnapshotSpanCollection) As IEnumerable(Of ITagSpan(Of
TextMarkerTag)) Implements ITagger(Of Microsoft.VisualStudio.Text.Tagging.TextMarkerTag).GetTags
    If spans.Count = 0 Then
        'there is no content in the buffer
        Exit Function
    End If

    'don't do anything if the current SnapshotPoint is not initialized or at the end of the buffer
    If Not CurrentChar.HasValue OrElse CurrentChar.Value.Position >= CurrentChar.Value.Snapshot.Length
Then
        Exit Function
    End If

    'hold on to a snapshot of the current character
    Dim currentChar__1 As SnapshotPoint = CurrentChar.Value

    'if the requested snapshot isn't the same as the one the brace is on, translate our spans to the
    expected snapshot
    If spans(0).Snapshot IsNot currentChar__1.Snapshot Then
        currentChar__1 = currentChar__1.TranslateTo(spans(0).Snapshot, PointTrackingMode.Positive)
    End If

    'get the current char and the previous char
    Dim currentText As Char = currentChar__1.GetChar()
    Dim lastChar As SnapshotPoint = If(CInt(currentChar__1) = 0, currentChar__1, currentChar__1 - 1)
    'if currentChar is 0 (beginning of buffer), don't move it back
    Dim lastText As Char = lastChar.GetChar()
    Dim pairSpan As New SnapshotSpan()

    If m_braceList.ContainsKey(currentText) Then
        'the key is the open brace
        Dim closeChar As Char
        m_braceList.TryGetValue(currentText, closeChar)
        If BraceMatchingTagger.FindMatchingCloseChar(currentChar__1, currentText, closeChar,
View.TextViewLines.Count, pairSpan) = True Then
            Exit Function
        End If
    ElseIf m_braceList.ContainsValue(lastText) Then
        'the value is the close brace, which is the *previous* character
        Dim open = From n In m_braceList _
            Where n.Value.Equals(lastText) _
            Select n.Key
        If BraceMatchingTagger.FindMatchingOpenChar(lastChar, CChar(open.ElementAt(0)), lastText,
View.TextViewLines.Count, pairSpan) = True Then
            Exit Function
        End If
    End If
End Function

```

9. Los siguientes métodos privados encuentran la llave coincidente en cualquier nivel de anidamiento. El primer método busca el carácter close que coincide con el carácter abierto:

```

private static bool FindMatchingCloseChar(SnapshotPoint startPoint, char open, char close, int
maxLines, out SnapshotSpan pairSpan)
{
    pairSpan = new SnapshotSpan(startPoint.Snapshot, 1, 1);
    ITextSnapshotLine line = startPoint.GetContainingLine();
    string lineText = line.GetText();
    int lineNumber = line.LineNumber;
    int offset = startPoint.Position - line.Start.Position + 1;

    int stopLineNumber = startPoint.Snapshot.LineCount - 1;
    if (maxLines > 0)
        stopLineNumber = Math.Min(stopLineNumber, lineNumber + maxLines);

    int openCount = 0;
    while (true)
    {
        //walk the entire line
        while (offset < line.Length)
        {
            char currentChar = lineText[offset];
            if (currentChar == close) //found the close character
            {
                if (openCount > 0)
                {
                    openCount--;
                }
                else //found the matching close
                {
                    pairSpan = new SnapshotSpan(startPoint.Snapshot, line.Start + offset, 1);
                    return true;
                }
            }
            else if (currentChar == open) // this is another open
            {
                openCount++;
            }
            offset++;
        }

        //move on to the next line
        if (++lineNumber > stopLineNumber)
            break;

        line = line.Snapshot.GetLineFromLineNumber(lineNumber);
        lineText = line.GetText();
        offset = 0;
    }

    return false;
}

```

```

Private Shared Function FindMatchingCloseChar(ByVal startPoint As SnapshotPoint, ByVal open As Char,
ByVal close As Char, ByVal maxLines As Integer, ByRef pairSpan As SnapshotSpan) As Boolean
    pairSpan = New SnapshotSpan(startPoint.Snapshot, 1, 1)
    Dim line As ITextSnapshotLine = startPoint.GetContainingLine()
    Dim lineText As String = line.GetText()
    Dim lineNumber As Integer = line.LineNumber
    Dim offset As Integer = startPoint.Position - line.Start.Position + 1

    Dim stopLineNumber As Integer = startPoint.Snapshot.LineCount - 1
    If maxLines > 0 Then
        stopLineNumber = Math.Min(stopLineNumber, lineNumber + maxLines)
    End If

    Dim openCount As Integer = 0
    While True
        'walk the entire line
        While offset < line.Length
            Dim currentChar As Char = lineText(offset)
            If currentChar = close Then
                'found the close character
                If openCount > 0 Then
                    openCount -= 1
                Else
                    'found the matching close
                    pairSpan = New SnapshotSpan(startPoint.Snapshot, line.Start + offset, 1)
                    Return True
                End If
            ElseIf currentChar = open Then
                ' this is another open
                openCount += 1
            End If
            offset += 1
        End While

        'move on to the next line
        If System.Threading.Interlocked.Increment(lineNumber) > stopLineNumber Then
            Exit While
        End If

        line = line.Snapshot.GetLineFromLineNumber(lineNumber)
        lineText = line.GetText()
        offset = 0
    End While

    Return False
End Function

```

10. El siguiente método auxiliar busca el carácter abierto que coincide con un carácter cercano:

```

private static bool FindMatchingOpenChar(SnapshotPoint startPoint, char open, char close, int maxLines,
out SnapshotSpan pairSpan)
{
    pairSpan = new SnapshotSpan(startPoint, startPoint);

    ITextSnapshotLine line = startPoint.GetContainingLine();

    int lineNumber = line.LineNumber;
    int offset = startPoint - line.Start - 1; //move the offset to the character before this one

    //if the offset is negative, move to the previous line
    if (offset < 0)
    {
        line = line.Snapshot.GetLineFromLineNumber(--lineNumber);
        offset = line.Length - 1;
    }

    string lineText = line.GetText();

    int stopLineNumber = 0;
    if (maxLines > 0)
        stopLineNumber = Math.Max(stopLineNumber, lineNumber - maxLines);

    int closeCount = 0;

    while (true)
    {
        // Walk the entire line
        while (offset >= 0)
        {
            char currentChar = lineText[offset];

            if (currentChar == open)
            {
                if (closeCount > 0)
                {
                    closeCount--;
                }
                else // We've found the open character
                {
                    pairSpan = new SnapshotSpan(line.Start + offset, 1); //we just want the character
                    return true;
                }
            }
            else if (currentChar == close)
            {
                closeCount++;
            }
            offset--;
        }

        // Move to the previous line
        if (--lineNumber < stopLineNumber)
            break;

        line = line.Snapshot.GetLineFromLineNumber(lineNumber);
        lineText = line.GetText();
        offset = line.Length - 1;
    }
    return false;
}

```

```

Private Shared Function FindMatchingOpenChar(ByVal startPoint As SnapshotPoint, ByVal open As Char,
ByVal close As Char, ByVal maxLines As Integer, ByRef pairSpan As SnapshotSpan) As Boolean
    pairSpan = New SnapshotSpan(startPoint, startPoint)

    Dim line As ITextSnapshotLine = startPoint.GetContainingLine()

    Dim lineNumber As Integer = line.LineNumber
    Dim offset As Integer = startPoint - line.Start - 1
    'move the offset to the character before this one
    'if the offset is negative, move to the previous line
    If offset < 0 Then
        line = line.Snapshot.GetLineFromLineNumber(System.Threading.Interlocked.Decrement(lineNumber))
        offset = line.Length - 1
    End If

    Dim lineText As String = line.GetText()

    Dim stopLineNumber As Integer = 0
    If maxLines > 0 Then
        stopLineNumber = Math.Max(stopLineNumber, lineNumber - maxLines)
    End If

    Dim closeCount As Integer = 0

    While True
        ' Walk the entire line
        While offset >= 0
            Dim currentChar As Char = lineText(offset)

            If currentChar = open Then
                If closeCount > 0 Then
                    closeCount -= 1
                Else
                    ' We've found the open character
                    pairSpan = New SnapshotSpan(line.Start + offset, 1)
                    'we just want the character itself
                    Return True
                End If
            ElseIf currentChar = close Then
                closeCount += 1
            End If
            offset -= 1
        End While

        ' Move to the previous line
        If System.Threading.Interlocked.Decrement(lineNumber) < stopLineNumber Then
            Exit While
        End If

        line = line.Snapshot.GetLineFromLineNumber(lineNumber)
        lineText = line.GetText()
        offset = line.Length - 1
    End While
    Return False
End Function

```

Implementar un proveedor de etiquetado de coincidencia de llaves

Además de implementar un tagger, también debe implementar y exportar un proveedor de etiquetas. En este caso, el tipo de contenido del proveedor es "texto". Por lo tanto, la coincidencia de llaves aparecerá en todos los tipos de archivos de texto, pero una implementación más completa aplica la coincidencia de llaves solo a un tipo de contenido específico.

Para implementar un proveedor de etiquetado de coincidencia de llaves

1. Declare un proveedor de etiqueta [IViewTaggerProvider](#) que herede de [IViewTaggerProvider](#), asílo sea [ContentTypeAttribute](#) [BraceMatchingTaggerProvider](#) [TagTypeAttribute](#) [TextMarkerTag](#) expórtelo con un de "texto" y un archivo de [.cs](#).

```
[Export(typeof(IViewTaggerProvider))]  
[ContentType("text")]  
[TagType(typeof(TextMarkerTag))]  
internal class BraceMatchingTaggerProvider : IViewTaggerProvider
```

```
<Export(GetType(IViewTaggerProvider))> _  
<ContentType("text")> _  
<TagType(GetType(TextMarkerTag))> _  
Friend Class BraceMatchingTaggerProvider  
    Implements IViewTaggerProvider
```

2. Implemente [CreateTagger](#) el método para crear una instancia de un [BraceMatchingTagger](#).

```
public ITagger<T> CreateTagger<T>(ITextView textView, ITextBuffer buffer) where T : ITag  
{  
    if (textView == null)  
        return null;  
  
    //provide highlighting only on the top-level buffer  
    if (textView.TextBuffer != buffer)  
        return null;  
  
    return new BraceMatchingTagger(textView, buffer) as ITagger<T>;  
}
```

```
Public Function CreateTagger(Of T As ITag)(ByVal textView As ITextView, ByVal buffer As ITextBuffer) As  
ITagger(Of T) Implements IViewTaggerProvider.CreateTagger  
    If textView Is Nothing Then  
        Return Nothing  
    End If  
  
    'provide highlighting only on the top-level buffer  
    If textView.TextBuffer IsNot buffer Then  
        Return Nothing  
    End If  
  
    Return TryCast(New BraceMatchingTagger(textView, buffer), ITagger(Of T))  
End Function
```

Compile y pruebe el código

Para probar este código, compile la solución [BraceMatchingTest](#) y ejecútela en la instancia experimental.

Para crear y probar la solución [BraceMatchingTest](#)

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Cree un archivo de texto y escriba texto que incluya llaves coincidentes.

```
hello {  
goodbye}  
  
{}  
  
{hello}
```

4. Al colocar el intercalador antes de una llave abierta, se deben resaltar tanto esa llave como la llave de cierre correspondiente. Al colocar el cursor justo después de la llave de cierre, se deben resaltar tanto esa llave como la llave abierta coincidente.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Mostrar información sobre herramientas quickInfo

06/04/2020 • 14 minutes to read • [Edit Online](#)

QuickInfo es una característica de IntelliSense que muestra las firmas y descripciones del método cuando un usuario mueve el puntero sobre un nombre de método. Puede implementar características basadas en lenguaje como QuickInfo definiendo los identificadores para los que desea proporcionar descripciones de QuickInfo y, a continuación, creando una información sobre herramientas en la que mostrar el contenido. Puede definir QuickInfo en el contexto de un servicio de lenguaje, o puede definir su propia extensión de nombre de archivo y tipo de contenido y mostrar QuickInfo solo para ese tipo, o puede mostrar QuickInfo para un tipo de contenido existente (por ejemplo, "texto"). En este tutorial se muestra cómo mostrar QuickInfo para el tipo de contenido "texto".

El ejemplo QuickInfo de este tutorial muestra la información sobre herramientas cuando un usuario mueve el puntero sobre un nombre de método. Este diseño requiere que usted implemente estas cuatro interfaces:

- interfaz de fuente
- interfaz del proveedor de fuentes
- interfaz del controlador
- interfaz del proveedor del controlador

Los proveedores de origen y controlador son componentes de Managed Extensibility Framework (MEF) y son responsables [ITextBufferFactoryService](#) de exportar las clases [IQuickInfoBroker](#) de origen y controlador e importar servicios y intermediarios como el [TextBufferFactoryService](#), que crea el búfer de texto de información sobre herramientas y el [QuickInfoBroker](#), que desencadena la sesión QuickInfo.

En este ejemplo, el origen QuickInfo usa una lista codificada de forma rígida de nombres de método y descripciones, pero en las implementaciones completas, el servicio de lenguaje y la documentación de idioma son responsables de proporcionar ese contenido.

Prerrequisitos

A partir de Visual Studio 2015, no es necesario instalar el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto MEF

Para crear un nuevo proyecto de MEF

1. Cree un proyecto de VSIX de C#. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C# Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `QuickInfoTest` nombre a la solución .
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.

Implementar el origen QuickInfo

El origen QuickInfo es responsable de recopilar el conjunto de identificadores y sus descripciones y agregar el contenido al búfer de texto de información sobre herramientas cuando se encuentra uno de los identificadores. En este ejemplo, los identificadores y sus descripciones se agregan en el constructor de origen.

Para implementar el origen QuickInfo

1. Agregue un archivo de clase y asígnele el nombre `TestQuickInfoSource`.
2. Agregue una referencia a *Microsoft.VisualStudio.Language.Intellisense*.
3. Agregue las importaciones siguientes.

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.Collections.ObjectModel
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Language.Intellisense
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Text.Operations
Imports Microsoft.VisualStudio.Text.Tagging
Imports Microsoft.VisualStudio.Utilities
```

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Language.Intellisense;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Operations;
using Microsoft.VisualStudio.Text.Tagging;
using Microsoft.VisualStudio.Utilities;
```

4. Declare una clase `IQuickInfoSource` que implemente `TestQuickInfoSource` y asímóctela.

```
Friend Class TestQuickInfoSource
    Implements IQuickInfoSource
```

```
internal class TestQuickInfoSource : IQuickInfoSource
```

5. Agregue campos para el proveedor de origen QuickInfo, el búfer de texto y un conjunto de nombres de método y firmas de método. En este ejemplo, los nombres de método `TestQuickInfoSource` y las firmas se inicializan en el constructor.

```
Private m_provider As TestQuickInfoSourceProvider
Private m_subjectBuffer As ITextBuffer
Private m_dictionary As Dictionary(Of String, String)
```

```
private TestQuickInfoSourceProvider m_provider;
private ITextBuffer m_subjectBuffer;
private Dictionary<string, string> m_dictionary;
```

6. Agregue un constructor que establezca el proveedor de origen QuickInfo y el búfer de texto y rellene el conjunto de nombres de método y las firmas y descripciones del método.

```

Public Sub New(ByVal provider As TestQuickInfoSourceProvider, ByVal subjectBuffer As ITextBuffer)
    m_provider = provider
    m_subjectBuffer = subjectBuffer

    'these are the method names and their descriptions
    m_dictionary = New Dictionary(Of String, String)()
    m_dictionary.Add("add", "int add(int firstInt, int secondInt)" & vbCrLf & "Adds one integer to
another.")
    m_dictionary.Add("subtract", "int subtract(int firstInt, int secondInt)" & vbCrLf & "Subtracts one
integer from another.")
    m_dictionary.Add("multiply", "int multiply(int firstInt, int secondInt)" & vbCrLf & "Multiplies one
integer by another.")
    m_dictionary.Add("divide", "int divide(int firstInt, int secondInt)" & vbCrLf & "Divides one integer
by another.")
End Sub

```

```

public TestQuickInfoSource(TestQuickInfoSourceProvider provider, ITextBuffer subjectBuffer)
{
    m_provider = provider;
    m_subjectBuffer = subjectBuffer;

    //these are the method names and their descriptions
    m_dictionary = new Dictionary<string, string>();
    m_dictionary.Add("add", "int add(int firstInt, int secondInt)\nAdds one integer to another.");
    m_dictionary.Add("subtract", "int subtract(int firstInt, int secondInt)\nSubtracts one integer from
another.");
    m_dictionary.Add("multiply", "int multiply(int firstInt, int secondInt)\nMultiplies one integer by
another.");
    m_dictionary.Add("divide", "int divide(int firstInt, int secondInt)\nDivides one integer by
another.");
}

```

7. Implemente el método [AugmentQuickInfoSession](#). En este ejemplo, el método busca la palabra actual, o la palabra anterior si el cursor está al final de una línea o un búfer de texto. Si la palabra es uno de los nombres de método, la descripción de ese nombre de método se agrega al contenido de QuickInfo.

```

Public Sub AugmentQuickInfoSession(ByVal session As IQuickInfoSession, ByVal qiContent As IList(Of
Object), ByRef applicableToSpan As ITrackingSpan) Implements IQuickInfoSource.AugmentQuickInfoSession
    ' Map the trigger point down to our buffer.
    Dim subjectTriggerPoint As System.Nullable(Of SnapshotPoint) =
session.GetTriggerPoint(m_subjectBuffer.CurrentSnapshot)
    If Not subjectTriggerPoint.HasValue Then
        applicableToSpan = Nothing
        Exit Sub
    End If

    Dim currentSnapshot As ITextSnapshot = subjectTriggerPoint.Value.Snapshot
    Dim querySpan As New SnapshotSpan(subjectTriggerPoint.Value, 0)

    'look for occurrences of our QuickInfo words in the span
    Dim navigator As ITextStructureNavigator =
m_provider.NavigatorService.GetTextStructureNavigator(m_subjectBuffer)
    Dim extent As TextExtent = navigator.GetExtentOfWord(subjectTriggerPoint.Value)
    Dim searchText As String = extent.Span.GetText()

    For Each key As String In m_dictionary.Keys
        Dim foundIndex As Integer = searchText.IndexOf(key, StringComparison.CurrentCultureIgnoreCase)
        If foundIndex > -1 Then
            'applicableToSpan =
currentSnapshot.CreateTrackingSpan(querySpan.Start.Add(foundIndex).Position, 9,
SpanTrackingMode.EdgeInclusive)

            applicableToSpan = currentSnapshot.CreateTrackingSpan(extent.Span.Start + foundIndex,
key.Length, SpanTrackingMode.EdgeInclusive)
            Dim value As String = ""
            m_dictionary.TryGetValue(key, value)
            If value IsNot Nothing Then
                qiContent.Add(value)
            Else
                qiContent.Add("")
            End If

            Exit Sub
        End If
    Next

    applicableToSpan = Nothing
End Sub

```

```

public void AugmentQuickInfoSession(IQuickInfoSession session, IList<object> qiContent, out
ITrackingSpan applicableToSpan)
{
    // Map the trigger point down to our buffer.
    SnapshotPoint? subjectTriggerPoint = session.GetTriggerPoint(m_subjectBuffer.CurrentSnapshot);
    if (!subjectTriggerPoint.HasValue)
    {
        applicableToSpan = null;
        return;
    }

    ITextSnapshot currentSnapshot = subjectTriggerPoint.Value.Snapshot;
    SnapshotSpan querySpan = new SnapshotSpan(subjectTriggerPoint.Value, 0);

    //look for occurrences of our QuickInfo words in the span
    ITextStructureNavigator navigator =
m_provider.NavigatorService.GetTextStructureNavigator(m_subjectBuffer);
    TextExtent extent = navigator.GetExtentOfWord(subjectTriggerPoint.Value);
    string searchText = extent.Span.GetText();

    foreach (string key in m_dictionary.Keys)
    {
        int foundIndex = searchText.IndexOf(key, StringComparison.CurrentCultureIgnoreCase);
        if (foundIndex > -1)
        {
            applicableToSpan = currentSnapshot.CreateTrackingSpan
            (
                //querySpan.Start.Add(foundIndex).Position, 9, SpanTrackingMode.EdgeInclusive
                extent.Span.Start + foundIndex, key.Length,
SpanTrackingMode.EdgeInclusive
            );

            string value;
            m_dictionary.TryGetValue(key, out value);
            if (value != null)
                qiContent.Add(value);
            else
                qiContent.Add("");

            return;
        }
    }

    applicableToSpan = null;
}

```

8. También debe implementar un método Dispose(), ya que [IQuickInfoSource](#) [IDisposable](#) implementa:

```

Private m_isDisposed As Boolean
Public Sub Dispose() Implements IDisposable.Dispose
    If Not m_isDisposed Then
        GC.SuppressFinalize(Me)
        m_isDisposed = True
    End If
End Sub

```

```

private bool m_isDisposed;
public void Dispose()
{
    if (!m_isDisposed)
    {
        GC.SuppressFinalize(this);
        m_isDisposed = true;
    }
}

```

Implementar un proveedor de origen QuickInfo

El proveedor del origen QuickInfo sirve principalmente para exportarse a sí mismo como una parte de componente MEF y crear instancias del origen QuickInfo. Debido a que es una pieza de componente MEF, puede importar otras piezas de componentes MEF.

Para implementar un proveedor de origen QuickInfo

1. Declare un proveedor de `TestQuickInfoSourceProvider` código `IQuickInfoSourceProvider` fuente QuickInfo denominado `NameAttribute` que implemente y expórtelo con un de "ToolTip QuickInfo Source", un `OrderAttribute` de Before-"default" y un `ContentTypeAttribute` de "text".

```

<Export(GetType(IQuickInfoSourceProvider))> _
<Name("ToolTip QuickInfo Source")> _
<Order(Before:= "Default Quick Info Presenter")> _
<ContentType("text")> _
Friend Class TestQuickInfoSourceProvider
    Implements IQuickInfoSourceProvider

```

```

[Export(typeof(IQuickInfoSourceProvider))]
[Name("ToolTip QuickInfo Source")]
[Order(Before = "Default Quick Info Presenter")]
[ContentType("text")]
internal class TestQuickInfoSourceProvider : IQuickInfoSourceProvider

```

2. Importe dos servicios `ITextStructureNavigatorSelectorService` `ITextBufferFactoryService` de editor `TestQuickInfoSourceProvider` y , como propiedades de .

```

Private _NavigatorService As ITextStructureNavigatorSelectorService
<Import()> _
Friend Property NavigatorService() As ITextStructureNavigatorSelectorService
    Get
        Return _NavigatorService
    End Get
    Set(ByVal value As ITextStructureNavigatorSelectorService)
        _NavigatorService = value
    End Set
End Property

Private _TextBufferFactoryService As ITextBufferFactoryService
<Import()> _
Friend Property TextBufferFactoryService() As ITextBufferFactoryService
    Get
        Return _TextBufferFactoryService
    End Get
    Set(ByVal value As ITextBufferFactoryService)
        _TextBufferFactoryService = value
    End Set
End Property

```



```
[Import]
internal ITextStructureNavigatorSelectorService NavigatorService { get; set; }

[Import]
internal ITextBufferFactoryService TextBufferFactoryService { get; set; }
```

3. Implementar [TryCreateQuickInfoSource](#) para devolver `TestQuickInfoSource` un nuevo archivo .

```
Public Function TryCreateQuickInfoSource(ByVal textBuffer As ITextBuffer) As IQuickInfoSource
Implements IQuickInfoSourceProvider.TryCreateQuickInfoSource
    Return New TestQuickInfoSource(Me, textBuffer)
End Function
```

```
public IQuickInfoSource TryCreateQuickInfoSource(ITextBuffer textBuffer)
{
    return new TestQuickInfoSource(this, textBuffer);
}
```

Implementar un controlador QuickInfo

Los controladores QuickInfo determinan cuándo se muestra QuickInfo. En este ejemplo, QuickInfo aparece cuando el puntero está sobre una palabra que corresponde a uno de los nombres de método. El quickInfo controlador implementa un controlador de eventos de desplazamiento del mouse que desencadena una sesión QuickInfo.

Para implementar un controlador QuickInfo

1. Declare una clase [IntellisenseController](#) que implemente `TestQuickInfoController` y asímóctela .

```
Friend Class TestQuickInfoController
Implements IIntellisenseController
```

```
internal class TestQuickInfoController : IIntellisenseController
```

2. Agregue campos privados para la vista de texto, los búferes de texto representados en la vista de texto, la sesión QuickInfo y el proveedor de controlador QuickInfo.

```
Private m_textView As ITextView
Private m_subjectBuffers As IList(Of ITextBuffer)
Private m_provider As TestQuickInfoControllerProvider
Private m_session As IQuickInfoSession
```

```
private ITextView m_textView;
private IList<ITextBuffer> m_subjectBuffers;
private TestQuickInfoControllerProvider m_provider;
private IQuickInfoSession m_session;
```

3. Agregue un constructor que establezca los campos y agregue el controlador de eventos de desplazamiento del mouse.

```

Friend Sub New(ByVal textView As ITextView, ByVal subjectBuffers As IList(Of ITextBuffer), ByVal
provider As TestQuickInfoControllerProvider)
    m_textView = textView
    m_subjectBuffers = subjectBuffers
    m_provider = provider

    AddHandler m_textView.MouseHover, AddressOf Me.OnTextViewMouseHover
End Sub

```

```

internal TestQuickInfoController(ITextView textView, IList<ITextBuffer> subjectBuffers,
TestQuickInfoControllerProvider provider)
{
    m_textView = textView;
    m_subjectBuffers = subjectBuffers;
    m_provider = provider;

    m_textView.MouseHover += this.OnTextViewMouseHover;
}

```

4. Agregue el controlador de eventos de desplazamiento del mouse que desencadena la sesión QuickInfo.

```

Private Sub OnTextViewMouseHover(ByVal sender As Object, ByVal e As MouseEventArgs)
    'find the mouse position by mapping down to the subject buffer
    Dim point As System.Nullable(Of SnapshotPoint) = m_textView.BufferGraph.MapDownToFirstMatch(New
SnapshotPoint(m_textView.TextSnapshot, e.Position), PointTrackingMode.Positive, Function(snapshot)
m_subjectBuffers.Contains(snapshot.TextBuffer), PositionAffinity.Predecessor)

    If point IsNot Nothing Then
        Dim triggerPoint As ITrackingPoint =
point.Value.Snapshot.CreateTrackingPoint(point.Value.Position, PointTrackingMode.Positive)

        If Not m_provider.QuickInfoBroker.IsQuickInfoActive(m_textView) Then
            m_session = m_provider.QuickInfoBroker.TriggerQuickInfo(m_textView, triggerPoint, True)
        End If
    End If
End Sub

```

```

private void OnTextViewMouseHover(object sender, MouseEventArgs e)
{
    //find the mouse position by mapping down to the subject buffer
    SnapshotPoint? point = m_textView.BufferGraph.MapDownToFirstMatch
        (new SnapshotPoint(m_textView.TextSnapshot, e.Position),
        PointTrackingMode.Positive,
        snapshot => m_subjectBuffers.Contains(snapshot.TextBuffer),
        PositionAffinity.Predecessor);

    if (point != null)
    {
        ITrackingPoint triggerPoint = point.Value.Snapshot.CreateTrackingPoint(point.Value.Position,
            PointTrackingMode.Positive);

        if (!m_provider.QuickInfoBroker.IsQuickInfoActive(m_textView))
        {
            m_session = m_provider.QuickInfoBroker.TriggerQuickInfo(m_textView, triggerPoint, true);
        }
    }
}

```

5. Implemente [Detach](#) el método para que quite el controlador de eventos de desplazamiento del mouse cuando el controlador se separa de la vista de texto.

```
Public Sub Detach(ByVal textView As ITextView) Implements IIntellisenseController.Detach
    If m_textView Is textView Then
        AddHandler m_textView.MouseHover, AddressOf Me.OnTextViewMouseHover
        m_textView = Nothing
    End If
End Sub
```

```
public void Detach(ITextView textView)
{
    if (m_textView == textView)
    {
        m_textView.MouseHover -= this.OnTextViewMouseHover;
        m_textView = null;
    }
}
```

6. Implemente [ConnectSubjectBuffer](#) el método [DisconnectSubjectBuffer](#) y el método como métodos vacíos para este ejemplo.

```
Public Sub ConnectSubjectBuffer(ByVal subjectBuffer As ITextBuffer) Implements
IIntellisenseController.ConnectSubjectBuffer

End Sub

Public Sub DisconnectSubjectBuffer(ByVal subjectBuffer As ITextBuffer) Implements
IIntellisenseController.DisconnectSubjectBuffer

End Sub
```

```
public void ConnectSubjectBuffer(ITextBuffer subjectBuffer)
{
}

public void DisconnectSubjectBuffer(ITextBuffer subjectBuffer)
{
}
```

Implementación del proveedor de controlador quick QuickInfo

El proveedor del controlador QuickInfo sirve principalmente para exportarse a sí mismo como una parte de componente MEF y crear instancias del controlador QuickInfo. Debido a que es una pieza de componente MEF, puede importar otras piezas de componentes MEF.

Para implementar el proveedor de controlador QuickInfo

1. Declare una `TestQuickInfoControllerProvider` clase denominada [IIntellisenseControllerProvider](#) que implemente y [NameAttribute](#) expórtela con un [ContentTypeAttribute](#) de "ToolTip QuickInfo Controller" y un de "text".

```
<Export(GetType(IIntellisenseControllerProvider))> _
<Name("ToolTip QuickInfo Controller")> _
<ContentType("text")> _
Friend Class TestQuickInfoControllerProvider
    Implements IIntellisenseControllerProvider
```

```
[Export(typeof(IIntellisenseControllerProvider))]
[Name("ToolTip QuickInfo Controller")]
[ContentType("text")]
internal class TestQuickInfoControllerProvider : IIntellisenseControllerProvider
```

2. Importe el [IQuickInfoBroker](#) como una propiedad.

```
Private _QuickInfoBroker As IQuickInfoBroker
<Import()> _
Friend Property QuickInfoBroker() As IQuickInfoBroker
    Get
        Return _QuickInfoBroker
    End Get
    Set(ByVal value As IQuickInfoBroker)
        _QuickInfoBroker = value
    End Set
End Property
```

```
[Import]
internal IQuickInfoBroker QuickInfoBroker { get; set; }
```

3. Implemente [TryCreateIntellisenseController](#) el método creando una instancia del controlador QuickInfo.

```
Public Function TryCreateIntellisenseController(ByVal textView As ITextView, ByVal subjectBuffers As
IList(Of ITextBuffer)) As IIntellisenseController Implements
IIntellisenseControllerProvider.TryCreateIntellisenseController
    Return New TestQuickInfoController(textView, subjectBuffers, Me)
End Function
```

```
public IIntellisenseController TryCreateIntellisenseController(ITextView textView, IList<ITextBuffer>
subjectBuffers)
{
    return new TestQuickInfoController(textView, subjectBuffers, this);
}
```

Construir y probar el Código

Para probar este código, compile la solución QuickInfoTest y ejecútela en la instancia experimental.

Para compilar y probar la solución QuickInfoTest

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Cree un archivo de texto y escriba algún texto que incluya las palabras "añadir" y "restar".
4. Mueva el puntero sobre una de las apariciones de "add". Se deben mostrar la `add` firma y la descripción del método.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Mostrar la Ayuda de firma

06/04/2020 • 22 minutes to read • [Edit Online](#)

La Ayuda de firma (también conocida como *Información de parámetros*) muestra la firma de un método en una información sobre herramientas cuando un usuario escribe el carácter de inicio de la lista de parámetros (normalmente un paréntesis de apertura). Como se escribe un parámetro y un separador de parámetros (normalmente una coma), la información sobre herramientas se actualiza para mostrar el siguiente parámetro en negrita. Puede definir la Ayuda de firma de las siguientes maneras: en el contexto de un servicio de lenguaje, definir su propia extensión de nombre de archivo y tipo de contenido y mostrar la Ayuda de firma solo para ese tipo, o mostrar la Ayuda de firma para un tipo de contenido existente (por ejemplo, "texto"). En este tutorial se muestra cómo mostrar la Ayuda de firma para el tipo de contenido "texto".

La Ayuda de firma normalmente se desencadena escribiendo un carácter específico, por ejemplo, "(" (entre paréntesis de apertura) y se descarta escribiendo otro carácter, por ejemplo, ")" (paréntesis de cierre). Las características de IntelliSense que se desencadenan escribiendo un carácter se [IOleCommandTarget](#) pueden implementar mediante un controlador [IVsTextViewCreationListener](#) de comandos para las pulsaciones de teclas (la interfaz) y un proveedor de controlador que implementa la interfaz. Para crear el origen de la Ayuda de firma, que es [ISignatureHelpSource](#) la lista de firmas [ISignatureHelpSourceProvider](#) que participan en la Ayuda de firma, implemente la interfaz y un proveedor de origen que ejecute la interfaz. Los proveedores son componentes de Managed Extensibility Framework (MEF) y son responsables de exportar las [ITextStructureNavigatorSelectorService](#) clases de origen y controlador e [ISignatureHelpBroker](#) importar servicios y intermediarios, por ejemplo, el `TextBuffer`, que le permite navegar en el búfer de texto, y el `TextBuffer`, que desencadena la sesión de ayuda de firma.

En este tutorial se muestra cómo configurar la Ayuda de firma para un conjunto de identificadores codificado de forma rígida. En las implementaciones completas, el lenguaje es responsable de proporcionar ese contenido.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Creación de un proyecto MEF

Para crear un nuevo proyecto de MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `SignatureHelpTest` nombre a la solución .
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.
4. Agregue las siguientes referencias al proyecto y asegúrese `false` de que **CopyLocal** está establecido en :

Microsoft.VisualStudio.Editor

Microsoft.VisualStudio.Language.Intellisense

Microsoft.VisualStudio.OLE.Interop

Microsoft.VisualStudio.Shell.14.0

Implementar firmas y parámetros de la Ayuda de firmas

El origen de la Ayuda de [ISignature](#) firma se basa en [IParameter](#) las firmas que implementan, cada una de las cuales contiene parámetros que implementan . En una implementación completa, esta información se obtendría de la documentación del lenguaje, pero en este ejemplo, las firmas están codificadas de forma rígida.

Para implementar las firmas y los parámetros de la Ayuda de firma

1. Agregue un archivo de clase y asígnele el nombre `SignatureHelpSource` .
2. Agregue las importaciones siguientes.

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.Collections.ObjectModel
Imports System.ComponentModel.Composition
Imports System.Runtime.InteropServices
Imports Microsoft.VisualStudio.Language.Intellisense
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.Utilities
Imports Microsoft.VisualStudio.Editor
Imports Microsoft.VisualStudio.Text.Operations
Imports Microsoft.VisualStudio
Imports Microsoft.VisualStudio.TextManager.Interop
Imports Microsoft.VisualStudio.OLE.Interop
```

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.Composition;
using System.Runtime.InteropServices;
using Microsoft.VisualStudio.Language.Intellisense;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Utilities;
using Microsoft.VisualStudio.Editor;
using Microsoft.VisualStudio.Text.Operations;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.TextManager.Interop;
using Microsoft.VisualStudio.OLE.Interop;
```

3. Agregue una `TestParameter` clase denominada [IParameter](#) que implemente .

```
Friend Class TestParameter
    Implements IParameter
```

```
internal class TestParameter : IParameter
```

4. Agregue un constructor que establezca todas las propiedades.

```
Public Sub New(ByVal documentation As String, ByVal locus As Span, ByVal name As String, ByVal
signature As ISignature)
    Me.privateDocumentation = documentation
    Me.privateLocus = locus
    Me.privateName = name
    Me.privateSignature = signature
End Sub
```

```
public TestParameter(string documentation, Span locus, string name, ISignature signature)
{
    Documentation = documentation;
    Locus = locus;
    Name = name;
    Signature = signature;
}
```

5. Agregue las [IParameter](#) propiedades de .

```
Private privateDocumentation As String
ReadOnly Property Documentation() As String Implements IParameter.Documentation
    Get
        Return privateDocumentation
    End Get
End Property
Private privateLocus As Span
ReadOnly Property Locus() As Span Implements IParameter.Locus
    Get
        Return privateLocus
    End Get
End Property
Private privateName As String
ReadOnly Property Name() As String Implements IParameter.Name
    Get
        Return privateName
    End Get
End Property
Private privateSignature As ISignature
ReadOnly Property Signature() As ISignature Implements IParameter.Signature
    Get
        Return privateSignature
    End Get
End Property
Private privatePrettyPrintedLocus As Span
ReadOnly Property PrettyPrintedLocus() As Span Implements IParameter.PrettyPrintedLocus
    Get
        Return privatePrettyPrintedLocus
    End Get
End Property
```

```
public string Documentation { get; private set; }
public Span Locus { get; private set; }
public string Name { get; private set; }
public ISignature Signature { get; private set; }
public Span PrettyPrintedLocus { get; private set; }
```

6. Agregue una `TestSignature` clase denominada [ISignature](#) que implemente .

```
Friend Class TestSignature
    Implements ISignature
```

```
internal class TestSignature : ISignature
```

7. Agregue algunos campos privados.

```
Private m_subjectBuffer As ITextBuffer
Private m_currentParameter As IParameter
Private m_content As String
Private m_documentation As String
Friend m_applicableToSpan As ITrackingSpan
Friend m_parameters As ReadOnlyCollection(Of IParameter)
Private m_printContent As String
```

```
private ITextBuffer m_subjectBuffer;
private IParameter m_currentParameter;
private string m_content;
private string m_documentation;
private ITrackingSpan m_applicableToSpan;
private ReadOnlyCollection<IParameter> m_parameters;
private string m_printContent;
```

8. Agregue un constructor que establezca los [Changed](#) campos y se suscriba al evento.

```
Friend Sub New(ByVal subjectBuffer As ITextBuffer, ByVal content As String, ByVal doc As String, ByVal
parameters As ReadOnlyCollection(Of IParameter))
    m_subjectBuffer = subjectBuffer
    m_content = content
    m_documentation = doc
    m_parameters = parameters
    AddHandler m_subjectBuffer.Changed, AddressOf OnSubjectBufferChanged
End Sub
```

```
internal TestSignature(ITextBuffer subjectBuffer, string content, string doc,
ReadOnlyCollection<IParameter> parameters)
{
    m_subjectBuffer = subjectBuffer;
    m_content = content;
    m_documentation = doc;
    m_parameters = parameters;
    m_subjectBuffer.Changed += new EventHandler<TextContentChangedEventArgs>(OnSubjectBufferChanged);
}
```

9. Declarar `CurrentParameterChanged` un evento. Este evento se produce cuando el usuario rellena uno de los parámetros de la firma.

```
Public Event CurrentParameterChanged As EventHandler(Of CurrentParameterChangedEventArgs) Implements
ISignature.CurrentParameterChanged
```

```
public event EventHandler<CurrentParameterChangedEventArgs> CurrentParameterChanged;
```

10. Implemente [CurrentParameter](#) la propiedad para que `CurrentParameterChanged` genere el evento cuando se

cambia el valor de la propiedad.

```
ReadOnly Property CurrentParameter() As IParameter Implements ISignature.CurrentParameter
    Get
        Return m_currentParameter
    End Get
End Property
```

```
public IParameter CurrentParameter
{
    get { return m_currentParameter; }
    internal set
    {
        if (m_currentParameter != value)
        {
            IParameter prevCurrentParameter = m_currentParameter;
            m_currentParameter = value;
            this.RaiseCurrentParameterChanged(prevCurrentParameter, m_currentParameter);
        }
    }
}
```

11. Agregue un método que `CurrentParameterChanged` genere el evento.

```
Private Sub RaiseCurrentParameterChanged(ByVal prevCurrentParameter As IParameter, ByVal
newCurrentParameter As IParameter)
    Dim tempHandler As EventHandler(Of CurrentParameterChangedEventArgs) =
Me.CurrentParameterChangedEvent
    If tempHandler IsNot Nothing Then
        tempHandler(Me, New CurrentParameterChangedEventArgs(prevCurrentParameter,
newCurrentParameter))
    End If
End Sub
```

```
private void RaiseCurrentParameterChanged(IParameter prevCurrentParameter, IParameter
newCurrentParameter)
{
    EventHandler<CurrentParameterChangedEventArgs> tempHandler = this.CurrentParameterChanged;
    if (tempHandler != null)
    {
        tempHandler(this, new CurrentParameterChangedEventArgs(prevCurrentParameter,
newCurrentParameter));
    }
}
```

12. Agregue un método que calcule el parámetro actual comparando [ApplicableToSpan](#) el número de comas en el número de comas de la firma.

```

Friend Sub ComputeCurrentParameter()
    If Parameters.Count = 0 Then
        Me.m_currentParameter = Nothing
        Return
    End If

    'the number of commas in the string is the index of the current parameter
    Dim sigText As String = ApplicableToSpan.GetText(m_subjectBuffer.CurrentSnapshot)

    Dim currentIndex As Integer = 0
    Dim commaCount As Integer = 0
    Do While currentIndex < sigText.Length
        Dim commaIndex As Integer = sigText.IndexOf(",", currentIndex)
        If commaIndex = -1 Then
            Exit Do
        End If
        commaCount += 1
        currentIndex = commaIndex + 1
    Loop

    If commaCount < Parameters.Count Then
        Me.m_currentParameter = Parameters(commaCount)
    Else
        'too many commas, so use the last parameter as the current one.
        Me.m_currentParameter = Parameters(Parameters.Count - 1)
    End If
End Sub

```

```

internal void ComputeCurrentParameter()
{
    if (Parameters.Count == 0)
    {
        this.CurrentParameter = null;
        return;
    }

    //the number of commas in the string is the index of the current parameter
    string sigText = ApplicableToSpan.GetText(m_subjectBuffer.CurrentSnapshot);

    int currentIndex = 0;
    int commaCount = 0;
    while (currentIndex < sigText.Length)
    {
        int commaIndex = sigText.IndexOf(',', currentIndex);
        if (commaIndex == -1)
        {
            break;
        }
        commaCount++;
        currentIndex = commaIndex + 1;
    }

    if (commaCount < Parameters.Count)
    {
        this.CurrentParameter = Parameters[commaCount];
    }
    else
    {
        //too many commas, so use the last parameter as the current one.
        this.CurrentParameter = Parameters[Parameters.Count - 1];
    }
}

```

13. Agregue un controlador [Changed](#) de eventos `ComputeCurrentParameter()` para el evento que llama a

método.

```
Friend Sub OnSubjectBufferChanged(ByVal sender As Object, ByVal e As TextContentChangedEventArgs)
    Me.ComputeCurrentParameter()
End Sub
```

```
internal void OnSubjectBufferChanged(object sender, TextContentChangedEventArgs e)
{
    this.ComputeCurrentParameter();
}
```

14. Implemente la propiedad [ApplicableToSpan](#). Esta propiedad contiene [ITrackingSpan](#) un que corresponde al intervalo de texto en el búfer al que se aplica la firma.

```
ReadOnly Property ApplicableToSpan() As ITrackingSpan Implements ISignature.ApplicableToSpan
    Get
        Return (m_applicableToSpan)
    End Get
End Property
```

```
public ITrackingSpan ApplicableToSpan
{
    get { return (m_applicableToSpan); }
    internal set { m_applicableToSpan = value; }
}
```

15. Implemente los demás parámetros.

```
ReadOnly Property Content() As String Implements ISignature.Content
    Get
        Return (m_content)
    End Get
End Property

ReadOnly Property Documentation() As String Implements ISignature.Documentation
    Get
        Return (m_documentation)
    End Get
End Property

ReadOnly Property Parameters() As ReadOnlyCollection(Of IParameter) Implements ISignature.Parameters
    Get
        Return (m_parameters)
    End Get
End Property

ReadOnly Property PrettyPrintedContent() As String Implements ISignature.PrettyPrintedContent
    Get
        Return (m_printContent)
    End Get
End Property
```

```

public string Content
{
    get { return (m_content); }
    internal set { m_content = value; }
}

public string Documentation
{
    get { return (m_documentation); }
    internal set { m_documentation = value; }
}

public ReadOnlyCollection<IParameter> Parameters
{
    get { return (m_parameters); }
    internal set { m_parameters = value; }
}

public string PrettyPrintedContent
{
    get { return (m_printContent); }
    internal set { m_printContent = value; }
}

```

Implementar el origen de la Ayuda de firma

El origen de la Ayuda de firma es el conjunto de firmas para las que se proporciona información.

Para implementar el origen de la Ayuda de firma

1. Agregue una `TestSignatureHelpSource` clase denominada [ISignatureHelpSource](#) que implemente .

```

Friend Class TestSignatureHelpSource
    Implements ISignatureHelpSource

```

```

internal class TestSignatureHelpSource : ISignatureHelpSource

```

2. Agregue una referencia al búfer de texto.

```

Private m_textBuffer As ITextBuffer

```

```

private ITextBuffer m_textBuffer;

```

3. Agregue un constructor que establezca el búfer de texto y el proveedor de origen de la Ayuda de firma.

```

Public Sub New(ByVal textBuffer As ITextBuffer)
    m_textBuffer = textBuffer
End Sub

```

```

public TestSignatureHelpSource(ITextBuffer textBuffer)
{
    m_textBuffer = textBuffer;
}

```

4. Implemente el método [AugmentSignatureHelpSession](#). En este ejemplo, las firmas están codificadas de

forma rígida, pero en una implementación completa obtendrá esta información de la documentación del lenguaje.

```
Public Sub AugmentSignatureHelpSession(ByVal session As ISignatureHelpSession, ByVal signatures As
IList(Of ISignature)) Implements ISignatureHelpSource.AugmentSignatureHelpSession
    Dim snapshot As ITextSnapshot = m_textBuffer.CurrentSnapshot
    Dim position As Integer = session.GetTriggerPoint(m_textBuffer).GetPosition(snapshot)

    Dim applicableToSpan As ITrackingSpan = m_textBuffer.CurrentSnapshot.CreateTrackingSpan(New
Span(position, 0), SpanTrackingMode.EdgeInclusive, 0)

    signatures.Add(CreateSignature(m_textBuffer, "add(int firstInt, int secondInt)", "Documentation for
adding integers.", applicableToSpan))
    signatures.Add(CreateSignature(m_textBuffer, "add(double firstDouble, double secondDouble)",
"Documentation for adding doubles.", applicableToSpan))
End Sub
```

```
public void AugmentSignatureHelpSession(ISignatureHelpSession session, IList<ISignature> signatures)
{
    ITextSnapshot snapshot = m_textBuffer.CurrentSnapshot;
    int position = session.GetTriggerPoint(m_textBuffer).GetPosition(snapshot);

    ITrackingSpan applicableToSpan = m_textBuffer.CurrentSnapshot.CreateTrackingSpan(
        new Span(position, 0), SpanTrackingMode.EdgeInclusive, 0);

    signatures.Add(CreateSignature(m_textBuffer, "add(int firstInt, int secondInt)", "Documentation for
adding integers.", applicableToSpan));
    signatures.Add(CreateSignature(m_textBuffer, "add(double firstDouble, double secondDouble)",
"Documentation for adding doubles.", applicableToSpan));
}
```

5. El método `CreateSignature()` auxiliar se proporciona solo para la ilustración.

```

Private Function CreateSignature(ByVal textBuffer As ITextBuffer, ByVal methodSig As String, ByVal
methodDoc As String, ByVal span As ITrackingSpan) As TestSignature
    Dim sig As New TestSignature(textBuffer, methodSig, methodDoc, Nothing)
    AddHandler textBuffer.Changed, AddressOf sig.OnSubjectBufferChanged

    'find the parameters in the method signature (expect methodname(one, two))
    Dim pars() As String = methodSig.Split(New Char() {"(c, ", ", "c, ")"}c})
    Dim paramList As New List(Of IParameter)()

    Dim locusSearchStart As Integer = 0
    For i As Integer = 1 To pars.Length - 1
        Dim param As String = pars(i).Trim()

        If String.IsNullOrEmpty(param) Then
            Continue For
        End If

        'find where this parameter is located in the method signature
        Dim locusStart As Integer = methodSig.IndexOf(param, locusSearchStart)
        If locusStart >= 0 Then
            Dim locus As New Span(locusStart, param.Length)
            locusSearchStart = locusStart + param.Length
            paramList.Add(New TestParameter("Documentation for the parameter.", locus, param, sig))
        End If
    Next i

    sig.m_Parameters = New ReadOnlyCollection(Of IParameter)(paramList)
    sig.m_ApplicableToSpan = span
    sig.ComputeCurrentParameter()
    Return sig
End Function

```

```

private TestSignature CreateSignature(ITextBuffer textBuffer, string methodSig, string methodDoc,
ITrackingSpan span)
{
    TestSignature sig = new TestSignature(textBuffer, methodSig, methodDoc, null);
    textBuffer.Changed += new EventHandler<TextContentChangedEventArgs>(sig.OnSubjectBufferChanged);

    //find the parameters in the method signature (expect methodname(one, two))
    string[] pars = methodSig.Split(new char[] { '(', ',', ')' });
    List<IParameter> paramList = new List<IParameter>();

    int locusSearchStart = 0;
    for (int i = 1; i < pars.Length; i++)
    {
        string param = pars[i].Trim();

        if (string.IsNullOrEmpty(param))
            continue;

        //find where this parameter is located in the method signature
        int locusStart = methodSig.IndexOf(param, locusSearchStart);
        if (locusStart >= 0)
        {
            Span locus = new Span(locusStart, param.Length);
            locusSearchStart = locusStart + param.Length;
            paramList.Add(new TestParameter("Documentation for the parameter.", locus, param, sig));
        }
    }

    sig.Parameters = new ReadOnlyCollection<IParameter>(paramList);
    sig.ApplicableToSpan = span;
    sig.ComputeCurrentParameter();
    return sig;
}

```

6. Implemente el método [GetBestMatch](#). En este ejemplo, hay solo dos firmas, cada una de las cuales tiene dos parámetros. Por lo tanto, este método no es necesario. En una implementación más completa, en la que hay más de un origen de Ayuda de firma disponible, este método se usa para decidir si el origen de Ayuda de firma de mayor prioridad puede proporcionar una firma coincidente. Si no puede, el método devuelve null y se pide al origen de prioridad más siguiente que proporcione una coincidencia.

```
Public Function GetBestMatch(ByVal session As ISignatureHelpSession) As ISignature Implements
ISignatureHelpSource.GetBestMatch
    If session.Signatures.Count > 0 Then
        Dim applicableToSpan As ITrackingSpan = session.Signatures(0).ApplicableToSpan
        Dim text As String = applicableToSpan.GetText(applicableToSpan.TextBuffer.CurrentSnapshot)

        If text.Trim().Equals("add") Then 'get only "add"
            Return session.Signatures(0)
        End If
    End If
    Return Nothing
End Function
```

```
public ISignature GetBestMatch(ISignatureHelpSession session)
{
    if (session.Signatures.Count > 0)
    {
        ITrackingSpan applicableToSpan = session.Signatures[0].ApplicableToSpan;
        string text = applicableToSpan.GetText(applicableToSpan.TextBuffer.CurrentSnapshot);

        if (text.Trim().Equals("add")) //get only "add"
            return session.Signatures[0];
    }
    return null;
}
```

7. Implemente `Dispose()` el método:

```
Private m_isDisposed As Boolean

Public Sub Dispose() Implements IDisposable.Dispose
    If Not m_isDisposed Then
        GC.SuppressFinalize(Me)
        m_isDisposed = True
    End If
End Sub
```

```
private bool m_isDisposed;
public void Dispose()
{
    if (!m_isDisposed)
    {
        GC.SuppressFinalize(this);
        m_isDisposed = true;
    }
}
```

Implementar el proveedor de origen de la Ayuda de firma

El proveedor de origen de la Ayuda de firma es responsable de exportar la parte del componente Managed Extensibility Framework (MEF) y de crear instancias del origen de la Ayuda de firma.

Para implementar el proveedor de origen de la Ayuda de firma

1. Agregue una `TestSignatureHelpSourceProvider` clase denominada `ISignatureHelpSourceProvider` que implemente y `NameAttribute` expórtela con un `,` un `ContentTypeAttribute` de "text" y un `OrderAttribute` de Before-"default".

```
<Export(GetType(ISignatureHelpSourceProvider)), Name("Signature Help source"),
Order(Before:="default"), ContentType("text")>
Friend Class TestSignatureHelpSourceProvider
    Implements ISignatureHelpSourceProvider
```

```
[Export(typeof(ISignatureHelpSourceProvider))]
[Name("Signature Help source")]
[Order(Before = "default")]
[ContentType("text")]
internal class TestSignatureHelpSourceProvider : ISignatureHelpSourceProvider
```

2. Implemente `TryCreateSignatureHelpSource` creando un `TestSignatureHelpSource` archivo .

```
Public Function TryCreateSignatureHelpSource(ByVal textBuffer As ITextBuffer) As ISignatureHelpSource
    Implements ISignatureHelpSourceProvider.TryCreateSignatureHelpSource
        Return New TestSignatureHelpSource(textBuffer)
End Function
```

```
public ISignatureHelpSource TryCreateSignatureHelpSource(ITextBuffer textBuffer)
{
    return new TestSignatureHelpSource(textBuffer);
}
```

Implementar el controlador de comandos

La Ayuda de firma normalmente se desencadena por un paréntesis de apertura "(" carácter y se descarta por un paréntesis de cierre ")". Puede controlar estas pulsaciones de `IoleCommandTarget` teclas ejecutando un para que desencadene una sesión de Ayuda de firma cuando recibe un carácter de paréntesis de apertura precedido por un nombre de método conocido y descarta la sesión cuando recibe un carácter de paréntesis de cierre.

Para implementar el controlador de comandos

1. Agregue una `TestSignatureHelpCommand` clase denominada `IoleCommandTarget` que implemente .

```
Friend NotInheritable Class TestSignatureHelpCommandHandler
    Implements IoleCommandTarget
```

```
internal sealed class TestSignatureHelpCommandHandler : IoleCommandTarget
```

2. Agregue campos privados para el `IVsTextView` adaptador (que le permite agregar el controlador de comandos a la `ITextStructureNavigator` cadena de `IoleCommandTarget` controladores de comandos), la vista de texto, el intermediario y la sesión de la Ayuda de firma, un `,` y el siguiente .

```
Private m_nextCommandHandler As IoleCommandTarget
Private m_textView As ITextView
Private m_broker As ISignatureHelpBroker
Private m_session As ISignatureHelpSession
Private m_navigator As ITextStructureNavigator
```



```
IOleCommandTarget m_nextCommandHandler;
ITextView m_textView;
ISignatureHelpBroker m_broker;
ISignatureHelpSession m_session;
ITextStructureNavigator m_navigator;
```

3. Agregue un constructor para inicializar estos campos y agregar el filtro de comandos a la cadena de filtros de comandos.

```
Friend Sub New(ByVal textViewAdapter As IVsTextView, ByVal textView As ITextView, ByVal nav As
ITextStructureNavigator, ByVal broker As ISignatureHelpBroker)
    Me.m_textView = textView
    Me.m_broker = broker
    Me.m_navigator = nav

    'add this to the filter chain
    textViewAdapter.AddCommandFilter(Me, m_nextCommandHandler)
End Sub
```

```
internal TestSignatureHelpCommandHandler(IVsTextView textViewAdapter, ITextView textView,
ITextStructureNavigator nav, ISignatureHelpBroker broker)
{
    this.m_textView = textView;
    this.m_broker = broker;
    this.m_navigator = nav;

    //add this to the filter chain
    textViewAdapter.AddCommandFilter(this, out m_nextCommandHandler);
}
```

4. Implemente [Exec](#) el método para desencadenar la sesión de Ayuda de firma cuando el filtro de comandos recibe un paréntesis de apertura "(" carácter después de uno de los nombres de método conocidos y para descartar la sesión cuando recibe un paréntesis de cierre ")" carácter mientras la sesión sigue activa. En todos los casos, se reenvía el comando.

```
Public Function Exec(ByRef pguidCmdGroup As Guid, ByVal nCmdID As UInteger, ByVal nCmdexcopt As
UInteger, ByVal pvaIn As IntPtr, ByVal pvaOut As IntPtr) As Integer Implements IOleCommandTarget.Exec
    Dim typedChar As Char = Char.MinValue

    If pguidCmdGroup = VSConstants.VSStd2K AndAlso nCmdID = CUInt(VSConstants.VSStd2KCmdID.TYPECHAR)
    Then
        typedChar = CChar(ChrW(CUShort(Marshal.GetObjectForNativeVariant(pvaIn))))
        If typedChar.Equals("("c) Then
            'move the point back so it's in the preceding word
            Dim point As SnapshotPoint = m_textView.Caret.Position.BufferPosition - 1
            Dim extent As TextExtent = m_navigator.GetExtentOfWord(point)
            Dim word As String = extent.Span.GetText()
            If word.Equals("add") Then
                m_session = m_broker.TriggerSignatureHelp(m_textView)
            End If

            ElseIf typedChar.Equals(")c) AndAlso m_session IsNot Nothing Then
                m_session.Dismiss()
                m_session = Nothing
            End If
        End If

        Return m_nextCommandHandler.Exec(pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn, pvaOut)
    End Function
```

```

public int Exec(ref Guid pguidCmdGroup, uint nCmdID, uint nCmdexecept, IntPtr pvaIn, IntPtr pvaOut)
{
    char typedChar = char.MinValue;

    if (pguidCmdGroup == VSConstants.VSStd2K && nCmdID == (uint)VSConstants.VSStd2KCmdID.TYPECHAR)
    {
        typedChar = (char)(ushort)Marshal.GetObjectForNativeVariant(pvaIn);
        if (typedChar.Equals('.'))
        {
            //move the point back so it's in the preceding word
            SnapshotPoint point = m_textView.Caret.Position.BufferPosition - 1;
            TextExtent extent = m_navigator.GetExtentOfWord(point);
            string word = extent.Span.GetText();
            if (word.Equals("add"))
                m_session = m_broker.TriggerSignatureHelp(m_textView);

        }
        else if (typedChar.Equals(' ')) && m_session != null)
        {
            m_session.Dismiss();
            m_session = null;
        }
    }
    return m_nextCommandHandler.Exec(ref pguidCmdGroup, nCmdID, nCmdexecept, pvaIn, pvaOut);
}

```

5. Implemente [QueryStatus](#) el método para que siempre reenvíe el comando.

```

Public Function QueryStatus(ByRef pguidCmdGroup As Guid, ByVal cCmds As UInteger, ByVal prgCmds() As
OLECMD, ByVal pCmdText As IntPtr) As Integer Implements IOleCommandTarget.QueryStatus
    Return m_nextCommandHandler.QueryStatus(pguidCmdGroup, cCmds, prgCmds, pCmdText)
End Function

```

```

public int QueryStatus(ref Guid pguidCmdGroup, uint cCmds, OLECMD[] prgCmds, IntPtr pCmdText)
{
    return m_nextCommandHandler.QueryStatus(ref pguidCmdGroup, cCmds, prgCmds, pCmdText);
}

```

Implementar el proveedor de comandos de ayuda de firma

Puede proporcionar el comando Ayuda [IVsTextViewCreationListener](#) de firma implementando la instancia del controlador de comandos cuando se crea la vista de texto.

Para implementar el proveedor de comandos de la Ayuda de firma

1. Agregue una `TestSignatureHelpController` clase denominada [IVsTextViewCreationListener](#) que la [NameAttribute](#) implemente [ContentTypeAttribute](#) y [TextViewRoleAttribute](#) expórtela con `,, y`.

```

<Export(GetType(IVsTextViewCreationListener)), Name("Signature Help controller"),
TextViewRole(PredefinedTextViewRoles.Editable), ContentType("text")>
Friend Class TestSignatureHelpCommandProvider
    Implements IVsTextViewCreationListener

```

```

[Export(typeof(IVsTextViewCreationListener))]
[Name("Signature Help controller")]
[TextViewRole(PredefinedTextViewRoles.Editable)]
[ContentType("text")]
internal class TestSignatureHelpCommandProvider : IVsTextViewCreationListener

```

- Importe [IVsEditorAdaptersFactoryService](#) el (utilizado [ITextView](#) para obtener [IVsTextView](#) el [ITextStructureNavigatorSelectorService](#) , dado el objeto), el [ISignatureHelpBroker](#) (utilizado para buscar la palabra actual) y el (para desencadenar la sesión de Ayuda de firma).

```
<Import()>
Friend AdapterService As IVsEditorAdaptersFactoryService

<Import()>
Friend Property NavigatorService() As ITextStructureNavigatorSelectorService

<Import()>
Friend SignatureHelpBroker As ISignatureHelpBroker
```

```
[Import]
internal IVsEditorAdaptersFactoryService AdapterService;

[Import]
internal ITextStructureNavigatorSelectorService NavigatorService { get; set; }

[Import]
internal ISignatureHelpBroker SignatureHelpBroker;
```

- Implemente [VsTextViewCreated](#) el método creando una `TestSignatureCommandHandler` instancia de .

```
Public Sub VsTextViewCreated(ByVal textViewAdapter As IVsTextView) Implements
IVsTextViewCreationListener.VsTextViewCreated
    Dim textView As ITextView = AdapterService.GetWpfTextView(textViewAdapter)
    If textView Is Nothing Then
        Return
    End If

    textView.Properties.GetOrCreateSingletonProperty(Function() New
TestSignatureHelpCommandHandler(textViewAdapter, textView,
NavigatorService.GetTextStructureNavigator(textView.TextBuffer), SignatureHelpBroker))
End Sub
```

```
public void VsTextViewCreated(IVsTextView textViewAdapter)
{
    ITextView textView = AdapterService.GetWpfTextView(textViewAdapter);
    if (textView == null)
        return;

    textView.Properties.GetOrCreateSingletonProperty(
        () => new TestSignatureHelpCommandHandler(textViewAdapter,
            textView,
            NavigatorService.GetTextStructureNavigator(textView.TextBuffer),
            SignatureHelpBroker));
}
```

Construir y probar el código

Para probar este código, compile la solución `SignatureHelpTest` y ejecútela en la instancia experimental.

Para compilar y probar la solución `SignatureHelpTest`

- Compile la solución.
- Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
- Cree un archivo de texto y escriba algún texto que incluya la palabra "añadir" más un paréntesis de

apertura.

4. Después de escribir el paréntesis de apertura, debería ver una información sobre `add()` herramientas que muestra una lista de las dos firmas para el método.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Mostrar la finalización de instrucciones

06/04/2020 • 18 minutes to read • [Edit Online](#)

Puede implementar la finalización de instrucciones basadas en lenguaje definiendo los identificadores para los que desea proporcionar la finalización y, a continuación, desencadenando una sesión de finalización. Puede definir la finalización de instrucciones en el contexto de un servicio de lenguaje, definir su propia extensión de nombre de archivo y tipo de contenido y, a continuación, mostrar la finalización solo para ese tipo. O bien, puede desencadenar la finalización de un tipo de contenido existente, por ejemplo, "texto sin formato". En este tutorial se muestra cómo desencadenar la finalización de instrucciones para el tipo de contenido "texto sin formato", que es el tipo de contenido de los archivos de texto. El tipo de contenido "texto" es el antepasado de todos los demás tipos de contenido, incluidos el código y los archivos XML.

La finalización de instrucciones normalmente se desencadena escribiendo ciertos caracteres, por ejemplo, escribiendo el principio de un identificador como "using". Normalmente se descarta presionando la tecla **Barra espaciadora**, **Tabulador** o **Intro** para confirmar una selección. Puede implementar las características de IntelliSense que se desencadenan al escribir un `ICommandTarget` carácter mediante un controlador `IVsTextViewCreationListener` de comandos para las pulsaciones de teclas (la interfaz) y un proveedor de controlador que implementa la interfaz. Para crear el origen de finalización, que es la `ICompletionSource` lista de identificadores que `ICompletionSourceProvider` participan en la finalización, implemente la interfaz y un proveedor de origen de finalización (la interfaz). Los proveedores son componentes de Managed Extensibility Framework (MEF). Son responsables de exportar las clases de origen y controlador e `ITextStructureNavigatorSelectorService` importar servicios y intermediarios, por `ICompletionBroker` ejemplo, el , que habilita la navegación en el búfer de texto y el , que desencadena la sesión de finalización.

En este tutorial se muestra cómo implementar la finalización de instrucciones para un conjunto de identificadores codificado de forma rígida. En las implementaciones completas, el servicio de lenguaje y la documentación de idioma son responsables de proporcionar ese contenido.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto MEF

Para crear un nuevo proyecto de MEF

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `CompletionTest` nombre a la solución .
2. Agregue una plantilla de elemento Clasificador de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.
4. Agregue las siguientes referencias al proyecto y asegúrese `false` de que **CopyLocal** está establecido en :

Microsoft.VisualStudio.Editor

Microsoft.VisualStudio.Language.Intellisense

Microsoft.VisualStudio.OLE.Interop

Microsoft.VisualStudio.Shell.15.0

Microsoft.VisualStudio.Shell.Immutable.10.0

Microsoft.VisualStudio.TextManager.Interop

Implementar la fuente de finalización

El origen de finalización es responsable de recopilar el conjunto de identificadores y agregar el contenido a la ventana de finalización cuando un usuario escribe un desencadenador de finalización, como las primeras letras de un identificador. En este ejemplo, los identificadores y sus descripciones [AugmentCompletionSession](#) están codificados de forma rígida en el método. En la mayoría de los usos del mundo real, usaría el analizador de su idioma para obtener los tokens para rellenar la lista de finalización.

Para implementar la fuente de finalización

1. Agregue un archivo de clase y asígnele el nombre `TestCompletionSource`.
2. Agregue estas importaciones:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Language.Intellisense;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Operations;
using Microsoft.VisualStudio.Utilities;
```

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.ComponentModel.Composition
Imports Microsoft.VisualStudio.Language.Intellisense
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Operations
Imports Microsoft.VisualStudio.Utilities
```

3. Modifique la declaración de clase para `TestCompletionSource` que [ICompletionSource](#) implemente:

```
internal class TestCompletionSource : ICompletionSource
```

```
Friend Class TestCompletionSource
    Implements ICompletionSource
```

4. Agregue campos privados para el proveedor de origen, el búfer de texto y una lista de [Completion](#) objetos (que corresponden a los identificadores que participarán en la sesión de finalización):

```
private TestCompletionSourceProvider m_sourceProvider;
private ITextBuffer m_textBuffer;
private List<Completion> m_complList;
```

```

Private m_sourceProvider As TestCompletionSourceProvider
Private m_textBuffer As ITextBuffer
Private m_complList As List(Of Completion)

```

5. Agregue un constructor que establezca el proveedor de origen y el búfer. La `TestCompletionSourceProvider` clase se define en pasos posteriores:

```

public TestCompletionSource(TestCompletionSourceProvider sourceProvider, ITextBuffer textBuffer)
{
    m_sourceProvider = sourceProvider;
    m_textBuffer = textBuffer;
}

```

```

Public Sub New(ByVal sourceProvider As TestCompletionSourceProvider, ByVal textBuffer As ITextBuffer)
    m_sourceProvider = sourceProvider
    m_textBuffer = textBuffer
End Sub

```

6. Implemente `AugmentCompletionSession` el método agregando un conjunto de finalización que contenga las terminaciones que desea proporcionar en el contexto. Cada conjunto de finalización contiene un conjunto de `Completion` terminaciones y corresponde a una pestaña de la ventana de finalización. (En los proyectos de Visual Basic, las pestañas de la ventana de finalización se denominan **Común** y **Todos**.) El `FindTokenSpanAtPosition` método se define en el paso siguiente.

```

void ICompletionSource.AugmentCompletionSession(ICompletionSession session, IList<CompletionSet> completionSets)
{
    List<string> strList = new List<string>();
    strList.Add("addition");
    strList.Add("adaptation");
    strList.Add("subtraction");
    strList.Add("summation");
    m_complList = new List<Completion>();
    foreach (string str in strList)
        m_complList.Add(new Completion(str, str, str, null, null));

    completionSets.Add(new CompletionSet(
        "Tokens",    //the non-localized title of the tab
        "Tokens",    //the display title of the tab
        FindTokenSpanAtPosition(session.GetTriggerPoint(m_textBuffer),
            session),
        m_complList,
        null));
}

```

```

Private Sub AugmentCompletionSession(ByVal session As ICompletionSession, ByVal completionSets As
IList(Of CompletionSet)) Implements ICompletionSource.AugmentCompletionSession
    Dim strList As New List(Of String)()
    strList.Add("addition")
    strList.Add("adaptation")
    strList.Add("subtraction")
    strList.Add("summation")

    m_complList = New List(Of Completion)()
    For Each str As String In strList
        m_complList.Add(New Completion(str, str, str, Nothing, Nothing))
    Next str

    completionSets.Add(New CompletionSet(
        "Tokens",
        "Tokens",
        FindTokenSpanAtPosition(session.GetTriggerPoint(m_textBuffer),
            session),
        m_complList,
        Nothing))
End Sub

```

7. El siguiente método se utiliza para encontrar la palabra actual desde la posición del cursor:

```

private ITrackingSpan FindTokenSpanAtPosition(ITrackingPoint point, ICompletionSession session)
{
    SnapshotPoint currentPoint = (session.TextView.Caret.Position.BufferPosition) - 1;
    ITextStructureNavigator navigator =
m_sourceProvider.Navigatorservice.GetTextStructureNavigator(m_textBuffer);
    TextExtent extent = navigator.GetExtentOfWord(currentPoint);
    return currentPoint.Snapshot.CreateTrackingSpan(extent.Span, SpanTrackingMode.EdgeInclusive);
}

```

```

Private Function FindTokenSpanAtPosition(ByVal point As ITrackingPoint, ByVal session As
ICompletionSession) As ITrackingSpan
    Dim currentPoint As SnapshotPoint = (session.TextView.Caret.Position.BufferPosition) - 1
    Dim navigator As ITextStructureNavigator =
m_sourceProvider.Navigatorservice.GetTextStructureNavigator(m_textBuffer)
    Dim extent As TextExtent = navigator.GetExtentOfWord(currentPoint)
    Return currentPoint.Snapshot.CreateTrackingSpan(extent.Span, SpanTrackingMode.EdgeInclusive)
End Function

```

8. Implemente `Dispose()` el método:

```

private bool m_isDisposed;
public void Dispose()
{
    if (!m_isDisposed)
    {
        GC.SuppressFinalize(this);
        m_isDisposed = true;
    }
}

```



```

Private m_isDisposed As Boolean
Public Sub Dispose() Implements IDisposable.Dispose
    If Not m_isDisposed Then
        GC.SuppressFinalize(Me)
        m_isDisposed = True
    End If
End Sub

```

Implementar el proveedor de origen de finalización

El proveedor de origen de finalización es la parte del componente MEF que crea una instancia del origen de finalización.

Para implementar el proveedor de origen de finalización

1. Agregue una `TestCompletionSourceProvider` clase denominada `ICompletionSourceProvider` que implemente . Exporte esta `ContentTypeAttribute` clase con un "texto plano" y un `NameAttribute` de "finalización de prueba".

```

[Export(typeof(ICompletionSourceProvider))]
[ContentType("plaintext")]
[Name("token completion")]
internal class TestCompletionSourceProvider : ICompletionSourceProvider

```

```

<Export(GetType(ICompletionSourceProvider)), ContentType("plaintext"), Name("token completion")>
Friend Class TestCompletionSourceProvider
    Implements ICompletionSourceProvider

```

2. Importar `ITextStructureNavigatorSelectorService` un , que encuentra la palabra actual en el origen de finalización.

```

[Import]
internal ITextStructureNavigatorSelectorService NavigatorService { get; set; }

```

```

<Import()>
Friend Property NavigatorService() As ITextStructureNavigatorSelectorService

```

3. Implemente `TryCreateCompletionSource` el método para crear instancias del origen de finalización.

```

public ICompletionSource TryCreateCompletionSource(ITextBuffer textBuffer)
{
    return new TestCompletionSource(this, textBuffer);
}

```

```

Public Function TryCreateCompletionSource(ByVal textBuffer As ITextBuffer) As ICompletionSource
Implements ICompletionSourceProvider.TryCreateCompletionSource
    Return New TestCompletionSource(Me, textBuffer)
End Function

```

Implementar el proveedor de controlador de comandos de finalización

El proveedor de controlador de `IVsTextViewCreationListener` comandos de finalización se deriva de un ,

[IVsTextView](#) que escucha un evento de creación de vista de texto [ITextView](#) y convierte la vista de un , que permite la adición del comando a la cadena de comandos del shell de Visual Studio, en un archivo . Dado que esta clase es una exportación MEF, también puede usarla para importar los servicios requeridos por el propio controlador de comandos.

Para implementar el proveedor de controlador de comandos de finalización

1. Agregue un `TestCompletionCommandHandler` archivo denominado .
2. Agregue estas directivas using:

```
using System;
using System.ComponentModel.Composition;
using System.Runtime.InteropServices;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Editor;
using Microsoft.VisualStudio.Language.Intellisense;
using Microsoft.VisualStudio.OLE.Interop;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.TextManager.Interop;
using Microsoft.VisualStudio.Utilities;
```

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.ComponentModel.Composition
Imports System.Runtime.InteropServices
Imports Microsoft.VisualStudio
Imports Microsoft.VisualStudio.Editor
Imports Microsoft.VisualStudio.Language.Intellisense
Imports Microsoft.VisualStudio.OLE.Interop
Imports Microsoft.VisualStudio.Shell
Imports Microsoft.VisualStudio.Text
Imports Microsoft.VisualStudio.Text.Editor
Imports Microsoft.VisualStudio.TextManager.Interop
Imports Microsoft.VisualStudio.Utilities
```

3. Agregue una `TestCompletionHandlerProvider` clase denominada [IVsTextViewCreationListener](#) que implemente . Exporte esta [NameAttribute](#) clase con un de [ContentTypeAttribute](#) "controlador de finalización [TextViewRoleAttribute](#) de [EditableToken](#)", un de "texto normal" y un de .

```
[Export(typeof(IVsTextViewCreationListener))]
[Name("token completion handler")]
[ContentType("plaintext")]
[TextViewRole(PredefinedTextViewRoles.Editable)]
internal class TestCompletionHandlerProvider : IVsTextViewCreationListener
```

```
<Export(GetType(IVsTextViewCreationListener))>
<Name("token completion handler")>
<ContentType("plaintext")> <TextViewRole(PredefinedTextViewRoles.Editable)>
Friend Class TestCompletionHandlerProvider
    Implements IVsTextViewCreationListener
```

4. Importe [IVsEditorAdaptersFactoryService](#), que [IVsTextView](#) permite [ITextView](#) la [ICompletionBroker](#) conversión [SVsServiceProvider](#) de a , un , y un que permite el acceso a los servicios estándar de Visual Studio.

```
[Import]
internal IVsEditorAdaptersFactoryService AdapterService = null;
[Import]
internal ICompletionBroker CompletionBroker { get; set; }
[Import]
internal SVsServiceProvider ServiceProvider { get; set; }
```

```
<Import()>
Friend AdapterService As IVsEditorAdaptersFactoryService = Nothing
<Import()>
Friend Property CompletionBroker() As ICompletionBroker
<Import()>
Friend Property ServiceProvider() As SVsServiceProvider
```

5. Implemente [VsTextViewCreated](#) el método para crear instancias del controlador de comandos.

```
public void VsTextViewCreated(IVsTextView textViewAdapter)
{
    ITextView textView = AdapterService.GetWpfTextView(textViewAdapter);
    if (textView == null)
        return;

    Func<TestCompletionCommandHandler> createCommandHandler = delegate() { return new
    TestCompletionCommandHandler(textViewAdapter, textView, this); };
    textView.Properties.GetOrCreateSingletonProperty(createCommandHandler);
}
```

```
Public Sub VsTextViewCreated(ByVal textViewAdapter As IVsTextView) Implements
IVsTextViewCreationListener.VsTextViewCreated
    Dim textView As ITextView = AdapterService.GetWpfTextView(textViewAdapter)
    If textView Is Nothing Then
        Return
    End If

    Dim createCommandHandler As Func(Of TestCompletionCommandHandler) = Function() New
    TestCompletionCommandHandler(textViewAdapter, textView, Me)
    textView.Properties.GetOrCreateSingletonProperty(createCommandHandler)
End Sub
```

Implementar el controlador de comandos de finalización

Dado que la finalización de instrucciones se [IOleCommandTarget](#) desencadena mediante pulsaciones de teclas, debe implementar la interfaz para recibir y procesar las pulsaciones de teclas que desencadenan, confirman y descartan la sesión de finalización.

Para implementar el controlador de comandos de finalización

1. Agregue una `TestCompletionCommandHandler` clase denominada [IOleCommandTarget](#) que implemente:

```
internal class TestCompletionCommandHandler : IOleCommandTarget
```

```
Friend Class TestCompletionCommandHandler
    Implements IOleCommandTarget
```

2. Agregue campos privados para el siguiente controlador de comandos (al que pasa el comando), la vista de texto, el proveedor de controlador de comandos (que permite el acceso a varios servicios) y una sesión de

finalización:

```
private IOleCommandTarget m_nextCommandHandler;
private ITextView m_textView;
private TestCompletionHandlerProvider m_provider;
private ICompletionSession m_session;
```

```
Private m_nextCommandHandler As IOleCommandTarget
Private m_textView As ITextView
Private m_provider As TestCompletionHandlerProvider
Private m_session As ICompletionSession
```

3. Agregue un constructor que establezca la vista de texto y los campos del proveedor y agregue el comando a la cadena de comandos:

```
internal TestCompletionCommandHandler(IVsTextView textViewAdapter, ITextView textView,
TestCompletionHandlerProvider provider)
{
    this.m_textView = textView;
    this.m_provider = provider;

    //add the command to the command chain
    textViewAdapter.AddCommandFilter(this, out m_nextCommandHandler);
}
```

```
Friend Sub New(ByVal textViewAdapter As IVsTextView, ByVal textView As ITextView, ByVal provider As
TestCompletionHandlerProvider)
    Me.m_textView = textView
    Me.m_provider = provider

    'add the command to the command chain
    textViewAdapter.AddCommandFilter(Me, m_nextCommandHandler)
End Sub
```

4. Implemente [QueryStatus](#) el método pasando el comando a lo largo:

```
public int QueryStatus(ref Guid pguidCmdGroup, uint cCmds, OLECMD[] prgCmds, IntPtr pCmdText)
{
    return m_nextCommandHandler.QueryStatus(ref pguidCmdGroup, cCmds, prgCmds, pCmdText);
}
```

```
Public Function QueryStatus(ByRef pguidCmdGroup As Guid, ByVal cCmds As UInteger, ByVal prgCmds() As
OLECMD, ByVal pCmdText As IntPtr) As Integer Implements IOleCommandTarget.QueryStatus
    Return m_nextCommandHandler.QueryStatus(pguidCmdGroup, cCmds, prgCmds, pCmdText)
End Function
```

5. Implemente el método [Exec](#). Cuando este método recibe una pulsación de tecla, debe hacer una de estas cosas:

- Permita que el carácter se escriba en el búfer y, a continuación, desencadene o filtre la finalización. (Los caracteres de impresión hacen esto.)
- Confirme la finalización, pero no permita que el carácter se escriba en el búfer. (Espacio en blanco, **Tabuladory Entrar** hacen esto cuando se muestra una sesión de finalización.)
- Permita que el comando se pase al siguiente controlador. (Todos los demás comandos.)

Dado que este método [IsInAutomationFunction](#) puede mostrar la interfaz de usuario, llame para asegurarse de que no se llama en un contexto de automatización:

```
public int Exec(ref Guid pguidCmdGroup, uint nCmdID, uint nCmdexcopt, IntPtr pvaIn, IntPtr
pvaOut)
{
    if (VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider))
    {
        return m_nextCommandHandler.Exec(ref pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn, pvaOut);
    }
    //make a copy of this so we can look at it after forwarding some commands
    uint commandID = nCmdID;
    char typedChar = char.MinValue;
    //make sure the input is a char before getting it
    if (pguidCmdGroup == VSConstants.VSStd2K && nCmdID ==
(uint)VSConstants.VSStd2KCmdID.TYPECHAR)
    {
        typedChar = (char)(ushort)Marshal.GetObjectForNativeVariant(pvaIn);
    }

    //check for a commit character
    if (nCmdID == (uint)VSConstants.VSStd2KCmdID.RETURN
        || nCmdID == (uint)VSConstants.VSStd2KCmdID.TAB
        || (char.IsWhiteSpace(typedChar) || char.IsPunctuation(typedChar)))
    {
        //check for a a selection
        if (m_session != null && !m_session.IsDismissed)
        {
            //if the selection is fully selected, commit the current session
            if (m_session.SelectedCompletionSet.SelectionStatus.IsSelected)
            {
                m_session.Commit();
                //also, don't add the character to the buffer
                return VSConstants.S_OK;
            }
            else
            {
                //if there is no selection, dismiss the session
                m_session.Dismiss();
            }
        }
    }

    //pass along the command so the char is added to the buffer
    int retVal = m_nextCommandHandler.Exec(ref pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn,
pvaOut);
    bool handled = false;
    if (!typedChar.Equals(char.MinValue) && char.IsLetterOrDigit(typedChar))
    {
        if (m_session == null || m_session.IsDismissed) // If there is no active session, bring
up completion
        {
            this.TriggerCompletion();
            m_session.Filter();
        }
        else //the completion session is already active, so just filter
        {
            m_session.Filter();
        }
        handled = true;
    }
    else if (commandID == (uint)VSConstants.VSStd2KCmdID.BACKSPACE //redo the filter if there
is a deletion
        || commandID == (uint)VSConstants.VSStd2KCmdID.DELETE)
    {
        if (m_session != null && !m_session.IsDismissed)
            m_session.Filter();
    }
}
```

```

        handled = true;
    }
    if (handled) return VSConstants.S_OK;
    return retVal;
}

```

```

Public Function Exec(ByRef pguidCmdGroup As Guid, ByVal nCmdID As UInteger, ByVal nCmdexcopt As
UInteger, ByVal pvaIn As IntPtr, ByVal pvaOut As IntPtr) As Integer Implements
IOleCommandTarget.Exec
    If VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider) Then
        Return m_nextCommandHandler.Exec(pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn, pvaOut)
    End If
    'make a copy of this so we can look at it after forwarding some commands
    Dim commandID As UInteger = nCmdID
    Dim typedChar As Char = Char.MinValue
    'make sure the input is a char before getting it
    If pguidCmdGroup = VSConstants.VSStd2K AndAlso nCmdID =
CUInt(VSConstants.VSStd2KCmdID.TYPECHAR) Then
        typedChar = CChar(ChrW(CUShort(Marshal.GetObjectForNativeVariant(pvaIn))))
    End If

    'check for a commit character
    If nCmdID = CUInt(VSConstants.VSStd2KCmdID.RETURN) OrElse nCmdID =
CUInt(VSConstants.VSStd2KCmdID.TAB) OrElse (Char.IsWhiteSpace(typedChar) OrElse
Char.IsPunctuation(typedChar)) Then
        'check for a a selection
        If m_session IsNot Nothing AndAlso (Not m_session.IsDismissed) Then
            'if the selection is fully selected, commit the current session
            If m_session.SelectedCompletionSet.SelectionStatus.IsSelected Then
                m_session.Commit()
                'also, don't add the character to the buffer
                Return VSConstants.S_OK
            Else
                'if there is no selection, dismiss the session
                m_session.Dismiss()
            End If
        End If
    End If

    'pass along the command so the char is added to the buffer
    Dim retVal As Integer = m_nextCommandHandler.Exec(pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn,
pvaOut)
    Dim handled As Boolean = False
    If (Not typedChar.Equals(Char.MinValue)) AndAlso Char.IsLetterOrDigit(typedChar) Then
        If m_session Is Nothing OrElse m_session.IsDismissed Then ' If there is no active
session, bring up completion
            Me.TriggerCompletion()
            m_session.Filter()
        Else 'the completion session is already active, so just filter
            m_session.Filter()
        End If
        handled = True
    ElseIf commandID = CUInt(VSConstants.VSStd2KCmdID.BACKSPACE) OrElse commandID =
CUInt(VSConstants.VSStd2KCmdID.DELETE) Then 'redo the filter if there is a deletion
        If m_session IsNot Nothing AndAlso (Not m_session.IsDismissed) Then
            m_session.Filter()
        End If
        handled = True
    End If
    If handled Then
        Return VSConstants.S_OK
    End If
    Return retVal
End Function

```

6. Este código es un método privado que desencadena la sesión de finalización:

```

private bool TriggerCompletion()
{
    //the caret must be in a non-projection location
    SnapshotPoint? caretPoint =
        m_textView.Caret.Position.Point.GetPoint(
            textBuffer => (!textBuffer.ContentType.IsOfType("projection")), PositionAffinity.Predecessor);
    if (!caretPoint.HasValue)
    {
        return false;
    }

    m_session = m_provider.CompletionBroker.CreateCompletionSession
        (m_textView,
            caretPoint.Value.Snapshot.CreateTrackingPoint(caretPoint.Value.Position,
                PointTrackingMode.Positive),
            true);

    //subscribe to the Dismissed event on the session
    m_session.Dismissed += this.OnSessionDismissed;
    m_session.Start();

    return true;
}

```

```

Private Function TriggerCompletion() As Boolean
    'the caret must be in a non-projection location
    Dim caretPoint As SnapshotPoint? = m_textView.Caret.Position.Point.GetPoint(Function(textBuffer)
        ((Not textBuffer.ContentType.IsOfType("projection"))), PositionAffinity.Predecessor)
    If Not caretPoint.HasValue Then
        Return False
    End If

    m_session = m_provider.CompletionBroker.CreateCompletionSession(m_textView,
        caretPoint.Value.Snapshot.CreateTrackingPoint(caretPoint.Value.Position, PointTrackingMode.Positive),
        True)

    'subscribe to the Dismissed event on the session
    AddHandler m_session.Dismissed, AddressOf OnSessionDismissed
    m_session.Start()

    Return True
End Function

```

7. El siguiente ejemplo es un método privado [Dismissed](#) que cancela las suscripciones del evento:

```

private void OnSessionDismissed(object sender, EventArgs e)
{
    m_session.Dismissed -= this.OnSessionDismissed;
    m_session = null;
}

```

```

Private Sub OnSessionDismissed(ByVal sender As Object, ByVal e As EventArgs)
    RemoveHandler m_session.Dismissed, AddressOf OnSessionDismissed
    m_session = Nothing
End Sub

```

Compile y pruebe el código

Para probar este código, compile la solución CompletionTest y ejecútela en la instancia experimental.

Para compilar y probar la solución CompletionTest

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Cree un archivo de texto y escriba texto que incluya la palabra "añadir".
4. A medida que escribe primero "a" y luego "d", debe aparecer una lista que contenga "adición" y "adaptación". Observe que la adición está seleccionada. Cuando escribe otra "d", la lista debe contener solo "adición", que ahora está seleccionada. Puede confirmar "adición" pulsando la tecla **Barra espaciadora**, **Tabulador** o **Intro**, o descartar la lista escribiendo Esc o cualquier otra tecla.

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Implementar fragmentos de código

06/04/2020 • 22 minutes to read • [Edit Online](#)

Puede crear fragmentos de código e incluirlos en una extensión de editor para que los usuarios de la extensión puedan agregarlos a su propio código.

Un fragmento de código es un fragmento de código u otro texto que se puede incorporar en un archivo. Para ver todos los fragmentos de código que se han registrado para determinados lenguajes de programación, en el menú **Herramientas**, haga clic en Administrador de **fragmentos** de código . Para insertar un fragmento de código en un archivo, haga clic con el botón derecho en el lugar donde desee el fragmento de código, haga clic en Insertar fragmento de código o **Rodear con**, busque el fragmento de código que desee y, a continuación, haga doble clic en él. Pulse **Tab** o **Shift+Tab** para modificar las partes relevantes del fragmento de código y, a continuación, pulse **Intro** o **Esc** para aceptarlo. Para obtener más información, consulte [Fragmentos](#) de código .

Un fragmento de código está contenido en un archivo XML que tiene la extensión de nombre de archivo .snippet*. Un fragmento de código puede contener campos que se resaltan después de insertar el fragmento de código para que el usuario pueda encontrarlos y cambiarlos. Un archivo de fragmento de código también proporciona información para el Administrador de **fragmentos** de código para que pueda mostrar el nombre del fragmento de código en la categoría correcta. Para obtener información sobre el esquema de fragmento de código, vea Referencia de esquema de [fragmentos](#) de código .

Este tutorial enseña cómo realizar estas tareas:

1. Crear y registrar fragmentos de código para un idioma específico.
2. Agregue el comando **Insertar fragmento de código** a un menú contextual.
3. Implementar la expansión de fragmentos de código.

Este tutorial se basa en [Tutorial: Mostrar finalización](#) de instrucciones .

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear y registrar fragmentos de código

Normalmente, los fragmentos de código están asociados a un servicio de lenguaje registrado. Sin embargo, no es [LanguageService](#) necesario implementar a para registrar fragmentos de código. En su lugar, solo especifique un GUID en el [ProvideLanguageCodeExpansionAttribute](#) archivo de índice de fragmento de código y, a continuación, use el mismo GUID en el que agregue al proyecto.

En los pasos siguientes se muestra cómo crear fragmentos de código y asociarlos a un GUID específico.

1. Cree la siguiente estructura de directorios:

```
%InstallDir%-TestSnippets-Snippets-1033\
```

donde *%InstallDir%* es la carpeta de instalación de Visual Studio. (Aunque esta ruta de acceso se utiliza normalmente para instalar fragmentos de código, puede especificar cualquier ruta de acceso.)

2. En la carpeta .1033, cree un archivo .xml y asímóquele **TestSnippets.xml**. (Aunque este nombre se utiliza

normalmente para un archivo de índice de fragmento de código, puede especificar cualquier nombre siempre que tenga una extensión de nombre de archivo *.xml*.) Agregue el texto siguiente y, a continuación, elimine el GUID de marcador de posición y agregue el suyo propio.

```
<?xml version="1.0" encoding="utf-8" ?>
<SnippetCollection>
  <Language Lang="TestSnippets" Guid="{00000000-0000-0000-0000-000000000000}">
    <SnippetDir>
      <OnOff>On</OnOff>
      <Installed>true</Installed>
      <Locale>1033</Locale>
      <DirPath>%InstallRoot%\TestSnippets\Snippets\%LCID%\</DirPath>
      <LocalizedName>Snippets</LocalizedName>
    </SnippetDir>
  </Language>
</SnippetCollection>
```

3. Cree un archivo en la carpeta de fragmentos de código, asígnelo **test** `.snippet` y, a continuación, agregue el texto siguiente:

```
<?xml version="1.0" encoding="utf-8" ?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Test replacement fields</Title>
      <Shortcut>test</Shortcut>
      <Description>Code snippet for testing replacement fields</Description>
      <Author>MSIT</Author>
      <SnippetTypes>
        <SnippetType>Expansion</SnippetType>
      </SnippetTypes>
    </Header>
    <Snippet>
      <Declarations>
        <Literal>
          <ID>param1</ID>
          <ToolTip>First field</ToolTip>
          <Default>first</Default>
        </Literal>
        <Literal>
          <ID>param2</ID>
          <ToolTip>Second field</ToolTip>
          <Default>second</Default>
        </Literal>
      </Declarations>
      <References>
        <Reference>
          <Assembly>System.Windows.Forms.dll</Assembly>
        </Reference>
      </References>
      <Code Language="TestSnippets">
        <![CDATA[MessageBox.Show("$param1$");
        MessageBox.Show("$param2$");]]>
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

Los pasos siguientes muestran cómo registrar los fragmentos de código.

Para registrar fragmentos de código para un GUID específico

1. Abra el proyecto **CompletionTest**. Para obtener información sobre cómo crear este proyecto, vea [Tutorial: Mostrar finalización](#) de instrucciones .

2. En el proyecto, agregue referencias a los ensamblados siguientes:

- Microsoft.VisualStudio.TextManager.Interop
- Microsoft.VisualStudio.TextManager.Interop.8.0
- microsoft.msxml

3. En el proyecto, abra el archivo **source.extension.vsixmanifest**.

4. Asegúrese de que la pestaña **Activos** contiene un tipo de contenido **VsPackage** y que **Project** se establece en el nombre del proyecto.

5. Seleccione el proyecto **CompletionTest** y, en la ventana **Propiedades**, establezca **Generar archivo Pkgdef** en **true**. Guarde el proyecto.

6. Agregue una `SnippetUtilities` clase estática al proyecto.

```
static class SnippetUtilities
```

```
Module SnippetUtilities
```

7. En la clase `SnippetUtilities`, defina un GUID y asíbse con el valor que usó en el archivo *SnippetsIndex.xml*.

```
internal const string LanguageServiceGuidStr = "00000000-0000-0000-0000-00000000";
```

```
Friend Const LanguageServiceGuidStr As String = "00000000-0000-0000-0000-00000000"
```

8. Agregue `ProvideLanguageCodeExpansionAttribute` el `TestCompletionHandler` a la clase. Este atributo se puede agregar a cualquier clase pública o interna (no estática) del proyecto. (Es posible que `using` tenga que agregar una directiva para el `Microsoft.VisualStudio.Shell` espacio de nombres.)

```
[ProvideLanguageCodeExpansion(  
    SnippetUtilities.LanguageServiceGuidStr,  
    "TestSnippets", //the language name  
    0,               //the resource id of the language  
    "TestSnippets", //the language ID used in the .snippet files  
    @"%InstallRoot%\TestSnippets\Snippets\%LCID%\TestSnippets.xml",  
        //the path of the index file  
    SearchPaths = @"%InstallRoot%\TestSnippets\Snippets\%LCID%",  
    ForceCreateDirs = @"%InstallRoot%\TestSnippets\Snippets\%LCID%")]  
internal class TestCompletionCommandHandler : IOleCommandTarget
```

```
<ProvideLanguageCodeExpansion(  
    SnippetUtilities.LanguageServiceGuidStr,  
    "TestSnippets",  
    0,  
    "TestSnippets",  
    "%InstallRoot%\TestSnippets\Snippets\%LCID%\SnippetsIndex.xml",  
    SearchPaths:="%InstallRoot%\TestSnippets\Snippets\%LCID%",  
    ForceCreateDirs:="%InstallRoot%\TestSnippets\Snippets\%LCID%")>  
    Friend Class TestCompletionCommandHandler  
        Implements IOleCommandTarget
```

9. Compile y ejecute el proyecto. En la instancia experimental de Visual Studio que se inicia cuando se ejecuta

el proyecto, el fragmento de código que acaba de registrar debe mostrarse en el Administrador de fragmentos de código en el lenguaje **TestSnippets**.

Agregue el comando Insertar fragmento de código al menú contextual

El comando **Insertar fragmento** de código no se incluye en el menú contextual de un archivo de texto. Por lo tanto, debe habilitar el comando.

Para agregar el comando Insertar fragmento de código al menú contextual

1. Abra `TestCompletionCommandHandler` el archivo de clase.

Dado que esta `IOleCommandTarget` clase implementa, puede activar `QueryStatus` el comando Insertar **fragmento** de código en el método. Antes de habilitar el comando, compruebe que no se llama a este método dentro de una función de automatización porque cuando se hace clic en el comando **Insertar fragmento** de código, muestra la interfaz de usuario (UI) del selector de fragmentos de código.

```
public int QueryStatus(ref Guid pguidCmdGroup, uint cCmds, OLECMD[] prgCmds, IntPtr pCmdText)
{
    if (!VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider))
    {
        if (pguidCmdGroup == VSConstants.VSStd2K && cCmds > 0)
        {
            // make the Insert Snippet command appear on the context menu
            if ((uint)prgCmds[0].cmdID == (uint)VSConstants.VSStd2KCmdID.INSERTSNIPPET)
            {
                prgCmds[0].cmdf = (int)Constants.MSOCMDF_ENABLED | (int)Constants.MSOCMDF_SUPPORTED;
                return VSConstants.S_OK;
            }
        }
    }

    return m_nextCommandHandler.QueryStatus(ref pguidCmdGroup, cCmds, prgCmds, pCmdText);
}
```

```
Public Function QueryStatus(ByRef pguidCmdGroup As Guid, ByVal cCmds As UInteger, ByVal prgCmds As OLECMD(), ByVal pCmdText As IntPtr) As Integer Implements IOleCommandTarget.QueryStatus
    If Not VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider) Then
        If pguidCmdGroup = VSConstants.VSStd2K AndAlso cCmds > 0 Then
            ' make the Insert Snippet command appear on the context menu
            If CUInt(prgCmds(0).cmdID) = CUInt(VSConstants.VSStd2KCmdID.INSERTSNIPPET) Then
                prgCmds(0).cmdf = CInt(Constants.MSOCMDF_ENABLED) Or CInt(Constants.MSOCMDF_SUPPORTED)
                Return VSConstants.S_OK
            End If
        End If
    End If

    Return m_nextCommandHandler.QueryStatus(pguidCmdGroup, cCmds, prgCmds, pCmdText)
End Function
```

2. Compile y ejecute el proyecto. En la instancia experimental, abra un archivo que tenga la extensión de nombre de archivo `.zzzy`, a continuación, haga clic con el botón derecho en cualquier lugar. El comando **Insertar fragmento** de código debe aparecer en el menú contextual.

Implementar la expansión de fragmentos de código en la interfaz de usuario del selector de fragmentos de código

En esta sección se muestra cómo implementar la expansión de fragmentos de código para que se muestre la interfaz de usuario del selector de fragmentos de código cuando se haga clic en **Insertar fragmento** de código en el menú contextual. Un fragmento de código también se expande cuando un usuario escribe el acceso directo

de fragmento de código y, a continuación, presiona **Tab**.

Para mostrar la interfaz de usuario del selector de [Exec](#) fragmentos de código y habilitar la navegación y la aceptación del fragmento de código posterior a la inserción, utilice el método. El método controla la [OnItemChosen](#) inserción en sí.

La implementación de la [Microsoft.VisualStudio.TextManager.Interop](#) expansión de fragmentos de código utiliza interfaces heredadas. Al traducir de las clases de editor actuales al código heredado, recuerde que las interfaces heredadas usan una combinación de números de línea y números de columna para especificar ubicaciones en un búfer de texto, pero las clases actuales usan un índice. Por lo tanto, si un búfer tiene tres líneas cada una de las cuales tiene 10 caracteres (más una nueva línea, que cuenta como un carácter), el cuarto carácter de la tercera línea está en la posición 27 de la implementación actual, pero está en la línea 2, posición 3 en la implementación anterior.

Para implementar la expansión de fragmentos de código

1. Para el archivo `TestCompletionCommandHandler` que contiene la `using` clase, agregue las siguientes directivas.

```
using Microsoft.VisualStudio.Text.Operations;
using MSXML;
using System.ComponentModel.Composition;
```

```
Imports Microsoft.VisualStudio.Text.Operations
Imports MSXML
```

2. Haga `TestCompletionCommandHandler` que la [IVsExpansionClient](#) clase implemente la interfaz.

```
internal class TestCompletionCommandHandler : IOleCommandTarget, IVsExpansionClient
```

```
<ProvideLanguageCodeExpansion(
SnippetUtilities.LanguageServiceGuidStr,
"TestSnippets",
0,
"TestSnippets",
"%InstallRoot%\TestSnippets\Snippets\%LCID%\SnippetsIndex.xml",
SearchPaths:="%InstallRoot%\TestSnippets\Snippets\%LCID%",
ForceCreateDirs:="%InstallRoot%\TestSnippets\Snippets\%LCID%")>
Friend Class TestCompletionCommandHandler
    Implements IOleCommandTarget
    Implements IVsExpansionClient
```

3. En `TestCompletionCommandHandlerProvider` la clase, [ITextStructureNavigatorSelectorService](#) importe el archivo

```
[Import]
internal ITextStructureNavigatorSelectorService NavigatorService { get; set; }
```

```
<Import()>
Friend Property NavigatorService As ITextStructureNavigatorSelectorService
```

4. Agregue algunos campos privados para las [IVsTextView](#) interfaces de expansión de código y el archivo .

```
IVsTextView m_vsTextView;
IVsExpansionManager m_exManager;
IVsExpansionSession m_exSession;
```

```
Dim m_vsTextView As IVsTextView
Dim m_exManager As IVsExpansionManager
Dim m_exSession As IVsExpansionSession
```

5. En el constructor `TestCompletionCommandHandler` de la clase, establezca los siguientes campos.

```
internal TestCompletionCommandHandler(IVsTextView textViewAdapter, ITextView textView,
TestCompletionHandlerProvider provider)
{
    this.m_textView = textView;
    m_vsTextView = textViewAdapter;
    m_provider = provider;
    //get the text manager from the service provider
    IVsTextManager2 textManager =
    (IVsTextManager2)m_provider.ServiceProvider.GetService(typeof(SVsTextManager));
    textManager.GetExpansionManager(out m_exManager);
    m_exSession = null;

    //add the command to the command chain
    textViewAdapter.AddCommandFilter(this, out m_nextCommandHandler);
}
```

```
Friend Sub New(ByVal textViewAdapter As IVsTextView, ByVal textView As ITextView, ByVal provider As
TestCompletionHandlerProvider)
    Me.m_textView = textView
    Me.m_provider = provider
    Me.m_vsTextView = textViewAdapter

    Dim textManager As IVsTextManager2 =
    DirectCast(m_provider.ServiceProvider.GetService(GetType(SVsTextManager)), IVsTextManager2)
    textManager.GetExpansionManager(m_exManager)
    m_exSession = Nothing

    'add the command to the command chain
    textViewAdapter.AddCommandFilter(Me, m_nextCommandHandler)
End Sub
```

6. Para mostrar el selector de fragmentos de código cuando el [Exec](#) usuario hace clic en el comando Insertar **fragmento** de código, agregue el código siguiente al método. (Para que esta explicación `Exec()` sea más legible, no se muestra el código que se usa para la finalización de instrucciones; en su lugar, los bloques de código se agregan al método existente.) Agregue el siguiente bloque de código después del código que comprueba si hay un carácter.

```

//code previously written for Exec
if (pguidCmdGroup == VSConstants.VSStd2K && nCmdID == (uint)VSConstants.VSStd2KCmdID.TYPECHAR)
{
    typedChar = (char)(ushort)Marshal.GetObjectForNativeVariant(pvaIn);
}
//the snippet picker code starts here
if (nCmdID == (uint)VSConstants.VSStd2KCmdID.INSERTSNIPPET)
{
    IVsTextManager2 textManager =
    (IVsTextManager2)m_provider.ServiceProvider.GetService(typeof(SVsTextManager));

    textManager.GetExpansionManager(out m_exManager);

    m_exManager.InvokeInsertionUI(
        m_vsTextView,
        this, //the expansion client
        new Guid(SnippetUtilities.LanguageServiceGuidStr),
        null, //use all snippet types
        0, //number of types (0 for all)
        0, //ignored if iCountTypes == 0
        null, //use all snippet kinds
        0, //use all snippet kinds
        0, //ignored if iCountTypes == 0
        "TestSnippets", //the text to show in the prompt
        string.Empty); //only the ENTER key causes insert

    return VSConstants.S_OK;
}

```

```

'code previously written for Exec
If pguidCmdGroup = VSConstants.VSStd2K AndAlso nCmdID = CUInt(VSConstants.VSStd2KCmdID.TYPECHAR) Then
    typedChar = ChrW(CUShort(Marshal.GetObjectForNativeVariant(pvaIn)))
End If
'the snippet picker code starts here
If nCmdID = CUInt(VSConstants.VSStd2KCmdID.INSERTSNIPPET) Then
    Dim textManager As IVsTextManager2 =
    DirectCast(m_provider.ServiceProvider.GetService(GetType(SVsTextManager)), IVsTextManager2)

    textManager.GetExpansionManager(m_exManager)
    m_exManager.InvokeInsertionUI(
        m_vsTextView,
        Me,
        New Guid(SnippetUtilities.LanguageServiceGuidStr),
        Nothing,
        0,
        0,
        Nothing,
        0,
        0,
        "TestSnippets",
        String.Empty)
    Return VSConstants.S_OK
End If

```

- Si un fragmento de código tiene campos que se pueden navegar, la sesión de expansión se mantiene abierta hasta que se acepte explícitamente la expansión; si el fragmento de código no tiene campos, la sesión se cierra y se devuelve como `null` por el `InvokeInsertionUI` método. En `Exec` el método, después del código de interfaz de usuario del selector de fragmentos de código que agregé en el paso anterior, agregue el código siguiente para controlar la navegación de fragmentos de código (cuando el usuario presiona **Tab** o **Shift+Tab** después de la inserción del fragmento de código).

```

//the expansion insertion is handled in OnItemChosen
//if the expansion session is still active, handle tab/backtab/return/cancel
if (m_exSession != null)
{
    if (nCmdID == (uint)VSConstants.VSStd2KCmdID.BACKTAB)
    {
        m_exSession.GoToPreviousExpansionField();
        return VSConstants.S_OK;
    }
    else if (nCmdID == (uint)VSConstants.VSStd2KCmdID.TAB)
    {
        m_exSession.GoToNextExpansionField(0); //false to support cycling through all the fields
        return VSConstants.S_OK;
    }
    else if (nCmdID == (uint)VSConstants.VSStd2KCmdID.RETURN || nCmdID ==
(uint)VSConstants.VSStd2KCmdID.CANCEL)
    {
        if (m_exSession.EndCurrentExpansion(0) == VSConstants.S_OK)
        {
            m_exSession = null;
            return VSConstants.S_OK;
        }
    }
}
}

```

```

'the expansion insertion is handled in OnItemChosen
'if the expansion session is still active, handle tab/backtab/return/cancel
If m_exSession IsNot Nothing Then
    If nCmdID = CUInt(VSConstants.VSStd2KCmdID.BACKTAB) Then
        m_exSession.GoToPreviousExpansionField()
        Return VSConstants.S_OK
    ElseIf nCmdID = CUInt(VSConstants.VSStd2KCmdID.TAB) Then
        m_exSession.GoToNextExpansionField(0)
        'false to support cycling through all the fields
        Return VSConstants.S_OK
    ElseIf nCmdID = CUInt(VSConstants.VSStd2KCmdID.[RETURN]) OrElse nCmdID =
CUInt(VSConstants.VSStd2KCmdID.CANCEL) Then
        If m_exSession.EndCurrentExpansion(0) = VSConstants.S_OK Then
            m_exSession = Nothing
            Return VSConstants.S_OK
        End If
    End If
End If
End If

```

8. Para insertar el fragmento de código cuando el **Tab** usuario escribe el **Exec** acceso directo correspondiente y, a continuación, presiona Tab, agregue código al método. El método privado que inserta el fragmento de código se mostrará en un paso posterior. Agregue el código siguiente después del código de navegación que agregó en el paso anterior.


```

//neither an expansion session nor a completion session is open, but we got a tab, so check whether the
last word typed is a snippet shortcut
if (m_session == null && m_exSession == null && nCmdID == (uint)VSConstants.VSStd2KCmdID.TAB)
{
    //get the word that was just added
    CaretPosition pos = m_textView.Caret.Position;
    TextExtent word =
m_provider.NavigatorService.GetTextStructureNavigator(m_textView.TextBuffer).GetExtentOfWord(pos.Buffer
Position - 1); //use the position 1 space back
    string textString = word.Span.GetText(); //the word that was just added
    //if it is a code snippet, insert it, otherwise carry on
    if (InsertAnyExpansion(textString, null, null))
        return VSConstants.S_OK;
}

```

```

'neither an expansion session nor a completion session is open, but we got a tab, so check whether the
last word typed is a snippet shortcut
If m_session Is Nothing AndAlso m_exSession Is Nothing AndAlso nCmdID =
CUInt(VSConstants.VSStd2KCmdID.TAB) Then
    'get the word that was just added
    Dim pos As CaretPosition = m_textView.Caret.Position
    Dim word As TextExtent =
m_provider.NavigatorService.GetTextStructureNavigator(m_textView.TextBuffer).GetExtentOfWord(pos.Buffer
Position - 1)
    Dim textString As String = word.Span.GetText()
    'if it is a code snippet, insert it, otherwise carry on
    If InsertAnyExpansion(textString, Nothing, Nothing) Then
        Return VSConstants.S_OK
    End If
End If

```

9. Implemente los métodos de la [IVsExpansionClient](#) interfaz. En esta implementación, los [EndExpansion](#) únicos métodos de interés son y [OnItemChosen](#). Los otros métodos [S_OK](#) solo deben devolver .

```

public int EndExpansion()
{
    m_exSession = null;
    return VSConstants.S_OK;
}

public int FormatSpan(IVsTextLines pBuffer, TextSpan[] ts)
{
    return VSConstants.S_OK;
}

public int GetExpansionFunction(IXMLDOMNode xmlFunctionNode, string bstrFieldName, out
IVsExpansionFunction pFunc)
{
    pFunc = null;
    return VSConstants.S_OK;
}

public int IsValidKind(IVsTextLines pBuffer, TextSpan[] ts, string bstrKind, out int pfIsValidKind)
{
    pfIsValidKind = 1;
    return VSConstants.S_OK;
}

public int IsValidType(IVsTextLines pBuffer, TextSpan[] ts, string[] rgTypes, int iCountTypes, out int
pfIsValidType)
{
    pfIsValidType = 1;
    return VSConstants.S_OK;
}

public int OnAfterInsertion(IVsExpansionSession pSession)
{
    return VSConstants.S_OK;
}

public int OnBeforeInsertion(IVsExpansionSession pSession)
{
    return VSConstants.S_OK;
}

public int PositionCaretForEditing(IVsTextLines pBuffer, TextSpan[] ts)
{
    return VSConstants.S_OK;
}

```

```

Public Function EndExpansion() As Integer Implements IVsExpansionClient.EndExpansion
    m_exSession = Nothing
    Return VSConstants.S_OK
End Function

Public Function FormatSpan(ByVal pBuffer As IVsTextLines, ByVal ts As TextSpan()) As Integer Implements
IVsExpansionClient.FormatSpan
    Return VSConstants.S_OK
End Function

Public Function GetExpansionFunction(ByVal xmlFunctionNode As IXMLDOMNode, ByVal bstrFieldName As
String, ByRef pFunc As IVsExpansionFunction) As Integer Implements
IVsExpansionClient.GetExpansionFunction
    pFunc = Nothing
    Return VSConstants.S_OK
End Function

Public Function IsValidKind(ByVal pBuffer As IVsTextLines, ByVal ts As TextSpan(), ByVal bstrKind As
String, ByRef pfIsValidKind As Integer) As Integer Implements IVsExpansionClient.IsValidKind
    pfIsValidKind = 1
    Return VSConstants.S_OK
End Function

Public Function IsValidType(ByVal pBuffer As IVsTextLines, ByVal ts() As TextSpan, ByVal rgTypes() As
String, ByVal iCountTypes As Integer, ByRef pfIsValidType As Integer) As Integer Implements
IVsExpansionClient.IsValidType
    pfIsValidType = 1
    Return VSConstants.S_OK
End Function

Public Function OnAfterInsertion(ByVal pSession As IVsExpansionSession) As Integer Implements
IVsExpansionClient.OnAfterInsertion
    Return VSConstants.S_OK
End Function

Public Function OnBeforeInsertion(ByVal pSession As IVsExpansionSession) As Integer Implements
IVsExpansionClient.OnBeforeInsertion
    Return VSConstants.S_OK
End Function

Public Function PositionCaretForEditing(ByVal pBuffer As IVsTextLines, ByVal ts As TextSpan()) As
Integer Implements IVsExpansionClient.PositionCaretForEditing
    Return VSConstants.S_OK
End Function

```

10. Implemente el método [OnItemChosen](#). El método auxiliar que realmente inserta las expansiones se trata en un paso posterior. La [TextSpan](#) información de línea y columna proporciona, que puede obtener de la [IVsTextView](#).

```

public int OnItemChosen(string pszTitle, string pszPath)
{
    InsertAnyExpansion(null, pszTitle, pszPath);
    return VSConstants.S_OK;
}

```

```

Public Function OnItemChosen(ByVal pszTitle As String, ByVal pszPath As String) As Integer Implements
IVsExpansionClient.OnItemChosen
    InsertAnyExpansion(Nothing, pszTitle, pszPath)
    Return VSConstants.S_OK
End Function

```

11. El siguiente método privado inserta un fragmento de código, basado en el acceso directo o en el título y la

ruta de acceso. A continuación, [InsertNamedExpansion](#) llama al método con el fragmento de código.

```
private bool InsertAnyExpansion(string shortcut, string title, string path)
{
    //first get the location of the caret, and set up a TextSpan
    int endColumn, startLine;
    //get the column number from the IVsTextView, not the ITextView
    m_vsTextView.GetCaretPos(out startLine, out endColumn);

    TextSpan addSpan = new TextSpan();
    addSpan.iStartIndex = endColumn;
    addSpan.iEndIndex = endColumn;
    addSpan.iStartLine = startLine;
    addSpan.iEndLine = startLine;

    if (shortcut != null) //get the expansion from the shortcut
    {
        //reset the TextSpan to the width of the shortcut,
        //because we're going to replace the shortcut with the expansion
        addSpan.iStartIndex = addSpan.iEndIndex - shortcut.Length;

        m_exManager.GetExpansionByShortcut(
            this,
            new Guid(SnippetUtilities.LanguageServiceGuidStr),
            shortcut,
            m_vsTextView,
            new TextSpan[] { addSpan },
            0,
            out path,
            out title);
    }

    if (title != null && path != null)
    {
        IVsTextLines textLines;
        m_vsTextView.GetBuffer(out textLines);
        IVsExpansion bufferExpansion = (IVsExpansion)textLines;

        if (bufferExpansion != null)
        {
            int hr = bufferExpansion.InsertNamedExpansion(
                title,
                path,
                addSpan,
                this,
                new Guid(SnippetUtilities.LanguageServiceGuidStr),
                0,
                out m_exSession);
            if (VSConstants.S_OK == hr)
            {
                return true;
            }
        }
    }

    return false;
}
```

```

Private Function InsertAnyExpansion(ByVal shortcut As String, ByVal title As String, ByVal path As
String) As Boolean
    Dim endColumn As Integer, startLine As Integer
    'get the column number from the IVsTextView, not the ITextView
    m_vsTextView.GetCaretPos(startLine, endColumn)

    Dim addSpan As New TextSpan()
    addSpan.iStartIndex = endColumn
    addSpan.iEndIndex = endColumn
    addSpan.iStartLine = startLine
    addSpan.iEndLine = startLine

    'get the expansion from the shortcut
    If shortcut IsNot Nothing Then
        'reset the TextSpan to the width of the shortcut, because we're going to replace the shortcut
        with the expansion
        addSpan.iStartIndex = addSpan.iEndIndex - shortcut.Length

        m_exManager.GetExpansionByShortcut(
            Me,
            New Guid(SnippetUtilities.LanguageServiceGuidStr),
            shortcut,
            m_vsTextView,
            New TextSpan() {addSpan},
            0,
            path,
            title)
    End If
    If title IsNot Nothing AndAlso path IsNot Nothing Then
        Dim textLines As IVsTextLines = Nothing
        m_vsTextView.GetBuffer(textLines)
        Dim bufferExpansion As IVsExpansion = DirectCast(textLines, IVsExpansion)

        If bufferExpansion IsNot Nothing Then
            Dim hr As Integer = bufferExpansion.InsertNamedExpansion(
                title,
                path,
                addSpan,
                Me,
                New Guid(SnippetUtilities.LanguageServiceGuidStr),
                0,
                m_exSession)
            If VSConstants.S_OK = hr Then
                Return True
            End If
        End If
    End If
    Return False
End Function

```

Crear y probar la expansión del fragmento de código

Puede probar si la expansión de fragmentos de código funciona en el proyecto.

1. Compile la solución. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
2. Abra un archivo de texto y escriba algo de texto.
3. Haga clic con el botón derecho en algún lugar del texto y, a continuación, haga clic en **Insertar fragmento de código**.
4. La interfaz de usuario del selector de fragmentos de código debe aparecer con una ventana emergente que diga Campos de **reemplazo de prueba**. Haga doble clic en la ventana emergente.

Se debe insertar el siguiente fragmento de código.

```
MessageBox.Show("first");  
MessageBox.Show("second");
```

No pulse **Intro** o **Esc**.

5. Pulse **Tab** y **Shift+Tab** para alternar entre "first" y "second".
6. Acepte la inserción pulsando **Intro** o **Esc**.
7. En una parte diferente del texto, escriba "test" y, a continuación, presione **Tab**. Dado que "test" es el acceso directo de fragmento de código, el fragmento de código debe insertarse de nuevo.

Pasos siguientes

Tutorial: Mostrar sugerencias de bombillas

06/04/2020 • 13 minutes to read • [Edit Online](#)

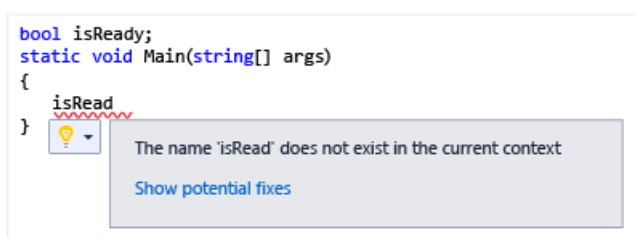
Las bombillas son iconos en el editor de Visual Studio que se expanden para mostrar un conjunto de acciones, por ejemplo, correcciones de problemas identificados por los analizadores de código integrados o la refactorización de código.

En los editores de Visual C y Visual Basic, también puede usar la plataforma de compilador de .NET ("Roslyn") para escribir y empaquetar sus propios analizadores de código con acciones que muestran bombillas automáticamente. Para obtener más información, consulte:

- [Cómo: Escribir una corrección de diagnóstico y código de C-](#)
- [Cómo: Escribir un diagnóstico de Visual Basic y una corrección de código](#)

Otros lenguajes como C++ también proporcionan bombillas para algunas acciones rápidas, como, una sugerencia para crear una implementación de código auxiliar de esa función.

Así es como se ve una bombilla. En un proyecto de Visual Basic o Visual C, aparece un ondulado rojo bajo un nombre de variable cuando no es válido. Si pasa el ratón sobre el identificador no válido, aparecerá una bombilla cerca del cursor.



Si hace clic en la flecha hacia abajo junto a la bombilla, aparecerá un conjunto de acciones sugeridas, junto con una vista previa de la acción seleccionada. En este caso, muestra los cambios realizados en el código si ejecuta la acción.



Puede utilizar bombillas para proporcionar sus propias acciones sugeridas. Por ejemplo, podría proporcionar acciones para mover llaves de apertura a una nueva línea o moverlas al final de la línea anterior. En el siguiente tutorial se muestra cómo crear una bombilla que aparece en la palabra actual y tiene dos acciones sugeridas: **Convertir a mayúsculas** y **Convertir a minúsculas**.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como

una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto de Managed Extensibility Framework (MEF)

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `LightBulbTest` nombre a la solución .
2. Agregue una plantilla de elemento **Clasificador** de editor al proyecto. Para obtener más información, consulte [Crear una extensión con una plantilla de elemento de editor](#).
3. Elimine los archivos de clase existentes.
4. Agregue la siguiente referencia al proyecto y `False` establezca **Copiar local** en:

Microsoft.VisualStudio.Language.Intellisense

5. Agregue un nuevo archivo de clase y asíse: **LightBulbTest**.
6. Agregue lo siguiente mediante directivas:

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.VisualStudio.Language.Intellisense;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Operations;
using Microsoft.VisualStudio.Utilities;
using System.ComponentModel.Composition;
using System.Threading;
```

Implementar el proveedor de fuente de bombilla

1. En el *LightBulbTest.cs* archivo de clase, elimine la clase `LightBulbTest`. Agregue una clase denominada **TestSuggestedActionsSourceProvider** que implemente [ISuggestedActionsSourceProvider](#). Expórtelo con un nombre [ContentTypeAttribute](#) de acciones sugeridas de prueba y un de "texto".

```
[Export(typeof(ISuggestedActionsSourceProvider))]
[Name("Test Suggested Actions")]
[ContentType("text")]
internal class TestSuggestedActionsSourceProvider : ISuggestedActionsSourceProvider
```

2. Dentro de la clase [ITextStructureNavigatorSelectorService](#) de proveedor de origen, importe y agréguelo como una propiedad.

```
[Import(typeof(ITextStructureNavigatorSelectorService))]
internal ITextStructureNavigatorSelectorService NavigatorService { get; set; }
```

3. Implemente [CreateSuggestedActionsSource](#) el método [ISuggestedActionsSource](#) para devolver un objeto. La fuente se describe en la siguiente sección.


```
public ISuggestedActionsSource CreateSuggestedActionsSource(ITextView textView, ITextBuffer textBuffer)
{
    if (textBuffer == null || textView == null)
    {
        return null;
    }
    return new TestSuggestedActionsSource(this, textView, textBuffer);
}
```

Implementar el ISuggestedActionSource

La fuente de acción sugerida es responsable de recopilar el conjunto de acciones sugeridas y agregarlas en el contexto correcto. En este caso, el contexto es la palabra actual y las acciones sugeridas son

UpperCaseSuggestedAction y **LowerCaseSuggestedAction**, que se describe en la sección siguiente.

1. Agregue una clase **TestSuggestedActionsSource** que implemente [ISuggestedActionsSource](#).

```
internal class TestSuggestedActionsSource : ISuggestedActionsSource
```

2. Agregue campos privados de solo lectura para el proveedor de origen de acciones sugerido, el búfer de texto y la vista de texto.

```
private readonly TestSuggestedActionsSourceProvider m_factory;
private readonly ITextBuffer m_textBuffer;
private readonly ITextView m_textView;
```

3. Agregue un constructor que establezca los campos privados.

```
public TestSuggestedActionsSource(TestSuggestedActionsSourceProvider
testSuggestedActionsSourceProvider, ITextView textView, ITextBuffer textBuffer)
{
    m_factory = testSuggestedActionsSourceProvider;
    m_textBuffer = textBuffer;
    m_textView = textView;
}
```

4. Agregue un método privado que devuelva la palabra que está actualmente debajo del cursor. El siguiente método examina la ubicación actual del cursor y solicita al navegador de estructura de texto la extensión de la palabra. Si el cursor está en [TextExtent](#) una palabra, se devuelve en el parámetro out; de lo `out` contrario, el parámetro es `null` y el método devuelve `false`.

```
private bool TryGetWordUnderCaret(out TextExtent wordExtent)
{
    ITextCaret caret = m_textView.Caret;
    SnapshotPoint point;

    if (caret.Position.BufferPosition > 0)
    {
        point = caret.Position.BufferPosition - 1;
    }
    else
    {
        wordExtent = default(TextExtent);
        return false;
    }

    ITextStructureNavigator navigator =
    m_factory.NavigatorService.GetTextStructureNavigator(m_textBuffer);

    wordExtent = navigator.GetExtentOfWord(point);
    return true;
}
```

5. Implemente el método [HasSuggestedActionsAsync](#). El editor llama a este método para averiguar si se debe mostrar la bombilla. Esta llamada se realiza a menudo, por ejemplo, cada vez que el cursor se mueve de una línea a otra, o cuando el ratón se desliza sobre un error ondulando. Es asíncrono para permitir que otras operaciones de interfaz de usuario continúen mientras este método funciona. En la mayoría de los casos, este método necesita realizar algún análisis y análisis de la línea actual, por lo que el procesamiento puede tardar algún tiempo.

En esta implementación, obtiene [TextExtent](#) de forma asíncrona y determina si la extensión es significativa, como en, si tiene algún texto que no sea espacios en blanco.

```
public Task<bool> HasSuggestedActionsAsync(ISuggestedActionCategorySet requestedActionCategories,
    SnapshotSpan range, CancellationToken cancellationToken)
{
    return Task.Factory.StartNew(() =>
    {
        TextExtent extent;
        if (TryGetWordUnderCaret(out extent))
        {
            // don't display the action if the extent has whitespace
            return extent.IsSignificant;
        }
        return false;
    });
}
```

6. Implemente [GetSuggestedActions](#) el método, que [SuggestedActionSet](#) devuelve una matriz [ISuggestedAction](#) de objetos que contienen los diferentes objetos. Este método se llama cuando se expande la bombilla.

WARNING

Debe asegurarse de que `HasSuggestedActionsAsync()` las `GetSuggestedActions()` implementaciones de y son coherentes; es decir, `HasSuggestedActionsAsync()` `true` si `GetSuggestedActions()` devuelve, entonces debe tener algunas acciones para mostrar. En muchos `HasSuggestedActionsAsync()` casos, se `GetSuggestedActions()` llama justo antes, pero esto no siempre es el caso. Por ejemplo, si el usuario invoca las acciones de bombilla `GetSuggestedActions()` presionando (CTRL+ .) sólo se llama.

```

public IEnumerable<SuggestedActionSet> GetSuggestedActions(ISuggestedActionCategorySet
requestedActionCategories, SnapshotSpan range, CancellationToken cancellationToken)
{
    TextExtent extent;
    if (TryGetWordUnderCaret(out extent) && extent.IsSignificant)
    {
        ITrackingSpan trackingSpan = range.Snapshot.CreateTrackingSpan(extent.Span,
SpanTrackingMode.EdgeInclusive);
        var upperAction = new UpperCaseSuggestedAction(trackingSpan);
        var lowerAction = new LowerCaseSuggestedAction(trackingSpan);
        return new SuggestedActionSet[] { new SuggestedActionSet(new ISuggestedAction[] { upperAction,
lowerAction }) };
    }
    return Enumerable.Empty<SuggestedActionSet>();
}

```

7. Defina `SuggestedActionsChanged` un evento.

```

public event EventHandler<EventArgs> SuggestedActionsChanged;

```

8. Para completar la implementación, `Dispose()` agregue `TryGetTelemetryId()` implementaciones para los métodos y. No desea realizar la telemetría, por `false` lo que `Empty` solo tiene que devolver y establecer el GUID en .

```

public void Dispose()
{
}

public bool TryGetTelemetryId(out Guid telemetryId)
{
    // This is a sample provider and doesn't participate in LightBulb telemetry
    telemetryId = Guid.Empty;
    return false;
}

```

Implementar acciones de bombilla

1. En el proyecto, agregue una referencia a *Microsoft.VisualStudio.Imaging.Interop.14.0.DesignTime.dll* establezca **Copy Local** en `False`.
2. Cree dos clases, la primera llamada `UpperCaseSuggestedAction` y la segunda llamada `LowerCaseSuggestedAction`. Ambas clases implementan `ISuggestedAction`.

```

internal class UpperCaseSuggestedAction : ISuggestedAction
internal class LowerCaseSuggestedAction : ISuggestedAction

```

Ambas clases son iguales, salvo que una llama a `ToUpper` y la otra llama a `ToLower`. Los siguientes pasos abarcan solo la clase de acción de mayúsculas, pero debe implementar ambas clases. Siga los pasos para implementar la acción de mayúsculas como un modelo para implementar la acción de minúsculas.

3. Agregue las siguientes directivas using para estas clases:

```
using Microsoft.VisualStudio.Imaging.Interop;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Media;
```

4. Declare un conjunto de campos privados.

```
private ITrackingSpan m_span;
private string m_upper;
private string m_display;
private ITextSnapshot m_snapshot;
```

5. Agregue un constructor que establezca los campos.

```
public UpperCaseSuggestedAction(ITrackingSpan span)
{
    m_span = span;
    m_snapshot = span.TextBuffer.CurrentSnapshot;
    m_upper = span.GetText(m_snapshot).ToUpper();
    m_display = string.Format("Convert '{0}' to upper case", span.GetText(m_snapshot));
}
```

6. Implemente [GetPreviewAsync](#) el método para que muestre la vista previa de la acción.

```
public Task<object> GetPreviewAsync(CancellationTokens cancellationTokens)
{
    var textBlock = new TextBlock();
    textBlock.Padding = new Thickness(5);
    textBlock.Inlines.Add(new Run() { Text = m_upper });
    return Task.FromResult<object>(textBlock);
}
```

7. Implemente [GetActionSetsAsync](#) el método para que [SuggestedActionSet](#) devuelva una enumeración vacía.

```
public Task<IEnumerable<SuggestedActionSet>> GetActionSetsAsync(CancellationTokens cancellationTokens)
{
    return Task.FromResult<IEnumerable<SuggestedActionSet>>(null);
}
```

8. Implemente las propiedades de la siguiente manera:

```

public bool HasActionSets
{
    get { return false; }
}
public string DisplayText
{
    get { return m_display; }
}
public ImageMoniker IconMoniker
{
    get { return default(ImageMoniker); }
}
public string IconAutomationText
{
    get
    {
        return null;
    }
}
public string InputGestureText
{
    get
    {
        return null;
    }
}
public bool HasPreview
{
    get { return true; }
}
}

```

9. Implemente el método `Invoke` reemplazando el texto en el intervalo por su equivalente en mayúsculas.

```

public void Invoke(Cancellation_token cancellation_token)
{
    m_span.TextBuffer.Replace(m_span.GetSpan(m_snapshot), m_upper);
}

```

WARNING

No se espera que el método **Invoke** de la acción de bombilla muestre la interfaz de usuario. Si la acción abre una nueva interfaz de usuario (por ejemplo, una vista previa o un cuadro de diálogo de selección), no muestre la interfaz de usuario directamente desde el método **Invoke**, sino que programe la visualización de la interfaz de usuario después de volver desde **Invoke**.

10. Para completar la implementación, agregue los `Dispose()` métodos y `TryGetTelemetryId()`

```

public void Dispose()
{
}

public bool TryGetTelemetryId(out Guid telemetryId)
{
    // This is a sample action and doesn't participate in LightBulb telemetry
    telemetryId = Guid.Empty;
    return false;
}

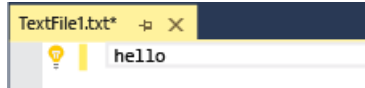
```

11. No se olvide de hacer `LowerCaseSuggestedAction` lo mismo para cambiar el texto de visualización `ToUpper` `ToLower` a "Convertir ' a minúsculas" y la llamada a .

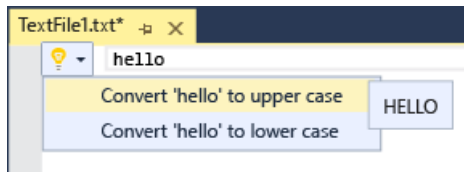
Compile y pruebe el código

Para probar este código, compile la solución LightBulbTest y ejecútela en la instancia experimental.

1. Compile la solución.
2. Al ejecutar este proyecto en el depurador, se inicia una segunda instancia de Visual Studio.
3. Cree un archivo de texto y escriba algún texto. Debería ver una bombilla a la izquierda del texto.



4. Apunte a la bombilla. Deberías ver una flecha hacia abajo.
5. Al hacer clic en la bombilla, deben mostrarse dos acciones sugeridas, junto con la vista previa de la acción seleccionada.



6. Si hace clic en la primera acción, todo el texto de la palabra actual debe convertirse a mayúsculas. Si hace clic en la segunda acción, todo el texto debe convertirse a minúsculas.

Tutorial: Utilice un comando shell con una extensión de editor

06/04/2020 • 21 minutes to read • [Edit Online](#)

Desde un VSPackage, puede agregar características como comandos de menú al editor. En este tutorial se muestra cómo agregar un adorno a una vista de texto en el editor invocando un comando de menú.

En este tutorial se muestra el uso de un VSPackage junto con un componente de Managed Extensibility Framework (MEF). Debe usar un VSPackage para registrar el comando de menú con el shell de Visual Studio. Además, puede utilizar el comando para acceder a la parte del componente MEF.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear una extensión con un comando de menú

Crear un VSPackage que coloca un comando de menú denominado **Agregar adornación** en el **herramientas** menú.

1. Cree un proyecto vSIX `MenuCommandTest` de C- con el nombre y agregue un nombre de plantilla de elemento de comando personalizado **AddAdornment**. Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. Se abre una solución denominada MenuCommandTest. El MenuCommandTestPackage archivo tiene el código que crea el comando de menú y lo coloca en el **herramientas** menú. En este punto, el comando solo hace que aparezca un cuadro de mensaje. Los pasos posteriores mostrarán cómo cambiar esto para mostrar el adorno del comentario.
3. Abra el archivo `source.extension.vsixmanifest` en el Editor de manifiestos de VSIX. La `Assets` pestaña debe tener una fila para un Microsoft.VisualStudio.VsPackage denominado MenuCommandTest.
4. Guarde y cierre el archivo `source.extension.vsixmanifest`.

Agregue una extensión MEF a la extensión de comando

1. En el **Explorador** de soluciones, haga clic con el botón secundario en el nodo de solución, haga clic en **Agregar**, a continuación, haga clic en **Nuevo proyecto**. En el cuadro de diálogo **Agregar nuevo proyecto**, **VSIX Project** haga clic en **Extensibilidad** en Visual C#. Asigne el nombre al proyecto `CommentAdornmentTest`.
2. Dado que este proyecto interactuará con el ensamblado VSPackage con nombre seguro, debe firmar el ensamblado. Puede reutilizar el archivo de clave ya creado para el ensamblado VSPackage.
 - a. Abra las propiedades del proyecto y seleccione la pestaña **Firma**.
 - b. Seleccione **Firmar el ensamblado**.
 - c. En **Elegir un archivo** de clave de nombre seguro, seleccione el archivo `Key.snk` que se generó para el ensamblado MenuCommandTest.

Consulte la extensión MEF en el proyecto VSPackage

Dado que va a agregar un componente MEF al VSPackage, debe especificar ambos tipos de activos en el manifiesto.

NOTE

Para obtener más información acerca de MEF, vea [Managed Extensibility Framework \(MEF\)](#).

Para hacer referencia al componente MEF en el proyecto VSPackage

1. En el proyecto MenuCommandTest, abra el archivo *source.extension.vsixmanifest* en el Editor de manifiestos VSIX.
2. En la pestaña **Activos**, haga clic en **Nuevo**.
3. En la lista **Tipo**, elija **Microsoft.VisualStudio.MefComponent**.
4. En la lista **Origen**, elija Un proyecto en la solución actual.
5. En la lista **Proyecto**, elija **CommentAdornmentTest**.
6. Guarde y cierre el archivo *source.extension.vsixmanifest*.
7. Asegúrese de que el MenuCommandTest proyecto tiene una referencia a la CommentAdornmentTest proyecto.
8. En el proyecto CommentAdornmentTest, establezca el proyecto para generar un ensamblado. En el **Explorador de soluciones**, seleccione el proyecto y busque en la ventana **Propiedades** de la propiedad **Copiar salida de compilación en OutputDirectory** y establézcalo en **true**.

Definir un adorno de comentario

El propio adorno de [ITrackingSpan](#) comentario consta de un que realiza un seguimiento del texto seleccionado y algunas cadenas que representan al autor y la descripción del texto.

Para definir un adorno de comentario

1. En el proyecto CommentAdornmentTest, agregue un `CommentAdornment` nuevo archivo de clase y asímóquele.
2. Agregue las siguientes referencias:
 - a. Microsoft.VisualStudio.CoreUtility
 - b. Microsoft.VisualStudio.Text.Data
 - c. Microsoft.VisualStudio.Text.Logic
 - d. Microsoft.VisualStudio.Text.UI
 - e. Microsoft.VisualStudio.Text.UI.Wpf
 - f. System.ComponentModel.Composition
 - g. PresentationCore
 - h. PresentationFramework
 - i. WindowsBase
3. Agregue la `using` siguiente directiva.


```
using Microsoft.VisualStudio.Text;
```

4. El archivo debe contener `CommentAdornment` una clase denominada .

```
internal class CommentAdornment
```

5. Agregue tres campos `CommentAdornment` a `ITrackingSpan` la clase para el , el autor y la descripción.

```
public readonly ITrackingSpan Span;  
public readonly string Author;  
public readonly string Text;
```

6. Agregue un constructor que inicialice los campos.

```
public CommentAdornment(SnapshotSpan span, string author, string text)  
{  
    this.Span = span.Snapshot.CreateTrackingSpan(span, SpanTrackingMode.EdgeExclusive);  
    this.Author = author;  
    this.Text = text;  
}
```

Cree un elemento visual para el adorno

Defina un elemento visual para el adorno. Para este tutorial, defina un control que herede de `Canvas` la clase Windows Presentation Foundation (WPF).

1. Cree una clase en el proyecto `CommentAdornmentTest` y `CommentBlock` asío.
2. Agregue las `using` siguientes directivas.

```
using Microsoft.VisualStudio.Text;  
using System;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Media;  
using System.Windows.Shapes;  
using System.ComponentModel.Composition;  
using Microsoft.VisualStudio.Text.Editor;  
using Microsoft.VisualStudio.Utilities;
```

3. Haga `CommentBlock` que la `Canvas` clase herede de .

```
internal class CommentBlock : Canvas  
{ }
```

4. Agregue algunos campos privados para definir los aspectos visuales del adorno.

```
private Geometry textGeometry;  
private Grid commentGrid;  
private static Brush brush;  
private static Pen solidPen;  
private static Pen dashPen;
```

5. Agregue un constructor que defina el adorno de comentarioy agregue el texto relevante.

```

public CommentBlock(double textRightEdge, double viewRightEdge,
    Geometry newTextGeometry, string author, string body)
{
    if (brush == null)
    {
        brush = new SolidColorBrush(Color.FromArgb(0x20, 0x00, 0xff, 0x00));
        brush.Freeze();
        Brush penBrush = new SolidColorBrush(Colors.Green);
        penBrush.Freeze();
        solidPen = new Pen(penBrush, 0.5);
        solidPen.Freeze();
        dashPen = new Pen(penBrush, 0.5);
        dashPen.DashStyle = DashStyles.Dash;
        dashPen.Freeze();
    }

    this.textGeometry = newTextGeometry;

    TextBlock tb1 = new TextBlock();
    tb1.Text = author;
    TextBlock tb2 = new TextBlock();
    tb2.Text = body;

    const int MarginWidth = 8;
    this.commentGrid = new Grid();
    this.commentGrid.RowDefinitions.Add(new RowDefinition());
    this.commentGrid.RowDefinitions.Add(new RowDefinition());
    ColumnDefinition cEdge = new ColumnDefinition();
    cEdge.Width = new GridLength(MarginWidth);
    ColumnDefinition cEdge2 = new ColumnDefinition();
    cEdge2.Width = new GridLength(MarginWidth);
    this.commentGrid.ColumnDefinitions.Add(cEdge);
    this.commentGrid.ColumnDefinitions.Add(new ColumnDefinition());
    this.commentGrid.ColumnDefinitions.Add(cEdge2);

    System.Windows.Shapes.Rectangle rect = new System.Windows.Shapes.Rectangle();
    rect.RadiusX = 6;
    rect.RadiusY = 3;
    rect.Fill = brush;
    rect.Stroke = Brushes.Green;

    Size inf = new Size(double.PositiveInfinity, double.PositiveInfinity);
    tb1.Measure(inf);
    tb2.Measure(inf);
    double middleWidth = Math.Max(tb1.DesiredSize.Width, tb2.DesiredSize.Width);
    this.commentGrid.Width = middleWidth + 2 * MarginWidth;

    Grid.SetColumn(rect, 0);
    Grid.SetRow(rect, 0);
    Grid.SetRowSpan(rect, 2);
    Grid.SetColumnSpan(rect, 3);
    Grid.SetRow(tb1, 0);
    Grid.SetColumn(tb1, 1);
    Grid.SetRow(tb2, 1);
    Grid.SetColumn(tb2, 1);
    this.commentGrid.Children.Add(rect);
    this.commentGrid.Children.Add(tb1);
    this.commentGrid.Children.Add(tb2);

    Canvas.SetLeft(this.commentGrid, Math.Max(viewRightEdge - this.commentGrid.Width - 20.0,
textRightEdge + 20.0));
    Canvas.SetTop(this.commentGrid, textGeometry.GetRenderBounds(solidPen).Top);

    this.Children.Add(this.commentGrid);
}

```

6. Implemente también [OnRender](#) un controlador de eventos que dibuje el adorno.

```
protected override void OnRender(DrawingContext dc)
{
    base.OnRender(dc);
    if (this.textGeometry != null)
    {
        dc.DrawGeometry(brush, solidPen, this.textGeometry);
        Rect textBounds = this.textGeometry.GetRenderBounds(solidPen);
        Point p1 = new Point(textBounds.Right, textBounds.Bottom);
        Point p2 = new Point(Math.Max(Canvas.GetLeft(this.commentGrid) - 20.0, p1.X), p1.Y);
        Point p3 = new Point(Math.Max(Canvas.GetLeft(this.commentGrid), p1.X),
            (Canvas.GetTop(this.commentGrid) + p1.Y) * 0.5);
        dc.DrawLine(dashPen, p1, p2);
        dc.DrawLine(dashPen, p2, p3);
    }
}
```

Agregar un [IWpfTextViewCreationListener](#)

Es [IWpfTextViewCreationListener](#) una parte del componente MEF que puede utilizar para escuchar eventos de creación.

1. Agregue un archivo de clase al proyecto `Connector` `CommentAdornmentTest` y asímócle el nombre .
2. Agregue las `using` siguientes directivas.

```
using System.ComponentModel.Composition;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Utilities;
```

3. Declare una clase [IWpfTextViewCreationListener](#) que implemente y [ContentTypeAttribute](#) expórtela [TextViewRoleAttribute](#) con `Document` un de "texto" y un archivo de archivo . El atributo de tipo de contenido especifica el tipo de contenido al que se aplica el componente. El tipo de texto es el tipo base para todos los tipos de archivo no binarios. Por lo tanto, casi todas las vistas de texto que se crean serán de este tipo. El atributo de rol de vista de texto especifica el tipo de vista de texto a la que se aplica el componente. Los roles de vista de texto de documento generalmente muestran texto que se compone de líneas y se almacena en un archivo.

```
<Export(GetType(IWpfTextViewCreationListener)), ContentType("text"),
TextViewRole(PredefinedTextViewRoles.Document)>
Public NotInheritable Class Connector
    Implements IWpfTextViewCreationListener
```

```
[Export(typeof(IWpfTextViewCreationListener))]
[ContentType("text")]
[TextViewRole(PredefinedTextViewRoles.Document)]
public sealed class Connector : IWpfTextViewCreationListener
```

4. Implemente [TextViewCreated](#) el método para que `Create()` llame `CommentAdornmentManager` al evento estático del archivo .

```
public void TextViewCreated(IWpfTextView textView)
{
    CommentAdornmentManager.Create(textView);
}
```

5. Agregue un método que pueda utilizar para ejecutar el comando.

```
static public void Execute(IWpfTextViewHost host)
{
    IWpfTextView view = host.TextView;
    //Add a comment on the selected text.
    if (!view.Selection.IsEmpty)
    {
        //Get the provider for the comment adornments in the property bag of the view.
        CommentAdornmentProvider provider = view.Properties.GetProperty<CommentAdornmentProvider>
        (typeof(CommentAdornmentProvider));

        //Add some arbitrary author and comment text.
        string author = System.Security.Principal.WindowsIdentity.GetCurrent().Name;
        string comment = "Four score...";

        //Add the comment adornment using the provider.
        provider.Add(view.Selection.SelectedSpans[0], author, comment);
    }
}
```

Definir una capa de adorno

Para agregar un nuevo adorno, debe definir una capa de adorno.

Para definir una capa de adorno

1. En `Connector` la clase, declare un `AdornmentLayerDefinition` campo público de `NameAttribute` tipo y expórtelo con `OrderAttribute` un que especifique un nombre único para la capa de adorno y un que defina la relación de orden Z de esta capa de adorno con las otras capas de vista de texto (texto, intercalador y selección).

```
[Export(typeof(AdornmentLayerDefinition))]
[Name("CommentAdornmentLayer")]
[Order(After = PredefinedAdornmentLayers.Selection, Before = PredefinedAdornmentLayers.Text)]
public AdornmentLayerDefinition commentLayerDefinition;
```

Proporcionar adornos de comentarios

Al definir un adorno, también implemente un proveedor de adornos de comentarios y un administrador de adornos de comentarios. El proveedor de adornos de comentarios mantiene una `Changed` lista de adornos de comentarios, escucha eventos en el búfer de texto subyacente y elimina los adornos de comentarios cuando se elimina el texto subyacente.

1. Agregue un nuevo archivo de clase al proyecto `CommentAdornmentProvider` `CommentAdornmentTest` y asímócle el nombre .
2. Agregue las `using` siguientes directivas.

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
```

3. Agregue una clase denominada `CommentAdornmentProvider` .

```
internal class CommentAdornmentProvider
{
}
```

4. Agregue campos privados para el búfer de texto y la lista de adornos de comentarios relacionados con el búfer.

```
private ITextBuffer buffer;
private IList<CommentAdornment> comments = new List<CommentAdornment>();
```

5. Agregue un `CommentAdornmentProvider` constructor para . Este constructor debe tener acceso privado porque `Create()` el método crea una instancia del proveedor. El constructor agrega `OnBufferChanged` el controlador `Changed` de eventos al evento.

```
private CommentAdornmentProvider(ITextBuffer buffer)
{
    this.buffer = buffer;
    //listen to the Changed event so we can react to deletions.
    this.buffer.Changed += OnBufferChanged;
}
```

6. Agregue el método `Create()` .

```
public static CommentAdornmentProvider Create(IWpfTextView view)
{
    return view.Properties.GetOrCreateSingletonProperty<CommentAdornmentProvider>(delegate { return new
    CommentAdornmentProvider(view.TextBuffer); });
}
```

7. Agregue el método `Detach()` .

```
public void Detach()
{
    if (this.buffer != null)
    {
        //remove the Changed listener
        this.buffer.Changed -= OnBufferChanged;
        this.buffer = null;
    }
}
```

8. Agregue `OnBufferChanged` el controlador de eventos.

```

private void OnBufferChanged(object sender, TextContentChangedEventArgs e)
{
    //Make a list of all comments that have a span of at least one character after applying the change.
    There is no need to raise a changed event for the deleted adornments. The adornments are deleted only
    if a text change would cause the view to reformat the line and discard the adornments.
    IList<CommentAdornment> keptComments = new List<CommentAdornment>(this.comments.Count);

    foreach (CommentAdornment comment in this.comments)
    {
        Span span = comment.Span.GetSpan(e.After);
        //if a comment does not span at least one character, its text was deleted.
        if (span.Length != 0)
        {
            keptComments.Add(comment);
        }
    }

    this.comments = keptComments;
}

```

```

Private Sub OnBufferChanged(ByVal sender As Object, ByVal e As TextContentChangedEventArgs)
    'Make a list of all comments that have a span of at least one character after applying the change.
    There is no need to raise a changed event for the deleted adornments. The adornments are deleted only
    if a text change would cause the view to reformat the line and discard the adornments.
    Dim keptComments As IList(Of CommentAdornment) = New List(Of CommentAdornment)(Me.comments.Count)

    For Each comment As CommentAdornment In Me.comments
        Dim span As Span = comment.Span.GetSpan(e.After)
        'if a comment does not span at least one character, its text was deleted.
        If span.Length <> 0 Then
            keptComments.Add(comment)
        End If
    Next comment

    Me.comments = keptComments
End Sub

```

9. Agregue una declaración para un `CommentsChanged` evento.

```

public event EventHandler<CommentsChangedEventArgs> CommentsChanged;

```

10. Cree `Add()` un método para agregar el adorno.

```

public void Add(SnapshotSpan span, string author, string text)
{
    if (span.Length == 0)
        throw new ArgumentOutOfRangeException("span");
    if (author == null)
        throw new ArgumentNullException("author");
    if (text == null)
        throw new ArgumentNullException("text");

    //Create a comment adornment given the span, author and text.
    CommentAdornment comment = new CommentAdornment(span, author, text);

    //Add it to the list of comments.
    this.comments.Add(comment);

    //Raise the changed event.
    EventHandler<CommentsChangedEventArgs> commentsChanged = this.CommentsChanged;
    if (commentsChanged != null)
        commentsChanged(this, new CommentsChangedEventArgs(comment, null));
}

```

11. Agregue `RemoveComments()` un método.

```

public void RemoveComments(SnapshotSpan span)
{
    EventHandler<CommentsChangedEventArgs> commentsChanged = this.CommentsChanged;

    //Get a list of all the comments that are being kept
    IList<CommentAdornment> keptComments = new List<CommentAdornment>(this.comments.Count);

    foreach (CommentAdornment comment in this.comments)
    {
        //find out if the given span overlaps with the comment text span. If two spans are adjacent,
        they do not overlap. To consider adjacent spans, use IntersectsWith.
        if (comment.Span.GetSpan(span.Snapshot).OverlapsWith(span))
        {
            //Raise the change event to delete this comment.
            if (commentsChanged != null)
                commentsChanged(this, new CommentsChangedEventArgs(null, comment));
        }
        else
            keptComments.Add(comment);
    }

    this.comments = keptComments;
}

```

12. Agregue `GetComments()` un método que devuelva todos los comentarios de un intervalo de instantáneas determinado.

```

public Collection<CommentAdornment> GetComments(SnapshotSpan span)
{
    IList<CommentAdornment> overlappingComments = new List<CommentAdornment>();
    foreach (CommentAdornment comment in this.comments)
    {
        if (comment.Span.GetSpan(span.Snapshot).OverlapsWith(span))
            overlappingComments.Add(comment);
    }

    return new Collection<CommentAdornment>(overlappingComments);
}

```

13. Agregue una `CommentsChangedEventArgs` clase denominada , como se indica a continuación.

```
internal class CommentsChangedEventArgs : EventArgs
{
    public readonly CommentAdornment CommentAdded;

    public readonly CommentAdornment CommentRemoved;

    public CommentsChangedEventArgs(CommentAdornment added, CommentAdornment removed)
    {
        this.CommentAdded = added;
        this.CommentRemoved = removed;
    }
}
```

Gestionar adornos de comentarios

El administrador de adornos de comentarios crea el adorno y lo agrega a la capa de adorno. Escucha los [LayoutChanged](#) eventos [Closed](#) y para que pueda mover o eliminar el adorno. También escucha el `CommentsChanged` evento que desencadena el proveedor de adornos de comentarios cuando se agregan o quitan comentarios.

1. Agregue un archivo de clase al proyecto `CommentAdornmentManager` `CommentAdornmentTest` y asímócle el nombre .
2. Agregue las `using` siguientes directivas.

```
using System;
using System.Collections.Generic;
using System.Windows.Media;
using Microsoft.VisualStudio.Text;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Formatting;
```

3. Agregue una clase denominada `CommentAdornmentManager` .

```
internal class CommentAdornmentManager
{
}
```

4. Agregue algunos campos privados.

```
private readonly IWpfTextView view;
private readonly IAdornmentLayer layer;
private readonly CommentAdornmentProvider provider;
```

5. Agregue un constructor que suscribe [LayoutChanged](#) [Closed](#) el administrador a `CommentsChanged` los eventos y, así como al evento. El constructor es privado porque el método `Create()` estático crea una instancia del administrador.


```
private CommentAdornmentManager(IWpfTextView view)
{
    this.view = view;
    this.view.LayoutChanged += OnLayoutChanged;
    this.view.Closed += OnClosed;

    this.layer = view.GetAdornmentLayer("CommentAdornmentLayer");

    this.provider = CommentAdornmentProvider.Create(view);
    this.provider.CommentsChanged += OnCommentsChanged;
}
```

6. Agregue `Create()` el método que obtiene un proveedor o crea uno si es necesario.

```
public static CommentAdornmentManager Create(IWpfTextView view)
{
    return view.Properties.GetOrCreateSingletonProperty<CommentAdornmentManager>(delegate { return new
    CommentAdornmentManager(view); });
}
```

7. Agregue `CommentsChanged` el controlador.

```
private void OnCommentsChanged(object sender, CommentsChangedEventArgs e)
{
    //Remove the comment (when the adornment was added, the comment adornment was used as the tag).
    if (e.CommentRemoved != null)
        this.layer.RemoveAdornmentsByTag(e.CommentRemoved);

    //Draw the newly added comment (this will appear immediately: the view does not need to do a
    layout).
    if (e.CommentAdded != null)
        this.DrawComment(e.CommentAdded);
}
```

8. Agregue `Closed` el controlador.

```
private void OnClosed(object sender, EventArgs e)
{
    this.provider.Detach();
    this.view.LayoutChanged -= OnLayoutChanged;
    this.view.Closed -= OnClosed;
}
```

9. Agregue `LayoutChanged` el controlador.

```

private void OnLayoutChanged(object sender, TextViewLayoutChangedEventArgs e)
{
    //Get all of the comments that intersect any of the new or reformatted lines of text.
    List<CommentAdornment> newComments = new List<CommentAdornment>();

    //The event args contain a list of modified lines and a NormalizedSpanCollection of the spans of
    the modified lines.
    //Use the latter to find the comments that intersect the new or reformatted lines of text.
    foreach (Span span in e.NewOrReformattedSpans)
    {
        newComments.AddRange(this.provider.GetComments(new SnapshotSpan(this.view.TextSnapshot,
span)));
    }

    //It is possible to get duplicates in this list if a comment spanned 3 lines, and the first and
    last lines were modified but the middle line was not.
    //Sort the list and skip duplicates.
    newComments.Sort(delegate(CommentAdornment a, CommentAdornment b) { return
a.GetHashCode().CompareTo(b.GetHashCode()); });

    CommentAdornment lastComment = null;
    foreach (CommentAdornment comment in newComments)
    {
        if (comment != lastComment)
        {
            lastComment = comment;
            this.DrawComment(comment);
        }
    }
}

```

10. Agregue el método privado que dibuja el comentario.

```

private void DrawComment(CommentAdornment comment)
{
    SnapshotSpan span = comment.Span.GetSpan(this.view.TextSnapshot);
    Geometry g = this.view.TextViewLines.GetMarkerGeometry(span);

    if (g != null)
    {
        //Find the rightmost coordinate of all the lines that intersect the adornment.
        double maxRight = 0.0;
        foreach (ITextViewLine line in this.view.TextViewLines.GetTextViewLinesIntersectingSpan(span))
            maxRight = Math.Max(maxRight, line.Right);

        //Create the visualization.
        CommentBlock block = new CommentBlock(maxRight, this.view.ViewportRight, g, comment.Author,
comment.Text);

        //Add it to the layer.
        this.layer.AddAdornment(span, comment, block);
    }
}

```

```

Private Sub DrawComment(ByVal comment As CommentAdornment)
    Dim span As SnapshotSpan = comment.Span.GetSpan(Me.view.TextSnapshot)
    Dim g As Geometry = Me.view.TextViewLines.GetMarkerGeometry(span)

    If g IsNot Nothing Then
        'Find the rightmost coordinate of all the lines that intersect the adornment.
        Dim maxRight As Double = 0.0
        For Each line As ITextViewLine In Me.view.TextViewLines.GetTextViewLinesIntersectingSpan(span)
            maxRight = Math.Max(maxRight, line.Right)
        Next line

        'Create the visualization.
        Dim block As New CommentBlock(maxRight, Me.view.ViewportRight, g, comment.Author, comment.Text)

        'Add it to the layer.
        Me.layer.AddAdornment(span, comment, block)
    End If
End Sub

```

Utilice el comando de menú para agregar el adorno de comentario

Puede usar el comando de menú para crear `MenuItemCallback` un adorno de comentario mediante la implementación del método de la `VSPackage`.

1. Agregue las siguientes referencias al proyecto `MenuCommandTest`:

- `Microsoft.VisualStudio.TextManager.Interop`
- `Microsoft.VisualStudio.Editor`
- `Microsoft.VisualStudio.Text.UI.Wpf`

2. Abra el archivo `AddAdornment.cs` `using` y agregue las siguientes directivas.

```

using Microsoft.VisualStudio.TextManager.Interop;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Editor;
using CommentAdornmentTest;

```

3. Elimine `Execute()` el método y agregue el siguiente controlador de comandos.

```

private async void AddAdornmentHandler(object sender, EventArgs e)
{
}

```

4. Agregue código para obtener la vista activa. Debe obtener `SVsTextManager` el shell de Visual Studio `IVsTextView` para obtener el archivo .

```

private async void AddAdornmentHandler(object sender, EventArgs e)
{
    IVsTextManager txtMgr = (IVsTextManager) await
        ServiceProvider.GetServiceAsync(typeof(SVsTextManager));
    IVsTextView vTextView = null;
    int mustHaveFocus = 1;
    txtMgr.GetActiveView(mustHaveFocus, null, out vTextView);
}

```

5. Si esta vista de texto es una instancia de una `IVsUserData` vista de `IWpfTextViewHost` texto del

[IWpfTextView](#) editor, puede convertirla en la interfaz y, a continuación, obtener el archivo . Utilice [IWpfTextViewHost](#) el para `Connector.Execute()` llamar al método, que obtiene el proveedor de adorno de comentarioy agrega el adorno. El controlador de comandos ahora debería parecerse a este código:

```
private async void AddAdornmentHandler(object sender, EventArgs e)
{
    IVsTextManager txtMgr = (IVsTextManager) await
ServiceProvider.GetServiceAsync(typeof(SVsTextManager));
    IVsTextView vTextView = null;
    int mustHaveFocus = 1;
    txtMgr.GetActiveView(mustHaveFocus, null, out vTextView);
    IVsUserData userData = vTextView as IVsUserData;
    if (userData == null)
    {
        Console.WriteLine("No text view is currently open");
        return;
    }
    IWpfTextViewHost viewHost;
    object holder;
    Guid guidViewHost = DefGuidList.guidIWpfTextViewHost;
    userData.GetData(ref guidViewHost, out holder);
    viewHost = (IWpfTextViewHost)holder;
    Connector.Execute(viewHost);
}
```

6. Establecer el AddAdornmentHandler método como el controlador para el AddAdornment comando en el AddAdornment constructor.

```
private AddAdornment(AsyncPackage package, OleMenuCommandService commandService)
{
    this.package = package ?? throw new ArgumentNullException(nameof(package));
    commandService = commandService ?? throw new ArgumentNullException(nameof(commandService));

    var menuCommandID = new CommandID(CommandSet, CommandId);
    var menuItem = new MenuCommand(this.AddAdornmentHandler, menuCommandID);
    commandService.AddCommand(menuItem);
}
```

Compile y pruebe el código

1. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.
2. Crear un archivo de texto Escriba algo de texto y, a continuación, selecciónelo.
3. En el menú **Herramientas** , haga clic en **Invocar agregar adorno**. Un globo debe mostrarse en el lado derecho de la ventana de texto y debe contener texto similar al texto siguiente.

YourUserName

Ochenta...

Vea también

- [Tutorial: Vincule un tipo de contenido a una extensión de nombre de archivo](#)

Tutorial: Use una tecla de método abreviado con una extensión de editor

06/04/2020 • 10 minutes to read • [Edit Online](#)

Puede responder a las teclas de método abreviado en la extensión del editor. En el siguiente tutorial se muestra cómo agregar un adorno de vista a una vista de texto mediante una tecla de método abreviado. Este tutorial se basa en la plantilla del editor de adornos de la ventana gráfica y le permite agregar el adorno mediante el carácter +.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto de managed Extensibility Framework (MEF)

1. Cree un proyecto de VSIX de C. (En el cuadro de diálogo **Nuevo proyecto** , seleccione **Visual C/ Extensibilidad**, a continuación, **Proyecto VSIX**.) Asigne un `KeyBindingTest` nombre a la solución .
2. Agregue una plantilla de elemento Adornment de `KeyBindingTest` texto del editor al proyecto y así saber. Para obtener más información, vea [Crear una extensión con una plantilla](#) de elemento de editor .
3. Agregue las siguientes referencias y `false` establezca **CopyLocal** en:

Microsoft.VisualStudio.Editor

Microsoft.VisualStudio.OLE.Interop

Microsoft.VisualStudio.Shell.14.0

Microsoft.VisualStudio.TextManager.Interop

En el `keyBindingTest` archivo de clase, cambie el nombre de clase a `PurpleCornerRadius`. Utilice la bombilla que aparece en el margen izquierdo para realizar los demás cambios apropiados. Dentro del constructor, cambie el nombre de la capa de adorno de **KeyBindingTest** a **PurpleCornerRadius**:

```
this.layer = view.GetAdornmentLayer("PurpleCornerRadius");
```

En el archivo de clase `KeyBindingTestTextViewCreationListener.cs`, cambie el nombre de `AdornmentLayer` de **KeyBindingTest** a **PurpleCornerRadius**:

```
[Export(typeof(AdornmentLayerDefinition))]  
[Name("PurpleCornerRadius")]  
[Order(After = PredefinedAdornmentLayers.Selection, Before = PredefinedAdornmentLayers.Text)]  
public AdornmentLayerDefinition editorAdornmentLayer;
```

Comando Handle TYPECHAR

Antes de Visual Studio 2017 versión 15.6, la única [IOleCommandTarget](#) manera de controlar comandos en una

extensión de editor era implementar un filtro de comandos basado. Visual Studio 2017 versión 15.6 introdujo un enfoque simplificado moderno basado en controladores de comandos del editor. Las dos secciones siguientes muestran cómo controlar un comando mediante el enfoque heredado y moderno.

Definir el filtro de comandos (antes de Visual Studio 2017 versión 15.6)

El filtro de comandos es una implementación de [IOleCommandTarget](#), que controla el comando creando una instancia del adorno.

1. Agregue un archivo de clase y asígnele el nombre `KeyBindingCommandFilter`.
2. Agregue las siguientes directivas using.

```
using System;
using System.Runtime.InteropServices;
using Microsoft.VisualStudio.OLE.Interop;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Text.Editor;
```

3. La clase denominada `KeyBindingCommandFilter` [IOleCommandTarget](#) debe heredar de .

```
internal class KeyBindingCommandFilter : IOleCommandTarget
```

4. Agregue campos privados para la vista de texto, el siguiente comando de la cadena de comandos y una marca para representar si el filtro de comandos ya se ha agregado.

```
private IWpfTextView m_textView;
internal IOleCommandTarget m_nextTarget;
internal bool m_added;
internal bool m_adorned;
```

5. Agregue un constructor que establezca la vista de texto.

```
public KeyBindingCommandFilter(IWpfTextView textView)
{
    m_textView = textView;
    m_adorned = false;
}
```

6. Implemente `QueryStatus()` el método de la siguiente manera.

```
int IOleCommandTarget.QueryStatus(ref Guid pguidCmdGroup, uint cCmds, OLECMD[] prgCmds, IntPtr pCmdText)
{
    return m_nextTarget.QueryStatus(ref pguidCmdGroup, cCmds, prgCmds, pCmdText);
}
```

7. Implemente `Exec()` el método para que agregue un cuadro morado `**+**` a la vista si se escribe un carácter de signo más (+).

```

int IOleCommandTarget.Exec(ref Guid pguidCmdGroup, uint nCmdID, uint nCmdexcopt, IntPtr pvaIn, IntPtr
pvaOut)
{
    if (m_adorned == false)
    {
        char typedChar = char.MinValue;

        if (pguidCmdGroup == VSConstants.VSStd2K && nCmdID == (uint)VSConstants.VSStd2KCmdID.TYPECHAR)
        {
            typedChar = (char)(ushort)Marshal.GetObjectForNativeVariant(pvaIn);
            if (typedChar.Equals('+'))
            {
                new PurpleCornerRadius(m_textView);
                m_adorned = true;
            }
        }
    }
    return m_nextTarget.Exec(ref pguidCmdGroup, nCmdID, nCmdexcopt, pvaIn, pvaOut);
}

```

Agregue el filtro de comandos (antes de Visual Studio 2017 versión 15.6)

El proveedor de adornos debe agregar un filtro de comandos a la vista de texto. En este ejemplo, el [IVsTextViewCreationListener](#) proveedor implementa para escuchar eventos de creación de vistas de texto. Este proveedor de adornos también exporta la capa de adorno, que define el orden Z del adorno.

1. En el archivo `KeyBindingTestTextViewCreationListener`, agregue las siguientes directivas using:

```

using System;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.OLE.Interop;
using Microsoft.VisualStudio.Utilities;
using Microsoft.VisualStudio.Editor;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.TextManager.Interop;

```

2. Para obtener el adaptador de vista [IVsEditorAdaptersFactoryService](#) de texto, debe importar el archivo .

```

[Import(typeof(IVsEditorAdaptersFactoryService))]
internal IVsEditorAdaptersFactoryService editorFactory = null;

```

3. Cambie `TextViewCreated` el método para que `KeyBindingCommandFilter` agregue el archivo .

```

public void TextViewCreated(IWpfTextView textView)
{
    AddCommandFilter(textView, new KeyBindingCommandFilter(textView));
}

```

4. El `AddCommandFilter` controlador obtiene el adaptador de vista de texto y agrega el filtro de comandos.

```

void AddCommandFilter(IWpfTextView textView, KeyBindingCommandFilter commandFilter)
{
    if (commandFilter.m_added == false)
    {
        //get the view adapter from the editor factory
        IOleCommandTarget next;
        IVsTextView view = editorFactory.GetViewAdapter(textView);

        int hr = view.AddCommandFilter(commandFilter, out next);

        if (hr == VSConstants.S_OK)
        {
            commandFilter.m_added = true;
            //you'll need the next target for Exec and QueryStatus
            if (next != null)
                commandFilter.m_nextTarget = next;
        }
    }
}

```

Implementar un controlador de comandos (a partir de Visual Studio 2017 versión 15.6)

En primer lugar, actualice las referencias de Nuget del proyecto para hacer referencia a la API del editor más reciente:

1. Haga clic con el botón derecho en el proyecto y seleccione **Administrar paquetes Nuget**.
2. En **Nuget Package Manager**, seleccione la pestaña **Actualizaciones**, active la casilla **Seleccionar todos los paquetes** y, a continuación, seleccione **Actualizar**.

El controlador de comandos es una implementación de [ICommandHandler<T>](#), que controla el comando creando una instancia del adorno.

1. Agregue un archivo de clase y asígnele el nombre `KeyBindingCommandHandler`.
2. Agregue las siguientes directivas using.

```

using Microsoft.VisualStudio.Commanding;
using Microsoft.VisualStudio.Text.Editor;
using Microsoft.VisualStudio.Text.Editor.Commanding.Commands;
using Microsoft.VisualStudio.Utilities;
using System.ComponentModel.Composition;

```

3. La clase denominada `KeyBindingCommandHandler` `ICommandHandler<TypeCharCommandArgs>` debe heredar [ICommandHandler](#) de , y exportarla como:

```

[Export(typeof(ICommandHandler))]
[ContentType("text")]
[Name("KeyBindingTest")]
internal class KeyBindingCommandHandler : ICommandHandler<TypeCharCommandArgs>

```

4. Agregue un nombre para mostrar del controlador de comandos:

```

public string DisplayName => "KeyBindingTest";

```

5. Implemente `GetCommandState()` el método de la siguiente manera. Dado que este controlador de comandos

controla el comando TYPECHAR del editor principal, puede delegar la habilitación del comando en el editor principal.

```
public CommandState GetCommandState(TypeCharCommandArgs args)
{
    return CommandState.Unspecified;
}
```

6. Implemente `ExecuteCommand()` el método para que agregue un cuadro morado**+** a la vista si se escribe un carácter de signo más ().

```
public bool ExecuteCommand(TypeCharCommandArgs args, CommandExecutionContext executionContext)
{
    if (args.TypedChar == '+')
    {
        bool alreadyAdorned = args.TextView.Properties.TryGetProperty(
            "KeyBindingTextAdorned", out bool adorned) && adorned;
        if (!alreadyAdorned)
        {
            new PurpleCornerRadius((IWPfTextView)args.TextView);
            args.TextView.Properties.AddProperty("KeyBindingTextAdorned", true);
        }
    }

    return false;
}
```

- g. Copie la definición de capa de adorno del archivo *de KeyBindingTestTextViewCreationListener.cs* al *KeyBindingCommandHandler.cs* y, a continuación, elimine *KeyBindingTestTextViewCreationListener.cs* archivo:

```
/// <summary>
/// Defines the adornment layer for the adornment. This layer is ordered
/// after the selection layer in the Z-order.
/// </summary>
[Export(typeof(AdornmentLayerDefinition))]
[Name("PurpleCornerRadius")]
[Order(After = PredefinedAdornmentLayers.Selection, Before = PredefinedAdornmentLayers.Text)]
private AdornmentLayerDefinition editorAdornmentLayer;
```

Haz que el adorno aparezca en cada línea

El adorno original apareció en cada carácter 'a' en un archivo de texto. Ahora que hemos cambiado el código para agregar + el adorno en respuesta al carácter, + agrega el adorno solo en la línea donde se escribe el carácter. Podemos cambiar el código de adorno para que el adorno aparezca una vez más en cada 'a'.

En el archivo *KeyBindingTest.cs*, cambie el `CreateVisuals()` método para recorrer en iteración todas las líneas de la vista para decorar el carácter 'a'.

```

private void CreateVisuals(ITextViewLine line)
{
    IWpfTextViewLineCollection textViewLines = this.view.TextViewLines;

    foreach (ITextViewLine textViewLine in textViewLines)
    {
        if (textViewLine.ToString().Contains("a"))
        {
            // Loop through each character, and place a box around any 'a'
            for (int charIndex = textViewLine.Start; charIndex < textViewLine.End; charIndex++)
            {
                if (this.view.TextSnapshot[charIndex] == 'a')
                {
                    SnapshotSpan span = new SnapshotSpan(this.view.TextSnapshot, Span.FromBounds(charIndex,
charIndex + 1));

                    Geometry geometry = textViewLines.GetMarkerGeometry(span);
                    if (geometry != null)
                    {
                        var drawing = new GeometryDrawing(this.brush, this.pen, geometry);
                        drawing.Freeze();

                        var drawingImage = new DrawingImage(drawing);
                        drawingImage.Freeze();

                        var image = new Image
                        {
                            Source = drawingImage,
                        };

                        // Align the image with the top of the bounds of the text geometry
                        Canvas.SetLeft(image, geometry.Bounds.Left);
                        Canvas.SetTop(image, geometry.Bounds.Top);

                        this.layer.AddAdornment(AdornmentPositioningBehavior.TextRelative, span, null, image,
null);
                    }
                }
            }
        }
    }
}

```

Compile y pruebe el código

1. Compile la solución KeyBindingTest y ejecútela en la instancia experimental.
2. Cree o abra un archivo de texto. Escriba algunas palabras que contengan + el carácter 'a' y, a continuación, escriba en cualquier lugar de la vista de texto.

Un cuadrado púrpura debe aparecer en cada carácter 'a' en el archivo.

Tutorial: Acceda al objeto DTE desde una extensión de editor

06/04/2020 • 2 minutes to read • [Edit Online](#)

En VSPackages, puede obtener el objeto [GetService](#) DTE llamando al método con el tipo de la DTE objeto. En extensiones de Managed Extensibility Framework [SVsServiceProvider](#) (MEF), [GetService](#) puede importar [DTEy](#), a continuación, llamar al método con un tipo de .

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Obtiene el objeto DTE.

1. Cree un proyecto vSIX de C- y asíse **DTETest**. Agregue una plantilla de elemento **Clasificador** de editor y **asísaber**, **asísaber**, **DTETest**.

Para obtener más información, consulte Crear una extensión con una plantilla de elemento de [editor](#).

2. Agregue las siguientes referencias de ensamblado al proyecto:

- Microsoft.VisualStudio.Shell.Framework
- Microsoft.VisualStudio.Shell.Immutable.10.0

3. En el archivo *DTETestProvider.cs*, agregue las siguientes `using` directivas:

```
using EnvDTE;  
using Microsoft.VisualStudio.Shell;
```

4. En `DTETestProvider` la clase, [SVsServiceProvider](#) importe un archivo .

```
[Import]  
internal SVsServiceProvider ServiceProvider = null;
```

5. En `GetClassifier()` el método, agregue el `return` código siguiente antes de la instrucción:

```
ThreadHelper.ThrowIfNotOnUIThread();  
DTE dte = (DTE)ServiceProvider.GetService(typeof(DTE));
```

2. Agregue las siguientes referencias de ensamblado al proyecto:

- EnvDTE
- Microsoft.VisualStudio.Shell.Framework

3. En el archivo *DTETestProvider.cs*, agregue las siguientes `using` directivas:

```
using EnvDTE;  
using Microsoft.VisualStudio.Shell;
```

4. En `DTETestProvider` la clase, [SVsServiceProvider](#) importe un archivo .

```
[Import]
internal SVsServiceProvider ServiceProvider = null;
```

5. En `GetClassifier()` el método, agregue el `return` código siguiente antes de la instrucción:

```
DTE dte = (DTE)ServiceProvider.GetService(typeof(DTE));
```

Vea también

- [Servicio de lenguaje y puntos de extensión del editor](#)
- [Inicio de Visual Studio con DTE](#)

Crear editores y diseñadores personalizados

06/04/2020 • 8 minutes to read • [Edit Online](#)

El entorno de desarrollo integrado (IDE) de Visual Studio puede hospedar diferentes tipos de editor:

- El editor principal de Visual Studio
- Editores personalizados
- Editores externos
- Diseñadores

La siguiente información le ayuda a elegir el tipo de editor que necesita.

Tipos de editor

Para obtener información sobre el editor principal de Visual Studio, vea [Extender el editor](#) y los servicios de lenguaje .

Editores personalizados

Un editor personalizado es aquel que está diseñado para trabajar en circunstancias especializadas. Por ejemplo, puede crear un editor cuya función sea leer y escribir datos en un repositorio específico, como un servidor de Microsoft Exchange. Elija un editor personalizado si desea un editor que solo funcione con el tipo de proyecto o si desea un editor que tenga solo unos pocos comandos específicos. Tenga en cuenta, sin embargo, que los usuarios no podrán usar un editor personalizado para editar proyectos estándar. Visual Studio

Un editor personalizado puede usar un generador de editores y agregar información sobre el editor al registro. Sin embargo, el tipo de proyecto asociado con el editor personalizado puede crear instancias del editor personalizado de otras maneras.

Un editor personalizado puede usar la activación in situ o la incrustación simplificada para implementar una vista.

Editores externos

Los editores externos son editores que no están integrados en Visual Studio, como Microsoft Word, el Bloc de notas o Microsoft FrontPage. Puede llamar a un editor de este tipo si, por ejemplo, está pasando texto a él desde el VSPackage. Los editores externos se registran a sí mismos y se pueden usar fuera de Visual Studio. Cuando se llama a un editor externo y se puede incrustar en una ventana host, aparece en una ventana del IDE. Si no es así, el IDE crea una ventana independiente para él.

El `IsDocumentInProject` método establece la prioridad `VSDOCUMENTPRIORITY` del documento mediante la enumeración. Si `DP_External` se especifica el valor, un editor externo puede abrir el archivo.

Decisiones de diseño del editor

Las siguientes preguntas de diseño le ayudarán a elegir el tipo de editor que mejor se adapte a su aplicación:

- ¿Guardará su aplicación sus datos en archivos o no? Si va a guardar sus datos en archivos, ¿estarán en un formato personalizado o estándar?

Si utiliza un formato de archivo estándar, otros tipos de proyecto además de su proyecto podrán abrir y leer/escribir datos en ellos. Sin embargo, si utiliza un formato de archivo personalizado, solo el tipo de proyecto podrá abrir los y leer/escribir datos en ellos.

Si el proyecto utiliza archivos, debe personalizar el editor estándar. Si el proyecto no utiliza archivos, sino que utiliza elementos en una base de datos u otro repositorio, debe crear un editor personalizado.

- ¿Es necesario que el editor hospede controles ActiveX?

Si el editor hospeda controles ActiveX, implemente un editor de activación in situ, como se describe en [Activación en contexto](#). Si no hospeda controles ActiveX, utilice un editor de incrustación simplificado o personalice el Visual Studio editor predeterminado.

- ¿Su editor admitirá varias vistas? Debe admitir varias vistas si desea que las vistas del editor estén visibles al mismo tiempo que el editor predeterminado.

Si el editor necesita admitir varias vistas, los datos del documento y los objetos de vista de documento para el editor deben ser objetos independientes. Para obtener más información, consulte [Compatibilidad con varias vistas](#) de documentos.

Si el editor admite varias vistas, Visual Studio ¿tiene previsto utilizar la [VsTextBuffer](#) implementación de búfer de texto (objeto) del editor principal para el objeto de datos de documento? Es decir, ¿desea admitir la vista del editor Visual Studio en paralelo con el editor principal? La capacidad de hacer esto es la base del diseñador de formularios..

- Si necesita alojar un editor externo, ¿se Visual Studio puede incrustar el editor en el interior?

Si se puede incrustar, debe crear una ventana host para [IsDocumentInProject](#) el editor [VSDOCUMENTPRIORITY](#) externo y, a continuación, llamar al método y establecer el valor de enumeración en `DP_External`. Si el editor no se puede incrustar, el IDE creará automáticamente una ventana independiente para él.

En esta sección

[Tutorial: Crear un editor personalizado](#)

Explica cómo crear un editor personalizado.

[Tutorial: Agregar características a un editor personalizado](#)

Explica cómo agregar características a un editor personalizado.

[Inicialización del diseñador y configuración de metadatos](#)

Explica cómo inicializar un diseñador.

[Soporte de deshacer de suministros a los diseñadores](#)

Explica cómo proporcionar soporte de deshacer para diseñadores.

[Coloración de sintaxis en editores personalizados](#)

Explica la diferencia entre el color de sintaxis en el editor principal y en los editores personalizados.

[Datos de documentos y vista de documentos en editores personalizados](#)

Explica cómo implementar datos de documentos y vistas de documentos en editores personalizados.

Secciones relacionadas

[Interfaces heredadas en el editor](#)

Explica cómo acceder al editor principal mediante la API heredada.

[Desarrollar un servicio de lenguaje heredado](#)

Explica cómo implementar un servicio de lenguaje.

[Ampliar otras partes de Visual Studio](#)

Explica cómo crear elementos de Visual Studio interfaz de usuario que coinciden con el resto de .

Vea también

- [IVsEditorFactory](#)

Tutorial: Crear un editor personalizado

06/04/2020 • 2 minutes to read • [Edit Online](#)

La plantilla de proyecto de VSPackage puede crear un editor personalizado simple en C++. La plantilla de proyecto de VSPackage ya no admite proyectos de Visual Basic o de C. Para obtener más información, vea SDK de [Visual Studio](#).

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

La plantilla de proyecto Paquete de Visual Studio

Puede encontrar la plantilla de proyecto Paquete de Visual Studio en el **cuadro de** diálogo Nuevo proyecto en la carpeta **Desmiensibilidad** de C++.

Para crear un VSPackage mediante la plantilla de paquete de Visual Studio

1. Cree un proyecto con la plantilla Paquete de Visual Studio.
2. Seleccione la opción **Editor personalizado** y haga clic en **Siguiente**. Aparece la página Opciones del editor.
3. Escriba el nombre del editor en el cuadro Nombre del **editor**. Escriba la extensión de archivo que desea asociar con el editor en el cuadro **Extensión de** archivo. El editor está disponible para archivos con esta extensión. La extensión de archivo solo está registrada para Visual Studio, no para Windows. Escriba el nombre de archivo predeterminado para los nuevos documentos creados con el editor en el cuadro Nombre de archivo **predeterminado**.
4. Haga clic en **Finalizar** para crear el VSPackage en la carpeta que especificó.

Para probar el editor personalizado

1. En el menú **Archivo**, seleccione **Nuevo** y, a continuación, haga clic en **Archivo**.
2. En el panel **Plantillas instaladas** del cuadro de diálogo **Nuevo archivo**, seleccione la plantilla de archivo y, a continuación, el tipo de archivo que registró.
3. Haga clic en **Abrir** para ver y editar el documento.

El editor admite operaciones de cortar y pegar, buscar y reemplazar y abrir y cargar.

Vea también

- [VSPackages](#)

Tutorial: Agregar características a un editor personalizado

06/04/2020 • 10 minutes to read • [Edit Online](#)

Después de crear un editor personalizado, puede agregarle más características.

Para crear un editor para un VSPackage

1. Cree un editor personalizado mediante la plantilla de proyecto Paquete de Visual Studio.

Para obtener más información, consulte [Tutorial: Crear un editor personalizado](#).

2. Decida si desea que el editor admita una sola vista o varias vistas.

Un editor que admite el comando **Nueva ventana** o tiene vista de formulario y vista de código, requiere objetos de datos de documento y objetos de vista de documento independientes. En un editor que admite una sola vista, el objeto de datos de documento y el objeto de vista de documento se pueden implementar en el mismo objeto.

Para obtener un ejemplo de varias vistas, consulte [Compatibilidad con varias vistas](#) de documentos.

3. Implemente una fábrica de [IVsEditorFactory](#) editores configurando la interfaz.

Para obtener más información, consulte Fábricas de [editores](#).

4. Decida si desea que el editor utilice la activación in situ o la incrustación simplificada para administrar la ventana del objeto de vista de documento.

Una ventana simplificada del editor de incrustación hospeda una vista de documento estándar, mientras que una ventana del editor de activación in situ hospeda un control ActiveX u otro objeto activo como su vista de documento. Para obtener más información, consulte [Incrustación simplificada](#) y [Activación in situ](#).

5. Implemente [IOleCommandTarget](#) la interfaz para controlar los comandos.

6. Proporcione la persistencia del documento y la respuesta a los cambios de archivo externo:

- a. Para conservar el [IVsPersistDocData2](#) archivo, implemente y [IPersistFileFormat](#) en el objeto de datos de documento del editor.
- b. Para responder a los [IVsFileChangeEx](#) cambios de archivos externos, implemente y [IVsDocDataFileChangeControl](#) en el objeto de datos de documento del editor.

NOTE

Llame `QueryService` [SVsFileChangeEx](#) para obtener un `IVsFileChangeEx` puntero a .

7. Coordinar eventos de edición de documentos con control de código fuente. Siga estos pasos:

- a. Obtenga un `IVsQueryEditQuerySave2` puntero `QueryService` llamando [SVsQueryEditQuerySavea](#) .
- b. Cuando se produzca el primer [QueryEditFiles](#) evento de edición, llame al método.

Este método solicita al usuario que desproteja el archivo si aún no está desprotegido. Asegúrese de controlar una condición de "archivo no desprotegido" para evitar errores.

- c. Del mismo modo, antes [QuerySaveFile](#) de guardar el archivo, llame al método.

Este método solicita al usuario que guarde el archivo si no se ha guardado o si ha cambiado desde la última vez que se guardó.

8. Habilite la ventana **Propiedades** para mostrar las propiedades del texto seleccionado en el editor. Siga estos pasos:
 - a. Llame [OnSelectChange](#) cada vez que cambie la [ISelectionContainer](#)selección de texto, pasando la implementación de .
 - b. Llame `QueryService` [STrackSelection](#) al servicio para [ITrackSelection](#) obtener un puntero a .
9. Permitir a los usuarios arrastrar y colocar elementos entre el editor y el cuadro de **herramientas**, o entre editores externos (como Microsoft Word) y **el cuadro de herramientas**. Siga estos pasos:
 - a. Implemente `IDropTarget` en el editor para alertar al IDE de que el editor es un destino de colocación.
 - b. Implemente [IVsToolboxUser](#) la interfaz en la vista para que el editor pueda habilitar y deshabilitar elementos en el **cuadro de herramientas**.
 - c. Implemente [ResetDefaults](#) y `QueryService` [SVsToolbox](#) llame al servicio para [IVsToolbox2](#) [IVsToolbox3](#) obtener un puntero a las interfaces y.

Estos pasos permiten que el `VSPackage` agregue nuevos elementos al **cuadro de herramientas**.

10. Decida si desea otras características opcionales para el editor.
 - Si desea que el editor admita [IVsFindTarget](#) buscar y reemplazar comandos, implemente .
 - Si desea utilizar una ventana de herramientas de `IVsDocOutlineProvider` esquema de documento en el editor, implemente .
 - Si desea utilizar una barra de estado [IVsStatusbarUser](#) en `QueryService` [SVsStatusbar](#) el editor, `IVsStatusBar` implemente y llame para obtener un puntero a .

Por ejemplo, un editor puede mostrar información de línea / columna, modo de selección (flujo / cuadro) y modo de inserción (insertar / sobreatacar).

- Si desea que el `Undo` editor admita el comando, el método recomendado es usar el modelo de administrador de deshacer OLE. Como alternativa, puede hacer que `Undo` el editor controle el comando directamente.
11. Crear información del registro, incluidos los GUID para el `VSPackage`, los menús, el editor y otras características.

A continuación se muestra un ejemplo genérico de código que pondría en el script de archivo `.rgs` para demostrar cómo registrar correctamente un editor.

```

NoRemove Editors
{
    ForceRemove {...guidEditor...} = s 'RTF Editor'
    {
        val Package = s '{...guidVsPackage...}'
        ForceRemove Extensions
        {
            val rtf = d 50
        }
    }
}
NoRemove Menus
{
    val {...guidVsPackage...} = s ',203,11'
}

```

12. Implemente la compatibilidad con la Ayuda contextual.

Este paso le permite proporcionar ayuda F1 y compatibilidad con la ventana Ayuda dinámica para los elementos del editor. Para obtener más información, consulte [Cómo: proporcionar contexto para editores](#).

13. Exponga un modelo de objetos `IDispatch` de automatización desde el editor implementando la interfaz.

Para obtener más información, consulta [Contributing to the Automation Model](#).

Programación sólida

- La instancia del editor se crea [CreateEditorInstance](#) cuando el IDE llama al método. Si el editor admite `CreateEditorInstance` varias vistas, crea los datos del documento y los objetos de vista de documento. Si el objeto de datos del documento `punkDocDataExisting` ya `IVsEditorFactory::CreateEditorInstance` está abierto, se pasa un valor no nulo a . La implementación de fábrica del editor debe determinar si un objeto de datos de documento existente es compatible consultando las interfaces adecuadas en él. Para obtener más información, consulte Compatibilidad con varias vistas de [documentos](#).
- Si utiliza el enfoque de incrustación simplificado, implemente la [IVsWindowPane](#) interfaz.
- Si decide utilizar la activación in situ, implemente las siguientes interfaces:

[IOleObject](#)

[IOleInPlaceActiveObject](#)

[IOleInPlaceComponent](#)

NOTE

La `IOleInPlaceComponent` interfaz se utiliza para evitar la fusión de menús OLE 2.

La `IOleCommandTarget` implementación controla comandos como **Cortar**, **Copiar** y **Pegar**. Al `IOleCommandTarget` implementar , decida si el editor requiere su propio archivo `.vsct` para definir su Visual Studio propia estructura de menú de comandos o si puede implementar comandos estándar definidos por . Normalmente, los editores usan y extienden los menús del IDE y definen sus propias barras de herramientas. Sin embargo, a menudo es necesario que un editor defina sus propios comandos específicos además de usar el conjunto de comandos estándar del IDE. El editor debe declarar los comandos estándar que utiliza y, a continuación, definir los nuevos comandos, menús contextuales, menús de nivel superior y barras de herramientas en un archivo `.vsct`. Si crea un editor de activación in situ, implemente [IOleInPlaceComponent](#) y defina los menús y barras de herramientas para el editor en un archivo `.vsct` en lugar de usar la combinación de menús OLE 2.

- Para evitar el hacinamiento del comando de menú en la interfaz de usuario, debe usar los comandos existentes en el IDE antes de inventar nuevos comandos. Los comandos compartidos se definen en *SharedCmdDef.vsct* y *ShellCmdDef.vsct*. Estos archivos se instalan de forma predeterminada en el subdirectorio VisualStudioIntegration-Common-Inc de la Visual Studio SDK instalación.
- `ISelectionContainer` pueden expresar selecciones individuales y múltiples. Cada objeto seleccionado se `IDispatch` implementa como un objeto.
- El IDE `IObjectUndoManager` se implementa como [CreateInstance](#) un servicio accesible desde un [CreateInstance](#) objeto o como un objeto que se puede crear una instancia a través de . El editor implementa `IObjectUndoUnit` la `Undo` interfaz para cada acción.
- Hay dos lugares en los que un editor personalizado puede exponer objetos de automatización:
 - `Document.Object`
 - `Window.Object`

Vea también

- [Contribuir al modelo de automatización](#)

Inicialización del diseñador y configuración de metadatos

06/04/2020 • 8 minutes to read • [Edit Online](#)

La manipulación de los metadatos y atributos de filtro asociados a un diseñador o componente [Type](#) de diseñador proporciona un mecanismo para que las aplicaciones definan qué herramientas usa un diseñador determinado para controlar diferentes objetos (como estructuras de datos, clases o entidades gráficas), cuando el diseñador está disponible y cómo el IDE de Visual Studio está configurado para admitir el diseñador (por ejemplo, qué categoría o pestaña de **Toolbox** está disponible).

Proporciona Visual Studio SDK varios mecanismos para facilitar el control de la inicialización de un diseñador o componente de diseñador y la manipulación de sus metadatos por un **VSPackage**.

Inicializar metadatos e información de configuración

Dado que se cargan a petición, **VSPackages** puede no haber sido cargado por el entorno de Visual Studio antes de la creación de instancias de un diseñador. Por lo tanto, **VSPackages** no puede usar el mecanismo estándar para configurar [DesignerCreated](#) un diseñador o componente de diseñador en la creación, que es controlar un evento. En su lugar, un **VSPackage** [DesignSurfaceExtension](#) implementa una instancia de la interfaz y se registra para proporcionar personalizaciones, denominadas extensiones de superficie de diseño.

Personalizar la inicialización

Personalizar un diseñador, un componente o una superficie de diseñador implica:

1. Modificar los metadatos del diseñador [Type](#) y cambiar eficazmente el acceso o la conversión de un determinado.

Esto se hace típicamente [UITypeEditor](#) [TypeConverter](#) a través de los mecanismos o.

Por ejemplo, [System.Windows.Forms](#) cuando se inicializan diseñadores basados [UITypeEditor](#) en [Image](#) -, el entorno de Visual Studio modifica los objetos for utilizados con el diseñador para usar el administrador de recursos para obtener mapas de bits en lugar del sistema de archivos.

2. Integración con el entorno, por ejemplo, suscribiéndose a eventos u obteniendo información de configuración del proyecto. Puede obtener información de configuración del proyecto [ITypeResolutionService](#) y suscribirse a eventos obteniendo la interfaz.
3. Modificación del entorno de usuario activando las categorías de **Toolbox** adecuadas o restringiendo la aplicabilidad del diseñador aplicando una instancia de la [ToolboxItemFilterAttribute](#) clase al diseñador.

Inicialización del diseñador por un **VSPackage**

Un **VSPackage** debe controlar la inicialización del diseñador mediante:

1. Crear un objeto [DesignSurfaceExtension](#) que implementa la clase.

NOTE

La [DesignSurfaceExtension](#) clase nunca debe implementarse en [Package](#) el mismo objeto que la clase.

2. Registrar la clase que [DesignSurfaceExtension](#) implementa como proporcionar compatibilidad con las extensiones de diseñador de **VSPackage**. Registre la clase aplicando

[DesignSurfaceExtensionAttribute](#) instancias de , [ProvideObjectAttribute](#), y [ProvideServiceAttribute](#) a la [Package](#) clase que proporciona la implementación de [VSPackage](#) de .

Cada vez que se crea un diseñador o componente de diseñador, el entorno de Visual Studio:

- Tiene acceso a cada proveedor de extensión de superficie de diseño registrado.
- Crea una instancia e inicializa una instancia del [DesignSurfaceExtension](#) objeto de cada proveedor de extensión de superficie de diseño.
- Llama al método o [OnDesignerCreated](#) método [OnComponentCreated](#) de cada proveedor de extensión de superficie de diseño (según corresponda).

Al implementar [DesignSurfaceExtension](#) el objeto como miembro de un [VSPackage](#), es importante entender que:

- El entorno de Visual Studio no proporciona ningún control `DesignSurfaceExtension` sobre qué metadatos u otras opciones de configuración modifica un proveedor determinado. Es posible que dos `DesignSurfaceExtension` o más proveedores modifiquen la misma característica de diseñador de maneras conflictivas, siendo la modificación final definitiva. Es indeterminado qué modificación se aplica por última vez.
- Es posible restringir explícitamente una implementación del [DesignSurfaceExtension](#) objeto a diseñadores [ToolboxItemFilterAttribute](#) específicos aplicando instancias de esa implementación. Para obtener más información sobre [ToolboxItemFilterAttribute](#) el [ToolboxItemFilterType](#) filtrado de elementos de [Toolbox](#), vea el archivo y .

Aprovisionamiento de metadatos adicionales

Un [VSPackage](#) puede cambiar la configuración de un diseñador o componente de diseñador que no sea en tiempo de diseño.

La [ProvideDesignerMetadataAttribute](#) clase se puede usar mediante programación o aplicar a un [VSPackage](#) que proporciona un diseñador.

Una instancia [ProvideDesignerMetadataAttribute](#) de la clase se utiliza para modificar los metadatos de los componentes creados en una superficie de diseño. Por ejemplo, se podría reemplazar un [CommonDialog](#) explorador de propiedades predeterminado utilizado por los objetos con un explorador de propiedades personalizado.

Las modificaciones proporcionadas [ProvideDesignerMetadataAttribute](#) por una instancia de [Package](#) aplicado a la implementación de un [VSPackage](#) de pueden tener uno de dos ámbitos:

- Global -- para todas las nuevas instancias de un componente determinado
- Local: pertenece solo a la instancia del componente creado en una superficie de diseño proporcionada por el [VSPackage](#) actual.

La `IsGlobal` propiedad [ProvideDesignerMetadataAttribute](#) de la instancia aplicada a [Package](#) la implementación de un [VSPackage](#) de determina este ámbito.

Aplicar el atributo a [Package](#) una [IsGlobal](#) implementación [ProvideDesignerMetadataAttribute](#) de con `true` la propiedad del objeto establecido en , como se muestra a continuación, cambia el explorador para todo el entorno de Visual Studio:

```
[ProvideDesignerMetadata(typeof(Color), typeof(CustomBrowser), IsGlobal=true)]
```

```
internal class MyPackage : Package {}
```

Si la marca global `false` se estableció en , el cambio de metadatos es local para el diseñador actual admitido por el

VSPackage actual:

```
[ProvideDesignerMetadata(typeof(Color), typeof(CustomBrowser), IsGlobal=false )]
```

```
internal class MyPackage : Package {}
```

NOTE

La superficie de diseño solo admite la creación de componentes y, por lo tanto, solo los componentes pueden tener metadatos locales. En el ejemplo anterior, intentamos modificar una propiedad, como la `Color` propiedad de un objeto. Si `false` se pasó para la `CustomBrowser` marca global, nunca aparecería `Color` porque el diseñador nunca crea realmente una instancia de . Establecer la marca `false` global en es útil para los componentes, como controles, temporizadores y cuadros de diálogo.

Vea también

- [DesignSurfaceExtension](#)
- [DesignSurfaceExtensionAttribute](#)
- [ToolboxItemFilterType](#)
- [Amplíe el soporte en tiempo de diseño](#)

Soporte de deshacer de suministros a los diseñadores

06/04/2020 • 6 minutes to read • [Edit Online](#)

Los diseñadores, al igual que los editores, normalmente necesitan admitir operaciones de deshacer para que los usuarios puedan revertir sus cambios recientes al modificar un elemento de código.

La mayoría de los diseñadores implementados en Visual Studio tienen compatibilidad con "deshacer" proporcionada automáticamente por el entorno.

Implementaciones de diseñador que necesitan proporcionar compatibilidad con la característica de deshacer:

- Proporcionar administración de deshacer mediante la implementación de la clase base abstracta [UndoEngine](#)
- Suelen dar persistencia y [IDesignerSerializationService](#) [IComponentChangeService](#) compatibilidad con CodeDOM implementando las clases y.

Para obtener más información sobre cómo escribir diseñadores con .NET Framework, vea Extender compatibilidad en [tiempo de diseño](#).

Proporciona Visual Studio SDK una infraestructura de deshacer predeterminada:

- Proporcionar implementaciones de administración [OleUndoEngine](#) [OleUndoEngine.UndoUnit](#) de deshacer a través de las clases y.
- Proporcionar persistencia y compatibilidad con [CodeDomComponentSerializationService](#) CodeDOM a través de las implementaciones y [IComponentChangeService](#) los valores predeterminados.

Obtener soporte de deshacer automáticamente

Cualquier diseñador creado en Visual Studio tiene compatibilidad de deshacer automática y completa si, el diseñador:

- Hace uso [Control](#) de una clase basada para su interfaz de usuario.
- Emplea el sistema estándar de generación y análisis de código basado en CodeDOM para la generación y persistencia de código.

Para obtener más información acerca de cómo trabajar con compatibilidad con Visual Studio CodeDOM, vea [Generación y compilación](#) dinámica de código fuente .

Cuándo usar el soporte explícito de deshacer del diseñador

Los diseñadores deben proporcionar su propia administración de deshacer si utilizan una interfaz gráfica [Control](#) de usuario, denominada adaptador de vista, distinta de la proporcionada por .

Un ejemplo de esto podría ser la creación de un producto con una interfaz de diseño gráfico basada en web en lugar de una interfaz gráfica basada en .NET Framework.

En estos casos, sería necesario registrar este adaptador [ProvideViewAdapterAttribute](#) de vista con Visual Studio mediante , y proporcionar administración de deshacer explícita.

Los diseñadores deben proporcionar CodeDOM y compatibilidad con la persistencia si [System.CodeDom](#) no usan el modelo de generación de código de Visual Studio proporcionado en el espacio de nombres.

Deshacer características de soporte del diseñador

El SDK de entorno proporciona implementaciones predeterminadas de las interfaces necesarias para proporcionar compatibilidad de deshacer que pueden usar los diseñadores que no usan [Control](#) clases basadas para sus interfaces de usuario o el modelo de persistencia y CodeDOM estándar.

La [OleUndoEngine](#) clase se deriva de [UndoEngine](#) la clase .NET Framework mediante una implementación de la [IOleUndoManager](#) clase para administrar las operaciones de deshacer.

Visual Studio proporciona la siguiente característica para deshacer el diseñador:

- Funcionalidad de deshacer vinculada entre varios diseñadores.
- Las unidades secundarias dentro de [IOleUndoUnit](#) un [IOleParentUndoUnit](#) [OleUndoEngine.UndoUnit](#) diseñador pueden interactuar con sus padres implementando y en .

El SDK de entorno proporciona CodeDOM y compatibilidad con la persistencia al proporcionar:

- [CodeDomComponentSerializationService](#) como una implementación de la [IDesignerSerializationService](#)
- A [IComponentChangeService](#) proporcionado por el host de diseño de Visual Studio.

Use las características del SDK de entorno para proporcionar compatibilidad con Deshacer

Para obtener compatibilidad de deshacer, un objeto que implementa [OleUndoEngine](#) un diseñador [IServiceProvider](#) debe crear instancias e inicializar una instancia de la clase con una implementación válida. Esta [IServiceProvider](#) clase debe proporcionar los siguientes servicios:

- [IDesignerHost](#).
- [IDesignerSerializationService](#)

Los diseñadores que usan la [CodeDomComponentSerializationService](#) serialización Visual Studio SDK de Visual Studio [IDesignerSerializationService](#) CodeDOM pueden optar por usar proporcionado como su implementación de la .

En este caso, la [IServiceProvider OleUndoEngine](#) clase proporcionada al constructor debe [IDesignerSerializationService](#) devolver este objeto como una implementación de la clase.

- [IComponentChangeService](#)

Se garantiza [DesignSurface](#) que los diseñadores que usan el valor predeterminado [IComponentChangeService](#) proporcionado por el host de diseño de Visual Studio tengan una implementación predeterminada de la clase.

Los diseñadores que implementan un [OleUndoEngine](#) mecanismo de deshacer basado realiza un seguimiento automático de los cambios si:

- Los cambios de [TypeDescriptor](#) propiedad se realizan a través del objeto.
- [IComponentChangeService](#) los eventos se generan manualmente cuando se confirma un cambio que se puede deshacer.
- La modificación en el diseñador [DesignerTransaction](#) se creó en el contexto de un archivo .
- El diseñador elige crear explícitamente unidades de deshacer mediante la unidad [UndoEngine.UndoUnit](#) de deshacer estándar [OleUndoEngine.UndoUnit](#) proporcionada por una [UndoEngine.UndoUnit](#) implementación o la [IOleUndoUnit IOleParentUndoUnit](#) implementación específica de Visual Studio , que deriva y también proporciona una implementación de ambos y .

Vea también

- [UndoEngine](#)
- [OleUndoEngine](#)
- [Amplíe el soporte en tiempo de diseño](#)

Colores de sintaxis en editores personalizados

06/04/2020 • 4 minutes to read • [Edit Online](#)

Los editores del SDK de Visual Studio Environment, incluido el editor principal, usan servicios de lenguaje para identificar elementos sintácticos específicos y mostrarlos con colores especificados para una vista de documento determinada.

Requisitos de coloración

Todos los editores que implementan el colorante de un servicio de lenguaje deben:

1. Utilice un [IVsTextBuffer](#) objeto que se implementa para administrar [IVsTextView](#) el texto que se va a colorear y un objeto que se va a implementar para proporcionar una vista de documento del texto.
2. Obtener una interfaz para un servicio de lenguaje determinado consultando el proveedor de servicios de VSPackage mediante el GUID de identificación del servicio de idiomas.
3. Llame [SetLanguageServiceID](#) al método del [IVsTextBuffer](#) objeto que implementa . Este método asocia el [IVsTextBuffer](#) servicio de lenguaje con la implementación que el VSPackage usa para administrar el texto que se va a colorear.

Core Editor Uso del Colorizador de un Servicio de Lenguaje

Cuando un servicio de lenguaje con un colorante se obtiene mediante una instancia del editor principal, el análisis y la representación de texto por el colorizador de un servicio de lenguaje se produce automáticamente sin necesidad de intervención adicional por su parte.

El IDE de forma transparente:

- Llama al colorante según sea necesario para analizar y analizar [IVsTextBuffer](#) el texto a medida que se agrega o modifica en la implementación de .
- Garantiza que la visualización proporcionada por [IVsTextView](#) la vista de documento proporcionada por la implementación se actualiza y se vuelve a pintar con la información devuelta por el colorante.

Uso no básico del editor de un servicio de lenguaje colorante

Las instancias de editor no principales también pueden usar el servicio de coloración de sintaxis de un servicio de lenguaje, pero deben recuperar y aplicar explícitamente el colorante del servicio y volver a pintar sus propias vistas de documento.

Para ello, un editor no principal debe:

1. Obtener el objeto colorizador de un [IVsColorizer](#) servicio [IVsColorizer2](#) de lenguaje (que implementa y). El VSPackage hace esto [GetColorizer](#) mediante una llamada al método en la interfaz del servicio de lenguaje.
2. Llame [ColorizeLine](#) al método para solicitar que se coloree un intervalo determinado de texto.

El [ColorizeLine](#) método devuelve una matriz de valores, uno para cada letra en el intervalo de texto que se está coloreando. También identifica el intervalo de texto como un tipo determinado de elemento coloreable, como un comentario, una palabra clave o un tipo de datos.

3. Utilice la información de [ColorizeLine](#) coloración devuelta para volver a pintar y mostrar su texto.

NOTE

Además de usar el colorizador de un servicio de lenguaje, un VSPackage puede elegir usar el mecanismo de coloración de texto del SDK de entorno de Visual Studio de uso general. Para obtener más información sobre este mecanismo, consulte [Uso de fuentes y colores](#).

Vea también

- [Colores de la sintaxis en un servicio de lenguaje heredado](#)
- [Implementación de colores de la sintaxis](#)
- [Uso de elementos coloreables integrados](#)
- [Elementos coloreables personalizados](#)

Datos de documentos y vista de documentos en editores personalizados

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un editor personalizado consta de dos partes: un objeto de datos de documento y un objeto de vista de documento. Como sugieren los nombres, el objeto de datos del documento representa los datos de texto que se van a mostrar. Del mismo modo, el objeto de vista de documento (o "vista") representa una o varias ventanas en las que se va a mostrar el objeto de datos del documento.

Objeto de datos de documento

Un objeto de datos de documento es una representación de datos de texto en el búfer de texto. Es un objeto COM que almacena texto del documento y otra información. El objeto de datos de documento también controla la persistencia del documento y habilita varias vistas de sus datos. Para obtener más información, vea

[DocumentData\[String\]](#) y [Documento de Windows](#).

Los editores y diseñadores [VsTextBuffer](#) personalizados pueden optar por usar el objeto o su propio búfer personalizado. [VsTextBuffer](#) sigue el modelo de incrustación simplificado para un editor estándar, admite varias vistas y proporciona interfaces de eventos que se utilizan para administrar varias vistas.

Objeto de vista de documento

Una ventana que muestra código y otro texto se conoce como vista o vista de documento. Al crear un editor, puede elegir una sola vista, en la que el texto se muestra en una sola ventana. O bien, puede elegir una vista múltiple, en la que el texto se muestra en más de una ventana. Su elección depende de su aplicación. Por ejemplo, si necesita una edición en paralelo, elegiría varias vistas. Cada vista está asociada a una entrada en la tabla de documentos en ejecución (RDT) del entorno de desarrollo integrado (IDE). Las ventanas de vista [IVsHierarchy](#) pertenecen a un proyecto o a un objeto.

Si el editor admite varias vistas de un objeto de datos de documento, los datos del documento y los objetos de vista de documento deben ser independientes. De lo contrario, se pueden agrupar. Para obtener más información, consulte [Compatibilidad con varias vistas](#) de documentos.

El IDE notifica las vistas sobre los eventos (por ejemplo, cuando se cierra una solución que contiene un documento) haciendo coincidir un identificador de elemento (ItemID) para cada entrada de la tabla de documento sin ejecución. Para obtener más información al respecto, consulte Ejecución de la tabla de [documentos](#).

Hay dos opciones para crear una vista para un editor personalizado. Uno es el modelo de activación en contexto, donde la vista se hospeda en una ventana mediante un control ActiveX o un objeto de datos de documento. El segundo es el modelo de incrustación Visual Studio simplificado, donde la vista está hospedada y [IVsWindowPane](#) se implementa para controlar los comandos de ventana. Para obtener información sobre el modelo de activación in situ, consulte [Activación en contexto](#). Para obtener información sobre el modelo de incrustación simplificado, consulte [Incrustación simplificada](#).

Vea también

- [Soporta múltiples vistas de documentos](#)
- [Incrustación simplificada](#)
- [Cómo: Adjuntar vistas a los datos del documento](#)

- Gestión del soporte de bloqueo de documentos
- Vistas de una y varias pestañas
- Guardar un documento estándar
- Persistencia y la tabla de documentos en ejecución
- Determinar qué editor abre un archivo en un proyecto

Compatibilidad con vistas de varios documentos

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede proporcionar más de una vista de un documento creando datos de documento y objetos de vista de documento independientes para el editor. Algunos casos en los que una vista de documento adicional sería útil son:

- Compatibilidad con ventanas nuevas: desea que el editor proporcione dos o más vistas del mismo tipo, de modo que un usuario que ya tiene una ventana abierta en el editor pueda abrir una nueva ventana seleccionando el comando **Nueva ventana** en el menú **Ventana**.
- Compatibilidad con la vista de formulario y código: desea que el editor proporcione vistas de diferentes tipos. Visual Basic, por ejemplo, proporciona una vista de formulario y una vista de código.

Para obtener más información acerca de esto, vea el Procedimiento `CreateEditorInstance` en el archivo `EditorFactory.cs` en el proyecto de editor personalizado creado por la plantilla de paquete de Visual Studio. Para obtener más información acerca de este proyecto, vea [Tutorial: crear un editor personalizado](#).

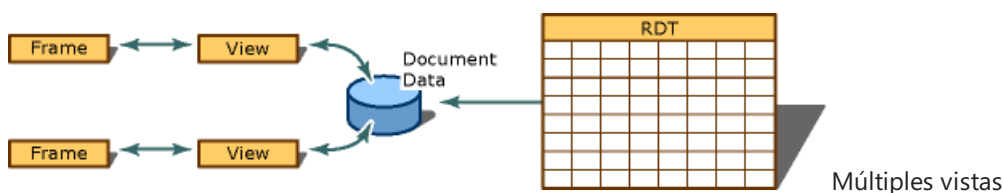
Sincronización de vistas

Al implementar varias vistas, el objeto de datos de documento es responsable de mantener todas las vistas sincronizadas con los datos. Puede utilizar las interfaces `VsTextBuffer` de control de eventos para sincronizar varias vistas con los datos.

Si no utiliza `VsTextBuffer` el objeto para sincronizar varias vistas, debe implementar su propio sistema de eventos para controlar los cambios realizados en el objeto de datos de documento. Puede utilizar diferentes niveles de granularidad para mantener varias vistas actualizadas. Con una configuración de granularidad máxima, a medida que escribe en una vista, las otras vistas se actualizan inmediatamente. Con una granularidad mínima, otras vistas no se actualizan hasta que se activan.

Determinar si los datos del documento ya están abiertos

La tabla de documentos en ejecución (RDT) en el entorno de desarrollo integrado (IDE) ayuda a realizar un seguimiento de si los datos de un documento ya están abiertos, como se muestra en el diagrama siguiente.



De forma predeterminada, cada vista (objeto de vista de `VsWindowFrame` documento) está contenida en su propio marco de ventana (). Sin embargo, como ya se ha indicado, los datos del documento se pueden mostrar en varias vistas. Para habilitar esto, Visual Studio comprueba el RDT para determinar si el documento en cuestión ya está abierto en un editor. Cuando el IDE llama `CreateEditorInstance` para crear el editor, `punkDocDataExisting` un valor no NULL devuelto en el parámetro indica que el documento ya está abierto en otro editor. Para obtener más información sobre cómo funciona el RDT, consulte Ejecución de tabla de [documentos](#).

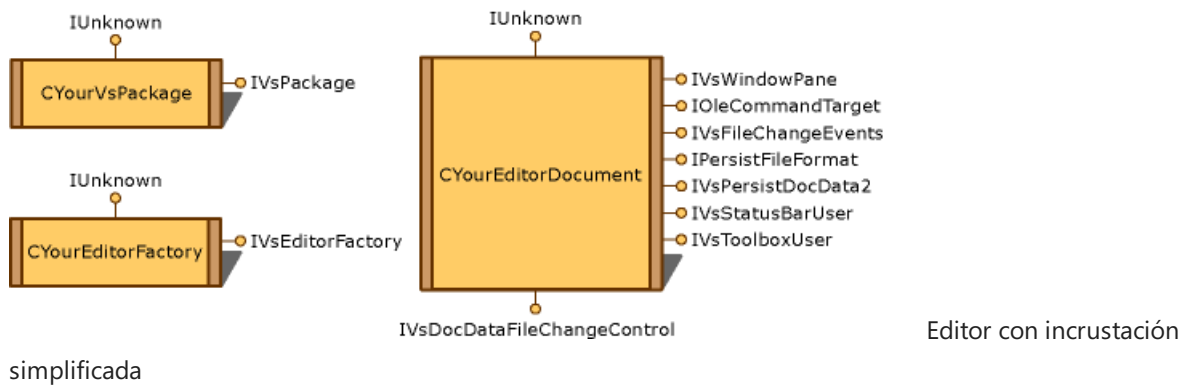
En `IvsEditorFactory` la implementación, examine el `punkDocDataExisting` objeto de datos de documento devuelto para determinar si los datos del documento son adecuados para el editor. (Por ejemplo, un editor HTML solo debe mostrar los datos HTML.) Si es apropiado, el generador del editor debe proporcionar una segunda vista para los datos. Si `punkDocDataExisting` el parámetro `NULL` no lo es, es posible que el objeto de datos del

documento esté abierto en otro editor o, lo que es más probable, que los datos del documento ya estén abiertos en una vista diferente con el mismo editor. Si los datos del documento están abiertos en un editor diferente que el generador del editor no admite, Visual Studio no puede abrir el generador del editor. Para obtener más información, consulte [Cómo: Adjuntar vistas a datos](#) de documento .

Inserción simplificada

06/04/2020 • 2 minutes to read • [Edit Online](#)

La incrustación simplificada se habilita en un editor cuando su objeto de vista Visual Studio de documento [IVsWindowPane](#) está parented to (es decir, hecho un elemento secundario de) y la interfaz se implementa para controlar sus comandos de ventana. Los editores de incrustación simplificados no pueden hospedar controles activos. Los objetos utilizados para crear un editor con incrustación simplificada se muestran en la siguiente ilustración.



NOTE

De los objetos de `CYourEditorFactory` esta ilustración, solo se requiere el objeto para crear un editor estándar basado en archivos. Si va a crear un editor personalizado, `IVsPersistDocData2` no es necesario implementar, porque es probable que el editor tenga su propio mecanismo de persistencia privada. Para los editores no personalizados, sin embargo, debe hacerlo.

Todas las interfaces implementadas para crear un editor `CYourEditorDocument` con incrustación simplificada están contenidas en el objeto. Sin embargo, para admitir varias vistas de datos de documento, divida las interfaces en datos independientes y vea objetos como se indica en la tabla siguiente.

INTERFAZ	UBICACIÓN DE LA INTERFAZ	USO
IVsWindowPane	Ver	Proporciona conexión a la ventana primaria.
IOleCommandTarget	Ver	Maneja comandos.
IVsStatusBarUser	Ver	Habilita las actualizaciones de la barra de estado.
IVsToolboxUser	Ver	Habilita los elementos de Toolbox .
IVsFileChangeEvents	data	Envía notificaciones cuando cambia el archivo.
IPersistFileFormat	data	Habilita la función Guardar como para un tipo de archivo.
IVsPersistDocData2	data	Habilita la persistencia del documento.

INTERFAZ	UBICACIÓN DE LA INTERFAZ	USO
IVsDocDataFileChangeControl	data	Permite la supresión de eventos de cambio de archivo, como la activación de recarga.

Cómo: Adjuntar vistas a los datos del documento

06/04/2020 • 3 minutes to read • [Edit Online](#)

Si tiene una nueva vista de documento, es posible que pueda adjuntarla a un objeto de datos de documento existente.

Para determinar si puede adjuntar una vista a un objeto de datos de documento existente

1. Implemente [CreateEditorInstance](#).
2. En la `IVsEditorFactory::CreateEditorInstance` implementación `QueryInterface` de , llame al objeto de `CreateEditorInstance` datos de documento existente cuando el IDE llame a la implementación.

La `QueryInterface` llamada le permite examinar el objeto de datos `punkDocDataExisting` de documento existente, que se especifica en el parámetro.

Las interfaces exactas que debe consultar, sin embargo, dependen del editor que abra el documento, como se describe en el paso 4.
3. Si no encuentra las interfaces adecuadas en el objeto de datos de documento existente, devuelva un código de error al editor que indique que el objeto de datos de documento es incompatible con el editor.

En la implementación del [OpenStandardEditor](#) IDE de , un cuadro de mensaje le notifica que el documento está abierto en otro editor y le pregunta si desea cerrarlo.
4. Si cierra este documento, Visual Studio llama al generador del editor por segunda vez. En esta llamada, el `DocDataExisting` parámetro es igual a NULL. A continuación, la implementación de fábrica del editor puede abrir el objeto de datos de documento en su propio editor.

NOTE

Para determinar si puede trabajar con un objeto de datos de documento existente, también puede Visual C++ usar el conocimiento privado de la implementación de la interfaz mediante la conversión de un puntero a la clase real de la implementación privada. Por ejemplo, todos los `IVsPersistFileFormat` editores estándar [IPersist](#) implementan , que hereda de . Por lo tanto, puede llamar `QueryInterface` a `GetClassID`, y si el identificador de clase en el objeto de datos de documento existente coincide con el identificador de clase de la implementación, puede trabajar con el objeto de datos de documento.

Programación sólida

Cuando Visual Studio llama [CreateEditorInstance](#) a la implementación del método, devuelve un `punkDocDataExisting` puntero al objeto de datos de documento existente en el parámetro, si existe uno. Examine el objeto de `punkDocDataExisting` datos de documento devuelto para determinar si el objeto de datos de documento es adecuado para el editor como se describe en la nota del paso 4 del procedimiento de este tema. Si es apropiado, el generador del editor debe proporcionar una segunda vista para los datos como se describe en [Admitir varias vistas](#) de documentos. Si no es así, debe mostrar un mensaje de error adecuado.

Vea también

- [Soporta múltiples vistas de documentos](#)

- [Datos de documentos y vista de documentos en editores personalizados](#)

Gestión del soporte de bloqueo de documentos

06/04/2020 • 7 minutes to read • [Edit Online](#)

La tabla de documentos en ejecución (RDT) mantiene un recuento de documentos abiertos y los bloqueos de edición que tienen. Puede colocar un bloqueo de edición en un documento en el RDT cuando se edita mediante programación en segundo plano sin que el usuario vea un documento abierto en una ventana de documento. Esta funcionalidad es utilizada a menudo por diseñadores que modifican varios archivos a través de una interfaz gráfica de usuario.

Escenarios de soporte de bloqueo de documentos

El archivo "a" depende del archivo "b"

Considere una situación en la que implemente un editor estándar "A" para el tipo de archivo "a", y cada archivo de tipo "a" tiene una referencia a (o dependencia de) un archivo de tipo "b". Existe un editor estándar "B" para archivos de tipo "b". Cuando el editor "A" abre el archivo "a" recupera la referencia al archivo correspondiente "b". El archivo "b" no se muestra, pero el editor "A" puede modificarlo. Editor "A" obtiene una referencia a los datos del [FindAndLockDocument](#) documento del archivo "b" del método y también mantiene un bloqueo de edición en el archivo "b". Después de que el editor "A" haya terminado de modificar el archivo "b" puede disminuir el recuento de bloqueo de edición en el archivo "b" llamando al [UnlockDocument](#) método. Puede omitir este paso si [FindAndLockDocument](#) ha llamado `dwRDTLockType` al método con el parámetro establecido en [_VSRDTFLAGS.RDT_NoLock](#).

El archivo "b" es abierto por un editor diferente

En el caso de que el archivo "b" ya esté abierto por el editor "B" cuando el editor "A" intenta abrirlo, hay dos escenarios independientes para manejar:

- Si el archivo "b" está abierto en un editor compatible, debe tener el editor [RegisterDocumentLockHolder](#) "A" registrar un bloqueo de edición de documento en el archivo "b" utilizando el método. Después de que el editor "A" haya terminado de modificar [UnregisterDocumentLockHolder](#) el archivo "b", anule el registro del bloqueo de edición de documentos utilizando el método.
- Si el archivo "b" está abierto de una manera incompatible, puede dejar que el intento de apertura del archivo "b" por el editor "A" falle, o puede dejar que la vista asociada con el editor "A" se abra parcialmente y mostrar un mensaje de error adecuado. El mensaje de error debe indicar al usuario que cierre el archivo "b" en el editor incompatible y, a continuación, vuelva a abrir el archivo "a" utilizando el editor "A". También puede implementar Visual Studio SDK [QueryCloseRunningDocument](#) el método para solicitar al usuario que cierre el archivo "b" que está abierto en el editor incompatible. Si el usuario cierra el archivo "b", la apertura del archivo "a" en el editor "A" continúa normalmente.

Consideraciones adicionales sobre el bloqueo de edición de documentos

Obtiene un comportamiento diferente si el editor "A" es el único editor que tiene un bloqueo de edición de documentos en el archivo "b" que si el editor "B" también tiene un bloqueo de edición de documento en el archivo "b". En Visual Studio, el Diseñador de **clases** es un ejemplo de un diseñador visual que no contiene un bloqueo de edición en el archivo de código asociado. Es decir, si el usuario tiene un diagrama de clases abierto en la vista de diseño y el archivo de código asociado abierto simultáneamente, y si el usuario modifica el archivo de código pero no guarda los cambios, los cambios también se pierden en el archivo de diagrama de clases (.cd). Si el Diseñador de **clases** tiene el único bloqueo de edición de documento en el archivo de código, no se pide al usuario que

guarde los cambios al cerrar el archivo de código. El IDE pide al usuario que guarde los cambios solo después de que el usuario cierre el **Diseñador de clases**. Los cambios guardados se reflejan en ambos archivos. Si tanto el Diseñador de **clases** como el editor de archivos de código mantuvieron bloqueos de edición de documentos en el archivo de código, se solicita al usuario que guarde al cerrar el archivo de código o el formulario. En ese momento, los cambios guardados se reflejan tanto en el formulario como en el archivo de código. Para obtener más información sobre los diagramas de clases, vea Trabajar con diagramas de [clases \(Diseñador de clases\)](#).

Tenga en cuenta que si necesita colocar un bloqueo de edición [IVsDocumentLockHolder](#) en un documento para un no editor, debe implementar la interfaz.

Muchas veces, un diseñador de interfaz de usuario que modifica los archivos de código mediante programación realiza cambios en más de un archivo. En tales [SaveItemsViaDlg](#) casos, el método controla el guardado de uno o más documentos mediante el cuadro de diálogo **¿Desea guardar los cambios en los siguientes elementos?**.

Vea también

- [Tabla de documentos en ejecución](#)
- [Persistencia y la tabla de documentos en ejecución](#)

Vistas de una sola pestaña y varias pestañas

06/04/2020 • 8 minutes to read • [Edit Online](#)

Un editor puede crear diferentes tipos de vistas. Un ejemplo es una ventana del editor de código, otro es un diseñador de formularios.

Una vista con varias pestañas es una vista que tiene varias fichas. Por ejemplo, el editor HTML tiene dos pestañas en la parte inferior: **Diseño** y **Origen**, cada una una de las dos una vista lógica. La vista de diseño muestra una página web representada, mientras que la otra muestra el código HTML que comprende la página web.

Acceso a vistas físicas

Las vistas físicas alojan objetos de vista de documento, cada uno de los cuales representa una vista de datos en el búfer, como código o un formulario. En consecuencia, cada objeto de vista de documento tiene una vista física (identificada por algo conocido como una cadena de vista física) y, por lo general, una única vista lógica.

En algunos casos, sin embargo, una vista física puede tener dos o más vistas lógicas. Algunos ejemplos son un editor que tiene una ventana dividida con vistas en paralelo, o un diseñador de formularios que tiene una vista de gui/diseño y una vista de código subyacente al formulario.

Para permitir que el editor tenga acceso a todas las vistas físicas disponibles, debe crear una cadena de vista física única para cada tipo de objeto de vista de documento que el generador del editor pueda crear. Por ejemplo, Visual Basic el generador de editores puede crear objetos de vista de documento para una ventana de código y una ventana del diseñador de formularios.

Creación de vistas con varias pestañas

Aunque un objeto de vista de documento debe estar asociado a una vista física a través de una cadena de vista física única, puede colocar varias pestañas dentro de la vista física para habilitar la visualización de datos de diferentes maneras. En esta configuración con varias pestañas, todas las pestañas están asociadas a la misma cadena de vista física, pero a cada pestaña se le da un GUID de vista lógica diferente.

Para crear una vista con varias pestañas para un editor, implemente la [IVsMultiViewDocumentView](#) interfaz y, a continuación, asocie un GUID de vista lógica diferente ([LogicalViewID](#)) a cada pestaña que cree.

El editor HTML de Visual Studio es un ejemplo de un editor con una vista de varias pestañas. Tiene pestañas **Diseño** y **Origen**. Para habilitar esto, se asocia una vista `LOGICALVIEWID_TextView` lógica diferente `LOGICALVIEWID_Code` a cada ficha, para la pestaña **Diseño** y para la pestaña **Origen**.

Al especificar la vista lógica adecuada, un `VSPackage` puede tener acceso a la vista que corresponde a un propósito determinado, como diseñar un formulario, editar código o depurar código. Sin embargo, una de las ventanas debe identificarse mediante la `LOGVIEWID_Primary` cadena NULL y esto debe corresponder a la vista lógica principal ().

En la tabla siguiente se enumeran los valores de vista lógica disponibles y su uso.

LOGVIEWID GUID	USO RECOMENDADO
<code>LOGVIEWID_Primary</code>	Vista predeterminada/primaria de la fábrica del editor. Todas las fábricas de editores deben admitir este valor. Esta vista debe usar la cadena NULL como su cadena de vista física. Se debe establecer al menos una vista lógica en este valor.

LOGVIEWID GUID	USO RECOMENDADO
LOGVIEWID_Debugging	Vista de depuración. Normalmente, LOGVIEWID_Debugging se asigna a LOGVIEWID_Code la misma vista que .
LOGVIEWID_Code	Vista iniciada por el comando Ver código .
LOGVIEWID_Designer	Vista iniciada por el comando Ver formulario .
LOGVIEWID_TextView	Vista del editor de texto. Esta es la IVsCodeWindow vista que devuelve, IVsTextView desde la que se puede acceder.
LOGVIEWID_UserChooseView	Solicita al usuario que elija qué vista utilizar.
LOGVIEWID_ProjectSpecificEditor	Pasado por el cuadro de diálogo Abrir con a OpenItem cuando el usuario elige la entrada "(Editor predeterminado del proyecto)".

Aunque los GUID de vista lógica son extensibles, solo puede usar los GUID de vista lógica definidos en el VSPackage.

Al apagarse, Visual Studio conserva el GUID del generador del editor y las cadenas de vista física asociadas a la ventana del documento para que se pueda usar para volver a abrir las ventanas de documento cuando se vuelva a abrir la solución. Solo las ventanas que están abiertas cuando se cierra una solución se conservan en el archivo de solución (.suo). Estos valores corresponden VSFPROPID_guidEditorType VSFPROPID_pszPhysicalView a los propid valores pasados en el parámetro en el [GetProperty](#) método.

Ejemplo

Este fragmento de [TextView](#) código ilustra cómo se usa [IVsCodeWindow](#) el objeto para tener acceso a una vista que implementa . En este caso, el [SVsUIShellOpenDocument](#) servicio [OpenDocumentViaProject](#) se [LOGVIEWID_TextView](#) utiliza para llamar y solicitar , que obtiene un puntero a un marco de ventana. Un puntero al objeto de vista [GetProperty](#) de documento se [VSFPROPID_DocView](#) obtiene llamando y especificando un valor de . Desde el objeto [QueryInterface](#) de vista [IVsCodeWindow](#) de documento, se llama para . La expectativa en este caso es que se devuelve un editor [GetProperty](#) de texto, por lo que el objeto de vista de documento devuelto en el método es una ventana de código.


```

HRESULT CFindTool::GotoFileLocation(const WCHAR * szFile, long iLine, long iStart, long iLen)
{
    HRESULT hr;
    if (NULL == szFile || !*szFile)
        return E_INVALIDARG;

    if (iLine == -1L)
        return S_FALSE;

    VSITEMID          itemid;
    VARIANT           var;
    RECT              rc;
    IVsUIShellOpenDocument * pOpenDoc    = NULL;
    IVsCodeWindow *      pCodeWin    = NULL;
    IVsTextView *        pTextView    = NULL;
    IVsUIHierarchy *      pHierarchy    = NULL;
    IVsWindowFrame *      pFrame      = NULL;
    IUnknown *           pUnk        = NULL;
    IVsHighlight *        pHighlight    = NULL;

    IfFailGo(CGlobalServiceProvider::HrQueryService(SID_SVsUIShellOpenDocument, IID_IVsUIShellOpenDocument,
(void **)&pOpenDoc));
    IfFailGo(pOpenDoc->OpenDocumentViaProject(szFile, LOGVIEWID_TextView, NULL, &pHierarchy, &itemid, &pFrame));
    pFrame->Show();
    VariantInit(&var);
    IfFailGo(pFrame->GetProperty(VSFPROPID_DocView, &var));
    if (VT_UNKNOWN != var.vt) { hr = E_FAIL; goto Error; }
    pUnk = V_UNKNOWN(&var);
    if (NULL != pUnk)
    {
        IfFailGo(pUnk->QueryInterface(IID_IVsCodeWindow, (void **)&pCodeWin));
        if (SUCCEEDED(hr = pCodeWin->GetLastActiveView(&pTextView)) ||
            SUCCEEDED(hr = pCodeWin->GetPrimaryView(&pTextView)) )
        {
            pTextView->SetSelection(iLine, iStart, iLine, iStart + iLen);
            // uncover selection
            IfFailGo(pTextView->QueryInterface(IID_IVsHighlight, (void **)&pHighlight));
            IfFailGo(SUCCEEDED(pHighlight->GetHighlightRect(&rc)));
            UncoverSelectionRect(&rc);
        }
    }
}

Error:
    CLEARINTERFACE(pHighlight);
    CLEARINTERFACE(pTextView);
    CLEARINTERFACE(pCodeWin);
    CLEARINTERFACE(pUnk);
    CLEARINTERFACE(pFrame);
    CLEARINTERFACE(pOpenDoc);
    CLEARINTERFACE(pHierarchy);
    RedrawWindow(m_hwndResults, NULL, NULL, RDW_ERASE|RDW_FRAME|RDW_INVALIDATE|RDW_ALLCHILDREN);
    return hr;
}

```

Veá también

- [Compatibilidad con vistas de varios documentos](#)
- [Anexión de vistas a datos de documentos](#)
- [Creación de diseñadores y editores personalizados](#)

Cómo: Abrir editores específicos de proyectos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Si un archivo de elemento abierto por un proyecto está intrínsecamente enlazado al editor concreto para ese proyecto, el proyecto debe abrir el archivo mediante un editor específico del proyecto. El archivo no se puede delegar en el mecanismo del IDE para seleccionar un editor. Por ejemplo, en lugar de usar un editor de mapa de bits estándar, puede usar esta opción de editor específica del proyecto para especificar un editor de mapa de bits específico que reconozca la información del archivo que es exclusiva del proyecto.

El IDE [OpenItem](#) llama al método cuando determina que un archivo debe abrirse por un proyecto específico. Para obtener más información, consulte [Mostrar archivos mediante el comando Abrir archivo](#). Use las siguientes directrices para implementar el método para que el `OpenItem` proyecto abra un archivo mediante un editor específico del proyecto.

Implementar el método `OpenItem` con un editor específico del proyecto

1. Llame [FindAndLockDocument](#) al `RDT_EditLock` método () para determinar si el archivo (objeto de datos de documento) ya está abierto.

NOTE

Para obtener más información sobre los datos de documento y los objetos de vista de documento, consulte Datos de documento y vista de [documento en editores personalizados](#).

2. Si el archivo ya está abierto, [IsDocumentOpen](#) vuelva a exponer el archivo `grfIDO` llamando al método y especificando un valor de `IDO_ActivateIfOpen` para el parámetro.

Si el archivo está abierto y el documento es propiedad de un proyecto distinto del proyecto de llamada, se mostrará una advertencia al usuario de que el editor que se está abriendo es de otro proyecto. A continuación, aparece la ventana de archivo.

3. Si el búfer de texto (objeto de datos de documento) ya está abierto y desea adjuntarle otra vista, es responsable de enlazar esa vista. El enfoque recomendado para crear instancias de una vista (objeto de vista de documento) del proyecto es el siguiente:

- a. Llame `QueryService` al [SLocalRegistry](#) servicio para obtener [ILocalRegistry2](#) un puntero a la interfaz.

- b. Llame [CreateInstance](#) al método para crear una instancia de la clase de vista de documento.

4. Llame [CreateDocumentWindow](#) al método, especificando el objeto de vista de documento.

Este método emplaza el objeto de vista de documento en una ventana de documento.

5. Realice las llamadas [InitNew](#) adecuadas [Load](#) a los métodos o a los métodos.

En este punto, la vista debe inicializarse completamente y estar lista para abrirse.

6. Llame [Show](#) al método para mostrar y abrir la vista.

Vea también

- [Abrir y guardar elementos de proyecto](#)
- [Cómo: Abrir editores estándar](#)
- [Cómo: Abrir editores para documentos abiertos](#)

Cómo: Abrir editores estándar

06/04/2020 • 3 minutes to read • [Edit Online](#)

Al abrir un editor estándar, permite que el IDE determine un editor estándar para un tipo de archivo designado, en lugar de especificar un editor específico del proyecto para el archivo.

Complete el siguiente procedimiento [OpenItem](#) para implementar el método. Esto abrirá un archivo de proyecto en un editor estándar.

Para implementar el método OpenItem con un editor estándar

1. Llame [IVsRunningDocumentTable](#) `RDT_EditLock` a () para determinar si el archivo de objeto de datos de documento ya está abierto.
2. Si el archivo ya está abierto, [IsDocumentOpen](#) vuelva a exponer el `IDO_ActivateIfOpen` archivo `grfIDO` llamando al método, especificando un valor para el parámetro.

Si el archivo está abierto y el documento es propiedad de un proyecto diferente del proyecto de llamada, el proyecto recibe una advertencia de que el editor que se está abriendo es de otro proyecto. A continuación, aparece la ventana de archivo.

3. Si el documento no está abierto o no [OpenStandardEditor](#) está `OSE_ChooseBestStdEditor` en la tabla de documentos en ejecución, llame al método () para abrir un editor estándar para el archivo.

Cuando se llama al método, el IDE realiza las siguientes tareas:

- a. En el Registro, el IDE examina la subclave Editors/-guidEditorType/Extensions para determinar qué editor puede abrir el archivo y tiene la prioridad más alta para hacerlo.
 - b. Después de que el IDE haya determinado qué [CreateEditorInstance](#) editor puede abrir el archivo, el IDE llama . La implementación del editor de este método devuelve información [CreateDocumentWindow](#) necesaria para que el IDE llame y sitio el documento recién abierto.
 - c. Por último, el IDE carga el documento mediante [IVsPersistDocData2](#) la interfaz de persistencia habitual, como .
 - d. Si el IDE ha determinado previamente que la jerarquía [GetItemContext](#) o el elemento de jerarquía está [IServiceProvider](#) disponible, el IDE llama al método en el proyecto para obtener un puntero de contexto de nivel de proyecto para volver a pasar con la [CreateDocumentWindow](#) llamada al método.
4. Devuelve [IServiceProvider](#) un puntero al IDE cuando [GetItemContext](#) el IDE llama al proyecto si desea permitir que el editor obtenga contexto del proyecto.

Al realizar este paso, el proyecto permite ofrecer servicios adicionales al editor.

Si la vista de documento o el objeto de vista de documento se ha [LoadDocData](#) ubicado correctamente en un marco de ventana, el objeto se inicializa con sus datos llamando a .

Vea también

- [IServiceProvider](#)
- [Abrir y guardar elementos del proyecto](#)
- [Cómo: Abrir editores específicos de proyectos](#)

- [Cómo: Abrir editores para documentos abiertos](#)
- [Mostrar archivos mediante el comando Abrir archivo](#)

Cómo: Abrir editores para documentos abiertos

06/04/2020 • 4 minutes to read • [Edit Online](#)

Antes de que un proyecto abra una ventana de documento, el proyecto primero debe determinar si el archivo ya está abierto en la ventana de documento para otro editor. El archivo puede estar abierto en un editor específico del proyecto Visual Studio en uno de los editores estándar registrados con .

Abra un editor específico del proyecto

Use el siguiente procedimiento para abrir un editor específico del proyecto para un archivo que ya está abierto.

Para abrir un editor específico del proyecto para un archivo abierto

1. Llame al método [IsDocumentOpen](#) .

Esta llamada devuelve punteros a la jerarquía del documento, el elemento de jerarquía y el marco de ventana, si procede.

2. Si el documento está abierto, el proyecto debe comprobar si solo existe un objeto de datos de documento o si también hay un objeto de vista de documento.

- Si existe un objeto de vista de documento y esta vista es para una jerarquía o elemento de jerarquía diferente, el proyecto utiliza el puntero al marco de ventana de la vista para resurgir la ventana existente.
- Si existe un objeto de vista de documento y esta vista es para el mismo elemento de jerarquía y jerarquía, el proyecto puede abrir una segunda vista si se puede asociar al objeto de datos de documento subyacente. De lo contrario, el proyecto debe usar el puntero al marco de ventana de la vista para resurgir la ventana existente.
- Si solo existe el objeto de datos del documento, el proyecto debe determinar si puede utilizar el objeto de datos de documento para su vista. Si el objeto de datos del documento es compatible, complete los pasos descritos en [Abrir un editor específico del proyecto](#).

Si el objeto de datos del documento no es compatible, se debe mostrar un error al usuario que indique que el archivo está actualmente en uso. Este error solo debe mostrarse en casos transitorios, como cuando se compila un archivo al mismo tiempo que Visual Studio el usuario intenta abrir el archivo mediante un editor que no sea el editor de texto principal. El editor de texto principal puede compartir el objeto de datos de documento con el compilador.

3. Si el documento no está abierto porque no hay ningún objeto de datos de documento o objeto de vista de documento, complete los pasos descritos en [Abrir un editor específico del proyecto](#).

Abrir un editor estándar

Utilice el siguiente procedimiento para abrir un editor estándar para un archivo que ya está abierto.

Para abrir un editor estándar para un archivo abierto

1. Llame a [OpenStandardEditor](#).

Este método comprueba primero que el documento [IsDocumentOpen](#) aún no está abierto llamando a . Si el documento ya está abierto, se vuelve a aparecer su ventana del editor.

2. Si el documento no está abierto, complete los pasos descritos en [Cómo: Abrir editores estándar](#).

Vea también

- [Abrir y guardar elementos de proyecto](#)
- [Cómo: Abrir editores específicos de proyectos](#)
- [Cómo: Abrir editores estándar](#)

Ampliar proyectos

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los proyectos y soluciones son las formas en que Visual Studio organiza los archivos de código y recursos en unidades de compilación e implementación. Puede encontrar más información sobre los proyectos en [Proyectos \(Visual Studio SDK\)](#).

Puede crear sus propios tipos de proyecto con el SDK de Visual Studio y Managed Package Framework for Projects, que puede descargar en [Managed Package Framework for Projects](#). Para entender cómo se implementan los proyectos personalizados, consulte Nueva generación de [proyectos: Bajo el capó, primera parte](#) y Nueva generación de [proyectos: Bajo el capó, segunda parte](#).

En los temas de esta sección se describe cómo crear proyectos personalizados y cómo administrar diferentes tipos de solución de Visual Studio.

En esta sección

- [Crear un sistema básico de proyectos, parte 1](#) Describe cómo crear un sistema de proyecto personalizado.
- [Crear un sistema básico de proyectos, parte 2](#) Describe cómo crear un sistema de proyecto personalizado.
- [Guardar datos en archivos](#) de proyecto Explica cómo agregar al proyecto (.proj*).
- [Verificar subtipos de un proyecto en tiempo](#) de ejecución Explica cómo comprobar el subtipo de un proyecto en tiempo de ejecución.
- [Agregar y eliminar páginas de propiedades](#) Explica cómo personalizar las páginas de propiedades para el proyecto personalizado.
- [Agregar un atributo a un elemento](#) de proyecto Explica cómo agregar un atributo a un elemento de proyecto personalizado.
- [Persistir la propiedad de un elemento de proyecto](#) Explica cómo conservar las propiedades de un elemento de proyecto personalizado.
- [Administrar proyectos universales de Windows](#) Explica cómo administrar proyectos universales.

Crear un sistema básico de proyectos, parte 1

06/04/2020 • 29 minutes to read • [Edit Online](#)

En Visual Studio, los proyectos son los contenedores que los desarrolladores usan para organizar archivos de código fuente y otros activos. Los proyectos aparecen como elementos secundarios de soluciones en el Explorador de **soluciones**. Los proyectos le permiten organizar, compilar, depurar e implementar código fuente y crear referencias a servicios web, bases de datos y otros recursos.

Los proyectos se definen en archivos de proyecto, por ejemplo, un archivo `.csproj` para un proyecto de Visual C#. Puede crear su propio tipo de proyecto que tenga su propia extensión de nombre de archivo de proyecto. Para obtener más información acerca de los tipos de proyecto, vea [Tipos de proyecto](#).

NOTE

Si necesita ampliar Visual Studio con un tipo de proyecto personalizado, se recomienda aprovechar el sistema de proyectos de [Visual Studio](#) (VSPS) que tiene una serie de ventajas sobre la creación de un sistema de proyecto desde cero:

- Incorporación más fácil. Incluso un sistema de proyecto básico requiere decenas de miles de líneas de código. Aprovechar VSPS reduce el costo de incorporación a unos pocos clics antes de estar listo para personalizarlo según sus necesidades.
- Mantenimiento más fácil. Al aprovechar VSPS, solo necesita mantener sus propios escenarios. Nos encargamos del mantenimiento de toda la infraestructura del sistema del proyecto.

Si necesita tener como destino versiones de Visual Studio anteriores a Visual Studio 2013, no podrá aprovechar VSPS en una extensión de Visual Studio. Si ese es el caso, este tutorial es un buen lugar para empezar.

En este tutorial se muestra cómo crear un tipo de proyecto que tiene la extensión de nombre de archivo de proyecto `.myproj`. En este tutorial se toma prestado del sistema de proyectos de Visual C# existente.

NOTE

Para obtener más ejemplos de proyectos de extensión, vea [ejemplos de VSSDK](#).

Este tutorial enseña cómo realizar estas tareas:

- Cree un tipo de proyecto básico.
- Cree una plantilla de proyecto básica.
- Registre la plantilla de proyecto con Visual Studio.
- Cree una instancia de proyecto abriendo el cuadro de diálogo **Nuevo proyecto** y, a continuación, utilizando la plantilla.
- Cree un generador de proyectos para el sistema de proyectos.
- Cree un nodo de proyecto para el sistema de proyectos.
- Agregue iconos personalizados para el sistema de proyectos.
- Implemente la sustitución básica de parámetros de plantilla.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

También debe descargar el código fuente para [Managed Package Framework para proyectos](#). Extraiga el archivo en una ubicación accesible para la solución que va a crear.

Crear un tipo de proyecto básico

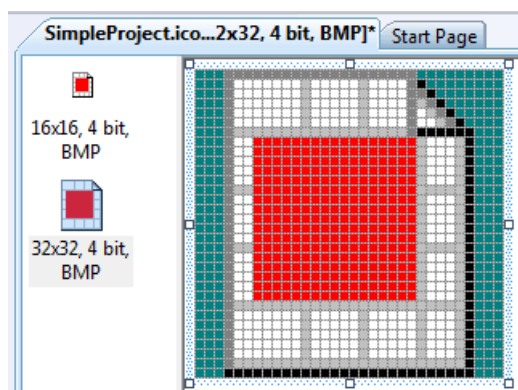
Cree un proyecto de VSIX de C- denominado **SimpleProject**. (**Archivo > nuevo > proyecto** y, a continuación, **proyecto VSIX de extensibilidad > de Visual C. >**). Agregue una plantilla de elemento de proyecto paquete de Visual Studio (en el Explorador de **soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento** y, a continuación, vaya a **Paquete de VisualStudio de extensibilidad >**). Asigne al archivo el nombre *SimpleProjectPackage*.

Creación de una plantilla de proyecto básica

Ahora, puede modificar este VSPackage básico para implementar el nuevo tipo de proyecto *.myproj*. Para crear un proyecto basado en el tipo de proyecto *.myproj*, Visual Studio tiene que saber qué archivos, recursos y referencias agregar al nuevo proyecto. Para proporcionar esta información, coloque los archivos de proyecto en una carpeta de plantilla de proyecto. Cuando un usuario utiliza el proyecto *.myproj* para crear un proyecto, los archivos se copian en el nuevo proyecto.

Para crear una plantilla de proyecto básica

1. Agregue tres carpetas al proyecto, una debajo de la otra: *Plantillas*, *Proyectos*, *SimpleProject*. (En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto **SimpleProject**, seleccione **Agregar**, a continuación, haga clic en **Nueva carpeta**. Asigne a la carpeta el nombre *Plantillas*. En la carpeta *Plantillas*, agregue una carpeta denominada *Proyectos*. En la carpeta *Proyectos*, agregue una carpeta denominada *SimpleProject*.)
2. En la carpeta *Templates-Projects-SimpleProject*, agregue un archivo de imagen de mapa de bits para usarlo como icono denominado *SimpleProject.ico*. Al hacer clic en **Agregar**, se abre el editor de iconos.
3. Haga que el icono sea distintivo. Este icono aparecerá en el cuadro de diálogo **Nuevo proyecto** más adelante en el tutorial.



4. Guarde el icono y cierre el editor de iconos.
5. En la carpeta *Plantillas*, *Proyectos*, *SimpleProject*, agregue un elemento **Clase** denominado *Program.cs*.
6. Reemplace el código existente por las siguientes líneas.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace $nameSpace$
{
    public class $className$
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello VSX!!!");
            Console.ReadKey();
        }
    }
}
```

IMPORTANT

Esta no es la forma final del código *Program.cs*; los parámetros de sustitución se tratarán en un paso posterior. Es posible que vea errores de compilación, pero siempre que **BuildAction** del archivo sea **Content**, debería poder compilar y ejecutar el proyecto como de costumbre.

7. Guarde el archivo.
8. Copie el archivo *de AssemblyInfo.cs* de la carpeta *Propiedades* a la carpeta *Proyectos-SimpleProject*.
9. En la carpeta *Projects-SimpleProject*, agregue un archivo XML denominado *SimpleProject.myproj*.

NOTE

La extensión de nombre de archivo para todos los proyectos de este tipo es *.myproj*. Si desea cambiarlo, debe cambiarlo en todas partes que se menciona en el tutorial.

10. Reemplace el contenido existente por las siguientes líneas.

```

<?xml version="1.0" encoding="utf-8" ?>
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid></ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>MyRootNamespace</RootNamespace>
    <AssemblyName>MyAssemblyName</AssemblyName>
    <EnableUnmanagedDebugging>>false</EnableUnmanagedDebugging>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Debug' ">
    <DebugSymbols>>true</DebugSymbols>
    <OutputPath>bin\Debug\</OutputPath>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Release' ">
    <DebugSymbols>>false</DebugSymbols>
    <OutputPath>bin\Release\</OutputPath>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="mscorlib" />
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="AssemblyInfo.cs">
      <SubType>Code</SubType>
    </Compile>
    <Compile Include="Program.cs">
      <SubType>Code</SubType>
    </Compile>
  </ItemGroup>
  <Import Project="$(MSBuildBinPath)\Microsoft.CSharp.targets" />
</Project>

```

11. Guarde el archivo.

12. En la ventana **Propiedades** , establezca la acción de **compilación** de *AssemblyInfo.cs*, *Program.cs*, *SimpleProject.ico* y *SimpleProject.myproj* en **Contenty** establezca sus propiedades **Include in VSIX** en **True**.

Esta plantilla de proyecto describe un proyecto básico de Visual C. que tiene una configuración de depuración y una configuración de versión. El proyecto incluye dos archivos de origen, *AssemblyInfo.cs* y *Program.cs*, y varias referencias de ensamblado. Cuando se crea un proyecto a partir de la plantilla, el Valor **ProjectGuid** se reemplaza automáticamente por un nuevo GUID.

En el Explorador de **soluciones**, la carpeta **Plantillas** expandidas debe aparecer de la siguiente manera:

```

Templates
  Projects
    SimpleProject
      AssemblyInfo.cs
      Program.cs
      SimpleProject.ico
      SimpleProject.myproj

```

Crear una fábrica de proyectos básica

Debe indicar a Visual Studio la ubicación de la carpeta de plantilla de proyecto. Para ello, agregue un atributo a la clase **VSPackage** que implementa el generador de proyectos para que la ubicación de la plantilla se escriba en el

registro del sistema cuando se compila el VSPackage. Comience por crear un generador de proyectos básico que se identifica mediante un GUID de generador de proyectos. Utilice [ProvideProjectFactoryAttribute](#) el atributo para conectar `SimpleProjectPackage` el generador de proyectos a la clase.

Para crear una fábrica de proyectos básica

1. Cree GUID para el generador de proyectos (en el menú **Herramientas**, haga clic en **Crear GUID**) o use el del ejemplo siguiente. Agregue los GUID a `SimpleProjectPackage` la clase cerca de `PackageGuidString` la sección con el ya definido. Los GUID deben estar en forma GUID y forma de cadena. El código resultante debe ser similar al ejemplo siguiente.

```
public sealed class SimpleProjectPackage : Package
{
    ...
    public const string SimpleProjectPkgString = "96bf4c26-d94e-43bf-a56a-f850b52bfad";
    public const string SimpleProjectFactoryString = "471EC4BB-E47E-4229-A789-D1F5F83B52D4";

    public static readonly Guid guidSimpleProjectFactory = new Guid(SimpleProjectFactoryString);
}
```

2. Agregue una clase a la carpeta *SimpleProject* superior denominada *SimpleProjectFactory.cs*.
3. Agregue lo siguiente mediante directivas:

```
using System.Runtime.InteropServices;
using Microsoft.VisualStudio.Shell;
```

4. Agregue un atributo `SimpleProjectFactory` GUID a la clase. El valor del atributo es el nuevo GUID de generador de proyectos.

```
[Guid(SimpleProjectPackage.SimpleProjectFactoryString)]
class SimpleProjectFactory
{
}
```

Ahora puede registrar su plantilla de proyecto.

Para registrar la plantilla de proyecto

1. En *SimpleProjectPackage.cs*, [ProvideProjectFactoryAttribute](#) agregue `SimpleProjectPackage` un atributo a la clase, como se indica a continuación.

```
[ProvideProjectFactory(    typeof(SimpleProjectFactory),    "Simple Project",
    "Simple Project Files (*.myproj);*.myproj", "myproj", "myproj",
    @"Templates\Projects\SimpleProject",    LanguageVsTemplate = "SimpleProject")]
[Guid(SimpleProjectPackage.PackageGuidString)]
public sealed class SimpleProjectPackage : Package
```

2. Vuelva a generar la solución y compruebe que se compila sin errores.

La reconstrucción registra la plantilla de proyecto.

Los `defaultProjectExtension` parámetros y `possibleProjectExtensions` se establecen en la extensión de nombre de archivo de proyecto (*.myproj*). El `projectTemplatesDirectory` parámetro se establece en la ruta relativa de la carpeta *Plantillas*. Durante la compilación, esta ruta de acceso se convertirá en una compilación completa y se agregará al registro para registrar el sistema del proyecto.

Pruebe el registro de la plantilla

El registro de plantilla indica a Visual Studio la ubicación de la carpeta de plantilla de proyecto para que Visual Studio pueda mostrar el nombre y el icono de la plantilla en el cuadro de diálogo **Nuevo proyecto**.

Para probar el registro de la plantilla

1. Presione F5 para iniciar la depuración de una instancia experimental de Visual Studio.
2. En la instancia experimental, cree un nuevo proyecto del tipo de proyecto recién creado. En el cuadro de diálogo **Nuevo proyecto**, debería ver **SimpleProject** en **Plantillas instaladas**.

Ahora tiene una fábrica de proyectos que está registrada. Sin embargo, todavía no puede crear un proyecto. El paquete de proyecto y el generador de proyectos trabajan juntos para crear e inicializar un proyecto.

Agregue el código de Managed Package Framework

Implemente la conexión entre el paquete del proyecto y el generador del proyecto.

- Importe los archivos de código fuente para Managed Package Framework.
 1. Descargue el proyecto SimpleProject (en el **Explorador** de soluciones, seleccione el nodo del proyecto y, en el menú contextual, haga clic en **Descargar proyecto**.) y abra el archivo de proyecto en el editor XML.
 2. Agregue los siguientes bloques al archivo <de proyecto (justo encima de los bloques Importar>). Establézalo `ProjectBasePath` en la ubicación del archivo *ProjectBase.files* en el código de Managed Package Framework que acaba de descargar. Es posible que tenga que agregar una barra diagonal inversa al nombre de ruta de acceso. Si no lo hace, es posible que el proyecto no encuentre el código fuente de Managed Package Framework.

```
<PropertyGroup>
  <ProjectBasePath>your path here\</ProjectBasePath>
  <RegisterWithCodebase>true</RegisterWithCodebase>
</PropertyGroup>
<Import Project="$(ProjectBasePath)\ProjectBase.Files" />
```

IMPORTANT

No olvide la barra diagonal inversa al final del camino.

3. Vuelva a cargar el proyecto.
4. Agregue referencias a los siguientes ensamblados:
 - `Microsoft.VisualStudio.Designer.Interfaces` (en * <VSSDK instalar>, VisualStudioIntegration, Common, Assemblies, v2.0*)
 - `WindowsBase`
 - `Microsoft.Build.Tasks.v4.0`

Para inicializar la fábrica del proyecto

1. En el archivo *SimpleProjectPackage.cs*, agregue la siguiente `using` directiva.

```
using Microsoft.VisualStudio.Project;
```

2. Derive `SimpleProjectPackage` la `Microsoft.VisualStudio.Package.ProjectPackage` clase de .

```
public sealed class SimpleProjectPackage : ProjectPackage
```

3. Registre la fábrica del proyecto. Agregue la siguiente `SimpleProjectPackage.Initialize` línea al `base.Initialize` método, justo después de .

```
base.Initialize();  
this.RegisterProjectFactory(new SimpleProjectFactory(this));
```

4. Implemente la `ProductUserContext` propiedad abstracta:

```
public override string ProductUserContext  
{  
    get { return ""; }  
}
```

5. En *SimpleProjectFactory.cs*, `using` agregue la `using` siguiente directiva después de las directivas existentes.

```
using Microsoft.VisualStudio.Project;
```

6. Derive `SimpleProjectFactory` la `ProjectFactory` clase de .

```
class SimpleProjectFactory : ProjectFactory
```

7. Agregue el siguiente método `SimpleProjectFactory` ficticio a la clase. Implementará este método en una sección posterior.

```
protected override ProjectNode CreateProject()  
{  
    return null;  
}
```

8. Agregue el siguiente campo `SimpleProjectFactory` y constructor a la clase. Esta `SimpleProjectPackage` referencia se almacena en caché en un campo privado para que se pueda usar en la configuración de un sitio de proveedor de servicios.

```
private SimpleProjectPackage package;  
  
public SimpleProjectFactory(SimpleProjectPackage package)  
    : base(package)  
{  
    this.package = package;  
}
```

9. Vuelva a generar la solución y compruebe que se compila sin errores.

Pruebe la implementación de la fábrica del proyecto

Compruebe si se llama al constructor de la implementación de fábrica de proyectos.

Para probar la implementación de la fábrica del proyecto

1. En el archivo *SimpleProjectFactory.cs*, establezca un punto de interrupción en la línea siguiente en el constructor.

```
this.package = package;
```

2. Presione F5 para iniciar una instancia experimental de Visual Studio.
3. En la instancia experimental, comience a crear un nuevo proyecto. En el cuadro de diálogo **Nuevo proyecto**, seleccione el tipo de proyecto **SimpleProject** y, a continuación, haga clic en **Aceptar**. La ejecución se detiene en el punto de interrupción.
4. Borre el punto de interrupción y detenga la depuración. Puesto que aún no hemos creado un nodo de proyecto, el código de creación del proyecto sigue generando excepciones.

Extender la clase ProjectNode

Ahora puede implementar `SimpleProjectNode` la clase, que `ProjectNode` deriva de la clase. La `ProjectNode` clase base controla las siguientes tareas de creación de proyectos:

- Copia el archivo de plantilla de proyecto, *SimpleProject.myproj*, en la nueva carpeta del proyecto. Se cambia el nombre de la copia según el nombre que se introduce en el cuadro de diálogo **Nuevo proyecto**. El `ProjectGuid` valor de la propiedad se reemplaza por un nuevo GUID.
- Recorre los elementos de MSBuild del archivo de plantilla de `Compile` proyecto, *SimpleProject.myproj*, y busca elementos. Para `Compile` cada archivo de destino, copia el archivo en la nueva carpeta del proyecto.

La clase `SimpleProjectNode` derivada controla estas tareas:

- Habilita los iconos de los nodos de proyecto y archivo en el Explorador de **soluciones** que se van a crear o seleccionar.
- Permite especificar sustituciones de parámetros de plantilla de proyecto adicionales.

Para extender la clase ProjectNode

1. Agregue una clase denominada `SimpleProjectNode.cs`.
2. Reemplace el código existente por el siguiente código.


```

using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.Project;

namespace SimpleProject
{
    public class SimpleProjectNode : ProjectNode
    {
        private SimpleProjectPackage package;

        public SimpleProjectNode(SimpleProjectPackage package)
        {
            this.package = package;
        }
        public override Guid ProjectGuid
        {
            get { return SimpleProjectPackage.guidSimpleProjectFactory; }
        }
        public override string ProjectType
        {
            get { return "SimpleProjectType"; }
        }

        public override void AddFileFromTemplate(
            string source, string target)
        {
            this.FileTemplateProcessor.UntokenFile(source, target);
            this.FileTemplateProcessor.Reset();
        }
    }
}

```

Esta `SimpleProjectNode` implementación de clase ha invalidado estos métodos:

- `ProjectGuid`, que devuelve el GUID del generador de proyectos.
- `ProjectType`, que devuelve el nombre localizado del tipo de proyecto.
- `AddFileFromTemplate`, que copia los archivos seleccionados de la carpeta de plantilla en el proyecto de destino. Este método se implementa en una sección posterior.

El `SimpleProjectNode` constructor, `SimpleProjectFactory` al igual que `SimpleProjectPackage` el constructor, almacena en caché una referencia en un campo privado para su uso posterior.

Para conectar `SimpleProjectFactory` la `SimpleProjectNode` clase a la clase, debe crear una instancia de un nuevo `SimpleProjectNode` en el `SimpleProjectFactory.CreateProject` método y almacenarla en caché en un campo privado para su uso posterior.

Para conectar la clase de generador de proyectos y la clase de nodo

1. En el archivo *SimpleProjectFactory.cs*, agregue la siguiente `using` directiva:

```
using IOleServiceProvider = Microsoft.VisualStudio.OLE.Interop.IServiceProvider;
```

2. Reemplace `SimpleProjectFactory.CreateProject` el método mediante el código siguiente.

```
protected override ProjectNode CreateProject()
{
    SimpleProjectNode project = new SimpleProjectNode(this.package);

    project.SetSite((IOleServiceProvider) ((IServiceProvider)this.package).GetService(
        typeof(IOleServiceProvider)));
    return project;
}
```

3. Vuelva a generar la solución y compruebe que se compila sin errores.

Pruebe la clase ProjectNode

Pruebe el generador de proyectos para ver si crea una jerarquía de proyectos.

Para probar la clase ProjectNode

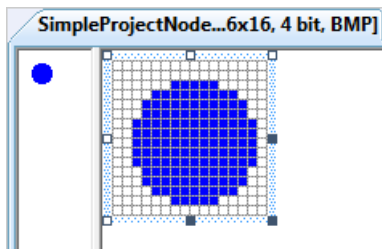
1. Pulse **F5** para iniciar la depuración. En la instancia experimental, cree un nuevo SimpleProject.
2. Visual Studio debe llamar a la fábrica de proyectos para crear un proyecto.
3. Cierre la instancia experimental de Visual Studio.

Agregar un icono de nodo de proyecto personalizado

El icono de nodo de proyecto en la sección anterior es un icono predeterminado. Puede cambiarlo a un icono personalizado.

Para agregar un icono de nodo de proyecto personalizado

1. En la carpeta **Resources**, agregue un archivo de mapa de bits denominado *SimpleProjectNode.bmp*.
2. En las **ventanas Propiedades**, reduzca el mapa de bits a 16 por 16 píxeles. Haga que el mapa de bits sea distintivo.



3. En la ventana **Propiedades**, cambie la acción **Generar** del mapa de bits a **Recurso incrustado**.
4. En *SimpleProjectNode.cs*, `using` agregue las siguientes directivas:

```
using System.Drawing;
using System.Windows.Forms;
```

5. Agregue el siguiente campo estático `SimpleProjectNode` y constructor a la clase.

```
private static ImageList imageList;

static SimpleProjectNode()
{
    imageList = Utilities.GetImageList(
        typeof(SimpleProjectNode).Assembly.GetManifestResourceStream(
            "SimpleProject.Resources.SimpleProjectNode.bmp"));
}
```

6. Agregue la siguiente propiedad al `SimpleProjectNode` principio de la clase.

```
internal static int imageIndex;
public override int ImageIndex
{
    get { return imageIndex; }
}
```

7. Reemplace el constructor de instancia por el código siguiente.

```
public SimpleProjectNode(SimpleProjectPackage package)
{
    this.package = package;

    imageIndex = this.ImageHandler.ImageList.Images.Count;

    foreach (Image img in imageList.Images)
    {
        this.ImageHandler.AddImage(img);
    }
}
```

Durante la `SimpleProjectNode` construcción estática, recupera el mapa de bits del nodo del proyecto de los recursos del manifiesto del ensamblado y lo almacena en caché en un campo privado para su uso posterior. Observe la sintaxis de la ruta de acceso de la `GetManifestResourceStream` imagen. Para ver los nombres de los recursos de `GetManifestResourceNames` manifiesto incrustados en un ensamblado, utilice el método. Cuando este método se `SimpleProject` aplica al ensamblado, los resultados deben ser los siguientes:

- *SimpleProject.Resources.resources*
- *VisualStudio.Project.resources*
- *SimpleProject.VSPackage.resources*
- *Resources.imagelis.bmp*
- *Microsoft.VisualStudio.Project.DontShowAgainDialog.resources*
- *Microsoft.VisualStudio.Project.SecurityWarningDialog.resources*
- *SimpleProject.Resources.SimpleProjectNode.bmp*

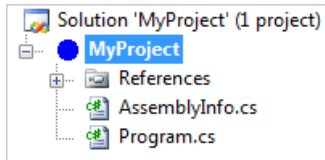
Durante la construcción `ProjectNode` de instancias, la clase base carga *Resources.imagelis.bmp*, en el que se incrustan comúnmente 16 x 16 mapas de bits de *Resources-imagelis.bmp*. Esta lista de mapas `SimpleProjectNode` `ImageHandler.ImageList` de bits está disponible para como . `SimpleProjectNode` anexa el mapa de bits del nodo del proyecto a la lista. El desplazamiento del mapa de bits del nodo de proyecto en la `ImageIndex` lista de imágenes se almacena en caché para su uso posterior como valor de la propiedad pública. Visual Studio usa esta propiedad para determinar qué mapa de bits se mostrará como el icono del nodo del proyecto.

Pruebe el icono del nodo de proyecto personalizado

Pruebe el generador de proyectos para ver si crea una jerarquía de proyectos que tiene el icono de nodo de proyecto personalizado.

Para probar el icono de nodo de proyecto personalizado

1. Inicie la depuración y, en la instancia experimental, cree un nuevo SimpleProject.
2. En el proyecto recién creado, observe que *SimpleProjectNode.bmp* se utiliza como el icono de nodo del proyecto.



3. Abra el archivo *Program.cs* en el editor de código. Debería ver el código fuente similar al código siguiente.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace $nameSpace$
{
    public class $className$
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello VSX!!!");
            Console.ReadKey();
        }
    }
}
```

Observe que los parámetros de plantilla `$nameSpace$` y `$className$` no tienen nuevos valores. Aprenderá a implementar la sustitución de parámetros de plantilla en la siguiente sección.

Parámetros de plantilla sustitutos

En una sección anterior, registró la plantilla de `ProvideProjectFactory` proyecto con Visual Studio mediante el atributo. Registrar la ruta de acceso de una carpeta de plantilla de esta manera `ProjectNode.AddFileFromTemplate` le permite habilitar la sustitución de parámetros de plantilla básico reemplazando y expandiendo la clase. Para obtener más información, consulte Nueva generación de [proyectos: Bajo el capó, segunda parte](#).

Ahora agregue código `AddFileFromTemplate` de reemplazo a la clase.

Para sustituir los parámetros de plantilla

1. En el archivo *SimpleProjectNode.cs*, agregue la siguiente `using` directiva.

```
using System.IO;
```

2. Reemplace `AddFileFromTemplate` el método mediante el código siguiente.

```

public override void AddFileFromTemplate(
    string source, string target)
{
    string nameSpace =
        this.FileTemplateProcessor.GetFileNamespace(target, this);
    string className = Path.GetFileNameWithoutExtension(target);

    this.FileTemplateProcessor.AddReplace($"$nameSpace$", nameSpace);
    this.FileTemplateProcessor.AddReplace($"$className$", className);

    this.FileTemplateProcessor.UntokenFile(source, target);
    this.FileTemplateProcessor.Reset();
}

```

3. Establezca un punto de interrupción `className` en el método, justo después de la instrucción de asignación.

Las instrucciones de asignación determinan valores razonables para un espacio de nombres y un nuevo nombre de clase. Las `ProjectNode.FileTemplateProcessor.AddReplace` dos llamadas al método reemplazan los valores de parámetro de plantilla correspondientes mediante estos nuevos valores.

Pruebe la sustitución del parámetro de plantilla

Ahora puede probar la sustitución de parámetros de plantilla.

Para probar la sustitución de parámetros de plantilla

1. Inicie la depuración y, en la instancia experimental, cree un nuevo SimpleProject.
2. La ejecución se detiene `AddFileFromTemplate` en el punto de interrupción del método.
3. Examine los valores `nameSpace` `className` de los parámetros y.
 - `nameSpace` se le da <el valor del elemento de> De_Arracensel raíz en el archivo de plantilla de proyecto . En este caso, el valor es `MyRootNamespace` .
 - `className` recibe el valor del nombre de archivo de origen de clase, sin la extensión de nombre de archivo. En este caso, el primer archivo que se copia en la carpeta de destino se *AssemblyInfo.cs*; por lo tanto, el `AssemblyInfo` valor de `className` es .
4. Quite el punto de interrupción y presione **F5** para continuar la ejecución.
Visual Studio debe terminar de crear un proyecto.
5. Abra el archivo *Program.cs* en el editor de código. Debería ver el código fuente similar al código siguiente.

```

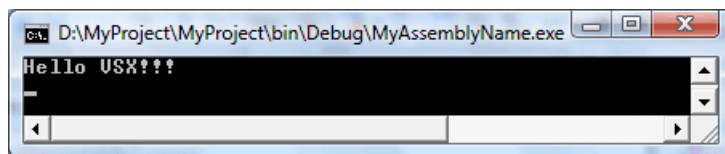
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyRootNamespace
{
    public class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello VSX!!!");
            Console.ReadKey();
        }
    }
}

```

Observe que el `MyRootNamespace` espacio de nombres `Program` es ahora y el nombre de clase es ahora .

6. Inicie la depuración del proyecto. El nuevo proyecto debe compilar, ejecutar y mostrar "Hello VSX!!!" en la ventana de la consola.



¡Enhorabuena! Ha implementado un sistema de proyectos administrado básico.

Crear un sistema básico de proyectos, parte 2

06/04/2020 • 24 minutes to read • [Edit Online](#)

El primer tutorial de esta serie, Crear un sistema de [proyecto básico, parte 1](#), muestra cómo crear un sistema de proyecto básico. Este tutorial se basa en el sistema de proyectos básico agregando una plantilla de Visual Studio, una página de propiedades y otras características. Debe completar el primer tutorial antes de iniciar este.

En este tutorial se enseña cómo crear un tipo de proyecto que tiene la extensión de nombre de archivo de proyecto *.myproj*. Para completar el tutorial, no es necesario crear su propio lenguaje porque el tutorial toma prestado del sistema de proyectos de Visual C.

Este tutorial enseña cómo realizar estas tareas:

- Cree una plantilla de Visual Studio.
- Implementar una plantilla de Visual Studio.
- Cree un nodo secundario de tipo de proyecto en el cuadro de diálogo **Nuevo proyecto**.
- Habilite la sustitución de parámetros en la plantilla de Visual Studio.
- Cree una página de propiedades de proyecto.

NOTE

Los pasos de este tutorial se basan en un proyecto de C. Sin embargo, excepto para detalles como las extensiones de nombre de archivo y el código, puede usar los mismos pasos para un proyecto de Visual Basic.

Crear una plantilla de Visual Studio

- Crear un sistema de [proyecto básico, la parte 1](#) muestra cómo crear una plantilla de proyecto básica y agregarla al sistema de proyectos. También se muestra cómo registrar esta plantilla [ProvideProjectFactoryAttribute](#) con Visual Studio mediante el atributo, que escribe la ruta de acceso completa de la * \ * carpeta Plantillas , Proyectos , SimpleProject en el registro del sistema.

Mediante el uso de una plantilla de Visual Studio (archivo *.vstemplate*) en lugar de una plantilla de proyecto básica, puede controlar cómo aparece la plantilla en el cuadro de diálogo **Nuevo proyecto** y cómo se sustituyen los parámetros de plantilla. Un archivo *.vstemplate* es un archivo XML que describe cómo se deben incluir los archivos de origen cuando se crea un proyecto mediante la plantilla del sistema de proyecto. El propio sistema del proyecto se compila recopilando el archivo *.vstemplate* y los archivos de origen en un archivo *.zip* y se implementa copiando el archivo *.zip* en una ubicación conocida por Visual Studio. Este proceso se explica con más detalle más adelante en este tutorial.

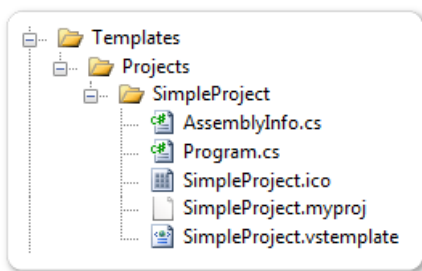
1. En Visual Studio, abra la solución SimpleProject que creó siguiendo Crear un sistema de [proyecto básico, parte 1](#).
2. En el archivo *SimpleProjectPackage.cs*, busque el atributo ProvideProjectFactory. Reemplace el segundo parámetro (el nombre del proyecto) por null y el cuarto parámetro (la ruta de acceso a la carpeta de plantilla de proyecto) por ". \NullPath", como se indica a continuación.

```
[ProvideProjectFactory(typeof(SimpleProjectFactory), null,
    "Simple Project Files (*.myproj);*.myproj", "myproj", "myproj",
    ".\\NullPath",
    LanguageVsTemplate = "SimpleProject")]]
```

3. Agregue un archivo XML denominado *SimpleProject.vstemplate* a la * \carpeta Templates-Projects-SimpleProject\ *
4. Reemplace el contenido de *SimpleProject.vstemplate* por el código siguiente.

```
<VSTemplate Version="2.0.0" Type="Project"
    xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>SimpleProject Application</Name>
    <Description>
      A project for creating a SimpleProject application
    </Description>
    <Icon>SimpleProject.ico</Icon>
    <ProjectType>SimpleProject</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="SimpleProject.myproj" ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true" OpenInEditor="true">
        Program.cs
      </ProjectItem>
      <ProjectItem ReplaceParameters="true" OpenInEditor="false">
        AssemblyInfo.cs
      </ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

5. En la ventana **Propiedades** , seleccione los cinco archivos de la **Build Action** * \carpeta Plantillas, Proyectos y SimpleProject\ * y establezca la Acción de compilación en **ZipProject**.



La <sección> TemplateData determina la ubicación y el aspecto del tipo de proyecto SimpleProject en el cuadro de diálogo **Nuevo proyecto**, como se indica a continuación:

- El <elemento Name> nombra la plantilla de proyecto para que sea SimpleProject Application.
- El <elemento de> Descripción contiene la descripción que aparece en el cuadro de diálogo **Nuevo proyecto** cuando se selecciona la plantilla de proyecto.
- El <elemento Icon> especifica el icono que aparece junto con el tipo de proyecto SimpleProject.
- El <elemento projectType> nombre el tipo de proyecto en el cuadro de diálogo **Nuevo proyecto**. Este nombre reemplaza el parámetro de nombre de proyecto de la ProvideProjectFactory atributo.

NOTE

El <elemento de> `LanguageVsTemplate` `ProjectType` `ProvideProjectFactory` debe coincidir con el argumento del atributo en el archivo `SimpleProjectPackage.cs`.

La <sección `TemplateContent`> describe estos archivos que se generan cuando se crea un nuevo proyecto:

- *SimpleProject.myproj*
- *Program.cs*
- *AssemblyInfo.cs*

Los tres `ReplaceParameters` archivos se han establecido en `true`, lo que permite la sustitución de parámetros. El *Program.cs* archivo `OpenInEditor` `Program.cs` se ha establecido en `true`, lo que hace que el archivo se abra en el editor de código cuando se crea un proyecto.

Para obtener más información acerca de los elementos del esquema de plantilla de Visual Studio, vea la referencia de esquema de plantilla de [Visual Studio](#).

NOTE

Si un proyecto tiene más de una plantilla de Visual Studio, cada plantilla está en una carpeta independiente. Cada archivo de esa carpeta debe tener la acción de **compilación** establecida en `ZipProject`.

Adición de un archivo .vsct mínimo

Visual Studio debe ejecutarse en modo de instalación para reconocer una plantilla de Visual Studio nueva o modificada. El modo de instalación requiere la presentación de un archivo `.vsct`. Por lo tanto, debe agregar un archivo `.vsct` mínimo al proyecto.

1. Agregue un archivo XML denominado *SimpleProject.vsct* al proyecto `SimpleProject`.
2. Reemplace el contenido del archivo *SimpleProject.vsct* por el código siguiente.

```
<?xml version="1.0" encoding="utf-8" ?>
<CommandTable
  xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable">
</CommandTable>
```

3. Establezca la acción de **compilación** de este archivo en **VSCTCompile**. Solo puede hacerlo en el archivo `.csproj`, no en la ventana **Propiedades**. Asegúrese de que la acción de **compilación** de este archivo está establecida en **Ninguno** en este momento.
 - a. Haga clic con el botón secundario en el nodo `SimpleProject` y, a continuación, haga clic en **Editar SimpleProject.csproj**.
 - b. En el archivo `.csproj`, busque el elemento *SimpleProject.vsct*.

```
<None Include="SimpleProject.vsct" />
```

- c. Cambie la acción de compilación a **VSCTCompile**.

```
<VSCTCompile Include="SimpleProject.vsct" />
```

- d. el archivo de proyecto y cierre el editor.
- e. Guarde el nodo SimpleProject y, a continuación, en el **Explorador** de soluciones, haga clic en **Volver a cargar proyecto**.

Examine los pasos de compilación de la plantilla de Visual Studio

El sistema de compilación de proyectos de VSPackage normalmente ejecuta Visual Studio en modo de instalación cuando se cambia el archivo *.vstemplate* o se vuelve a generar el proyecto que contiene el archivo *.vstemplate*. Puede seguir el nivel de detalle de MSBuild en Normal o superior.

1. En el menú **Herramientas**, haga clic en **Opciones**.
2. Expanda el nodo **Proyectos y soluciones** y, a continuación, seleccione **Compilar y ejecutar**.
3. Establezca la verbosidad de la generación de compilación del proyecto de **MSBuild** en **Normal**. Haga clic en **OK**.
4. Vuelva a generar el proyecto SimpleProject.

El paso de compilación para crear el archivo de proyecto *.zip* debe ser similar al siguiente ejemplo.

```
ZipProjects:
1> Zipping ProjectTemplates
1> Zipping <path>\SimpleProject\SimpleProject\obj\Debug\SimpleProject.zip...
1> Copying file from "<path>\SimpleProject\SimpleProject\obj\Debug\SimpleProject.zip" to "
<%LOCALAPPDATA%\Microsoft\VisualStudio\14.0Exp\ProjectTemplates\\SimpleProject.zip".
1> Copying file from "<path>\SimpleProject\SimpleProject\obj\Debug\SimpleProject.zip" to
"bin\Debug\ProjectTemplates\\SimpleProject.zip".
1> SimpleProject -> <path>\SimpleProject\SimpleProject\bin\Debug\ProjectTemplates\SimpleProject.zip
1> ZipItems:
1> Zipping ItemTemplates
1> SimpleProject ->
```

Implementar una plantilla de Visual Studio

Las plantillas de Visual Studio no contienen información de ruta de acceso. Por lo tanto, el archivo *.zip* de plantilla debe implementarse en una ubicación conocida por Visual Studio. La ubicación de la carpeta ProjectTemplates suele ser *<%LOCALAPPDATA%>.Microsoft-VisualStudio-14.0Exp-ProjectTemplates*.

Para implementar el generador de proyectos, el programa de instalación debe tener privilegios de administrador.

Implementa plantillas en el nodo de instalación de Visual Studio:

Probar una plantilla de Visual Studio

Pruebe el generador de proyectos para ver si crea una jerarquía de proyectos mediante la plantilla de Visual Studio.

1. Restablezca la instancia experimental del SDK de Visual Studio.

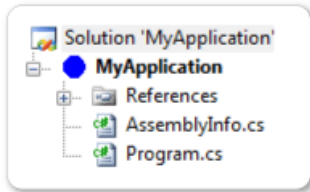
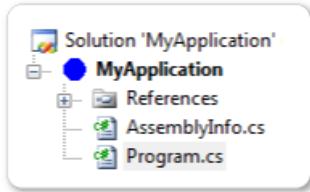
En: Windows 7 en el menú **Inicio**, busque la carpeta **MICROSOFT Visual Studio/Microsoft Visual Studio SDK/Tools** y, a continuación, seleccione **Restablecer la instancia experimental de Microsoft Visual Studio**.

En versiones posteriores de Windows: en la pantalla **Inicio**, escriba **Restablecer la versión <** de Microsoft Visual Studio> Instancia experimental.

2. Aparece una ventana del símbolo del sistema. Cuando vea las palabras **Pulse cualquier tecla para**

continuar, haga clic en **ENTRAR**. Después de cerrar la ventana, abra Visual Studio.

3. Vuelva a generar el proyecto SimpleProject e inicie la depuración. Aparece la instancia experimental.
4. En la instancia experimental, cree un proyecto SimpleProject. En el cuadro de diálogo **Nuevo proyecto**, seleccione **SimpleProject**.
5. Debería ver una nueva instancia de SimpleProject.



Crear un nodo secundario de tipo de proyecto

Puede agregar un nodo secundario a un nodo de tipo de proyecto en el cuadro de diálogo **Nuevo proyecto**. Por ejemplo, para el tipo de proyecto SimpleProject, podría tener nodos secundarios para aplicaciones de consola, aplicaciones de ventana, aplicaciones web, etc.

Los nodos secundarios se crean modificando <el archivo de <proyecto y agregando OutputSubPath> elementos secundarios a los elementos de> ZipProject. Cuando se copia una plantilla durante la compilación o la implementación, cada nodo secundario se convierte en una subcarpeta de la carpeta de plantillas de proyecto.

En esta sección se muestra cómo crear un nodo secundario de consola para el tipo de proyecto SimpleProject.

1. Cambie el nombre de la * \ * carpeta Plantillas, Proyectos, SimpleProject a * \Plantillas, Proyectos, ConsolaApp*.
2. En la ventana **Propiedades**, seleccione los cinco archivos de la **Build Action** * \carpeta Plantillas, Proyectos y \Consolación* y asegúrese de que la acción de compilación está establecida en **ZipProject**.
3. En el archivo SimpleProject.vstemplate, agregue la siguiente <línea al final de la sección TemplateData>, justo antes de la etiqueta de cierre.

```
<NumberOfParentCategoriesToRollUp>1</NumberOfParentCategoriesToRollUp>
```

Esto hace que la plantilla de aplicación de consola aparezca tanto en el nodo secundario de la consola como en el nodo primario SimpleProject, que está un nivel por encima del nodo secundario.

4. Guarde el archivo *SimpleProject.vstemplate*.
5. En el archivo *.csproj* agregue <OutputSubPath> a cada uno de los ZipProject elementos. Descargue el proyecto, como antes, y edite el archivo de proyecto.
6. Busque <los elementos de> ZipProject. A <cada elemento de> <ZipProject, agregue un elemento de> OutputSubPath y asípiase el valor Console. El ZipProject

```
<ZipProject Include="Templates\Projects\ConsoleApp\AssemblyInfo.cs">
  <OutputSubPath>Console</OutputSubPath>
</ZipProject>
<ZipProject Include="Templates\Projects\ConsoleApp\Program.cs">
  <OutputSubPath>Console</OutputSubPath>
</ZipProject>
<ZipProject Include="Templates\Projects\ConsoleApp\SimpleProject.myproj">
  <OutputSubPath>Console</OutputSubPath>
</ZipProject>
<ZipProject Include="Templates\Projects\ConsoleApp\SimpleProject.vstemplate">
  <OutputSubPath>Console</OutputSubPath>
</ZipProject>
<ZipProject Include="Templates\Projects\ConsoleApp\SimpleProject.ico">
  <OutputSubPath>Console</OutputSubPath>
</ZipProject>
```

7. Agregue <este> PropertyGroup al archivo de proyecto:

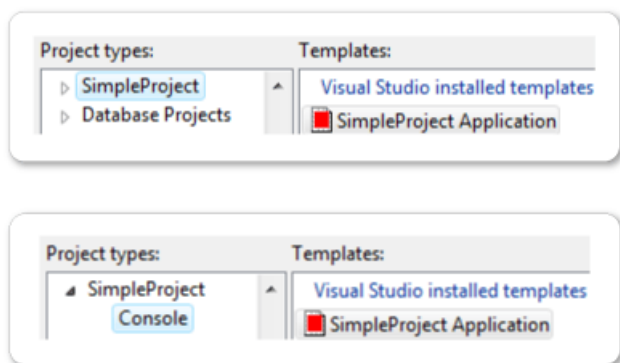
```
<PropertyGroup>
  <VsTemplateLanguage>SimpleProject</VsTemplateLanguage>
</PropertyGroup>
```

8. Guarde el archivo de proyecto y vuelva a cargar el proyecto.

Pruebe el nodo secundario del tipo de proyecto

Pruebe el archivo de proyecto modificado para ver si el nodo secundario **Consola** aparece en el cuadro de diálogo **Nuevo proyecto**.

1. Ejecute la herramienta Restablecer la instancia experimental de **Microsoft Visual Studio**.
2. Vuelva a generar el proyecto SimpleProject e inicie la depuración. La instancia experimental debe aparecer
3. En el cuadro de diálogo **Nuevo proyecto**, haga clic en el nodo **SimpleProject**. La plantilla Aplicación de consola debe aparecer en el panel **Plantillas**.
4. Expanda el nodo **SimpleProject**. Debería aparecer el nodo secundario **Consola**. La plantilla **Aplicación SimpleProject** sigue apareciendo en el panel **Plantillas**.
5. Haga clic en **Cancelar** y detenga la depuración.



Parámetros de plantilla de proyecto sustitutos

- Al crear un sistema de proyecto básico, la parte 1 mostró cómo sobrescribir el `ProjectNode.AddFileFromTemplate` método para realizar un tipo básico de sustitución de parámetros de plantilla. En esta sección se enseña cómo usar los parámetros de plantilla de Visual Studio más sofisticados.

Al crear un proyecto mediante una plantilla de Visual Studio en el cuadro de diálogo **Nuevo proyecto**, los parámetros de plantilla se reemplazan por cadenas para personalizar el proyecto. Un parámetro de plantilla es un token especial que comienza y termina con un signo de dólar, por ejemplo, \$time\$. Los dos parámetros siguientes son especialmente útiles para habilitar la personalización en proyectos basados en la plantilla:

- \$GUID[1-10]\$ es reemplazada por un nuevo Guid. Puede especificar hasta 10 GUID únicos, por ejemplo, \$guid1\$.
- \$safeprojectname\$ es el nombre proporcionado por un usuario en el cuadro de diálogo **Nuevo proyecto**, modificado para quitar todos los caracteres y espacios no seguros.

Para obtener una lista completa de parámetros de plantilla, vea [Parámetros de plantilla](#).

Para sustituir los parámetros de la plantilla de proyecto

1. En el archivo *SimpleProjectNode.cs*, quite el `AddFileFromTemplate` método.
2. En el `< * \archivo Templates-Projects-ConsoleApp-SimpleProject.myproj,*` busque la propiedad `>RootNamespace` y cambie su valor a `$safeprojectname$`.

```
<RootNamespace>$safeprojectname$</RootNamespace>
```

3. En el `* \archivo Templates-Projects-SimpleProject-Program.cs,*` sustituya el contenido del archivo por el código siguiente:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;    // Guid

namespace $safeprojectname$
{
    [Guid("$guid1$")]
    public class $safeprojectname$
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello VSX!!!");
            Console.ReadKey();
        }
    }
}
```

4. Vuelva a generar el proyecto SimpleProject e inicie la depuración. Debería aparecer la instancia experimental.
5. Cree una nueva aplicación De simpleProject Console. (En el panel Tipos de **proyecto**, seleccione **SimpleProject**. En **Plantillas instaladas** de Visual Studio , seleccione Aplicación de **consola**.)
6. En el proyecto recién creado, abra *Program.cs*. Debe tener un aspecto similar al siguiente (los valores GUID del archivo diferirán):

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;    // Guid

namespace Console_Application1
{
    [Guid("00000000-0000-0000-00000000-00000000")]
    public class Console_Application1
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello VSX!!!");
            Console.ReadKey();
        }
    }
}

```

Crear una página de propiedades de proyecto

Puede crear una página de propiedades para el tipo de proyecto para que los usuarios puedan ver y cambiar las propiedades de los proyectos que se basan en la plantilla. En esta sección se muestra cómo crear una página de propiedades independiente de la configuración. Esta página de propiedades básica utiliza una cuadrícula de propiedades para mostrar las propiedades públicas que expone en la clase de página de propiedades.

Derive la clase de `SettingsPage` página de propiedades de la clase base. La cuadrícula de `SettingsPage` propiedades proporcionada por la clase es consciente de la mayoría de los tipos de datos primitivos y sabe cómo mostrarlos. Además, `SettingsPage` la clase sabe cómo conservar los valores de propiedad en el archivo de proyecto.

La página de propiedades que cree en esta sección le permite modificar y guardar estas propiedades del proyecto:

- AssemblyName
- OutputType
- RootNamespace.

1. En el archivo *SimpleProjectPackage.cs*, agregue este `ProvideObject` atributo a la `SimpleProjectPackage` clase:

```

[ProvideObject(typeof(GeneralPropertyPage))]
public sealed class SimpleProjectPackage : ProjectPackage

```

Esto registra la clase `GeneralPropertyPage` de página de propiedades con COM.

2. En el archivo *SimpleProjectNode.cs*, agregue estos `SimpleProjectNode` dos métodos invalidados a la clase:

```
protected override Guid[] GetConfigurationIndependentPropertyPages()
{
    Guid[] result = new Guid[1];
    result[0] = typeof(GeneralPropertyPage).GUID;
    return result;
}
protected override Guid[] GetPriorityProjectDesignerPages()
{
    Guid[] result = new Guid[1];
    result[0] = typeof(GeneralPropertyPage).GUID;
    return result;
}
```

Ambos métodos devuelven una matriz de GUID de página de propiedades. El GUID de `GeneralPropertyPage` es el único elemento de la matriz, por lo que el cuadro de diálogo **Páginas** de propiedades mostrará solo una página.

3. Agregue un archivo de clase denominado *GeneralPropertyPage.cs* al proyecto SimpleProject.
4. Reemplace el contenido de este archivo utilizando el siguiente código:

```

using System;
using System.Runtime.InteropServices;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Project;
using System.ComponentModel;

namespace SimpleProject
{
    [ComVisible(true)]
    [Guid("6BC7046B-B110-40d8-9F23-34263D8D2936")]
    public class GeneralPropertyPage : SettingsPage
    {
        private string assemblyName;
        private OutputType outputType;
        private string defaultNamespace;

        public GeneralPropertyPage()
        {
            this.Name = "General";
        }

        [Category("AssemblyName")]
        [DisplayName("AssemblyName")]
        [Description("The output file holding assembly metadata.")]
        public string AssemblyName
        {
            get { return this.assemblyName; }
        }

        [Category("Application")]
        [DisplayName("OutputType")]
        [Description("The type of application to build.")]
        public OutputType OutputType
        {
            get { return this.outputType; }
            set { this.outputType = value; this.IsDirty = true; }
        }

        [Category("Application")]
        [DisplayName("DefaultNamespace")]
        [Description("Specifies the default namespace for added items.")]
        public string DefaultNamespace
        {
            get { return this.defaultNamespace; }
            set { this.defaultNamespace = value; this.IsDirty = true; }
        }

        protected override void BindProperties()
        {
            this.assemblyName = this.ProjectMgr.GetProjectProperty("AssemblyName", true);
            this.defaultNamespace = this.ProjectMgr.GetProjectProperty("RootNamespace", false);

            string outputType = this.ProjectMgr.GetProjectProperty("OutputType", false);
            this.outputType = (OutputType)Enum.Parse(typeof(OutputType), outputType);
        }

        protected override int ApplyChanges()
        {
            this.ProjectMgr.SetProjectProperty("AssemblyName", this.assemblyName);
            this.ProjectMgr.SetProjectProperty("OutputType", this.outputType.ToString());
            this.ProjectMgr.SetProjectProperty("RootNamespace", this.defaultNamespace);
            this.IsDirty = false;

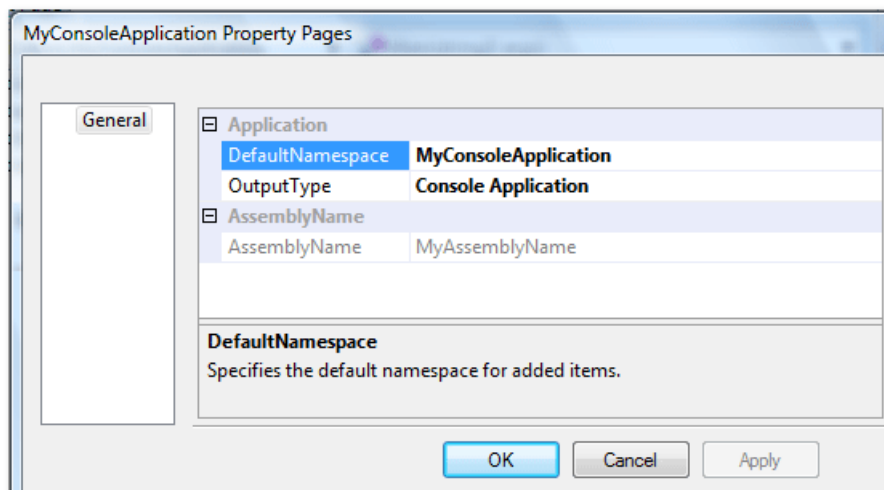
            return VSConstants.S_OK;
        }
    }
}

```


La `GeneralPropertyPage` clase expone las tres propiedades públicas `AssemblyName`, `OutputType` y `RootNamespace`. Dado que `AssemblyName` no tiene ningún método set, se muestra como una propiedad de solo lectura. `OutputType` es una constante enumerada, por lo que aparece como lista desplegable.

La `SettingsPage` clase `ProjectMgr` base proporciona para conservar las propiedades. El `BindProperties` método `ProjectMgr` se utiliza para recuperar los valores de propiedad persistentes y establecer las propiedades correspondientes. El `ApplyChanges` método `ProjectMgr` se utiliza para obtener los valores de las propiedades y conservarlos en el archivo de proyecto. El método de `IsDirty` conjunto de propiedades se establece en true para indicar que las propiedades deben conservarse. La persistencia se produce al guardar el proyecto o la solución.

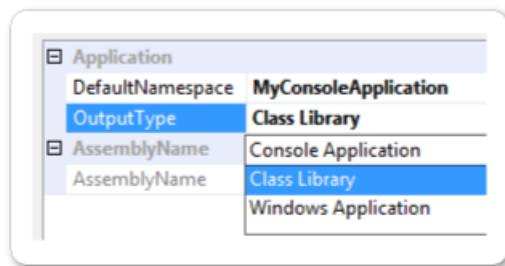
5. Vuelva a generar la solución SimpleProject e inicie la depuración. Debería aparecer la instancia experimental.
6. En la instancia experimental, cree una nueva aplicación SimpleProject.
7. Visual Studio llama al generador de proyectos para crear un proyecto mediante la plantilla de Visual Studio. El nuevo archivo *Program.cs* se abre en el editor de código.
8. Haga clic con el botón secundario en el nodo del proyecto en el **Explorador** de soluciones y, a continuación, haga clic en **Propiedades**. Se muestra el cuadro de diálogo **Páginas de propiedades**.



Pruebe la página de propiedades del proyecto

Ahora puede probar si puede modificar y cambiar los valores de propiedad.

1. En el cuadro de diálogo Páginas de propiedades **MyConsoleApplication**, cambie **DefaultNamespace** a **MyApplication**.
2. Seleccione la propiedad **OutputType** y, a continuación, seleccione **Biblioteca de clases**.
3. Haga clic en **Aplicar** y en **Aceptar**.
4. Vuelva a abrir el cuadro de diálogo **Páginas de propiedades** y compruebe que los cambios se han pertenido.
5. Cierre la instancia experimental de Visual Studio.
6. Vuelva a abrir la instancia experimental.
7. Vuelva a abrir el cuadro de diálogo **Páginas de propiedades** y compruebe que los cambios se han pertenido.



8. Cierre la instancia experimental de Visual Studio.

Guardar datos en archivos de proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un subtipo de proyecto puede guardar y recuperar datos específicos del subtipo en el archivo de proyecto. Managed Package Framework (MPF) proporciona dos interfaces para realizar esta tarea:

- La [IVsBuildPropertyStorage](#) interfaz permite tener acceso a los valores de propiedad desde la sección **MSBuild** del archivo de proyecto. Los métodos [IVsBuildPropertyStorage](#) proporcionados por pueden ser llamados por cualquier usuario cada vez que el usuario necesita cargar o guardar datos relacionados con la compilación.
- Se [IPersistXMLFragment](#) utiliza para conservar datos no relacionados con la compilación en XML de forma libre. Se llama [IPersistXMLFragment](#) a los Visual Studio Visual Studio métodos proporcionados por cada vez que es necesario conservar los datos no relacionados con la compilación en el archivo de proyecto.

Para obtener más información sobre cómo conservar los datos relacionados con la compilación y la no compilación, vea [Persistdata data in the MSBuild project file](#).

Guardar y recuperar datos relacionados con la compilación

Para guardar datos relacionados con la compilación en el archivo de proyecto

- Llame [SetPropertyValue](#) al método para guardar una ruta de acceso completa del archivo de proyecto.

```
private SpecializedProject project;
IVsBuildPropertyStorage projectStorage = (IVsBuildPropertyStorage)project;
string newFullPath = GetNewFullPath();
// Set a full path of the project file.
ErrorHandler.ThrowOnFailure(projectStorage.SetPropertyValue(
    "MSBuildProjectDirectory",
    String.Empty,
    (uint)_PersistStorageType.PST_PROJECT_FILE, newFullPath));
```

Para recuperar datos relacionados con la compilación del archivo de proyecto

- Llame [GetPropertyValue](#) al método para recuperar una ruta de acceso completa del archivo de proyecto.

```
private SpecializedProject project;
IVsBuildPropertyStorage projectStorage = (IVsBuildPropertyStorage)project;
string fullPath;
// Get a full path of the project file.
ErrorHandler.ThrowOnFailure(projectStorage.GetPropertyValue(
    "MSBuildProjectDirectory",
    String.Empty,
    (uint)_PersistStorageType.PST_PROJECT_FILE, out fullPath));
```

Guardar y recuperar datos no relacionados con la compilación

Para guardar datos no relacionados con la compilación en el archivo de proyecto

1. Implemente [IsFragmentDirty](#) el método para determinar si un fragmento XML ha cambiado desde la última vez que se guardó en su archivo actual.

```

public int IsFragmentDirty(uint storage, out int pfDirty)
{
    pfDirty = 0;
    switch (storage)
    {
        case (uint)_PersistStorageType.PST_PROJECT_FILE:
        {
            if (isDirty)
                pfDirty |= 1;
            break;
        }
        case (uint)_PersistStorageType.PST_USER_FILE:
        {
            // We do not store anything in the user file.
            break;
        }
    }

    // Forward the call to inner flavor(s)
    if (pfDirty == 0 && innerCfg != null && this.innerCfg is IPersistXMLFragment)
        return ((IPersistXMLFragment)this.innerCfg).IsFragmentDirty(storage, out pfDirty);

    return VSConstants.S_OK;
}

```

2. Implemente [Save](#) el método para guardar los datos XML en el archivo de proyecto.

```

public int Save(ref Guid guidFlavor, uint storage, out string pbstrXMLFragment, int fClearDirty)
{
    pbstrXMLFragment = null;

    if (IsMyFlavorGuid(ref guidFlavor))
    {
        switch (storage)
        {
            case (uint)_PersistStorageType.PST_PROJECT_FILE:
            {
                // Create XML for our data.
                XmlDocument doc = new XmlDocument();
                XmlNode root = doc.CreateElement(this.GetType().Name);

                XmlNode node = doc.CreateElement(targetsTag);
                node.AppendChild(doc.CreateTextNode(this.TargetsToExecute));
                root.AppendChild(node);

                node = doc.CreateElement(updateTargetsTag);
                node.AppendChild(doc.CreateTextNode(this.UpdateTargetList.ToString()));
                root.AppendChild(node);

                doc.AppendChild(root);
                // Get XML fragment representing our data
                pbstrXMLFragment = doc.InnerXml;

                if (fClearDirty != 0)
                {
                    isDirty = false;
                    break;
                }
            }
            case (uint)_PersistStorageType.PST_USER_FILE:
            {
                // We do not store anything in the user file.
                break;
            }
        }
    }

    // Forward the call to inner flavor(s)
    if (this.innerCfg != null && this.innerCfg is IPersistXMLFragment)
        return ((IPersistXMLFragment)this.innerCfg).Save(ref guidFlavor, storage, out pbstrXMLFragment, fClearDirty);

    return VSConstants.S_OK;
}

```

Para recuperar datos no relacionados con la compilación en el archivo de proyecto

1. Implemente [InitNew](#) el método para inicializar las propiedades de extensión del proyecto y otros datos independientes de la compilación. Se llama a este método si no hay datos de configuración XML presentes en el archivo de proyecto.

```

public int InitNew(ref Guid guidFlavor, uint storage)
{
    //Return, if it is our guid.
    if (IsMyFlavorGuid(ref guidFlavor))
        return VSConstants.S_OK;

    //Forward the call to inner flavor(s).
    if (this.innerCfg != null && this.innerCfg is IPersistXMLFragment)
        return ((IPersistXMLFragment)this.innerCfg).InitNew(ref guidFlavor, storage);

    return VSConstants.S_OK;
}

```

2. Implemente [Load](#) el método para cargar los datos XML desde el archivo de proyecto.

```
public int Load(ref Guid guidFlavor, uint storage, string pszXMLFragment)
{
    if (IsMyFlavorGuid(ref guidFlavor))
    {
        switch (storage)
        {
            case (uint)_PersistStorageType.PST_PROJECT_FILE:
            {
                // Load our data from the XML fragment.
                XmlDocument doc = new XmlDocument();
                XmlNode node = doc.CreateElement(this.GetType().Name);
                node.InnerXml = pszXMLFragment;
                if (node == null
                    || node.FirstChild == null
                    || node.FirstChild.ChildNodes.Count == 0
                    || node.FirstChild.ChildNodes[0].Name != targetsTag)
                {
                    break;
                }
                this.TargetsToExecute = node.FirstChild.ChildNodes[0].InnerText;

                if (node.FirstChild.ChildNodes.Count <= 1
                    || node.FirstChild.ChildNodes[1].Name != updateTargetsTag)
                {
                    break;
                }
                this.UpdateTargetList = bool.Parse(node.FirstChild.ChildNodes[1].InnerText);
                break;
            }
            case (uint)_PersistStorageType.PST_USER_FILE:
            {
                // We do not store anything in the user file.
                break;
            }
        }
    }

    // Forward the call to inner flavor(s)
    if (this.innerCfg != null && this.innerCfg is IPersistXMLFragment)
    {
        return ((IPersistXMLFragment)this.innerCfg).Load(ref guidFlavor, storage, pszXMLFragment);
    }

    return VSConstants.S_OK;
}
```

NOTE

Todos los ejemplos de código proporcionados en este tema son partes de un ejemplo más amplio en [ejemplos VSSDK](#).

Vea también

- [Persistir datos en el archivo de proyecto de MSBuild](#)

Verificar subtipos de un proyecto en tiempo de ejecución

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un VSPackage que depende de un subtipo de proyecto personalizado debe incluir lógica para buscar ese subtipo para que pueda producir un error correctamente si el subtipo no está presente. El siguiente procedimiento muestra cómo comprobar la presencia de un subtipo especificado.

Para verificar la presencia de un subtipo

1. Obtener la jerarquía del proyecto de [IVsHierarchy](#) los objetos de proyecto y solución como un objeto agregando el código siguiente al VSPackage.

```
EnvDTE.DTE dte;
dte = (EnvDTE.DTE)Package.GetService(typeof(EnvDTE.DTE));

EnvDTE.Project project;
project = dte.Solution.Projects.Item(1);

IVsSolution solution;
solution = (IVsSolution)Package.GetService(typeof(SVsSolution));

IVsHierarchy hierarchy;
hierarchy = solution.GetProjectOfUniqueName(project.UniqueName);
```

2. Convierta la jerarquía [IVsAggregatableProjectCorrected](#) en la interfaz.

```
IVsAggregatableProjectCorrected AP;
AP = hierarchy as IVsAggregatableProjectCorrected;
```

3. Obtenga la lista de GUID de tipo [GetAggregateProjectTypeGuids](#) de proyecto invocando el archivo .

```
string projTypeGuids = AP.GetAggregateProjectTypeGuids().ToUpper();
```

4. Compruebe la lista para el GUID del subtipo especificado.

```
// Replace the string "MyGUID" with the GUID of the subtype.
string guidMySubtype = "MyGUID";
if (projTypeGuids.IndexOf(guidMySubtype) > 0)
{
    // The specified subtype is present.
}
```

Vea también

- [Subtipos de proyecto](#)
- [Diseño de subtipos de proyecto](#)
- [Propiedades y métodos extendidos por subtipos de proyecto](#)

Agregar y eliminar páginas de propiedades

06/04/2020 • 5 minutes to read • [Edit Online](#)

El Diseñador de proyectos proporciona una ubicación centralizada para Visual Studio administrar las propiedades, la configuración y los recursos del proyecto en . Aparece como una sola Visual Studio ventana en el entorno de desarrollo integrado (IDE) y contiene una serie de paneles a la derecha a los que se accede a través de las pestañas de la izquierda. Los paneles (a menudo denominados páginas de propiedades) en el Diseñador de proyectos varían según el tipo de proyecto y el idioma. Se puede tener acceso al Diseñador de proyectos con el comando **Propiedades** del menú **Proyecto**.

Un subtipo de proyecto con frecuencia necesita mostrar páginas de propiedades adicionales en el Diseñador de proyectos. Del mismo modo, algunos subtipos de proyecto pueden requerir que se quiten las páginas de propiedades integradas. Para ello, el subtipo de **IVsHierarchy** proyecto debe **GetProperty** implementar la interfaz e invalidar el método. Al reemplazar este método `propId` y usar el parámetro `__VSHPROPID2` que contiene uno de los valores de la enumeración, puede filtrar, agregar o quitar propiedades del proyecto. Por ejemplo, es posible que deba agregar una página a las páginas de propiedades dependientes de la configuración. Para ello, debe filtrar las páginas de propiedades dependientes de la configuración y, a continuación, agregar una nueva página a la lista existente.

Agregar y quitar páginas de propiedades en el Diseñador de proyectos

Quitar una página de propiedades

1. Invalide `GetProperty(uint itemId, int propId, out object property)` el método para filtrar `clsids` las páginas de propiedades y obtener una lista.

```
Protected Overrides int GetProperty(uint itemId, int propId, out object property)
Protected Overrides Function GetProperty(ByVal itemId As UInteger, ByVal propId As Integer, ByRef
[property] As Object) As Integer
    'Use propId to filter configuration-independent property pages.
    Select Case propId
        ....
        Case CInt(Fix(__VSHPROPID2.VSHPROPID_PropertyPagesCLSIDList))
            'Get a semicolon-delimited list of clsids of the configuration-independent property pages
            ErrorHandler.ThrowOnFailure(MyBase.GetProperty(itemId, propId, [property]))
            Dim propertyPagesList As String = ((String)
[property]).ToUpper(CultureInfo.InvariantCulture)
            'Remove the property page here
            ....
        ....
    End Select
    ....
    Return MyBase.GetProperty(itemId, propId, [property])
End Function
```



```
protected override int GetProperty(uint itemId, int propId, out object property)
{
    //Use propId to filter configuration-independent property pages.
    switch (propId)
    {
        . . . .

        case (int) __VSHPROPID2.VSHPROPID_PropertyPagesCLSIDList:
        {
            //Get a semicolon-delimited list of clsids of the configuration-independent property
            pages
            ErrorHandler.ThrowOnFailure(base.GetProperty(itemId, propId, out property));
            string propertyPagesList = ((string)property).ToUpper(CultureInfo.InvariantCulture);
            //Remove the property page here
            . . . .
        }
        . . . .
    }
    . . . .
    return base.GetProperty(itemId, propId, out property);
}
```

2. Quite la página **Eventos** `clsids` de compilación de la lista obtenida.

```
Private buildEventsPageGuid As String = "{1E78F8DB-6C07-4D61-A18F-7514010ABD56}"
Private index As Integer = propertyPagesList.IndexOf(buildEventsPageGuid)
If index <> -1 Then
    ' GUIDs are separated by ';' so if you remove the last GUID, also remove the last ';'
    Dim index2 As Integer = index + buildEventsPageGuid.Length + 1
    If index2 >= propertyPagesList.Length Then
        propertyPagesList = propertyPagesList.Substring(0, index).TrimEnd(";")
    Else
        propertyPagesList = propertyPagesList.Substring(0, index) + propertyPagesList.Substring(index2)
    End If
End If
'New property value
property = propertyPagesList
```

```
string buildEventsPageGuid = "{1E78F8DB-6C07-4D61-A18F-7514010ABD56}";
int index = propertyPagesList.IndexOf(buildEventsPageGuid);
if (index != -1)
{
    // GUIDs are separated by ';' so if you remove the last GUID, also remove the last ';'
    int index2 = index + buildEventsPageGuid.Length + 1;
    if (index2 >= propertyPagesList.Length)
        propertyPagesList = propertyPagesList.Substring(0, index).TrimEnd(';');
    else
        propertyPagesList = propertyPagesList.Substring(0, index) +
propertyPagesList.Substring(index2);
}
//New property value
property = propertyPagesList;
```

Añadir una página de propiedades

1. Cree una página de propiedades que desee agregar.

```

Class DeployPropertyPage
    Inherits Form
    Implements Microsoft.VisualStudio.OLE.Interop.IPropertyPage
    'Summary: Return a stucture describing your property page.
    ....
    Public Sub GetPageInfo(ByVal pPageInfo As Microsoft.VisualStudio.OLE.Interop.PROPPAGEINFO())
        Dim info As PROPPAGEINFO = New PROPPAGEINFO()
        info.cb = CUInt(Marshal.SizeOf(GetType(PROPPAGEINFO)))
        info.dwHelpContext = 0
        info.pszDocString = Nothing
        info.pszHelpFile = Nothing
        info.pszTitle = "Deployment" 'Assign tab name
        info.SIZE.cx = Me.Size.Width
        info.SIZE.cy = Me.Size.Height
        If Not pPageInfo Is Nothing AndAlso pPageInfo.Length > 0 Then
            pPageInfo(0) = info
        End If
    End Sub
End Class

```

```

class DeployPropertyPage : Form, Microsoft.VisualStudio.OLE.Interop.IPropertyPage
{
    . . . .
    //Summary: Return a stucture describing your property page.
    public void GetPageInfo(Microsoft.VisualStudio.OLE.Interop.PROPPAGEINFO[] pPageInfo)
    {
        PROPPAGEINFO info = new PROPPAGEINFO();
        info.cb = (uint)Marshal.SizeOf(typeof(PROPPAGEINFO));
        info.dwHelpContext = 0;
        info.pszDocString = null;
        info.pszHelpFile = null;
        info.pszTitle = "Deployment"; //Assign tab name
        info.SIZE.cx = this.Size.Width;
        info.SIZE.cy = this.Size.Height;
        if (pPageInfo != null && pPageInfo.Length > 0)
            pPageInfo[0] = info;
    }
}

```

2. Registre su nueva página de propiedades.

```
<MSVSIP.ProvideObject(GetType(DeployPropertyPage), RegisterUsing = RegistrationMethod.CodeBase)>
```

```
[MSVSIP.ProvideObject(typeof(DeployPropertyPage), RegisterUsing = RegistrationMethod.CodeBase)]
```

3. Invalide `GetProperty(uint itemId, int propId, out object property)` el método para filtrar `clsids` páginas de propiedades, obtener una lista y agregar una nueva página de propiedades.

```

Protected Overrides Function GetProperty(ByVal itemId As UInteger, ByVal propId As Integer, ByRef
[property] As Object) As Integer
    'Use propId to filter configuration-dependent property pages.
    Select Case propId
        ....
        case CInt(Fix(__VSHPROPID2.VSHPROPID_CfgPropertyPagesCLSIDList)):
            'Get a semicolon-delimited list of clsids of the configuration-dependent property pages.
            ErrorHandler.ThrowOnFailure(MyBase.GetProperty(itemId, propId, [property]))
            'Add the Deployment property page.
            [property] &= ";"c + GetType(DeployPropertyPage).GUID.ToString("B")
    End Select
    ....
    Return MyBase.GetProperty(itemId, propId, [property])
End Function

```

```

protected override int GetProperty(uint itemId, int propId, out object property)
{
    //Use propId to filter configuration-dependent property pages.
    switch (propId)
    {
        . . . . .
        case (int)__VSHPROPID2.VSHPROPID_CfgPropertyPagesCLSIDList:
            {
                //Get a semicolon-delimited list of clsids of the configuration-dependent property
                pages.
                ErrorHandler.ThrowOnFailure(base.GetProperty(itemId, propId, out property));
                //Add the Deployment property page.
                property += ';' + typeof(DeployPropertyPage).GUID.ToString("B");
            }
        }
        . . . . .
    }
    return base.GetProperty(itemId, propId, out property);
}

```

Vea también

- [Subtipos de proyecto](#)

Agregar un atributo a un elemento de proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los [GetItemAttribute](#) métodos y [SetItemAttribute](#) obtener y establecer el valor de los atributos de un elemento de proyecto. SetItemAttribute crea el atributo si aún no existe, agregándolo a los metadatos del elemento de proyecto.

Agregar un atributo a un elemento de proyecto

- El código siguiente [DTE](#) utiliza el [SetItemAttribute](#) objeto de automatización y el método para agregar un atributo a un elemento de proyecto. El identificador del elemento de proyecto se obtiene del nombre del elemento de proyecto "program.cs". El atributo "MyAttribute" se agrega a este elemento de proyecto y se le da el valor "MyValue".

```
EnvDTE.DTE dte = (EnvDTE.DTE)Package.GetService(typeof(EnvDTE.DTE));
EnvDTE.Project project = dte.Solution.Projects.Item(1);

string uniqueName = project.UniqueName;
IVsSolution solution = (IVsSolution)Package.GetService(typeof(SVsSolution));
IVsHierarchy hierarchy;
solution.GetProjectOfUniqueName(uniqueName, out hierarchy);
IVsBuildPropertyStorage buildPropertyStorage = hierarchy as IVsBuildPropertyStorage;
if (buildPropertyStorage != null)
{
    uint itemId;
    string fullPath =
(string)project.ProjectItems.Item("Program.cs").Properties.Item("FullPath").Value;
    hierarchy.ParseCanonicalName(fullPath, out itemId);
    buildPropertyStorage.SetItemAttribute(itemId, "MyAttribute", "MyValue");
}
```

Vea también

- [Persistir datos en el archivo de proyecto de MSBuild](#)

Persistir la propiedad de un elemento de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Es posible que desee conservar una propiedad que agregue a un elemento de proyecto, como el autor de un archivo de origen. Puede hacerlo almacenando la propiedad en el archivo de proyecto.

El primer paso para conservar una propiedad en un archivo de [IVsHierarchy](#) proyecto es obtener la jerarquía del proyecto como una interfaz. Puede obtener esta interfaz mediante Automation [IVsMonitorSelection](#) o mediante . Una vez obtenida la interfaz, puede usarla para determinar qué elemento de proyecto está seleccionado actualmente. Una vez que tenga el identificador [SetItemAttribute](#) de elemento de proyecto, puede usar para agregar la propiedad.

En los procedimientos siguientes, `Author` se conserva `Tom` la propiedad `VsPkg.cs` con el valor del archivo de proyecto.

Para obtener la jerarquía del proyecto con el objeto DTE

1. Agregue el código siguiente al VSPackage:

```
EnvDTE.DTE dte = (EnvDTE.DTE)Package.GetService(typeof(EnvDTE.DTE));
EnvDTE.Project project = dte.Solution.Projects.Item(1);

string uniqueName = project.UniqueName;
IVsSolution solution = (IVsSolution)Package.GetService(typeof(SVsSolution));
IVsHierarchy hierarchy;
solution.GetProjectOfUniqueName(uniqueName, out hierarchy);
```

Para conservar la propiedad de elemento de proyecto con el objeto DTE

1. Agregue el código siguiente al código especificado en el método del procedimiento anterior:

```
IVsBuildPropertyStorage buildPropertyStorage =
    hierarchy as IVsBuildPropertyStorage;
if (buildPropertyStorage != null)
{
    uint itemId;
    string fullPath = (string)project.ProjectItems.Item(
        "VsPkg.cs").Properties.Item("FullPath").Value;
    hierarchy.ParseCanonicalName(fullPath, out itemId);
    buildPropertyStorage.SetItemAttribute(itemId, "Author", "Tom");
}
```

Para obtener la jerarquía del proyecto mediante IVsMonitorSelection

1. Agregue el código siguiente al VSPackage:

```

IVsHierarchy hierarchy = null;
IntPtr hierarchyPtr = IntPtr.Zero;
IntPtr selectionContainer = IntPtr.Zero;
uint itemid;

// Retrieve shell interface in order to get current selection
IVsMonitorSelection monitorSelection = Package.GetService(typeof(SVsShellMonitorSelection))
as IVsMonitorSelection;
if (monitorSelection == null)
    throw new InvalidOperationException();

try
{
    // Get the current project hierarchy, project item, and selection container for the current
    selection
    // If the selection spans multiple hierarchies, hierarchyPtr is Zero
    IVsMultiItemSelect multiItemSelect = null;
    ErrorHandler.ThrowOnFailure(
        monitorSelection.GetCurrentSelection(
            out hierarchyPtr, out itemid,
            out multiItemSelect, out selectionContainer));

    // We only care if there is only one node selected in the tree
    if (!(itemid == VSConstants.VSITEMID_NIL ||
        hierarchyPtr == IntPtr.Zero ||
        multiItemSelect != null ||
        itemid == VSConstants.VSITEMID_SELECTION))
    {
        hierarchy = Marshal.GetObjectForIUnknown(hierarchyPtr)
as IVsHierarchy;
    }
}
finally
{
    if (hierarchyPtr != IntPtr.Zero)
        Marshal.Release(hierarchyPtr);
    if (selectionContainer != IntPtr.Zero)
        Marshal.Release(selectionContainer);
}

```

Para conservar la propiedad de elemento de proyecto seleccionada, dada la jerarquía del proyecto

1. Agregue el código siguiente al código especificado en el método del procedimiento anterior:

```

IVsBuildPropertyStorage buildPropertyStorage =
    hierarchy as IVsBuildPropertyStorage;
if (buildPropertyStorage != null)
{
    buildPropertyStorage.SetItemAttribute(itemid, "Author", "Tom");
}

```

Para comprobar que la propiedad se conserva

1. Inicie Visual Studio y, a continuación, abra o cree una solución.
2. Seleccione el elemento de proyecto VsPkg.cs en el **Explorador de soluciones**.
3. Use un punto de interrupción o determine de otro modo que el VSPackage está cargado y que SetItemAttribute se ejecuta.

NOTE

Puede cargar automáticamente un VSPackage [SolutionExists_guiden](#) el contexto de la interfaz de usuario. Para obtener más información, vea [Cargar VSPackages](#).

4. Cierre Visual Studio y, a continuación, abra el archivo de proyecto en el Bloc de notas. Debería ver <la etiqueta Author> con el valor Tom, como se indica a continuación:

```
<Compile Include="VsPkg.cs">
  <Author>Tom</Author>
</Compile>
```

Vea también

- [Herramientas personalizadas](#)

Administrar proyectos universales de Windows

06/04/2020 • 21 minutes to read • [Edit Online](#)

Las aplicaciones universales de Windows son aplicaciones destinadas tanto a Windows 8.1 como a Windows Phone 8.1, lo que permite a los desarrolladores usar código y otros activos en ambas plataformas. El código y los recursos compartidos se mantienen en un proyecto compartido, mientras que el código y los recursos específicos de la plataforma se mantienen en proyectos independientes, uno para Windows y el otro para Windows Phone. Para obtener más información acerca de las aplicaciones universales de Windows, vea [Aplicaciones universales de Windows](#). Las extensiones de Visual Studio que administran proyectos deben tener en cuenta que los proyectos de aplicaciones universales de Windows tienen una estructura que difiere de las aplicaciones de plataforma única. En este tutorial se muestra cómo navegar por el proyecto compartido y administrar los elementos compartidos.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Navegar por el proyecto compartido

1. Cree un proyecto de VSIX de C- denominado **TestUniversalProject**. (**Archivo** > **nuevo** > **proyecto** y, a continuación, paquete de VisualStudio de extensibilidad > de C. >). Agregue una plantilla de elemento de proyecto **Comando personalizado** (en el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar** > **nuevo elemento** y, a continuación, vaya a **Extensibilidad**). Asigne al archivo el nombre **TestUniversalProject**.
2. Agregue una referencia a *Microsoft.VisualStudio.Shell.Interop.12.1.DesignTime.dll* y *Microsoft.VisualStudio.Shell.Interop.14.0.DesignTime.dll* (en la sección **Extensiones**).
3. Abra *TestUniversalProject.cs* y `using` agregue las siguientes directivas:

```
using EnvDTE;
using EnvDTE80;
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.PlatformUI;
using Microsoft.Internal.VisualStudio.PlatformUI;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;
```

4. En `TestUniversalProject` la clase, agregue un campo privado que apunte a la ventana **Salida**.

```
public sealed class TestUniversalProject
{
    IVsOutputWindowPane output;
    . . .
}
```

5. Establezca la referencia en el panel de salida dentro del constructor `TestUniversalProject`:


```

private TestUniversalProject(Package package)
{
    if (package == null)
    {
        throw new ArgumentNullException("package");
    }

    this.package = package;

    OleMenuCommandService commandService = this.ServiceProvider.GetService(typeof(IMenuCommandService))
as OleMenuCommandService;
    if (commandService != null)
    {
        CommandID menuCommandID = new CommandID(MenuGroup, CommandId);
        EventHandler eventHandler = this.ShowMessageBox;
        MenuCommand menuItem = new MenuCommand(eventHandler, menuCommandID);
        commandService.AddCommand(menuItem);
    }
    // get a reference to the Output window
    output =
(IVsOutputWindowPane)ServiceProvider.GetService(typeof(SVsGeneralOutputWindowPane));
}

```

6. Quite el código `ShowMessageBox` existente del método:

```

private void ShowMessageBox(object sender, EventArgs e)
{
}

```

7. Obtenga el objeto DTE, que usaremos para varios propósitos diferentes en este tutorial. Además, asegúrese de que se carga una solución cuando se hace clic en el botón de menú.

```

private void ShowMessageBox(object sender, EventArgs e)
{
    var dte = (EnvDTE.DTE)this.ServiceProvider.GetService(typeof(EnvDTE.DTE));
    if (dte.Solution != null)
    {
        . . .
    }
    else
    {
        MessageBox.Show("No solution is open");
        return;
    }
}

```

8. Busque el proyecto compartido. El proyecto compartido es un contenedor puro; no construye ni produce salidas. El método siguiente busca el primer proyecto compartido [IVsHierarchy](#) en la solución buscando el objeto que tiene la capacidad del proyecto compartido.

```

private IVsHierarchy FindSharedProject()
{
    var sln = (IVsSolution)this.ServiceProvider.GetService(typeof(SVsSolution));
    Guid empty = Guid.Empty;
    IEnumHierarchies enumHiers;

    //get all the projects in the solution
    ErrorHandler.ThrowOnFailure(sln.GetProjectEnum((uint)__VSENUMPROJFLAGS.EPF_LOADEDINSOLUTION, ref
empty, out enumHiers));
    foreach (IVsHierarchy hier in ComUtilities.EnumerableFrom(enumHiers))
    {
        if (PackageUtilities.IsCapabilityMatch(hier, "SharedAssetsProject"))
        {
            return hier;
        }
    }
    return null;
}

```

9. En `ShowMessageBox` el método, genere el título (el nombre del proyecto que aparece en el Explorador de soluciones) del proyecto compartido.

```

private void ShowMessageBox(object sender, EventArgs e)
{
    var dte = (DTE)this.ServiceProvider.GetService(typeof(DTE));

    if (dte.Solution != null)
    {
        var sharedHier = this.FindSharedProject();
        if (sharedHier != null)
        {
            string sharedCaption = HierarchyUtilities.GetHierarchyProperty<string>(sharedHier,
(uint)VSCONSTANTS.VSITEMID.Root,
(int)__VSHPROPID.VSHPROPID_Caption);
            output.OutputStringThreadSafe(string.Format("Found shared project: {0}\n", sharedCaption));
        }
        else
        {
            MessageBox.Show("Solution has no shared project");
            return;
        }
    }
    else
    {
        MessageBox.Show("No solution is open");
        return;
    }
}

```

10. Obtenga el proyecto de plataforma activo. Los proyectos de plataforma son los proyectos que contienen recursos y código específicos de la plataforma. El siguiente método utiliza [VSHPROPID_SharedItemContextHierarchy](#) el nuevo campo para obtener el proyecto de plataforma activo.

```

private IVsHierarchy GetActiveProjectContext(IVsHierarchy hierarchy)
{
    IVsHierarchy activeProjectContext;
    if (HierarchyUtilities.TryGetHierarchyProperty(hierarchy, (uint)VSConstants.VSITEMID.Root,
        (int)__VSHPROPID7.VSHPROPID_SharedItemContextHierarchy, out activeProjectContext))
    {
        return activeProjectContext;
    }
    else
    {
        return null;
    }
}

```

11. En `ShowMessageBox` el método, genere el título del proyecto de plataforma activo.

```

private void ShowMessageBox(object sender, EventArgs e)
{
    var dte = (DTE)this.ServiceProvider.GetService(typeof(DTE));

    if (dte.Solution != null)
    {
        var sharedHier = this.FindSharedProject();
        if (sharedHier != null)
        {
            string sharedCaption = HierarchyUtilities.GetHierarchyProperty<string>(sharedHier,
                (uint)VSConstants.VSITEMID.Root,
                (int)__VSHPROPID.VSHPROPID_Caption);
            output.OutputStringThreadSafe(string.Format("Shared project: {0}\n", sharedCaption));

            var activePlatformHier = this.GetActiveProjectContext(sharedHier);
            if (activePlatformHier != null)
            {
                string activeCaption = HierarchyUtilities.GetHierarchyProperty<string>
                    (activePlatformHier,
                        (uint)VSConstants.VSITEMID.Root, (int)__VSHPROPID.VSHPROPID_Caption);
                output.OutputStringThreadSafe(string.Format("Active platform project: {0}\n",
                    activeCaption));
            }
            else
            {
                MessageBox.Show("Shared project has no active platform project");
            }
        }
        else
        {
            MessageBox.Show("Solution has no shared project");
            return;
        }
    }
    else
    {
        {
            MessageBox.Show("No solution is open");
            return;
        }
    }
}

```

12. Recorre en iteración los proyectos de la plataforma. El siguiente método obtiene todos los proyectos de importación (plataforma) del proyecto compartido.

```
private IEnumerable<IVsHierarchy> EnumImportingProjects(IVsHierarchy hierarchy)
{
    IVsSharedAssetsProject sharedAssetsProject;
    if (HierarchyUtilities.TryGetHierarchyProperty(hierarchy, (uint)VSConstants.VSITEMID.Root,
        (int)__VSHPROPID7.VSHPROPID_SharedAssetsProject, out sharedAssetsProject)
        && sharedAssetsProject != null)
    {
        foreach (IVsHierarchy importingProject in sharedAssetsProject.EnumImportingProjects())
        {
            yield return importingProject;
        }
    }
}
```

IMPORTANT

Si el usuario ha abierto un proyecto de aplicación universal de Windows de C++ en la instancia experimental, el código anterior produce una excepción. Este es un problema conocido. Para evitar la excepción, reemplace el `foreach` bloque anterior por lo siguiente:

```
var importingProjects = sharedAssetsProject.EnumImportingProjects();
for (int i = 0; i < importingProjects.Count; ++i)
{
    yield return importingProjects[i];
}
```

13. En `ShowMessageBox` el método, genere el título de cada proyecto de plataforma. Inserte el código siguiente después de la línea que genera el título del proyecto de plataforma activa. Solo los proyectos de plataforma que se cargan aparecen en esta lista.

```
output.OutputStringThreadSafe("Platform projects:\n");

IEnumerable<IVsHierarchy> projects = this.EnumImportingProjects(sharedHier);

bool isActiveProjectSet = false;
foreach (IVsHierarchy platformHier in projects)
{
    string platformCaption = HierarchyUtilities.GetHierarchyProperty<string>(platformHier,
        (uint)VSConstants.VSITEMID.Root,
        (int)__VSHPROPID7.VSHPROPID_Caption);
    output.OutputStringThreadSafe(string.Format(" * {0}\n", platformCaption));
}
```

14. Cambie el proyecto de plataforma activa. El método siguiente establece [SetProperty](#) el proyecto activo mediante .

```
private int SetActiveProjectContext(IVsHierarchy hierarchy, IVsHierarchy activeProjectContext)
{
    return hierarchy.SetProperty((uint)VSConstants.VSITEMID.Root,
        (int)__VSHPROPID7.VSHPROPID_SharedItemContextHierarchy, activeProjectContext);
}
```

15. En `ShowMessageBox` el método, cambie el proyecto de plataforma activa. Inserte este código `foreach` dentro del bloque.

```

bool isActiveProjectSet = false;
string platformCaption = null;
foreach (IVsHierarchy platformHier in projects)
{
    platformCaption = HierarchyUtilities.GetHierarchyProperty<string>(platformHier,
(uint)VSConstants.VSITEMID.Root,
    (int)__VSHPROPID.VSHPROPID_Caption);
    output.OutputStringThreadSafe(string.Format(" * {0}\n", platformCaption));

    // if this project is neither the shared project nor the current active platform project,
    // set it to be the active project
    if (!isActiveProjectSet && platformHier != activePlatformHier)
    {
        this.SetActiveProjectContext(sharedHier, platformHier);
        activePlatformHier = platformHier;
        isActiveProjectSet = true;
    }
}
output.OutputStringThreadSafe("set active project: " + platformCaption +'\n');

```

16. Ahora Pruébalo. Presione F5 para lanzar la instancia experimental. Cree un proyecto de aplicación de concentrador universal de C. en la instancia experimental (en el cuadro de diálogo Nuevo **proyecto**, Aplicación de concentrador **Hub App****universal** > de **Windows** > 8 de Windows > Visual C. > Después de cargar la solución, vaya al menú **Herramientas** y haga clic en **Invocar TestUniversalProject**, a continuación, compruebe el texto en el panel **Salida** . Debe ver algo parecido a lo siguiente:

```

Found shared project: HubApp.Shared
The active platform project: HubApp.Windows
Platform projects:
 * HubApp.Windows
 * HubApp.WindowsPhone
set active project: HubApp.WindowsPhone

```

Administrar los elementos compartidos en el proyecto de plataforma

1. Busque los elementos compartidos en el proyecto de plataforma. Los elementos del proyecto compartido aparecen en el proyecto de plataforma como elementos compartidos. No puede verlos en el Explorador de **soluciones**, pero puede recorrer la jerarquía del proyecto para encontrarlos. El siguiente método recorre la jerarquía y recopila todos los elementos compartidos. Opcionalmente, genera el título de cada elemento,. Los elementos compartidos se [VSHPROPID_IsSharedItem](#) identifican mediante la nueva propiedad .

```

private void InspectHierarchyItems(IVsHierarchy hier, uint itemid, int level, List<uint> itemIds, bool
getSharedItems, bool printItems)
{
    string caption = HierarchyUtilities.GetHierarchyProperty<string>(hier, itemid,
(int)___VSHPROPID.VSHPROPID_Caption);
    if (printItems)
        output.OutputStringThreadSafe(string.Format("{0}{1}\n", new string('\t', level), caption));

    // if getSharedItems is true, inspect only shared items; if it's false, inspect only unshared items
    bool isSharedItem;
    if (HierarchyUtilities.TryGetHierarchyProperty(hier, itemid,
(int)___VSHPROPID.VSHPROPID_IsSharedItem, out isSharedItem)
        && (isSharedItem == getSharedItems))
    {
        itemIds.Add(itemid);
    }

    uint child;
    if (HierarchyUtilities.TryGetHierarchyProperty(hier, itemid, (int)___VSHPROPID.VSHPROPID_FirstChild,
Unbox.AsUInt32, out child)
        && child != (uint)VSConstants.VSITEMID.Nil)
    {
        this.InspectHierarchyItems(hier, child, level + 1, itemIds, isSharedItem, printItems);

        while (HierarchyUtilities.TryGetHierarchyProperty(hier, child,
(int)___VSHPROPID.VSHPROPID_NextSibling, Unbox.AsUInt32, out child)
            && child != (uint)VSConstants.VSITEMID.Nil)
        {
            this.InspectHierarchyItems(hier, child, level + 1, itemIds, isSharedItem, printItems);
        }
    }
}

```

2. En `ShowMessageBox` el método, agregue el código siguiente para recorrer los elementos de jerarquía del proyecto de plataforma. Insértelo dentro del `foreach` bloque.

```

output.OutputStringThreadSafe("Walk the active platform project:\n");
var sharedItemIds = new List<uint>();
this.InspectHierarchyItems(activePlatformHier, (uint)VSConstants.VSITEMID.Root, 1, sharedItemIds, true,
true);

```

3. Lea los elementos compartidos. Los elementos compartidos aparecen en el proyecto de plataforma como archivos vinculados ocultos y puede leer todas las propiedades como archivos vinculados ordinarios. El código siguiente lee la ruta de acceso completa del primer elemento compartido.

```

var sharedItemId = sharedItemIds[0];
string fullPath;
ErrorHandler.ThrowOnFailure(((IVsProject)activePlatformHier).GetMkDocument(sharedItemId, out
fullPath));
output.OutputStringThreadSafe(string.Format("Shared item full path: {0}\n", fullPath));

```

4. Ahora pruébalo. Presione **F5** para lanzar la instancia experimental. Cree un proyecto de aplicación de concentrador universal de C- en la instancia experimental (en el cuadro de diálogo **Nuevo proyecto, Hub App** Aplicación de concentrador universal > de Windows > 8 de Windows de > Visual C) > vaya al menú **Herramientas** y haga clic en Invocar **TestUniversalProjecty**, a continuación, compruebe el texto en el panel **Salida** . Debe ver algo parecido a lo siguiente:

```

Found shared project: HubApp.Shared
The active platform project: HubApp.Windows
Platform projects:
* HubApp.Windows
* HubApp.WindowsPhone
set active project: HubApp.WindowsPhone
Walk the active platform project:
  HubApp.WindowsPhone
    <HubApp.Shared>
      App.xaml
      App.xaml.cs
      Assets
        DarkGray.png
        LightGray.png
        MediumGray.png
      Common
        NavigationHelper.cs
        ObservableDictionary.cs
        RelayCommand.cs
        SuspensionManager.cs
      DataModel
        SampleData.json
        SampleDataSource.cs
      HubApp.Shared.projitems
      Strings
        en-US
          Resources.resw
      Assets
        HubBackground.theme-dark.png
        HubBackground.theme-light.png
        Logo.scale-240.png
        SmallLogo.scale-240.png
        SplashScreen.scale-240.png
        Square71x71Logo.scale-240.png
        StoreLogo.scale-240.png
        WideLogo.scale-240.png
      HubPage.xaml
        HubPage.xaml.cs
      ItemPage.xaml
        ItemPage.xaml.cs
      Package.appxmanifest
      Properties
        AssemblyInfo.cs
      References
        .NET for Windows Store apps
        HubApp.Shared
        Windows Phone 8.1
      SectionPage.xaml
        SectionPage.xaml.cs

```

Detectar cambios en proyectos de plataforma y proyectos compartidos

1. Puede usar eventos de jerarquía y de proyecto para detectar cambios en proyectos compartidos, al igual que para los proyectos de plataforma. Sin embargo, los elementos de proyecto en el proyecto compartido no están visibles, lo que significa que ciertos eventos no se desencadenan cuando se cambian los elementos de proyecto compartidos.

Tenga en cuenta la secuencia de eventos cuando se cambia el nombre de un archivo de un proyecto:

- a. El nombre del archivo se cambia en el disco.
- b. El archivo de proyecto se actualiza para incluir el nuevo nombre del archivo.

Los eventos de [IVsHierarchyEvents](#) jerarquía (por ejemplo,) suelen realizar un seguimiento de los cambios que se muestran en la interfaz de usuario, como en el **Explorador** de soluciones . Los

eventos de jerarquía consideran que una operación de cambio de nombre de archivo consta de una eliminación de archivos y, a continuación, una adición de archivo. Sin embargo, cuando se cambian los elementos invisibles, el sistema de eventos de jerarquía desencadena un [OnItemDeleted](#) evento, pero no un [OnItemAdded](#) evento. Por lo tanto, si cambia el nombre [OnItemDeleted](#) de [OnItemAdded](#) un archivo en un proyecto de plataforma, [OnItemDeleted](#) obtendrá ambos y , pero si cambia el nombre de un archivo en un proyecto compartido, solo obtendrá .

Para realizar un seguimiento de los cambios en los elementos [ProjectItemsEventsClass](#) de proyecto, puede controlar los eventos de elemento de proyecto DTE (los que se encuentran en). Sin embargo, si está controlando un gran número [IVsTrackProjectDocuments2](#) de eventos, puede obtener un mejor rendimiento controlando los eventos en . En este tutorial solo se muestran los eventos de jerarquía y los eventos DTE. En este procedimiento se agrega un detector de eventos a un proyecto compartido y a un proyecto de plataforma. A continuación, al cambiar el nombre de un archivo en un proyecto compartido y otro archivo en un proyecto de plataforma, puede ver los eventos que se desencadenan para cada operación de cambio de nombre.

En este procedimiento se agrega un detector de eventos a un proyecto compartido y a un proyecto de plataforma. A continuación, al cambiar el nombre de un archivo en un proyecto compartido y otro archivo en un proyecto de plataforma, puede ver los eventos que se desencadenan para cada operación de cambio de nombre.

2. Agregue un detector de eventos. Agregue un nuevo archivo de clase al proyecto y llámelo *HierarchyEventListener.cs*.
3. Abra el archivo *HierarchyEventListener.cs* y agregue las siguientes directivas using:

```
using Microsoft.VisualStudio.Shell.Interop;  
using Microsoft.VisualStudio;  
using System.IO;
```

4. Haga `HierarchyEventListener` que [IVsHierarchyEvents](#) la clase implemente:

```
class HierarchyEventListener : IVsHierarchyEvents  
{ }
```

5. Implemente los [IVsHierarchyEvents](#) miembros de , como en el código siguiente.


```

class HierarchyEventListener : IVsHierarchyEvents
{
    private IVsHierarchy hierarchy;
    IVsOutputWindowPane output;

    internal HierarchyEventListener(IVsHierarchy hierarchy, IVsOutputWindowPane outputWindow) {
        this.hierarchy = hierarchy;
        this.output = outputWindow;
    }

    int IVsHierarchyEvents.OnInvalidateIcon(IntPtr hIcon) {
        return VSConstants.S_OK;
    }

    int IVsHierarchyEvents.OnInvalidateItems(uint itemIDParent) {
        return VSConstants.S_OK;
    }

    int IVsHierarchyEvents.OnItemAdded(uint itemIDParent, uint itemIDSiblingPrev, uint itemIDAdded) {
        output.OutputStringThreadSafe("IVsHierarchyEvents.OnItemAdded: " + itemIDAdded + "\n");
        return VSConstants.S_OK;
    }

    int IVsHierarchyEvents.OnItemDeleted(uint itemID) {
        output.OutputStringThreadSafe("IVsHierarchyEvents.OnItemDeleted: " + itemID + "\n");
        return VSConstants.S_OK;
    }

    int IVsHierarchyEvents.OnItemsAppended(uint itemIDParent) {
        output.OutputStringThreadSafe("IVsHierarchyEvents.OnItemsAppended\n");
        return VSConstants.S_OK;
    }

    int IVsHierarchyEvents.OnPropertyChanged(uint itemID, int propID, uint flags) {
        output.OutputStringThreadSafe("IVsHierarchyEvents.OnPropertyChanged: item ID " + itemID +
"\n");
        return VSConstants.S_OK;
    }
}

```

6. En la misma clase, agregue otro **ItemRenamed** controlador de eventos para el evento DTE , que se produce siempre que se cambia el nombre de un elemento de proyecto.

```

public void OnItemRenamed(EnvDTE.ProjectItem projItem, string oldName)
{
    output.OutputStringThreadSafe(string.Format("[Event] Renamed {0} to {1} in project {2}\n",
        oldName, Path.GetFileName(projItem.get_FileNames(1)), projItem.ContainingProject.Name));
}

```

7. Regístrese para los eventos de jerarquía. Debe registrarse por separado para cada proyecto que está rastreando. Agregue el código `ShowMessageBox` siguiente en , uno para el proyecto compartido y el otro para uno de los proyectos de plataforma.

```
// hook up the event listener for hierarchy events on the shared project
HierarchyEventListener listener1 = new HierarchyEventListener(sharedHier, output);
uint cookie1;
sharedHier.AdviseHierarchyEvents(listener1, out cookie1);

// hook up the event listener for hierarchy events on the
active project
HierarchyEventListener listener2 = new HierarchyEventListener(activePlatformHier, output);
uint cookie2;
activePlatformHier.AdviseHierarchyEvents(listener2, out cookie2);
```

8. Regístrese para el evento [ItemRenamed](#) de elemento de proyecto DTE . Agregue el código siguiente después de conectar el segundo agente de escucha.

```
// hook up DTE events for project items
Events2 dteEvents = (Events2)dte.Events;
dteEvents.ProjectItemsEvents.ItemRenamed += listener1.OnItemRenamed;
```

9. Modifique el elemento compartido. No se pueden modificar elementos compartidos en un proyecto de plataforma; en su lugar, debe modificarlos en el proyecto compartido que es el propietario real de estos elementos. Puede obtener el identificador de elemento [IsDocumentInProject](#) correspondiente en el proyecto compartido con , lo que le da la ruta de acceso completa del elemento compartido. A continuación, puede modificar el elemento compartido. El cambio se propaga a los proyectos de plataforma.

IMPORTANT

Debe averiguar si un elemento de proyecto es un elemento compartido antes de modificarlo.

El método siguiente modifica el nombre de un archivo de elemento de proyecto.

```
private void ModifyFileNameInProject(IVsHierarchy project, string path)
{
    int found;
    uint projectItemID;
    VSDOCUMENTPRIORITY[] priority = new VSDOCUMENTPRIORITY[1];
    if (ErrorHandler.Succeeded(((IVsProject)project).IsDocumentInProject(path, out found, priority, out projectItemID))
        && found != 0)
    {
        var name = DateTime.Now.Ticks.ToString() + Path.GetExtension(path);
        project.SetProperty(projectItemID, (int)__VSHPROPID.VSHPROPID_EditLabel, name);
        output.OutputStringThreadSafe(string.Format("Renamed {0} to {1}\n", path, name));
    }
}
```

10. Llame a este método después de todo el otro código para `ShowMessageBox` modificar el nombre de archivo del elemento en el proyecto compartido. Inserte esto después del código que obtiene la ruta de acceso completa del elemento en el proyecto compartido.

```
// change the file name of an item in a shared project
this.InspectHierarchyItems(activePlatformHier, (uint)VSConstants.VSITEMID.Root, 1, sharedItemIds, true, true);
ErrorHandler.ThrowOnFailure(((IVsProject)activePlatformHier).GetMkDocument(sharedItemId, out fullPath));
output.OutputStringThreadSafe(string.Format("Shared project item ID = {0}, full path = {1}\n", sharedItemId, fullPath));
this.ModifyFileNameInProject(sharedHier, fullPath);
```

11. Compile y ejecute el proyecto. Cree una aplicación de concentrador universal de C- en la instancia experimental, vaya al menú **Herramientas** y haga clic en **Invocar TestUniversalProjecty** compruebe el texto en el panel de salida general. El nombre del primer elemento del proyecto compartido (esperamos que sea el archivo *App.xaml*) debe cambiarse y debería ver que el [ItemRenamed](#) evento se ha desencadenado. En este caso, dado que cambiar el nombre de *App.xaml* hace que *App.xaml.cs* se renombró también, debería ver cuatro eventos (dos para cada proyecto de plataforma). (Los eventos DTE no realizan un seguimiento de los elementos del proyecto compartido.) Debería ver [OnItemDeleted](#) dos eventos (uno para cada [OnItemAdded](#) uno de los proyectos de plataforma), pero ningún evento.
12. Ahora intente cambiar el nombre de un archivo en un proyecto de plataforma y puede ver la diferencia en los eventos que se desencadenan. Agregue el código `ShowMessageBox` siguiente después de la llamada a `ModifyFileName`.

```
// change the file name of an item in a platform project
var unsharedItemIds = new List<uint>();
this.InspectHierarchyItems(activePlatformHier, (uint)VSConstants.VSITEMID.Root, 1, unsharedItemIds,
false, false);

var unsharedItemId = unsharedItemIds[0];
string unsharedPath;
ErrorHandler.ThrowOnFailure(((IVsProject)activePlatformHier).GetMkDocument(unsharedItemId, out
unsharedPath));
output.OutputStringThreadSafe(string.Format("Platform project item ID = {0}, full path = {1}\n",
unsharedItemId, unsharedPath));

this.ModifyFileNameInProject(activePlatformHier, unsharedPath);
```

13. Compile y ejecute el proyecto. Cree un proyecto universal de C- en la instancia experimental, vaya al menú **Herramientas** y haga clic en **Invocar TestUniversalProjecty** compruebe el texto en el panel de salida general. Después de cambiar el nombre del archivo en [OnItemAdded](#) el [OnItemDeleted](#) proyecto de plataforma, debería ver un evento y un evento. Dado que el cambio del archivo no provocó que se cambiaran otros archivos, y dado que los cambios en los elementos de un proyecto de plataforma no se propagan en ningún lugar, solo hay uno de estos eventos.

Ampliar las soluciones

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una solución es una agrupación de uno o varios proyectos que trabajan juntos para crear una aplicación.

En esta sección

En los siguientes tutoriales se muestran algunas formas en las que puede ampliar las soluciones.

1. [Extender el filtro Explorador](#) de soluciones explica cómo filtrar lo que muestra el Explorador de soluciones.
2. Administrar la carga de [proyectos en una solución](#) explica cómo administrar la carga de proyectos para mejorar el rendimiento cuando un usuario carga una solución.

Extienda el filtro del Explorador de soluciones

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede ampliar la funcionalidad de filtro del **Explorador** de soluciones para mostrar u ocultar diferentes archivos. Por ejemplo, puede crear un filtro que muestre solo los archivos de fábrica de clases de C- en el **Explorador** de soluciones , como se muestra en este tutorial.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear un proyecto de paquete de Visual Studio

1. Cree un proyecto `FileFilter` VSIX denominado . Agregue una plantilla de elemento de comando personalizada denominada `FileFilter` . Para obtener más información, consulte [Crear una extensión con un comando de menú](#).
2. Agregue una `System.ComponentModel.Composition` referencia `Microsoft.VisualStudio.Utilities` a y .
3. Haga que el comando de menú aparezca en la barra de herramientas del Explorador de **soluciones**. Abra el archivo `FileFilterPackage.vsct`.
4. Cambie `<Button>` el bloque a lo siguiente:

```
<Button guid="guidFileFilterPackageCmdSet" id="FileFilterId" priority="0x0400" type="Button">
  <Parent guid="guidSHLMainMenu" id="IDG_VS_TOOLBAR_PROJWIN_FILTERS" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <ButtonText>FileNameFilter</ButtonText>
  </Strings>
</Button>
```

Actualización del archivo de manifiesto

1. En el archivo `source.extension.vsixmanifest`, agregue un recurso que sea un componente MEF.
2. En la pestaña **Assets(Assets)**, elija el botón **New (Nuevo)**.
3. En el campo **Tipo** , elija `Microsoft.VisualStudio.MefComponent`.
4. En el campo **Origen**, elija Un proyecto en la **solución actual**.
5. En el campo **Proyecto**, elija `FileFilter`, a continuación, elija el botón **Aceptar**.

Agregue el código de filtro

1. Agregue algunos GUID al archivo `FileFilterPackageGuids.cs`:

```
public const string guidFileFilterPackageCmdSetString = "00000000-0000-0000-0000-00000000"; // get your
GUID from the .vsct file
public const int FileFilterId = 0x100;
```

2. Agregue un archivo de clase al proyecto `FileFilter` denominado `FileNameFilter.cs`.

3. Reemplace el espacio de nombres vacío y la clase vacía por el código siguiente.

El `Task<ReadOnlyObservableSet> GetIncludedItemsAsync(IEnumerable<IVsHierarchyItem rootItems>)` método toma la colección que `rootItems` contiene la raíz de la solución () y devuelve la colección de elementos que se incluirán en el filtro.

El `ShouldIncludeInFilter` método filtra los elementos de la jerarquía del **Explorador** de soluciones en función de la condición que especifique.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.Composition;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using Microsoft.Internal.VisualStudio.PlatformUI;
using Microsoft.VisualStudio.Shell;

namespace FileFilter
{
    // Implements ISolutionTreeFilterProvider. The SolutionTreeFilterProvider attribute declares it as
    // a MEF component
    [SolutionTreeFilterProvider(FileFilterPackageGuids.guidFileFilterPackageCmdSetString, (uint)
    (FileFilterPackageGuids.FileFilterId))]
    public sealed class FileNameFilterProvider : HierarchyTreeFilterProvider
    {
        SVsServiceProvider svcProvider;
        IVsHierarchyItemCollectionProvider hierarchyCollectionProvider;

        // Constructor required for MEF composition
        [ImportingConstructor]
        public FileNameFilterProvider(SVsServiceProvider serviceProvider,
        IVsHierarchyItemCollectionProvider hierarchyCollectionProvider)
        {
            this.svcProvider = serviceProvider;
            this.hierarchyCollectionProvider = hierarchyCollectionProvider;
        }

        // Returns an instance of Create filter class.
        protected override HierarchyTreeFilter CreateFilter()
        {
            return new FileNameFilter(this.svcProvider, this.hierarchyCollectionProvider,
            FileNamePattern);
        }

        // Regex pattern for CSharp factory classes
        private const string FileNamePattern = @"\\w*factory\\w*(\\.cs$)";

        // Implementation of file filtering
        private sealed class FileNameFilter : HierarchyTreeFilter
        {
            private readonly Regex regexp;
            private readonly IServiceProvider svcProvider;
            private readonly IVsHierarchyItemCollectionProvider hierarchyCollectionProvider;

            public FileNameFilter(
                IServiceProvider serviceProvider,
                IVsHierarchyItemCollectionProvider hierarchyCollectionProvider,
                string fileNamePattern)
            {
                this.svcProvider = serviceProvider;
                this.hierarchyCollectionProvider = hierarchyCollectionProvider;
                this.regexp = new Regex(fileNamePattern, RegexOptions.IgnoreCase);
            }

            // Gets the items to be included from this filter provider.
            // rootItems is a collection that contains the root of your solution
            // Returns a collection of items to be included as part of the filter

```

```

// Returns a collection of items to be included as part of the filter
protected override async Task<IObservableSet<IVsHierarchyItem>
GetIncludedItemsAsync(IEnumerable<IVsHierarchyItem> rootItems)
{
    IVsHierarchyItem root = HierarchyUtilities.FindCommonAncestor(rootItems);
    IObservableSet<IVsHierarchyItem> sourceItems;
    sourceItems = await hierarchyCollectionProvider.GetDescendantsAsync(
        root.HierarchyIdentity.NestedHierarchy,
        CancellationToken);

    IFilteredHierarchyItemSet includedItems = await
hierarchyCollectionProvider.GetFilteredHierarchyItemsAsync(
        sourceItems,
        ShouldIncludeInFilter,
        CancellationToken);
    return includedItems;
}

// Returns true if filters hierarchy item name for given filter; otherwise, false</returns>
private bool ShouldIncludeInFilter(IVsHierarchyItem hierarchyItem)
{
    if (hierarchyItem == null)
    {
        return false;
    }
    return this.regexp.IsMatch(hierarchyItem.Text);
}
}
}
}

```

4. En *FileFilter.cs*, quite el código de ubicación y control de comandos del constructor *FileFilter*. El resultado debería tener este aspecto:

```

private FileFilter(Package package)
{
    if (package == null)
    {
        throw new ArgumentNullException("package");
    }

    this.package = package;
}

```

Elimine `ShowMessageBox()` el método también.

5. En *FileFilterPackage.cs*, reemplace `Initialize()` el código del método por el siguiente:

```

protected override void Initialize()
{
    Debug.WriteLine (string.Format(CultureInfo.CurrentCulture, "Entering Initialize() of: {0}",
this.ToString()));
    base.Initialize();
}

```

Prueba del código

1. Compile y ejecute el proyecto. Se muestra una segunda instancia de Visual Studio. Esto se denomina instancia experimental.
2. En la instancia experimental de Visual Studio, abra un proyecto de C.
3. Busque el botón que agregó en la barra de herramientas del Explorador de **soluciones**. Debe ser el cuarto

botón de la izquierda.

4. Al hacer clic en el botón, todos los archivos deben filtrarse y debería ver **todos los elementos se han filtrado de** la vista. en el **Explorador de soluciones**.

Gestionar la carga de proyectos en una solución

06/04/2020 • 10 minutes to read • [Edit Online](#)

Las soluciones de Visual Studio pueden contener un gran número de proyectos. El comportamiento predeterminado de Visual Studio es cargar todos los proyectos en una solución en el momento en que se abre la solución y no permitir que el usuario tenga acceso a ninguno de los proyectos hasta que todos ellos hayan terminado de cargarse. Cuando el proceso de carga del proyecto durará más de dos minutos, se muestra una barra de progreso que muestra el número de proyectos cargados y el número total de proyectos. El usuario puede descargar proyectos mientras trabaja en una solución con varios proyectos, pero este procedimiento tiene algunas desventajas: los proyectos descargados no se compilan como parte de un comando Reconstruir solución y no se muestran descripciones de IntelliSense de tipos y miembros de proyectos cerrados.

Los desarrolladores pueden reducir los tiempos de carga de la solución y administrar el comportamiento de carga de proyectos mediante la creación de un administrador de carga de soluciones. El administrador de carga de soluciones puede asegurarse de que los proyectos se cargan antes de iniciar una compilación en segundo plano, retrasar la carga en segundo plano hasta que se completen otras tareas en segundo plano y realizar otras tareas de administración de carga de proyectos.

Crear un gestor de carga de soluciones

Los desarrolladores pueden crear [IVsSolutionLoadManager](#) un administrador de carga de soluciones implementando y aconsejando a Visual Studio que el administrador de carga de la solución está activo.

Activar un gestor de carga de soluciones

Visual Studio solo permite un administrador de carga de solución en un momento dado, por lo que debe informar a Visual Studio cuando desee activar el administrador de carga de la solución. Si un administrador de carga de segunda solución se activa más adelante, el administrador de carga de la solución se desconectará.

Debe obtener [SVsSolution](#) el servicio y establecer el [__VSPROPID4.VSPROPID_ActiveSolutionLoadManager](#) propiedad:

```
IVsSolution pSolution = GetService(typeof(SVsSolution)) as IVsSolution;
object objLoadMgr = this; //the class that implements IVsSolutionManager
pSolution.SetProperty((int)__VSPROPID4.VSPROPID_ActiveSolutionLoadManager, objLoadMgr);
```

Se [OnDisconnect](#) llama al método cuando Visual Studio se está cerrando o cuando un paquete [SetProperty](#) diferente se ha tomado el control como el administrador de carga de la solución activa mediante una llamada con el [__VSPROPID4.VSPROPID_ActiveSolutionLoadManager](#) propiedad.

Estrategias para diferentes tipos de gestor de carga de soluciones

Puede implementar administradores de carga de soluciones de diferentes maneras, dependiendo de los tipos de soluciones que están destinados a administrar.

Si el administrador de carga de la solución está diseñado para administrar la carga de la solución en general, se puede implementar como parte de un VSPackage. El paquete debe establecerse en [ProvideAutoLoadAttribute](#) autoload agregando el [SolutionOpening_guidVSPackage](#) con un valor de . El administrador de carga de [Initialize](#) la solución se puede activar en el método.

NOTE

Para obtener más información acerca de la carga automática de paquetes, vea [Cargar VSPackages](#).

Dado que Visual Studio reconoce solo el último administrador de carga de solución que se va a activar, los administradores de carga de soluciones generales siempre deben detectar si hay un administrador de carga existente antes de activarse a sí mismos. Si `GetProperty()` llama al servicio de solución para `__VSPROPID4.VSPROPID_ActiveSolutionLoadManager` `null` devuelve, no hay ningún administrador de carga de solución activo. Si no devuelve null, compruebe si el objeto es el mismo que el administrador de carga de la solución.

Si el administrador de carga de la solución está diseñado para administrar solo unos `AdviseSolutionEvents` pocos tipos de `OnBeforeOpenSolution` solución, el VSPackage puede suscribirse a eventos de carga de la solución (mediante una llamada) y usar el controlador de eventos para activar el administrador de carga de la solución.

Si el administrador de carga de soluciones está diseñado para administrar solo soluciones específicas, la información de activación se puede conservar como parte del archivo de solución llamando a `WriteSolutionProps` la sección previa a la solución.

Los administradores de carga de `OnAfterCloseSolution` soluciones específicos deben desactivarse en el controlador de eventos, para no entrar en conflicto con otros administradores de carga de soluciones.

Si necesita un administrador de carga de soluciones solo para conservar las propiedades de carga global del `OnAfterOpenProject` proyecto (por ejemplo, las propiedades establecidas en una página Opciones), puede activar el administrador de carga de la solución en el controlador de eventos, conservar la configuración en las propiedades de la solución y, a continuación, desactivar el administrador de carga de la solución.

Controlar eventos de carga de soluciones

Para suscribirse a eventos `AdviseSolutionEvents` de carga de soluciones, llame cuando active el administrador de carga de soluciones. Si implementa `IVsSolutionLoadEvents`, puede responder a eventos relacionados con diferentes propiedades de carga de proyectos.

- `OnBeforeOpenSolution`: este evento se desencadena antes de que se abra una solución.
- `OnBeforeBackgroundSolutionLoadBegins`: este evento se desencadena después de que la solución se haya cargado por completo, pero antes de que la carga del proyecto en segundo plano comience de nuevo.
- `OnAfterBackgroundSolutionLoadComplete`: este evento se desencadena después de que una solución se haya cargado inicialmente por completo, independientemente de si hay o no un administrador de carga de la solución. También se desencadena después de la carga en segundo plano o la carga de demanda cada vez que la solución se carga por completo. Al mismo tiempo, `SolutionExistsAndFullyLoaded_guid` se reactiva.
- `OnQueryBackgroundLoadProjectBatch`: este evento se desencadena antes de la carga de un proyecto (o proyectos). Para asegurarse de que se completan otros `pfShouldDelayLoadToNextIdle` procesos en segundo plano antes de cargar los proyectos, establezca `true`.
- `OnBeforeLoadProjectBatch`: este evento se desencadena cuando un lote de proyectos está a punto de cargarse. Si `fIsBackgroundIdleBatch` es true, los proyectos se cargarán en segundo plano; si `fIsBackgroundIdleBatch` es false, los proyectos se cargarán sincrónicamente como resultado de una solicitud de usuario, por ejemplo, si el usuario expande un proyecto pendiente en el Explorador de soluciones. Puede controlar este evento para realizar un trabajo costoso `OnAfterOpenProject` que, de lo contrario, tendría que realizarse en .
- `OnAfterLoadProjectBatch`: este evento se desencadena después de cargar un lote de proyectos.

Detectar y gestionar la carga de soluciones y proyectos

Para detectar el estado de carga de [GetProperty](#) proyectos y soluciones, llame con los siguientes valores:

- `__VSPROPID4.VSPROPID_IsSolutionFullyLoaded` `var` : `true` devuelve si se carga la `false` solución y todos sus proyectos, de lo contrario .
- `__VSPROPID4.VSPROPID_IsInBackgroundIdleLoadProjectBatch` `var` : `true` devuelve si se está cargando un `false` lote de proyectos en segundo plano, de lo contrario .
- `__VSPROPID4.VSPROPID_IsInSyncDemandLoadProjectBatch` `var` : `true` devuelve si un lote de proyectos se está cargando sincrónicamente `false` como resultado de un comando de usuario u otra carga explícita, de lo contrario .
- `__VSPROPID2.VSPROPID_IsSolutionClosing` `var` : `true` devuelve si la solución `false` se está cerrando actualmente, de lo contrario .
- `__VSPROPID.VSPROPID_IsSolutionOpening` `var` : `true` devuelve si se está `false` abriendo una solución, de lo contrario .

También puede asegurarse de que los proyectos y soluciones se cargan llamando a uno de los métodos siguientes:

- [EnsureSolutionIsLoaded](#): llamar a este método fuerza los proyectos de una solución para cargar antes de que se devuelva el método.
- [EnsureProjectIsLoaded](#): llamar a este `guidProject` método obliga a los proyectos a cargarse antes de que se devuelva el método.
- [EnsureProjectsAreLoaded](#): llamar a este `guidProjectID` método obliga al proyecto a cargarse antes de que se devuelva el método.

Crear plantillas de proyectos y elementos personalizadas

06/04/2020 • 7 minutes to read • [Edit Online](#)

El SDK de Visual Studio incluye plantillas de proyecto que crean una plantilla de proyecto personalizada y una plantilla de elemento personalizado. Estas plantillas incluyen algunas sustituciones de parámetros comunes y se compilan como archivos zip. No se implementan automáticamente y no están disponibles en la instancia experimental. Debe copiar el archivo zip generado en el directorio de plantilla de usuario.

Las plantillas de creación de plantillas le permiten incluir plantillas en extensiones más grandes. Esto le permite implementar el control de versiones en los archivos de origen y crear un grupo de proyectos de plantilla en un paquete VSIX.

También puede configurar una plantilla para instalar paquetes NuGet. Para obtener más información, vea [Paquetes NuGet en plantillas de Visual Studio](#).

Para escenarios básicos de creación de plantillas, debe usar el Asistente **para exportar plantilla**, que se genera en un archivo comprimido. Para obtener más información sobre la creación de plantillas básicas, vea [Creación de plantillas de proyecto y elemento](#).

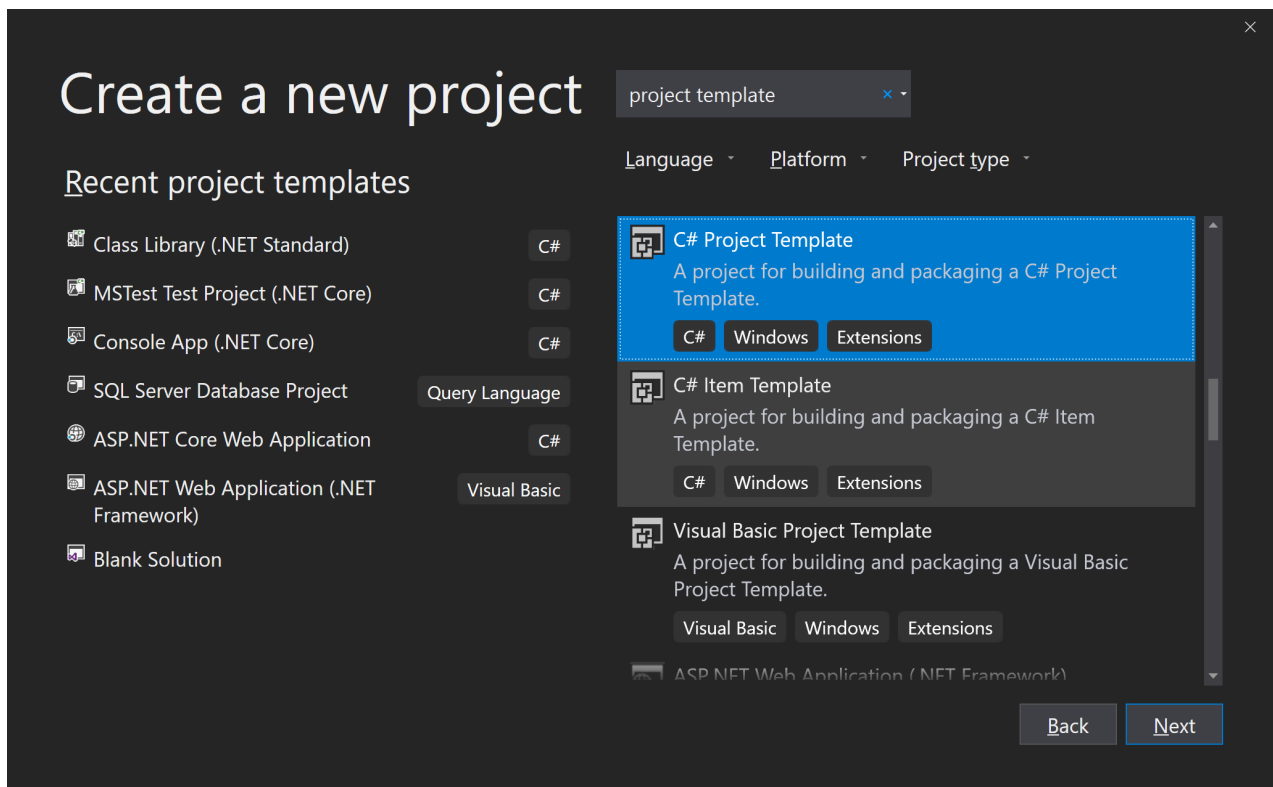
NOTE

A partir de Visual Studio 2017, ya no se realizará el análisis de plantillas de proyecto y elemento personalizadas. En su lugar, la extensión debe proporcionar archivos de manifiesto de plantilla que describan la ubicación de instalación de estas plantillas. Puede usar Visual Studio 2017 para actualizar las extensiones VSIX. Si implementa la extensión mediante un MSI, debe generar los archivos de manifiesto de plantilla a mano. Para obtener más información, vea [Actualizar plantillas de proyecto y elemento personalizadas para Visual Studio 2017](#). El esquema de manifiesto de plantilla se documenta en la referencia de esquema de manifiesto de plantilla de [Visual Studio](#).

Creación de una plantilla de proyecto

1. Cree un proyecto de plantilla de proyecto. Puede encontrar la plantilla de proyecto en el cuadro de diálogo **Nuevo proyecto**, buscando "plantilla de proyecto" y seleccionando la versión de Visual Basic o de C.

La plantilla genera un archivo de clase, un icono, un archivo *.vstemplate*, un archivo de proyecto editable denominado *ProjectTemplate.vbproj* o *ProjectTemplate.csproj* algunos archivos que normalmente generan otros tipos de proyecto, como un archivo *resources.resx*, un archivo *AssemblyInfo* y un archivo *.settings*. Cada archivo de código contiene sustituciones de parámetros comunes cuando sea apropiado.



2. Agregue y quite elementos del proyecto según sea necesario para el proyecto. No quite el archivo de proyecto editable, el archivo *AssemblyInfo* o el archivo *.vstemplate*.
3. Actualice el archivo *.vstemplate* para reflejar las adiciones y eliminaciones. El elemento **Project** debe contener un elemento **ProjectItem** para cada archivo que se va a incluir en la plantilla.
4. Modifique los archivos de código y otro contenido orientado al usuario y agregue las sustituciones de parámetros adecuadas.
5. Modifique el contenido generado según sea necesario.
6. Compile el proyecto.

Visual Studio crea un archivo *.zip* que contiene la plantilla. No se implementa y no está disponible en la instancia experimental.

Creación de una plantilla de elemento

1. Cree un proyecto de plantilla de elemento.

La plantilla genera un archivo de clase, un icono, un archivo *.vstemplate* y un archivo *AssemblyInfo*. El archivo de clase contiene algunas sustituciones de parámetros comunes.

2. Agregue y quite elementos del proyecto según sea necesario para el proyecto.
3. Actualice el archivo *.vstemplate* para reflejar las adiciones y eliminaciones. El elemento **Project** debe contener un elemento **ProjectItem** para cada archivo que se va a incluir en la plantilla.
4. Modifique los archivos de código y otro contenido orientado al usuario y agregue las sustituciones de parámetros adecuadas.
5. Modifique el contenido generado según sea necesario.
6. Compile el proyecto.

Visual Studio crea un archivo comprimido que contiene la plantilla. No se implementa y no está disponible en la instancia experimental.

Implementación

Para implementar la plantilla de proyecto o elemento

1. Crear un proyecto de VSIX. Para obtener más información, consulte Plantilla de [proyecto VSIX](#).
2. Establezca el proyecto VSIX como el proyecto de inicio. En el **Explorador** de soluciones , seleccione el nodo del proyecto VSIX, haga clic con el botón derecho y seleccione **Establecer como proyecto** de inicio .
3. Establezca el proyecto de plantilla de proyecto como un recurso del proyecto VSIX. Abra el archivo *.vsixmanifest*. Vaya a la pestaña **Activos** y haga clic en **Nuevo**.
 - a. Establezca el campo **Type** en **Microsoft.VisualStudio.ProjectTemplate** o **Microsoft.VisualStudio.ItemTemplate**.
 - b. Para el origen, seleccione la opción Un proyecto en la **solución actual** y, a continuación, seleccione el proyecto que contiene la plantilla.
4. Compile la solución y presione F5. Aparece la instancia experimental.
5. Para un proyecto de plantilla de proyecto, debería ver la plantilla de proyecto que aparece en el cuadro de diálogo **Nuevo proyecto** (**Archivo** > **nuevo** > **proyecto**), en el nodo Visual C o Visual Basic. Para un proyecto de plantilla de elemento, debería ver la plantilla de elemento en el cuadro de diálogo **Agregar nuevo elemento**. Para ver el cuadro de diálogo **Agregar nuevo elemento**, en el **Explorador** de soluciones , seleccione el nodo del proyecto y haga clic en **Agregar** > **nuevo elemento**).

Vea también

- [Referencia de plantilla de Visual Studio](#)
- [Paquetes NuGet en plantillas de Visual Studio](#)

Actualizar plantillas personalizadas de proyecto y elemento para Visual Studio 2017

06/04/2020 • 9 minutes to read • [Edit Online](#)

A partir de Visual Studio 2017, Visual Studio detecta plantillas de proyecto y elemento que se han instalado mediante un .vsix o un .msi de una manera diferente a las versiones anteriores de Visual Studio. Si tiene extensiones que usan plantillas de proyecto o elemento personalizadas, debe actualizar las extensiones. En este artículo se explica lo que debe hacer.

Este cambio solo afecta a Visual Studio 2017. No afecta a versiones anteriores de Visual Studio.

Si desea crear un proyecto o una plantilla de elemento como parte de una extensión VSIX, consulte [Creación de plantillas de proyecto y elemento personalizadas](#).

Escaneo de plantillas

En versiones anteriores de Visual Studio, `devenv /setup` o `devenv /installvstemplates` examinaban el disco local para buscar plantillas de proyecto y elemento. A partir de Visual Studio 2017, el análisis se realiza solo para la ubicación de nivel de usuario. La ubicación de nivel de usuario predeterminada es `%USERPROFILE%\Documents\<versión>\de Visual Studio`. Esta ubicación se usa para las plantillas generadas por el > comando Plantillas de exportación de proyectos..., si la opción **Importar automáticamente la plantilla en Visual Studio** está seleccionada en el asistente.

Para otras ubicaciones (no de usuario), debe incluir un archivo manifest(.vstman) que especifique la ubicación y otras características de la plantilla. El archivo .vstman se genera junto con el archivo .vstemplate utilizado para las plantillas. Si instala la extensión con un .vsix, puede hacerlo volviendo a compilar la extensión en Visual Studio 2017. Pero si utiliza un archivo .msi, debe realizar los cambios manualmente. Para obtener una lista de lo que debe hacer para realizar estos cambios, consulte **Actualizaciones de extensiones instaladas con un archivo . MSI** más adelante en esta página.

Cómo actualizar una extensión VSIX con plantillas de proyecto o elemento

1. Abra la solución en Visual Studio 2017. Se le pedirá que actualice el código. Haga clic en **OK**.
2. Una vez completada la actualización, es posible que deba cambiar la versión del destino de instalación. En el proyecto VSIX, abra el archivo `source.extension.vsixmanifest` y seleccione la pestaña **Install Targets**. Si el campo Intervalo de **versiones** es `[14.0]`, haga clic en **Editar** y cámbielo para incluir Visual Studio 2017. Por ejemplo, puede establecerlo en `[14.0,15.0]` para instalar la extensión en Visual Studio 2015 o Visual Studio 2017, o en `[15.0]` para instalarla solo en Visual Studio 2017.
3. Vuelva a compilar el código.
4. Cierre Visual Studio.
5. Instale el VSIX.
6. Puede probar la actualización haciendo lo siguiente:
 - a. El cambio de análisis de archivos se activa mediante la siguiente clave del Registro:

```
reg add hklm-software-microsoft-visualstudio-15.0-VSTemplate /v
```

`DisableTemplateScanning /t REG_DWORD /d 1 /reg:32`

- b. Después de agregar la clave, ejecute `devenv /installvstemplates`.
- c. Vuelva a abrir Visual Studio. Debe encontrar la plantilla en la ubicación esperada.

NOTE

Las plantillas de proyecto de extensibilidad de Visual Studio no están disponibles cuando la clave del Registro está presente. Debe eliminar la clave del Registro (y volver a ejecutar `devenv /installvstemplates`) para usarlas.

Otras recomendaciones para implementar plantillas de proyectos y elementos

- Evite el uso de archivos de plantilla comprimidos. Los archivos de plantilla comprimidos deben descomprimirse para recuperar recursos y contenido, por lo que serán más costosos de usar. En su lugar, debe implementar plantillas de proyecto y elemento como archivos individuales en su propio directorio para acelerar la inicialización de plantillas. Para las extensiones VSIX, las tareas de compilación del SDK descomprimirán automáticamente cualquier plantilla comprimida al crear el archivo VSIX.
- Evite utilizar entradas de ID de paquete/recurso para el nombre de la plantilla, la descripción, el icono o la vista previa para evitar cargas innecesarias del ensamblado de recursos durante la detección de plantillas. En su lugar, puede usar manifiestos localizados para crear una entrada de plantilla para cada configuración regional, que usa nombres o propiedades localizadas.
- Si incluye plantillas como elementos de archivo, es posible que la generación de manifiestos no proporcione los resultados esperados. En ese caso, tendrá que agregar un manifiesto generado manualmente al proyecto VSIX.

Cambios de archivo en las plantillas de proyecto y elemento

Mostramos los puntos de diferencia entre las versiones de Visual Studio 2015 y Visual Studio 2017 de los archivos de plantilla, para que pueda crear los nuevos archivos correctamente.

Este es el archivo .vstemplate de proyecto predeterminado creado por Visual Studio 2015:


```
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Project" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
xmlns:sdk="http://schemas.microsoft.com/developer/vstemplate-sdkextension/2010">
  <TemplateData>
    <Name>ProjectTemplate1</Name>
    <Description>ProjectTemplate1</Description>
    <Icon>ProjectTemplate1.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
    <SortOrder>1000</SortOrder>
    <TemplateID>05b79cc9-2146-4716-a8e5-7e085cdd2221</TemplateID>
    <CreateNewFolder>true</CreateNewFolder>
    <DefaultName>ProjectTemplate1</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
  </TemplateData>
  <TemplateContent>
    <Project File="ProjectTemplate.csproj" ReplaceParameters="true">
      <ProjectItem ReplaceParameters="true"
TargetFileName="Properties\AssemblyInfo.cs">AssemblyInfo.cs</ProjectItem>
      <ProjectItem ReplaceParameters="true" OpenInEditor="true">Class1.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Aquí está el archivo .vstman (puede encontrarlo en el directorio de manifiesto del proyecto VSIX) que resultó de la reconstrucción del proyecto VSIX:

```
<VSTemplateManifest Version="1.0" Locale="1033"
xmlns="http://schemas.microsoft.com/developer/vstemplatemanifest/2015">
  <VSTemplateContainer TemplateType="Project">
    <RelativePathOnDisk>CSharp\1033\ProjectTemplate1</RelativePathOnDisk>
    <TemplateFileName>ProjectTemplate1.vstemplate</TemplateFileName>
    <VSTemplateHeader>
      <TemplateData xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
        <Name>ProjectTemplate1</Name>
        <Description>ProjectTemplate1</Description>
        <Icon>ProjectTemplate1.ico</Icon>
        <ProjectType>CSharp</ProjectType>
        <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
        <SortOrder>1000</SortOrder>
        <TemplateID>05b79cc9-2146-4716-a8e5-7e085cdd2221</TemplateID>
        <CreateNewFolder>true</CreateNewFolder>
        <DefaultName>ProjectTemplate1</DefaultName>
        <ProvideDefaultName>true</ProvideDefaultName>
      </TemplateData>
    </VSTemplateHeader>
  </VSTemplateContainer>
</VSTemplateManifest>
```

La información proporcionada por el [elemento TemplateData](#) sigue siendo la misma. El ** <elemento VSTemplateContainer> ** apunta al archivo .vstemplate de la plantilla asociada.

Este es el elemento predeterminado .vstemplate archivo creado por Visual Studio 2015:

```
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Item" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
xmlns:sd="http://schemas.microsoft.com/developer/vstemplate-sdkextension/2010">
  <TemplateData>
    <Name>ItemTemplate1</Name>
    <Description>ItemTemplate1</Description>
    <Icon>ItemTemplate1.ico</Icon>
    <TemplateID>bfeadf8e-a251-4109-b605-516b88e38c8d</TemplateID>
    <ProjectType>CSharp</ProjectType>
    <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
    <NumberOfParentCategoriesToRollUp>1</NumberOfParentCategoriesToRollUp>
    <DefaultName>Class.cs</DefaultName>
  </TemplateData>
  <TemplateContent>
    <References>
      <Reference>
        <Assembly>System</Assembly>
      </Reference>
    </References>
    <ProjectItem ReplaceParameters="true">Class.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

Aquí está el archivo .vstman (puede encontrarlo en el directorio de manifiesto del proyecto VSIX) que resultó de la reconstrucción del proyecto VSIX:

```
<VSTemplateManifest Version="1.0" Locale="1033"
xmlns="http://schemas.microsoft.com/developer/vstemplatemanifest/2015">
  <VSTemplateContainer TemplateType="Item">
    <RelativePathOnDisk>CSharp\1033\ItemTemplate1</RelativePathOnDisk>
    <TemplateFileName>ItemTemplate1.vstemplate</TemplateFileName>
    <VSTemplateHeader>
      <TemplateData xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
        <Name>ItemTemplate1</Name>
        <Description>ItemTemplate1</Description>
        <Icon>ItemTemplate1.ico</Icon>
        <TemplateID>bfeadf8e-a251-4109-b605-516b88e38c8d</TemplateID>
        <ProjectType>CSharp</ProjectType>
        <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
        <NumberOfParentCategoriesToRollUp>1</NumberOfParentCategoriesToRollUp>
        <DefaultName>Class.cs</DefaultName>
      </TemplateData>
    </VSTemplateHeader>
  </VSTemplateContainer>
</VSTemplateManifest>
```

La información proporcionada por el ** <elemento de>TemplateData** sigue siendo la misma. El ** <elemento VSTemplateContainer> ** apunta al archivo .vstemplate para la plantilla asociada

Para obtener más información acerca de los diferentes elementos del archivo .vstman, vea Referencia de esquema de manifiesto de plantilla de [Visual Studio](#).

Actualizaciones de extensiones instaladas con un archivo . Msi

Algunas extensiones basadas en MSI implementan plantillas en ubicaciones de plantillas comunes, como los directorios siguientes:

- <Directorio de instalación de Visual Studio>, Common7, IDE<ProjectTemplates/ItemTemplates>
- <Directorio de instalación de Visual Studio>, Common7, IDE, ExtensionName\ > \

< <Project/ItemTemplates>

Si la extensión realiza una implementación basada en MSI, debe generar el manifiesto de plantilla manualmente y asegurarse de que se incluye en la configuración de la extensión. Compare los ejemplos .vstman enumerados anteriormente y la Referencia de esquema de manifiesto de plantilla de [Visual Studio](#).

Cree manifiestos independientes para las plantillas de proyecto y elemento, y deben apuntar al directorio de plantillas raíz como se especificó anteriormente. Cree un manifiesto por extensión y configuración regional.

Vea también

- [Solución de problemas de detección de plantillas](#)
- [Creación de plantillas de proyectos y elementos personalizadas](#)

Solución de problemas de instalación de plantillas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Si tiene problemas para implementar las plantillas de proyecto o elemento, puede habilitar el registro de diagnóstico.

1. Cree un archivo pkgdef en la carpeta *Common7-IDE-CommonExtensions* para la instalación. Por ejemplo, C:-
Archivos de programa (x86)-Microsoft Visual Studio-2017-Empresa-Common7-IDE-CommonExtensions-EnablePkgDefLogging.pkgdef.
1. Cree un archivo pkgdef en la carpeta *Common7-IDE-CommonExtensions* para la instalación. Por ejemplo, C:-
Archivos de programa (x86)-Microsoft Visual Studio-2019-Empresa-Common7-IDE-CommonExtensions-EnablePkgDefLogging.pkgdef.
2. Agregue lo siguiente al archivo pkgdef:

```
[${RootKey}\VsTemplate]
"EnableTemplateDiscoveryLog"=dword:00000001
```

3. Abra un símbolo del sistema `devenv /updateConfiguration` para [desarrolladores](#) para la instalación y ejecute .
4. Abra Visual Studio e inicie los cuadros de diálogo Nuevo proyecto y Nuevo elemento para inicializar ambos árboles de plantilla.

El registro de la plantilla ahora aparece en **%LOCALAPPDATA%-Microsoft-VisualStudio-15.0_[instanceid]-VsTemplateDiagnosticsList.csv** (instanceid corresponde al identificador de instalación de la instancia de Visual Studio). Cada inicialización del árbol de plantillas anexa entradas a este registro.

4. Abra Visual Studio e inicie los cuadros de diálogo **Crear un nuevo proyecto** y **Nuevo elemento** para inicializar ambos árboles de plantilla.

El registro de la plantilla ahora aparece en **%LOCALAPPDATA%-Microsoft-VisualStudio-16.0_[instanceid]-VsTemplateDiagnosticsList.csv** (instanceid corresponde al identificador de instalación de la instancia de Visual Studio). Cada inicialización del árbol de plantillas anexa entradas a este registro.

El archivo de registro contiene las siguientes columnas:

- **FullPathToTemplate**, que tiene los siguientes valores:
 - 1 para la implementación basada en manifiestos
 - 0 para la implementación basada en disco
- **TemplateName**
- Otras propiedades de plantilla

NOTE

Para deshabilitar el registro, quite el archivo pkgdef o cambie el valor de `EnableTemplateDiscoveryLog` a y, a continuación, vuelva a `dword:00000000` ejecutarlo. `devenv /updateConfiguration`

Vea también

- [Creación de plantillas de proyectos y elementos personalizadas](#)

Cómo: Usar asistentes con plantillas de proyecto

06/04/2020 • 18 minutes to read • [Edit Online](#)

Visual Studio proporciona la interfaz [IWizard](#) que, cuando se implementa, permite ejecutar código personalizado cuando un usuario crea un proyecto a partir de una plantilla.

La personalización de plantillas de proyecto se puede usar para mostrar la interfaz de usuario personalizada que recopila la entrada del usuario para personalizar la plantilla, agregar archivos adicionales a la plantilla o cualquier otra acción permitida en un proyecto.

Los [IWizard](#) métodos de interfaz se llaman en varios momentos mientras se crea el proyecto, comenzando tan pronto como un usuario hace clic en **Aceptar** en el **nuevo proyecto** cuadro de diálogo. Cada método de la interfaz se denomina de modo que describa el punto en el que se llama. Por ejemplo, Visual [RunStarted](#) Studio llama inmediatamente cuando comienza a crear el proyecto, por lo que es una buena ubicación para escribir código personalizado para recopilar la entrada del usuario.

Crear un proyecto de plantilla de proyecto con un proyecto VSIX

Empieza a crear una plantilla personalizada con el proyecto de plantilla de proyecto, que forma parte del SDK de Visual Studio. En este procedimiento, usaremos un proyecto de plantilla de proyecto de C, pero también hay un proyecto de plantilla de proyecto de Visual Basic. A continuación, agregue un proyecto VSIX a la solución que contiene el proyecto de plantilla de proyecto.

1. Cree un proyecto de plantilla de proyecto **File** > de C- (en Visual Studio, seleccione **Nuevo** > **proyecto** de archivo y busque "plantilla de proyecto"). Así lo llamo **MyProjectTemplate**.

NOTE

Es posible que se le pida que instale el SDK de Visual Studio. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

2. Agregue un nuevo proyecto VSIX en la misma solución que el proyecto de plantilla de proyecto (en el Explorador de **soluciones**, seleccione el nodo de solución, haga clic con el botón derecho y seleccione **Agregar** > **nuevo proyecto** y busque "vsix"). Así lo llamo **MyProjectWizard**.
3. Establezca el proyecto VSIX como el proyecto de inicio. En el **Explorador** de soluciones, seleccione el nodo del proyecto VSIX, haga clic con el botón derecho y seleccione **Establecer como proyecto** de inicio.
4. Agregue el proyecto de plantilla como un recurso del proyecto VSIX. En el **Explorador** de soluciones, en el nodo del proyecto VSIX, busque el archivo *source.extension.vsixmanifest*. Haga doble clic en él para abrirlo en el editor de manifiestos.
5. En el editor de **manifiestos**, seleccione la pestaña **Activos** en el lado izquierdo de la ventana.
6. En la pestaña **Activos**, seleccione **Nuevo**. En la ventana **Agregar nuevo recurso**, para el campo **Tipo**, seleccione **Microsoft.VisualStudio.ProjectTemplate**. En el campo **Origen**, seleccione **Un proyecto en la solución actual**. En el campo **Proyecto**, seleccione **MyProjectTemplate**. A continuación, haga clic en **Aceptar**.
7. Compile la solución y comience la depuración. Se muestra una segunda instancia de Visual Studio. Esto puede tardar unos minutos.

8. En la segunda instancia de Visual Studio, intente crear un nuevo proyecto con la nueva plantilla (**Archivo > nuevo > proyecto**, busque "myproject"). El nuevo proyecto debe aparecer con una clase denominada **Class1**. ¡Ahora ha creado una plantilla de proyecto personalizada! Detenga la depuración ahora.

Crear un asistente de plantilla personalizado

Este procedimiento muestra cómo crear un asistente personalizado que abre un formulario Windows Forms antes de crear el proyecto. El formulario permite a los usuarios agregar un valor de parámetro personalizado que se agrega al código fuente durante la creación del proyecto.

1. Configure el proyecto VSIX para permitirle crear un ensamblaje.
2. En el **Explorador de soluciones**, seleccione el nodo del proyecto VSIX. Debajo del **Explorador de soluciones**, debería ver la ventana **Propiedades**. Si no lo hace, seleccione **Ver > ventana Propiedades** o pulse **F4**. En la ventana **Propiedades**, seleccione ☐ **true** los siguientes campos para:
 - **Incluir ensamblaje en el contenedor VSIX**
 - **Incluir símbolos de depuración en el contenedor VSIX**
 - **Incluir símbolos de depuración en la implementación local de VSIX**
3. Agregue el ensamblado como un recurso al proyecto VSIX. Abra el archivo *source.extension.vsixmanifest* y seleccione la pestaña **Assets**. En la ventana **Agregar nuevo recurso**, en **Tipo**, seleccione **Microsoft.VisualStudio.Assembly**, en **Origen**, seleccione **Un proyecto en la solución actual**, en **Proyecto**, seleccione **MyProjectWizard**.
4. Agregue las siguientes referencias al proyecto VSIX. (En el **Explorador de soluciones**, en el nodo del proyecto VSIX, seleccione **Referencias**, haga clic con el botón derecho y seleccione **Agregar referencia**.) En el cuadro de diálogo **Agregar referencia**, en la pestaña **Marco de trabajo**, busque el ensamblado **System.Windows.Forms** y selecciónelo. Busque y seleccione también los ensamblajes **System** y **System.Drawing**. Ahora seleccione la pestaña **EnvDTE Extensiones**. También busque el **Microsoft.VisualStudio.TemplateWizardInterface** ensamblado y selecciónelo. Haga clic en **OK**.
5. Agregue una clase para la implementación del asistente al proyecto VSIX. (En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto VSIX y seleccione **Agregar**, a continuación, **Nuevo elemento**, a continuación, **Clase**.) Asigne a la clase el nombre **WizardImplementation**.
6. Reemplace el código del archivo *WizardImplementationClass.cs* por el código siguiente:

```

using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TemplateWizard;
using System.Windows.Forms;
using EnvDTE;

namespace MyProjectWizard
{
    public class WizardImplementation:IWizard
    {
        private UserInputForm inputForm;
        private string customMessage;

        // This method is called before opening any item that
        // has the OpenInEditor attribute.
        public void BeforeOpeningFile(ProjectItem projectItem)
        {
        }

        public void ProjectFinishedGenerating(Project project)
        {
        }

        // This method is only called for item templates,
        // not for project templates.
        public void ProjectItemFinishedGenerating(ProjectItem
            projectItem)
        {
        }

        // This method is called after the project is created.
        public void RunFinished()
        {
        }

        public void RunStarted(object automationObject,
            Dictionary<string, string> replacementsDictionary,
            WizardRunKind runKind, object[] customParams)
        {
            try
            {
                // Display a form to the user. The form collects
                // input for the custom message.
                inputForm = new UserInputForm();
                inputForm.ShowDialog();

                customMessage = UserInputForm.CustomMessage;

                // Add custom parameters.
                replacementsDictionary.Add("$custommessage$",
                    customMessage);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        // This method is only called for item templates,
        // not for project templates.
        public bool ShouldAddProjectItem(string filePath)
        {
            return true;
        }
    }
}

```


El **UserInputForm** al que se hace referencia en este código se implementará más adelante.

La clase `WizardImplementation` contiene las implementaciones de método para cada miembro de [IWizard](#). En este ejemplo, solo el método [RunStarted](#) realiza una tarea. Los demás métodos no hacen nada o devuelven `true`.

El método [RunStarted](#) acepta cuatro parámetros:

- Un parámetro [Object](#) que se puede convertir en el objeto `_DTE` raíz, para permitirle personalizar el proyecto.
- Un parámetro [Dictionary<TKey,TValue>](#) que contiene una colección de todos los parámetros predefinidos en la plantilla. Para obtener más información sobre los parámetros de plantilla, consulte [Parámetros de plantilla](#).
- Un parámetro [WizardRunKind](#) que contiene información sobre qué tipo de plantilla se utiliza.
- Matriz [Object](#) que contiene un conjunto de parámetros pasados al asistente por Visual Studio.

Este ejemplo agrega un valor de parámetro del formulario de entrada de datos al parámetro [Dictionary<TKey,TValue>](#). Cada instancia del parámetro `$custommessage$` del proyecto se reemplazará por el texto escrito por el usuario.

7. Ahora cree **userInputForm**. En el archivo *WizardImplementation.cs*, agregue el código

`WizardImplementation` siguiente después del final de la clase.

```

public partial class UserInputForm : Form
{
    private static string customMessage;
    private TextBox textBox1;
    private Button button1;

    public UserInputForm()
    {
        this.Size = new System.Drawing.Size(155, 265);

        button1 = new Button();
        button1.Location = new System.Drawing.Point(90, 25);
        button1.Size = new System.Drawing.Size(50, 25);
        button1.Click += button1_Click;
        this.Controls.Add(button1);

        textBox1 = new TextBox();
        textBox1.Location = new System.Drawing.Point(10, 25);
        textBox1.Size = new System.Drawing.Size(70, 20);
        this.Controls.Add(textBox1);
    }
    public static string CustomMessage
    {
        get
        {
            return customMessage;
        }
        set
        {
            customMessage = value;
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {
        customMessage = textBox1.Text;
        this.Close();
    }
}

```

El formulario de datos proporcionados por el usuario proporciona un formulario simple para escribir un parámetro personalizado. El formulario contiene un cuadro de texto denominado `textBox1` y un botón denominado `button1`. Cuando se hace clic en el botón, el texto del cuadro de texto se almacena en el parámetro `customMessage`.

Conecte el asistente a la plantilla personalizada

Para que la plantilla de proyecto personalizada use el asistente personalizado, debe firmar el ensamblado del asistente y agregar algunas líneas a la plantilla de proyecto personalizada para que sepa dónde encontrar la implementación del asistente cuando se crea un nuevo proyecto.

1. Firme el ensamblado. En el **Explorador de soluciones**, seleccione el proyecto VSIX, haga clic con el botón derecho y seleccione **Propiedades del proyecto**.
2. En la ventana Propiedades del **proyecto**, seleccione la ficha **Firma** en la ficha **Firma**, active **Firmar el ensamblado**. En el campo Elegir un archivo `** < **` de clave de nombre **seguro**, seleccione **Nueva**>. En la ventana Crear clave de **nombre seguro**, en el campo Nombre de **archivo** de clave, escriba **key.snk**. Desactive el campo **Proteger mi archivo de claves con una contraseña**.
3. En el Explorador de **soluciones**, seleccione el proyecto VSIX y busque la ventana **Propiedades**.
4. Establezca el campo **Copiar salida de compilación en Directorio** de salida en **true**. Esto permite que el ensamblado se copie en el directorio de salida cuando se reconstruya la solución. Todavía está contenido en

`.vsix` el archivo. Debe ver el ensamblado para averiguar su clave de firma.

5. Recompila la solución.
6. Ahora puede encontrar el archivo `key.snk` en el directorio del proyecto `MyProjectWizard*` <(la ubicación del disco>, `MyProjectTemplate`, `MyProjectWizard`, `key.snk*`). Copie el archivo `key.snk`.
7. Vaya al directorio de salida y busque el ensamblado* <(la ubicación del disco>, `MyProjectTemplate/MyProjectWizard`, `bin`, `Debug`, `MyProjectWizard.dll*`). Pegue el archivo `key.snk` aquí. (Esto no es absolutamente necesario, pero facilitará los siguientes pasos.)
8. Abra una ventana de comandos y cambie al directorio en el que se ha creado el ensamblado.
9. Busque la herramienta de firma `sn.exe`. Por ejemplo, en un sistema operativo Windows 10 de 64 bits, una ruta de acceso típica sería la siguiente:

C:\Archivos de programa\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.6.1 Herramientas

Si no puede encontrar la herramienta, intente ejecutar **donde** `/R . sn.exe` en la ventana de comandos. Tome nota de la ruta.

10. Extraiga la clave pública del archivo `key.snk`. En la ventana de comandos, escriba

<ubicación de sn.exe> `-sn.exe -p key.snk outfile.key`.

¡No olvide rodear la ruta de acceso de `sn.exe` con comillas si hay espacios en los nombres de directorio!

11. Obtenga el token de clave pública del archivo de salida:

<ubicación de sn.exe> `-sn.exe -t outfile.key`.

Una vez más, no olvide las comillas. Debería ver una línea en la salida como esta

El token <de clave pública es token>

Anote este valor.

12. Agregue la referencia al asistente personalizado al archivo `.vstemplate` de la plantilla de proyecto. En el Explorador de **soluciones**, busque el archivo denominado `MyProjectTemplate.vstemplate` ábralo. Después del <final de la sección `TemplateContent`>, agregue la siguiente sección:

```
<WizardExtension>
  <Assembly>MyProjectWizard, Version=1.0.0.0, Culture=Neutral, PublicKeyToken=token</Assembly>
  <FullClassName>MyProjectWizard.WizardImplementation</FullClassName>
</WizardExtension>
```

Donde **MyProjectWizard** es el nombre del ensamblado y **token** es el token que copió en el paso anterior.

13. Guarde todos los archivos en el proyecto y vuelva a generar.

Agregue el parámetro personalizado a la plantilla

En este ejemplo, el proyecto utilizado como plantilla muestra el mensaje especificado en el formulario de entrada de usuario del asistente personalizado.

1. En el Explorador de **soluciones**, vaya al proyecto **MyProjectTemplate** y abra `Class1.cs`.
2. En el método `Main` de la aplicación, agregue la siguiente línea de código.

```
Console.WriteLine($"$custommessage$");
```

El parámetro `$custommessage$` se reemplaza por el texto escrito en el formulario de datos proporcionados por el usuario cuando se crea un proyecto a partir de la plantilla.

Aquí está el archivo de código completo antes de que se haya exportado a una plantilla.

```
using System;
using System.Collections.Generic;
$if$ ($targetframeworkversion$ >= 3.5)using System.Linq;
$endif$using System.Text;

namespace $safeprojectname$
{
    public class Class1
    {
        static void Main(string[] args)
        {
            Console.WriteLine("$custommessage$");
        }
    }
}
```

Utilice el asistente personalizado

Ahora puede crear un proyecto a partir de la plantilla y utilizar el asistente personalizado.

1. Vuelva a generar la solución e inicie la depuración. Aparece una segunda instancia de Visual Studio.
2. Cree un nuevo proyecto MyProjectTemplate. (**Archivo > nuevo > proyecto**).
3. En el cuadro de diálogo **Nuevo proyecto**, busque "myproject" para buscar la plantilla, escriba un nombre y haga clic en **Aceptar**.

Se abrirá el formulario de datos proporcionados por el usuario.

4. Escriba un valor para el parámetro personalizado y haga clic en el botón.

El formulario de datos proporcionados por el usuario del asistente se cierra y se crea un proyecto a partir de la plantilla.

5. En el **Explorador** de soluciones, haga clic con el botón secundario en el archivo de código fuente y haga clic en **Ver código**.

Observe que `$custommessage$` se ha reemplazado con el texto escrito en el formulario de datos proporcionados por el usuario del asistente.

Vea también

- [IWizard](#)
- [Personalización de plantillas](#)
- [Elemento WizardExtension \(plantillas de Visual Studio\)](#)
- [Paquetes NuGet en plantillas de Visual Studio](#)

Referencia de esquema de plantilla de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta sección contiene información sobre los elementos XML en archivos *.vstemplate*, que son archivos que almacenan metadatos para plantillas de proyecto, plantillas de elementos y kits de inicio.

Puede usar *vstemplate.xsd* para validar archivos *.vstemplate* personalizados. Este archivo está disponible en .. Carpeta \de instalación de Visual Studio>, Xml, Esquemas, 1033, vstemplate.xsd <.

ELEMENTO	ELEMENTOS SECUNDARIOS	ATRIBUTOS
AppliesTo	None	None
Assembly (plantilla)	--	--
Assembly (extensión de asistente)	--	--
BuildProjectOnload	--	--
CreateInPlace	--	--
CreateNewFolder	--	--
CustomDataSignature	--	--
CustomParameter	--	Nombre Value
CustomParameters	CustomParameter	--
DefaultName	--	--
Descripción	--	Paquete id
EnableEditOfLocationField	--	--
EnableLocationBrowseButton	--	--
Carpeta	ProjectItem Carpeta	Nombre
	[en desuso]	--
FullClassName	--	--

ELEMENTO	ELEMENTOS SECUNDARIOS	ATRIBUTOS
Oculto	--	--
Icono	--	Paquete id
LocationField	--	--
LocationFieldMRUPrefix	--	--
MaxFrameworkVersion	--	--
Nombre	--	Paquete id
NumberOfParentCategoriesToRollUp	--	--
PreviewImage	--	--
Project	Carpeta ProjectItem	Archivo TargetFileName ReplaceParameters
ProjectCollection	ProjectTemplateLink SolutionFolder	--
ProjectItem (plantillas de elementos)	--	SubType CustomTool ItemType ReplaceParameters TargetFileName
ProjectItem (plantillas de proyecto)	--	TargetFileName ReplaceParameters OpenInEditor OpenOrder OpenInWebBrowser OpenInHelpBrowser
ProjectSubType	--	--
ProjectTemplateLink	--	ProjectName

ELEMENTO	ELEMENTOS SECUNDARIOS	ATRIBUTOS
ProjectType	--	--
PromptForSaveOnCreation	--	--
ProvideDefaultName	--	--
Referencia	Ensamblado	--
Referencias.	Referencia	--
RequiredFrameworkVersion	--	--
RequiredPlatformVersion	--	Versión
SDKReference	--	Paquete
ShowByDefault	--	--
SolutionFolder	ProjectTemplateLink SolutionFolder	Nombre
SortOrder	--	--
SupportsCodeSeparation	--	--
SupportsLanguageDropDown	--	--
SupportsMasterPage	--	--
TargetPlatformName	RequiredPlatformVersion	--
TemplateContent	ProjectCollection proyecto Referencias ProjectItem CustomParameters	BuildOnLoad

ELEMENTO	ELEMENTOS SECUNDARIOS	ATRIBUTOS
TemplateData	Nombre	--
	Descripción	
	Icono	
	PreviewImage	
	ProjectType	
	ProjectSubType	
	TemplateID	
	TemplateGroupID	
	SortOrder	
	CreateNewFolder	
	DefaultName	
	ProvideDefaultName	
	PromptForSaveOnCreation	
	EnableLocationBrowseButton	
	EnableEditOfLocationField	
	Hidden	
	DisplayInParentCategories	
	LocationFieldMRUPrefix	
	NumberOfParentCategoriesToRollUp	
	CreateInPlace	
	BuildOnLoad	
	BuildProjectOnload	
	ShowByDefault	
	LocationField	
	SupportsMasterPage	
	SupportsCodeSeparation	
	SupportsLanguageDropDown	
	RequiredFrameworkVersion	
	FrameworkVersion	
	MaxFrameworkVersion	
	CustomDataSignature	
	TargetPlatformName	

ELEMENTO	targetPlatformName ELEMENTOS SECUNDARIOS	ATRIBUTOS
TemplateGroupID	--	--
TemplateID	--	--
VSTemplate	TemplateData TemplateContent WizardExtension WizardData	Tipo Versión
WizardData	--	Nombre
WizardExtension	Ensamblado FullClassName	--

Vea también

- [Crear plantillas de proyecto y elemento](#)

Elemento AppliesTo (plantillas de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Especifica una expresión opcional para que coincida [VsProjectCapabilityExpressionMatcher](#) con una o varias capacidades (consulte). Las capacidades se exponen mediante tipos de proyecto a través de la jerarquía como una propiedad [_VSHPROPID5.VSHPROPID_ProjectCapabilities](#). De esta manera, la plantilla se puede compartir en varios tipos de proyecto que tengan funciones aplicables comunes.

Este elemento es opcional. Puede haber un máximo de una instancia en un archivo de plantilla. Este elemento solo sirve para designar una plantilla de elemento como aplicable, de acuerdo con las funciones del proyecto activo actualmente seleccionado. No se puede utilizar para designar una plantilla de elemento como no aplicable. Si `AppliesTo` no está presente o la expresión no es capaz de indicar si la plantilla es aplicable, se utiliza `TemplateID` o `TemplateGroupID` para crear la plantilla aplicable, como en las versiones anteriores del producto.

Apareció por primera vez en Visual Studio 2013 Update 2. Para hacer referencia a la versión correcta, vea [Hacer referencia a ensamblados entregados en Visual Studio 2013 SDK Update 2](#).

```
<VSTemplate>
  <TemplateData>
    <AppliesTo>
```

Sintaxis

```
<AppliesTo>Capability1</AppliesTo>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla.

Valor de texto

Se requiere un valor de texto. Este texto especifica las funciones del proyecto.

La sintaxis de expresión válida se define como:

- La expresión de capacidad, como "(VisualC | CSharp) + (MSTest | NUnit)".
- El "|" es el operador OR.

- Los caracteres "&" y "+" son operadores AND.
- El carácter "!" es el operador NOT.
- Los paréntesis indican el orden de prioridad de la evaluación.
- Una expresión null o vacía se evalúa como una coincidencia.
- Las capacidades del proyecto pueden ser cualquier carácter excepto estos\caracteres reservados: ""';,;+ -*/ !- |&%\$-()-[] <>?{} \t\b\n\r

Ejemplo

En el ejemplo siguiente se muestran tres plantillas diferentes. `Template1` se aplica a todos los tipos de proyecto `WindowsAppContainer` de C- o a cualquier otro tipo de proyecto que admita la capacidad. `Template2` se aplica a todos los proyectos de C- de cualquier tipo. `Template3` se aplica a los proyectos de C# que no son proyectos `WindowsAppContainer`.

```
<!-- Template 1 -->
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Item" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <AppliesTo>CSharp | WindowsAppContainer</AppliesTo>
  </TemplateData>
</VSTemplate>

<!-- Template 2 -->
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Item" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <AppliesTo>CSharp</AppliesTo>
  </TemplateData>
</VSTemplate>

<!-- Template 3 -->
<?xml version="1.0" encoding="utf-8"?>
<VSTemplate Version="3.0.0" Type="Item" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <AppliesTo>CSharp_Class + (!WindowsAppContainer)</AppliesTo>
  </TemplateData>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento Assembly (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica información sobre un ensamblado, que la plantilla utiliza para agregar una referencia de ese ensamblado a los proyectos.

```
<VSTemplate <> TemplateContent < <> <referencias> referencia> ensamblado>
```

Sintaxis

```
<Assembly> AssemblyName </Assembly>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Referencia	Especifica la referencia de ensamblado para agregar cuando el elemento se agrega a un proyecto.

Valor de texto

Se requiere un valor de texto.

Este texto especifica el ensamblado que se va a agregar a un proyecto cuando se crea una instancia de la plantilla de elemento. Este nombre de ensamblado debe especificarse de una de las siguientes maneras:

- Como nombre completo del ensamblado. Por ejemplo:

```
<Assembly>  
  MyAssembly, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a, Custom = null.  
</Assembly>
```

- Como referencia de texto simple. Por ejemplo:

```
<Assembly> System </Assembly>
```

Observaciones

`Assembly` es un elemento secundario obligatorio de `Reference`.

Los `Reference` `References`, elementos, y `Assembly` solo se pueden utilizar `Type` en `Item` archivos *.vstemplate* que tengan un valor de atributo de .

Ejemplo

En el ejemplo `TemplateContent` siguiente se muestra el elemento de una plantilla de elemento. Este XML agrega referencias a los ensamblados *System.dll* y *System.Data.dll*.

```
<TemplateContent>
  <References>
    <Reference>
      <Assembly>
        System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
    <Reference>
      <Assembly>
        System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
  </References>
  ...
</TemplateContent>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento Assembly (extensión del asistente para plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica el nombre o el nombre seguro `IWizard` del ensamblado que implementa la interfaz.

`<VSTemplate <> WizardExtension> <> de ensamblados`

Sintaxis

```
<Assembly>AssemblyName</Assembly>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
WizardExtension	Contiene los elementos de registro para personalizar el asistente de plantilla.

Valor de texto

Se requiere un valor de texto.

Este texto especifica el ensamblado `IWizard` que implementa la interfaz. Este nombre de ensamblado debe especificarse como un nombre de ensamblado completo. Por ejemplo,

```
MyAssembly, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11dd0a3a, Custom = null .
```

Observaciones

`Assembly` es un elemento secundario obligatorio de `WizardExtension`.

Ejemplo

En el ejemplo siguiente se muestran los Visual C# metadatos de la plantilla de proyecto estándar para una aplicación de Windows.

```
<VSTemplate Version="3.0.0" Type="Item"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyTemplate</Name>
    <Description>Template using IWizard extension</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
  <WizardExtension>
    <Assembly>MyWizard, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11dd0a3a,
Custom=null</Assembly>
    <FullClassName>MyWizard.CustomWizard</FullClassName>
  </WizardExtension>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Cómo: Usar asistentes con plantillas de proyecto](#)

Atributo y elemento BuildOnLoad

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si se va a compilar el proyecto inmediatamente después de crearlo. **BuildOnLoad** es un atributo y un elemento.

Jerarquía de elementos:

```
<VSTemplate>
  <TemplateData>
    <BuildOnLoad>
```

Sintaxis de elementos

```
<BuildOnLoad> true/false </BuildOnLoad>
```

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto para el elemento **BuildOnLoad**. El texto debe `true` `false` ser o , que indica si se va a compilar el proyecto inmediatamente después de crearlo.

Observaciones

BuildOnLoad es un atributo opcional. El valor predeterminado es `false` .

Ejemplo

En el ejemplo siguiente se muestran los metadatos de una plantilla de C- cuando se utiliza **BuildOnLoad** como elemento:


```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <BuildOnLoad>true</BuildOnLoad>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Elemento BuildProjectOnLoad](#)
- [Elemento TemplateContent](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

Elemento BuildProjectOnLoad (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Crea solo nuevos proyectos a medida que los crea y los agrega a una solución. La solución completa no está creada.

Jerarquía de elementos:

```
<VSTemplate>
  <TemplateData>
    <BuildProjectOnLoad>
```

Sintaxis

```
<BuildProjectOnLoad> true/false </BuildProjectOnLoad>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Categoriza la plantilla y define cómo aparece en los cuadros de diálogo Nuevo proyecto y Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o para indicar si se va a compilar solo el nuevo proyecto cuando se crea a partir de la plantilla.

Observaciones

`BuildProjectOnLoad` es un elemento opcional. El valor predeterminado es `false`.

Ejemplo

En el ejemplo siguiente se muestran los metadatos de una plantilla de Visual C.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <BuildProjectOnLoad>true</BuildProjectOnLoad>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Atributo y elemento BuildOnLoad](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

Elemento CreateInPlace (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si se debe crear el proyecto y realizar la sustitución de parámetros en la ubicación especificada, o realizar la sustitución de parámetros en una ubicación temporal y, a continuación, guardar el proyecto en la ubicación especificada.

<VSTemplate <> TemplateData> <> De CreateInPlace

Sintaxis

```
<CreateInPlace> true/false </CreateInPlace>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ser `true` o `false` . Si `true` , se crea el proyecto y se realiza la sustitución de parámetros en la ubicación especificada en el cuadro de diálogo **Nuevo proyecto** . Si `false` , la sustitución de parámetros se realiza en una ubicación temporal y, a continuación, el proyecto se copia en la ubicación especificada.

Observaciones

`CreateInPlace` es un elemento opcional. El valor predeterminado es `true` .

Ejemplo

En el siguiente ejemplo se muestran los metadatos de una plantilla de Visual C#:

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <CreateInPlace>false</CreateInPlace>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Crear plantillas de proyecto y elemento](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

Elemento CreateNewFolder (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Determina si hay que comprobar que el directorio de destino donde se va a crear el proyecto no existe. Si el directorio existe, se puede crear un directorio nuevo para el proyecto. Esta configuración se suele sobrescribir por medio de la marca del Registro `NewProjectRequiresNewFolder(VsTemplate)` (

`HKEY_LOCAL_MACHINE/SOFTWARE(/Wow6432Node)/Microsoft/VisualStudio/<version number>/Projects/<project GUID>`) que usan todos los tipos de proyecto comunes para determinar si un proyecto nuevo se creará en un directorio nuevo.

`<VsTemplate <> TemplateData> <> CreateNewFolder`

Sintaxis

```
<CreateNewFolder>
  true/false
</CreateNewFolder>
```

Tipo

Boolean

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ser `true` o `false`, e indica si se debe crear o no una nueva carpeta contenedora cuando se cree un proyecto a partir de la plantilla.

Observaciones

`CreateNewFolder` es un elemento opcional. El valor predeterminado es `true` .

El valor especificado en el elemento `CreateNewFolder` solo se admite en Visual Studio si el sistema de proyectos subyacente lo admite.

Ejemplo

En el ejemplo de código siguiente se especifica que no se debe crear una nueva carpeta al crear un proyecto a partir de la plantilla.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <CreateNewFolder>false</CreateNewFolder>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento CustomDataSignature (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la firma de texto para localizar los datos personalizados.

<VSTemplate <> TemplateData> <> De CustomDataSignature

Sintaxis

```
<CustomDataSignature>"string"</CustomDataSignature>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto es una cadena que tiene la firma de texto necesaria para localizar los datos personalizados.

Observaciones

`CustomDataSignature` es un elemento opcional.

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento CustomParameter (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Contiene un nombre de parámetro personalizado y un valor para usar cuando se crea un proyecto o elemento a partir de la plantilla.

Sintaxis

```
<CustomParameter Name="name" Value="value">
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Name</code>	Necesario. El nombre del parámetro. El formato de los parámetros es <code>\$name\$</code> .
<code>Value</code>	Necesario. El valor de reemplazo para el parámetro.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>CustomParameters</code>	Agrupar los parámetros personalizados que se van a pasar al asistente de plantilla cuando el asistente realiza reemplazos de parámetros.

Observaciones

Cuando una `CustomParameter` plantilla contiene elementos, cada instancia del `Name` atributo se reemplaza por el `Value` atributo en los archivos de proyecto o elemento creados.

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar varios parámetros personalizados en una plantilla. Cuando se crea un proyecto o elemento a partir de `$color1$` una `$color2$` plantilla con los siguientes `Red` parámetros personalizados, todas las instancias de y en los archivos de plantilla se reemplazarán con y `Blue`, respectivamente.

```
<CustomParameters>
  <CustomParameter Name="$color1$" Value="Red"/>
  <CustomParameter Name="$color2$" Value="Blue "/>
</CustomParameters>
```

Vea también

- [Elemento CustomParameters \(plantillas de Visual Studio\)](#)
- [Parámetros de plantilla](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

Elemento CustomParameters (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Agrupar los parámetros personalizados que se van a pasar al asistente de plantilla cuando el asistente realiza reemplazos de parámetros.

Sintaxis

```
<CustomParameters>
  <CustomParameter/>
  <CustomParameter/>
</CustomParameters>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
CustomParameter	Elemento opcional. Contiene un nombre de parámetro personalizado y un valor para usar cuando se crea un proyecto o elemento a partir de la plantilla. Puede haber cero o más elementos <code>CustomParameter</code> en un elemento <code>CustomParameters</code> .

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateContent	Especifica el contenido de la plantilla.

Observaciones

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar varios parámetros personalizados en una plantilla. Cuando se crea un proyecto o elemento a partir de `$color1$` una `$color2$` plantilla con los siguientes `Red` parámetros personalizados, todas las instancias de `y` en los archivos de plantilla se reemplazarán con `Blue`, respectivamente.

```
<CustomParameters>
  <CustomParameter Name="$color1$" Value="Red"/>
  <CustomParameter Name="$color2$" Value="Blue "/>
</CustomParameters>
```

Vea también

- [Elemento CustomParameter \(plantillas de Visual Studio\)](#)
- [Parámetros de plantilla](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

Elemento DefaultName (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica el nombre que generará el sistema de proyectos de Visual Studio para el proyecto o elemento cuando se crea.

<VSTemplate < > TemplateData> < > DefaultName

Sintaxis

```
<DefaultName>  
    Default Project Name  
</DefaultName>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

Este texto especifica el nombre predeterminado del proyecto o elemento.

Observaciones

`DefaultName` es un elemento opcional.

Para los proyectos, este elemento especifica el nombre del directorio que almacena el proyecto en el disco. Para los elementos, especifica el nombre de archivo del archivo de origen.

Al crear un proyecto o elemento, puede modificar el nombre predeterminado mediante la opción **Nombre**, que está disponible en el cuadro de diálogo **Nuevo proyecto** o en el cuadro de diálogo **Agregar nuevo elemento**.

Si no desea que el sistema de proyectos genere el nombre predeterminado para el `False` proyecto o elemento, establezca el elemento `ProvideDefaultName` en .

Ejemplo

En el ejemplo siguiente se muestran los Visual C# metadatos de la plantilla de elemento estándar para una clase.

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <DefaultName>MyClass.cs</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem ReplaceParameters="true">MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento Description (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la descripción de la plantilla tal como aparece en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**.

<DESCRIPción <de la <> de templateData> de VSTemplate>

Sintaxis

```
<Description>
  Template Description
</Description>
```

```
<Description Package="{PackageID}" ID="ResourceID" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Package</code>	Atributo opcional, para escenarios de usuario avanzados. Un identificador GUID que especifica el id. paquete de Visual Studio.
<code>ID</code>	Atributo opcional, para escenarios de usuario avanzados. Especifica el identificador de recurso de Visual Studio.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto a menos que se usen los atributos `Package` y `ID`.

El texto proporciona una descripción de la plantilla.

Observaciones

`Description` es un elemento secundario `TemplateData` necesario del elemento.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento EnableEditOfLocationField (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si el usuario puede editar el campo de ubicación.

```
<VSTemplate <> TemplateData> <> EnableEditOfLocationField
```

Sintaxis

```
<EnableEditOfLocationField> true/false </EnableEditOfLocationField>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

None

Elementos secundarios

None

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ☐ true ☐ false ser o , que indica si el usuario puede o no editar el cuadro de texto **Ubicación** en el cuadro de diálogo **Nuevo proyecto**.

Observaciones

`EnableEditOfLocationField` es un elemento opcional. El valor ☐ true predeterminado es , que permite al usuario editar el valor en el cuadro de texto **Ubicación** del cuadro de diálogo **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, el cuadro de texto **Ubicación** especifica el directorio donde se guarda un nuevo proyecto.

Ejemplo

En el ejemplo siguiente se Visual C# muestran los metadatos de una aplicación de Windows.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <EnableEditOfLocationField>false</EnableEditOfLocationField>
    <EnableLocationBrowseButton>false</EnableLocationBrowseButton>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento EnableLocationBrowseButton (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si el botón **Examinar** está disponible en el cuadro de diálogo **Nuevo proyecto**, para que los usuarios puedan modificar fácilmente el directorio predeterminado donde se guarda un nuevo proyecto.

<VSTemplate <> TemplateData> <EnableLocationBrowseButton>

Sintaxis

```
<EnableLocationBrowseButton> true/false </EnableLocationBrowseButton>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o , que indica si se debe mostrar o no el botón **Examinar** en el cuadro de diálogo **Nuevo proyecto**.

Observaciones

`EnableLocationBrowseButton` es un elemento opcional. El valor `true` predeterminado es , que muestra el botón **Examinar** en el cuadro de diálogo **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, el cuadro de texto **Ubicación** especifica el directorio donde se guarda un nuevo proyecto. El botón **Examinar** le ayuda a modificar este directorio mostrando el cuadro de diálogo **Ubicación del proyecto**, que le permite navegar fácilmente a un directorio diferente que está disponible desde el equipo y, a continuación, elegirlo como el directorio donde se guarda el nuevo proyecto.

Ejemplo

En el ejemplo siguiente se Visual C# muestran los metadatos de una aplicación de Windows.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <EnableLocationBrowseButton>false</EnableLocationBrowseButton>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento Folder (plantillas de proyecto de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Especifica una carpeta que se agregará al proyecto.

<VSTemplate <> TemplateContent < <>> de carpeta de> de proyecto

Sintaxis

```
<Folder Name="Project Folder">
  <Folder> ... </Folder>
  <ProjectItem> ... </ProjectItem>
</Folder>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Name</code>	Atributo necesario. El nombre de la carpeta del proyecto.
<code>TargetFolderName</code>	Atributo opcional. Especifica el nombre que se va a asignar a la carpeta cuando se crea un proyecto a partir de la plantilla. Este atributo es útil para usar el reemplazo de parámetros para crear un nombre de carpeta o asignar un nombre a una carpeta con una cadena internacional que no se puede usar directamente en el archivo <i>.zip</i> .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
<code>Folder</code>	Especifica una carpeta que se va a agregar al proyecto. <code>Folder</code> elementos pueden <code>Folder</code> contener elementos secundarios.
<code>ProjectItem</code>	Especifica un archivo que se va a agregar al proyecto.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>Project</code>	Elemento secundario opcional de <code>TemplateContent</code> .

Observaciones

`Folder` es un hijo `Project` opcional de .

Puede utilizar cualquiera de los métodos siguientes para organizar los elementos del proyecto en carpetas de una plantilla:

- Incluya las carpetas en el archivo *.zip* de plantilla y agréguelas al proyecto en `ProjectItem` el archivo `Folder.vstemplate` especificando la ruta de acceso al archivo en los elementos, sin elementos. Éste es el método recomendado. Por ejemplo:

...

```
<ProjectItem>\Folder\item.cs</ProjectItem>
```

```
<ProjectItem>Form1.cs</ProjectItem>
```

...

- Incluya las carpetas en el archivo *.zip* de plantilla y agréguelas al proyecto en el archivo *.vstemplate* con `Folder` elementos. Por ejemplo:

...

```
<Folder name="Folder">
```

```
<ProjectItem>item.cs</ProjectItem>
```

```
</Folder>
```

```
<ProjectItem>Form1.cs</ProjectItem>
```

...

- No incluya carpetas en el archivo *.zip* `TargetFileName` de plantilla, pero agregue carpetas con el atributo del `ProjectItem` elemento. Por ejemplo:

...

```
<ProjectItem TargetFileName="\Folder\item.cs">item.cs</ProjectItem>
```

```
<ProjectItem>Form1.cs</ProjectItem>
```

...

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación de Windows.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <Folder Name="Properties">
        <ProjectItem>AssemblyInfo.cs</ProjectItem>
        <ProjectItem>Resources.resx</ProjectItem>
        <ProjectItem>Resources.Designer.cs</ProjectItem>
        <ProjectItem>Settings.settings</ProjectItem>
        <ProjectItem>Settings.Designer.cs</ProjectItem>
      </Folder>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)
- [ProjectItem \(Elemento, Plantillas de elementos de Visual Studio\)](#)

Elemento FullClassName (extensión del asistente de plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

El nombre completo de la clase `IWizard` que implementa la interfaz.

`<VSTemplate <> WizardExtension> ... <FullClassName>`

Sintaxis

```
<FullClassName>ClassName</FullClassName>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
WizardExtension	Contiene los elementos de registro para personalizar el asistente de plantilla.

Valor de texto

Se requiere un valor de texto.

Este texto especifica la clase `IWizard` que implementa la interfaz. La clase especificada debe existir en el ensamblado especificado por el [Assembly](#) elemento.

Observaciones

`FullClassName` es un elemento secundario obligatorio de `WizardExtension`.

Ejemplo

En el ejemplo siguiente se muestran los Visual C# metadatos de la plantilla de proyecto estándar para una aplicación de Windows.


```
<VSTemplate Version="3.0.0" Type="Item"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyTemplate</Name>
    <Description>Template using IWizard extension</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
  <WizardExtension>
    <Assembly>MyWizard, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
Custom=null</Assembly>
    <FullClassName>MyWizard.CustomWizard</FullClassName>
  </WizardExtension>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)
- [Cómo: Usar asistentes con plantillas de proyecto](#)

Elemento oculto (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si la plantilla aparece en los cuadros de diálogo Nuevo proyecto o **Agregar nuevo elemento**.

```
<VSTemplate>
  <TemplateData>
    <Hidden>
```

Sintaxis

```
<Hidden>true</Hidden>
<Hidden>false</Hidden>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o , que indica si la plantilla aparecerá o no en los cuadros de diálogo **Nuevo proyecto** o **Agregar nuevo elemento** .

Observaciones

`Hidden` es un elemento opcional.

Si se especifica, no se `TemplateData` requiere ningún otro elemento secundario del elemento.

Ejemplo

En el ejemplo siguiente se muestran los metadatos de una plantilla de C.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <Hidden>true</Hidden>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento Icon (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la ruta de acceso y el nombre de archivo del archivo de imagen que actúa como icono, que aparece en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**, para la plantilla.

<VSTemplate <> TemplateData> <Icon>

Sintaxis

```
<Icon>  
  IconFileName  
</Icon>
```

```
<Icon Package="{PackageID}" ID="ResourceID" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Package</code>	Atributo opcional, para escenarios de usuario avanzados. Un identificador GUID que especifica el id. paquete de Visual Studio.
<code>ID</code>	Atributo opcional, para escenarios de usuario avanzados. Especifica el identificador de recurso de Visual Studio.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto a menos que se usen los atributos `Package` y `ID`.

El texto proporciona la ruta de acceso y el nombre de archivo del icono de plantilla que aparecerá en el cuadro de diálogo **Nuevo proyecto**.

Observaciones

`Icon` es un elemento secundario obligatorio de `TemplateData`.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento LocationField (plantillas de proyecto de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si el cuadro de texto **Ubicación** del cuadro de diálogo **Nuevo proyecto** está habilitado, deshabilitado u oculto para la plantilla de proyecto.

<VSTemplate <> TemplateData> <> de LocationField

Sintaxis

```
<LocationField> Enabled/Disabled/Hidden </LocationField>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el nuevo proyecto .

Valor de texto

Se requiere un valor de texto.

Los valores de texto válidos son:

- `Enabled`, que especifica que el cuadro **Ubicación** del cuadro de diálogo **Nuevo proyecto** está habilitado.
- `Disabled`, que especifica que el cuadro **Ubicación** del cuadro de diálogo **Nuevo proyecto** está deshabilitado.
- `Hidden`, que especifica que el cuadro **Ubicación** del cuadro de diálogo **Nuevo proyecto** está oculto.

Observaciones

El valor predeterminado es `Enabled`.

El cuadro de texto **Ubicación** del cuadro de diálogo **Nuevo proyecto** permite a los usuarios cambiar el directorio predeterminado en el que se guardan los nuevos proyectos.

El valor especificado `Location` en el elemento solo lo respeta el cuadro de diálogo si el sistema de proyecto subyacente lo admite.

Ejemplo

En el siguiente ejemplo se muestran los metadatos de una plantilla de Visual C#:

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <LocationField>Disabled</LocationField>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento LocationFieldMRUPrefix (Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica las rutas de acceso utilizadas más recientemente (MRU) en el cuadro de diálogo **Nuevo proyecto** y **Agregar nuevo elemento**.

Sintaxis

```
<LocationFieldMRUPrefix> ... </LocationFieldMRUPrefix>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Observaciones

Este elemento solo se debe utilizar Programa Visual Studio Industry Partner para las plantillas producidas a través del archivo .

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento MaxFrameworkVersion (plantillas de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Especifica la versión máxima de .NET Framework que requiere la plantilla. Determina el valor más alto disponible en la lista desplegable **Versión** de marco de trabajo de destino del cuadro de diálogo **Nuevo proyecto**. Para que los usuarios puedan seleccionar una versión de framework, también debe especificar [RequiredFrameworkVersion](#) como la versión mínima de .NET Framework para la plantilla.

IMPORTANT

A partir de Visual Studio 2017 versión 15.6, la lista desplegable **Versión** de marco de trabajo de destino ya no es un filtro para las plantillas mostradas en la sección **Plantillas** del cuadro de diálogo **Nuevo proyecto**. En su lugar, la lista desplegable **Versión** de marco de trabajo de destino funciona como un selector de marco de trabajo para la plantilla seleccionada.

```
<VSTemplate <> TemplateData> <> MaxFrameworkVersion
```

Sintaxis

```
<MaxFrameworkVersion> ... </MaxFrameworkVersion>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ser el número de versión más alto de .NET Framework permitido por la plantilla.

Observaciones

`MaxFrameworkVersion` es un elemento opcional. El `MaxFrameworkVersion` elemento debe omitirse a menos que sea necesario, para no limitar inadvertidamente el intervalo admitido de versiones de .NET Framework para la plantilla. También debe omitirse si .NET Framework no es aplicable a la plantilla.

Ejemplo

En el ejemplo siguiente se Visual C# muestran los metadatos de una plantilla de clase estándar.

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class template.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <RequiredFrameworkVersion>3.0</RequiredFrameworkVersion>
    <MaxFrameworkVersion>4.7.1</MaxFrameworkVersion>
    <DefaultName>MyClass</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem>MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

En este ejemplo, la versión máxima de .NET Framework que `MaxFrameworkVersion` requiere la plantilla, representada por , es 4.7.1. Un proyecto creado con esta plantilla puede tener como destino versiones de .NET Framework hasta 4.7.1.

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento Name (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica el nombre de la plantilla tal como aparece en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**.

<NOMBRE de <> <de TemplateData de VS>Template>

Sintaxis

```
<Name> Template Name </Name>
```

```
<Name Package="{PackageID}" ID="ResourceID" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Package</code>	Atributo opcional, para escenarios de usuario avanzados. Un identificador GUID que especifica el id. paquete de Visual Studio.
<code>ID</code>	Atributo opcional, para escenarios de usuario avanzados. Especifica el identificador de recurso de Visual Studio.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto a menos que se usen los atributos `Package` y `ID`.

El texto proporciona el nombre de la plantilla.

Observaciones

`Name` es un elemento secundario obligatorio de `TemplateData` .

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento NumberOfParentCategoriesToRollUp (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica el número de categorías primarias que mostrarán la plantilla en el cuadro de diálogo **Nuevo proyecto**.

```
<VSTemplate <> TemplateData> <NumberOfParentCategoriesToRollUp>
```

Sintaxis

```
<NumberOfParentCategoriesToRollUp>  
1  
</NumberOfParentCategoriesToRollUp>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se `integer` requiere un valor.

Este valor especifica el número de categorías primarias que mostrarán la plantilla en el cuadro de diálogo **Nuevo proyecto**.

Observaciones

`NumberOfParentCategoriesToRollUp` es un elemento opcional.

Ejemplo

En este ejemplo se Visual C# muestran los metadatos de una aplicación de Windows. Si una plantilla con estos metadatos se coloca Visual C# dos niveles de carpeta por debajo del nodo de nivel superior, la plantilla aparecerá en el nodo de nivel superior en el cuadro de diálogo **Nuevo proyecto**. Si `NumberOfParentCategoriesToRollUp` no se establece, la plantilla solo aparece en el nodo en el que se encuentra físicamente.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <NumberOfParentCategoriesToRollUp>2</NumberOfParentCategoriesToRollUp>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento PreviewImage (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la imagen de vista previa, como un nombre de archivo, para la imagen de vista previa que aparecerá en el cuadro de diálogo **Nuevo proyecto** o Agregar nuevo elemento.

<VSTemplate <> TemplateData> <> PreviewImage

Sintaxis

```
<PreviewImage>"filename"</PreviewImage>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento.

Valor de texto

Se requiere un valor de texto.

El texto debe ser una cadena que represente un nombre de archivo.

Observaciones

`PreviewImage` es un elemento opcional.

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento Project (plantillas de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Especifica los archivos o directorios que se van a agregar al proyecto.

```
<VSTemplate <> TemplateContent> <Project>
```

Sintaxis

```
<Project
  File="MyProject.proj"
  TargetFileName="MyTargetProject.proj"
  ReplaceParameters="true/false">
  IgnoreProjectParameter="$myCustomParameter$"
  ...
</Project>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>File</code>	Atributo necesario. Especifica el nombre del archivo de proyecto en el archivo <i>.zip</i> de plantilla.
<code>ReplaceParameters</code>	Atributo opcional. Valor booleano que especifica si el archivo de proyecto tiene valores de parámetro que se deben reemplazar cuando se crea un proyecto a partir de la plantilla. El valor predeterminado es <code>false</code> .
<code>TargetFileName</code>	Atributo opcional. Especifica el nombre del archivo de proyecto cuando se crea un proyecto a partir de la plantilla.
<code>IgnoreProjectParameter</code>	Atributo opcional. Especifica si el proyecto se debe agregar a la solución actual. Si el valor del parámetro personalizado, <code>"\$myCustomParameter\$"</code> existe en el archivo de reemplazo de parámetros, el proyecto se crea pero no se agrega como parte de la solución abierta actualmente.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Carpeta	Elemento opcional. Especifica una carpeta que se va a agregar al proyecto.
ProjectItem	Elemento opcional. Especifica un archivo que se va a agregar a un proyecto.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateContent	Elemento necesario.

Observaciones

`Project` es un elemento secundario opcional de `TemplateContent`.

El `Project` elemento se utiliza para especificar un proyecto y, por lo tanto, solo es válido en plantillas de proyecto.

`Project` Los elementos pueden tener [elementos](#) secundarios `Folder` o `Folder` elementos secundarios [ProjectItem](#), pero no una mezcla de elementos secundarios y `ProjectItem` secundarios.

Visual Studiocambia automáticamente el nombre del archivo de proyecto en función del nombre especificado por el usuario en el cuadro de diálogo **Nuevo proyecto**. Utilice `TargetFileName` el atributo si desea proporcionar un nombre de archivo alternativo para los archivos de proyecto creados con la plantilla.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Elemento ProjectItem \(plantillas de proyecto de Visual Studio\)](#)
- [Elemento Folder \(plantillas de proyecto de Visual Studio\)](#)

Elemento ProjectCollection (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la organización y el contenido de las plantillas de varios proyectos.

<VSTemplate <> TemplateContent> <> ProjectCollection

Sintaxis

```
<ProjectCollection>
  <ProjectTemplateLink> ... </ProjectTemplateLink>
  <SolutionFolder> ... </SolutionFolder>
</ProjectCollection>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
ProjectTemplateLink	Elemento opcional. Especifica un proyecto en una plantilla de varios proyectos.
SolutionFolder	Elemento opcional. Agrupar los proyectos en plantillas de varios proyectos.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateContent	Elemento necesario. Especifica el contenido de la plantilla.

Observaciones

Las plantillas de varios proyectos actúan como contenedores de dos o más proyectos. El `ProjectCollection` elemento se utiliza para especificar los proyectos que se contienen en la plantilla. Para obtener más información sobre las plantillas de varios proyectos, consulte [Cómo: Crear plantillas de varios proyectos](#).

Ejemplo

En este ejemplo se muestra un archivo *.vstemplate* raíz de varios proyectos simple. En este ejemplo, la plantilla contiene dos proyectos, `My Windows Application` y `My Class Library`. El atributo `ProjectName` del elemento `ProjectTemplateLink` establece el nombre que Visual Studio debe asignar a este proyecto. Si `ProjectName` el atributo no existe, el nombre del archivo *.vstemplate* se utiliza como nombre del proyecto.

```
<VSTemplate Version="3.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-Project Template Sample</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="My Windows Application">
        WindowsApp\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="My Class Library">
        ClassLib\MyTemplate.vstemplate
      </ProjectTemplateLink>
    </ProjectCollection>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)
- [Cómo: Crear plantillas de varios proyectos](#)

Elemento ProjectItem (plantillas de proyecto de Visual Studio)

06/04/2020 • 7 minutes to read • [Edit Online](#)

Especifica un archivo que se incluye en la plantilla de proyecto.

NOTE

El `ProjectItem` elemento acepta atributos diferentes dependiendo de si la plantilla es para un proyecto o un elemento. En este `ProjectItem` tema se explica el elemento para las plantillas de proyecto. Para obtener una `ProjectItem` explicación del elemento para las plantillas de elemento, vea [ProjectItem Element \(Plantillasde elemento de Visual Studio\)](#).

```
<VSTemplate <> TemplateContent <> ProjectItem>>
```

Sintaxis

```
<ProjectItem
  TargetFileName="TargetFileName.ext"
  ReplaceParameters="true/false"
  OpenInEditor="true/false"
  OpenInWebBrowser="true/false"
  OpenInHelpBrowser="true/false"
  OpenOrder="Value">
  FileName.ext
</ProjectItem>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>TargetFileName</code>	Atributo opcional. Especifica el nombre y la ruta de acceso del elemento de proyecto cuando se crea un proyecto a partir de la plantilla. Este atributo es útil para crear una estructura de directorios diferente de la estructura de directorios en el archivo <i>.zip</i> de plantilla o para usar el reemplazo de parámetros para crear un nombre de elemento.
<code>ReplaceParameters</code>	Atributo opcional. Valor booleano que especifica si el elemento tiene valores de parámetro que se deben reemplazar cuando se crea un proyecto a partir de la plantilla. El valor predeterminado es <code>false</code> .

ATRIBUTO	DESCRIPCIÓN
<code>OpenInEditor</code>	Atributo opcional. Valor booleano que especifica si el elemento debe abrirse en su editor respectivo cuando se crea un proyecto a partir de la plantilla. Los <code>OpenInWebBrowser</code> <code>OpenInHelpBrowser</code> atributos y se omiten en un elemento con un <code>OpenInEditor</code> valor de <code>true</code> . El valor predeterminado es <code>false</code> .
<code>OpenInWebBrowser</code>	Atributo opcional. Valor booleano que especifica si el elemento debe abrirse el explorador web cuando se crea un proyecto a partir de la plantilla. Solo los archivos HTML y los archivos de texto que son locales para el proyecto se pueden abrir en el explorador web. Las direcciones URL externas no se pueden abrir con este atributo. El valor predeterminado es <code>false</code> .
<code>OpenInHelpBrowser</code>	Atributo opcional. Valor booleano que especifica si el elemento se debe abrir en el visor de ayuda cuando se crea un proyecto a partir de la plantilla. Solo los archivos HTML y los archivos de texto que son locales para el proyecto se pueden abrir en el explorador de ayuda. Las direcciones URL externas no se pueden abrir con este atributo. El valor predeterminado es <code>false</code> .
<code>OpenOrder</code>	Atributo opcional. Especifica un valor numérico que representa el orden en que se abrirán los elementos en sus respectivos editores. Todos los valores deben ser múltiplos de 10. Los elementos con valores más altos <code>OpenOrder</code> se abren primero.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>Project</code>	Especifica los archivos o directorios que se van a agregar al proyecto.

Valor de texto

Se requiere un valor de texto.

A `string` que representa el nombre o la ruta de acceso a un archivo en el archivo `.zip` de plantilla.

Observaciones

`ProjectItem` es un hijo `Project` opcional de .

El `TargetFileName` atributo se puede utilizar para crear una estructura de directorios diferente de la estructura de directorios en el archivo `.zip` de plantilla. Por ejemplo, si el archivo `MyFile.vb` existe en la raíz del archivo `.zip` de plantilla, pero desea que el archivo se coloque en un directorio denominado `CustomFiles` en todos los proyectos creados a partir de la plantilla, usaría el siguiente XML:

```
<ProjectItem TargetFileName="CustomFiles\MyFile.vb">MyFile.vb</ProjectItem>
```

El `TargetFileName` atributo también se puede utilizar para cambiar el nombre de los archivos que contienen caracteres internacionales en sus nombres de archivo. Por ejemplo, un archivo `.zip` de plantilla no puede contener nombres de archivo con caracteres Unicode, por lo que se debe cambiar el nombre del archivo antes de que se pueda comprimir en un archivo `.zip`. El `TargetFileName` atributo se puede utilizar para volver a establecer el nombre de archivo en el nombre de archivo Unicode original.

El `TargetFileName` atributo también se puede utilizar para cambiar el nombre de los archivos con parámetros. El siguiente procedimiento explica cómo cambiar el nombre del archivo `MyFile.vb`, que existe en el directorio raíz del archivo `.zip` de plantilla, a un nombre de archivo basado en el nombre del proyecto.

Para cambiar el nombre de los archivos con parámetros

1. Utilice el siguiente XML en el archivo `.vstemplate`:

```
<ProjectItem TargetFileName="$safeprojectname$.vb">MyFile.vb</ProjectItem>
```

2. Abra el archivo de proyecto (`.vbproj` para un Visual Basic proyecto) en un editor de texto o Visual Studio.
3. Busque la línea en el archivo de proyecto que sea similar al siguiente XML:

```
<Compile Include="MyFile.vb">
```

4. Reemplace la línea de código por el siguiente XML:

```
<Compile Include="$safeprojectname$.vb">
```

Cuando se crea un proyecto a partir de esta plantilla, el nombre de archivo se basará en el nombre que el usuario ha especificado en el cuadro de diálogo **Nuevo proyecto**, con todos los caracteres y espacios no seguros eliminados. Para obtener más información, consulte [Parámetros de plantilla](#) .

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem ReplaceParameters="true">Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)
- [Parámetros de plantilla](#)
- [ProjectItem \(Elemento, Plantillas de elementos de Visual Studio\)](#)

ProjectItem (Elemento, Plantillas de elementos de Visual Studio)

06/04/2020 • 4 minutes to read • [Edit Online](#)

Especifica un archivo que se incluye en la plantilla de elemento.

NOTE

El `ProjectItem` elemento acepta atributos diferentes dependiendo de si la plantilla es para un proyecto o un elemento. En este `ProjectItem` tema se explica el elemento para el elemento. Para obtener una `ProjectItem` explicación del elemento para las plantillas de proyecto, vea [Elemento ProjectItem \(plantillasde proyecto de Visual Studio\)](#).

```
<VSTemplate <> TemplateContent> <ProjectItem>
```

Sintaxis

```
<ProjectItem
  SubType="Form/Component/CustomControl/UserControl"
  CustomTool="string"
  ItemType="string"
  ReplaceParameters="true/false"
  TargetFileName="TargetFileName.ext">
  FileName.ext
</ProjectItem>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>SubType</code>	Atributo opcional. Especifica el subtipo de un elemento en una plantilla de elemento de varios archivos. Este valor se utiliza para Visual Studio determinar el editor que se usará para abrir el elemento.
<code>CustomTool</code>	Atributo opcional. Establece CustomTool para el elemento en el archivo de proyecto.
<code>ItemType</code>	Atributo opcional. Establece el ItemType para el elemento en el archivo de proyecto.

ATRIBUTO	DESCRIPCIÓN
<code>ReplaceParameters</code>	Atributo opcional. Valor booleano que especifica si el elemento tiene valores de parámetro que se deben reemplazar cuando se crea un proyecto a partir de la plantilla. El valor predeterminado es <code>false</code> .
<code>TargetFileName</code>	Atributo opcional. Especifica el nombre del elemento que se crea a partir de la plantilla. Este atributo es útil para usar el reemplazo de parámetros para crear un nombre de elemento.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateContent</code>	Especifica el contenido de la plantilla.

Valor de texto

Se requiere un valor de texto.

A `string` que representa el nombre de un archivo en el archivo `.zip` de plantilla.

Observaciones

`ProjectItem` es un hijo `TemplateContent` opcional de .

El `TargetFileName` atributo se puede utilizar para cambiar el nombre de los archivos con parámetros. Por ejemplo, si el archivo `MyFile.vb` existe en el directorio raíz del archivo `.zip` de plantilla, pero desea que el archivo se asigne en función del nombre de archivo proporcionado por el usuario en el cuadro de diálogo **Agregar nuevo elemento**, usaría el siguiente XML:

```
<ProjectItem TargetFileName="$fileinputname$.vb">MyFile.vb</ProjectItem>
```

Cuando se crea un elemento a partir de esta plantilla, el nombre de archivo se basará en el nombre que el usuario ha especificado en el cuadro de diálogo **Agregar nuevo elemento**. Esto es útil al crear plantillas de elementos de varios archivos. Para obtener más información, consulte [Cómo: Crear plantillas de elementos de varios archivos](#) y [Parámetros de plantilla](#).

Ejemplo

En el ejemplo siguiente se muestran los Visual C# metadatos de la plantilla de elemento estándar para una clase.

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <DefaultName>MyClass.cs</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem ReplaceParameters="true">MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Cómo: Crear plantillas de elementos de varios archivos](#)
- [Parámetros de plantilla](#)

Elemento ProjectSubType (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Clasifica la plantilla en una subcategoría del valor `ProjectType` especificado en el elemento.

`<VSTemplate <> TemplateData> <> ProjectSubType`

Sintaxis

```
<ProjectSubType> SubType </ProjectSubType>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

Este valor especifica la subcategoría de la plantilla.

Observaciones

`ProjectSubType` es un elemento secundario opcional de `TemplateData`.

El `ProjectSubType` elemento proporciona una subcategoría para el `ProjectType` elemento. Este valor puede incluir:

- `SmartDevice-NETCFv1`: especifica que la .NET Compact Framework plantilla tiene como destino la versión 1.0.
- `SmartDevice-NETCFv2`: especifica que la .NET Compact Framework plantilla tiene como destino la versión 2.0.

Si una plantilla `ProjectType` contiene un `Web` elemento `ProjectSubType` con un valor de , el elemento especifica el lenguaje de programación de la plantilla. Este elemento puede tener los siguientes valores:

- `CSharp`: especifica que la Visual C# plantilla crea un proyecto web o un elemento.
- `VisualBasic`: especifica que la Visual Basic plantilla crea un proyecto web o un elemento.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos .NET Compact Framework de una plantilla de proyecto para una aplicación de dispositivo destinada a la versión 2.0.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic device template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <ProjectSubType>SmartDevice-NETCFv2</ProjectSubType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)
- [Elemento ProjectType \(plantillas de Visual Studio\)](#)

ProjectTemplateLink (elemento, plantillas de Visual Studio)

06/04/2020 • 4 minutes to read • [Edit Online](#)

Especifica la ruta de acceso al archivo *.vstemplate* de un proyecto en una plantilla de varios proyectos.

```
<VSTemplate <> <TemplateContent> <ProjectCollection> ProjectTemplateLink <-o-> <TemplateContent> >
ProjectCollection <> <SolutionFolder> <ProjectTemplateLink>
```

Sintaxis

```
<ProjectTemplateLink ProjectName="Name">
    PathToTemplateFile
</ProjectTemplateLink>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>ProjectName</code>	Atributo opcional. Especifica el nombre de cada proyecto individual en una plantilla de varios proyectos. El cuadro de diálogo Nuevo proyecto no puede asignar nombres a proyectos individuales.
<code>CopyParameters</code>	Permite que todas las variables de la plantilla de grupo principal se copien en cada una de las plantillas vinculadas. Los parámetros de las plantillas vinculadas tienen un prefijo <code>"\$ext_*\$"</code>. Por ejemplo, si en la <code>\$projectname\$</code> plantilla de grupo primario el parámetro tiene un valor ExampleProject1, <code>\$ext_projectname\$</code> cuando la plantilla vinculada <code>\$projectname\$</code> obtiene su turno para ejecutarse, adquiere un parámetro, que es una copia del parámetro de la plantilla de grupo primario. Esto permite que las plantillas vinculadas compartan algunos parámetros comunes, para que estos solo se tengan que crear en la plantilla del grupo primario. Este atributo es opcional y se establece de forma automática en <code>false</code> cuando no se incluye. Apareció por primera vez en Visual Studio 2013 Update 2. Para hacer referencia a la versión correcta del producto, vea Ensamblados de referencia entregados en Visual Studio 2013 SDK Update 2.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
ProjectCollection	Especifica la organización y el contenido de las plantillas de varios proyectos.
SolutionFolder	Agrupar los proyectos en plantillas de varios proyectos.

Valor de texto

Se requiere un valor de texto.

Este texto especifica la ruta de acceso al archivo *.vstemplate* de la plantilla.

Observaciones

Las plantillas de varios proyectos actúan como contenedores de dos o más proyectos. El `ProjectTemplateLink` elemento se utiliza para especificar la ubicación del archivo *.vstemplate* para uno de los proyectos de la plantilla. El archivo *.vstemplate* de una plantilla `ProjectTemplateLink` de varios proyectos contiene un elemento para cada proyecto de la plantilla. Para obtener más información sobre las plantillas de varios proyectos, consulte [Cómo: Crear plantillas de varios proyectos](#).

Ejemplo

En este ejemplo se muestra un archivo *.vstemplate* raíz de varios proyectos simple. En este ejemplo, la plantilla contiene dos proyectos, `My Windows Application` y `My Class Library`. El atributo `ProjectName` del elemento `ProjectTemplateLink` establece el nombre que Visual Studio debe asignar a este proyecto. Si `ProjectName` el atributo no existe, el nombre del archivo *.vstemplate* se utiliza como nombre del proyecto.

```
<VSTemplate Version="3.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-Project Template Sample</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="My Windows Application">
        WindowsApp\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="My Class Library" CopyParameters="true">
        ClassLib\MyTemplate.vstemplate
      </ProjectTemplateLink>
    </ProjectCollection>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

- [Cómo: Crear plantillas de varios proyectos](#)

Elemento ProjectType (plantillas de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Categoriza la plantilla de proyecto para que aparezca bajo el grupo especificado en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**.

WARNING

Las plantillas de proyecto se admiten para C++ a partir de Visual Studio 2012. No se admiten para C++ en Visual Studio 2010 y versiones anteriores.

```
<VSTemplate <> TemplateData> <projectType>
```

Sintaxis

```
<ProjectType> CSharp/VisualBasic/VC/Web </ProjectType>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

Este valor especifica el tipo de proyecto que creará la plantilla y debe contener uno de los siguientes valores:

- `CSharp`: especifica que la Visual C# plantilla crea un proyecto o elemento.
- `VisualBasic`: especifica que la Visual Basic plantilla crea un proyecto o elemento.
- `Web`: especifica que la plantilla crea un proyecto web o un elemento. Si `ProjectType` el elemento contiene este valor, el idioma del proyecto o elemento se define en el [elemento ProjectSubType \(plantillas de Visual Studio\)](#).

Observaciones

`ProjectType` es un elemento secundario obligatorio de `TemplateData`.

El valor `ProjectType` del elemento especifica dónde se encuentra la plantilla en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**. Por ejemplo, una `ProjectType` plantilla `CSharp` con un valor de aparece en el nodo **Visual C**. en el cuadro de diálogo **Nuevo proyecto**.

Se puede especificar un subtipo de plantilla mediante el elemento [ProjectSubType](#).

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)
- [Elemento ProjectSubType \(plantillas de Visual Studio\)](#)

Elemento PromptForSaveOnCreation (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si se solicita al usuario una ubicación de guardado del proyecto a través del cuadro de diálogo **Nuevo proyecto** al crear un proyecto. Si este elemento `true` se establece en , se solicita al usuario una ubicación de guardado. Si `false` , no se les solicita (es decir, se crea un proyecto temporal).

```
\<VSTemplate>
\<TemplateData>
\<PromptForSaveOnCreation>
```

Sintaxis

```
<PromptForSaveOnCreation> true/false </PromptForSaveOnCreation>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser `true` o , lo que indica que se solicitará al usuario una ubicación de guardado al crear un nuevo proyecto.

Observaciones

`PromptForSaveOnCreation` es un elemento opcional. El valor predeterminado es `false` .

Los proyectos temporales son proyectos que se pueden crear y modificar sin guardar el contenido de ese proyecto en el disco.

Ejemplo

En el ejemplo siguiente `PromptForSaveOnCreation` se `false` establece el valor de igual a , que especifica permitir que el proyecto se cree como un proyecto temporal.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <PromptForSaveOnCreation>>false</PromptForSaveOnCreation>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento ProvideDefaultName (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si Visual Studio el sistema de proyectos generará un nombre predeterminado para la plantilla en el cuadro de diálogo **Agregar nuevo elemento** o **Nuevo proyecto**.

<VSTemplate <> TemplateData> <> ProvideDefaultName

Sintaxis

```
<ProvideDefaultName> true/false </ProvideDefaultName>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o , que indica si se debe generar o no un nombre predeterminado para la plantilla en el **agregar nuevo elemento** o nuevo **proyecto** cuadro de diálogo.

Observaciones

`ProvideDefaultName` es un elemento opcional. El valor predeterminado es `true`.

Si `ProvideDefaultName` el `false` elemento es , los cuadros **Nombre** de los `<Enter_name>` cuadros de diálogo **Agregar nuevo elemento** y **Nuevo proyecto** contienen el valor .

Utilice el elemento [DefaultName](#) para especificar el nombre predeterminado del proyecto o elemento en los cuadros de diálogo **Agregar nuevo elemento** y **Nuevo proyecto**. Cuando el valor `ProvideDefaultName` del `true` elemento es `DefaultName` , la omisión del elemento para los proyectos rellena el cuadro de diálogo con el nombre de la plantilla, es decir, el valor del elemento [Name](#).

Ejemplo

En el ejemplo `ProvideDefaultName` de `false` código siguiente se establece el elemento en .

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <ProvideDefaultName>false</ProvideDefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem>MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento Reference (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la referencia de ensamblado para agregar cuando el elemento se agrega a un proyecto.

```
<VSTemplate <> TemplateContent < <> referencias>>
```

Sintaxis

```
<Reference>
  <Assembly> ... </Assembly>
</Reference>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
ensamblaje	Elemento necesario. Especifica información sobre un ensamblado, que la plantilla utiliza para agregar una referencia de ese ensamblado a los proyectos. Debe haber <code>Assembly</code> un elemento <code>Reference</code> en cada elemento.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Referencias.	Agrupar las referencias de ensamblado que la plantilla agrega a los proyectos.

Observaciones

`Reference` es un elemento secundario obligatorio de `References`.

Los `Reference` `References` elementos y solo se pueden utilizar `Type` en archivos `Item` `.vstemplate` que tengan un valor de atributo de .

Ejemplo

En el ejemplo `TemplateContent` siguiente se muestra el elemento de una plantilla de elemento. Este XML agrega referencias a los ensamblados `System.dll` y `System.Data.dll`.

```
<TemplateContent>
  <References>
    <Reference>
      <Assembly>
        System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
    <Reference>
      <Assembly>
        System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
  </References>
  ...
</TemplateContent>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)

Elemento References (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Agrupar las referencias de ensamblado que la plantilla agrega a los proyectos.

```
<VSTemplate <> TemplateContent> <referencias>
```

Sintaxis

```
<References>
  <Reference>... </Reference>
  <Reference>... </Reference>
  ...
</References>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Referencia	Elemento necesario. Especifica la referencia de ensamblado para agregar cuando el elemento se agrega a un proyecto. Debe haber uno <code>Reference</code> o más <code>References</code> elementos en un elemento.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateContent	Especifica el contenido de la plantilla.

Observaciones

`References` es un elemento secundario opcional de `TemplateContent`.

Los `Reference` `References` elementos y solo se pueden utilizar `Type` en archivos `Item.vstemplate` que tengan un valor de atributo de .

Ejemplo

En el ejemplo `TemplateContent` siguiente se muestra el elemento de una plantilla de elemento. Este XML agrega referencias a los ensamblados *System.dll* y *System.Data.dll*.

```
<TemplateContent>
  <References>
    <Reference>
      <Assembly>
        System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
    <Reference>
      <Assembly>
        System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089
      </Assembly>
    </Reference>
  </References>
  ...
</TemplateContent>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

Elemento RequiredFrameworkVersion (plantillas de Visual Studio)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Especifica la versión mínima de .NET Framework que requiere la plantilla. Hace que el menú desplegable **Versión** de marco de trabajo de destino se muestre en el cuadro de diálogo **Nuevo proyecto**. El

`RequiredFrameworkVersion` elemento también determina el valor más bajo disponible en la lista desplegable.

IMPORTANT

A partir de Visual Studio 2017 versión 15.6, la lista desplegable **Versión** de marco de trabajo de destino ya no es un filtro para las plantillas mostradas en la sección **Plantillas** del cuadro de diálogo **Nuevo proyecto**. En su lugar, la lista desplegable funciona como un selector de marco de trabajo para la plantilla seleccionada.

<VSTemplate <> TemplateData> <>

Sintaxis

```
<RequiredFrameworkVersion> .... </RequiredFrameworkVersion>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ser el número de versión mínimo de .NET Framework que se requiere para la plantilla.

Observaciones

`RequiredFrameworkVersion` es un elemento opcional. Use este elemento solo si la plantilla admite una versión mínima específica (y versiones posteriores, si las hay) de .NET Framework. Si especifica `RequiredFrameworkVersion`

el elemento y la plantilla no admite una versión mínima específica de .NET Framework, la lista desplegable **Versión de marco de destino** se muestra cuando no es aplicable.

Ejemplo

En el ejemplo siguiente se Visual C# muestran los metadatos de una plantilla de clase estándar.

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class template.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <RequiredFrameworkVersion>3.0</RequiredFrameworkVersion>
    <MaxFrameworkVersion>4.7.1</MaxFrameworkVersion>
    <DefaultName>MyClass</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem>MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

En este ejemplo, la versión mínima de .NET Framework que `RequiredFrameworkVersion` requiere la plantilla, representada por , es 3.0. Un proyecto creado con esta plantilla puede tener como destino versiones de .NET Framework a partir de 3.0.

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Crear plantillas de proyecto y elemento](#)
- [Información general sobre destinos de Framework](#)

Elemento RequiredPlatformVersion (plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la versión mínima del sistema operativo que la plantilla de proyecto requiere para funcionar correctamente. Este elemento se utiliza para Tienda Windows 8.x las plantillas de proyecto que crean aplicaciones.

El `RequiredPlatformVersion` valor se compara directamente con la versión del sistema operativo. Si `RequiredPlatformVersion` la versión del sistema operativo es superior a la del sistema operativo, la plantilla no aparece en el cuadro de diálogo **Nuevo proyecto**. Para especificar una Windows 8 plantilla para `RequiredPlatformVersion` o superior, establezca en 6.2.0. Para especificar una Windows 8.1 plantilla para `RequiredPlatformVersion` o superior, establezca en 6.3.0.

Las plantillas `RequiredPlatformVersion` que especifican el Tienda Windows 8.x número 8 son compatibles con las plantillas de cliente anteriores.

VSTemplate TemplateData TargetPlatformName RequiredPlatformVersion

Sintaxis

```
<RequiredPlatformVersion> OperatingSystem </RequiredPlatformVersion>
```

Atributos y elementos

Ninguno.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplatePlatformName	Especifica la plataforma a la que está orientada la plantilla del proyecto.

Valor de texto

Se requiere un valor de texto.

Observaciones

Este texto especifica la versión mínima del sistema operativo requerida por la plantilla.

Ejemplo

Este ejemplo especifica que la plantilla del proyecto está orientada a Windows 8 o posterior.

```
<VSTemplate Type="Project" Version="3.0.0" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <TargetPlatformName>Windows</TargetPlatformName>
    <RequiredPlatformVersion>6.3.0</RequiredPlatformVersion>

  </TemplateData>
  <TemplateContent>

  </TemplateContent>
</VSTemplate>
```

Vea también

- [Elemento TargetPlatformName \(plantillas de Visual Studio\)](#)
- [Crear plantillas de proyecto y elemento](#)
- [Referencia de esquema de plantilla de Visual Studio](#)

SDKReference (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica que la plantilla del elemento usa una referencia de SDK.

Sintaxis

```
<VSTemplate>
  <TemplateContent>
    <References>
      <Reference>
        <SDKReference>SDKName</SDKReference>
      </Reference>
    </References>
  </TemplateContent>
</VSTemplate>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Referencia	Especifica la referencia de ensamblado para agregar cuando el elemento se agrega a un proyecto.

Valor de texto

Se requiere un valor de texto.

Observaciones

Este texto especifica la referencia SDK que se debe agregar a un proyecto cuando se crea una instancia de la plantilla de elemento.

```
<VSTemplate Version="3.0.0" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005" Type="Item">
  <TemplateData> . . . </TemplateData>
  <TemplateContent>
    <References>
      <Reference>
        <SDKReference>Microsoft Visual C++ Runtime Package, Version=11.0.0.0</SDKReference>
      </Reference>
    </References>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [References \(Elemento, Plantillas de Visual Studio\)](#)

- [Reference \(Elemento, Plantillas de Visual Studio\)](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [Referencia de esquema de plantillas de Visual Studio](#)

Elemento ShowByDefault (Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Si `false`, especifica que la plantilla solo se mostrará en el `TemplateGroupID` especificado.

`<VSTemplate <> TemplateData> <> ShowByDefault`

Sintaxis

```
<ShowByDefault> true/false </ShowByDefault>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ser `true` o `false`. Si es `true`, especifica que la plantilla se muestra para todos los tipos de proyecto. Si es `false`, la plantilla solo se muestra en el `TemplateGroupID` especificado.

Observaciones

`ShowByDefault` es un elemento opcional. El valor predeterminado es `true`.

Ejemplo

En el siguiente ejemplo se muestran los metadatos de una plantilla de Visual C#:

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic template</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <ShowByDefault>>false</ShowByDefault>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [Referencia de esquema de plantillas de Visual Studio](#)
- [TemplateGroupID \(Elemento, Plantillas de Visual Studio\)](#)

SolutionFolder (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Agrupar los proyectos en plantillas de varios proyectos.

<VSTemplate <> TemplateContent <> <ProjectCollection> SolutionFolder>

Sintaxis

```
<SolutionFolder Name="DirectoryName">
  ...
</SolutionFolder>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Name</code>	Atributo necesario. Nombre de la carpeta de soluciones.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
<code>ProjectTemplateLink</code>	Elemento opcional. Especifica la ruta de acceso al archivo .vstemplate de un proyecto en una plantilla de varios proyectos.
<code>SolutionFolder</code>	Elemento opcional. Agrupar los proyectos en plantillas de varios proyectos.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
<code>ProjectCollection</code>	Especifica la organización y el contenido de las plantillas de varios proyectos.
<code>SolutionFolder</code>	Agrupar los proyectos en plantillas de varios proyectos.

Observaciones

Las plantillas de varios proyectos actúan como contenedores de dos o más proyectos. El elemento `SolutionFolder` se usa para organizar los proyectos de la plantilla en grupos. Las carpetas especificadas por los elementos

`SolutionFolder` se crean como carpetas de soluciones en el proyecto en Visual Studio. Para obtener más información sobre las plantillas de varios proyectos, vea [Cómo: Crear plantillas de varios proyectos](#).

Ejemplo

Este ejemplo usa el elemento `SolutionFolder` para dividir la plantilla de varios proyectos en dos grupos, `Math Classes` y `Graphics Classes`. La plantilla contiene cuatro proyectos, dos de los cuales se colocan en cada carpeta de soluciones.

```
<VSTemplate Version="3.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Multi-Project Template Sample</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>VisualBasic</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <SolutionFolder Name="Math Classes">
        <ProjectTemplateLink ProjectName="MathClassLib1">
          MathClassLib1\MyTemplate.vstemplate
        </ProjectTemplateLink>
        <ProjectTemplateLink ProjectName="MathClassLib2">
          MathClassLib2\MyTemplate.vstemplate
        </ProjectTemplateLink>
      </SolutionFolder>
      <SolutionFolder Name="Graphics Classes">
        <ProjectTemplateLink ProjectName="GraphicsClassLib1">
          GraphicsClassLib1\MyTemplate.vstemplate
        </ProjectTemplateLink>
        <ProjectTemplateLink ProjectName="GraphicsClassLib2">
          GraphicsClassLib2\MyTemplate.vstemplate
        </ProjectTemplateLink>
      </SolutionFolder>
    </ProjectCollection>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [Cómo: Crear plantillas de varios proyectos](#)

SortOrder (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica un valor que se utiliza para organizar la plantilla, entre otras plantillas de la misma categoría, como aparece en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**.

<VSTemplate <> TemplateData> <>

Sintaxis

```
<SortOrder> ... </SortOrder>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

Un `integer` que representa el valor del criterio de ordenación.

Observaciones

`SortOrder` es un elemento opcional. El valor predeterminado es 100 y todos los valores deben ser múltiplos de 10.

El `SortOrder` elemento se omite para las plantillas creadas por el usuario. Todas las plantillas creadas por el usuario se ordenan alfabéticamente.

Las plantillas que tienen valores de criterio de ordenación bajos aparecen en el cuadro de diálogo **Nuevo proyecto** o **Nuevo elemento** de adición antes de las plantillas que tienen valores de criterio de ordenación altos.

Ejemplo

En el ejemplo siguiente se Visual C# muestran los metadatos de una plantilla de clase estándar.

```
<VSTemplate Type="Item" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyClass</Name>
    <Description>My custom C# class template.</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
    <SortOrder>290</SortOrder>
    <DefaultName>MyClass</DefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectItem>MyClass.cs</ProjectItem>
  </TemplateContent>
</VSTemplate>
```

En este ejemplo, el `SortOrder` elemento es relativamente alto. Es probable que Visual C# otras plantillas `SortOrder` de elemento `290` tengan un valor inferior y aparecerán antes de esta plantilla en el cuadro de diálogo **Nuevo elemento**.

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

SupportsCodeSeparation (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si la casilla **Colocar código en un archivo independiente** está habilitada en el cuadro de diálogo **Agregar nuevo elemento**.

```
<VSTemplate <> TemplateData> <SupportsCodeSeparation>
```

Sintaxis

```
<SupportsCodeSeparation> true/false </SupportsCodeSeparation>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Categoriza la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o , indicando si la casilla de verificación **Colocar código en archivo independiente** está habilitada en el cuadro de diálogo **Agregar nuevo elemento**.

Observaciones

`SupportsCodeSeparation` es un elemento opcional. El valor predeterminado es `false`.

El `SupportsCodeSeparation` elemento solo está disponible para plantillas de elementos web.

La separación de código, o el modelo de página de código subyacente, le permite mantener el marcado en un archivo y el código de programación en otro archivo. ASP.NET y otros lenguajes .NET usan este modelo.

Ejemplo

En el ejemplo siguiente se especifica que se muestre la opción Colocar código en un archivo independiente.

```
<VSTemplate Version="3.0.0" Type="Project"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">>
  <TemplateData>
    <Name>MyWebProjecStarterKit</Name>
    <Description>A simple Web template</Description>
    <Icon>icon.ico</Icon>
    <ProjectType>Web</ProjectType>
    <ProjectSubType>CSharp</ProjectSubType>
    <DefaultName>WebSite</DefaultName>
    <SupportsCodeSeparation>true</SupportsCodeSeparation>
  </TemplateData>
  <TemplateContent>
    <Project File="WebApplication.webproj">
      <ProjectItem>icon.ico</ProjectItem>
      <ProjectItem OpenInEditor="true">Default.aspx</ProjectItem>
      <ProjectItem>Default.aspx.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

SupportsLanguageDropDown (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si la plantilla de elemento web es idéntica para varios idiomas y si la opción **Idioma** está habilitada en el cuadro de diálogo **Agregar nuevo elemento**.

```
<VSTemplate <> TemplateData> <SupportsLanguageDropDown>
```

Sintaxis

```
<SupportsLanguageDropDown> true/false </SupportsLanguageDropDown>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe `true` `false` ser o , que indica si la opción **Idioma** está disponible en el cuadro de diálogo **Agregar nuevo elemento**.

Observaciones

`SupportsLanguageDropDown` es un elemento opcional. El valor predeterminado es `false`.

El `SupportsLanguageDropDown` elemento solo está disponible para plantillas de elementos web.

Si el valor de este `true` elemento se establece en , la plantilla de elemento es idéntica para todos los lenguajes de programación y la opción **Idioma** está habilitada en el cuadro de diálogo **Agregar nuevo elemento**. Esta opción le permite elegir el lenguaje de programación del nuevo elemento que desea crear a partir de la plantilla.

Ejemplo

En el ejemplo siguiente se especifica que se muestre la opción desplegable **Idioma**.

```
<VSTemplate Version="3.0.0" Type="Project"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">>
  <TemplateData>
    <Name>MyWebProjecStarterKit</Name>
    <Description>A simple Web template</Description>
    <Icon>icon.ico</Icon>
    <ProjectType>Web</ProjectType>
    <ProjectSubType>CSharp</ProjectSubType>
    <DefaultName>WebSite</DefaultName>
    <SupportsLanguageDropDown>true</SupportsLanguageDropDown>
  </TemplateData>
  <TemplateContent>
    <Project File="WebApplication.webproj">
      <ProjectItem>icon.ico</ProjectItem>
      <ProjectItem OpenInEditor="true">Default.aspx</ProjectItem>
      <ProjectItem>Default.aspx.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

SupportsMasterPage (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica si la casilla de verificación **Seleccionar página maestra** está habilitada en el cuadro de diálogo Agregar nuevo **elemento**.

<VSTemplate <> TemplateData> <SupportsMasterPage>

Sintaxis

```
<SupportsMasterPage> true/false </SupportsMasterPage>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Especifica los datos que clasifican la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto debe ☐ true ☐ false ser o , indicando si la casilla **Seleccionar página maestra** está habilitada en el cuadro de diálogo Agregar nuevo **elemento**.

Observaciones

`SupportsMasterPage` es un elemento opcional. El valor predeterminado es `false`.

El `SupportsMasterPage` elemento solo está disponible para plantillas de elementos web.

Ejemplo

En el ejemplo siguiente se muestran los metadatos de un proyecto Web que incluye compatibilidad con una página maestra.

```
<VSTemplate Version="3.0.0" Type="Project"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">>
  <TemplateData>
    <Name>MyWebProjecStarterKit</Name>
    <Description>A simple Web template</Description>
    <Icon>icon.ico</Icon>
    <ProjectType>Web</ProjectType>
    <ProjectSubType>CSharp</ProjectSubType>
    <DefaultName>WebSite</DefaultName>
    <SupportsMasterPage>true</SupportsMasterPage>
  </TemplateData>
  <TemplateContent>
    <Project File="WebApplication.webproj">
      <ProjectItem>icon.ico</ProjectItem>
      <ProjectItem OpenInEditor="true">Default.aspx</ProjectItem>
      <ProjectItem>Default.aspx.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

TargetPlatformName (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica la plataforma a la que está orientada la plantilla del proyecto. Este elemento se usa para especificar que se usará una plantilla de proyecto para crear aplicaciones de Tienda Windows 8.x .

Sintaxis

```
<VSTemplate>
  <TemplateData>
    <TargetPlatformName> OperatingSystem</TargetPlatformName>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
RequiredPlatformVersion	Especifica la versión del sistema operativo al que está destinada la plantilla del proyecto.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

Observaciones

El texto debe ser **Windows**.

Ejemplo

Este ejemplo especifica que la plantilla del proyecto está orientada a Windows 8 o posterior.

```
<VSTemplate Type="Project" Version="3.0.0" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <TargetPlatformName>Windows</TargetPlatformName>
    <RequiredPlatformVersion>8</RequiredPlatformVersion>
  </TemplateData>
  <TemplateContent> </TemplateContent>
</VSTemplate>
```

Vea también

- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [Referencia de esquema de plantillas de Visual Studio](#)

TemplateContent (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica el contenido de la plantilla.

Jerarquía de elementos:

```
<VSTemplate>
  <TemplateContent>
```

Sintaxis

```
<TemplateContent>
  ...
</TemplateContent>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
BuildOnLoad	Especifica si se va a compilar la solución cuando se crea un proyecto a partir de la plantilla.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
ProjectCollection	Elemento opcional. Especifica la organización y el contenido de las plantillas de varios proyectos.
Project	Elemento opcional. Especifica los archivos o directorios que se van a agregar al proyecto.
Referencias.	Elemento opcional. Especifica las referencias de ensamblado necesarias para una plantilla de elemento.
ProjectItem	Elemento opcional. Especifica un archivo contenido en la plantilla.

ELEMENTO	DESCRIPCIÓN
CustomParameters	Elemento opcional. Especifica los parámetros personalizados que se van a utilizar cuando se crea un proyecto o elemento a partir de la plantilla.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
VSTemplate	Elemento necesario. Contiene todos los metadatos de la plantilla de proyecto, la plantilla de elemento o el kit de inicio.

Observaciones

`TemplateContent` es un elemento requerido.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

TemplateData (Elemento, Plantillas de Visual Studio)

06/04/2020 • 6 minutes to read • [Edit Online](#)

Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo **Nuevo proyecto** o **Agregar nuevo elemento**.

<VSTemplate <>> TemplateData

Sintaxis

```
<TemplateData>
  <Name> ... </Name>
  <Description> ... </Description>
  <Icon> ... </Icon>
  <ProjectType> ... </ProjectType>
  ...
</TemplateData>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Nombre	Elemento necesario. Especifica el nombre de la plantilla tal como aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .
Descripción	Elemento necesario. Especifica la descripción de la plantilla tal como aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .
Icono	Elemento necesario. Especifica la ruta de acceso y el nombre de archivo del archivo de imagen que actúa como icono, que aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento , para la plantilla.

ELEMENTO	DESCRIPCIÓN
ProjectType	Elemento necesario. Categoriza la plantilla de proyecto para que aparezca bajo el grupo especificado en el cuadro de diálogo Nuevo proyecto.
ProjectSubType	Elemento opcional. Clasifica la plantilla de proyecto para que aparezca en la subcategoría especificada en el cuadro de diálogo Nuevo proyecto.
TemplateID	Elemento opcional. Especifica el identificador de plantilla.
TemplateGroupID	Elemento opcional. Especifica el identificador del grupo de plantillas.
SortOrder	Elemento opcional. Especifica un valor que se utiliza para organizar la plantilla, entre otras plantillas de la misma categoría, como aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento.
CreateNewFolder	Elemento opcional. Especifica si se crea una carpeta contenedora en la creación de instancias del proyecto.
DefaultName	Elemento opcional. Especifica el nombre que generará el sistema de proyectos de Visual Studio para el proyecto o elemento cuando se crea.
ProvideDefaultName	Elemento opcional. Especifica si el sistema de proyectos de Visual Studio generará el nombre predeterminado para un proyecto o elemento cuando se crea.
PromptForSaveOnCreation	Elemento opcional. Especifica si el proyecto se puede crear como un proyecto temporal (solo Visual Studio 2017).
EnableLocationBrowseButton	Elemento opcional. Especifica si el botón Examinar está disponible en el cuadro de diálogo Nuevo proyecto, para que los usuarios puedan modificar fácilmente el directorio predeterminado donde se guarda un nuevo proyecto.

ELEMENTO	DESCRIPCIÓN
Oculto	Elemento opcional. Especifica si la plantilla aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento.
NumberOfParentCategoriesToRollUp	Elemento opcional. Especifica el número de categorías primarias que mostrarán la plantilla en el cuadro de diálogo Nuevo proyecto.
LocationFieldMRUPrefix	Elemento opcional.
LocationField	Elemento opcional. Especifica si el cuadro de texto Ubicación del cuadro de diálogo Nuevo proyecto está habilitado, deshabilitado u oculto para la plantilla de proyecto.
RequiredFrameworkVersion	Elemento opcional. Use este elemento si la plantilla solo admite una versión mínima específica y versiones posteriores, si las hay, de .NET Framework.
SupportsMasterPage	Elemento opcional. Especifica si la plantilla admite una página maestra para proyectos web.
SupportsCodeSeparation	Elemento opcional. Especifica si la plantilla admite la separación de código o el modelo de página de código subyacente para proyectos web.
SupportsLanguageDropDown	Elemento opcional. Especifica si la plantilla es idéntica para varios idiomas y si la opción Idioma está disponible en el cuadro de diálogo Nuevo proyecto.
TargetPlatformName	Elemento opcional. Especifica la plataforma a la que está orientada la plantilla del proyecto. Este elemento especifica que se usa Tienda Windows 8.x una plantilla de proyecto para crear aplicaciones.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
VSTemplate	Elemento necesario. Contiene todos los metadatos de la plantilla de proyecto, la plantilla de elemento o el kit de inicio.

Observaciones

`TemplateData` es un elemento requerido.

Si no incluye un elemento opcional, se utiliza el valor predeterminado para ese elemento.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

TemplateGroupID (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica en qué tipo de proyecto se mostrará una plantilla de elemento. Este elemento es significativo cuando [ShowByDefault \(Plantillas de Visual Studio\)](#) se establece en `false`. Cuando [ShowByDefault \(Plantillas de Visual Studio\)](#) se establece en `true`, a continuación, una plantilla de elemento está disponible en todos los tipos de proyecto.

```
<VSTemplate <> TemplateData> <> TemplateGroupID
```

Sintaxis

```
<TemplateGroupID> ... </TemplateGroupID>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

Se requiere un valor de texto.

El texto especifica un identificador para una categoría de plantillas de elementos.

Observaciones

`TemplateGroupID` es un elemento.

El valor `TemplateGroupID` del elemento se utiliza junto con el registro del sistema del proyecto (HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, <>), para filtrar las plantillas que aparecen en el cuadro de diálogo **Agregar nuevo elemento**.

VALOR DE VISUAL C++	SIGNIFICADO

VALOR DE VISUAL C ++	SIGNIFICADO
VC-Native	Se usa en objetos nativos. También el valor predeterminado si no se puede determinar un tipo de proyecto.
VC-Managed	Utilizado para proyectos (/ clr) administrados
VC-Windows	Utilizado para todos los proyectos destinados a la plataforma de Windows (nativo/administrado/almacén)
WinRT-Native-UAP	Utilizado para proyectos de la Tienda Windows 10
CodeSharing-Native	Utilizado para proyectos de elementos compartidos
WinRT-Native-6.3	Utilizado para proyectos de la Tienda Windows 8.1
WinRT-Native-Phone-6.3	Utilizado para proyectos de Windows Phone 8.1
WinRT-Native	Utilizado para proyectos de la Tienda Windows 8.0
VC-Android	Utilizado para proyectos Android

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)

TemplateID (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica un identificador para una plantilla de elemento que se clasifica en un grupo de plantillas de elemento por el [TemplateGroupID](#) elemento.

<VSTemplate < > TemplateData> <TemplateID>

Sintaxis

```
<TemplateID> ... </TemplateID>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
TemplateData	Elemento necesario. Clasifica la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .

Valor de texto

A `string` que representa un identificador para una plantilla de elemento que `TemplateGroupID` se clasifica en un grupo de plantillas de elemento por el elemento.

Observaciones

`TemplateID` es un elemento opcional.

Si un archivo .vstemplate `TemplateID` omite el elemento, el elemento [Name](#) se utiliza como identificador de la plantilla.

El valor `TemplateID` del elemento se utiliza junto con el registro del sistema de proyectos\ (HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 11,0, Projects) para filtrar las plantillas que aparecen en el cuadro de diálogo **Agregar nuevo elemento**.

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)

- [Crear plantillas para proyectos y elementos en Visual Studio](#)

Elemento VSTemplate (Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Contiene todos los metadatos sobre la plantilla de proyecto, la plantilla de elemento o el kit de inicio.

Sintaxis

```
<VSTemplate Type="TemplateType" Version="x.x.x">
  <TemplateData>    </TemplateData>
  <TemplateContent>  </TemplateContent>
  ...
</VSTemplate>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Type</code>	Identifica la plantilla como una plantilla de proyecto o una plantilla de elemento. Este atributo puede tener <code>Project</code> <code>Item</code> un valor de o .
<code>Version</code>	Especifica un número de versión para la plantilla. Plantillas Visual Studio 2010 y Visual Studio 2012 tienen <code>Version</code> un <code>3.0.0</code> valor de atributo de .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
<code>TemplateData</code>	Elemento necesario. Especifica los datos que clasifican la plantilla y define cómo se muestra en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento .
<code>TemplateContent</code>	Elemento necesario. Especifica el contenido de la plantilla.
<code>WizardExtension</code>	Elemento opcional.
<code>WizardData</code>	Elemento opcional.

Elementos primarios

Ninguno.

Observaciones

El `<VSTemplate>` elemento es el elemento raíz de los archivos *.vstemplate*.

Ejemplo

En el ejemplo siguiente se muestran Visual C# los metadatos de una plantilla de proyecto para una aplicación.

```
<VSTemplate Type="Project" Version="3.0.0"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>My template</Name>
    <Description>A basic starter kit</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyStarterKit.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantilla de Visual Studio](#)
- [Creación de plantillas de proyecto y elemento](#)

WizardData (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica XML personalizado

```
\<VSTemplate>
\<WizardData>
```

Sintaxis

```
<WizardData>
  <!-- XML to pass to the custom wizard extension -->
  ...
</WizardData>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
VSTemplate	Elemento necesario. Contiene todos los metadatos de la plantilla de proyecto, la plantilla de elemento o el kit de inicio.

Valor de texto

El valor de texto es opcional.

Este texto especifica el XML personalizado que se va a pasar a la extensión del asistente personalizado especificada en el [WizardExtension](#) elemento.

Observaciones

Cualquier XML se puede especificar en este elemento. El XML se pasará como parámetro a la extensión del asistente personalizado, lo que permite que la extensión use el contenido de este elemento. No se realiza ninguna validación en estos datos.

El contenido del elemento **WizardData** se pasa, sin cambios, como `IWizard.RunStarted` un parámetro dentro del diccionario de cadenas de parámetros en el método. La clave del `$wizarddata$` diccionario se denomina .

Ejemplo

En el ejemplo siguiente se muestran los metadatos de la plantilla de proyecto estándar para una aplicación de Windows de C.

```
<VSTemplate Version="3.0.0" Type="Item"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyTemplate</Name>
    <Description>Template using IWizard extension</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
  <WizardExtension>
    <Assembly>MyWizard, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
Custom=null</Assembly>
    <FullClassName>MyWizard.CustomWizard</FullClassName>
  </WizardExtension>
  <WizardData>
    <!-- XML to pass to the custom wizard extension -->
  </WizardData>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [WizardExtension \(Elemento, Plantillas de Visual Studio\)](#)
- [Cómo: Utilizar los asistentes con las plantillas de proyectos](#)

WizardExtension (Elemento, Plantillas de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Contiene los elementos de registro para personalizar el asistente de plantilla.

<VSTemplate> ... <WizardExtension>

Sintaxis

```
<WizardExtension>
  <Assembly>... </Assembly>
  <FullClassName>... </FullClassName>
</WizardExtension>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
ensamblaje	Elemento necesario. Especifica el nombre o el nombre seguro de un ensamblado que aparece en la caché global de ensamblados. Debe haber al <code>Assembly</code> menos un <code>WizardExtension</code> elemento en un elemento.
FullClassName	Elemento necesario. El nombre completo de la clase <code>IWizard</code> que implementa la interfaz. Debe haber al <code>FullClassName</code> menos un <code>WizardExtension</code> elemento en un elemento.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
VSTemplate	Contiene todos los metadatos de la plantilla de proyecto, la plantilla de elemento o el kit de inicio.

Observaciones

`WizardExtension` es un elemento secundario opcional de `VSTemplate`.

Ejemplo

En el ejemplo siguiente se muestran los Visual C# metadatos de la plantilla de proyecto estándar para una aplicación de Windows.

```
<VSTemplate Version="3.0.0" Type="Item"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyTemplate</Name>
    <Description>Template using IWizard extension</Description>
    <Icon>TemplateIcon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <Project File="MyTemplate.csproj">
      <ProjectItem>Form1.cs</ProjectItem>
      <ProjectItem>Form1.Designer.cs</ProjectItem>
      <ProjectItem>Program.cs</ProjectItem>
      <ProjectItem>Properties\AssemblyInfo.cs</ProjectItem>
      <ProjectItem>Properties\Resources.resx</ProjectItem>
      <ProjectItem>Properties\Resources.Designer.cs</ProjectItem>
      <ProjectItem>Properties\Settings.settings</ProjectItem>
      <ProjectItem>Properties\Settings.Designer.cs</ProjectItem>
    </Project>
  </TemplateContent>
  <WizardExtension>
    <Assembly>MyWizard, Version=1.0.3300.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,
Custom=null</Assembly>
    <FullClassName>MyWizard.CustomWizard</FullClassName>
  </WizardExtension>
</VSTemplate>
```

Vea también

- [Referencia de esquema de plantillas de Visual Studio](#)
- [Crear plantillas para proyectos y elementos en Visual Studio](#)
- [Cómo: Utilizar los asistentes con las plantillas de proyectos](#)

Referencia de esquema de manifiesto de plantilla de Visual Studio

06/04/2020 • 4 minutes to read • [Edit Online](#)

Este esquema describe el formato de los archivos de manifiesto de plantilla de Visual Studio (*.vstman*) que se generan para las plantillas de proyecto o elemento de Visual Studio. El esquema también describe la ubicación y otra información relevante sobre la plantilla.

: Dado que hay directorios de plantillas de proyecto y elemento sin elementos, un manifiesto nunca debe tener una combinación de plantillas de elemento y proyecto.

IMPORTANT

Este manifiesto está disponible a partir de Visual Studio 2017.

Elemento VSTemplateManifest

El elemento raíz del manifiesto.

Atributos

- **Versión:** una cadena que representa la versión del manifiesto de plantilla. Necesario.
- **Configuración regional:** cadena que representa la configuración regional o las configuraciones regionales del manifiesto de plantilla. El valor de configuración regional se aplica a todas las plantillas. Debe usar un manifiesto independiente para cada configuración regional. Opcional.

Elementos secundarios

- **VSTemplateContainer** Opcional.
- **VSTemplateDir** Opcional.

Elemento primario

Ninguno.

VSTemplateContainer

El contenedor de los elementos de manifiesto de plantilla. Un manifiesto tiene un contenedor de plantillas para cada plantilla que define.

Atributos

VSTemplateType: un valor de cadena que `"Project"` especifica `"Item"` el `"ProjectGroup"` tipo de la plantilla (, , o). Obligatorio

Elementos secundarios

- **RelativePathOnDisk:** la ruta de acceso relativa del archivo de plantilla en el disco. Esta ubicación también define la ubicación de la plantilla en el árbol de plantillas que se muestra en el **cuadro de** diálogo Nuevo proyecto o Nuevo **elemento**. Para las plantillas implementadas como un directorio y archivos individuales, esta ruta de acceso hace referencia al directorio que contiene los archivos de plantilla. Para las plantillas implementadas como un archivo *.zip*, esta ruta de acceso debe ser la ruta de acceso al archivo *.zip*.

- ****VSTemplateHeader**: un elemento [TemplateData](#) que describe el encabezado.

Elemento primario

VSTemplateManifest

VSTemplateDir

Describe el directorio donde se encuentra la plantilla. Un manifiesto puede contener varias entradas **VSTemplateDir** para proporcionar el nombre localizado y el criterio de ordenación para que los directorios controlen su apariencia en el árbol de categorías de plantilla.

Debido a su diseño, las entradas **VSTemplateDir** deben aparecer solo en manifiestos no especificados en la configuración regional.

Atributos

Ninguno.

Elementos secundarios

- **RelativePath**: La ruta de acceso de la plantilla. Solo puede haber una entrada por ruta, por lo que la primera ganará para todos los manifiestos.
- **LocalizedName**: un elemento **NameDescriptionIcon** que especifica el nombre localizado. Opcional.
- **SortOrder**: cadena que especifica el criterio de ordenación. Opcional.
- **ParentFolderOverrideName**: el nombre invalidado de la carpeta primaria. Opcional. Este elemento tiene un atributo **Name**, que es un valor de cadena que especifica el nombre.

Elemento primario

VSTemplateManifest

NameDescriptionIcon

Especifica el nombre y la descripción, posiblemente para las plantillas localizadas. Consulte **LocalizedName** arriba.

Atributos

- **Package**: un valor de cadena que especifica el paquete. Opcional.
- **ID**: un valor de cadena que especifica el identificador. Opcional.

Elementos secundarios

Ninguno.

Elemento primario

LocalizedName

Ejemplos

El código siguiente es un ejemplo de un archivo *.vstman* de plantilla de proyecto.


```

<VSTemplateManifest Version="1.0" Locale="1033"
xmlns="http://schemas.microsoft.com/developer/vstemplatemanifest/2015">
  <VSTemplateContainer TemplateType="Project">
    <RelativePathOnDisk>CSharp\1033\TestProjectTemplate</RelativePathOnDisk>
    <TemplateFileName>TestProjectTemplate.vstemplate</TemplateFileName>
    <VSTemplateHeader>
      <TemplateData xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
        <Name>TestProjectTemplate</Name>
        <Description>TestProjectTemplate</Description>
        <Icon>TestProjectTemplate.ico</Icon>
        <ProjectType>CSharp</ProjectType>
        <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
        <SortOrder>1000</SortOrder>
        <TemplateID>aac0aeea-7883-4003-992f-937d53d70ab1</TemplateID>
        <CreateNewFolder>true</CreateNewFolder>
        <DefaultName>TestProjectTemplate</DefaultName>
        <ProvideDefaultName>true</ProvideDefaultName>
      </TemplateData>
    </VSTemplateHeader>
  </VSTemplateContainer>
</VSTemplateManifest>

```

El código siguiente es un ejemplo de un archivo *.vstman* de plantilla de elemento.

```

<VSTemplateManifest Version="1.0" Locale="1033"
xmlns="http://schemas.microsoft.com/developer/vstemplatemanifest/2015">
  <VSTemplateContainer TemplateType="Item">
    <RelativePathOnDisk>CSharp\1033\ItemTemplate1</RelativePathOnDisk>
    <TemplateFileName>ItemTemplate1.vstemplate</TemplateFileName>
    <VSTemplateHeader>
      <TemplateData xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
        <Name>ItemTemplate1</Name>
        <Description>ItemTemplate1</Description>
        <Icon>ItemTemplate1.ico</Icon>
        <TemplateID>bfeadf8e-a251-4109-b605-516b88e38c8d</TemplateID>
        <ProjectType>CSharp</ProjectType>
        <RequiredFrameworkVersion>2.0</RequiredFrameworkVersion>
        <NumberOfParentCategoriesToRollUp>1</NumberOfParentCategoriesToRollUp>
        <DefaultName>Class.cs</DefaultName>
      </TemplateData>
    </VSTemplateHeader>
  </VSTemplateContainer>
</VSTemplateManifest>

```

Ampliar la configuración y las opciones del usuario

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio hay dos formas principales de mantener las preferencias del usuario. Las **Tools > páginas Opciones** de herramientas permiten a los usuarios establecer sus propios valores para muchos tipos diferentes de configuración, tanto generales como específicas del idioma. La configuración de usuario permite a los usuarios especificar propiedades generales de configuración de Visual Studio. Para obtener más información acerca de las páginas de opciones, consulte [Opciones y páginas de opciones](#) . Para obtener más información acerca de la configuración de usuario, consulte [Compatibilidad con la configuración de usuario](#) .

- [Crear una página de opciones](#) Explica cómo crear una página de opciones.
- [Crear una categoría de configuración](#) Explica cómo crear una categoría de configuración.
- [Utilice la tienda de ajustes](#) Explica cómo usar el almacén de configuración.
- [Obtener información de servicio del almacén de configuración](#) Explica cómo obtener los servicios disponibles desde el almacén de configuración.
- [Escribir en el almacén de configuración de usuario](#) Explica cómo escribir en el almacén de configuración de usuario.

Crear una página de opciones

06/04/2020 • 12 minutes to read • [Edit Online](#)

Este tutorial crea una página sencilla Herramientas/Opciones que usa una cuadrícula de propiedades para examinar y establecer propiedades.

Para guardar estas propiedades y restaurarlas desde un archivo de configuración, siga estos pasos y, a continuación, consulte [Crear una categoría](#) de configuración .

El MPF proporciona dos clases para ayudarle [Package](#) a [DialogPage](#) crear páginas de opciones de herramientas, la clase y la clase. Crear un `VSPackage` para proporcionar un contenedor para `Package` estas páginas mediante la subclase de la clase. Puede crear cada página de opciones `DialogPage` de herramientas derivando de la clase.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear una página de cuadrícula Opciones de herramientas

En esta sección, creará una cuadrícula de propiedades Opciones de herramientas simple. Utilice esta cuadrícula para mostrar y cambiar el valor de una propiedad.

Para crear el proyecto VSIX y agregar un VSPackage

1. Cada extensión de Visual Studio comienza con un proyecto de implementación de VSIX, que contendrá los activos de extensión. Cree Visual Studio un proyecto `MyToolsOptionsExtension` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Agregue un `VSPackage` agregando una plantilla `MyToolsOptionsPackage` de elemento de paquete de Visual Studio denominada . En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad** de elementos > de **Visual C.** y seleccione **Paquete de Visual Studio**. En el campo **Nombre** en la parte inferior `MyToolsOptionsPackage.cs` del cuadro de diálogo, cambie el nombre del archivo a . Para obtener más información acerca de cómo crear un `VSPackage`, vea [crear una extensión con un VSPackage](#).

Para crear la cuadrícula de propiedades Opciones de herramientas

1. Abra el archivo `MyToolsOptionsPackage` en el editor de código.
2. Agregue la siguiente instrucción using.

```
using System.ComponentModel;
```

3. Declare `OptionPageGrid` una clase y `DialogPage` derivarla de .

```
public class OptionPageGrid : DialogPage
{ }
```

4. Aplique [ProvideOptionPageAttribute](#) a `VSPackage` a la clase para asignar a la clase una categoría de

opciones y el nombre de página de opciones para el `OptionPageGrid`. El resultado debería tener este aspecto:

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)]
[ProvideMenuResource("Menus.ctmenu", 1)]
[Guid(GuidList.guidMyToolsOptionsPkgString)]
[ProvideOptionPage(typeof(OptionPageGrid),
    "My Category", "My Grid Page", 0, 0, true)]
public sealed class MyToolsOptionsPackage : Package
```

5. Agregue `OptionInteger` una propiedad `OptionPageGrid` a la clase.

- Aplique `System.ComponentModel.CategoryAttribute` a para asignar a la propiedad una categoría de cuadrícula de propiedades.
- Aplique `System.ComponentModel.DisplayNameAttribute` a para asignar a la propiedad un nombre.
- Aplique `System.ComponentModel.DescriptionAttribute` a para asignar a la propiedad una descripción.

```
public class OptionPageGrid : DialogPage
{
    private int optionInt = 256;

    [Category("My Category")]
    [DisplayName("My Integer Option")]
    [Description("My integer option")]
    public int OptionInteger
    {
        get { return optionInt; }
        set { optionInt = value; }
    }
}
```

NOTE

La implementación `DialogPage` predeterminada de admite propiedades que tienen convertidores adecuados o que son estructuras o matrices que se pueden expandir en propiedades que tienen convertidores adecuados. Para obtener una lista de `System.ComponentModel` convertidores, consulte el espacio de nombres.

6. Compile la solución y comience la depuración.

7. En la instancia experimental de Visual Studio, en el menú **Herramientas**, haga clic en **Opciones**.

En el panel izquierdo, debería ver **Mi categoría**. (Las categorías de opciones se enumeran en orden alfabético, por lo que debería aparecer aproximadamente a la mitad de la lista.) Abra **Mi categoría** y, a continuación, haga clic en Mi página de **cuadrícula**. La cuadrícula de opciones aparece en el panel derecho. La categoría de propiedad es **Mis opciones** y el nombre de la propiedad es **Mi opción entero**. La descripción de la propiedad, **Mi opción de entero**, aparece en la parte inferior del panel. Cambie el valor de su valor inicial de 256 a otra cosa. Haga clic en **Aceptary**, a continuación, vuelva a abrir **Mi página de cuadrícula**. Puede ver que el nuevo valor persiste.

La página de opciones también está disponible a través del cuadro de búsqueda de Visual Studio. En el cuadro de búsqueda situado cerca de la parte superior del IDE, escriba **Mi categoría** y verá **Mi categoría** -> **Mi página de cuadrícula** en la lista de resultados.

Crear una página personalizada Opciones de herramientas

En esta sección, creará una página Opciones de herramientas con una interfaz de usuario personalizada. Utilice esta página para mostrar y cambiar el valor de una propiedad.

1. Abra el archivo *MyToolsOptionsPackage* en el editor de código.
2. Agregue la siguiente instrucción using.

```
using System.Windows.Forms;
```

3. Agregue `OptionPageCustom` una clase, `OptionPageGrid` justo antes de la clase. Derive la nueva `DialogPage` clase de .

```
public class OptionPageCustom : DialogPage
{
    private string optionValue = "alpha";

    public string OptionString
    {
        get { return optionValue; }
        set { optionValue = value; }
    }
}
```

4. Agregue un atributo GUID. Agregue una propiedad `OptionString`:

```
[Guid("00000000-0000-0000-0000-000000000000")]
public class OptionPageCustom : DialogPage
{
    private string optionValue = "alpha";

    public string OptionString
    {
        get { return optionValue; }
        set { optionValue = value; }
    }
}
```

5. Aplicar un [ProvideOptionPageAttribute](#) segundo a la clase `VSPackage`. Este atributo asigna a la clase una categoría de opciones y un nombre de página de opciones.

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)]
[ProvideMenuResource("Menus.ctmenu", 1)]
[Guid(GuidList.guidMyToolsOptionsPkgString)]
[ProvideOptionPage(typeof(OptionPageGrid),
    "My Category", "My Grid Page", 0, 0, true)]
[ProvideOptionPage(typeof(OptionPageCustom),
    "My Category", "My Custom Page", 0, 0, true)]
public sealed class MyToolsOptionsPackage : Package
```

6. Agregue un nuevo control de **usuario** denominado `MyUserControl` al proyecto.
7. Agregue un control `TextBox` al control de usuario.

En la ventana **Propiedades**, en la barra de herramientas, haga clic en el botón **Eventos** y, a continuación, haga doble clic en el evento **Leave**. El nuevo controlador de eventos aparece en el *código de MyUserControl.cs*.

8. Agregue un `OptionsPage` campo `Initialize` público, un método a la clase de control y actualice el controlador de eventos para establecer el valor de opción en el contenido del cuadro de texto:

```
public partial class MyUserControl : UserControl
{
    public MyUserControl()
    {
        InitializeComponent();

        internal OptionPageCustom optionsPage;

        public void Initialize()
        {
            textBox1.Text = optionsPage.OptionString;
        }

        private void textBox1_Leave(object sender, EventArgs e)
        {
            optionsPage.OptionString = textBox1.Text;
        }
    }
}
```

El `optionsPage` campo contiene una referencia `OptionPageCustom` a la instancia primaria. El `Initialize` método `OptionString` se muestra en `textBox`. El controlador de eventos escribe el valor `OptionString` actual de la `TextBox` en el momento en que el foco deja el `textBox`.

9. En el archivo de código de `OptionPageCustom.Window` paquete, `OptionPageCustom` agregue una invalidación de `MyUserControl` la propiedad a la clase para crear, inicializar y devolver una instancia de . La clase ahora debería tener este aspecto:

```
[Guid("00000000-0000-0000-0000-000000000000")]
public class OptionPageCustom : DialogPage
{
    private string optionValue = "alpha";

    public string OptionString
    {
        get { return optionValue; }
        set { optionValue = value; }
    }

    protected override IWin32Window Window
    {
        get
        {
            MyUserControl page = new MyUserControl();
            page.optionsPage = this;
            page.Initialize();
            return page;
        }
    }
}
```

10. Compile y ejecute el proyecto.
11. En la instancia **Tools** > experimental, haga clic en **Opciones de herramientas**.
12. Busque **Mi categoría** y, a continuación, Mi página **personalizada**.
13. Cambie el valor de **OptionString**. Haga clic en **Aceptary**, a continuación, vuelva a abrir **Mi página personalizada**. Puede ver que el nuevo valor ha persistido.

Opciones de acceso

En esta sección, obtendrá el valor de una opción del VSPackage que hospeda la página opciones de herramientas asociada. La misma técnica se puede utilizar para obtener el valor de cualquier propiedad pública.

1. En el archivo de código de paquete, agregue una propiedad pública denominada **OptionInteger** a la clase **MyToolsOptionsPackage**.

```
public int OptionInteger
{
    get
    {
        OptionPageGrid page = (OptionPageGrid)GetDialogPage(typeof(OptionPageGrid));
        return page.OptionInteger;
    }
}
```

Este código [GetDialogPage](#) llama para `OptionPageGrid` crear o recuperar una instancia. `OptionPageGrid` llamadas [LoadSettingsFromStorage](#) para cargar sus opciones, que son propiedades públicas.

2. Ahora agregue una plantilla de elemento de comando personalizada denominada **MyToolsOptionsCommand** para mostrar el valor. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C-** > y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a *MyToolsOptionsCommand.cs*.
3. En el archivo *MyToolsOptionsCommand*, reemplace el `ShowMessageBox` cuerpo del método del comando por el siguiente:

```
private void ShowMessageBox(object sender, EventArgs e)
{
    MyToolsOptionsPackage myToolsOptionsPackage = this.package as MyToolsOptionsPackage;
    System.Windows.Forms.MessageBox.Show(string.Format(CultureInfo.CurrentCulture, "OptionInteger: {0}", myToolsOptionsPackage.OptionInteger));
}
```

4. Compile la solución y comience la depuración.
5. En la instancia experimental, en el menú **Herramientas**, haga clic en **Invocar MyToolsOptionsCommand**.

Un cuadro de mensaje `OptionInteger` muestra el valor actual de .

Vea también

- [Páginas de opciones y opciones](#)

Crear una categoría de configuración

06/04/2020 • 8 minutes to read • [Edit Online](#)

En este tutorial se crea una categoría de configuración de Visual Studio y se usa para guardar los valores y restaurar los valores de un archivo de configuración. Una categoría de configuración es un grupo de propiedades relacionadas que aparecen como un "punto de configuración personalizado"; es decir, como casilla de verificación en el asistente **Importar y exportación de configuración**. (Puede encontrarlo en el menú **Herramientas**.) La configuración se guarda o restaura como una categoría, y la configuración individual no se muestra en el asistente. Para obtener más información, vea [Configuración del entorno](#).

Puede crear una categoría de configuración [DialogPage](#) derivando de la clase.

Para iniciar este tutorial, primero debe completar la primera sección de [la página Crear una página de opciones](#). La cuadrícula de propiedades Opciones resultante le permite examinar y cambiar las propiedades de la categoría. Después de guardar la categoría de propiedad en un archivo de configuración, examine el archivo para ver cómo se almacenan los valores de propiedad.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear una categoría de configuración

En esta sección, se utiliza un punto de configuración personalizado para guardar y restaurar los valores de la categoría de configuración.

Para crear una categoría de configuración

1. Complete la [página Crear opciones](#).
2. Abra el archivo *VSPackage.resx* y agregue estos tres recursos de cadena:

NOMBRE	VALUE
106	Mi categoría
107	Mi configuración
108	OptionInteger y OptionFloat

Esto crea recursos que nombran la categoría "Mi categoría", el objeto "Mi configuración" y la descripción de categoría "OptionInteger y OptionFloat".

NOTE

De estos tres, solo el nombre de la categoría no aparece en el asistente **Importar y exportar configuración**.

3. En *MyToolsOptionsPackage.cs*, `float` agregue `OptionFloat` una `OptionPageGrid` propiedad con nombre a la clase, como se muestra en el ejemplo siguiente.


```

public class OptionPageGrid : DialogPage
{
    private int optionInt = 256;
    private float optionFloat = 3.14F;

    [Category("My Options")]
    [DisplayName("My Integer option")]
    [Description("My integer option")]
    public int OptionInteger
    {
        get { return optionInt; }
        set { optionInt = value; }
    }
    [Category("My Options")]
    [DisplayName("My Float option")]
    [Description("My float option")]
    public float OptionFloat
    {
        get { return optionFloat; }
        set { optionFloat = value; }
    }
}

```

NOTE

La `OptionPageGrid` categoría denominada "Mi categoría" ahora `OptionInteger` consta `OptionFloat` de las dos propiedades y .

4. Agregue `ProvideProfileAttribute` un `MyToolsOptionsPackage` a la clase y asíbóeles el `CategoryName` "My Category", asírácele el `ObjectName` "My Settings" y establezca `isToolsOptionPage` en true. Establezca `categoryResourceID`, `objectNameResourceID` y `DescriptionResourceID` en los identificadores de recursos de cadena correspondientes creados anteriormente.

```

[ProvideProfileAttribute(typeof(OptionPageGrid),
    "My Category", "My Settings", 106, 107, isToolsOptionPage:true, DescriptionResourceID = 108)]

```

5. Compile la solución y comience la depuración. En la instancia experimental debería ver que **Mi página** de cuadrícula ahora tiene valores enteros y flotantes.

Examine el archivo de configuración

En esta sección, exportará los valores de categoría de propiedad a un archivo de configuración. Examine el archivo y, a continuación, vuelva a importar los valores en la categoría de propiedad.

1. Inicie el proyecto en modo de depuración presionando **F5**. Esto inicia la instancia experimental.
2. Abra el cuadro de diálogo **Opciones de herramientas**. >
3. En la vista de árbol del panel izquierdo, expanda **Mi categoría** y, a continuación, haga clic en **Mi página de cuadrícula**.
4. Cambie el valor de **OptionFloat** a 3.1416 y **OptionInteger** a 12. Haga clic en **OK**.
5. En el menú **Herramientas**, haga clic en **Importar y exportar configuraciones**.

Aparece el asistente **Importar y exportar parámetros**.

6. Asegúrese de que la opción **Exportar configuración de entorno seleccionada** esté seleccionada y, a continuación, haga clic en **Siguiente**.

Aparece la página **Elegir configuración para exportar**.

7. Haga clic en **Mi configuración**.

La **descripción** cambia a **OptionInteger** y **OptionFloat**.

8. Asegúrese de que **Mi configuración** es la única categoría seleccionada y, a continuación, haga clic en **Siguiente**.

Aparece la página Nombre del archivo de **configuración**.

9. Asigne al nuevo archivo de configuración el nombre *MySettings.vssettings* y guárdelo en un directorio adecuado. Haga clic en **Finalizar**

La página **Exportar completado** informa de que la configuración se ha exportado correctamente.

10. En el menú **Archivo**, seleccione **Abrir** haga clic en **Archivo**. Busque *MySettings.vssettings* y ábralo.

Puede encontrar la categoría de propiedad que exportó en la siguiente sección del archivo (sus GUID diferirán).

```
<Category name="My Category_My Settings"
  Category="{4802bc3e-3d9d-4591-8201-23d1a05216a6}"
  Package="{6bb6942e-014c-489e-a612-a935680f703d}"
  RegisteredName="My Category_My Settings">
  PackageName="MyToolsOptionsPackage">
    <PropertyValue name="OptionFloat">3.1416</PropertyValue>
    <PropertyValue name="OptionInteger">12</PropertyValue>
  </Category>
```

Observe que el nombre completo de la categoría está formado por la adición de un carácter de subrayado al nombre de categoría seguido del nombre del objeto. OptionFloat y OptionInteger aparecen en la categoría, junto con sus valores exportados.

11. Cierre el archivo de configuración sin cambiarlo.
12. En el menú **Herramientas**, haga clic en **Opciones**, expanda **Mi categoría**, haga clic en Mi página de **cuadrícula** y, a continuación, cambie el valor de **OptionFloat** a 1.0 y **OptionInteger** a 1. Haga clic en **OK**.
13. En el menú **Herramientas**, haga clic en **Importar y exportar configuración**, seleccione **Importar configuración de entorno seleccionada**, a continuación, haga clic en **Siguiente**.

Aparece la página **Guardar configuración actual**.

14. Seleccione **No, solo tiene que importar la nueva configuración** y, a continuación, haga clic en **Siguiente**.

Aparece la página **Elegir una colección de configuraciones para importar**.

15. Seleccione el archivo *MySettings.vssettings* en el nodo **Mis configuraciones** de la vista de árbol. Si el archivo no aparece en la vista de árbol, haga clic en **Examinar** y búsquelo. Haga clic en **Next**.

Aparece el cuadro de diálogo **Elegir configuración para importar**.

16. Asegúrese de que **Mi configuración** está seleccionada y, a continuación, haga clic en **Finalizar**. Cuando aparezca la página **Importar completada**, haga clic en **Cerrar**.
17. En el menú **Herramientas**, haga clic en **Opciones**, expanda **Mi categoría**, haga clic en Mi página de **cuadrícula** y compruebe que se han restaurado los valores de categoría de propiedad.

Uso del almacén de configuración

06/04/2020 • 4 minutes to read • [Edit Online](#)

Hay dos tipos de tiendas de configuración:

- Configuración, que son de solo lectura Visual Studio y VSPackage configuración. Visual Studio combina la configuración de todos los archivos .pkgdef conocidos en este almacén.
- Configuración de usuario, que son configuraciones que se pueden escribir, como las que se muestran en las páginas del cuadro de diálogo **Opciones**, las páginas de propiedades y otros cuadros de diálogo. Las extensiones de Visual Studio pueden usarlas para el almacenamiento local de pequeñas cantidades de datos.

En este tutorial se muestra cómo leer datos del almacén de configuración. Consulte [Escritura en el almacén de configuración de usuario](#) para obtener una explicación de cómo escribir en el almacén de configuración de usuario.

Creación del proyecto de ejemplo

En esta sección se muestra cómo crear un proyecto de extensión simple con un comando de menú para la demostración.

1. Cada extensión de Visual Studio comienza con un proyecto de implementación de VSIX que contendrá los activos de extensión. Cree Visual Studio un proyecto `SettingsStoreExtension` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto en Visual C. / Extensibilidad**.
2. Ahora agregue una plantilla de elemento de comando personalizada denominada **SettingsStoreCommand**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a Visual C/ **Extensibilidad** y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a **SettingsStoreCommand.cs**. Para obtener más información sobre cómo crear un comando personalizado, consulte [Creación de una extensión con un comando de menú](#)

Uso del almacén de configuración

En esta sección se muestra cómo detectar y mostrar los valores de configuración.

1. En el archivo `SettingsStorageCommand.cs`, agregue las siguientes directivas using:

```
using System.Collections.Generic;
using Microsoft.VisualStudio.Settings;
using Microsoft.VisualStudio.Shell.Settings;
using System.Windows.Forms;
```

2. En `MenuItemCallback`, quite el cuerpo del método y agregue estas líneas obtenga el almacén de valores de configuración:

```
SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);
SettingsStore configurationSettingsStore =
    settingsManager.GetReadOnlySettingsStore(SettingsScope.Configuration);
```

Es [ShellSettingsManager](#) una clase auxiliar [IVsSettingsManager](#) administrada sobre el servicio.

3. Ahora averigua si las herramientas de Windows Phone están instaladas. El código debe tener este aspecto:

```
private void MenuItemCallback(object sender, EventArgs e)
{
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);
    SettingsStore configurationSettingsStore =
settingsManager.GetReadOnlySettingsStore(SettingsScope.Configuration);
    bool arePhoneToolsInstalled =
configurationSettingsStore.CollectionExists(@"InstalledProducts\Microsoft Windows Phone Developer
Tools");
    string message = "Microsoft Windows Phone Developer Tools: " + arePhoneToolsInstalled;
    MessageBox.Show(message);
}
```

4. Prueba del código Compile la solución y comience la depuración.
5. En la instancia experimental, en el menú **Herramientas** , haga clic en **Invocar SettingsStoreCommand**.

Debería ver un cuadro de mensaje que dice Herramientas de desarrollo de **Microsoft Windows Phone**: seguido de **Verdadero** o **Falso**.

Visual Studio mantiene el almacén de configuración en el registro del sistema.

Usar un editor del Registro para verificar la configuración

1. Abra regedit.exe.
2. Navegue hasta HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 14,0Exp_Config, InstalledProducts\.

NOTE

Asegúrese de que está mirando la clave que contiene el valor de\14.0Exp_Config y no el 14,0_Config . Al ejecutar la instancia experimental de Visual Studio, los valores de configuración se encuentran en el subárbol del Registro "14.0Exp_Config".

3. Expanda el nodo "Productos instalados". Si el mensaje de los pasos anteriores es **Microsoft Windows Phone Developer Tools Installed: True**, entonces , los productos instalados deben contener un nodo de Microsoft Windows Phone Developer Tools. Si el mensaje es **Microsoft Windows Phone Developer Tools Installed: False**, no debe contener un nodo Herramientas de desarrollo de Microsoft Windows Phone.

Obtener información de servicio del almacén de configuración

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede usar el almacén de configuración para buscar todos los servicios disponibles o para determinar si un servicio determinado está instalado. Debe conocer el tipo de la clase de servicio.

Para enumerar los servicios disponibles

1. Cree un proyecto `FindServicesExtension` VSIX con nombre `FindServicesCommand` y, a continuación, agregue un comando personalizado denominado `FindServicesCommand`. Para obtener más información sobre cómo crear un comando personalizado, consulte [Crear una extensión con un comando](#) de menú.
2. En `FindServicesCommand.cs`, agregue las siguientes directivas using:

```
using System.Collections.Generic;  
using Microsoft.VisualStudio.Settings;  
using Microsoft.VisualStudio.Shell.Settings;  
using System.Windows.Forms;
```

3. Obtenga el almacén de valores de configuración y, a continuación, busque la subcolección denominada `Services`. Esta colección incluye todos los servicios disponibles. En `MenuItemCommand` el método, quite el código existente y sustitúyalo por lo siguiente:

```
private void MenuItemCallback(object sender, EventArgs e)  
{  
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);  
    SettingsStore configurationSettingsStore =  
settingsManager.GetReadOnlySettingsStore(SettingsScope.Configuration);  
    string message = "Available services:\n";  
    IEnumerable<string> collection = configurationSettingsStore.GetSubCollectionNames("Services");  
    int n = 0;  
    foreach (string service in collection)  
    {  
        message += configurationSettingsStore.GetString("Services\\" + service, "Name", "Unknown") +  
"\n";  
    }  
  
    MessageBox.Show(message);  
}
```

4. Compile la solución y comience la depuración. Aparece la instancia experimental.
5. En la instancia experimental, en el menú **Herramientas**, haga clic en **Invocar FindServicesCommand**.

Debería ver un cuadro de mensaje con todos los servicios.

Para comprobar esta configuración, puede usar el editor del Registro.

Encuentre un servicio específico

También puede utilizar `CollectionExists` el método para determinar si un servicio determinado está instalado. Debe conocer el tipo de la clase de servicio.

1. En el MenuItemCallback del proyecto que creó en el procedimiento `Services` anterior, busque en el almacén de valores de configuración la colección que tiene la subcolección nombrada por el GUID del servicio. En este caso buscaremos el servicio de Ayuda.

```
private void MenuItemCallback(object sender, EventArgs e)
{
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);
    SettingsStore configurationSettingsStore =
settingsManager.GetReadOnlySettingsStore(SettingsScope.Configuration);
    string helpServiceGUID = typeof(SVsHelpService).GUID.ToString("B").ToUpper();
    bool hasHelpService = configurationSettingsStore.CollectionExists("Services\\" + helpServiceGUID);
    string message = "Help Service Available: " + hasHelpService;

    MessageBox.Show(message);
}
```

2. Compile la solución y comience la depuración.
3. En la instancia experimental, en el menú **Herramientas** , haga clic en **Invocar FindServicesCommand**.

Debería ver un mensaje con el texto Servicio de **ayuda disponible**: seguido de **Verdadero** o **Falso**. Para comprobar esta configuración, puede usar un editor del Registro, como se muestra en los pasos anteriores.

Escritura en el almacén de configuración de usuario

06/04/2020 • 3 minutes to read • [Edit Online](#)

La configuración de usuario es una configuración grabable como las del cuadro de diálogo **Herramientas / Opciones**, las ventanas de propiedades y algunos otros cuadros de diálogo. Las extensiones de Visual Studio pueden usarlas para almacenar pequeñas cantidades de datos. En este tutorial se muestra cómo agregar el Bloc de notas a Visual Studio como una herramienta externa leyendo y escribiendo en el almacén de configuración de usuario.

Escritura en el almacén de configuración de usuario

1. Cree un proyecto VSIX denominado `UserSettingsStoreExtension` y, a continuación, agregue un comando personalizado denominado `UserSettingsStoreCommand`. Para obtener más información sobre cómo crear un comando personalizado, consulte [Creación de una extensión con un comando](#) de menú
2. En `UserSettingsStoreCommand.cs`, agregue las siguientes directivas using:

```
using System.Collections.Generic;  
using Microsoft.VisualStudio.Settings;  
using Microsoft.VisualStudio.Shell.Settings;
```

3. En `MenuItemCallback`, elimine el cuerpo del método y obtenga el almacén de configuración de usuario, como se indica a continuación:

```
private void MenuItemCallback(object sender, EventArgs e)  
{  
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);  
    WritableSettingsStore userSettingsStore =  
        settingsManager.GetWritableSettingsStore(SettingsScope.UserSettings);  
}
```

4. Ahora descubra si el Bloc de notas ya está configurado como una herramienta externa. Debe recorrer en iteración todas las herramientas externas para determinar si el ajuste `ToolCmd` es "Notepad", como se indica a continuación:

```
private void MenuItemCallback(object sender, EventArgs e)
{
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);
    WritableSettingsStore userSettingsStore =
settingsManager.GetWritableSettingsStore(SettingsScope.UserSettings);

    // Find out whether Notepad is already an External Tool.
    int toolCount = userSettingsStore.GetInt32("External Tools", "ToolNumKeys");
    bool hasNotepad = false;
    CompareInfo Compare = CultureInfo.InvariantCulture.CompareInfo;
    for (int i = 0; i < toolCount; i++)
    {
        if (Compare.IndexOf(userSettingsStore.GetString("External Tools", "ToolCmd" + i), "Notepad",
CompareOptions.IgnoreCase) >= 0)
        {
            hasNotepad = true;
            break;
        }
    }
}
```

5. Si el Bloc de notas no se ha establecido como una herramienta externa, establézcalo de la siguiente manera:

```
private void MenuItemCallback(object sender, EventArgs e)
{
    SettingsManager settingsManager = new ShellSettingsManager(ServiceProvider);
    WritableSettingsStore userSettingsStore =
settingsManager.GetWritableSettingsStore(SettingsScope.UserSettings);

    // Find out whether Notepad is already installed.
    int toolCount = userSettingsStore.GetInt32("External Tools", "ToolNumKeys");
    bool hasNotepad = false;
    CompareInfo Compare = CultureInfo.InvariantCulture.CompareInfo;
    for (int i = 0; i < toolCount; i++)
    {
        if (Compare.IndexOf(userSettingsStore.GetString("External Tools", "ToolCmd" + i), "Notepad",
CompareOptions.IgnoreCase) >= 0)
        {
            hasNotepad = true;
            break;
        }
    }

    string message = (hasNotepad) ? "Notepad already installed" : "Installing Notepad";
    if (!hasNotepad)
    {
        userSettingsStore.SetString("External Tools", "ToolTitle" + toolCount, "&Notepad");
        userSettingsStore.SetString("External Tools", "ToolCmd" + toolCount,
"C:\\Windows\\notepad.exe");
        userSettingsStore.SetString("External Tools", "ToolArg" + toolCount, "");
        userSettingsStore.SetString("External Tools", "ToolDir" + toolCount, "${ProjectDir}");
        userSettingsStore.SetString("External Tools", "ToolSourceKey" + toolCount, "");
        userSettingsStore.SetUInt32("External Tools", "ToolOpt" + toolCount, 0x00000011);

        userSettingsStore.SetInt32("External Tools", "ToolNumKeys", toolCount + 1);
    }
}
```

6. Prueba del código Recuerde que agrega el Bloc de notas como una herramienta externa, por lo que debe revertir el registro antes de ejecutarlo una segunda vez.

7. Compile el código e inicie la depuración.

8. En el menú **Herramientas** , haga clic en **Invocar UserSettingsStoreCommand**. Esto agregará el Bloc de notas al menú **Herramientas**.
9. Ahora debería ver el Bloc de notas en el menú Herramientas / Opciones, y al hacer clic en **Bloc de notas** debe abrir una instancia del Bloc de notas.

Extender propiedades y la ventana Propiedad

06/04/2020 • 2 minutes to read • [Edit Online](#)

La ventana **Propiedades** se utiliza para ver y editar las propiedades y eventos en tiempo de diseño de los objetos seleccionados.

Al seleccionar un Visual Studioobjeto en , las propiedades públicas de ese objeto aparecen en la ventana **Propiedades**. Para obtener más información, consulte [Introducción](#) a la ventana Propiedades .

En esta sección

- [Exponer propiedades a la ventana Propiedades](#) Explica cómo puede mostrar las propiedades personalizadas en la ventana Propiedades.
- [Obtener propiedades del proyecto](#) Explica cómo obtener propiedades del proyecto.

Exponer propiedades a la ventana Propiedades

06/04/2020 • 10 minutes to read • [Edit Online](#)

Este tutorial expone las propiedades públicas de un objeto a la **propiedades** ventana. Los cambios realizados en estas propiedades se reflejan en la ventana **Propiedades**.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Exponer propiedades a la ventana Propiedades

En esta sección, creará una ventana de herramientas personalizada y mostrará las propiedades públicas del objeto de panel de ventana asociado en la ventana **Propiedades**.

Para exponer propiedades a la ventana Propiedades

1. Cada extensión de Visual Studio comienza con un proyecto de implementación de VSIX, que contendrá los activos de extensión. Cree Visual Studio un proyecto `MyObjectPropertiesExtension` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Agregue una ventana de herramientas agregando `MyToolWindow` una plantilla de elemento de ventana de herramientas personalizada denominada . En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad** de elementos > de **Visual C.** y seleccione **Ventana de herramientas personalizadas**. En el campo **Nombre** en la parte inferior del cuadro de diálogo, cambie el nombre del archivo a `MyToolWindow.cs`. Para obtener más información sobre cómo crear una ventana de herramientas personalizada, consulte [Crear una extensión con una ventana](#) de herramientas .
3. Abra `MyToolWindow.cs` y agregue la siguiente instrucción using:

```
using System.Collections;
using System.ComponentModel;
using Microsoft.VisualStudio.Shell.Interop;
```

4. Ahora agregue los siguientes `MyToolWindow` campos a la clase.

```
private ITrackSelection trackSel;
private SelectionContainer selContainer;
```

5. Agregue el siguiente código a la clase `MyToolWindow` .

```

private ITrackSelection TrackSelection
{
    get
    {
        if (trackSel == null)
            trackSel =
                GetService(typeof(STrackSelection)) as ITrackSelection;
        return trackSel;
    }
}

public void UpdateSelection()
{
    ITrackSelection track = TrackSelection;
    if (track != null)
        track.OnSelectChange((ISelectionContainer)selContainer);
}

public void SelectList(ArrayList list)
{
    selContainer = new SelectionContainer(true, false);
    selContainer.SelectableObjects = list;
    selContainer.SelectedObjects = list;
    UpdateSelection();
}

public override void OnToolWindowCreated()
{
    ArrayList listObjects = new ArrayList();
    listObjects.Add(this);
    SelectList(listObjects);
}

```

La `TrackSelection` propiedad `GetService` se `STrackSelection` utiliza para obtener `ITrackSelection` un servicio, que proporciona una interfaz. El `OnToolWindowCreated` controlador `SelectList` de eventos y el método juntos crean una lista de objetos seleccionados que contiene solo el propio objeto de panel de la ventana de herramientas. El `UpdateSelection` método indica a la ventana **Propiedades** que muestre las propiedades públicas del panel de la ventana de herramientas.

6. Compile la solución y comience la depuración. Debería aparecer la instancia experimental de Visual Studio.
7. Si la ventana **Propiedades** no está visible, ábrala pulsando F4.
8. Abra la ventana **MyToolWindow**. Puede encontrarlo en **Ver > otras ventanas**.

Se abre la ventana y las propiedades públicas del panel de la ventana aparecen en la ventana **Propiedades**.

9. Cambie la propiedad **Título** de la ventana **Propiedades** a **Propiedades de mi objeto**.

El título de la ventana `MyToolWindow` cambia en consecuencia.

Exponer las propiedades de la ventana de herramientas

En esta sección, agregará una ventana de herramientas y expone sus propiedades. Los cambios realizados en las propiedades se reflejan en la ventana **Propiedades**.

Para exponer las propiedades de la ventana de herramientas

1. Abra `MyToolWindow.cs` agregue la propiedad booleana pública `IsChecked` a la `MyToolWindow` clase.

```
[Category("My Properties")]
[Description("MyToolWindowControl properties")]
public bool IsChecked
{
    get {
        if (base.Content == null) return false;
        return (bool)(( MyToolWindowControl) base.Content).checkBox.IsChecked;
    }
    set {
        ((MyToolWindowControl) base.Content).checkBox.IsChecked = value;
    }
}
```

Esta propiedad obtiene su estado de la casilla de verificación WPF que creará más adelante.

- Abra *MyToolWindowControl.xaml.cs* y reemplace el constructor *MyToolWindowControl* por el código siguiente.

```
private MyToolWindow pane;
public MyToolWindowControl(MyToolWindow pane)
{
    InitializeComponent();
    this.pane = pane;
    checkBox.IsChecked = false;
}
```

Esto `MyToolWindowControl` da acceso `MyToolWindow` al panel.

- En *MyToolWindow.cs*, `MyToolWindow` cambie el constructor de la siguiente manera:

```
base.Content = new MyToolWindowControl(this);
```

- Cambie a la vista de diseño de *MyToolWindowControl*.
- Elimine el botón y agregue una casilla de verificación del Cuadro de **herramientas** a la esquina superior izquierda.
- Agregue los eventos *Checked* y *Unchecked*. Seleccione la casilla de verificación en la vista de diseño. En la ventana **Propiedades**, haga clic en el botón Controladores de eventos (en la parte superior derecha de la ventana **Propiedades**). Busque **Checked** y escriba `checkbox_Checked` en el cuadro de texto y, a continuación, busque **Unchecked** y escriba `checkbox_Unchecked` en el cuadro de texto.
- Agregue los controladores de eventos de casilla de verificación:

```
private void checkBox_Checked(object sender, RoutedEventArgs e)
{
    pane.IsChecked = true;
    pane.UpdateSelection();
}
private void checkBox_Unchecked(object sender, RoutedEventArgs e)
{
    pane.IsChecked = false;
    pane.UpdateSelection();
}
```

- Compile la solución y comience la depuración.
- En la instancia experimental, abra la ventana **MyToolWindow**.

Busque las propiedades de la ventana en la ventana **Propiedades**. La propiedad **IsChecked** aparece en la parte inferior de la ventana, en la categoría **Mis propiedades**.

10. Active la casilla de verificación en la ventana **MyToolWindow**. **IsChecked** en la ventana **Propiedades** cambia a **True**. Desactive la casilla de verificación de la ventana **MyToolWindow**. **IsChecked** en la ventana **Propiedades** cambia a **False**. Cambie el valor de **IsChecked** en la ventana **Propiedades**. La casilla de verificación de la ventana **MyToolWindow** cambia para que coincida con el nuevo valor.

NOTE

Si debe eliminar un objeto que se muestra en `OnSelectChange` la `null` ventana **Propiedades**, llame primero con un contenedor de selección. Después de desechar la propiedad u objeto, `SelectableObjects` puede `SelectedObjects` cambiar a un contenedor de selección que se haya actualizado y enumera.

Cambiar listas de selección

En esta sección, agregará una lista de selección para una clase de propiedad básica y utilizará la interfaz de la ventana de herramientas para elegir la lista de selección que desea mostrar.

Para cambiar las listas de selección

1. Abra `MyToolWindow.cs` y agregue `Simple` una clase pública denominada .

```
public class Simple
{
    private string someText = "";

    [Category("My Properties")]
    [Description("Simple Properties")]
    [DisplayName("My Text")]
    public string SomeText
    {
        get { return someText; }
        set { someText = value; }
    }

    [Category("My Properties")]
    [Description("Read-only property")]
    public bool ReadOnly
    {
        get { return false; }
    }
}
```

2. Agregue `SimpleObject` una propiedad `MyToolWindow` a la clase, además de dos métodos para cambiar la selección de la ventana **Propiedades** entre el panel de la ventana y el `Simple` objeto.

```

private Simple simpleObject = null;
public Simple SimpleObject
{
    get
    {
        if (simpleObject == null) simpleObject = new Simple();
        return simpleObject;
    }
}

public void SelectSimpleList()
{
    ArrayList listObjects = new ArrayList();
    listObjects.Add(SimpleObject);
    SelectList(listObjects);
}

public void SelectThisList()
{
    ArrayList listObjects = new ArrayList();
    listObjects.Add(this);
    SelectList(listObjects);
}

```

3. En *MyToolWindowControl.cs*, reemplace los controladores de casilla de verificación por estas líneas de código:

```

private void checkbox_Checked(object sender, RoutedEventArgs e)
{
    pane.IsChecked = true;
    pane.SelectSimpleList();
    pane.UpdateSelection();
}
private void checkbox_Unchecked(object sender, RoutedEventArgs e)
{
    pane.IsChecked = false;
    pane.SelectThisList();
    pane.UpdateSelection();
}

```

4. Compile la solución y comience la depuración.
5. En la instancia experimental, abra la ventana **MyToolWindow**.
6. Active la casilla de verificación en la ventana **MyToolWindow**. La **Properties** ventana Propiedades Simple muestra las propiedades del objeto, **SomeText** y **ReadOnly**. Desactive la casilla. Las propiedades públicas de la ventana aparecen en la ventana **Propiedades**.

NOTE

El nombre para mostrar de **SomeText** es **Mi texto**.

Procedimiento recomendado

En este [ISelectionContainer](#) tutorial, se implementa para que la colección de objetos seleccionables y la colección de objetos seleccionados sean la misma colección. Solo el objeto seleccionado aparece en la lista Navegador de propiedades. Para obtener una implementación más completa de [ISelectionContainer](#), vea el [Reference.ToolWindow](#) ejemplos.

Las ventanas de herramientas de Visual Studio persisten entre sesiones de Visual Studio. Para obtener más

información sobre cómo [ProvideProfileAttribute](#) conservar el estado de la ventana de herramientas, consulte .

Vea también

- [Extender propiedades y la ventana Propiedad](#)

Obtener propiedades del proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo mostrar las propiedades del proyecto en una ventana de herramientas.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Para crear un proyecto VSIX y agregar una ventana de herramientas

1. Cada extensión de Visual Studio comienza con un proyecto de implementación de VSIX, que contendrá los activos de extensión. Cree Visual Studio un proyecto `ProjectPropertiesExtension` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Agregue una ventana de herramientas agregando `ProjectPropertiesToolWindow` una plantilla de elemento de ventana de herramientas personalizada denominada . En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de elementos > de Visual C.** y seleccione **Ventana de herramientas personalizadas**. En el campo **Nombre** en la parte inferior `ProjectPropertiesToolWindow.cs` del cuadro de diálogo, cambie el nombre del archivo a . Para obtener más información sobre cómo crear una ventana de herramientas personalizada, consulte [Crear una extensión con una ventanade herramientas](#) .
3. Compile la solución y compruebe que se compila sin errores.

Para mostrar las propiedades del proyecto en una ventana de herramientas

1. En el archivo `ProjectPropertiesToolWindowCommand.cs`, agregue las siguientes directivas using.

```
using EnvDTE;  
using System.Windows.Controls;
```

2. En `ProjectPropertiesToolWindowControl.xaml`, quite el botón existente y agregue un `TreeView` desde el cuadro de herramientas. También puede quitar el controlador de eventos click del archivo `ProjectPropertiesToolWindowControl.xaml.cs`.
3. En `ProjectPropertiesToolWindowCommand.cs`, `ShowToolWindow()` utilice el método para abrir el proyecto y leer sus propiedades y, a continuación, agregue las propiedades a `TreeView`. El código de `ShowToolWindow` debe tener el siguiente aspecto:

```

private void ShowToolWindow(object sender, EventArgs e)
{
    ToolWindowPane window = this.package.FindToolWindow(typeof(ProjectPropertiesToolWindow), 0, true);
    if ((null == window) || (null == window.Frame))
    {
        throw new NotSupportedException("Cannot create window.");
    }
    IVsWindowFrame windowFrame = (IVsWindowFrame)window.Frame;
    Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(windowFrame.Show());

    // Get the tree view and populate it if there is a project open.
    ProjectPropertiesToolWindowControl control = (ProjectPropertiesToolWindowControl)window.Content;
    TreeView treeView = control.treeView;

    // Reset the TreeView to 0 items.
    treeView.Items.Clear();

    DTE dte = (DTE)this.ServiceProvider.GetService(typeof(DTE));
    Projects projects = dte.Solution.Projects;
    if (projects.Count == 0) // no project is open
    {
        TreeViewItem item = new TreeViewItem();
        item.Name = "Projects";
        item.ItemsSource = new string[]{ "no projects are open." };
        item.IsExpanded = true;
        treeView.Items.Add(item);
        return;
    }

    Project project = projects.Item(1);
    TreeViewItem item1 = new TreeViewItem();
    item1.Header = project.Name + "Properties";
    treeView.Items.Add(item1);

    foreach (Property property in project.Properties)
    {
        TreeViewItem item = new TreeViewItem();
        item.ItemsSource = new string[] { property.Name };
        item.IsExpanded = true;
        treeView.Items.Add(item);
    }
}

```

4. Compile la solución y comience la depuración. Debería aparecer la instancia experimental.

5. En la instancia experimental, abra un proyecto.

6. En **Ver > otras ventanas**, haga clic en **ProjectPropertiesToolWindow**.

Debería ver el control de árbol en la ventana de herramientas junto con el nombre del primer proyecto y de todas sus propiedades del proyecto.

Ampliar otras partes de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

Hay muchas más partes de la interfaz de usuario de Visual Studio que puede ampliar. Aquí te mostramos sólo unos cuantos.

Crear un VSPackage

Los bloques de creación básicos de la extensibilidad de Visual Studio son VSPackages. Obtenga información sobre cómo agregar un VSPackage: [crear una extensión con un VSPackage](#)

Ampliar la caja de herramientas

Obtenga información sobre cómo agregar nuevos controles y otros elementos al cuadro de herramientas y cómo usar la funcionalidad de Toolbox:

- [Crear un control de cuadro de herramientas de WPF](#)
- [Crear un control de cuadro de herramientas de formularios Windows Forms](#)

Ampliar la barra de estado

Aprenda a leer y escribir en la barra de estado y la barra de progreso, y cómo proporcionar animaciones y otra interfaz de usuario: [Ampliar la barra de estado](#).

Crear páginas de inicio personalizadas

Aprenda a crear su propia página de inicio, ya sea desde cero o desde un ejemplo de página de inicio descargable: [Crear una página de inicio personalizada](#).

Escribir en el registro de actividad

Obtenga información sobre cómo escribir en el registro de actividad: [Cómo: usar el registro de actividad](#).

Crear una extensión con un VSPackage

06/04/2020 • 4 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo crear un proyecto VSIX y agregar un elemento de proyecto de VSPackage. Usaremos el VSPackage para obtener el servicio Del Shell con el fin de mostrar un cuadro de mensaje.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

Crear un VSPackage

1. Cree un proyecto VSIX denominado **FirstPackage**. Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Cuando se abra el proyecto, agregue una plantilla de elemento de paquete de Visual Studio denominada **FirstPackage**. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C#** y seleccione **Paquete de Visual Studio**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a *FirstPackage.cs*.

3. Compile la solución y comience la depuración.

Aparece la instancia experimental de Visual Studio. Para obtener más información acerca de la instancia experimental, consulte [La instancia experimental](#).

4. En la instancia experimental, abra la ventana **Extensiones y actualizaciones de herramientas**. > Debería ver la extensión **FirstPackage** aquí. (Si abre **Extensiones y actualizaciones** en la instancia de trabajo de Visual Studio, no verá **FirstPackage**).

Cargar el VSPackage

En este momento, la extensión no se carga porque no hay nada que haga que se cargue. Por lo general, puede cargar una extensión cuando interactúa con su interfaz de usuario (haciendo clic en un comando de menú, abrir una ventana de herramientas) o especificando que el VSPackage debe cargarse en un contexto de interfaz de usuario específico. Para obtener más información acerca de cómo cargar VSPackages y contextos de interfaz de usuario, vea [cargar VSPackages](#). Para este procedimiento, le mostraremos cómo cargar un VSPackage cuando una solución está abierta.

1. Abra el archivo *FirstPackage.cs*. Busque la declaración `FirstPackage` de la clase. Sustituya los atributos existentes por los siguientes atributos:

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[InstalledProductRegistration("#110", "#112", "1.0", IconResourceID = 400)] // Info on this package
for Help/About
[ProvideAutoLoad(UIContextGuids80.SolutionExists)]
[Guid(FirstPackage.PackageGuidString)]
public sealed class FirstPackage : Package
```

2. Vamos a agregar un mensaje que nos permite saber que el VSPackage se ha cargado. Usamos el método

`Initialize()` de `VSPackage` para hacer esto, porque puede obtener servicios de Visual Studio solo después de que el `VSPackage` se ha sitio. (Para obtener más información acerca de cómo obtener servicios, consulte [Cómo: Obtener un servicio](#).) Reemplace `Initialize()` el `FirstPackage` método de código `SVsUIShell` que obtiene el `IVsUIShell` servicio, obtiene `ShowMessageBox` la interfaz y llama a su método.

```
protected override void Initialize()
{
    base.Initialize();

    IVsUIShell uiShell = (IVsUIShell)GetService(typeof(SVsUIShell));
    Guid clsid = Guid.Empty;
    int result;
    Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(uiShell.ShowMessageBox(
        0,
        ref clsid,
        "FirstPackage",
        string.Format(CultureInfo.CurrentCulture, "Inside {0}.Initialize()",
            this.GetType().FullName),
        string.Empty,
        0,
        OLEMSGBUTTON.OLEMSGBUTTON_OK,
        OLEMSGDEFBUTTON.OLEMSGDEFBUTTON_FIRST,
        OLEMSGICON.OLEMSGICON_INFO,
        0,
        out result));
}
```

3. Compile la solución y comience la depuración. Aparece la instancia experimental.
4. Abra una solución en la instancia experimental. Debería ver un cuadro de mensaje que dice **First Package Inside Initialize()**.

Crear un control de cuadro de herramientas de WPF

06/04/2020 • 6 minutes to read • [Edit Online](#)

La plantilla Control de cuadro de herramientas de WPF (Windows Presentation Framework) permite crear controles WPF que se agregan automáticamente al cuadro de **herramientas** cuando se instala la extensión. En este tutorial se muestra cómo usar la plantilla para crear un control **de cuadro** de herramientas que puede distribuir a otros usuarios.

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear el control de cuadro de herramientas

Cree una extensión con un control DE cuadro de herramientas de WPF

1. Cree un proyecto `MyToolboxControl` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Cuando se abra el proyecto, agregue `MyToolboxControl` una plantilla de elemento Control de cuadro de herramientas de WPF denominada . En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento** , vaya a **Extensibilidad de Visual C- >** y seleccione **Control de cuadro de herramientas de WPF**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a *MyToolboxControl.cs*.

La solución ahora contiene `ProvideToolboxControlAttribute` [RegistrationAttribute](#) un control de usuario, un que agrega el control al cuadro de **herramientas** y una entrada **Microsoft.VisualStudio.ToolboxControl** Asset en el manifiesto VSIX para la implementación.

Para crear la interfaz de usuario del control

1. Abra *MyToolboxControl.xaml* en el diseñador.

El diseñador muestra un control [Grid](#) que contiene un control [Button](#).

2. Organice el diseño de cuadrícula. Al seleccionar [Grid](#) el control, aparecen barras de control azules en los bordes superior e izquierdo de la cuadrícula. Puede agregar filas y columnas a la cuadrícula haciendo clic en las barras.
3. Agregue controles secundarios a la cuadrícula. Puede colocar un control secundario arrastrándolo desde el **cuadro** de herramientas `Grid.Row` `Grid.Column` a una sección de la cuadrícula o estableciendo sus atributos y sus atributos en el XAML. En el ejemplo siguiente se agregan dos etiquetas en la fila superior de la cuadrícula y un botón en la segunda fila.

```
<Grid>
  <Label Grid.Row="0" Grid.Column="0" Name="label1" />
  <Label Grid.Row="0" Grid.Column="1" Name="label2" />
  <Button Name="button1" Click="button1_Click" Grid.Row="1" Grid.ColumnSpan="2" />
</Grid>
```

Cambiar el nombre del control

De forma predeterminada, el control aparecerá en el cuadro de **herramientas** como **MyToolboxControl** en un

grupo denominado **MyToolboxControl.MyToolboxControl**. Puede cambiar estos nombres en el archivo *MyToolboxControl.xaml.cs*.

1. Abra *MyToolboxControl.xaml.cs* en la vista de código.
2. Busque `MyToolboxControl` la clase y cámbiele el nombre a **TestControl**. (La forma más rápida de hacerlo es cambiar el nombre de la clase, luego seleccione **Cambiar nombre** en el menú contextual y completar los pasos. (Para obtener más información acerca del comando **Cambiar nombre**, vea [Cambiar el nombre de la refactorización \(C-\)](#).)
3. Vaya al `ProvideToolboxControl` atributo y cambie el valor del primer parámetro a **Test**. Este es el nombre del grupo que contendrá el control en el cuadro de **herramientas**.

El código resultante debería tener este aspecto:

```
[ProvideToolboxControl("Test", true)]
public partial class TestControl : UserControl
{
    public TestControl()
    {
        InitializeComponent();
    }
}
```

Compilación, prueba e implementación

Al depurar el proyecto, debe encontrar el control instalado en el cuadro de **herramientas** de la instancia experimental de Visual Studio.

Para compilar y probar el control

1. Vuelva a generar el proyecto e inicie la depuración.
2. En la nueva instancia de Visual Studio, cree un proyecto de aplicación WPF. Asegúrate de que el Diseñador XAML está abierto.
3. Busque el control en el **Cuadro de herramientas** y arrástrelo a la superficie de diseño.
4. Inicie la depuración de la aplicación WPFWPF.
5. Compruebe que aparece el control.

Para implementar el control

1. Después de compilar el proyecto probado, puede encontrar el archivo *.vsix* en la carpeta **-bin-debug** del proyecto.
2. Puede instalarlo en un equipo local haciendo doble clic en el archivo *.vsix* y siguiendo el procedimiento de instalación. Para desinstalar el control, vaya a **> Extensiones de herramientas y actualizaciones** y busque la extensión de control y, a continuación, haga clic en **Desinstalar**.
3. Cargue el archivo *.vsix* en una red o en un sitio Web.

Si carga el archivo en el sitio Web de [Visual Studio Marketplace](#), otros usuarios pueden usar **extensiones > y actualizaciones** de herramientas en Visual Studio para buscar el control en línea e instalarlo.

Crear un control de cuadro de herramientas de formularios Windows Forms

06/04/2020 • 12 minutes to read • [Edit Online](#)

La plantilla de elemento Control de cuadro de herramientas de Windows Forms que se incluye en las herramientas de extensibilidad de Visual Studio (VS SDK), permite crear un control **de cuadro** de herramientas que se agrega automáticamente cuando se instala la extensión. En este tutorial se muestra cómo usar la plantilla para crear un control de contador simple que puede distribuir a otros usuarios.

Prerrequisitos

A partir de Visual Studio 2015, no se instala el SDK de Visual Studio desde el centro de descarga. Se incluye como una característica opcional en la configuración de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK](#) de Visual Studio .

Crear el control de cuadro de herramientas

La plantilla Control de cuadro de herramientas de formularios Windows Forms crea un control de usuario indefinido y proporciona toda la funcionalidad necesaria para agregar el control al **cuadro de herramientas**.

Cree una extensión con un control de cuadro de herramientas de formularios Windows Forms

1. Cree un proyecto `MyWinFormsControl` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto**, buscando "vsix".
2. Cuando se abra el proyecto, agregue una `Counter` plantilla de elemento Control de cuadro de herramientas de Formularios Windows Forms denominada . En el **Explorador** de soluciones , haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C-** > y seleccione **Control de cuadro de herramientas de formularios Windows Forms**
3. Esto agrega un control `ProvideToolboxControlAttribute` [RegistrationAttribute](#) de usuario, un para colocar el control en el cuadro de **herramientas** y un `Microsoft.VisualStudio.ToolboxControl` Asset entrada en el manifiesto VSIX para la implementación.

Crear una interfaz de usuario para el control

El `Counter` control requiere dos controles [Label](#) secundarios: a para [Button](#) mostrar el recuento actual y a para restablecer el recuento a 0. No se requiere ningún otro control secundario porque los llamadores incrementarán el contador mediante programación.

Para crear la interfaz de usuario

1. En el **Explorador** de soluciones, haga doble clic en `Counter.cs` para abrirlo en el diseñador.
2. Retire el **haga clic aquí !** botón que se incluye de forma predeterminada al agregar la plantilla de elemento Control de cuadro de herramientas de Windows Forms.
3. Desde el **cuadro** `Label` de herramientas `Button` , arrastre un control y, a continuación, un control debajo de él a la superficie de diseño.
4. Cambie el tamaño del control de usuario general a 150, 50 píxeles y cambie el tamaño del control de botón a 50, 20 píxeles.

5. En la ventana **Propiedades**, establezca los siguientes valores para los controles de la superficie de diseño.

CONTROL	PROPIEDAD	VALUE
	Texto	""
	Nombre	btnReset
	Texto	Reset

Codificar el control de usuario

El  control expondrá un método para incrementar el contador, un evento que se generará siempre que se incremente el contador, un botón **Restablecer** y tres propiedades para almacenar el recuento actual, el texto para mostrar y si se va a mostrar u ocultar el botón **Restablecer**. El atributo  determina la ubicación del **Cuadro de herramientas** donde se mostrará el control .

Para codificar el control de usuario

1. Haga doble clic en el formulario para abrir su controlador de eventos de carga en la ventana de código.
2. Por encima del método de controlador de eventos, en la clase de control cree un entero para almacenar el valor del contador y una cadena para almacenar el texto para mostrar como se muestra en el ejemplo siguiente.

```
int currentValue;  
string displayText;
```

3. Cree las siguientes declaraciones de propiedad pública.

```
public int Value {  
    get { return currentValue; }  
}  
  
public string Message {  
    get { return displayText; }  
    set { displayText = value; }  
}  
  
public bool ShowReset {  
    get { return btnReset.Visible; }  
    set { btnReset.Visible = value; }  
}
```

Los llamadores pueden acceder a estas propiedades para obtener y establecer el texto de visualización del contador y para mostrar u ocultar el botón **Restablecer**. Los llamadores pueden obtener el  valor actual de la propiedad de solo lectura, pero no pueden establecer el valor directamente.

4. Coloque el código  siguiente en el evento para el control.

```
private void Counter_Load(object sender, EventArgs e)  
{  
    currentValue = 0;  
    label1.Text = Message + Value;  
}
```

Establecer **Label** el texto **Load** Etiqueta en el evento permite que las propiedades de destino se carguen antes de aplicar sus valores. Establecer el texto **Label** en el constructor daría como resultado un **Label** vacío.

5. Cree el siguiente método público para incrementar el contador.

```
public void Increment()
{
    currentValue++;
    label1.Text = displayText + Value;
    Incremented(this, EventArgs.Empty);
}
```

6. Agregue una declaración para el **Incremented** evento a la clase de control.

```
public event EventHandler Incremented;
```

Los llamadores pueden agregar controladores a este evento para responder a los cambios en el valor del contador.

7. Vuelva a la vista de **Reset** diseño y **btnReset_Click** haga doble clic en el botón Restablecer para generar el controlador de eventos y, a continuación, rellénelo como se muestra en el ejemplo siguiente.

```
private void btnReset_Click(object sender, EventArgs e)
{
    currentValue = 0;
    label1.Text = displayText + Value;
}
```

8. Justo encima de la definición de clase en la declaración de atributos de **ProvideToolboxControl**, cambie el valor del primer parámetro de **"MyWinFormsControl.Counter"** a **"General"**. Se establece el nombre del grupo de elementos que va a hospedar el control en el **Cuadro de herramientas**.

En el ejemplo siguiente se muestra el atributo **ProvideToolboxControl** y la definición de clase ajustada.

```
[ProvideToolboxControl("General", false)]
public partial class Counter : UserControl
```

Prueba del control

Para probar un control **Toolbox**, pruébelo primero en el entorno de desarrollo y, a continuación, pruébelo en una aplicación compilada.

Para probar el control

1. Presione **F5** para **iniciar la depuración**.

Este comando compila el proyecto y abre una segunda instancia experimental de Visual Studio que tiene el control instalado.

2. En la instancia experimental de Visual Studio, cree un proyecto de aplicación de **Windows Forms**.
3. En el **Explorador** de soluciones, haga doble clic en **Form1.cs** para abrirlo en el diseñador si aún no está abierto.
4. En el cuadro **Counter** de **herramientas**, el control debe mostrarse en la sección **General**.
5. Arrastre **Counter** un control al formulario y, a continuación, selecciónelo. Las **Value** **Message** propiedades

`ShowReset` , , y se mostrarán en la ventana **Propiedades**, junto con las propiedades heredadas de `UserControl`.

6. Establezca la propiedad `Message` en `Count:` .
7. Arrastre `Button` un control al formulario y, a continuación, establezca `Test` las propiedades de nombre y texto del botón en .
8. Haga doble clic en el botón para abrir `Form1.cs` en la vista de código y crear un controlador de clics.
9. En el controlador `counter1.Increment()` de clics, llame a .
10. En la función constructora, `counter1``.``Incremented +=` después de la llamada a , escriba y, a `InitializeComponent` continuación, presione **Tab** dos veces.
Visual Studio genera un controlador `counter1.Incremented` de nivel de formulario para el evento.
11. Resalte `Throw` la instrucción en `mbox` el controlador de eventos, escriba y, a continuación, presione **Tab** dos veces para generar un cuadro de mensaje a partir del fragmento de código mbox.
12. En la siguiente línea, `if` / `else` agregue el siguiente bloque para establecer la visibilidad del botón **Restablecer**.

```
if (counter1.Value < 5) counter1.ShowReset = false;  
else counter1.ShowReset = true;
```

13. Presione **F5**.
Se abre el formulario. El `Counter` control muestra el texto siguiente.
Recuento: 0
14. Haga clic en **Probar**.
El contador aumenta y Visual Studio muestra un cuadro de mensaje.
15. Cierre el cuadro de mensaje.
El botón **Restablecer** desaparece.
16. Haga clic en **Probar** hasta que el contador alcance 5 cerrando los cuadros de mensaje cada vez.
El botón **Restablecer** vuelve a aparecer.
17. Haga clic en **Restablecer**.
El contador se restablece a **0**.

Pasos siguientes

Al compilar un control **de cuadro** de herramientas, Visual Studio crea un archivo denominado `ProjectName.vsix` en la carpeta . Puede implementar el control cargando el archivo `.vsix` en una red o en un sitio Web. Cuando un usuario abre el archivo `.vsix`, el control se instala y se agrega al cuadro de **herramientas** de Visual Studio en el equipo del usuario. Como alternativa, puede cargar el archivo `.vsix` en [Visual Studio Marketplace](#) para que los usuarios puedan encontrarlo examinando en el cuadro de diálogo **Extensiones y actualizaciones de herramientas**. >

Vea también

- [Ampliar otras partes de Visual Studio](#)

- Crear un control de cuadro de herramientas de WPF
- Ampliar otras partes de Visual Studio
- Los formularios Windows Forms controlan los conceptos básicos de desarrollo

Ampliar la barra de estado

06/04/2020 • 7 minutes to read • [Edit Online](#)

Puede usar la barra de estado de Visual Studio en la parte inferior del IDE para mostrar información.

Al ampliar la barra de estado, puede mostrar información e interfaz de usuario en cuatro regiones: la región de comentarios, la barra de progreso, la región de animación y la región del diseñador. La región de comentarios le permite mostrar texto y resaltar el texto mostrado. La barra de progreso muestra el progreso incremental de las operaciones de ejecución corta, como guardar un archivo. La región de animación muestra una animación en bucle continuo para operaciones de ejecución prolongada u operación de longitud indeterminada, como la creación de varios proyectos en una solución. Y la región del diseñador muestra el número de línea y columna de la ubicación del cursor.

Puede obtener la barra de [IVsStatusbar](#) estado mediante [SVsStatusbar](#) la interfaz (desde el servicio). Además, cualquier objeto situado en un marco de ventana puede [IVsStatusbarUser](#) registrarse como un objeto de cliente de barra de estado mediante la implementación de la interfaz. Cada vez que se activa una ventana, Visual Studio [IVsStatusbarUser](#) consulta el objeto incrustado en esa ventana para la interfaz. Si se encuentra, [SetInfo](#) llama al método en la interfaz devuelta y el objeto puede actualizar la barra de estado desde dentro de ese método. Las ventanas de documento, [SetInfo](#) por ejemplo, pueden usar el método para actualizar la información de la región del diseñador cuando se activan.

En los procedimientos siguientes se supone que entiende cómo crear un proyecto VSIX y agregar un comando de menú personalizado. Para obtener información, consulte [Crear una extensión con un comando de menú](#).

Modificar la barra de estado

Este procedimiento muestra cómo establecer y obtener texto, mostrar texto estático y resaltar el texto mostrado en la región de comentarios de la barra de estado.

Leer y escribir en la barra de estado

1. Cree un proyecto VSIX denominado **TestStatusBarExtension** y agregue un comando de menú denominado **TestStatusBarCommand**.
2. En *TestStatusBarCommand.cs*, reemplace el `MenuItemCallback` código del método del controlador de comandos () por lo siguiente:

```

private void MenuItemCallback(object sender, EventArgs e)
{
    IVsStatusbar statusBar = (IVsStatusbar)ServiceProvider.GetService(typeof(SVsStatusbar));

    // Make sure the status bar is not frozen
    int frozen;

    statusBar.IsFrozen(out frozen);

    if (frozen != 0)
    {
        statusBar FreezeOutput(0);
    }

    // Set the status bar text and make its display static.
    statusBar.SetText("We just wrote to the status bar.");

    // Freeze the status bar.
    statusBar FreezeOutput(1);

    // Get the status bar text.
    string text;
    statusBar.GetText(out text);
    System.Windows.Forms.MessageBox.Show(text);

    // Clear the status bar text.
    statusBar FreezeOutput(0);
    statusBar.Clear();
}

```

3. Compile el código e inicie la depuración.
4. Abra el menú **Herramientas** en la instancia experimental de Visual Studio. Haga clic en el botón **Invocar TestStatusBarCommand**.

Debería ver que el texto en la barra de estado ahora dice Acabamos de escribir en la barra de **estado**. y el cuadro de mensaje que aparece tiene el mismo texto.

Actualizar la barra de progreso

1. En este procedimiento vamos a mostrar cómo inicializar y actualizar la barra de progreso.
2. Abra *TestStatusBarCommand.cs* el archivo TestStatusBarCommand.cs `MenuItemCallback` y reemplace el método por el código siguiente:

```

private void MenuItemCallback(object sender, EventArgs e)
{
    IVsStatusbar statusBar = (IVsStatusbar)ServiceProvider.GetService(typeof(SVsStatusbar));
    uint cookie = 0;
    string label = "Writing to the progress bar";

    // Initialize the progress bar.
    statusBar.Progress(ref cookie, 1, "", 0, 0);

    for (uint i = 0, total = 20; i <= total; i++)
    {
        // Display progress every second.
        statusBar.Progress(ref cookie, 1, label, i, total);
        System.Threading.Thread.Sleep(1000);
    }

    // Clear the progress bar.
    statusBar.Progress(ref cookie, 0, "", 0, 0);
}

```

3. Compile el código e inicie la depuración.
4. Abra el menú **Herramientas** en la instancia experimental de Visual Studio. Haga clic en el botón **Invocar TestStatusBarCommand**.

Debería ver que el texto de la barra de estado ahora lee Escritura en la barra de **progreso**. También debería ver que la barra de progreso se actualiza cada segundo durante 20 segundos. Después de eso, se borran la barra de estado y la barra de progreso.

Mostrar una animación

1. La barra de estado muestra una animación de bucle que indica una operación de ejecución prolongada (por ejemplo, la creación de varios proyectos en una solución). Si no ve esta animación, asegúrese **Tools > de** que tiene la configuración correcta de **Opciones** de herramientas:

Vaya a la **pestaña Opciones > generales de herramientas >** y desactive **Ajustar automáticamente la experiencia visual en función del rendimiento del cliente**. A continuación, compruebe la subopción **Habilitar experiencia visual** de cliente enriquecido. Ahora debería poder ver la animación al compilar el proyecto en la instancia experimental de Visual Studio.

En este procedimiento se muestra la animación estándar de Visual Studio que representa la creación de un proyecto o solución.

2. Abra *TestStatusBarCommand.cs* el archivo TestStatusBarCommand.cs `MenuItemCallback` y reemplace el método por el código siguiente:

```
private void MenuItemCallback(object sender, EventArgs e)
{
    IVsStatusbar statusBar = (IVsStatusbar)ServiceProvider.GetService(typeof(SVsStatusbar));

    // Use the standard Visual Studio icon for building.
    object icon = (short)Microsoft.VisualStudio.Shell.Interop.Constants.SBAI_Build;

    // Display the icon in the Animation region.
    statusBar.Animation(1, ref icon);

    // The message box pauses execution for you to look at the animation.
    System.Windows.Forms.MessageBox.Show("showing?");

    // Stop the animation.
    statusBar.Animation(0, ref icon);
}
```

3. Compile el código e inicie la depuración.
4. Abra el menú **Herramientas** en la instancia experimental de Visual Studio y haga clic en **Invocar TestStatusBarCommand**.

Cuando vea el cuadro de mensaje, también debería ver la animación en la barra de estado en el extremo derecho. Cuando se descarta el cuadro de mensaje, la animación desaparece.

Creación de una página de inicio personalizada

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede crear una página de inicio personalizada siguiendo los pasos de este documento.

Crear una página de inicio en blanco

En primer lugar, cree una página de inicio en blanco mediante la creación de un archivo *.xaml* que tenga una estructura de etiquetas que Visual Studio reconocerá. A continuación, agregue marcado y código subyacente para producir la apariencia y la funcionalidad que desee.

1. Cree un nuevo proyecto del tipo **Aplicación WPF** (Escritorio de Windows de Visual C# . >
2. Agregue una referencia a `Microsoft.VisualStudio.Shell.14.0`.
3. Abra el archivo XAML en el editor <XML y cambie <el elemento de> Window de nivel superior a un elemento de> UserControl sin quitar ninguna de las declaraciones de espacio de nombres.
4. Quite `x:Class` la declaración del elemento de nivel superior. Esto hace que el contenido XAML sea compatible con la ventana de herramientas de Visual Studio que hospeda la página de inicio.
5. Agregue las siguientes declaraciones de <espacio de nombres al elemento> UserControl de nivel superior.

```
xmlns:vs="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Shell.14.0"
xmlns:vsfx="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.14.0"
```

Estos espacios de nombres le permiten tener acceso a los comandos, controles y configuración de la interfaz de usuario de Visual Studio. Para obtener más información, vea [Agregar comandos](#) de Visual Studio a una página de inicio .

En el ejemplo siguiente se muestra el marcado en el archivo *.xaml* para una página de inicio en blanco. Cualquier contenido personalizado debe [Grid](#) ir en el elemento interno.

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:MyNamespace="clr-namespace:ManualStartPage;assembly=ManualStartPage"
  xmlns:vs="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Shell.14.0"
  xmlns:vsfx="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.14.0"
  xmlns:local="clr-namespace:StartPageHost"
  mc:Ignorable="d"
  Height="350" Width="525">
  <Grid>
    <!--Add content here.-->
  </Grid>
</UserControl>
```

6. Agregue controles al <elemento> UserControl vacío para rellenar la página de inicio personalizada. Para obtener información acerca de cómo agregar funcionalidad específica de Visual Studio, vea [Agregar comandos](#) de Visual Studio a una página de inicio .

Pruebe y aplique la página de inicio personalizada

No establezca la instancia principal de Visual Studio para ejecutar la página de inicio personalizada hasta que compruebe que no se bloquea Visual Studio. En su lugar, pruébelo en la instancia experimental.

Para probar una página de inicio personalizada creada manualmente

1. Copie el archivo XAML y los archivos de texto o archivos de marcado compatibles en la carpeta
*%USERPROFILE%-Mis documentos-Visual Studio 2015-StartPages.\ *
2. Si la página de inicio hace referencia a los controles o tipos de ensamblados que Visual Studio no instala, copie los ensamblados y, a continuación, péguelos en la *carpeta**de instalación de Visual Studio.
3. En un símbolo del sistema de Visual Studio, escriba **devenv /rootsuffix Exp** para abrir una instancia experimental de Visual Studio.
4. En la instancia experimental, vaya a **Environment > la > páginaInicio** del entorno de **opciones > de herramientas** y seleccione el archivo XAML en el menú desplegable **Personalizar página de inicio**.
5. En el menú **Vista**, haga clic en **Página de inicio**.

Se debe mostrar la página de inicio personalizada. Si desea cambiar cualquier archivo, debe cerrar la instancia experimental, realizar los cambios, copiar y pegar los archivos modificados y, a continuación, volver a abrir la instancia experimental para ver los cambios.

Para aplicar la página de inicio personalizada en la instancia principal de Visual Studio

- Después de probar la página de inicio y encontrar que es estable, use la opción **Personalizar página** de inicio en el cuadro de diálogo **Opciones** para seleccionarla como la página de inicio en la instancia principal de Visual Studio

Vea también

- [Tutorial: Agregar XAML personalizado a la página de inicio](#)
- [Agregar control de usuario a la página de inicio](#)
- [Agregar comandos de Visual Studio a una página de inicio](#)
- [Tutorial: Guardar la configuración de usuario en una página de inicio](#)
- [Implementar páginas de inicio personalizadas](#)

Tutorial: Agregar XAML personalizado a la página de inicio

06/04/2020 • 3 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo crear una página de inicio de Visual Studio personalizada que contiene un explorador web.

Agregar XAML personalizado

1. Cree una página de inicio siguiendo las instrucciones de [Crear una página de inicio personalizada](#).
2. En el archivo *MainWindow.xaml*, busque la <sección Grid>.
3. Agregue <un elemento> <TabControl y < un> TabItem dentro del elemento Grid>, como se muestra en el ejemplo siguiente.

```
<Grid>
  <TabControl>
    <TabItem Header="Bing" Height="Auto">
      <Frame Source="http://www.bing.com" />
    </TabItem>
  </TabControl>
</Grid>
```

4. Agregue un <segundo> TabItem, con un <elemento Button> que abra un nuevo proyecto:

```
<Grid>
  <TabControl>
    <TabItem Header="MyButton" Height="Auto">
      <Button Name="btnNewProj" Content="New Project"
        Command="{x:Static vscom:VSCommands.ExecuteCommand}"
        CommandParameter="File.NewProject" >
      </Button>
    </TabItem>
    <TabItem Header="Bing" Height="Auto">
      <Frame Source="http://www.bing.com" />
    </TabItem>
  </TabControl>
</Grid>
```

Pruebe la página de inicio personalizada

1. Presione F5.

Se abre la instancia experimental de Visual Studio, con la página de inicio personalizada instalada pero no seleccionada.
2. En la instancia experimental de Visual Studio, abra la página **Herramientas /Opciones / Entorno**.
3. Seleccione **Inicio**. En la lista Personalizar página de inicio, seleccione el archivo *.xaml* y haga clic en **Aceptar**.
4. En el menú **Vista**, haga clic en **Página de inicio**.

5. Haga clic en la pestaña **Bing**.

Debería ver una página web de Bing.

6. Haga clic en la pestaña **MyButton**.

Debería ver un botón **MyProject**, que abre el cuadro de diálogo **Nuevo proyecto**.

7. Cierre la instancia experimental.

Para aplicar la página de inicio personalizada, en **Entorno de opciones** > de **herramientas** > , seleccione **Inicio**. En la lista Personalizar página de inicio, seleccione el archivo *.xaml* y haga clic en **Aceptar**.

Pasos siguientes

La página de inicio de Visual Studio ahora contiene una pestaña que muestra una pestaña del explorador web y una pestaña MyButton. Puede crear páginas de inicio personalizadas que tengan otras funciones mediante el modelo *de código subyacente* para agregar un archivo .dll personalizado, como se muestra en Agregar control de usuario a la página de inicio. Puede compartir páginas de inicio personalizadas con otros usuarios publicando el archivo .vsix resultante en el sitio Web de [Visual Studio Marketplace](#) o en otro sitio Web o recurso compartido de red. Para obtener más información, consulta [Deploying Custom Start Pages](#).

Vea también

- [Personalizar la página de inicio](#)
- [Controles de contenedor WPF](#)

Agregar control de usuario a la página de inicio

06/04/2020 • 8 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo agregar una referencia DLL a una página de inicio personalizada. En el ejemplo se agrega un control de usuario a la solución, se compila el control de usuario y, a continuación, se hace referencia al ensamblado compilado desde el archivo *.xaml* de la página de inicio. Una nueva pestaña hospeda el control de usuario, que funciona como un explorador web básico.

Puede usar el mismo proceso para agregar cualquier ensamblado al que se pueda llamar desde un archivo *.xaml*.

Agregar un control de usuario WPF wpf a la solución

En primer lugar, agregue un control de usuario de Windows Presentation Foundation (WPF) a la solución de página de inicio.

1. Cree una página de inicio mediante la creación en [Crear una página de inicio personalizada](#).
2. En el **Explorador de soluciones**, haga clic con el botón secundario en la solución, haga clic en **Agregar**, a continuación, haga clic en **Nuevo proyecto**.
3. En el panel izquierdo del cuadro de diálogo **Nuevo proyecto**, expanda el nodo **Visual Basic** o **Visual C#** y haga clic en **Windows**. En el panel central, seleccione Biblioteca de controles de usuario de WPF.
4. Asigne un `WebUserControl1` nombre al control y, a continuación, haga clic en **Aceptar**.

Implementar el control de usuario

Para implementar un control de usuario de WPF, compile la interfaz de usuario (UI) en XAML y, a continuación, escriba los eventos de código subyacente en C# u otro lenguaje .NET.

Para escribir el XAML para el control de usuario

1. Abra el archivo XAML para el control de usuario. En `<Grid>` el elemento, agregue las siguientes definiciones de fila al control.

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*" />
</Grid.RowDefinitions>
```

2. En el `<Grid>` elemento principal, `<Grid>` agregue el siguiente elemento nuevo, que contiene un cuadro de texto para escribir direcciones Web y un botón para establecer la nueva dirección.

```
<Grid Grid.Row="0">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="Auto" />
    </Grid.ColumnDefinitions>
    <TextBox x:Name="UserSource" Grid.Column="0" />
    <Button Grid.Column="1" x:Name="SetButton" Content="Set Address" Click="SetButton_Click" />
</Grid>
```

3. Agregue el siguiente marco al `<Grid>` elemento de `<Grid>` nivel superior justo después del elemento que

contiene el botón y el cuadro de texto.

```
<Frame Grid.Row="1" x:Name="WebFrame" Source="http://www.bing.com" Navigated="WebFrame_Navigated" />
```

4. En el ejemplo siguiente se muestra el XAML completado para el control de usuario.

```
<UserControl x:Class="WebUserControl.UserControl1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid Grid.Row="0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <TextBox x:Name="UserSource" Grid.Column="0" />
            <Button Grid.Column="1" x:Name="SetButton" Content="Set Address" Click="SetButton_Click" />
        </Grid>
        <Frame Grid.Row="1" x:Name="WebFrame" Source="http://www.bing.com"
        Navigated="WebFrame_Navigated" />
    </Grid>
</UserControl>
```

Para escribir los eventos de código subyacente para el control de usuario

1. En el diseñador XAML, haz doble clic en el botón Establecer dirección que agregaste al control.

El archivo *UserControl1.cs* se abre en el editor de código.

2. Rellene el controlador de eventos SetButton_Click de la siguiente manera.

```
private void SetButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        this.WebFrame.Source = new Uri(this.UserSource.Text, UriKind.Absolute);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}
```

Este código establece la dirección Web que se escribe en el cuadro de texto como destino para el explorador web. Si la dirección no es válida, el código produce un error.

3. También debe controlar el evento WebFrame_Navigated:

```
private void WebFrame_Navigated(object sender, EventArgs e)
{ }
```

4. Compile la solución.

Agregue el control de usuario a la página de inicio

Para que este control esté disponible para el proyecto de página de inicio, en el archivo de proyecto de página de inicio, agregue una referencia a la nueva biblioteca de controles. A continuación, puede agregar el control al marcado XAML de la página de inicio.

1. En el **Explorador de soluciones**, en el proyecto **Página de inicio**, haga clic con el botón secundario en **Referencias** y, a continuación, haga clic en **Agregar referencia**.
2. En la pestaña **Proyectos**, seleccione **WebUserControl** y, a continuación, haga clic en **Aceptar**.
3. En el menú **Compilar**, haga clic en **Compilar solución**.

La creación de la solución hace que el control de usuario esté disponible para IntelliSense para otros archivos de la solución.

Para agregar el control al marcado XAML de la página de inicio, agregue una referencia de espacio de nombres al ensamblado y, a continuación, coloque el control en la página.

Para agregar el control al marcado

1. En el Explorador de **soluciones**, abra el archivo *.xaml* de la página de inicio.
2. En el panel **XAML**, agregue la siguiente **Grid** declaración de espacio de nombres al elemento de nivel superior.

```
xmlns:vsc="clr-namespace:WebUserControl;assembly=WebUserControl"
```

3. En el panel **XAML**, desplázate hasta la <sección Cuadrícula>.

La sección **TabControl** contiene un **Grid** elemento de un elemento.

4. Agregue <un TabControl> <elemento que contiene un TabItem> que contiene una referencia al control de usuario.

```
<TabItem Header="Web" Height="Auto">
    <vsc:UserControl1 />
</TabItem>
```

Ahora puede probar el control.

Pruebe una página de inicio personalizada creada manualmente

1. Copie el archivo XAML y los archivos de texto o archivos de marcado compatibles en la carpeta
*%USERPROFILE%-Mis documentos-Visual Studio 2015-StartPages.\ *
2. Si la página de inicio hace referencia a los controles o tipos de ensamblados que Visual Studio no instala, copie los ensamblados y péguelos en la carpeta de instalación de *Visual Studio*.\
3. En un símbolo del sistema de Visual Studio, escriba **devenv /rootsuffix Exp** para abrir una instancia experimental de Visual Studio.
4. En la instancia experimental, vaya a **Environment > la > páginaInicio** del entorno de **opciones > de herramientas** y seleccione el archivo XAML en el menú desplegable **Personalizar página de inicio**.
5. En el menú **Vista**, haga clic en **Página de inicio**.

Se debe mostrar la página de inicio personalizada. Si desea cambiar cualquier archivo, debe cerrar la instancia experimental, realizar los cambios, copiar y pegar los archivos modificados y, a continuación, volver a abrir la instancia experimental para ver los cambios.

Vea también

- [Controles de contenedor WPF](#)
- [Tutorial: Agregar XAML personalizado a la página de inicio](#)

Agregar comandos de Visual Studio a una página de inicio

06/04/2020 • 5 minutes to read • [Edit Online](#)

Al crear una página de inicio personalizada, puede agregarle comandos de Visual Studio. En este documento se describen las diferentes formas de enlazar comandos de Visual Studio a objetos XAML en una página de inicio.

Para obtener más información acerca de los comandos en XAML, vea [Información general sobre comandos](#)

Añadir comandos desde el pozo de comandos

La página de inicio creada en [Microsoft.VisualStudio.PlatformUI](#) Crear [Microsoft.VisualStudio.Shell](#) una página de [inicio personalizada](#) agregó los espacios de nombres y, como se indica a continuación.

```
xmlns:vs="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Shell.14.0"
xmlns:vsfx="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.14.0"
```

Agregue otro espacio de nombres para Microsoft.VisualStudio.Shell desde el ensamblado *Microsoft.VisualStudio.Shell.Immutable.11.0.dll*. (Es posible que deba agregar una referencia a este ensamblado en el proyecto.)

```
xmlns:vscom="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.Immutable.11.0"
```

Puede usar `vscom:` el alias para enlazar comandos de Visual [Command](#) Studio a controles

`vscom:VSCommands.ExecuteCommand` XAML en la página estableciendo la propiedad del control en `.` A continuación, [CommandParameter](#) puede establecer la propiedad en el nombre del comando que se va a ejecutar, como se muestra en el ejemplo siguiente.

```
<Button Name="btnNewProj" Content="New Project"
        Command="{x:Static vscom:VSCommands.ExecuteCommand}"
        CommandParameter="File.NewProject" >
</Button>
```

NOTE

El `x:` alias, que hace referencia al esquema XAML, es necesario al principio de todos los comandos.

Puede establecer el valor `Command` de la propiedad en cualquier comando al que se pueda tener acceso desde la ventana **Comando**. Para obtener una lista de los comandos disponibles, vea [Alias de comandos](#) de Visual Studio .

Si el comando que se va a agregar requiere un `CommandParameter` parámetro adicional, puede agregarlo al valor de la propiedad. Separe los parámetros de los comandos mediante espacios, como se muestra en el ejemplo siguiente.

```
<Button Content="Web Search"
        Command="{x:Static vscom:VSCommands.ExecuteCommand}"
        CommandParameter="View.WebBrowser www.bing.com" />
```


Extensiones de llamada desde el pozo de comandos

Puede llamar a comandos de VSPackages registrados mediante la misma sintaxis que se usa para llamar a otros comandos de Visual Studio. Por ejemplo, si un VSPackage instalado agrega un comando de **página principal** al menú **Ver**, puede llamar a ese comando estableciendo `CommandParameter` en `View.HomePage`.

NOTE

Si llama a un comando asociado a un VSPackage, el paquete debe cargarse cuando se invoca el comando.

Agregar comandos desde ensamblados

Para llamar a un comando desde un ensamblado o para tener acceso al código de un VSPackage que no está asociado a un comando de menú, debe crear un alias para el ensamblado y, a continuación, llamar al alias.

Para llamar a un comando desde un ensamblado

1. En la solución, agregue una referencia al ensamblado.
2. En la parte superior del archivo *StartPage.xaml*, agregue una directiva de espacio de nombres para el ensamblado, como se muestra en el ejemplo siguiente.

```
xmlns:vsc="clr-namespace:WebUserControl;assembly=WebUserControl"
```

3. Invoque el `Command` comando estableciendo la propiedad de un objeto XAML, como se muestra en el ejemplo siguiente.

Xaml

```
<vs:Button Text="Hide me" Command="{x:Static vsc:HideControl}" .../>
```

NOTE

Debe copiar el ensamblado y, a continuación, pegarlo en `*..`. \Para asegurarse de que se carga antes de llamarla, para asegurarse de que se carga antes de llamarla.*

Añadir comandos con el objeto DTE

Puede tener acceso al objeto DTE desde una página de inicio, tanto en el marcado como en el código.

En el marcado, puede tener acceso a él [DTE](#) mediante el uso de la extensión de [marcado](#) de enlace sintaxis para llamar al objeto. Puede usar este enfoque para enlazar a propiedades simples como las que devuelven colecciones, pero no puede enlazar a métodos o servicios. En el ejemplo [TextBlock](#) siguiente se muestra [Name](#) un control [ListBox](#) que se enlaza [Caption](#) a la propiedad y [Windows](#) un control que enumera las propiedades de la colección devuelta por la propiedad.

```
<TextBlock Text="{Binding Path=DTE.Name}" FontSize="12" HorizontalAlignment="Center"/>
<ListBox ItemsSource="{Binding Path=DTE.Windows}">
  <ListBox.ItemTemplate>
    <DataTemplate>
      <TextBlock Text="{Binding Path=Caption}"/>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
```

Para obtener un ejemplo, vea [Tutorial: Guardar la configuración](#) de usuario en una página de inicio .

Vea también

- [Adición de control de usuario a la página de inicio](#)

Tutorial: Guardar la configuración de usuario en una página de inicio

06/04/2020 • 8 minutes to read • [Edit Online](#)

Puede conservar la configuración de usuario de la página de inicio. Siguiendo este tutorial, puede crear un control que guarde una configuración en el registro cuando el usuario hace clic en un botón y, a continuación, recupera esa configuración cada vez que se carga la página de inicio. Dado que la plantilla de proyecto de página de inicio incluye un control de usuario personalizable y el XAML de página de inicio predeterminado llama a ese control, no es que se debe modificar la propia página de inicio.

El almacén de configuración que se crea [IVsWritableSettingsStore](#) una instancia en este tutorial es una instancia de la interfaz, que lee y escribe en la siguiente ubicación del Registro cuando se llama: **HKCU, Software, Microsoft, VisualStudio, 14.0\<CollectionName>**

Cuando se ejecuta en la instancia experimental de Visual Studio, el almacén de configuración lee y escribe en **HKCU, Software, Microsoft, VisualStudio, 14,0Exp,\<CollectionName>**.

Para obtener más información sobre cómo conservar la configuración, consulte [Ampliación](#) de la configuración y las opciones de usuario .

Prerrequisitos

NOTE

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Puede descargar la plantilla de proyecto de página de inicio mediante el **Administrador de extensiones**.

Configuración del proyecto

1. Cree un proyecto de página de inicio como se describe en [Crear una página de inicio personalizada](#) . Asigne al proyecto el nombre **SaveMySettings**.
2. En el Explorador de **soluciones**, agregue las siguientes referencias de ensamblado al proyecto StartPageControl:
 - EnvDTE
 - EnvDTE80
 - Microsoft.VisualStudio.OLE.Interop
 - Microsoft.VisualStudio.Shell.Interop.11.0
3. Abra *MyControl.xaml*.
4. En el panel XAML, en [UserControl](#) la definición de elemento de nivel superior, agregue la siguiente declaración de evento después de las declaraciones de espacio de nombres.

```
Loaded="OnLoaded"
```

5. En el panel de diseño, haga clic en el área principal del control y, a continuación, presione **Eliminar**.

Este paso quita [Border](#) el elemento y todo lo que [Grid](#) contiene, y deja solo el elemento de nivel superior.

- Desde el cuadro [StackPanel](#) de **herramientas**, arrastre un control a la cuadrícula.
- Ahora arrastre [TextBlock](#) un [TextBox](#), un [StackPanel](#) y un botón al archivo .
- Agregue un atributo **x:Name** para el [TextBox](#) archivo , y un `Click` evento para el [Button](#), como se muestra en el ejemplo siguiente.

```
<StackPanel Width="300" HorizontalAlignment="Center" VerticalAlignment="Center">
    <TextBlock Width="140" FontSize="14">Enter your setting</TextBlock>
    <TextBox x:Name="txtblk" Margin="0, 5, 0, 10" Width="140" />
    <Button Click="Button_Click" Width="100">Save My Setting</Button>
</StackPanel>
```

Implementar el control de usuario

- En el panel XAML, `Click` haga clic [Button](#) con el botón secundario en el atributo del elemento y, a continuación, haga clic en **Navegar al controlador de eventos**.

Este paso abre *MyControl.xaml.cs* crea un `Button_Click` controlador de código auxiliar para el evento.

- Agregue las `using` siguientes directivas a la parte superior del archivo.

```
using EnvDTE80;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;
using System.ComponentModel;
```

- Agregue una `SettingsStore` propiedad privada, como se muestra en el ejemplo siguiente.

```
private IVsWritableSettingsStore _settingsStore = null;
private IVsWritableSettingsStore SettingsStore
{
    get
    {
        if (_settingsStore == null)
        {
            // Get a reference to the DTE from the DataContext.
            var typeDescriptor = DataContext as ICustomTypeDescriptor;
            var propertyCollection = typeDescriptor.GetProperties();
            var dte = propertyCollection.Find("DTE", false).GetValue(
                DataContext) as DTE2;

            // Get the settings manager from the DTE.
            var serviceProvider = new ServiceProvider(
                (Microsoft.VisualStudio.OLE.Interop.IServiceProvider)dte);
            var settingsManager = serviceProvider.GetService(
                typeof(SVsSettingsManager)) as IVsSettingsManager;

            // Write the user settings to _settingsStore.
            settingsManager.GetWritableSettingsStore(
                (uint)_VsSettingsScope.SettingsScope_UserSettings,
                out _settingsStore);
        }
        return _settingsStore;
    }
}
```

Esta propiedad primero obtiene una [DTE2](#) referencia a la interfaz, [DataContext](#) que contiene el modelo de

objetos Automation, desde [IVsSettingsManager](#) el control de usuario y, a continuación, usa el DTE para obtener una instancia de la interfaz. A continuación, utiliza esa instancia para devolver la configuración de usuario actual.

4. Rellene el `Button_Click` evento de la siguiente manera.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    int exists = 0;
    SettingsStore.CollectionExists("MySettings", out exists);
    if (exists != 1)
    {
        SettingsStore.CreateCollection("MySettings");
    }
    SettingsStore.SetString("MySettings", "MySetting", txtblk.Text);
}
```

Esto escribe el contenido del cuadro de texto en un campo "MySetting" en una colección "MySettings" en el registro. Si la colección no existe, se crea.

5. Agregue el siguiente `OnLoaded` controlador para el evento del control de usuario.

```
private void OnLoaded(Object sender, RoutedEventArgs e)
{
    string value;
    SettingsStore.GetStringOrDefault(
        "MySettings", "MySetting", "", out value);
    txtblk.Text = value;
}
```

Este código establece el texto del cuadro de texto en el valor actual de "MySetting".

6. Compile el control de usuario.
7. En el **Explorador de soluciones**, abra *source.extension.vsixmanifest*.
8. En el editor de manifiestos, establezca **Nombre del producto** en Guardar la página de inicio de la configuración.

Esta característica establece el nombre de la página de inicio tal como debe aparecer en la lista **Personalizar página de inicio** del cuadro de diálogo **Opciones**.

9. Compile *StartPage.xaml*.

Prueba del control

1. Presione F5.

Se abre la instancia experimental de Visual Studio.

2. En la instancia experimental, en el menú **Herramientas**, haga clic en **Opciones**.
3. En el nodo **Entorno**, haga clic en **Inicio**, a continuación, en la lista **Personalizar página de inicio**, seleccione **[Extensión instalada] Guardar mi página de inicio de configuración**.

Haga clic en **OK**.

4. Cierre la página de inicio si está abierta y, a continuación, en el menú **Ver**, haga clic en **Página de inicio**.
5. En la página de inicio, haga clic en la pestaña **MyControl**.

6. En el cuadro de texto, escriba **Caty**, a continuación, haga clic en **Guardar mi configuración**.

7. Cierre la página de inicio y, a continuación, vuelva a abrirla.

La palabra "Gato" debe mostrarse en el cuadro de texto.

8. Reemplace la palabra "Cat" por la palabra "Dog". No haga clic en el botón.

9. Cierre la página de inicio y, a continuación, vuelva a abrirla.

La palabra "Perro" debe mostrarse en el cuadro de texto, aunque no haya guardado la configuración porque Visual Studio mantiene las ventanas de herramientas en la memoria, incluso si están cerradas, hasta que Visual Studio se cierra.

10. Cierre la instancia experimental de Visual Studio.

11. Presione F5 para volver a abrir la instancia experimental.

12. La palabra "Gato" debe mostrarse en el cuadro de texto.

Pasos siguientes

Puede modificar este control de usuario para guardar y recuperar cualquier número de valores personalizados mediante el uso de valores diferentes de diferentes controladores de eventos para obtener y establecer la `SettingsStore` propiedad. Siempre que use un `propertyName` parámetro diferente `SetString` para cada llamada a , los valores no se sobrescriben entre sí en el registro.

Vea también

- [EnvDTE80.DTE2](#)
- [Agregar comandos de Visual Studio a una página de inicio](#)

Implementar páginas de inicio personalizadas

06/04/2020 • 8 minutes to read • [Edit Online](#)

Puede implementar páginas de inicio personalizadas mediante la implementación de VSIX o copiando los archivos en las ubicaciones correctas en el equipo de destino.

Implementación de VSIX mediante la plantilla de proyecto Página de inicio

Al crear una página de inicio mediante la plantilla de proyecto de página de inicio y, a continuación, compilar el proyecto, Visual Studio crea un archivo `.vsix` que puede distribuir. Empaquetar una página de inicio en un archivo `.vsix` le ofrece las siguientes opciones de implementación, en función de la audiencia prevista:

- Puede colocar el archivo `.vsix` en un recurso compartido de red o en un sitio Web público. Cuando alguien abre el archivo, la página de inicio se instala automáticamente.
- Puede cargar el archivo `.vsix` en el sitio Web de [Visual Studio Marketplace](#) para que los usuarios puedan instalarlo mediante **Extension Manager**.

La plantilla de proyecto Página de inicio crea una copia de la página de inicio de Visual Studio predeterminada para que pueda modificar la copia y conservar el original.

Puede obtener la plantilla de proyecto de página de inicio mediante el **Administrador de extensiones** o descargándola desde el sitio Web.

Implementación de VSIX sin usar la plantilla de proyecto de página de inicio

Una implementación correcta de VSIX requiere que se instale una extensión en carpetas reconocidas por el proceso de registro de VSIX y por **Extension Manager**. Dado que la plantilla de proyecto de página de inicio ya especifica las carpetas correctas, se recomienda usarla siempre que desee empaquetar una extensión para la implementación de VSIX. Sin embargo, si tiene un caso en el que no puede usar la plantilla, puede crear una implementación VSIX sin usarla.

Para crear una implementación VSIX sin usar la plantilla de proyecto Página de inicio, primero cree un archivo `.vsix` para la página de inicio de cualquiera de estas dos maneras:

- Agregando los archivos de la página de inicio personalizada a un proyecto VSIX vacío. Para obtener más información, consulte Plantilla de [proyecto VSIX](#).
- Creando manualmente un archivo `.vsix`. Para crear un archivo `.vsix` manualmente:
 1. Cree el archivo `extension.vsixmanifest` y el archivo `[Content_Types].xml` en una carpeta nueva. Para obtener más información, consulte [Anatomía de un paquete VSIX](#).
 2. En el Explorador de Windows, haga clic con el botón secundario en la carpeta que contiene los dos archivos XML, haga clic en **Enviar a**, a continuación, haga clic en **Carpeta comprimida** (comprimida). Cambie el nombre del archivo `.zip` resultante a `Filename.vsix`, donde `Filename` es el nombre del archivo redistribuible que instala el paquete.

Para que Visual Studio reconozca `Content Element` una página de inicio, `CustomExtension Element` el `Type` manifiesto de `"StartPage"` VSIX debe contener un que tenga el atributo establecido en `.` Aparece una extensión de

página de inicio que se ha instalado mediante la implementación de VSIX en la lista **Personalizar página de inicio** de la página Opciones de **inicio** como Nombre de *extensión* **[Extensión instalada]** .

Si el paquete de la página de inicio incluye ensamblados, debe agregar el registro de ruta de acceso de enlace para que estén disponibles cuando se inicie Visual Studio. Para ello, asegúrese de que el paquete incluye un archivo *.pkgdef* que tiene la siguiente información.

```
[$RootKey$\BindingPaths\{Insert a new GUID here}]
"$PackageFolder$"=""
```

Implementación de VSIX para todos los usuarios

De forma predeterminada, las extensiones implementadas en paquetes VSIX solo se instalan para el usuario actual. Puede realizar una instalación de la página de inicio para todos los usuarios del equipo de destino mediante la creación de una implementación de todos los usuarios.

Para crear una implementación de todos los usuarios

1. Abra el archivo *extension.vsixmanifest* en la vista de código.
2. En `Identifier` el elemento del manifiesto vsix, agregue `AllUsers` un `true` elemento que tenga un valor de `.`

```
<AllUsers>true</AllUsers>
```

Esto hace que el instalador de vsix solicite permisos de administrador y, a continuación, instale los archivos en `.`

3. Abra el archivo *.pkgdef*.
4. Modifique el archivo *.pkgdef* para establecer la página de inicio predeterminada en HKLM agregando lo siguiente, donde *MyStartPage.xaml* es el nombre del archivo *.xaml* que contiene la página de inicio.

```
[$RootKey$-StartPage-Predeterminado]
```

```
"Uri"-"$PackageFolder$\MyStartPage.xaml"
```

Esto indica a Visual Studio que busque en la nueva ubicación de la página de inicio.

Implementación de copia de archivos

No es necesario crear un archivo *.vsix* para implementar una página de inicio personalizada. En su lugar, puede copiar el marcado y los archivos *auxiliares* directamente en la carpeta de la página de inicio del usuario.

Customize Start Page La lista * Personalizar página de inicio de la página Opciones de **inicio** muestra todos los archivos *.xaml* de esa carpeta, junto con la ruta de acceso, por ejemplo, *%USERPROFILE%\Mis documentos, Visual Studio, versión, StartPages, \Nombre de archivo y .xaml* . Si la página de inicio incluye referencias a ensamblados privados, debe* copiarlos y pegarlos en la carpeta * .

Para distribuir una página de inicio que ha creado sin empaquetarla en un archivo *.vsix*, le recomendamos que utilice una estrategia básica de copia de archivos, por ejemplo, un script por lotes o cualquier otra tecnología de implementación que le permita colocar los archivos en los directorios necesarios.

Para instalar manualmente una página de inicio personalizada

1. Copie el archivo *.xaml* que contiene el marcado de la página de inicio, junto con los archivos* auxiliares que no sean ensamblados, y péguelos en la carpeta **-StartPages* del usuario.
2. Si la página de inicio requiere ensamblados, cópielos y péguelos en *.. •Carpeta de instalación de Visual Studio, Common7, IDE, PrivateAssemblies\.* \

3. En la lista **Personalizar página** de inicio de la página Opciones de inicio, seleccione la nueva página de inicio. Para obtener más información, consulte [Personalizar la página de inicio](#).

Vea también

- [Personalizar la página de inicio](#)
- [Agregar control de usuario a la página de inicio](#)

Cómo: Usar el registro de actividad

06/04/2020 • 2 minutes to read • [Edit Online](#)

VSPackages puede escribir mensajes en el registro de actividad. Esta característica es especialmente útil para depurar VSPackages en entornos comerciales.

TIP

El registro de actividad siempre está activado. Visual Studio mantiene un búfer móvil de las últimas 100 entradas, así como las primeras 10 entradas, que tienen información de configuración general.

Para escribir una entrada en el registro de actividad

1. Inserte este código [Initialize](#) en el método o en cualquier otro método excepto el constructor VSPackage:

```
IVsActivityLog log = GetService(typeof(SVsActivityLog)) as IVsActivityLog;
if (log == null) return;

int hr = log.LogEntry((UInt32)___ACTIVITYLOG_ENTRYTYPE.ALE_INFORMATION,
    this.ToString(),
    string.Format(CultureInfo.CurrentCulture,
        "Called for: {0}", this.ToString()));
```

Este código obtiene [SVsActivityLog](#) el servicio y [IVsActivityLog](#) lo convierte en una interfaz.

[LogEntry](#) escribe una entrada informativa en el registro de actividad utilizando el contexto cultural actual.

2. Cuando se carga el VSPackage (normalmente cuando se invoca un comando o se abre una ventana), el texto se escribe en el registro de actividad.

Para examinar el registro de actividad

1. Ejecute Visual Studio con el modificador de línea de comandos [/Log](#) para escribir ActivityLog.xml en el disco durante la sesión.
2. Después de cerrar Visual Studio, busque el registro de actividad en la subcarpeta para los datos de Visual Studio:

```
%AppData%> .\<
```

3. Abra el registro de actividad con cualquier editor de texto. Aquí está una entrada típica:

```
Called for: Company.MyApp.MyAppPackage ...
```

Programación sólida

Dado que el registro de actividad es un servicio, el registro de actividad no está disponible en el constructor de VSPackage.

Debe obtener el registro de actividad justo antes de escribir en él. No almacene en caché ni guarde el registro de actividad para usarlo en el futuro.

Vea también

- [/Log \(devenv.exe\)](#)
- [IVsActivityLog](#)
- [__ACTIVITYLOG_ENTRYTYPE](#)
- [Solución de problemas de VSPackages](#)
- [VSPackages](#)

Visual Studio C++ sistema extensibilidad y el conjunto de herramientas de integración de Project

02/12/2019 • 41 minutes to read • [Edit Online](#)

El sistema de proyectos de Visual C++ se usa para archivos vcxproj. Se basa en el [sistema común de proyecto \(CPS\) de Visual Studio](#) y proporciona adicionales, puntos de extensibilidad específica de C++ para facilitar la integración de nuevos conjuntos de herramientas, las arquitecturas de compilación y las plataformas de destino.

Estructura de los destinos de MSBuild de C++

Todos los archivos vcxproj importación estos archivos:

```
<Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
<Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
<Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
```

Estos archivos definen poco por sí mismos. En su lugar, importan otros archivos en función de estos valores de propiedad:

- `$(ApplicationType)`

Ejemplos: Windows Store, Android, Linux

- `$(ApplicationTypeRevision)`

Esto debe ser una cadena de versión válida del formulario de.Revision]].

Ejemplos: 1.0, 10.0.0.0

- `$(Platform)`

La arquitectura de la compilación, denominada "Plataforma" por motivos históricos.

Ejemplos: Win32, x86, x64, ARM

- `$(PlatformToolset)`

Ejemplos: v140, v141, v141_xp, llvm

Estos valores de propiedad especifican los nombres de carpeta en el `$(VCTargetsPath)` carpeta raíz:

```
$(VCTargetsPath) \
  Tipo de aplicación\
    $(ApplicationType) \
      $(ApplicationTypeRevision) \
        Plataformas\
          $(Platform) \
            PlatformToolsets\
              $(PlatformToolset)
  Plataformas\
    $(Platform) \
      PlatformToolsets\
```

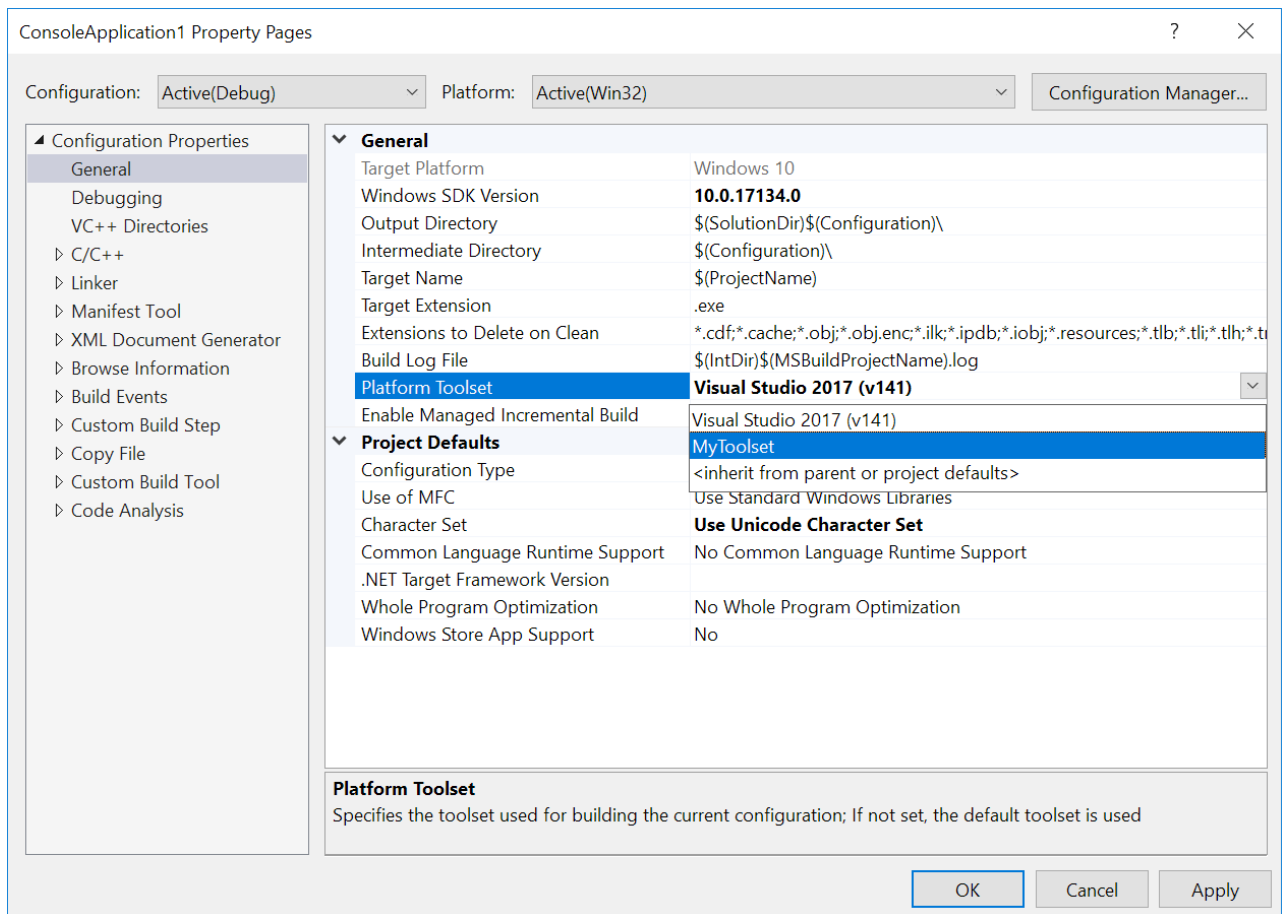
\$(PlatformToolset)

El `$(VCTargetsPath) \plataformas` carpeta se usa cuando `$(ApplicationType)` está vacío para los proyectos de escritorio de Windows.

Agregar un nuevo conjunto de herramientas de plataforma

Para agregar un nuevo conjunto de herramientas, por ejemplo, "MyToolset" para la plataforma existente de Win32, cree un *MyToolset* carpeta bajo `$(VCTargetsPath) \plataformas\Win32\PlatformToolsets\` y crear *Toolset.props* y *Toolset.targets* archivos en ella.

Cada nombre de la carpeta en *PlatformToolsets* aparece en el **las propiedades del proyecto** diálogo como una disponibilidad **conjunto de herramientas de plataforma** para la plataforma especificada, como se muestra aquí:

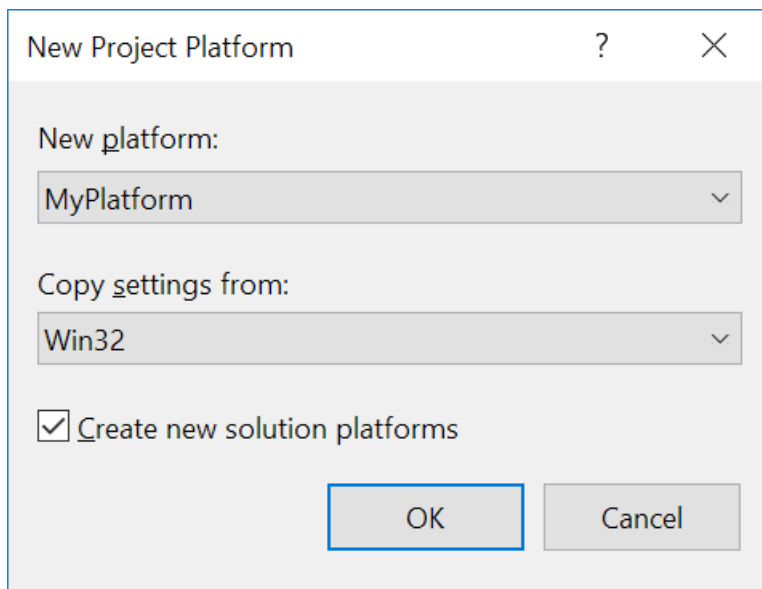


Crear similar *MyToolset* carpetas y *Toolset.props* y *Toolset.targets* admite este conjunto de herramientas de los archivos de cada carpeta de plataforma existente.

Agregar una nueva plataforma

Para agregar una nueva plataforma, por ejemplo, "MyPlatform", cree un *MyPlatform* carpeta bajo `$(VCTargetsPath) \plataformas\` y crear *Platform.Default.props*, *Platform.props*, y *Platform.targets* archivos en ella. Cree también un `$(VCTargetsPath) \plataformas\MyPlatform\PlatformToolsets\` carpeta y cree al menos un conjunto de herramientas en ella.

Todos los nombres de carpeta la *plataformas* carpeta para cada `$(ApplicationType)` y `$(ApplicationTypeRevision)` aparecen en el IDE como disponible **plataforma** opciones para un proyecto.



Agregar un nuevo tipo de aplicación

Para agregar un nuevo tipo de aplicación, cree un *MyApplicationType* carpeta bajo `$(VCTargetsPath) \tipo de aplicación\` y cree un *Defaults.props* archivos en ella. Al menos una revisión es necesaria para un tipo de aplicación, por lo que también cree un `$(VCTargetsPath) \tipo de aplicación\MyApplicationType\1.0` carpeta y crear un *Defaults.props* archivos en ella. También debe crear un `$(VCTargetsPath) \ApplicationType\MyApplicationType\1.0\plataformas` carpeta y cree al menos una plataforma en ella.

`$(ApplicationType)` y `$(ApplicationTypeRevision)` propiedades no son visibles en la interfaz de usuario. Se definen en las plantillas de proyecto y no se puede cambiar después de crear el proyecto.

El árbol de importación .vcxproj

Un árbol simplificado de las importaciones de archivos de propiedades y destinos de Microsoft C++ tiene el siguiente aspecto:

```
$(VCTargetsPath) \Microsoft.Cpp.Default.props
  $(MSBuildExtensionsPath) \ $(MSBuildToolsVersion) \Microsoft.Common.props
  $(VCTargetsPath) \ImportBefore\predeterminado\*.props
  $(VCTargetsPath) \Tipo de aplicación\ $(ApplicationType) \Default.props
  $(VCTargetsPath) \Tipo de aplicación\ $(ApplicationType) \ $(ApplicationTypeRevision) \Default.props
  $(VCTargetsPath) \Tipo de aplicación\ $(ApplicationType) \ $(ApplicationTypeRevision) \plataformas \
$(Platform) \ Platform.Default.props
  $(VCTargetsPath) \ImportAfter\predeterminado\*.props
```

Los proyectos de escritorio de Windows no definen `$(ApplicationType)`, por lo que se importan solo

```
$(VCTargetsPath) \Microsoft.Cpp.Default.props
  $(MSBuildExtensionsPath) \ $(MSBuildToolsVersion) \Microsoft.Common.props
  $(VCTargetsPath) \ImportBefore\predeterminado\*.props
  $(VCTargetsPath) \Plataformas\ $(Platform) \Platform.default.props
  $(VCTargetsPath) \ImportAfter\predeterminado\*.props
```

Vamos a usar el `$(PlatformFolder)` propiedad para alojar el `$(Platform)` las ubicaciones de carpeta de plataforma. Esta propiedad es

```
$(VCTargetsPath) \Plataformas\ $(Platform)
```

para las aplicaciones de escritorio de Windows, y

```
$(VCTargetsPath)\Tipo de aplicación\$(ApplicationType)\$(ApplicationTypeRevision)\plataformas\  
$(Platform)
```

para todo lo demás.

Los archivos props se importan en este orden:

```
$(VCTargetsPath)\Microsoft.Cpp.props  
  $(PlatformFolder)\Platform.props  
  $(VCTargetsPath)\Microsoft.Cpp.Platform.props  
    $(PlatformFolder)\ImportBefore\*.props  
    $(PlatformFolder)\PlatformToolsets\$(PlatformToolset)\Toolset.props  
    $(PlatformFolder)\ImportAfter\*.props
```

Los archivos de destinos se importan en este orden:

```
$(VCTargetsPath)\Microsoft.Cpp.targets  
  $(VCTargetsPath)\Microsoft.Cpp.Current.targets  
  $(PlatformFolder)\Platform.targets  
  $(VCTargetsPath)\Microsoft.Cpp.Platform.targets  
    $(PlatformFolder)\ImportBefore\*.destinos  
    $(PlatformFolder)\PlatformToolsets\$(PlatformToolset)\Toolset.target  
    $(PlatformFolder)\ImportAfter\*.targets
```

Si tiene que definir algunas propiedades predeterminadas para el conjunto de herramientas, puede agregar archivos a las carpetas ImportBefore e ImportAfter correspondientes.

Archivos Toolset.props autor y Toolset.targets

Toolset.props y *Toolset.targets* archivos tienen control total sobre lo que sucede durante una compilación cuando se usa este conjunto de herramientas. También pueden controlar los depuradores disponibles, algunas de la interfaz de usuario IDE, como el contenido de la **páginas de propiedades** cuadro de diálogo y algunos otros aspectos del comportamiento del proyecto.

Aunque un conjunto de herramientas puede invalidar el proceso de compilación completa, normalmente tan solo quiere su conjunto de herramientas para modificar o agregar que algunos pasos de compilación o usar herramientas de compilación diferentes, como parte de un proceso de compilación existente. Para lograr este objetivo, hay una serie de archivos comunes de propiedades y destinos, que puede importar el conjunto de herramientas. Dependiendo de qué desea hacer su conjunto de herramientas, estos archivos pueden resultar útiles para utilizarla como importaciones o como ejemplos:

- `$(VCTargetsPath)\Microsoft.CppCommon.targets`

Este archivo define las partes principales del proceso de compilación nativa y también importa:

- `$(VCTargetsPath)\Microsoft.CppBuild.targets`
- `$(VCTargetsPath)\Microsoft.BuildSteps.targets`
- `$(MSBuildToolsPath)\Microsoft.Common.Targets`

- `$(VCTargetsPath)\Microsoft.Cpp.Common.props`

Establece los valores predeterminados para los conjuntos de herramientas que usan los compiladores de Microsoft y Windows de destino.

- `$(VCTargetsPath)\Microsoft.Cpp.WindowsSDK.props`

Este archivo determina la ubicación del SDK de Windows y define algunas propiedades importantes para aplicaciones destinadas a Windows.

Integrar destinos específicos del conjunto de herramientas con la predeterminada del proceso de compilación de C++

El proceso de compilación de C++ de predeterminado se define en *Microsoft.CppCommon.targets*. Allí los destinos no llame a las herramientas de compilación concreta; especifican los principales pasos, el orden y sus dependencias de compilación.

La compilación de C++ tiene tres pasos principales que están representados por los siguientes destinos:

- `BuildGenerateSources`
- `BuildCompile`
- `BuildLink`

Dado que cada paso de compilación se puede ejecutar de forma independiente, los destinos que se ejecutan en un paso no pueden confiar en los grupos de elementos y propiedades definidos en los destinos que se ejecutan como parte de un paso diferente. Esta división permite cierta compilación optimizaciones de rendimiento. Aunque no se usa de forma predeterminada, está todavía le anima a respetar esta separación.

Los destinos que se ejecutan dentro de cada paso se controlan mediante estas propiedades:

- `$(BuildGenerateSourcesTargets)`
- `$(BuildCompileTargets)`
- `$(BeforeBuildLinkTargets)`

Cada paso tiene también propiedades precedentes y posteriores.

```
<Target
  Name="_BuildGenerateSourcesAction"

  DependsOnTargets="$(CommonBuildOnlyTargets);$(BeforeBuildGenerateSourcesTargets);$(BuildGenerateSourcesTargets);$(AfterBuildGenerateSourcesTargets)" />

<Target
  Name="_BuildCompileAction"

  DependsOnTargets="$(CommonBuildOnlyTargets);$(BeforeBuildCompileTargets);$(BuildCompileTargets);$(AfterBuildCompileTargets)" />

<Target
  Name="_BuildLinkAction"

  DependsOnTargets="$(CommonBuildOnlyTargets);$(BeforeBuildLinkTargets);$(BuildLinkTargets);$(AfterBuildLinkTargets)" />
```

Consulte la *Microsoft.CppBuild.targets* archivo para ver ejemplos de los destinos que se incluyen en cada paso:


```
<BuildCompileTargets Condition="'$(ConfigurationType)'!='Utility'">
  $(BuildCompileTargets);
  _ClCompile;
  _ResGen;
  _ResourceCompile;
  $(BuildLibTargets);
</BuildCompileTargets>
```

Si observa los destinos, como `_ClCompile`, verá que no hacen nada directamente por sí mismos, pero en su lugar, dependen de otros destinos, incluidos `ClCompile`:

```
<Target Name="_ClCompile"

  DependsOnTargets="$(BeforeClCompileTargets);$(ComputeCompileInputsTargets);MakeDirsForCl;ClCompile;$(AfterClCompileTargets)" >
</Target>
```

`ClCompile` y otro compilar destinos específicos de la herramienta se definen como destinos vacíos de *Microsoft.CppBuild.targets*:

```
<Target Name="ClCompile"/>
```

Dado que el `ClCompile` destino está vacío, a menos que se haya reemplazado por un conjunto de herramientas, no se realiza ninguna acción de compilación real. Los destinos del conjunto de herramientas pueden reemplazar el `ClCompile` de destino, es decir, puede contener otro `ClCompile` definición después de importar *Microsoft.CppBuild.targets*:

```
<Target Name="ClCompile"
  Condition="'@(ClCompile)' != ''"
  DependsOnTargets="SelectClCompile">
  <!-- call some MSBuild tasks -->
</Target>
```

A pesar de su nombre, que se creó antes de Visual Studio implementa la compatibilidad multiplataforma, el `ClCompile` destino no tiene que llamar a CL.exe. También puede llamar gcc, Clang o con otros compiladores mediante el uso de las tareas de MSBuild adecuadas.

El `ClCompile` destino no debe contener todas las dependencias, excepto el `SelectClCompile` destino, que es necesario para el comando de compilación de único archivo trabajar en el IDE.

Tareas de MSBuild para usar en los destinos del conjunto de herramientas

Para invocar una herramienta de compilación real, el destino debe llamar a una tarea de MSBuild. Hay un [basic tarea Exec](#) que le permite especificar una línea de comandos para ejecutar. Sin embargo, las herramientas de compilación normalmente tienen muchas opciones, las entradas y las salidas para realizar el seguimiento de compilaciones incrementales, por lo que tiene más sentido tener tareas especiales para ellos. Por ejemplo, la `CL` tarea traduce las propiedades de MSBuild en conmutadores de CL.exe, escribe en un archivo de respuesta y llama a CL.exe. También realiza un seguimiento todos los archivos de entrada y salidos para las compilaciones incrementales más adelante. Para obtener más información, consulte [las compilaciones incrementales y comprobaciones actualizadas](#).

El Microsoft.Cpp.Common.Tasks.dll implementa estas tareas:

- `BSCMake`
- `CL`
- `ClangCompile` (modificadores de clang y gcc)
- `LIB`
- `LINK`
- `MIDL`
- `Mt`
- `RC`
- `XDCMake`
- `CustomBuild` (como Exec pero con la entrada y salida de seguimiento)
- `SetEnv`
- `GetOutOfDateItems`

Si tiene una herramienta que realiza la misma acción que una herramienta existente y que tiene los modificadores de línea de comandos similar (como clang-cl y CL), puede usar la misma tarea para ambos.

Si necesita crear una nueva tarea para una herramienta de compilación, puede elegir entre las siguientes opciones:

1. Si usa esta tarea con poca frecuencia, o si no importan unos segundos para la compilación, puede usar las tareas de MSBuild 'inline':

- Tarea de XAML (una regla de compilación personalizada)

Para obtener un ejemplo de una declaración de la tarea de Xaml, vea `$(VCTargetsPath) \ BuildCustomizations\masm.xml` para su uso, consulte `$(VCTargetsPath) \ BuildCustomizations\masm.targets`.

- [Tarea de código](#)

2. Si desea obtener un mejor rendimiento de la tarea o simplemente necesita funcionalidad más compleja, use el normal de MSBuild [escribir tareas](#) proceso.

Si no todas las entradas y salidas de la herramienta se muestran en la línea de comandos de la herramienta, como en el `CL`, `MIDL`, y `RC` casos y, si desea entrada automática y seguimiento de archivos de salida y creación de archivos .tlog, derive su tarea desde el `Microsoft.Build.CPPTasks.TrackedVCToolTask` clase. En este momento, aunque no hay documentación para la base de [ToolTask](#) de clases, no existen ejemplos o documentación para obtener los detalles de la `TrackedVCToolTask` clase. Si esto es de especial interés, agregar su voz a una solicitud en developercommunity.visualstudio.com.

Las compilaciones incrementales y comprobaciones actualizadas

La compilación incremental de MSBuild predeterminado tiene como destino uso `Inputs` y `Outputs` atributos. Si se especifica, MSBuild llama al destino sólo si cualquiera de las entradas tiene una marca de tiempo más reciente de todas las salidas. Porque a menudo archivos de código fuente incluirán o importan otros archivos y generación herramientas generan resultados diferentes dependiendo de las opciones de herramienta, es difícil especificar todas las posibles entradas y salidas en destinos de MSBuild.

Para administrar este problema, la compilación de C++ usa una técnica diferente para admitir las compilaciones incrementales. La mayoría de los destinos no especificar entradas y salidas y como resultado, siempre se ejecutan

durante la compilación. Las tareas que se llama a destinos de escriben información acerca de todas las entradas y salidas en *tlog* archivos que tienen una extensión .tlog. Los archivos .tlog son usados por compilaciones posteriores para comprobar qué ha cambiado y debe volver a generar y, lo que está actualizado. Los archivos .tlog también son el único origen para la comprobación de actualización de la compilación de forma predeterminada en el IDE.

Para determinar todas las entradas y salidas, las tareas de la herramienta nativa usan `tracker.exe` y `FileTracker` clase proporcionada por MSBuild.

Microsoft.Build.CPPTasks.Common.dll define el `TrackedVCToolTask` clase base abstracta pública. La mayoría de las tareas de la herramienta nativa se deriva de esta clase.

A partir de Visual Studio 2017 update 15,8, puede usar el `GetOutOfDateItems` implementado en `Microsoft.Cpp.Common.Tasks.dll` para generar archivos .tlog para destinos personalizados con conocidos entradas y salidas de tareas. Como alternativa, puede crear mediante la `WriteLinesToFile` tarea. Consulte la `_WriteMasmTlogs` tener como destino en `$(VCTargetsPath) \ BuildCustomizations\masm.targets` como ejemplo.

archivos .tlog

Hay tres tipos de archivos .tlog: *leer*, *escribir*, y *de línea de comandos*. Archivos .tlog de lectura y escritura se usan las compilaciones incrementales y la comprobación actualizada en el IDE. Archivos .tlog de línea de comandos solo se usan en las compilaciones incrementales.

MSBuild proporciona estas clases auxiliares para leer y escribir archivos .tlog:

- `CanonicalTrackedInputFiles`
- `CanonicalTrackedOutputFiles`

El `FlatTrackingData` clase puede utilizarse para tener acceso tanto de lectura y escribir archivos .tlog e identificar las entradas que son más recientes que da como resultado, o si falta una salida. Se utiliza en la comprobación de actualización.

Archivos .tlog de línea de comandos contienen información acerca de las líneas de comandos que se usará en la compilación. Solo se usan para las compilaciones incrementales, las comprobaciones no actualizadas, por lo que el formato interno viene determinada por la tarea de MSBuild que los genera.

Formato de lectura .tlog

Lectura archivos .tlog (*Read*. registro de transacciones) contienen información sobre los archivos de origen y sus dependencias.

Un símbolo de intercalación (^) al principio de una línea indica uno o varios orígenes. Orígenes que comparten las mismas dependencias están separados por una barra vertical (|).

Archivos de dependencia se muestran después de los orígenes, cada uno en su propia línea. Todos los nombres de archivo son rutas de acceso completas.

Por ejemplo, suponga que los orígenes del proyecto se encuentran en

`F:\probar\ConsoleApplication1\ConsoleApplication1`. Si el archivo de origen, `Class1.cpp`, estos se incluye,

```
#include "stdafx.h" //precompiled header
#include "Class1.h"
```

el `CL.read.1.tlog` archivo contiene el archivo de origen, seguido de sus dos dependencias:

```
^F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\CLASS1.CPP
F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\DEBUG\CONSOLEAPPLICATION1.PCH
F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\CLASS1.H
```

No es necesario escribir los nombres de archivo en mayúsculas, pero resulta práctico para algunas herramientas.

Escribir .tlog formato

Escribir .tlog (*.write.*. archivos de registro de transacciones) conectan los orígenes y las salidas.

Un símbolo de intercalación (^) al principio de una línea indica uno o varios orígenes. Varios orígenes están separados por una barra vertical (|).

Los archivos de salida que se crean a partir de los orígenes deben aparecer después de los orígenes, cada uno en su propia línea. Todos los nombres de archivo deben ser rutas de acceso completas.

Por ejemplo, para un proyecto de aplicación de consola simple que tiene un archivo de origen adicionales *Class1.cpp*, *link.write.1.tlog* archivo puede contener:

```
^F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\DEBUG\CLASS1.OBJ|F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\DEBUG\CONSOLEAPPLICATION1.OBJ|F:\TEST\CONSOLEAPPLICATION1\CONSOLEAPPLICATION1\DEBUG\STDAFX.OBJ
F:\TEST\CONSOLEAPPLICATION1\DEBUG\CONSOLEAPPLICATION1.ILK
F:\TEST\CONSOLEAPPLICATION1\DEBUG\CONSOLEAPPLICATION1.EXE
F:\TEST\CONSOLEAPPLICATION1\DEBUG\CONSOLEAPPLICATION1.PDB
```

Compilación en tiempo de diseño

En el IDE, .vcxproj proyectos utilizan un conjunto de destinos de MSBuild para obtener información adicional desde el proyecto y volver a generar los archivos de salida. Solo algunos de estos destinos se usan en las compilaciones de tiempo de diseño, pero muchas de ellas se utilizan en las compilaciones regulares y las compilaciones de tiempo de diseño.

Para obtener información general sobre las compilaciones de tiempo de diseño, vea la documentación de CPS para [se basa en tiempo de diseño](#). Esta documentación solo está parcialmente aplicable a proyectos de Visual C++.

El `CompileDesignTime` y `Compile` documentación nunca ejecutado para proyectos .vcxproj se basa los destinos que se mencionan en el tiempo de diseño. Proyectos de Visual C++ .vcxproj usan distintos destinos de tiempo de diseño para obtener información de IntelliSense.

Objetivos de tiempo de diseño de la información de IntelliSense

Los objetivos de tiempo de diseño usados en los proyectos .vcxproj se definen en `$(VCTargetsPath) \Microsoft.Cpp.DesignTime.targets`.

El `GetClCommandLines` destino recopila las opciones del compilador de IntelliSense:

```
<Target
  Name="GetClCommandLines"
  Returns="@ (ClCommandLines)"
  DependsOnTargets="$(DesignTimeBuildInitTargets);$(ComputeCompileInputsTargets)">
```

- `DesignTimeBuildInitTargets` : tiempo de diseño solo la inicialización de compilación de destinos, necesarios para el tiempo de diseño. Entre otras cosas, estos destinos deshabilitar algunas de las funcionalidades de compilación normal para mejorar el rendimiento.
- `ComputeCompileInputsTargets` : un conjunto de destinos que modifica las opciones del compilador y los elementos. Estos destinos se ejecutan en las compilaciones regulares y de tiempo de diseño.

Las llamadas de destino la `CLCommandLine` tarea para crear la línea de comandos para IntelliSense. De nuevo, a pesar de su nombre, puede controlar no sólo las opciones de CL, sino también las opciones Clang y gcc. El tipo de los modificadores de compilador se controla mediante el `CLangMode` propiedad.

Actualmente, la línea de comandos generada por el `CLCommandLine` tarea siempre usa conmutadores de CL (incluso en modo de Clang) porque son más fáciles para el motor de IntelliSense para analizar.

Si va a agregar un destino que se ejecuta antes de la compilación, ya sea regular o tiempo de diseño, asegúrese de que no se interrumpe el tiempo de diseño se basa o afectar al rendimiento. La manera más sencilla para probar su destino es abrir un símbolo del sistema para desarrolladores y ejecute este comando:

```
msbuild /p:SolutionDir=*solution-directory-with-trailing-
backslash*;Configuration=Debug;Platform=Win32;BuildingInsideVisualStudio=true;DesignTimeBuild=true
/t:_PerfIntellisenseInfo /v:d /fl /fileloggerparameters:PerformanceSummary \*.vcxproj
```

Este comando genera un registro de compilación detalladas, *msbuild.log*, que tiene un resumen de los destinos y tareas de rendimiento al final.

Asegúrese de usar `Condition="'$(DesignTimeBuild)' != 'true'"` en todas las operaciones que solo tienen sentido para las compilaciones periódicas y no para las compilaciones de tiempo de diseño.

Objetivos de tiempo de diseño que generan fuentes

Esta característica está deshabilitada de forma predeterminada para proyectos nativos de escritorio y no se admite actualmente en los proyectos almacenados en caché.

Si `GeneratorTarget` los metadatos se definen para un elemento de proyecto, el destino se ejecuta automáticamente tanto cuando se carga el proyecto y cuando se cambia el archivo de origen.

Por ejemplo, para generar automáticamente .cpp o .h archivos desde los archivos .xaml, el `$(VSInstallDir) \ MSBuild\Microsoft\ WindowsXaml\v15.0\* \ Microsoft.Windows.UI.Xaml.CPP.Targets` archivos definen estas entidades:

Por ejemplo, para generar automáticamente .cpp o .h archivos desde los archivos .xaml, el `$(VSInstallDir) \ MSBuild\Microsoft\ WindowsXaml\v16.0\* \ Microsoft.Windows.UI.Xaml.CPP.Targets` archivos definen estas entidades:

```
<ItemDefinitionGroup>
  <Page>
    <GeneratorTarget>DesignTimeMarkupCompilation</GeneratorTarget>
  </Page>
  <ApplicationDefinition>
    <GeneratorTarget>DesignTimeMarkupCompilation</GeneratorTarget>
  </ApplicationDefinition>
</ItemDefinitionGroup>
<Target Name="DesignTimeMarkupCompilation">
  <!-- BuildingProject is used in Managed builds (always true in Native) -->
  <!-- DesignTimeBuild is used in Native builds (always false in Managed) -->
  <CallTarget Condition="'$(BuildingProject)' != 'true' Or $(DesignTimeBuild) == 'true'"
    Targets="DesignTimeMarkupCompilationCT" />
</Target>
```

Para usar `Task.HostObject` para obtener el contenido que no haya guardado de archivos de origen, se deben registrar los destinos y tareas como [MsbuildHostObjects](#) para los proyectos en un pkgdef determinados:

```
\[$RootKey$\Projects\{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}\MSBuildHostObjects\]
\[$RootKey$\Projects\{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}\MSBuildHostObjects\DesignTimeMarkupCompilationCT;CompileXaml\]
@="{83046B3F-8984-444B-A5D2-8029DEE2DB70}"
```

Extensibilidad de proyectos de Visual C++ en el IDE de Visual Studio

El sistema de proyectos de Visual C++ se basa en el [sistema del proyecto de VS](#) y utiliza sus puntos de extensibilidad. Sin embargo, la implementación de la jerarquía de proyecto es específica de Visual C++ y no se basa en CPS, extensibilidad de la jerarquía se limita a los elementos de proyecto.

Páginas de propiedades del proyecto

Para obtener información general de diseño, vea [Framework Multi-Targeting para proyectos de VC++](#).

En otras palabras, las páginas de propiedades se vea en el **las propiedades del proyecto** cuadro de diálogo para un proyecto de C++ se definen mediante *regla* archivos. Un archivo de reglas especifica un conjunto de propiedades para mostrar en una página de propiedades y cómo y dónde debe guardarse en el proyecto de archivos. Los archivos de reglas son archivos .xml que usar el formato Xaml. Se describen los tipos utilizados para serializarlos en [Microsoft.Build.Framework.XamlTypes](#). Para obtener más información sobre el uso de archivos de reglas en los proyectos, vea [los archivos de reglas de XML de la página de propiedades](#).

Los archivos de regla deben agregarse a la `PropertyPageSchema` grupo de elementos:

```
<ItemGroup>
  <PropertyPageSchema Include="$(VCTargetsPath)$(LangID)\general.xml;" />
  <PropertyPageSchema Include="$(VCTargetsPath)$(LangID)\general_file.xml">
    <Context>File</Context>
  </PropertyPageSchema>
</ItemGroup>
```

`Context` metadatos limita la visibilidad de la regla, que también se controla mediante el tipo de regla y puede tener uno de estos valores:

`Project` | `File` | `PropertySheet`

CPS admite otros valores para el tipo de contexto, pero no se usan en proyectos de Visual C++.

Si la regla debe estar visible en más de un contexto, use punto y coma (;) para separar los valores de contexto, como se muestra aquí:

```
<PropertyPageSchema Include="$(MyFolder)\MyRule.xml">
  <Context>Project;PropertySheet</Context>
</PropertyPageSchema>
```

Formato de regla y los tipos principales

El formato de la regla es sencillo, por lo que en esta sección solo se describe los atributos que afectan al aspecto de la regla en la interfaz de usuario.

```
<Rule
  Name="ConfigurationGeneral"
  DisplayName="General"
  PageTemplate="generic"
  Description="General"
  xmlns="http://schemas.microsoft.com/build/2009/properties">
```

El `PageTemplate` atributo define cómo se muestra la regla en el **páginas de propiedades** cuadro de diálogo. El atributo puede tener uno de estos valores:

ATRIBUTO	DESCRIPCIÓN
<code>generic</code>	Todas las propiedades se muestran en una sola página bajo los encabezados de categoría. La regla puede ser visible para <code>Project</code> y <code>PropertySheet</code> contextos, pero no <code>File</code>. Ejemplo: <code>\$(VCTargetsPath)\1033\general.xml</code>
<code>tool</code>	Las categorías se muestran como subpáginas. La regla puede ser visible en todos los contextos: <code>Project</code>, <code>PropertySheet</code> y <code>File</code>. La regla sólo está visible en las propiedades del proyecto si el proyecto tiene elementos con el <code>ItemType</code> definido en <code>Rule.DataSource</code>, a menos que se incluye el nombre de regla en el <code>ProjectTools</code> grupo de elementos. Ejemplo: <code>\$(VCTargetsPath)\1033\clang.xml</code>
<code>debugger</code>	La página se muestra como parte de la página de depuración. Actualmente se omiten las categorías. El nombre de la regla debe coincidir con el objeto de MEF de iniciador de depuración <code>ExportDebugger</code> atributo. Ejemplo: <code>\$(VCTargetsPath)\1033\debugger_local_windows.xml</code>
<code>custom</code>	Plantilla personalizada. Debe coincidir con el nombre de la plantilla de la <code>ExportPropertyPageUIFactoryProvider</code> atributo de la <code>PropertyPageUIFactoryProvider</code> objeto MEF. Consulte Microsoft.VisualStudio.ProjectSystem.Designers.Properties.IPropertyPageUIFactoryProvider. Ejemplo: <code>\$(VCTargetsPath)\1033\userMacros.xml</code>

Si la regla usa una de las plantillas de la cuadrícula de propiedades, pueden usar estos puntos de extensibilidad para sus propiedades:

- [Editores de valores de propiedad](#)
- [Proveedor de valores de enumeración dinámica](#)

Extender una regla

Si desea usar una regla existente, pero necesita agregar o quitar (que es, ocultar) solo unas pocas propiedades, puede crear un [regla extensión](#).

Invaldar una regla

Quizás desee su conjunto de herramientas para usar la mayoría de las reglas predeterminadas de proyecto, pero para reemplazar solo uno o algunos de ellos. Por ejemplo, supongamos que desea cambiar la regla de C/C++ para mostrar los modificadores de compilador diferentes. Puede proporcionar una nueva regla con el mismo nombre y nombre para mostrar que la regla existente e incluirlo en el `PropertyPageSchema` grupo de elementos después de la importación de destinos de cpp de forma predeterminada. Se usa solo una regla con el nombre especificado en el proyecto y la última de ellas se incluye en el `PropertyPageSchema` wins de grupo de elementos.

elementos de proyecto

El *ProjectItemsSchema.xml* archivo define la `ContentType` y `ItemType` valores para los elementos que se tratan como elementos de proyecto y define `FileExtension` elementos para determinar qué grupo de elementos que se agrega un nuevo archivo a.

El archivo de ProjectItemsSchema predeterminado se encuentra en `$(VCTargetsPath)\1033\ProjectItemsSchema.xml`. Incluir en ella, debe crear un archivo de esquema con un nuevo nombre, como *MyProjectItemsSchema.xml*.

```
<ProjectSchemaDefinitions xmlns="http://schemas.microsoft.com/build/2009/properties">

  <ItemType Name="MyItemType" DisplayName="C/C++ compiler"/>

  <ContentType
    Name="MyItems"
    DisplayName="My items"
    ItemType=" MyItemType ">
  </ContentType>

  <FileExtension Name=".abc" ContentType=" MyItems"/>

</ProjectSchemaDefinitions>
```

A continuación, en el archivo de destinos, agregue:

```
<ItemGroup>
  <PropertyPageSchema Include="MyProjectItemsSchema.xml"/>
</ItemGroup>
```

Ejemplo: `$(VCTargetsPath)\BuildCustomizations\masm.xml`

Depuradores

El servicio de depuración en Visual Studio admite la extensibilidad para el motor de depuración. Para obtener más información, vea estos ejemplos:

- [MIEngine, proyecto de código abierto que admite la depuración lldb](#)
- [Ejemplo de motor de depuración de Visual Studio](#)

Para especificar los motores de depuración y otras propiedades para la sesión de depuración, debe implementar un [iniciador de depuración](#) componente MEF y agregue un `debugger` regla. Para obtener un ejemplo, vea el `$(VCTargetsPath)\1033\depurador_local_windows.xml` archivo.

Implementar

los proyectos de .vcxproj usan la extensibilidad del sistema de proyectos de Visual Studio para [implementar proveedores](#).

Comprobación de actualización de compilación

De forma predeterminada, la comprobación de actualización de compilación requiere lectura .tlog y escribir los archivos de .tlog que se van a crearse en el `$(TLogLocation)` carpeta durante la compilación para todas las compilación entradas y salidas.

Para usar una comprobación de actualización personalizada:

1. Deshabilitar la comprobación de actualización predeterminado agregando la `NoVCDefaultBuildUpToDateCheckProvider` capacidad en el *Toolset.targets* archivo:


```
<ItemGroup>
  <ProjectCapability Include="NoVCDefaultBuildUpToDateCheckProvider" />
</ItemGroup>
```

2. Implementar su propia [IBuildUpToDateCheckProvider](#).

Actualización del proyecto

Actualizador de proyecto .vcxproj predeterminado

Los cambios de actualizador del proyecto predeterminado .vcxproj el `PlatformToolset`, `ApplicationTypeRevision`, conjunto de herramientas de MSBuild versión y .net framework. Las dos últimas siempre se cambian los valores predeterminados de Visual Studio versión, pero `PlatformToolset` y `ApplicationTypeRevision` puede controlarse mediante propiedades especiales de MSBuild.

El actualizador utiliza estos criterios para decidir si se puede actualizar un proyecto o no:

1. Para los proyectos que definen `ApplicationType` y `ApplicationTypeRevision`, hay una carpeta con un mayor número de revisión de la actual.
2. La propiedad `_UpgradePlatformToolsetFor_<safe_toolset_name>` está definido para el conjunto de herramientas actual y su valor no es igual que el conjunto de herramientas actual.

En estos nombres de propiedad, `<safe_toolset_name>` representa el nombre del conjunto de herramientas con todos los caracteres no alfanuméricos reemplazados por un carácter de subrayado (_).

Cuando se puede actualizar un proyecto, participa en *redestinar solución*. Para obtener más información, consulte [IVsTrackProjectRetargeting2](#).

Si desea agregar adornos a nombres de proyecto de el **Explorador de soluciones** cuando los proyectos usan un conjunto de herramientas específica, defina un `_PlatformToolsetShortNameFor_<safe_toolset_name>` propiedad.

Para obtener ejemplos de `_UpgradePlatformToolsetFor_<safe_toolset_name>` y `_PlatformToolsetShortNameFor_<safe_toolset_name>` definiciones de propiedades, vea el *Microsoft.Cpp.Default.props* archivo. Para obtener ejemplos de uso, consulte el `$(VCTargetPath) \ Microsoft.Cpp.Platform.targets` archivo.

Actualizador de proyecto personalizadas

Para usar un objeto del actualizador de proyecto personalizadas, implemente un componente MEF, como se muestra aquí:

```
/// </summary>
[Export("MyProjectUpgrader", typeof(IProjectRetargetHandler))]
[Export(typeof(IProjectRetargetHandler))]
[ExportMetadata("Name", "MyProjectUpgrader")]
[OrderPrecedence(20)]
[PartMetadata(ProjectCapabilities.Requires, ProjectCapabilities.VisualStudio)]

internal class MyProjectUpgrader: IProjectRetargetHandler
{
    // ...
}
```

El código puede importar y llamar al objeto de actualizador .vcxproj predeterminada:

```
// ...
[Import("VCDDefaultProjectUpgrader")]
// ...
    IProjectRetargetHandler Lazy<IProjectRetargetHandler>
    VCDDefaultProjectUpgrader { get; set; }
// ...
```

`IProjectRetargetHandler` se define en *Microsoft.VisualStudio.ProjectSystem.VS.dll* y es similar a `IVsRetargetProjectAsync`.

Definir la `VCProjectUpgraderObjectName` propiedad para indicar al sistema de proyecto para utilizar su objeto personalizado del actualizador:

```
<PropertyGroup>
  <VCProjectUpgraderObjectName>MyProjectUpgrader</VCProjectUpgraderObjectName>
</PropertyGroup>
```

Deshabilitar la actualización del proyecto

Para deshabilitar las actualizaciones del proyecto, use un `NoUpgrade` valor:

```
<PropertyGroup>
  <VCProjectUpgraderObjectName>NoUpgrade</VCProjectUpgraderObjectName>
</PropertyGroup>
```

Caché de proyectos y extensibilidad

Para mejorar el rendimiento cuando se trabaja con grandes soluciones de C++ en Visual Studio 2017, el [proyecto caché](#) introducida. Se implementa como una base de datos de SQLite rellena con datos del proyecto y, a continuación, se usa para cargar los proyectos sin cargar los proyectos de MSBuild o CPS en la memoria.

Dado que no hay ningún objeto de CPS presente para los proyectos de .vcxproj cargados desde la memoria caché, los componentes MEF de la extensión importar `UnconfiguredProject` o `ConfiguredProject` no se puede crear. Para admitir la extensibilidad, la caché del proyecto no se usa cuando Visual Studio detecta si un proyecto utiliza (o es probable que use) en las extensiones MEF.

Estos tipos de proyecto siempre se carguen por completo y tienen objetos de CPS en memoria, por lo que se crean todas las extensiones MEF para ellos:

- Proyectos de inicio
- Los proyectos que tienen un actualizador de proyecto personalizadas, es decir, definen un `VCProjectUpgraderObjectName` propiedad
- Los proyectos que no están destinadas a Windows Desktop, es decir, definen un `ApplicationType` propiedad
- Compartir proyectos de elementos (.vcxitems) y todos los proyectos que hacen referencia a ellos mediante la importación de proyectos .vcxitems.

Si se detecta ninguna de estas condiciones, se crea una memoria caché del proyecto. La memoria caché incluye todos los datos desde el proyecto de MSBuild necesarios para responder a `get` consultas en `VCProjectEngine` interfaces. Esto significa que todas las modificaciones en las propiedades de MSBuild y nivel de archivo de destinos realiza por una extensión debería funcionar en los proyectos cargados desde la memoria caché.

La extensión de envío

Para obtener información sobre cómo crear archivos VSIX, vea [envío extensiones de Visual Studio](#). Para obtener

información acerca de cómo agregar archivos a ubicaciones de instalación especial, por ejemplo, para agregar archivos en `$(VCTargetsPath)`, consulte [instalación fuera de la carpeta de extensiones](#).

Recursos adicionales

El sistema de compilación de Microsoft ([MSBuild](#)) proporciona el motor de compilación y el formato extensible basado en XML para archivos de proyecto. Debe estar familiarizado con basic [conceptos de MSBuild](#) y con la forma [MSBuild de Visual C++](#) funciona con el fin de ampliar Visual C++ de sistema de proyectos.

Managed Extensibility Framework ([MEF](#)) proporciona la extensión de API que se usan por el sistema de proyectos de Visual C++ y CPS. Para obtener información general de cómo se usa MEF CPS, consulte [CPS y MEF](#) en el [VSProjectSystem introducción de MEF](#).

Puede personalizar el sistema de compilación existente para agregar pasos de compilación o nuevos tipos de archivo. Para obtener más información, consulte [información general sobre MSBuild \(Visual C++\)](#) y [trabajar con las propiedades del proyecto](#).

Instrucciones para la experiencia de usuario de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

Siga las directrices de experiencia de usuario de Visual Studio si diseña nuevas características para Visual Studio. Estas directrices contienen información sobre los modelos de usuario comunes y los patrones de interacción para que los diseñadores de nuevas interfaces de usuario (UI) puedan crear experiencias de usuario fluidas y coherentes en el entorno de desarrollo.

Si desarrolla software para productos de Microsoft, aprenda a usar estos recursos:

- Las directrices de [interacción](#) de la experiencia del usuario de Windows son los conceptos básicos para el comportamiento y la apariencia del escritorio de Windows.
- Estas directrices de Visual Studio, que son directrices específicas de la plataforma que reemplazan las directrices generales de Windows siempre que sea apropiado para nuestro entorno.
- Los [controles y patrones para aplicaciones para UWP](#) son una referencia para los patrones emergentes que se pueden usar en algunas instancias de Visual Studio.

Estas directrices proporcionan información sobre qué hacer al diseñar para Visual Studio y, en algunos casos, también se incluye información sobre cómo hacerlo. Para obtener información de implementación más detallada y obtener información sobre interfaces de programación de aplicaciones (API) específicas que permiten escribir código para soluciones que se integran en el entorno de desarrollo integrado (IDE) de Visual Studio, use el Kit de desarrollo de software ([SDK](#)) de Visual Studio.

Fundamentos de la experiencia de usuario de Visual Studio

06/04/2020 • 14 minutes to read • [Edit Online](#)

Procedimientos recomendados

1. Sea coherente en el entorno de Visual Studio.

- Siga los patrones de [interacción](#) existentes dentro del shell.
- Diseño de características para ser coherente con el lenguaje visual de la cáscara y [los requisitos de artesanía](#).
- Utilice comandos y controles compartidos cuando existan.
- Comprender la jerarquía de Visual Studio y cómo establece el contexto y controla la interfaz de usuario.

2. Utilice el servicio de entorno para fuentes y colores.

- La interfaz de usuario debe respetar la configuración de fuente de [entorno](#) actual a menos que se exponga para su personalización en la página Fuentes y colores del cuadro de diálogo Opciones.
- Los elementos de interfaz de usuario deben usar el [servicio VSColor](#), mediante tokens de entorno compartido o tokens específicos de la característica.

3. Haga que todas las imágenes sean coherentes con el nuevo estilo VS.

- Siga los principios de diseño de Visual Studio para iconos, glifos y otros gráficos.
- No coloque texto en elementos gráficos.

4. Diseño desde una perspectiva centrada en el usuario.

- Cree el flujo de tareas antes de las entidades individuales que contiene.
- Familiarícese con sus usuarios y haga que ese conocimiento sea explícito en sus especificaciones.
- Al revisar la interfaz de usuario, evalúe la experiencia completa, así como los detalles.
- Diseña tu interfaz de usuario para que siga siendo funcional y atractiva independientemente de la configuración regional o el idioma.

Resolución de pantalla

Resolución mínima

- La resolución mínima para Visual Studio 2015 es **1280x720**. Esto significa que es *posible* usar Visual Studio en esta resolución, aunque podría no ser una experiencia de usuario óptima. No hay garantía de que todos los aspectos sean utilizables en resoluciones inferiores a 1280x720.
- La resolución de destino para Visual Studio es **1366x768**. Esta es la resolución más baja en la que prometemos una *buena* experiencia de usuario.
- La altura inicial del cuadro de diálogo debe ser inferior a **700 píxeles**, por lo que se ajusta a la resolución mínima del marco IDE a 96 ppp.

Pantallas de alta densidad

La interfaz de usuario de Visual Studio debe funcionar bien en todos los factores de escalado de PPP que Windows admite de forma inmediata: 150%, 200% y 250%.

Antipatrones

Visual Studio contiene muchos ejemplos de interfaz de usuario que siguen nuestras directrices y procedimientos recomendados. En un esfuerzo por ser coherentes, los desarrolladores a menudo toman prestados de patrones de diseño de interfaz de usuario de productos similares a lo que están creando. Aunque este es un buen enfoque que nos ayuda a impulsar la coherencia en la interacción del usuario y el diseño visual, en ocasiones en ocasiones enviamos características con algunos detalles que no cumplen con nuestras directrices debido a restricciones de programación o priorización de defectos. En estos casos, no queremos que los equipos copien uno de estos "antipatrones" porque proliferan la interfaz de usuario incorrecta o incoherente dentro del entorno de Visual Studio.

Campos/configuraciones obligatorios mostrados en estado de error de forma predeterminada

Objetivos del equipo destacado

- Advierta a los usuarios que han agregado un elemento que se debe configurar.
- Atraiga la atención del usuario sobre las áreas que necesitan entrada.

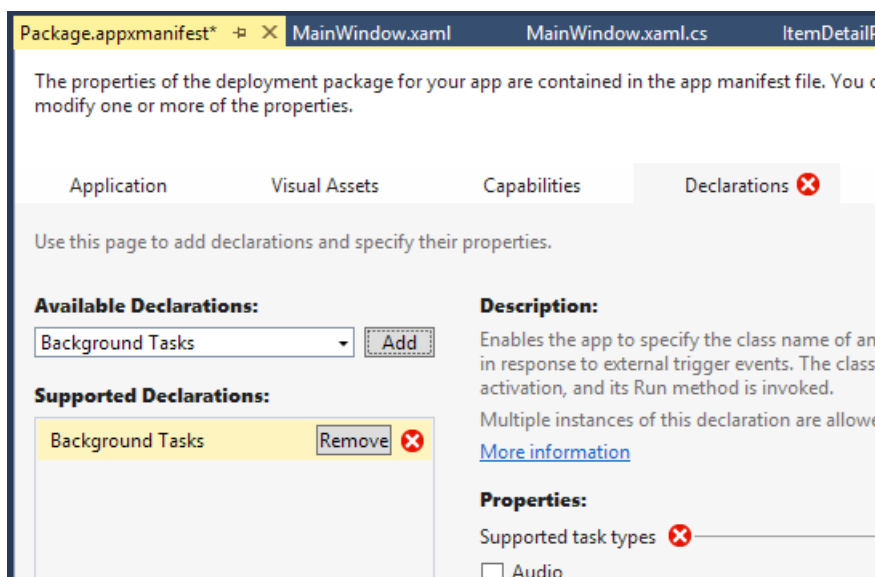
Solución anti-patrón

Tan pronto como el usuario haya iniciado una acción y antes de que haya completado la tarea, coloque inmediatamente iconos de parada crítica junto a las áreas que necesitan configuración.

Ejemplo: Declaraciones de Manifest Designer

Agregar una declaración a la lista la coloca inmediatamente en un estado de error, que persiste hasta que el usuario establece las propiedades necesarias.

En este caso, hay una preocupación adicional porque el icono utilizado para la alerta contiene un icono "X", por lo que el icono de eliminación común no se puede utilizar junto a él. Como resultado, la interfaz de usuario usa un botón Quitar, un control más torpe.



Colocar la interfaz de usuario en un estado de error de forma predeterminada es un antipatrón de Visual Studio.

Alternativas

Una mejor solución a este problema es:

- Permitir al usuario agregar una declaración sin advertencia y, a continuación, mover inmediatamente para establecer propiedades en el elemento.
- Agregue el icono de advertencia (triángulo dorado) cuando el foco se mueva desde el elemento, como agregar otra declaración a la lista o intentar cambiar las pestañas dentro del diseñador.
- Si el usuario intenta cambiar las pestañas antes de establecer propiedades en cualquier declaración, haga clic en un cuadro de diálogo que explique que la aplicación no se compilará (o cualquiera que sean las

implicaciones) hasta que se resuelvan las advertencias. Si el usuario descarta el cuadro de diálogo y cambia las pestañas de todos modos, se agrega un icono (crítico o de advertencia, según corresponda) a la pestaña Declaraciones.

Múltiples clics para descartar la interfaz de usuario

Objetivos del equipo destacado

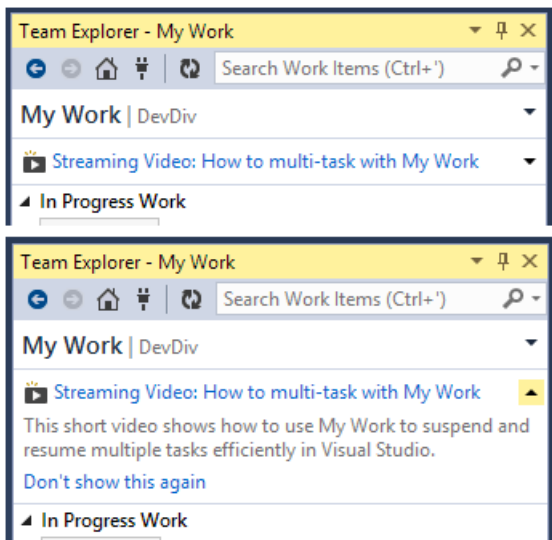
No permita que el usuario descarte la interfaz de usuario sin ver primero el texto de explicación.

Anti-patrón

El equipo que inserta los vínculos de vídeo en varios lugares dentro de la interfaz de usuario de VS decidió en contra del patrón común del botón "x" cerrar y la explicación de información sobre herramientas según lo especificado por LA experiencia de usuario, y en su lugar implementó un menú desplegable y "No volver a mostrar".

Ejemplo: Enlaces de vídeo en Team Explorer

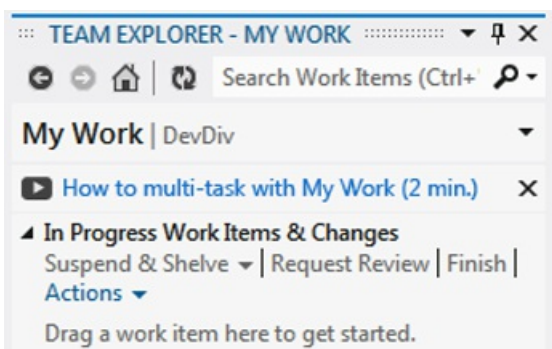
Obligar al usuario a leer texto explicativo antes de descartar la interfaz de usuario es un antipatrón dentro de Visual Studio. Correctamente diseñados, los enlaces de vídeo deben mostrar una×información sobre herramientas con información adicional al mantener el mouse, y al hacer clic en el botón " " debe descartar el mensaje sin necesidad de más interacción.



Patrón de enlace de vídeo incorrecto

En lugar de un simple botón de cierre (un clic), el usuario se ve obligado a usar dos clics para simplemente descartar la interfaz de usuario en cada lugar en el que aparezcan los vínculos de vídeo.

El diseño correcto para esta situación es seguir el patrón común a Internet Explorer, Office y Visual Studio: al mantener el mouse, el usuario puede ver la descripción de información sobre herramientas y un clic oculta la interfaz de usuario.



Corregir el patrón de enlace de vídeo

Uso de barras de comandos para la configuración

La figura A representa este antipatrón: colocar una configuración debajo de un botón de comando que se aplica a

algo más que el comando. En este boceto, hay comandos además de Iniciar depuración, como Ver en el navegador, Iniciar sin depurar y Paso a paso, que respetarán la configuración seleccionada.

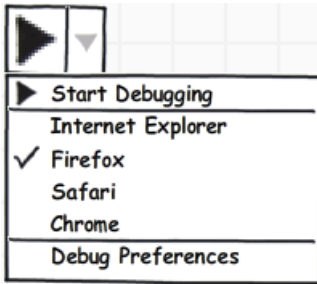


Figura A: Barra de comandos antipatrón

Ligeramente mejor, pero todavía indeseable, es poner la configuración de este tipo en las barras de herramientas, como se muestra en la figura B. Mientras que los botones divididos toman menos espacio y, por lo tanto, son una mejora sobre los menús desplegables, ambos diseños todavía están utilizando una barra de herramientas para promover algo que no es realmente un comando.

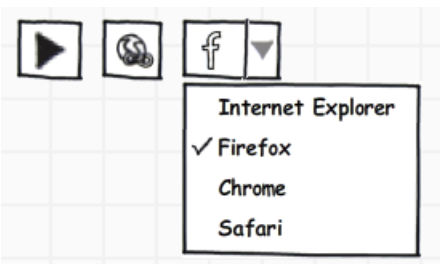


Figura B: Mejor, pero sigue siendo una barra de comandos anti-patrón

En el enfoque correcto que se muestra en la figura C, la configuración está vinculada a una serie de comandos. No hay ninguna configuración global que se establece y sólo estamos cambiando entre cuatro comandos. Esta es la única situación en la que los comandos de la barra de herramientas son aceptables.

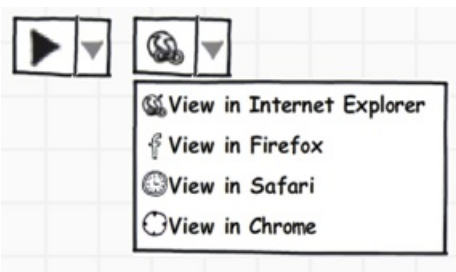


Figura C: Uso correcto del patrón de barra de comandos de Visual Studio

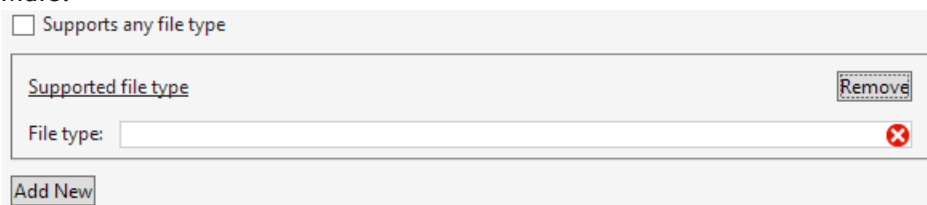
Controlar antipatrones

Algunos antipatrones son simplemente un uso incorrecto o la presentación de un control o grupo de controles.

Subrayado utilizado como etiqueta de grupo, no como hipervínculo

El texto subrayado solo debe utilizarse para hipervínculos.

Malo:



El texto subrayado que no es un hipervínculo es un antipatrón de Visual Studio.

bien:

☐ Supports any file type

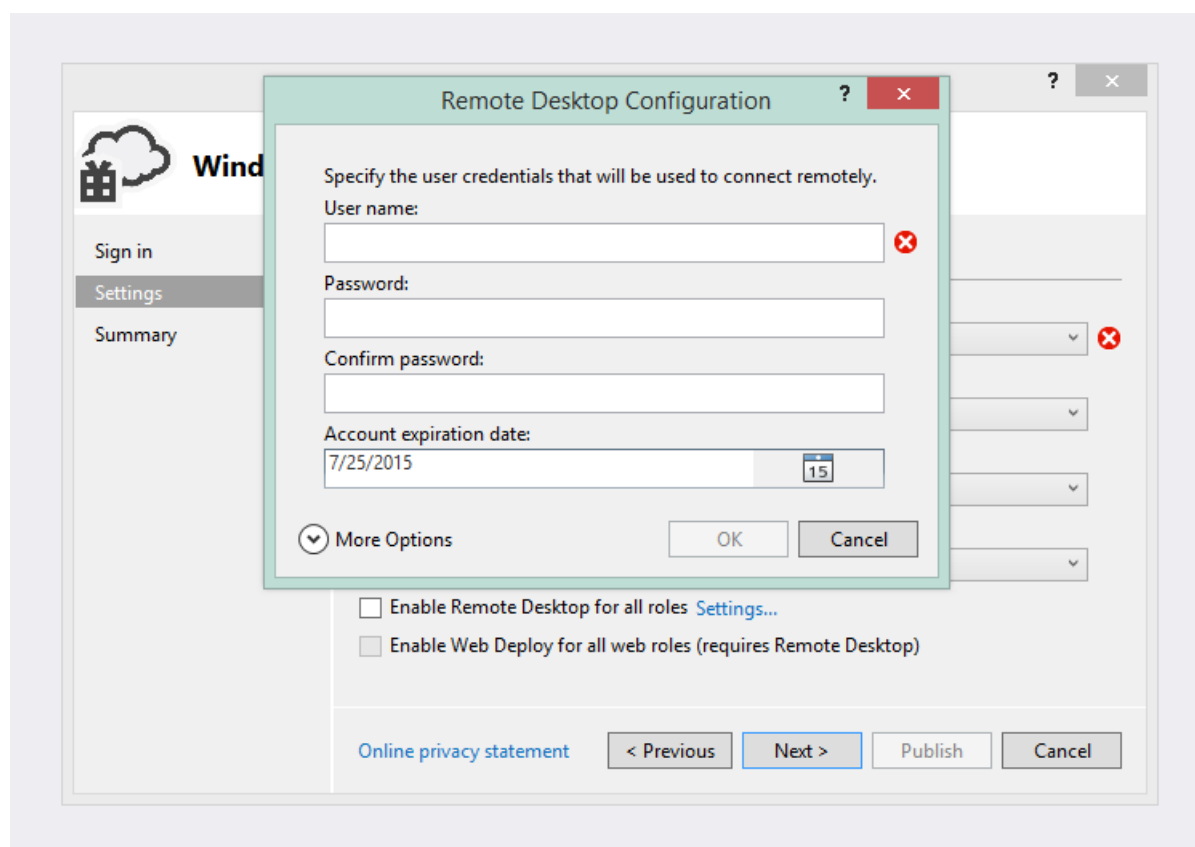
Supported file type:  [Remove](#)

[Add New](#)

Con el estilo correcto, el texto que no es de hipervínculo aparece sin adornos en la fuente del entorno.

Al hacer clic en una casilla de verificación, aparece un cuadro de diálogo emergente

Al hacer clic en la casilla de verificación "Habilitar Escritorio remoto para todos los roles" en el asistente "Publicar aplicación de Windows Azure" aparece inmediatamente un cuadro de diálogo emergente, un antipatrón de Visual Studio. Además, el campo de casilla de verificación no se rellena con una casilla de verificación después de haber sido seleccionado, otro antipatrón de interacción.

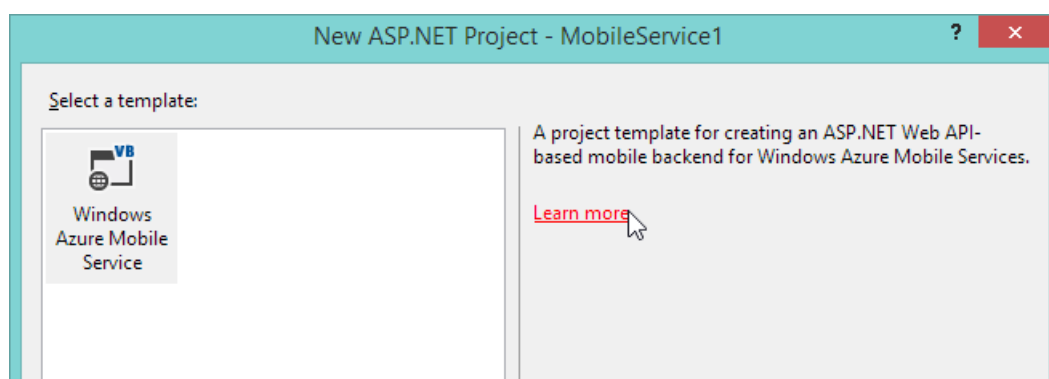


Al abrir un cuadro de diálogo después de hacer clic en una casilla de verificación, es un antipatrón de Visual Studio.

Antipatrones de hipervínculo

El ejemplo siguiente contiene dos antipatrones:

1. El primer plano que se vuelve rojo al mantener el mouse significa que no se está utilizando el color compartido correcto del servicio de fuentes.
2. "Aprender más" no es el texto adecuado para un enlace a un tema conceptual. El objetivo del usuario no es aprender más, es entender las ramificaciones de su elección.



Ignorar el servicio de color y usar "Más información" para hipervínculos son antipatrones de Visual Studio.

Mejor solución: Haga la pregunta que el usuario estaría haciendo haciendo clic en el enlace. Por ejemplo:

- ¿Cómo funcionan los servicios de Windows Azure?
- ¿Cuándo necesito un proyecto de Windows Azure Mobile Services?

Uso de "Haga clic aquí" para enlaces

Los hipervínculos deben ser autodescriptivos. Es un anti-patrón para utilizar "Haga clic aquí" o cualquier variación similar.

Malo: "Haga clic aquí para obtener instrucciones sobre cómo crear un nuevo proyecto."

Bien: "¿Cómo puedo crear un nuevo proyecto?"

Fuentes y formato de Visual Studio

06/04/2020 • 32 minutes to read • [Edit Online](#)

La fuente del entorno

Todas las fuentes de Visual Studio deben exponerse al usuario para su personalización. Esto se hace principalmente a través de la página **Fuentes y colores** en el cuadro de diálogo **Herramientas > Opciones**. Las tres categorías principales de ajustes de fuente son:

- **Fuente** de entorno: la fuente principal para el IDE (entorno de desarrollo integrado), que se utiliza para todos los elementos de la interfaz, incluidos los cuadros de diálogo, los menús, las ventanas de herramientas y las ventanas de documentos. De forma predeterminada, la fuente del entorno está vinculada a una fuente del sistema que aparece como interfaz de usuario de Segoe de 9 pt en las versiones actuales de Windows. El uso de una fuente para todos los elementos de la interfaz ayuda a garantizar un aspecto de fuente coherente en todo el IDE.
- **Editor** de texto: los elementos que asolan el código y otros editores basados en texto se pueden personalizar en la página Editor de texto de **Herramientas > Opciones**.
- **Colecciones específicas**: las ventanas de diseño que ofrecen la personalización por parte del usuario de sus elementos de interfaz pueden exponer fuentes específicas de su superficie de diseño en su propia página de configuración en **Herramientas > Opciones**.

Personalización y cambio de tamaño de la fuente del editor

Los usuarios a menudo ampliarán o amplían el tamaño y/o el color del texto en el editor según sus preferencias, independientemente de la interfaz de usuario general. Dado que la fuente de entorno se utiliza en elementos que pueden aparecer dentro o como parte de un editor o diseñador, es importante tener en cuenta el comportamiento esperado cuando se cambia una de estas clasificaciones de fuente.

Al crear elementos de interfaz de usuario que aparecen en el editor pero no forman parte del *contenido*, es importante usar la fuente de entorno y no la fuente de texto para que los elementos cambian de tamaño de una manera predecible.

1. Para el texto de código en el editor, cambie el tamaño con la configuración de fuente de texto de código y responda al nivel de zoom del texto del editor.
2. Todos los demás elementos de la interfaz deben estar vinculados a la configuración de fuente del entorno y responder a cualquier cambio global en el entorno. Esto incluye (pero no se limita a):
 - Texto en menús contextuales
 - Texto en un adorno del editor, como el texto del menú de la bombilla, el panel del editor de búsqueda rápida y navegar hasta el panel
 - Etiquetar texto en cuadros de diálogo, como **Buscar en archivos** o **Refactorizar**

Acceso a la fuente del entorno

En código nativo o WinForms, se puede tener `IUIHostLocale::GetDialogFont` acceso a la `SID_SUIHostLocale` fuente de entorno llamando al método después de consultar la interfaz desde el servicio.

Para Windows Presentation Foundation (WPF), derive la clase `DialogWindow` de ventana de `Window` cuadro de diálogo de la clase del shell en lugar de la clase de WPFWPF.

En XAML, el código tiene este aspecto:

```
<ui:DialogWindow
    x:Class"MyNameSpace.MyWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:s="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ui="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Shell.11.0"
    ShowInTaskbar="False"
    WindowStartupLocation="CenterOwner"
    Title="My Dialog">
</ui:DialogWindow>
```

Código detrás:

```
internal partial class WebConfigModificationWindow : DialogWindow
{
}
}
```

(Reemplazar `Microsoft.VisualStudio.Shell.11.0` con la versión actual de la DLL MPF.)

Para mostrar el cuadro `ShowModal()` de diálogo, `ShowDialog()` llame a " " en la clase sobre . `ShowModal()` establece el estado modal correcto en el shell, garantiza que el cuadro de diálogo se centra en la ventana primaria, etc.

El código es el siguiente:

```
MyWindow window = new MyWindow();
window.ShowDialog();
```

`ShowModal` devuelve un bool? (booleano que acepta `DialogResult` valores NULL) con el , que se puede utilizar si es necesario. El valor devuelto es true si el cuadro de diálogo se cerró con OK.

Si necesita mostrar alguna interfaz de usuario de WPF `HwndSource` que no es un cuadro de diálogo y se hospeda en su propio , como `FontFamily` `FontSize` una ventana emergente o una ventana secundaria WPF de una ventana primaria Win32/WinForms, deberá establecer el elemento raíz y en el elemento WPFWPF. (El shell establece las propiedades en la ventana principal, `HWND` pero no se heredarán más allá de a). El shell proporciona recursos a los que se pueden enlazar las propiedades, de la siguiente manera:

```
<Setter Property="FontFamily" Value="{DynamicResource VsFont.EnvironmentFontFamily}" />
<Setter Property="FontSize" Value="{DynamicResource VsFont.EnvironmentFontSize}" />
```

Referencia de formato (escalado/bolding)

Algunos cuadros de diálogo requieren que el texto en particular esté en negrita o un tamaño distinto de la fuente del entorno. Anteriormente, las fuentes más grandes que `environment font +2` la fuente del entorno se codificaban como " " o similares. El uso de los fragmentos de código proporcionados admitirá monitores de alto PPP y garantizará que el texto de visualización siempre aparezca con el tamaño y el peso correctos (como Light o Semilight).

NOTE

Antes de aplicar formato, asegúrese de seguir las instrucciones que se encuentran en [Estilo de texto](#) .**

Para escalar la fuente del entorno, establezca el estilo de TextBlock o Label como se indica. Cada uno de estos fragmentos de código, utilizados correctamente, generará la fuente correcta, incluidas las variaciones de tamaño y peso adecuadas.

Donde `vsui` " " es una `Microsoft.VisualStudio.Shell` referencia al espacio de nombres:

```
xmlns:vsui="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.14.0"
```

375% Fuente de entorno + Luz

Aparece como: 34 pt Segoe UI Light

Usar para: (rara) interfaz de usuario de marca única, como en la página de inicio

Usar para: (rara) interfaz de usuario de marca única

Código procesal: Dónde `textBlock` es un `TextBlock` definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,
    VsResourceKeys.TextBlockEnvironment375PercentFontSizeStyleKey);
label.SetResourceReference(Label.StyleProperty,
    VsResourceKeys.LabelEnvironment375PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de `TextBlock` o `Label` como se muestra.

```
<TextBlock Style="{DynamicResource {x:Static
vsui:VsResourceKeys.TextBlockEnvironment375PercentFontSizeStyleKey}}">TextBlock: 375 Percent
Scaling</TextBlock>
<Label Style="{DynamicResource {x:Static
vsui:VsResourceKeys.LabelEnvironment375PercentFontSizeStyleKey}}">Label: 375 Percent Scaling</Label>
```

310% Fuente de entorno + Luz

Aparece como: 28 pt Segoe UI Light **Use for:** large signature dialog titles, main heading in reports

Código procesal: Dónde `textBlock` es un `TextBlock` definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,
    VsResourceKeys.TextBlockEnvironment310PercentFontSizeStyleKey);
label.SetResourceReference(Label.StyleProperty,
    VsResourceKeys.LabelEnvironment310PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de `TextBlock` o `Label` como se muestra.

```
<TextBlock Style="{DynamicResource {x:Static
vsui:VsResourceKeys.TextBlockEnvironment310PercentFontSizeStyleKey}}">TextBlock: 310 Percent
Scaling</TextBlock>
<Label Style="{DynamicResource {x:Static
vsui:VsResourceKeys.LabelEnvironment310PercentFontSizeStyleKey}}">Label: 310 Percent Scaling</Label>
```

200% Fuente de entorno + Semilight

Aparece como: 18 pt Segoe UI Semilight **Use for:** subheadings, titles in small and medium dialogs

Código procesal: Dónde `textBlock` es un `TextBlock` definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,
    VsResourceKeys.TextBlockEnvironment200PercentFontSizeStyleKey);
label.SetResourceReference(Label.StyleProperty,
    VsResourceKeys.LabelEnvironment200PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de `TextBlock` o `Label` como se muestra:

```
<TextBlock Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.TextBlockEnvironment200PercentFontSizeStyleKey}}">TextBlock: 200 Percent  
Scaling</TextBlock>  
<Label Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.LabelEnvironment200PercentFontSizeStyleKey}}">Label: 200 Percent Scaling</Label>
```

155% Fuente de medio ambiente

Aparece como: 14 pt Segoe UI Use for: section headings in document well UI or reports

Código procesal: Dónde `textBlock` es un TextBlock definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,  
    VsResourceKeys.TextBlockEnvironment155PercentFontSizeStyleKey);  
label.SetResourceReference(Label.StyleProperty,  
    VsResourceKeys.LabelEnvironment155PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de TextBlock o Label como se muestra:

```
<TextBlock Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.TextBlockEnvironment155PercentFontSizeStyleKey}}">TextBlock: 155 Percent  
Scaling</TextBlock>  
<Label Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.LabelEnvironment155PercentFontSizeStyleKey}}">Label: 155 Percent Scaling</Label>
```

133% Fuente Environment

Aparece como: 12 pt Segoe UI Use for: subheadings más pequeños en los cuadros de diálogo de firma y documento bien UI

Código procesal: Dónde `textBlock` es un TextBlock definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,  
    VsResourceKeys.TextBlockEnvironment133PercentFontSizeStyleKey);  
label.SetResourceReference(Label.StyleProperty,  
    VsResourceKeys.LabelEnvironment133PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de TextBlock o Label como se muestra:

```
<TextBlock Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.TextBlockEnvironment133PercentFontSizeStyleKey}}">TextBlock: 133 Percent  
Scaling</TextBlock>  
<Label Style="{DynamicResource {x:Static  
vsui:VsResourceKeys.LabelEnvironment133PercentFontSizeStyleKey}}">Label: 133 Percent Scaling</Label>
```

122% Fuente De entorno

Aparece como: 11 pt Segoe UI Use for: section headings in signature dialogs, top nodes in tree view, vertical tab navigation

Código procesal: Dónde `textBlock` es un TextBlock definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,
    VsResourceKeys.TextBlockEnvironment122PercentFontSizeStyleKey);
label.SetResourceReference(Label.StyleProperty,
    VsResourceKeys.LabelEnvironment122PercentFontSizeStyleKey);
```

XAML: Establezca el estilo de TextBlock o Label como se muestra:

```
<TextBlock Style="{DynamicResource {x:Static
vsui:VsResourceKeys.TextBlockEnvironment122PercentFontSizeStyleKey}}">TextBlock: 122 Percent
Scaling</TextBlock>
<Label Style="{DynamicResource {x:Static
vsui:VsResourceKeys.LabelEnvironment122PercentFontSizeStyleKey}}">Label: 122 Percent Scaling</Label>
```

Fuente del entorno + negrita

Aparece como: bolded 9 pt Segoe UI **Use for:** labels and subheads in signature dialogs, reports, and document well UI

Código procesal: Dónde `textBlock` es un TextBlock definido previamente y `label` es una etiqueta definida previamente:

```
textBlock.SetResourceReference(TextBlock.StyleProperty,
    VsResourceKeys.TextBlockEnvironmentBoldStyleKey);
label.SetResourceReference(Label.StyleProperty,
    VsResourceKeys.LabelEnvironmentBoldStyleKey);
```

XAML: Establezca el estilo de TextBlock o Label como se muestra:

```
<TextBlock Style="{DynamicResource {x:Static vsui:VsResourceKeys.TextBlockEnvironmentBoldStyleKey}}"> Bold
TextBlock</TextBlock>
<Label Style="{DynamicResource {x:Static vsui:VsResourceKeys.LabelEnvironmentBoldStyleKey}}"> Bold
Label</Label>
```

Estilos localizables

En algunos casos, los localizadores tendrán que modificar los estilos de fuente para diferentes configuraciones regionales, como la eliminación de negritas del texto para idiomas de Asia oriental. Para que la localización de estilos de fuente sea posible, esos estilos deben estar dentro del archivo .resx. La mejor manera de lograr esto y seguir editando estilos de fuente en el Diseñador de formularios de Visual Studio es establecer explícitamente los estilos de fuente en tiempo de diseño. Aunque esto crea un objeto de fuente completo y puede parecer romper la herencia de fuentes primarias, solo el FontStyle propiedad se utiliza para establecer la fuente.

La solución es enlazar el `FontChanged` evento del formulario de diálogo. En `FontChanged` el caso, pasee todos los controles y compruebe si su fuente está establecida. Si se establece, cámbielo a una nueva fuente basada en la fuente del formulario y el estilo de fuente anterior del control. Un ejemplo de esto en el código es:

```

private void Form1_FontChanged(object sender, System.EventArgs e)
{
    SetFontStyles();
}

/// <summary>
/// SetFontStyles - This function will iterate all controls on a page
/// and recreate their font with the desired fontstyle.
/// It should be called in the OnFontChanged handler (and also in the constructor
/// in case the IUIService is not available so OnFontChange doesn't fire).
/// This way, when the VS shell font is given to us the controls that have
/// a different style for the font (bolded for example) will recreate their font
/// and use the VS shell font but with a style variation (bolded ...).
/// </summary>
protected void SetFontStyles()
{
    SetFontStyles(this, this, this.Font);
}

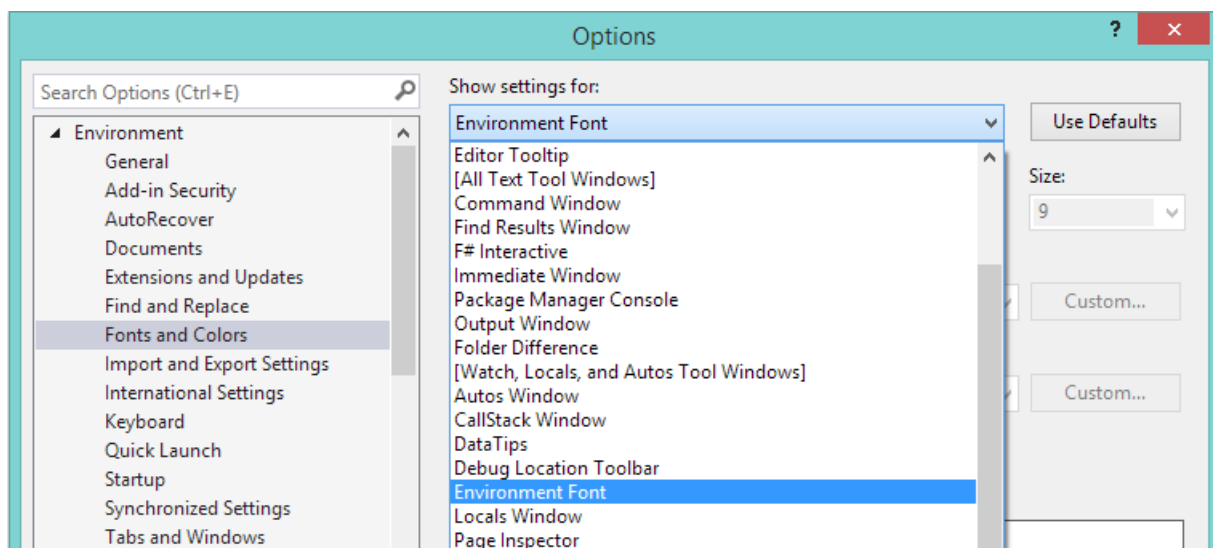
protected static void SetFontStyles(Control topControl, Control parent, Font referenceFont)
{
    foreach(Control c in parent.Controls)
    {
        if (c.Controls != null && c.Controls.Count > 0) {
            SetFontStyles(topControl, c, referenceFont);
        }
        if (c.Font != topControl.Font) {
            c.Font = new Font(referenceFont, c.Font.Style);
        }
    }
}
}

```

El uso de este código garantiza que cuando se actualiza la fuente del formulario, las fuentes de los controles también se actualizarán. Este método también debe llamarse desde el constructor del formulario, porque `IUIService` el `FontChanged` cuadro de diálogo podría no obtener una instancia de y el evento nunca se desencadenará. El `FontChanged` enlace permitirá que los cuadros de diálogo recojan dinámicamente la nueva fuente incluso si el cuadro de diálogo ya está abierto.

Probar la fuente del entorno

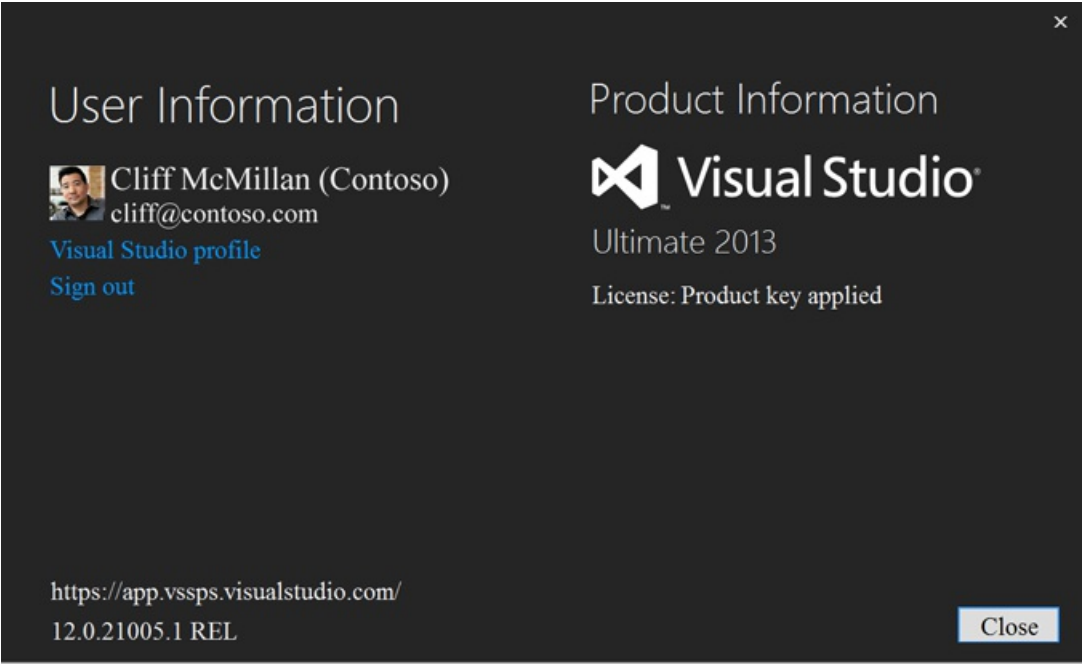
Para asegurarse de que la interfaz de usuario utiliza la fuente de entorno y respeta la configuración de tamaño, abra **Herramientas > Opciones > Entorno > Fuentes y colores** y seleccione "Fuente de entorno" en el menú desplegable "Mostrar configuración para:".



Configuración de fuentes y > colores en el cuadro de diálogo Opciones de herramientas

Establezca la fuente en algo muy diferente del valor predeterminado. Para que sea obvio qué interfaz de usuario

no se actualiza, elija una fuente con serifs (como "Times New Roman") y establezca un tamaño muy grande. A continuación, pruebe la interfaz de usuario para asegurarse de que respeta el entorno. A continuación se muestra un ejemplo utilizando el cuadro de diálogo de licencia:



Ejemplo de texto de interfaz de usuario que no respeta la fuente del entorno

En este caso, "Información del usuario" y "Información del producto" no respetan la fuente. En algunos casos, esto puede ser una opción de diseño explícita, pero puede ser un error si la fuente explícita no se especifica como parte de las especificaciones de línea roja.

Para restablecer la fuente, haga clic en "Usar valores predeterminados" en **Herramientas > Opciones > Entorno > Fuentes y Colores**.

Estilo de texto

El estilo de texto hace referencia al tamaño de fuente, el peso y la carcasa. Para obtener instrucciones sobre la implementación, consulte [La fuente de entorno](#).

Carcasa de texto

Todas las gorras

No use todas las mayúsculas para títulos o etiquetas en Visual Studio.

Todas las minúsculas

No use todas las minúsculas para títulos o etiquetas en Visual Studio.

Sentencia y estuche de título

El texto de Visual Studio debe usar mayúsculas y minúsculas de título o de frase, dependiendo de la situación.

UTILICE EL CASO DE TÍTULO PARA:	USE EL CASO DE LA SENTENCIA PARA:
Títulos de diálogo	Etiquetas
Cajas de grupo	Casillas
Elementos de menú	Botones de radio
Elementos del menú contextual	Elementos del cuadro de lista

UTILICE EL CASO DE TÍTULO PARA:	USE EL CASO DE LA SENTENCIA PARA:
Botones	Barras de estado
Etiquetas de mesa	
Encabezados de columna	
Información sobre herramientas	

Tipo título

El caso de título es un estilo en el que las primeras letras de la mayoría o todas las palabras dentro de una frase están en mayúsculas. En Visual Studio, el caso de título se usa para muchos elementos, entre ellos:

- **Tooltips.** Ejemplo: "Vista previa de los elementos seleccionados"
- **Encabezados de columna.** Ejemplo: "Respuesta del sistema"
- **Elementos del menú.** Ejemplo: "Guardar todo"

Al usar mayúsculas y minúsculas, estas son las pautas para cuándo poner las palabras en mayúsculas y cuándo dejarlas en minúsculas:

UPPERCASE	COMENTARIOS Y EJEMPLOS
Todos los sustantivos	
Todos los verbos	Incluyendo "Is" y otras formas de "ser"
Todos los adverbios	Incluyendo "Than" y "When"
Todos los adjetivos	Incluyendo "Esto" y "Eso"
Todos los pronombres	Incluyendo el posesivo "Its" así como "It's", una contracción del pronombre "it" y el verbo "is"
Primera y última palabra, independientemente de las partes del habla	
Preposiciones que forman parte de una frase verbal	"Cerrando todas las ventanas" o "Apagar el sistema"
Todas las letras de un acrónimo	HTML, XML, URL, IDE, RGB
La segunda palabra en una palabra compuesta si es un sustantivo o un adjetivo adecuado, o si las palabras tienen el mismo peso	Referencia cruzada, software de Pre-Microsoft, acceso de lectura/escritura, tiempo de ejecución

MINÚSCULAS	EJEMPLOS
La segunda palabra en una palabra compuesta si es otra parte del habla o un participio modificando la primera palabra	Cómo hacerlo, despegue
Artículos, a menos que uno sea la primera palabra en el título	un, una, el, la
Coordinar conjunciones	y, pero, por, ni, o

MINÚSCULAS	EJEMPLOS
Preposiciones con palabras de cuatro o menos letras fuera de una frase verbal	en, en, en, como para, fuera de, en la parte superior de
"Para" cuando se usa en una frase infinitiva	"Cómo formatear su disco duro"

Caso de sentencia

El caso de la sentencia es el método de mayúsculas estándar para escribir en el que sólo la primera palabra de la oración está en mayúsculas, junto con cualquier sustantivo apropiado y el pronombre "I". En general, el caso de oraciones es más fácil de leer para una audiencia mundial, especialmente cuando el contenido será traducido por una máquina. Use el caso de la sentencia para:

1. **Mensajes de la barra de estado.** Estos son simples, cortos y proporcionan solo información de estado. Ejemplo: "Cargar archivo de proyecto"
2. **Todos los demás elementos** de la interfaz de usuario, incluidas las etiquetas, las casillas de verificación, los botones de opción y los elementos del cuadro de lista. Ejemplo: "Seleccionar todos los elementos de la lista"

Formato del texto

El formato de texto predeterminado en Visual Studio 2013 se controla mediante [la fuente de entorno](#). Este servicio ayuda a garantizar una apariencia de fuente coherente en todo el IDE (entorno de desarrollo integrado) y debe usarlo para garantizar una experiencia coherente para los usuarios.

El tamaño predeterminado utilizado por el servicio de fuentes de Visual Studio proviene de Windows y aparece como 9 pt.

Puede aplicar formato a la fuente del entorno. En este tema se explica cómo y dónde usar estilos. Para obtener información sobre la implementación, consulte [La fuente de entorno](#).

Texto en negrita

El texto en negrita se usa con moderación en Visual Studio y debe reservarse para:

- etiquetas de preguntas en los magos
- designar el proyecto activo en el Explorador de soluciones
- valores anulados en la ventana de herramientas Propiedades
- ciertos eventos en las listas desplegables del editor de Visual Basic
- contenido generado por el servidor en el esquema del documento para las páginas web
- encabezados de sección en un cuadro de diálogo complejo o en la interfaz de usuario del diseñador

Cursiva

Visual Studio no usa texto en cursiva o en cursiva en negrita.

Color

- El azul está reservado para hipervínculos (navegación y comandos) y nunca se debe utilizar para la orientación.
- Los encabezados más grandes (fuente de entorno x 155% o superior) se pueden colorear para estos fines:
 - Para proporcionar atractivo visual a la interfaz de usuario de Visual Studio de firma
 - Para llamar la atención sobre un área específica
 - Para ofrecer alivio del color de texto estándar del entorno gris oscuro/negro

- El color en los encabezados debe aprovechar los colores de marca de Visual Studio existentes, principalmente el púrpura principal, #FF68217A.
- Al usar el color en los encabezados, debe cumplir las directrices de color de [Windows](#), incluida la relación de contraste y otras consideraciones de accesibilidad.

Tamaño de fuente

El diseño de la interfaz de usuario de Visual Studio presenta un aspecto más claro con más espacio en blanco. Siempre que sea posible, las barras de cromo y título se han reducido o eliminado. Aunque la densidad de la información es un requisito en Visual Studio, la tipografía sigue siendo importante, con énfasis en el espaciado de línea más abierto y una variación de los tamaños y pesos de fuente.

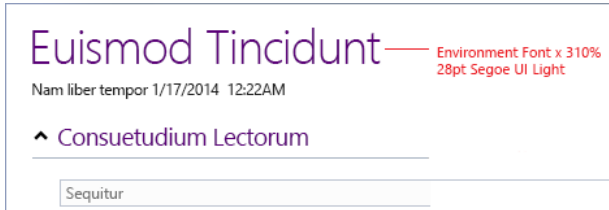
Las tablas siguientes incluyen detalles de diseño y ejemplos visuales para las fuentes de presentación utilizadas en Visual Studio. Algunas variaciones de fuente de visualización tienen el tamaño y el peso, como Semilight o Light, codificados en su apariencia.

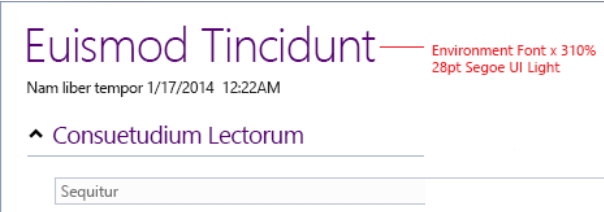
Los fragmentos de código de implementación para todas las fuentes de visualización se pueden encontrar en [la referencia Formato \(escalado/en negrita\)](#).

375% Fuente de entorno + Luz

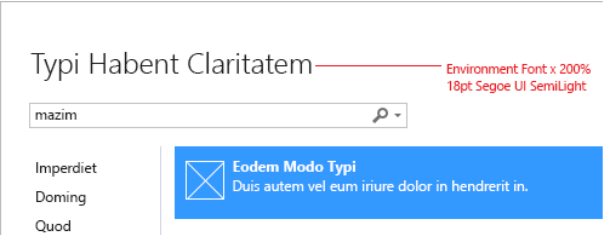
Uso: Raro. Solo interfaz de usuario de marca única. hacer: - Usar caso de sentencia - Utilice siempre peso ligero No: - Uso para la interfaz de usuario que no sea la interfaz de usuario de firma, como la página de inicio - Negrita, cursiva o negrita cursiva - Uso para el texto del cuerpo - Uso en ventanas de herramientas	Aparece como: 34 pt Segoe UI Light Ejemplo visual: <i>Actualmente no se utiliza. Se puede usar en la página de inicio de Visual Studio 2017.</i>

310% Fuente de entorno + Luz

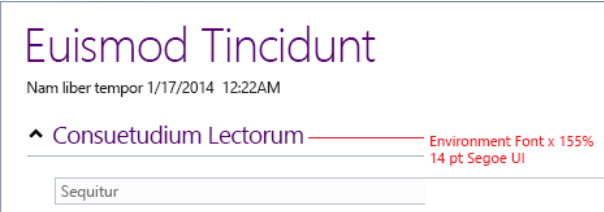
Uso: - Encabezado más grande en los diálogos de firma - Encabezado principal del informe hacer: - Usar caso de sentencia - Utilice siempre peso ligero No: - Uso para la interfaz de usuario que no sea la interfaz de usuario de firma, como la página de inicio - Negrita, cursiva o negrita cursiva - Uso para el texto del cuerpo - Uso en ventanas de herramientas	Aparece como: 28 pt Segoe UI Light Ejemplo visual: 

Uso: - Encabezado más grande en los diálogos de firma - Encabezado principal del informe hacer: - Usar caso de sentencia - Utilice siempre peso ligero No: - Usar para la interfaz de usuario que no sea la interfaz de usuario de firma - Negrita, cursiva o negrita cursiva - Uso para el texto del cuerpo - Uso en ventanas de herramientas	Aparece como: 28 pt Segoe UI Light Ejemplo visual: 

200% Fuente de entorno + Semilight

Uso: - Subpartidas - Títulos en diálogos pequeños y medianos hacer: - Usar caso de sentencia - Utilice siempre el peso semiligero No: - Negrita, cursiva o negrita cursiva - Uso para el texto del cuerpo - Uso en ventanas de herramientas	Aparece como: 18 pt Segoe UI Semilight Ejemplo visual: 

155% Fuente de medio ambiente

Uso: - Encabezados de sección en la interfaz de usuario de documentos bien - Informes Hacer: Usar caso de sentencia No: - Negrita, cursiva o negrita cursiva - Uso para el texto del cuerpo - Uso en controles estándar de Visual Studio - Uso en ventanas de herramientas	Aparece como: 14 pt Segoe UI Ejemplo visual: 

133% Fuente Environment

Uso:

- Subpartidas más pequeñas en los diálogos de firmas
- Subpartidas más pequeñas en la interfaz de usuario de pozos de documentos

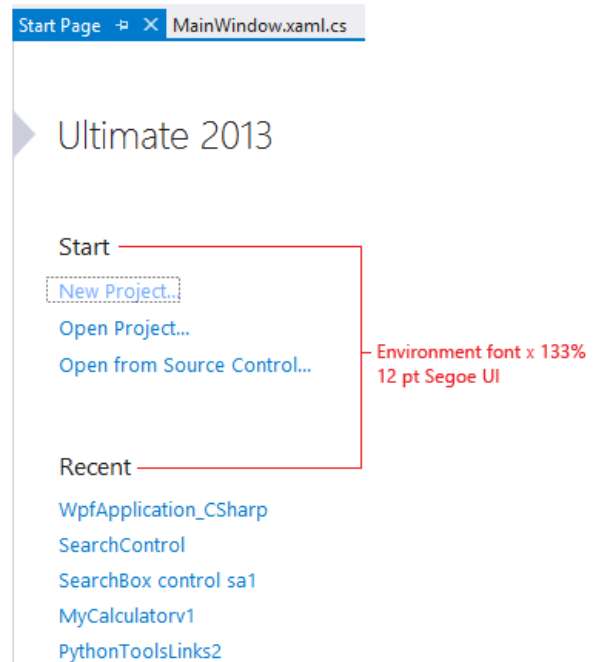
Hacer: Usar caso de sentencia

No:

- Negrita, cursiva o negrita cursiva
- Uso para el texto del cuerpo
- Uso en controles estándar de Visual Studio
- Uso en ventanas de herramientas

Aparece como: 12 pt Segoe UI

Ejemplo visual:



122% Fuente De entorno

Uso:

- Encabezados de sección en diálogos de firmas
- Nodos superiores en la vista de árbol
- Navegación vertical por pestañas

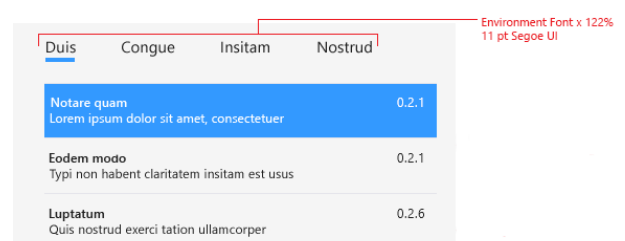
Hacer: Usar caso de sentencia

No:

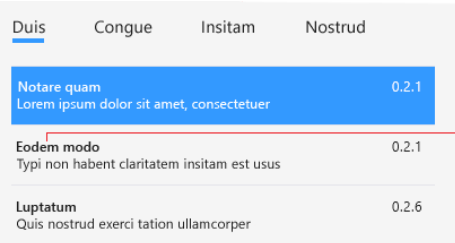
- Negrita, cursiva o negrita cursiva
- Uso para el texto del cuerpo
- Uso en controles estándar de Visual Studio
- Uso en ventanas de herramientas

Aparece como: 11 pt Segoe UI

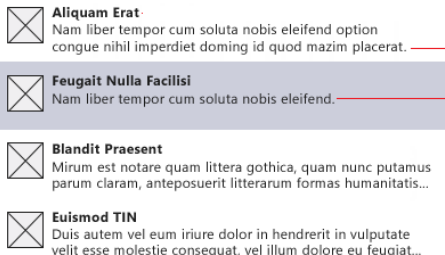
Ejemplo visual:



Fuente del entorno + negrita

Uso: - Etiquetas y subjeses en diálogos de firma - Etiquetas y subjeses en los informes - Etiquetas y subjeses en la interfaz de usuario de documentos bien hacer: - Usar caso de sentencia - Utilice el peso audaz No: - Cursiva o cursiva en negrita - Uso para el texto del cuerpo - Uso en controles estándar de Visual Studio - Uso en ventanas de herramientas	Aparece como: bolded 9 pt Segoe UI Ejemplo visual: 

Fuente de entorno

Uso: El resto del texto Hacer: Usar caso de sentencia No: Cursiva o cursiva en negrita	Aparece como: 9 pt Segoe UI Ejemplo visual: 

Relleno y espaciado

Los encabezados requieren espacio a su alrededor para darles el énfasis adecuado. Este espacio varía en función del tamaño del punto y de lo que esté cerca del encabezado, como una regla horizontal o una línea de texto en la fuente del entorno.

- El relleno ideal para un encabezado por sí mismo debe ser el 90% del espacio de altura del carácter capital. Por ejemplo, un encabezado Segoe UI Light de 28 pt tiene una altura de límite de 26 pt, y el relleno debe ser de aproximadamente 23 pt, o unos 31 píxeles.
- El espacio mínimo alrededor de un encabezado debe ser el 50% de la altura del carácter capital. Se puede utilizar menos espacio cuando un encabezado va acompañado de una regla u otro elemento de ajuste ajustado.
- El texto de fuente de entorno en negrita debe seguir el espaciado de altura de línea y el relleno predeterminados.

Vea también

- [MSDN: Fuentes \(Windows\)](#)
- [MSDN: Texto de interfaz de usuario \(Windows\)](#)

Colores y estilos para Visual Studio

06/04/2020 • 38 minutes to read • [Edit Online](#)

Usar color en Visual Studio

En Visual Studio, el color se usa principalmente como una herramienta de comunicación, no solo como decoración. Utilice el color mínimamente y resérvelo para situaciones en las que desee:

- Comunicar significado o afiliación (por ejemplo, modificadores de plataforma o idioma)
- Atraer la atención (por ejemplo, indicando un cambio de estado)
- Mejorar la legibilidad y proporcionar puntos de referencia para navegar por la interfaz de usuario
- Aumentar la deseabilidad

Existen varias opciones para asignar colores a elementos de interfaz de usuario en Visual Studio. A veces puede ser difícil averiguar qué opción se supone que debes usar o cómo usarla correctamente. Este tema le ayudará a:

- Comprender los diferentes servicios y sistemas utilizados para definir colores en Visual Studio.
- Seleccione la opción correcta para un elemento determinado.
- Utilice correctamente la opción que ha elegido.

NOTE

Nunca codifique los colores hexadecimales, RGB o del sistema a los elementos de la interfaz de usuario. El uso de los servicios permite flexibilidad en el tono de afinación. Además, sin el servicio, no podrá aprovechar las capacidades de conmutación de temas [del servicio VSColor](#).

Métodos para asignar color a elementos de interfaz de Visual Studio

Elija el método que mejor se adapte a los elementos de la interfaz de usuario.

TU INTERFAZ DE USUARIO	MÉTODO	¿QUÉ SON?
Tiene cuadros de diálogo incrustados o independientes.	Colores del sistema	Nombres de sistema que permiten al sistema operativo definir el color y la apariencia de los elementos de la interfaz de usuario, como los controles de cuadro de diálogo comunes.
Tiene una interfaz de usuario personalizada que desea que sea coherente con el entorno VS general y tiene elementos de interfaz de usuario que coinciden con la categoría y el significado semántico de los tokens compartidos.	Colores compartidos comunes	Nombres de token de color predefinidos existentes para elementos específicos de la interfaz de usuario
Tiene una entidad o grupo de entidades individual y no hay ningún color compartido para elementos similares.	Colores personalizados	Nombres de token de color que son específicos de un área y no están destinados a compartirse con otra interfaz de usuario

TU INTERFAZ DE USUARIO	MÉTODO	¿QUÉ SON?
Desea permitir que el usuario final personalice la interfaz de usuario o el contenido (por ejemplo, para editores de texto o ventanas de diseñador especializadas).	Personalización del usuario final (diálogo > Opciones de herramientas)	Configuración definida en la página "Fuentes y colores" del cuadro de diálogo ** > Opciones** de herramientas o en una página especializada específica de una característica de interfaz de usuario.

Temas de Visual Studio

Visual Studio presenta tres temas de color diferentes: claro, oscuro y azul. También detecta el modo de alto contraste, que es un tema de color de todo el sistema diseñado para la accesibilidad.

Se pide a los usuarios que seleccionen un tema durante su primer uso de Visual Studio y podrán cambiar de tema más adelante yendo a **Opciones > > ** de herramientas Entorno general y eligiendo un nuevo tema en el menú desplegable "tema de color".

Los usuarios también pueden utilizar el Panel de control para cambiar todos sus sistemas en uno de varios temas de alto contraste. Si un usuario selecciona un tema de contraste alto, el selector de tema de color de Visual Studio ya no afecta a los colores en Visual Studio, aunque los cambios de tema se guardan para cuando el usuario sale del modo de contraste alto. Para obtener más información sobre el modo contraste alto, consulte [Elegir colores](#) de contraste alto .

El servicio VSColor

Visual Studio proporciona un servicio de color de entorno, conocido como el servicio VSColor, que permite enlazar los valores de color de los elementos de interfaz de usuario a una entrada con nombre que contiene valores de color para cada tema de Visual Studio. Esto garantiza que los colores cambiarán automáticamente para reflejar el tema seleccionado por el usuario actual o el modo de contraste alto del sistema. El uso del servicio significa que la implementación de todos los cambios de color relacionados con el tema se controla en un solo lugar y, si usa colores compartidos comunes del servicio, la interfaz de usuario reflejará automáticamente nuevos temas en versiones futuras de Visual Studio.

Implementación

El código fuente de Visual Studio incluye varios archivos de definición de paquete que contienen listas de nombres de token y los valores de color respectivos para cada tema. El servicio de color lee los VSColors definidos en estos archivos de definición de paquete. Se hace referencia a estos colores en el `IVsUIShell15.GetThemedColor` marcado XAML o en el código y, a continuación, se cargan a través del método o una asignación DynamicResource.

Colores del sistema

Los controles comunes hacen referencia a los colores del sistema de forma predeterminada. Si desea que la interfaz de usuario use colores del sistema, como cuando está creando un cuadro de diálogo incrustado o independiente, no es necesario hacer nada.

Colores compartidos comunes en el servicio VSColor

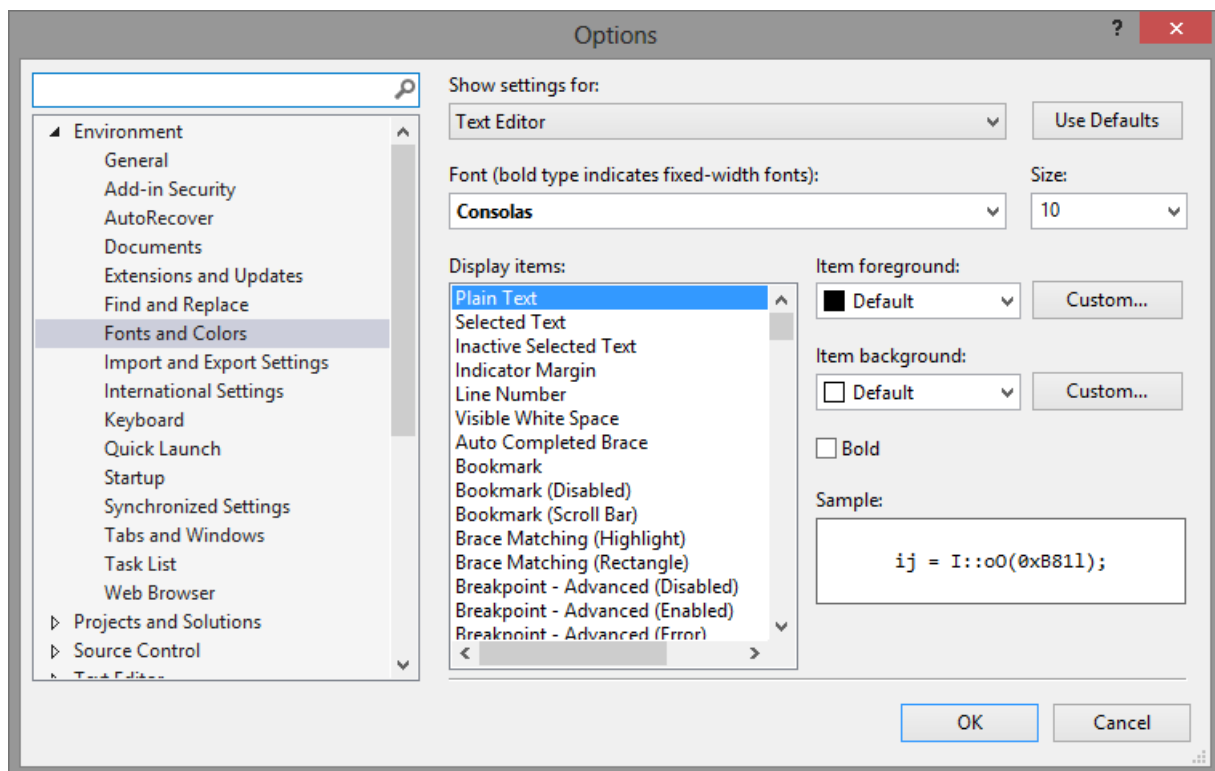
Los elementos de la interfaz deben reflejar el entorno general de Visual Studio. Al reutilizar los colores compartidos comunes que son adecuados para el componente de interfaz de usuario que está diseñando, se asegura de que la interfaz es coherente con otras interfaces de Visual Studio y que los colores se actualizarán automáticamente cuando se agreguen o actualicen temas.

Antes de usar colores compartidos comunes, asegúrese de entender cómo usarlos correctamente. El uso incorrecto de colores compartidos comunes puede dar lugar a una experiencia incoherente, frustrante o confusa para los usuarios.

Colores personalizables por el usuario

Consulte: [Exposición de colores para los usuarios finales](#)

A veces, querrá permitir que el usuario final personalice la interfaz de usuario, como cuando está creando un editor de código o una superficie de diseño. Los componentes de interfaz de usuario personalizables se encuentran en la sección **Fuentes y colores** del cuadro de diálogo ** > Opciones** de herramientas, donde los usuarios pueden elegir cambiar el color de primer plano, el color de fondo o ambos.



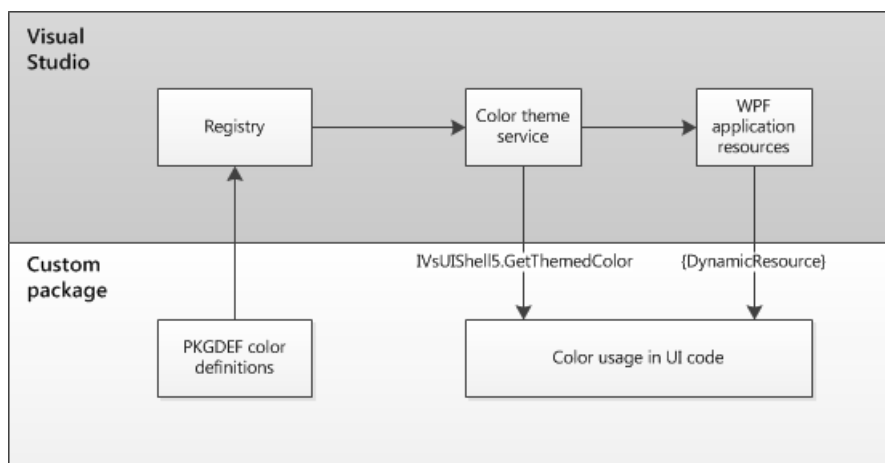
Cuadro > de diálogo Opciones de herramientas

El servicio VSColor

Visual Studio proporciona un servicio de color de entorno, también denominado el servicio VSColor o el servicio de color de shell. Este servicio le permite enlazar los valores de color de los elementos de la interfaz de usuario a un conjunto de colores nombre-valor que contiene colores para cada tema. El servicio VSColor se debe usar para todos los elementos de la interfaz de usuario, de modo que los colores cambien automáticamente para reflejar el tema seleccionado por el usuario actual y para que la interfaz de usuario enlazada al servicio de color del entorno se integre con nuevos temas en versiones futuras de Visual Studio.

Funcionamiento del servicio

El servicio de color de entorno lee VSColors definidos en el .pkgdef para el componente de interfaz de usuario. A continuación, se hace referencia a estos VSColors `IVsUIShell15.GetThemedColor` en `DynamicResource` el código o marcado XAML y se cargan a través de la asignación o una.



Arquitectura de servicio de color del entorno

Acceso al servicio

Hay varias maneras diferentes de tener acceso al servicio VSColor, dependiendo del tipo de tokens de color que esté utilizando y qué tipo de código tiene.

Colores predefinidos del entorno

Desde código nativo

El shell proporciona un servicio `COLORREF` que da acceso a los colores. El servicio/interfaz es:

```
IVsUIShell12::GetVSSysColorEx(VSSYSCOLOR dwSysColIndex, DWORD *pdwRGBval)
```

En el archivo VSShell80.idl, la enumeración `__VSSYSCOLOREX` tiene constantes de color de shell. Para usarlo, pase como valor de índice uno `enum __VSSYSCOLOREX` de los valores del documentado en MSDN `GetSysColor` o un número de índice normal que la API del sistema de Windows, , acepta. Al hacerlo, se recupera el valor RGB del color que se debe utilizar en el segundo parámetro.

Si almacena un lápiz o pincel con `AdviseBroadcastMessages` un nuevo color, debe (fuera del shell de Visual Studio) y escuchar `WM_SYSCOLORCHANGE` y `WM_THEMECHANGED` mensajes.

Para tener acceso al servicio de color en código nativo, hará una llamada similar a esta:

```
pUIShell12->GetVSSysColorEx(VSCOLOR_COLOR_NAME, &rgbLOCAL_COLOR);
```

NOTE

Los `COLORREF` valores `GetVSSysColorEx()` devueltos por contienen sólo componentes R,G,B de un color de tema. Si una entrada de tema utiliza transparencia, el valor de canal alfa se descarta antes de devolver. Por lo tanto, si el color del entorno de interés debe utilizarse en un lugar donde el canal de transparencia es importante, debe usar

`IVsUIShell15.GetThemedColor` en lugar de `IVsUIShell12::GetVSSysColorEx` , como se describe más adelante en este tema.

Desde código administrado

El acceso al servicio VSColor a través de código nativo es bastante sencillo. Sin embargo, si está trabajando a través de código administrado, determinar cómo usar el servicio puede ser complicado. Teniendo esto en cuenta, aquí hay un fragmento de código de C- que demuestra este proceso:

```
private void VSColorPaint(object sender, System.Windows.Forms.PaintEventArgs e)
{
    //getIVsUIShell12
    IVsUIShell12 uiShell12 = Package.GetService(typeof(SVsUIShell)) as IVsUIShell12;
    Debug.Assert (uiShell12 != null, "failed to get IVsUIShell12");

    if (uiShell12 != null)
    {
        //get the COLORREF structure
        uint win32Color;
        uiShell12.GetVSSysColorEx((int)__VSSYSCOLOREX.VSCOLOR_SMARTTAG_HOVER_FILL, out win32Color);

        //translate it to a managed Color structure
        Color myColor = ColorTranslator.FromWin32((int)win32Color);
        //use it
        e.Graphics.FillRectangle(new SolidBrush(myColor), 0, 0, 100, 100);
    }
}
```

Si está trabajando en Visual Basic, utilice:

```
Dim myColor As Color = ColorTranslator.FromWin32((Integer)win32Color)
```

Desde la interfaz de usuario de WPF

Puede enlazar a colores de Visual Studio `ResourceDictionary` a través de valores exportados a la aplicación . A continuación se muestra un ejemplo del uso de recursos de la tabla de colores, así como el enlace a los datos de fuente de entorno en XAML.

```
<Style TargetType="{x:Type Button}">
    <Setter Property="TextBlock.FontFamily"
        Value="{DynamicResource VsFont.EnvironmentFontFamily}" />
    <Setter Property="TextBlock.FontSize"
        Value="{DynamicResource VsFont.EnvironmentFontSize}" />
    <Setter Property="Background"
        Value="{DynamicResource VsBrush.EnvironmentBackgroundGradient}" />
</Style>
```

Clases y métodos auxiliares para código administrado

Para el código administrado, la biblioteca `Microsoft.VisualStudio.Shell.12.0.dll` de Managed Package Framework del shell () contiene un par de clases auxiliares que facilitan el uso de colores temáticos.

Los métodos `Microsoft.VisualStudio.Shell.VsColors` auxiliares de `GetThemedGDIColor()` la `GetThemedWPFColor()` clase en MPF incluyen y . Esos métodos auxiliares devuelven el `System.Drawing.Color` valor `System.Windows.Media.Color` de color de una entrada de tema como o , que se usará en WinForms o WPF WPF UI.

```

IVsUIShell15 shell15;
Button button = new Button();
button.BackColor = GetThemedGDIColor(shell15, SolutionExplorerColors.SelectedItemBrushKey);
button.ForeColor = GetThemedGDIColor(shell15, SolutionExplorerColors.SelectedItemTextBrushKey);

/// <summary>
/// Gets a System.Drawing.Color value from the current theme for the given color key.
/// </summary>
/// <param name="vsUIShell">The IVsUIShell15 service, used to get the color's value.</param>
/// <param name="themeResourceKey">The key to find the color for.</param>
/// <returns>The current theme's value of the named color.</returns>
public static System.Drawing.Color GetThemedGDIColor(this IVsUIShell15 vsUIShell, ThemeResourceKey
themeResourceKey)
{
    Validate.IsNotNull(vsUIShell, "vsUIShell");
    Validate.IsNotNull(themeResourceKey, "themeResourceKey");

    byte[] colorComponents = GetThemedColorRgba(vsUIShell, themeResourceKey);

    // Note: The Win32 color we get back from IVsUIShell15.GetThemedColor is ABGR
    return System.Drawing.Color.FromArgb(colorComponents[3], colorComponents[0], colorComponents[1],
colorComponents[2]);
}

private static byte[] GetThemedColorRgba(IVsUIShell15 vsUIShell, ThemeResourceKey themeResourceKey)
{
    Guid category = themeResourceKey.Category;
    __THEMEDCOLORTYPE colorType = __THEMEDCOLORTYPE.TCT_Foreground
    if (themeResourceKey.KeyType == ThemeResourceKeyType.BackgroundColor || themeResourceKey.KeyType ==
ThemeResourceKeyType.BackgroundBrush)
    {
        colorType = __THEMEDCOLORTYPE.TCT_Background;
    }

    // This call will throw an exception if the color is not found
    uint rgbaColor = vsUIShell.GetThemedColor(ref category, themeResourceKey.Name, (uint)colorType);
    return BitConverter.GetBytes(rgbaColor);
}
public static System.Windows.Media.Color GetThemedWPFColor(this IVsUIShell15 vsUIShell, ThemeResourceKey
themeResourceKey)
{
    Validate.IsNotNull(vsUIShell, "vsUIShell");
    Validate.IsNotNull(themeResourceKey, "themeResourceKey");

    byte[] colorComponents = GetThemedColorComponents(vsUIShell, themeResourceKey);

    return System.Windows.Media.Color.FromArgb(colorComponents[3], colorComponents[0], colorComponents[1],
colorComponents[2]);
}

```

La clase también se puede usar para obtener identificadores VSCOLOR para una clave de recurso de color WPFWPF determinada, o viceversa.

```

public static string GetColorBaseKey(int vsSysColor);
public static bool TryGetColorIDFromBaseKey(string baseKey, out int vsSysColor);

```

Los métodos de `VsColors` clase consultan el servicio VSColor para devolver el valor de color cada vez que se invocan. Para obtener un `System.Drawing.Color` valor de color como , una alternativa `Microsoft.VisualStudio.PlatformUI.VSThemeColor` con un mejor rendimiento es utilizar en su lugar los métodos de la clase, que almacena en caché los valores de color obtenidos del servicio VSColor. La clase se suscribe internamente a los eventos de mensajes de difusión de shell y descarta el valor almacenado en caché cuando se produce un evento de cambio de tema. Además, la clase proporciona un archivo . Evento compatible con NET para

suscribirse a los cambios de tema. Utilice `ThemeChanged` el evento para agregar un `GetThemedColor()` nuevo controlador y use `ThemeResourceKeys` el método para obtener valores de color para el de interés. Un código de ejemplo podría tener este aspecto:

```
public MyWindowPanel()
{
    InitializeComponent();

    // Subscribe to theme changes events so we can refresh the colors
    VSColorTheme.ThemeChanged += VSColorTheme_ThemeChanged;

    RefreshColors();
}

private void VSColorTheme_ThemeChanged(ThemeChangedEventArgs e)
{
    RefreshColors();

    // Also post a message to all the children so they can apply the current theme appropriately
    foreach (System.Windows.Forms.Control child in this.Controls)
    {
        NativeMethods.SendMessage(child.Handle, e.Message, IntPtr.Zero, IntPtr.Zero);
    }
}

private void RefreshColors()
{
    this.BackColor = VSColorTheme.GetThemedColor(EnvironmentColors.ToolWindowBackgroundColorKey);
    this.ForeColor = VSColorTheme.GetThemedColor(EnvironmentColors.ToolWindowTextColorKey);
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        VSColorTheme.ThemeChanged -= this.VSColorTheme_ThemeChanged;
        base.Dispose(disposing);
    }
}
```

Elegir colores de alto contraste

Información general

Windows utiliza varios temas de alto contraste a nivel del sistema que aumentan el contraste de color del texto, los fondos y las imágenes, lo que hace que los elementos parezcan más distintos en la pantalla. Por razones de accesibilidad, es importante que los elementos de la interfaz de Visual Studio respondan correctamente cuando los usuarios cambian a un tema de contraste alto.

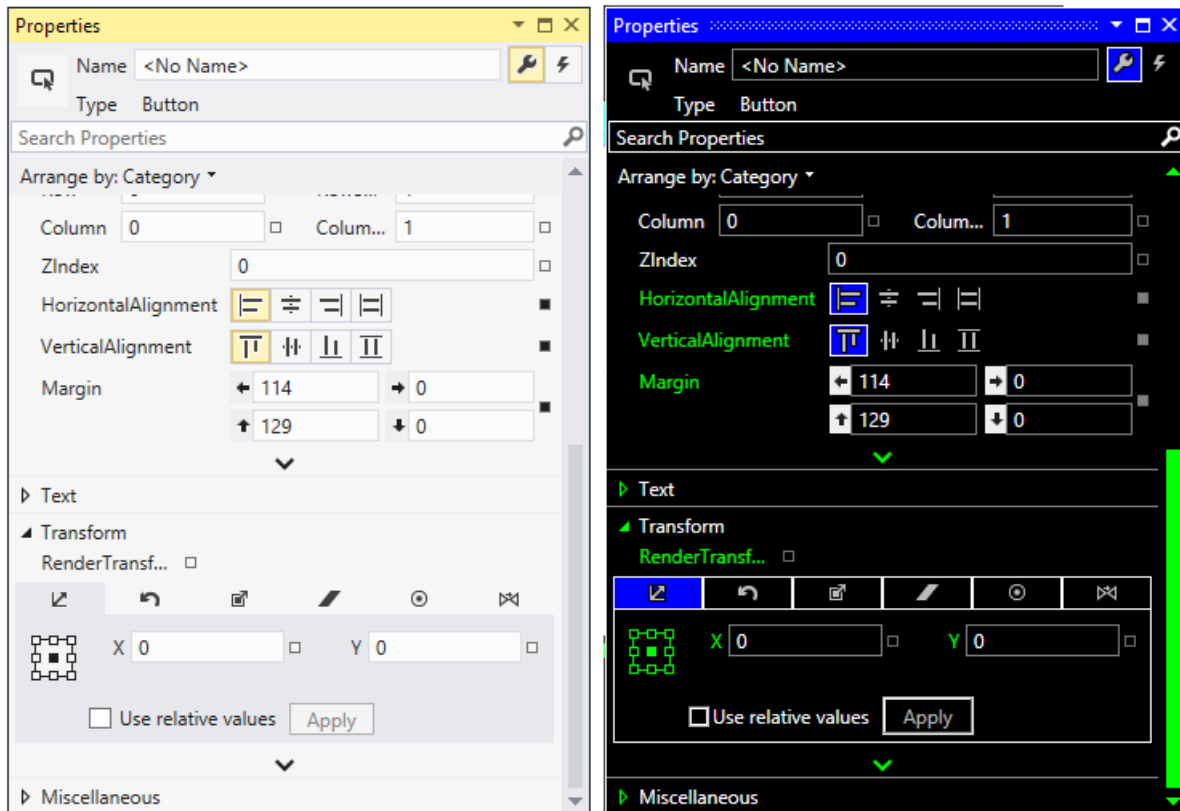
Solo se pueden utilizar un puñado de colores del sistema para temas de alto contraste. Al elegir los nombres de color del sistema, recuerde los siguientes consejos:

- **Elija los colores del sistema que tengan el mismo significado semántico** que el elemento que está coloreando. Por ejemplo, si elige un color de alto contraste para el texto dentro de una ventana, utilice `WindowText` y no `ControlText`.
- **Elija pares de primer plano/fondo** juntos o no estará seguro de que su elección de color funcionará en todos los temas de alto contraste.
- Determina qué partes de la interfaz de usuario son las más importantes y asegúrate de que las áreas de **contenido destaquen**. Perderá sin muchos detalles que las diferencias sutiles en el tono de color normalmente distinguirían, por lo que el uso de colores de borde fuertes es común para definir áreas de contenido, porque no hay variantes de color para diferentes áreas de contenido.

Conjunto de colores del sistema

La tabla en [WPF Team Blog: SystemColors Reference](#) indica el conjunto completo de nombres de color del sistema y los tonos correspondientes que se muestran en cada tema.

Al aplicar este conjunto limitado de colores a la interfaz de usuario, *se espera que pierda detalles sutiles que estaban presentes en los temas "normales"*. Este es un ejemplo de la interfaz de usuario con colores grises sutiles que se utilizan para distinguir áreas dentro de una ventana de herramientas. Cuando se empareja con la misma ventana que se muestra en el modo de contraste alto, puede ver que todos los fondos son del mismo tono y los bordes de esas áreas se indican solo por borde:

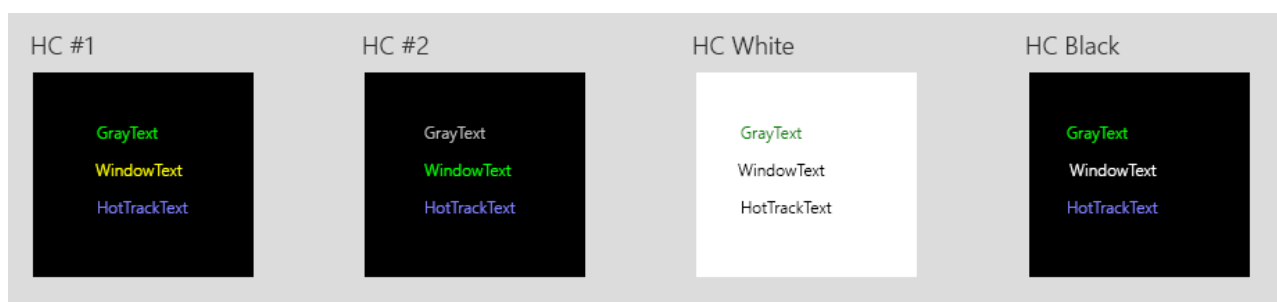


Ejemplo de cómo se pierden detalles sutiles en Contraste Alto

Elegir colores de texto en un editor

El texto coloreado se utiliza en un editor o en una superficie de diseño para indicar significado, como permitir una fácil identificación de grupos de elementos similares. En un tema de contraste alto, sin embargo, no tiene la capacidad de diferenciar entre más de tres colores de texto. WindowText, GrayText y HotTrackText son los únicos colores disponibles en las superficies WindowBackground. Puesto que no puede utilizar más de tres colores, elija cuidadosamente las diferencias más importantes que desea mostrar en el modo de contraste alto.

Tonos para cada uno de los nombres de token permitidos en una superficie del editor, tal como aparecen en cada tema de contraste alto:



Comparación de editor de contraste alto

Ejemplos de la superficie del editor en el tema Azul:

```

Form1.cs
MatchingGame.Form1
Form1()

/// </summary>
1 reference
private void AssignIconsToSquares()
{
    // The TableLayoutPanel has 16 labels,
    // and the icon list has 16 icons,
    // so an icon is pulled at random from the list
    // and added to each label
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(icons.Count);
            iconLabel.Text = icons[randomNumber];
        }
    }
}

```

Editor en el tema azul

```

Form1.cs
MatchingGame.Form1
Form1()

/// </summary>
1 reference
private void AssignIconsToSquares()
{
    // The TableLayoutPanel has 16 labels,
    // and the icon list has 16 icons,
    // so an icon is pulled at random from the list
    // and added to each label
    foreach (Control control in tableLayoutPanel1.Controls)
    {
        Label iconLabel = control as Label;
        if (iconLabel != null)
        {
            int randomNumber = random.Next(icons.Count);
            iconLabel.Text = icons[randomNumber];
        }
    }
}

```

Editor en tema #1 de alto contraste

Patrones de uso

Muchos elementos comunes de la interfaz de usuario ya tienen colores de contraste alto definidos. Puede hacer referencia a estos patrones de uso al elegir sus propios nombres de color del sistema, de modo que los elementos de la interfaz de usuario sean coherentes con componentes similares.

COLOR DEL SISTEMA	USO
ActiveCaption	- IDE activo y glifos de botón de ventana en balsa al pasar el ratón y pulse - Fondo de la barra de título para IDE y ventanas en balsa - Fondo de la barra de estado predeterminada
ActiveCaptionText	- IDE activo y ventanas en balsadas para el primer plano de la barra de título (texto y glifos) - Fondo y borde de los botones de la ventana activa al pasar el ratón y pulse
Control	- Cuadro combinado, lista desplegable, y el control de búsqueda por defecto y el fondo deshabilitado, incluyendo el botón desplegable - Fondo del botón de destino del muelle - Fondo de la barra de comandos - Fondo de la ventana de la herramienta

COLOR DEL SISTEMA	USO
ControlDark	- Antecedentes IDE - Separadores de menú y barra de comandos - Borde de la barra de comandos - Sombras de menú - Ventana de herramientas pestaña por defecto y pasar el ratón borde y separador - Documento bien desbordado botón de fondo - Borde del glifo objetivo del muelle
ControlDarkDark	- Desenfocado, ventana de la pestaña del documento seleccionado
ControlLight	- Ocultar automáticamente el borde de la pestaña - Caja combinada y borde de lista desplegable - Fondo y borde del objetivo del muelle
ControlLightLight	- Frontera provisional seleccionada y enfocada
ControlText	- Cuadro combinado y glifo de lista desplegable - Ventana de herramientas de texto de pestaña no seleccionado
GrayText	- Cuadro combinado y lista desplegable de borde deshabilitado, glifo desplegable, texto y texto del elemento de menú - Texto del menú deshabilitado - Control de búsqueda 'opciones de búsqueda' texto de encabezado - Separador de sección de control de búsqueda
Resaltar	- Todos los fondos y bordes flotantes y presionados, excepto el fondo del botón desplegable del cuadro combinado y el borde del botón de desbordamiento del documento bien - Fondos de elementos seleccionados
HighlightText	- Todos los primeros planos flotantes y presionados (texto y glifos) - Ventana de herramientas enfocada y control de la ventana de la pestaña del documento en primer plano - Borde de la barra de título de la ventana de herramientas enfocada - Enfocado, seleccionado en primer plano de la pestaña provisional - Documentar bien borde del botón de desbordamiento en el hover y pulse - Borde de icono seleccionado
HotTrack	- Desplazamiento de fondo del pulgar de la barra y borde en la prensa - Glifo de flecha de barra de desplazamiento al presionar
InactiveCaption	- IDE inactivo y glifos de botón de ventana en balsa al pasar el ratón - Fondo de la barra de título para IDE y ventanas en balsa - Fondo de control de búsqueda deshabilitado

COLOR DEL SISTEMA	USO
InactiveCaptionText	- IDE inactivo y ventanas en balsadas en primer plano de la barra de título (texto y glifos) - Botones de ventana inactivos fondo y borde al pasar el ratón - Desenfocado el fondo del botón de la ventana de la herramienta y el borde - Pretexto de control de búsqueda deshabilitado
Menú	- Fondo del menú desplegable - Fondo de la marca de verificación marcada marcada y desactivada
MenuText	- Borde del menú desplegable - Marcas de verificación - Menú glifos - Texto del menú desplegable - Borde de icono seleccionado
Barra de desplazamiento	- Barra de desplazamiento y fondo de flecha de la barra de desplazamiento, todos los estados
Periodo	- Ocultar automáticamente el fondo de la pestaña - Barra de menús y fondo de estante de comandos - Fondo de pestaña de la ventana del documento desenfocado o no seleccionado y borde del documento, tanto para las pestañas abiertas como para las provisionales - Fondo de la barra de título de la ventana de herramientas desenfocada - Fondo de la pestaña de la ventana de la herramienta, tanto seleccionado como no seleccionado
WindowFrame	- Borde IDE
WindowText	- Ocultar automáticamente el primer plano de la pestaña - Seleccionado en primer plano de la pestaña de la ventana de herramientas - Pestaña de la ventana del documento desenfocada y desenfocado o no seleccionado en primer plano de la pestaña provisional - Vista de árbol por defecto de primer plano y pasar el ratón sobre el glifo no seleccionado - Ventana de herramientas seleccionada borde de la pestaña - Fondo del pulgar de la barra de desplazamiento, borde y glifo

Exposición de colores para los usuarios finales

Información general

A veces querrá permitir que el usuario final personalice la interfaz de usuario, como cuando crea un editor de código o una superficie de diseño. La forma más común de hacerlo es mediante el cuadro de diálogo **** > Opciones**** de herramientas. A menos que tenga una interfaz de usuario altamente especializada que requiera controles especiales, la forma más fácil de presentar la personalización es a través de la página **Fuentes y colores** dentro de la sección **Entorno** del cuadro de diálogo. Para cada elemento que expone para la personalización, el usuario puede elegir cambiar el color de primer plano, el color de fondo o ambos.

Creación de un VSPackage para sus colores personalizables

Un VSPackage puede controlar las fuentes y colores a través de categorías personalizadas y mostrar elementos en el fuentes y colores página de propiedad. Al usar este mecanismo, VSPackages debe implementar el [IVsFontAndColorDefaultsProvider](#) interfaz y sus interfaces asociadas.

En principio, este mecanismo se puede utilizar para modificar todos los elementos de visualización existentes y las categorías que los contienen. Sin embargo, no se debe utilizar para modificar la categoría Editor de texto o sus elementos de visualización. Para obtener más información sobre la categoría Editor de texto, consulte Información general sobre [fuentes y colores](#).

Para implementar categorías personalizadas o mostrar elementos, un VSPackage debe:

- **Crear o identificar categorías en el registro.** La implementación del IDE de la página de propiedades **Fuentes y colores** usa esta información para consultar correctamente el servicio que admite una categoría determinada.
- **Crear o identificar grupos en el registro (opcional).** Puede ser útil definir un grupo, que representa la unión de dos o más categorías. Si se define un grupo, el IDE combina automáticamente subcategorías y distribuye elementos de visualización dentro del grupo.
- **Implementar compatibilidad con IDE.**
- **Controle los cambios de fuente y color.**

Para crear o identificar categorías

Construya un tipo especial de

```
[HKLM\SOFTWARE\Microsoft \Visual Studio\<Visual Studio version>\FontAndColors\<Category>] <Category>
```

entrada de registro de categoría en donde está el nombre no localizado de la categoría.

Rellene el registro con dos valores:

NOMBRE	TIPO	DATA	DESCRIPCIÓN
Category	REG_SZ	GUID	Guid creado para identificar la categoría
Paquete	REG_SZ	GUID	El GUID del servicio VSPackage que admite la categoría

El servicio especificado en el registro debe proporcionar una implementación de [IVsFontAndColorDefaults](#) para la categoría correspondiente.

Para crear o identificar grupos

Construya un tipo especial de

```
[HKLM\SOFTWARE\Microsoft \Visual Studio\<Visual Studio version>\FontAndColors\<group>] <group> entrada de
```

registro de categoría en donde está el nombre no localizado del grupo.

Rellene el registro con dos valores:

NOMBRE	TIPO	DATA	DESCRIPCIÓN
Category	REG_SZ	GUID	Guid creado para identificar la categoría
Paquete	REG_SZ	GUID	El GUID del servicio VSPackage que admite la categoría

El servicio especificado en el registro [IVsFontAndColorGroup](#) debe proporcionar una implementación del grupo correspondiente.

FileExtensionMapping

FindAndReplace

FontAndColors

All Text Tool Windows

AllDebuggerVarWnds

Architecture Diagrams

Autos

Name	Type	Data
(Default)	REG_SZ	(value not set)
Category	REG_SZ	{35B25E75-AB53-4C5D-80EA-6682EBB2BBBD}
NameID	REG_DWORD	0x00000326 (806)
Package	REG_SZ	{3B70A4AE-BB91-4ABE-A05C-C4DE07B9763E}

Implementación de `IVsFontAndColorGroup`

Para implementar la compatibilidad con IDE

Implementar [GetObject](#), que devuelve un [IVsFontAndColorDefaults](#) interfaz o una [IVsFontAndColorGroup](#) interfaz al IDE para cada categoría o grupo GUID proporcionado.

Para cada categoría que admite, un VSPackage implementa una instancia independiente de la [IVsFontAndColorDefaults](#) interfaz.

Los métodos implementados a través de [IVsFontAndColorDefaults](#) deben proporcionar el IDE con:

- Listas de elementos de visualización en la categoría
- Nombres localizables para elementos de visualización
- Mostrar información para cada miembro de la categoría

NOTE

Cada categoría debe contener al menos un elemento de visualización.

El IDE [IVsFontAndColorGroup](#) utiliza la interfaz para definir una unión de varias categorías.

Su implementación proporciona al IDE:

- Una lista de las categorías que componen un grupo determinado
- Acceso a instancias de [IVsFontAndColorDefaults](#) que admiten cada categoría dentro del grupo
- Nombres de grupos localizables

Actualización del IDE

El IDE almacena en caché información sobre la configuración de fuente y color. Por lo tanto, después de cualquier modificación de la configuración de fuente y color IDE, asegurarse de que la memoria caché está actualizada es una práctica recomendada.

La actualización de la memoria caché se realiza a través de la [ivsFontAndColorCacheManager](#) interfaz y se puede realizar globalmente o simplemente en los elementos seleccionados.

Manejo de cambios de fuente y color

Para admitir correctamente la coloración del texto que muestra un VSPackage, el servicio de coloración que admite el VSPackage debe responder a los cambios iniciados por el usuario realizados a través de la fuentes y colores página de propiedades.

Para ello, un VSPackage debe:

- **Controlar eventos generados** por IDE mediante la implementación de la [IVsFontAndColorEvents](#) interfaz. El IDE llama al método adecuado después de las modificaciones del usuario de la fuentes y colores página. Por ejemplo, llama al método [OnFontChanged](#) si se selecciona una nueva fuente.

- **sondear el IDE en buscade cambios** . Esto se puede hacer a través de la interfaz [IVsFontAndColorStorage](#) implementada por el sistema. Aunque principalmente para admitir la persistencia, el [GetItem](#) método puede obtener información de fuente y color para mostrar elementos. Para obtener más información sobre la configuración de fuentes y colores, consulte el artículo de MSDN Acceso a la configuración de [fuente almacenada y color](#).

NOTE

Para asegurarse de que los resultados de sondeo son correctos, utilice el [IVsFontAndColorCacheManager](#) interfaz para determinar si se necesita un vaciado de caché y actualización antes de llamar a los métodos de recuperación de la [IVsFontAndColorStorage](#) interfaz.

Registro de fuente personalizada y categoría de color sin implementar interfaces

En el ejemplo de código siguiente se muestra cómo registrar la fuente personalizada y la categoría de color sin implementar interfaces:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\FontAndColors\CSharp Tool Window]
"Package"="{F5E7E71D-1401-11D1-883B-0000F87579D2}"
"Category"="{9FF46859-A47E-47bf-8AC5-EC3DBE69D1FE}"
"ToolWindowPackage"="{7259e420-6241-4e0d-b535-5b820671d183}"

"NameID"=dword:00000064
```

Para este ejemplo de código:

- `"NameID"` • el identificador de recurso del nombre de categoría localizado en el paquete
- `"ToolWindowPackage"` • GUID del paquete
- `"Category"="{9FF46859-A47E-47bf-8AC5-EC3DBE69D1FE}"` es sólo un ejemplo y el valor real puede ser un nuevo GUID proporcionado por el implementador.

Establezca el GUID de la categoría de propiedades Fuente y Color

En el ejemplo de código siguiente se muestra cómo establecer GUID de categoría.

```
// m_pView is your IVsTextView
IVsTextEditorPropertyCategoryContainer spPropCatContainer =
    (IVsTextEditorPropertyCategoryContainer)m_pView;
if (spPropCatContainer != null)
{
    IVsTextEditorPropertyContainer spPropContainer;
    Guid GUID_EditPropCategory_View_MasterSettings =
        new Guid("{D1756E7C-B7FD-49a8-B48E-87B14A55655A}");
    hr = spPropCatContainer.GetPropertyCategory(
        ref GUID_EditPropCategory_View_MasterSettings,
        out spPropContainer);
    if(hr == 0)
    {
        hr =
            spPropContainer.SetProperty(
                VSEDITPROPID.VSEDITPROPID_ViewGeneral_FontCategory,
                catGUID);
        hr =
            spPropContainer.SetProperty(
                VSEDITPROPID.VSEDITPROPID_ViewGeneral_ColorCategory,
                catGUID);
    }
}
```

Colores compartidos para Visual Studio

06/04/2020 • 92 minutes to read • [Edit Online](#)

Al diseñar la interfaz de usuario que usa elementos de shell de Visual Studio comunes, o si desea que el elemento de interfaz sea coherente con características similares, use nombres de token existentes en los archivos de definición de paquete para elegir y asignar colores. Esto garantiza que la interfaz de usuario mantenga la coherencia con el entorno general de Visual Studio y que se actualice automáticamente cuando se agreguen o actualicen temas.

En este artículo se describen los elementos de interfaz de usuario comunes y los nombres de token que estos usan y a los que se puede hacer referencia al crear una interfaz de usuario similar. Para obtener información específica sobre cómo tener acceso a estos tokens de color, vea [The VSColor Service](#).

Asegúrese de usar correctamente los nombres de token:

- **Usar nombres de token basados en función, no en el color en sí.** Los colores comunes compartidos están asociados a elementos específicos de la interfaz y solo están destinados para características iguales o similares. Por ejemplo, no vuelva a usar el color de un cuadro combinado presionado para una animación de progreso de giro solo porque le gusta el color. Las funciones del cuadro combinado y la animación son diferentes, y si el color asociado con el cuadro combinado cambia, es posible que ya no sea un color adecuado para el elemento de animación. Un uso coherente del color ayuda a orientar a los usuarios y evitar confusiones.
- **Usar colores de fondo y de texto en la combinación correcta.** Los colores de fondo destinados para usarse con texto tendrán un color de texto asociado. No use colores de texto que no sean los que se especifican para el fondo. Si no hay un color de texto asociado, no utilice ese color de fondo para ninguna superficie en la que espere mostrar texto. Otras combinaciones de texto y colores de fondo pueden dar lugar a una interfaz ilegible.
- **Usar colores de control adecuados para su ubicación.** En algunos estados, algunos controles de Visual Studio no tienen colores de borde y fondo independientes. En su lugar, toman los colores de las superficies que están detrás de ellos. Procure usar siempre los nombres de token que sean adecuados para la ubicación donde coloca el control.

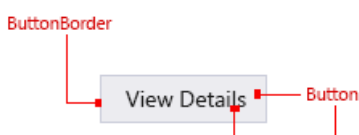
IMPORTANT

No uses tokens que se encuentran en las categorías "Página de inicio" o "Cider".

Controles comunes compartidos

Cuando use una barra de comandos estándar de Visual Studio en la característica, tendrá acceso a controles de shell con estilo. No debería volver a crear plantillas de estos controles comunes. Sin embargo, si necesita crear una barra de comandos personalizada, también podría ser necesario crear controles personalizados. En ese caso, asegúrese de usar los nombres de token correctos para cada uno de los siguientes controles de modo que la interfaz de usuario sea coherente con el resto de Visual Studio.

Controles de botón



USO...	NO UTILICE ...
... para los botones del documento bien que desea integrar con temas de Visual Studio (luz, oscuro, azul o un tema de alto contraste del sistema).	... para los botones que se mostrarán en un fondo personalizado que no forma parte de un tema de Visual Studio.

Botón: estado estándar



Botón estándar

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.Button</code>
Borde de botón	<code>CommonControls.ButtonBorder</code>

Botón: estado predeterminado



Botón predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.ButtonDefault</code>
Borde de botón	<code>CommonControls.ButtonBorderDefault</code>

Botón: estado desactivado



Botón desactivado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.ButtonDisabled</code>
Borde de botón	<code>CommonControls.ButtonBorderDisabled</code>

Botón: estado de desplazamiento



Botón al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.ButtonHover</code>
Borde de botón	<code>CommonControls.ButtonBorderHover</code>

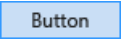
Botón: estado pulsado



Botón pulsado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.ButtonPressed</code>
Borde de botón	<code>CommonControls.ButtonBorderPressed</code>

Botón: estado enfocado



Botón enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Botón	<code>CommonControls.ButtonFocused</code>
Borde de botón	<code>CommonControls.ButtonBorderFocused</code>

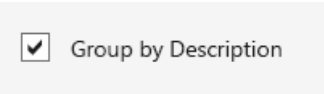
Controles de casilla



Casilla de verificación (línea roja)

USO...	NO UTILICE ...
... para los controles de casilla de verificación contenidos en el pozo del documento.	... para cualquier interfaz de usuario que no sea un control de casilla de verificación.

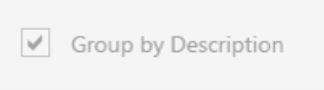
Casilla de verificación: estado predeterminado



Casilla de verificación predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.CheckBoxBackground</code>
Borde	<code>CommonControls.CheckBoxBorder</code>
Texto	<code>CommonControls.CheckBoxText</code>
Glifo	<code>CommonControls.CheckBoxGlyph</code>

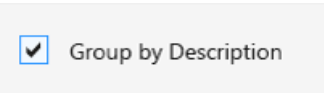
Casilla de verificación: estado deshabilitado



Casilla de verificación deshabilitada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.CheckBoxBackgroundDisabled</code>
Borde	<code>CommonControls.CheckBoxBorderDisabled</code>
Texto	<code>CommonControls.CheckBoxTextDisabled</code>
Glifo	<code>CommonControls.CheckBoxGlyphDisabled</code>

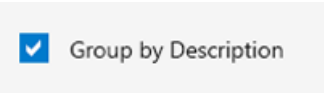
Casilla de verificación: estado de desplazamiento



Casilla de verificación al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.CheckBoxBackgroundHover</code>
Borde	<code>CommonControls.CheckBoxBorderHover</code>
Texto	<code>CommonControls.CheckBoxTextHover</code>
Glifo	<code>CommonControls.CheckBoxGlyphHover</code>

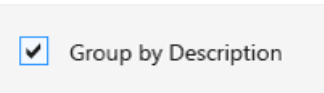
Casilla de verificación: estado pulsado



Casilla de verificación presionada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.CheckBoxBackgroundPressed</code>
Borde	<code>CommonControls.CheckBoxBorderPressed</code>
Texto	<code>CommonControls.CheckBoxTextPressed</code>
Glifo	<code>CommonControls.CheckBoxGlyphPressed</code>

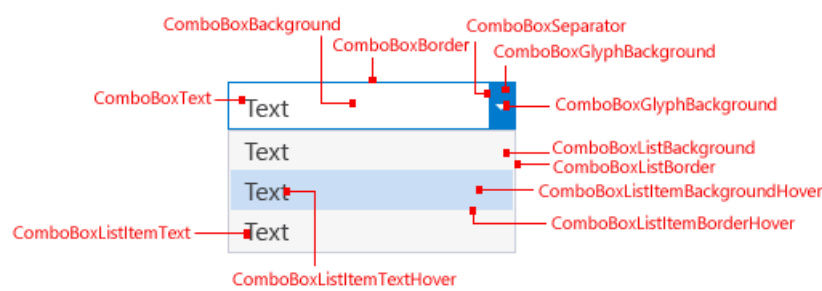
Casilla de verificación: estado enfocado



Casilla de verificación Enfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.CheckBoxBackgroundFocused</code>
Borde	<code>CommonControls.CheckBoxBorderFocused</code>
Texto	<code>CommonControls.CheckBoxTextFocused</code>
Glifo	<code>CommonControls.CheckBoxGlyphFocused</code>

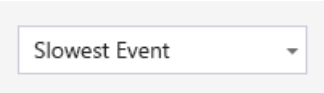
Desplegables y cajas combinadas



Cuadro desplegable/combo (línea roja)

USO...	NO UTILICE ...
... para menús desplegables y cuadros combinados en el documento bien.	... para cualquier interfaz de usuario que no sea un cuadro desplegable o combinado.
	... para menús desplegables de la barra de comandos o cuadros combinados . combo boxes

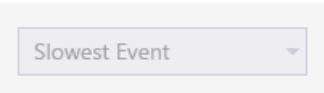
Desplegables y cuadros combinados: estado predeterminado



Cuadro desplegable/combo predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxBackground</code>
Borde	<code>CommonControls.ComboBoxBorder</code>
Texto	<code>CommonControls.ComboBoxText</code>
Separador	<code>CommonControls.ComboBoxSeparator</code>
Glifo	<code>CommonControls.ComboBoxGlyph</code>
Fondo de glifo	<code>CommonControls.ComboBoxGlyphBackground</code>

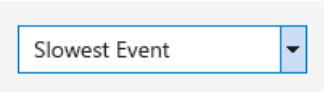
Desplegables y cuadros combinados: estado deshabilitado



Cuadro desplegable/combo deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxBackgroundDisabled</code>
Borde	<code>CommonControls.ComboBoxBorderDisabled</code>
Texto	<code>CommonControls.ComboBoxTextDisabled</code>
Separador	<code>CommonControls.ComboBoxSeparatorDisabled</code>
Glifo	<code>CommonControls.ComboBoxGlyphDisabled</code>
Fondo de glifo	<code>CommonControls.ComboBoxGlyphBackgroundDisabled</code>

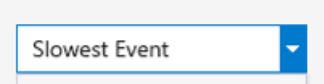
Desplegables y cuadros combinados: estado de desplazamiento



Desplegable o cuadro combinado al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxBackgroundHover</code>
Borde	<code>CommonControls.ComboBoxBorderHover</code>
Texto	<code>CommonControls.ComboBoxTextHover</code>
Separador	<code>CommonControls.ComboBoxSeparatorHover</code>
Glifo	<code>CommonControls.ComboBoxGlyphHover</code>
Fondo de glifo	<code>CommonControls.ComboBoxGlyphBackgroundHover</code>

Desplegables y cuadros combinados: estado presionado

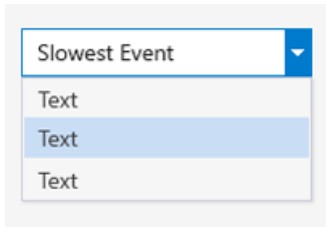


Cuadro desplegable/combo presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxBackgroundPressed</code>
Borde	<code>CommonControls.ComboBoxBorderPressed</code>
Texto	<code>CommonControls.ComboBoxTextPressed</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Separador	<code>CommonControls.ComboBoxSeparatorPressed</code>
Glifo	<code>CommonControls.ComboBoxGlyphPressed</code>
Fondo de glifo	<code>CommonControls.ComboBoxGlyphBackgroundPressed</code>

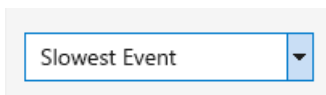
Vista de elementos de lista de cuadros desplegables y cuadros combinados: estado presionado



Desplegable/cuadro combinado presionado vista de elementos de lista

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxListBackground</code> <code>CommonControls.ComboBoxListBackgroundHover</code> <code>CommonControls.ComboBoxListItemBackgroundPressed</code> <code>CommonControls.ComboBoxListItemBackgroundFocused</code>
Borde	<code>CommonControls.ComboBoxListBorder</code> <code>CommonControls.ComboBoxListBorderHover</code> <code>CommonControls.ComboBoxListBorderPressed</code> <code>CommonControls.ComboBoxListBorderFocused</code>
Texto del elemento	<code>CommonControls.ComboBoxListItemText</code> <code>CommonControls.ComboBoxListItemTextHover</code> <code>CommonControls.ComboBoxListItemTextPressed</code> <code>CommonControls.ComboBoxListItemTextFocused</code>
Sombra de fondo	<code>CommonControls.ComboBoxListBackgroundShadow</code>

Desplegables y cuadros combinados: estado enfocado

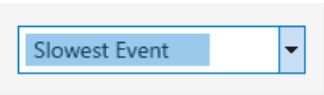


Drop-down/combo box con enfoque

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>CommonControls.ComboBoxBackgroundFocused</code>
Borde	<code>CommonControls.ComboBoxBorderFocused</code>
Texto	<code>CommonControls.ComboBoxTextFocused</code>
Separador	<code>CommonControls.ComboBoxSeparatorFocused</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Glifo	<code>CommonControls.ComboBoxGlyphFocused</code>
Fondo de glifo	<code>CommonControls.ComboBoxGlyphBackgroundFocused</code>

Desplegables y cuadros combinados: selección de entrada de texto

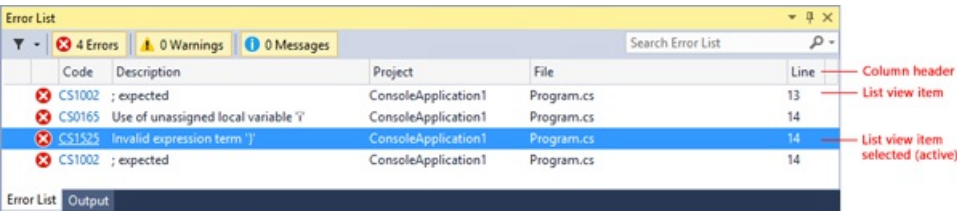


Selección de entrada de texto de cuadro desplegable/combo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Resaltar	<code>CommonControls.ComboBoxTextInputSelection</code>

Controles de datos tabulares (cuadrícula)

Los controles de datos tabulares, también conocidos como controles de cuadrícula, son controles comunes de Visual Studio que se pueden usar para presentar grandes cantidades de datos en varias columnas. Los controles estándar de datos tabulares están en varias ubicaciones de Visual Studio: la ventana de herramientas Lista de errores, los informes de IntelliTrace y la vista de montón de memoria, entre otros. Use siempre los controles de datos tabulares estándar que se proporcionan. En raras ocasiones, podría no tener acceso a los controles estándar de datos tabulares. En estos casos, use los siguientes nombres de token para asegurarse de que la interfaz de usuario sea coherente con otros controles de datos tabulares de Visual Studio.



Control de datos tabulares/cuadrícula (línea roja)

USO...	NO UTILICE ...
... para controles tabulares o de cuadrícula.	... para cualquier interfaz de usuario que no sea un control tabular o de cuadrícula.

Encabezados de columna

Los encabezados de columna constan de un fondo, un borde, el texto del título y un glifo opcional que suele usarse cuando se ordena una cuadrícula por esa columna.

Encabezado de columna: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Header.Default</code>
Primer plano (texto)	<code>Environment.CommandBarTextActive</code>
Primer plano (glifo)	<code>Header.Glyph</code>
Borde	<code>Header.SeparatorLine</code>

Encabezado de columna: estado de desplazamiento

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Header.MouseOver
Primer plano (texto)	Environment.CommandBarTextHover
Primer plano (glifo)	Header.MouseOverGlyph
Borde	Header.SeparatorLine

Encabezado de columna: estado presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	CommonControls.CheckBoxBackgroundPressed
Primer plano (texto)	CommonControls.CheckBoxBorderPressed
Primer plano (glifo)	CommonControls.CheckBoxTextPressed
Borde	CommonControls.CheckBoxGlyphPressed

Elementos de vista lista

Los elementos de vista de lista constan de un fondo y el contenido. El contenido puede ser texto, un icono o ambos.

Elementos de vista de lista: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Transparente
Primer plano (texto)	Environment.CommandBarTextActive
Borde	None

Elementos de vista de lista: estado activo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemActive
Primer plano (texto)	TreeView.SelectedItemActiveText
Borde	None

Elementos de vista de lista: estado inactivo

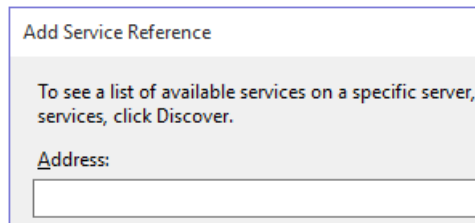
ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemInactive

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>TreeView.SelectedItemInactiveText</code>
Borde	None

Texto de la interfaz de usuario

Texto instructivo

El texto instructivo ofrece una explicación principal prominente de qué hacer en un cuadro de diálogo o página de documento.

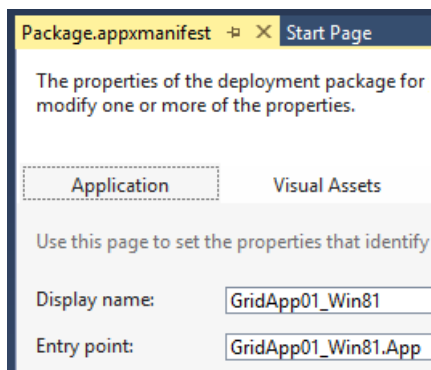


Texto instructivo predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.ControlText</code>

Texto instructivo secundario

En las páginas de documentos con mucho texto y controles, algunos textos instructivos utilizan un valor de color diferente. Esto ayuda a transmitir qué información es más importante y disminuye la densidad general de los elementos de la interfaz de usuario. (Véase también la sección siguiente sobre el texto de la sugerencia.)



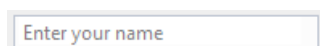
Texto instructivo secundario

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.ControlEditHintText</code>

Texto de sugerencia

El texto de sugerencia aparece en un control vacío, debajo de un control o en una superficie de documento vacía para mostrar al usuario qué hacer a continuación. Puede utilizar texto de sugerencia con fondos de ventana o control.

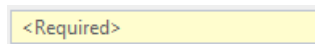
Texto de sugerencia predeterminado



Texto de sugerencia predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.ControlEditHintText</code>

Texto de sugerencia requerido

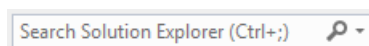


Texto de sugerencia requerido

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.ControlRequiredHintText</code>
Información previa	<code>Environment.ControlRequiredBackground</code>

Texto de control del cuadro de búsqueda

Consulte [cuadros de búsqueda](#) de otros tokens de color relacionados con el control de búsqueda.



Texto de control del cuadro de búsqueda

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>SearchControl.UnfocusedWatermarkText</code>

Hyperlink

El hipervínculo es un control que no tiene un par de primer plano/fondo. En todos los casos, utilice el color del hipervínculo de primer plano, que aparecerá correctamente en fondos oscuros, grises y blancos. Si no utiliza el token de color para el control de hipervínculo, verá el color predeterminado del sistema para "presionado", que parpadeará en rojo. Esa es la señal de que el control no está utilizando el token de color de entorno correcto.

[Add New Data Source...](#) Text

Hipervínculo (línea roja)

USO...	NO UTILICE ...
... cuando necesite crear un hipervínculo personalizado.	... para cualquier cosa que no sea un hipervínculo.

Hipervínculo: estado predeterminado

[Add New Data Source...](#)

Hipervínculo predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.PanelHyperlink</code>

Hipervínculo: estado de desplazamiento

[Add New Data Source...](#)

Hipervínculo al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.PanelHyperlinkHover</code>

Hipervínculo: estado presionado

[Add New Data Source...](#)

Hipervínculo presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.PanelHyperlinkPressed</code>

Hipervínculo: estado deshabilitado

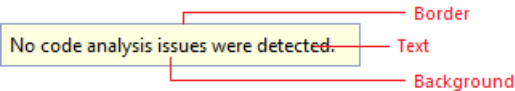
[Add New Data Source...](#)

Hipervínculo deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.PanelHyperlinkDisabled</code>

Barras de información

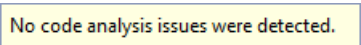
Las barras de información se usan para proporcionar más información sobre un determinado contexto y siempre se muestran en la parte superior de una ventana de documento o una ventana de herramientas.



Barra de información (línea roja)

USO...	NO UTILICE ...
... al crear barras de información personalizadas.	... para elementos de interfaz de usuario que no son similares a una barra de información.

Barra de información: estado predeterminado



Barra de información predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.InfoBarBackground</code>
Primer plano (texto)	<code>InfoBar.InfoBar</code>
Borde	<code>InfoBar.InfoBarBorder</code>

Botón Cerrar×() de la barra de información: estado predeterminado



Barra de información×predeterminada Cerrar () botón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.CloseButton</code>
Borde	<code>InfoBar.CloseButtonBorder</code>
Glifo	<code>InfoBar.CloseButtonGlyph</code>

Botón Cerrar×() de la barra de información: estado de desplazamiento



Botón Cerrar×() de la barra de información al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.CloseButtonHover</code>
Borde	<code>InfoBar.CloseButtonHoverBorder</code>
Glifo	<code>InfoBar.CloseButtonHoverGlyph</code>

Botón Cerrar×() de la barra de información: estado pulsado



Botón Cerrar×() de la barra de información pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.CloseButtonDown</code>
Borde	<code>InfoBar.CloseButtonDownBorder</code>
Glifo	<code>InfoBar.CloseButtonDownGlyph</code>

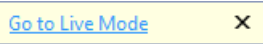
Botón de hipervínculo de la barra de información: estado predeterminado



Botón de hipervínculo de la barra de información predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>InfoBar.Hyperlink</code>

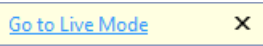
Botón de hipervínculo de la barra de información: estado de desplazamiento



Botón de hipervínculo de la barra de información al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>InfoBar.HyperlinkMouseOver</code> (Con subrayado)

Botón de hipervínculo de la barra de información: estado pulsado



Botón de hipervínculo de la barra de información pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Infobar.HyperlinkMouseDown</code> (Con subrayado)

Hipervínculo en línea de la barra de información (dentro de una oración): estado predeterminado



Botón de hipervínculo de la barra de infoens en línea predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>InfoBar.Hyperlink</code>

Hipervínculo en línea de la barra de información (dentro de una oración): estado de desplazamiento



Botón de hipervínculo en línea de la barra de información al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Infobar.HyperlinkMouseOver</code> (Con subrayado)

Hiperenlace en línea de la barra de información (dentro de una oración): estado presionado



Botón de hipervínculo en línea de la barra de información pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Infobar.HyperlinkMouseDown</code> (Con subrayado)

Botón de la barra de información: estado predeterminado



Botón de barra de información predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.Button</code>
Primer plano (texto)	<code>InfoBar.Button</code>
Borde	<code>InfoBar.ButtonBorder</code>

Botón de la barra de información: estado de desplazamiento



Botón de la barra de información al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.ButtonMouseOver</code>
Primer plano (texto)	<code>InfoBar.ButtonMouseOver</code>
Borde	<code>InfoBar.ButtonMouseOverBorder</code>

Botón de la barra de información: estado pulsado



Botón de barra de información pulsado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.ButtonMouseDown</code>
Primer plano (texto)	<code>InfoBar.ButtonMouseDown</code>
Borde	<code>InfoBar.ButtonMouseDownBorder</code>

Botón de la barra de información: estado deshabilitado



Botón de barra de información desactivado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.ButtonDisabled</code>
Primer plano (texto)	<code>InfoBar.ButtonDisabled</code>
Borde	<code>InfoBar.ButtonDisabledBorder</code>

Botón de la barra de información: estado enfocado

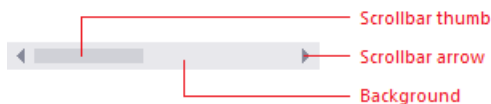


Botón de barra de información enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>InfoBar.ButtonFocus</code>
Primer plano (texto)	<code>InfoBar.ButtonFocus</code>
Borde	<code>InfoBar.ButtonFocusBorder</code>

Barras de desplazamiento

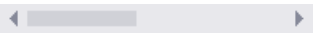
Las barras de desplazamiento tienen el estilo del entorno de Visual Studio y no es necesario que se themeen. Sin embargo, puede decidir que desea aprovechar los colores utilizados en las barras de desplazamiento para que la interfaz de usuario siempre parezca coherente con esta parte del entorno de Visual Studio.



Barra de desplazamiento (línea roja)

USO...	NO UTILICE ...
... al crear la interfaz de usuario que desea que coincida con las barras de desplazamiento de Visual Studio.	... para cualquier cosa que no desee que siempre coincida con la interfaz de usuario de la barra de desplazamiento.

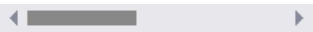
Barra de desplazamiento: estado predeterminado



Barra de desplazamiento predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Barra de desplazamiento	<code>Environment.ScrollBarBackground</code>
Primer plano (control)	<code>Environment.ScrollBarThumbBackground</code>

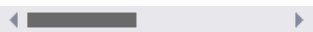
Barra de desplazamiento: estado de desplazamiento



Barra de desplazamiento al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Barra de desplazamiento	<code>Environment.ScrollBarBackground</code>
Primer plano (control)	<code>Environment.ScrollBarThumbMouseOverBackground</code>

Barra de desplazamiento: estado pulsado*



Barra de desplazamiento pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Barra de desplazamiento	<code>Environment.ScrollBarBackground</code>
Primer plano (control)	<code>Environment.ScrollBarThumbPressedBackground</code>

Flecha de barra de desplazamiento: estado predeterminado



Flecha de barra de desplazamiento predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ScrollBarArrowBackground</code> (Establecer en el mismo color que la barra de desplazamiento.)
Primer plano (glifo)	<code>Environment.ScrollBarArrowGlyph</code>

Flecha de la barra de desplazamiento: estado de desplazamiento



Flecha de barra de desplazamiento al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ScrollBarArrowMouseOverBackground</code> (Establecer en el mismo color que la barra de desplazamiento.)
Primer plano (glifo)	<code>Environment.ScrollBarArrowGlyphMouseOver</code>

Flecha de la barra de desplazamiento: estado pulsado



Flecha de barra de desplazamiento pulsada

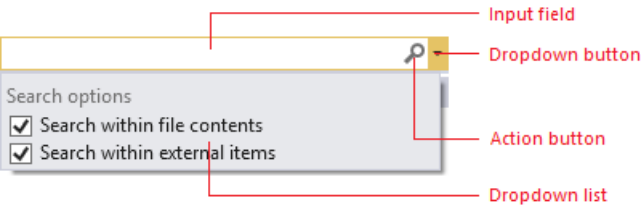
ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ScrollBarArrowPressedBackground</code> (Establecer en el mismo color que la barra de desplazamiento.)
Primer plano (glifo)	<code>Environment.ScrollBarArrowGlyphPressed</code>

Cuadros de búsqueda

Siempre que sea posible, use el control de búsqueda común proporcionado por el entorno de Visual Studio. Los colores del cuadro de búsqueda se encuentran en la categoría "SearchControl" del archivo `ShellColors.pkgdef` , que contiene los nombres de token para el campo de entrada, el botón de acción, el botón desplegable y el menú desplegable.

Un cuadro de búsqueda puede tener uno de varios estados, algunos de los cuales son mutuamente excluyentes:

- "Con foco" o "sin foco" se refiere a si el cursor está en el cuadro de texto.
- "Activo" o "inactivo" se refiere a si el usuario ha especificado una consulta de búsqueda en el cuadro de texto.
- "Desplazar el puntero" significa que el usuario ha colocado el puntero sobre el cuadro de búsqueda con el mouse (este estado invalida todos los demás estados).
- "Deshabilitado" significa que la función de búsqueda se ha desactivado para el contexto actual.



Cuadro de búsqueda (línea roja)

USO...	NO UTILICE ...
... cuando está diseñando un cuadro de búsqueda personalizado.	... para cualquier cosa que no sea un cuadro de búsqueda.

USO...	NO UTILICE ...
	... para cualquier cosa que no quieras que siempre coincida con la interfaz de usuario del cuadro de búsqueda.

Campo de entrada de búsqueda focalizado



Campo de entrada de búsqueda focalizado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.FocusedBackground</code>
Primer plano (texto)	<code>SearchControl.FocusedBackground</code>
Borde	<code>SearchControl.FocusedBorder</code>
Separador	<code>SearchControl.FocusedDropDownSeparator</code>

Campo de entrada de búsqueda activo y desenfocado



Campo de entrada de búsqueda activo y desenfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.SearchActiveBackground</code>
Primer plano (texto)	<code>SearchControl.SearchActiveBackground</code>
Borde	<code>SearchControl.UnfocusedBorder</code>
Separador	<code>SearchControl.DropDownSeparator</code>

Campo de entrada de búsqueda desenfocado e inactivo



Campo de entrada de búsqueda desenfocado e inactivo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.Unfocused</code>
Primer plano (texto)	<code>SearchControl.Unfocused</code>
Borde	<code>SearchControl.UnfocusedBorder</code>
Separador	<code>SearchControl.DropDownSeparator</code>

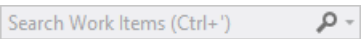
Campo de entrada de búsqueda resaltado (solo texto)



Campo de entrada de búsqueda resaltado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.Selection</code>
Primer plano (texto)	<code>SearchControl.FocusedBackground</code>
Borde	None
Separador	<code>SearchControl.FocusedDropDownSeparator</code>

Campo de entrada de búsqueda deshabilitado



Campo de entrada de búsqueda deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.Disabled</code>
Primer plano (texto)	<code>SearchControl.Disabled</code>
Borde	<code>SearchControl.DisabledBorder</code>
Separador	<code>SearchControl.DropDownSeparator</code>

Botón de acción de búsqueda focalizada



Botón de acción de búsqueda focalizada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (glifo de búsqueda)	<code>SearchControl.SearchGlyph</code>
Primer plano (glifo de detención)	<code>SearchControl.StopGlyph</code>
Primer plano (glifo de borrado)	<code>SearchControl.ClearGlyph</code>
Borde	N/D

Botón de acción de búsqueda desenfocada



Botón de acción de búsqueda desenfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (glifo de búsqueda)	<code>SearchControl.SearchGlyph</code>
Primer plano (glifo de detención)	<code>SearchControl.StopGlyph</code>
Primer plano (glifo de borrado)	<code>SearchControl.ClearGlyph</code>
Borde	N/D

Botón de acción de búsqueda presionado



Botón de acción de búsqueda presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.ActionButtonMouseDown</code>
Primer plano (glifo)	<code>SearchControl.ActionButtonMouseDownGlyph</code>
Borde	<code>SearchControl.ActionButtonMouseDownBorder</code>

Botón de acción de búsqueda deshabilitado



Botón de acción de búsqueda deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (glifo)	<code>SearchControl.ActionButtonDisabledGlyph</code>
Borde	None

Botón desplegable de búsqueda focalizada



Botón desplegable de búsqueda focalizada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.FocusedDropDownButton</code>
Primer plano (glifo)	<code>SearchControl.FocusedDropDownButtonGlyph</code>
Borde	<code>SearchControl.FocusedDropDownButtonBorder</code>

Botón desplegable búsqueda desenfocada



Botón desplegable búsqueda desenfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.UnfocusedDropDownButton</code>
Primer plano (glifo)	<code>SearchControl.UnfocusedDropDownButtonGlyph</code>
Borde	<code>SearchControl.UnfocusedDropDownButtonBorder</code>

Botón desplegable de búsqueda presionado



Botón desplegable de búsqueda presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.MouseDownDropDownButton</code>
Primer plano (glifo)	<code>SearchControl.MouseDownDropDownButtonGlyph</code>
Borde	<code>SearchControl.MouseDownDropDownButtonBorder</code>

Botón desplegable de búsqueda deshabilitado

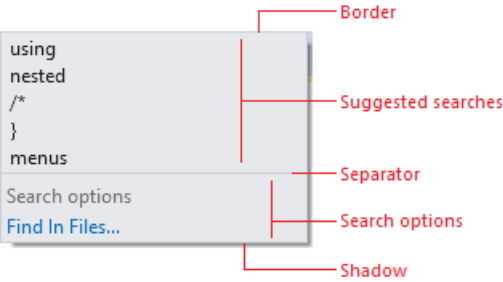


Botón desplegable de búsqueda deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (glifo)	<code>SearchControl.DisabledDownButtonGlyph</code>
Borde	None

Listas desplegables de búsqueda

El menú desplegable del cuadro de búsqueda tiene el potencial de ser un poco más complejo que otros menús desplegables en Visual Studio. Las secciones "búsquedas sugeridas" y "opciones de búsqueda" pueden aparecer solas o juntas en el menú, y cada una de ellas se colorea por separado. Además, estas secciones están separadas por una línea cuando aparecen juntas y un borde rodea todo el menú desplegable.



Lista desplegable de búsqueda (línea roja)

USO...	NO UTILICE ...
... cuando está creando una lista desplegable de búsqueda personalizada.	... para listas desplegables que aparecen en otros contextos.

USO...	NO UTILICE ...
... los nombres de token correctos para los componentes de lista correctos.	... en cualquier combinación de fondo/primer plano que no sea especificada.

Elementos de la lista desplegable de búsqueda

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Borde	<code>SearchControl.PopupBorder</code>
Separador	<code>SearchControl.PopupSectionHeaderSeparator</code>
Shadow	<code>Environment.DropShadowBackground</code>

Búsquedas sugeridas: estado predeterminado

```
nested
/*
}
```

Búsquedas sugeridas por defecto

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.PopupItemsListBackgroundGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>SearchControl.PopupItemText</code>

Búsquedas sugeridas: estado de desplazamiento

```
nested
/*
}
```

Búsquedas sugeridas al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.PopupControlMouseOverBackgroundGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>SearchControl.PopupMouseOverItemText</code>
Borde	<code>SearchControl.PopupControlMouseOverBorder</code>

Opciones de búsqueda: estado predeterminado

Search options

☒ Search within file contents

☒ Search within external items

Opciones de búsqueda predeterminadas (casilla de verificación)

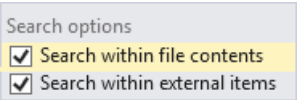
Search options

[Find In Files...](#)

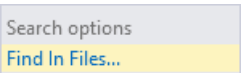
Opciones de búsqueda predeterminadas (enlace)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.PopupSectionBackgroundGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto de casilla)	<code>SearchControl.PopupCheckboxText</code>
Primer plano (texto de vínculo)	<code>SearchControl.PopupButtonText</code>
Fondo de encabezado	<code>SearchControl.PopupSectionHeaderGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto de encabezado)	<code>SearchControl.PopupSectionHeaderText</code>

Opciones de búsqueda: estado de desplazamiento



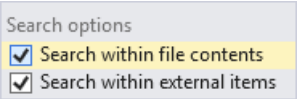
Opciones de búsqueda (casilla de verificación) al mantener el mouse



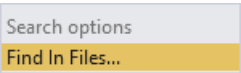
Opciones de búsqueda (enlace) al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>SearchControl.PopupControlMouseOverBackgroundGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto de casilla)	<code>SearchControl.PopupCheckboxMouseDownText</code>
Primer plano (texto de vínculo)	<code>SearchControl.PopupButtonMouseDownText</code>
Borde	<code>SearchControl.PopupControlMouseOverBorder</code>

Opciones de búsqueda: estado presionado



Opciones de búsqueda pulsadas (casilla de verificación)



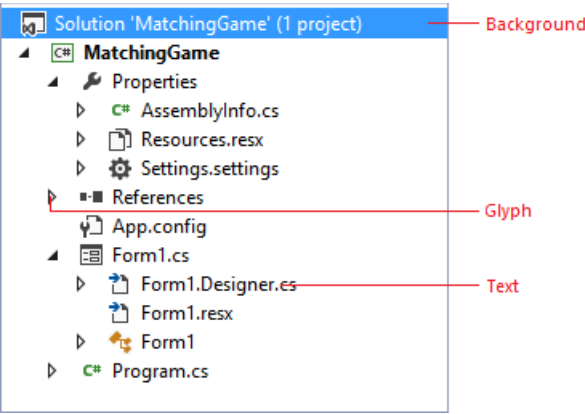
Opciones de búsqueda prensadas (enlace)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo de casilla	SearchControl.PopupControlMouseDownBackgroundGradientBegin SearchControl.PopupControlMouseDownBackgroundGradientEnd (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto de casilla)	SearchControl.PopupCheckboxMouseDownText
Fondo de vínculo	SearchControl.PopupButtonMouseDownBackgroundGradientBegin (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto de vínculo)	SearchControl.PopupButtonMouseDownText

Vistas a los árboles

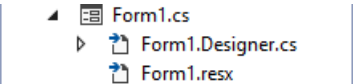
Varias ventanas de herramientas, como el Explorador de soluciones, el Explorador de servidores y `TreeView` la vista de clases, implementan un esquema organizativo jerárquico cuyos colores se controlan mediante nombres de color en la categoría. Todos los elementos de una vista de árbol tienen colores de fondo y de texto. Los elementos que tienen elementos secundarios anidados también tienen glifos que indican si el elemento está expandido o contraído.



Vista de árbol (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar que necesite implementar una vista organizativa jerárquica.	... para cualquier cosa que no sea similar a una vista de árbol.
	... en cualquier combinación de fondo/primer plano que no sea especificada.

Elemento de vista de árbol: estado predeterminado



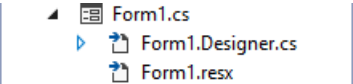
Elemento de vista de árbol predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.Background
Primer plano (texto)	TreeView.Background

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR

Primer plano (glifo)	TreeView.Glyph
Borde	None

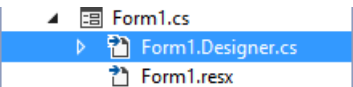
Elemento de vista de árbol: estado de desplazamiento



Elemento de vista de árbol al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.Background
Primer plano (texto)	TreeView.Background
Primer plano (glifo)	TreeView.GlyphMouseOver
Borde	None

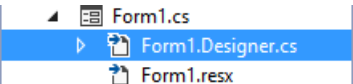
Elemento de vista de árbol: arrastre sobre el estado



Elemento de vista de árbol al arrastrar sobre

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.DragOverItem
Primer plano (texto)	TreeView.DragOverItem
Primer plano (glifo)	TreeView.DragOverItemGlyph
Borde	None

Elemento de vista de árbol: seleccionado, estado enfocado

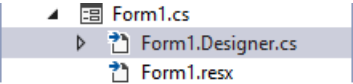


Elemento de vista de árbol seleccionado y enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemActive
Primer plano (texto)	TreeView.SelectedItemActive

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (glifo)	TreeView.SelectedItemActiveGlyph
Borde	TreeView.FocusVisualBorder

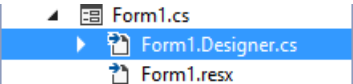
Elemento de vista de árbol: seleccionado, estado desenfocado



Elemento de vista de árbol seleccionado y desenfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemInactive
Primer plano (texto)	TreeView.SelectedItemInactive
Primer plano (glifo)	TreeView.SelectedItemInactiveGlyph
Borde	None

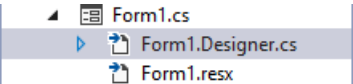
Elemento de vista de árbol: estado de desplazamiento, selección y enfoque



Elemento de vista de árbol seleccionado y enfocado al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemActive
Primer plano (texto)	TreeView.SelectedItemActive
Primer plano (glifo)	TreeView.SelectedItemActiveGlyphMouseOver
Borde	TreeView.FocusVisualBorder

Elemento de vista de árbol: estado de desplazamiento, selección y desenfocado



Elemento de vista de árbol seleccionado y desenfocado al mantener el mouse

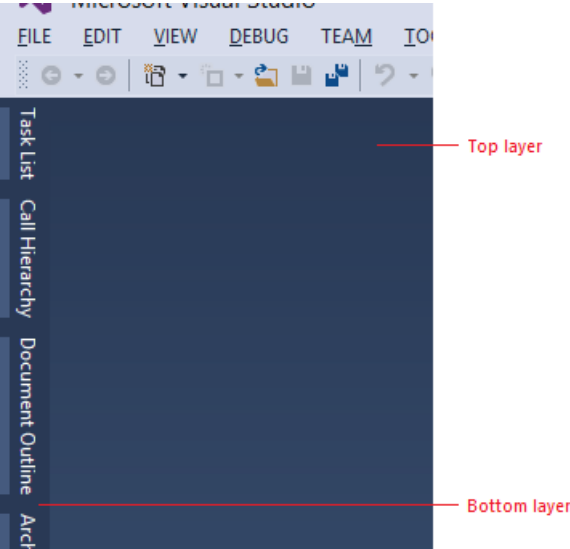
ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemInactive
Primer plano (texto)	TreeView.SelectedItemInactive
Primer plano (glifo)	TreeView.SelectedItemActiveGlyphMouseOver

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Borde	None

Apariencia de Shell

Información previa

El fondo del entorno consta de dos niveles. El nivel inferior es un color sólido que cubre todo el IDE. El nivel superior se encuentra debajo del área de comandos y entre los canales de ocultación automática de la ventana de herramientas en los extremos izquierdo y derecho del IDE. Las capas de fondo superior e inferior se establecen en el mismo color en los temas Luz y Oscuridad.



Fondo del shell de Visual Studio (línea roja)

USO...	NO UTILICE ...
... para los lugares donde desea que coincida con el fondo del entorno de Visual Studio.	... como relleno para lugares que no son superficies de fondo.
	... como fondo para colocar elementos de primer plano.

Apariencia del vaciado de la capa inferior

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.EnvironmentBackground</code>

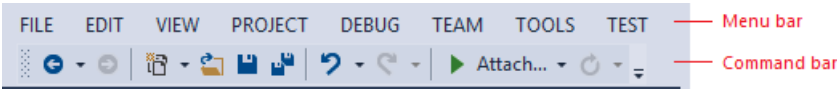
Apariencia del vaciado de capa superior

Los delimitadores de degradado se establecen en el mismo valor de color en los temas claro y oscuro de Visual Studio 2013.

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.EnvironmentBackgroundGradientBegin</code> <code>Environment.EnvironmentBackgroundGradientEnd</code> <code>Environment.EnvironmentBackgroundGradientMiddle1</code> <code>Environment.EnvironmentBackgroundGradientMiddle2</code>

Área de comandos

Para los fondos del área de comandos se usan dos conjuntos de nombres de token: un conjunto para la ubicación donde se coloca la barra de menús y uno para la ubicación donde se colocan las barras de comandos. Un grupo individual de la barra de comandos tiene sus propios valores de color de fondo, que se describen con más detalle en la sección "Barra de comandos". El texto de la barra de menús y la barra de comandos se describe en las secciones de barra de menús y barra de comandos, respectivamente.



Estante de comandos de Visual Studio (línea roja)

USO...	NO UTILICE ...
... para las áreas en las que se colocan menús o barras de herramientas.	... para áreas que no son similares a un estante de comandos.
... con la combinación correcta de nombre de token de fondo/primer plano.	

Barra de menús del estante de comandos

Los delimitadores de degradado se establecen en el mismo valor de color en los temas claro y oscuro de Visual Studio 2013.

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandShelfHighlightGradientBegin</code> <code>Environment.CommandShelfHighlightGradientMiddle</code> <code>Environment.CommandShelfHighlightGradientEnd</code>

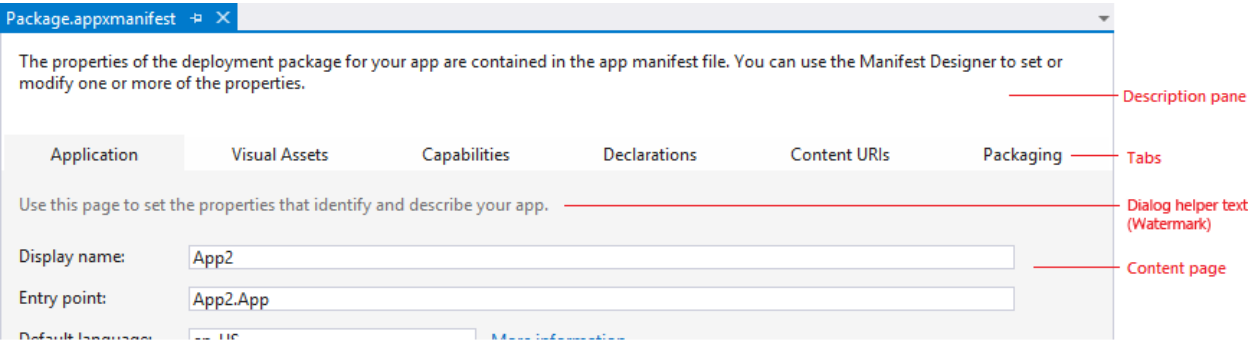
Barra de comandos del estante de comandos

Los delimitadores de degradado se establecen en el mismo valor de color en los temas claro y oscuro de Visual Studio 2013.

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandShelfBackgroundGradientBegin</code> <code>Environment.CommandShelfBackgroundGradientMiddle</code> <code>Environment.CommandShelfBackgroundGradientEnd</code>

Diseñador de manifiestos

El Diseñador de manifiestos se creó para que resulte más fácil editar el archivo de manifiesto en los proyectos de Windows 8 y Windows Phone 8. Aunque no hay ningún marco compartido disponible para su consumo, es conveniente que haga coincidir el diseño y los colores de las pestañas de navegación u orientación con la estructura general. Para obtener más información sobre el diseño, vea [Layout for Visual Studio](#).



Diseñador de manifiestos (línea roja)

USO...	NO UTILICE ...
... para diseñadores que son similares al Diseñador de manifiestos.	... si tienes más de seis pestañas.
... en lugar de usar controles de pestaña comunes en la parte superior de un editor dentro del documento.	... para cualquier interfaz de usuario que no esté estructurada como el Diseñador de manifiestos.

Ficha seleccionada del Diseñador de manifiestos: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	ManifestDesigner.TabActive
Borde	None

Panel de descripción seleccionado del Diseñador de manifiestos: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	ManifestDesigner.DescriptionPane

Página de contenido seleccionado del Diseñador de manifiestos: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	ManifestDesigner.Background
Texto del asistente del cuadro de diálogo	ManifestDesigner.WatermarkText (Este nombre de token no coincide con su función.)

Ficha Diseñador de manifiestos: estado no seleccionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	ManifestDesigner.Tab.Inactive

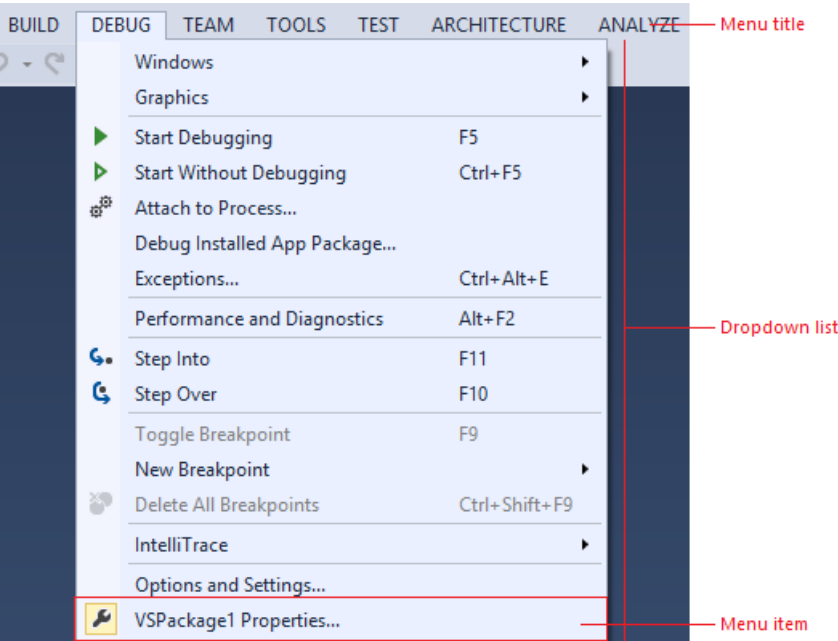
Ficha Diseñador de manifiestos: estado de desplazamiento

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	ManifestDesigner.Tab.Mouseover

Estructuras de comandos

Menús

Los menús pueden producirse en varios lugares de Visual Studio: la barra de menús principal, incrustada en las ventanas de documentos o herramientas, o al hacer clic con el botón derecho en varias ubicaciones del IDE. Las implementaciones de menús asociados con otros elementos de la interfaz de usuario se describen en la sección del elemento respectivo. Se debe usar siempre la implementación de menú estándar proporcionada por el entorno de Visual Studio. Sin embargo, en algunas ocasiones podría no tener acceso a los menús estándar de Visual Studio. En estos casos, use los siguientes nombres de token para asegurarse de que la interfaz de usuario sea coherente con otros menús de Visual Studio.

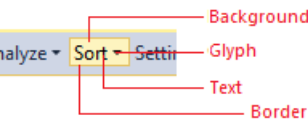


Menú Visual Studio (línea roja)

USO...	NO UTILICE ...
... cuando necesite crear un menú personalizado.	... el color de fondo solo. Use siempre la combinación de primer plano y fondo tal como se especifica.
... cuando tenga un nuevo componente de interfaz de usuario que desea que coincida con los menús de Visual Studio.	

Títulos del menú

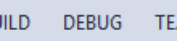
Los títulos de menú constan de un fondo, un borde y el texto del título, así como un glifo opcional, que suele usarse cuando el menú se encuentra en una barra de comandos.



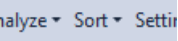
Título del menú (línea roja)

USO...	NO UTILICE ...
... cada vez que esté creando un título de menú personalizado.	... para cualquier cosa que no quieras que coincida siempre con el título del menú.
	... en cualquier combinación de fondo/primer plano que no sea especificada.

Título del menú: estado predeterminado



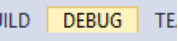
Título del menú predeterminado



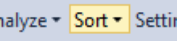
Título del menú predeterminado con glifo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (texto)	Environment.CommandBarTextActive
Primer plano (glifo)	Environment.CommandBarMenuGlyph
Borde	None

Título del menú: estado de desplazamiento



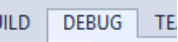
Título de menú al mantener el mouse



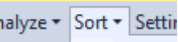
Título de menú con glifo al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Environment.CommandBarMouseOverBackgroundBegin (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	Environment.CommandBarTextHover
Primer plano (glifo)	Environment.CommandBarMenuMouseOverGlyph
Borde	Environment.CommandBarBorder

Título del menú: estado pulsado



Título del menú presionado

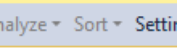


Título del menú presionado con glifo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Environment.CommandBarMenuBackgroundGradientBegin (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.CommandBarTextActive</code>
Primer plano (glifo)	<code>Environment.CommandBarMenuMouseDownGlyph</code>
Borde	<code>Environment.CommandBarMenuBorder</code> (Solo lados izquierdo, superior y derecho.)

Título del menú: estado deshabilitado

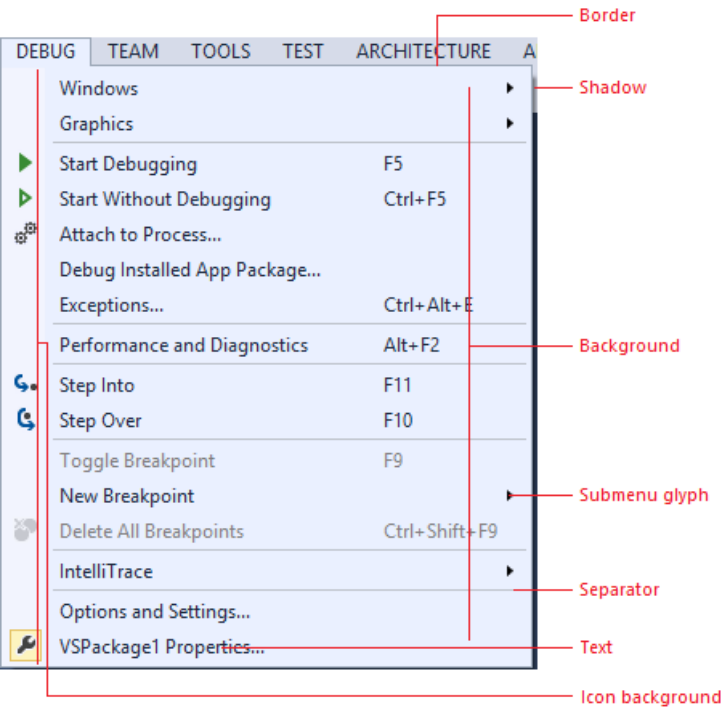


Título del menú deshabilitado con glifo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (texto)	<code>Environment.CommandBarTextInactive</code>
Primer plano (glifo)	<code>Environment.CommandBarTextInactive</code>
Borde	None

Elementos de menú

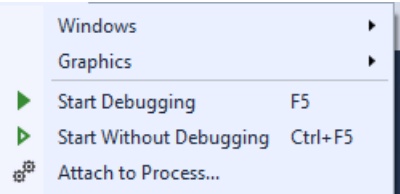
Un elemento de menú individual se compone del texto del menú y un icono opcional, una casilla o un glifo de submenú. Su color de fondo y de texto cambian al mantener el puntero. Este token de color es un par de primer plano y fondo.



USO...	NO UTILICE ...
... para cualquier lista desplegable que se inicie desde una barra de menús o una barra de comandos.	... para cualquier lista desplegable en otro contexto.

USO...	NO UTILICE ...
	... en cualquier combinación de fondo/primer plano que no sea especificada.

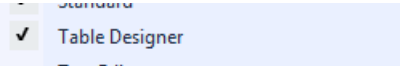
Elementos del menú: estado predeterminado



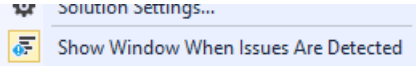
Elementos de menú predeterminados

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarMenuBackgroundGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.CommandBarTextActive</code>
Primer plano (glifo de submenú)	<code>Environment.CommandBarMenuSubmenuGlyph</code>
Borde	<code>Environment.CommandBarMenuBorder</code>
Fondo de canal de iconos	<code>Environment.CommandBarMenuIconBackground</code>
Separador	<code>Environment.CommandBarMenuSeparator</code>
Shadow	<code>Environment.DropShadowBackground</code>

Elementos del menú: estados marcados y seleccionados



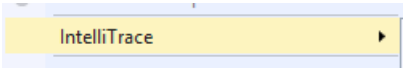
Elemento de menú activado



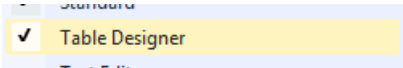
Elemento de menú seleccionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Marca de verificación	<code>Environment.CommandBarCheckBox</code>
Fondo de marca de verificación	<code>Environment.CommandBarSelectedIcon</code>
Fondo de icono	<code>Environment.CommandBarSelected</code>
Borde de icono	<code>Environment.CommandBarSelectedBorder</code>

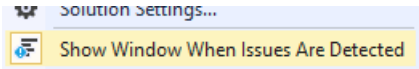
Elementos del menú: estado de desplazamiento



Elemento del menú al pasar el ratón



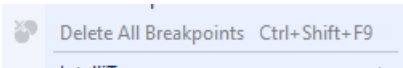
Elemento de menú activado al pasar el ratón



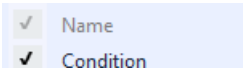
Elemento de menú seleccionado al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarItemMouseOver</code>
Primer plano (texto)	<code>Environment.CommandBarItemMouseOverText</code>
Primer plano (glifo de submenú)	<code>Environment.CommandBarMenuMouseOverSubMenuGlyph</code>
Marca de verificación	<code>Environment.CommandBarCheckBoxMouseOver</code>
Fondo de marca de verificación	<code>Environment.CommandBarHoverOverSelectedIcon</code>
Fondo de icono	<code>Environment.CommandBarHoverOverSelected</code>
Borde de icono	<code>Environment.CommandBarHoverOverSelectedIconBorder</code>

Elementos del menú: estado deshabilitado



Elemento de menú deshabilitado



Elemento de menú deshabilitado con marca de verificación

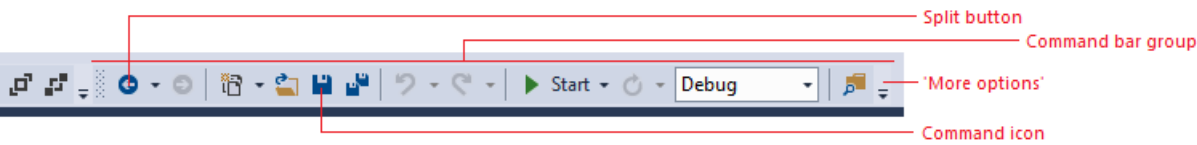
ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.CommandBarTextInactive</code>
Primer plano (glifo de submenú)	<code>Environment.CommandBarMenuSubMenuGlyph</code>
Marca de verificación	<code>Environment.CommandBarCheckBoxDisabled</code>
Fondo de marca de verificación	<code>Environment.CommandBarSelectedIconDisabled</code>

Barras de comandos

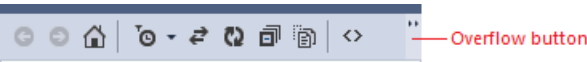
Una barra de comandos puede aparecer en varios lugares dentro del IDE de Visual Studio, especialmente el estante de comandos e incrustado en ventanas de herramientas o documentos.

En general, use siempre la implementación de menús estándar proporcionada por el entorno de Visual Studio. El uso del mecanismo estándar, garantiza que todos los detalles visuales se mostrarán correctamente y que los elementos interactivos se comportarán de forma coherente con otros controles de barra de comandos de Visual

Studio. Sin embargo, si necesita crear su propia barra de comandos, asegúrese de diseñarla correctamente mediante los siguientes nombres de token.



Barra de comandos (línea roja)

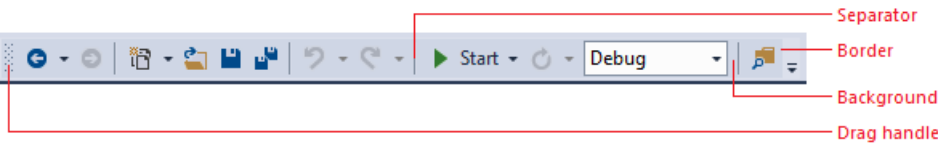


Botón de desbordamiento (línea roja)

USO...	NO UTILICE ...
... en lugares donde necesita una barra de comandos incrustada, pero no puede usar la implementación estándar de la barra de comandos de Visual Studio.	... para los elementos de la interfaz de usuario que no son similares a una barra de comandos.
	... para componentes de barra de comandos distintos de los para los que se especifican nombres de token.

Grupos de barras de comandos

Un grupo de la barra de comandos se compone de un conjunto relacionado de controles de barra de comandos y puede incluir cualquier número de botones, botones de expansión, menús desplegables, cuadros combinados o menús. Los colores de dichos controles se regulan mediante nombres de token independientes y se describen individualmente en esta guía. Se usa una línea divisoria para dividir un grupo de la barra de comandos en subgrupos relacionados.



Grupo de barras de comandos (línea roja)

USO...	NO UTILICE ...
... en lugares donde necesita una barra de comandos incrustada, pero no puede usar la implementación estándar de la barra de comandos de Visual Studio.	... para los elementos de la interfaz de usuario que no son similares a una barra de comandos.
	... para componentes de barra de comandos distintos de los para los que se especifican nombres de token.

Grupo de barras de comandos: estado predeterminado

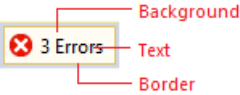
ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Borde	<code>Environment.CommandBarToolBarBorder</code>
Controlador de arrastre	<code>Environment.CommandBarDragHandle</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Separador	Environment.CommandBarToolBarSeparator Environment.CommandBarToolBarSeparatorHighlight

Iconos de comando



Icono de comando (línea roja)



Icono de comando con texto (línea roja)

USO...	NO UTILICE ...
... para cualquier botón que se colocará en una barra de comandos.	... para los controles que tienen sus propios nombres de token.
	... en cualquier combinación de fondo/primer plano que no sea especificada.

Icono de comando: estado predeterminado



Icono de comando predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D (se hereda del fondo de la barra de comandos)
Primer plano (texto)	Environment.CommandBarTextActive
Borde	N/D

Icono de comando: estado predeterminado, seleccionado



Por defecto, icono de comando seleccionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Environment.CommandBarSelected
Primer plano (texto)	Environment.CommandBarTextSelected
Borde	Environment.CommandBarSelectedBorder

Icono de comando: estados de desplazamiento o enfoque



Icono de comando al pasar el ratón o enfocen

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.CommandBarTextHover</code>
Borde	<code>Environment.CommandBarBorder</code>

Icono de comando: estados de desplazamiento o enfoque, seleccionados



Icono de comando seleccionado al mantener el mouse o el enfoque

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarHoverOverSelected</code>
Primer plano (texto)	<code>Environment.CommandBarTextHoverOverSelected</code>
Borde	<code>Environment.CommandBarHoverOverSelectedIconBorder</code>

Icono de comando: estado presionado



Icono de comando presionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarMouseDownBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.CommandBarTextMouseDown</code>
Borde	<code>Environment.CommandBarBorder</code>

Icono de comando: estado deshabilitado



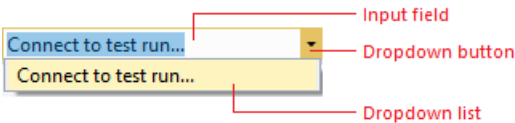
Icono de comando deshabilitado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D (se hereda del fondo de la barra de comandos)
Primer plano (texto)	<code>Environment.CommandBarTextInactive</code>
Borde	N/D

Cuadros combinados de la barra de comandos

IMPORTANT

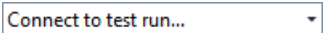
Los cuadros combinados son similares a las listas desplegables, pero incluyen un área de texto editable. Si el menú desplegable no incluye una región de texto editable, utilice los tokens de color para los **menús desplegables** de la barra de comandos.



Cuadro combinado barra de comandos (línea roja)

USO...	NO UTILICE ...
... al crear cuadros combinados personalizados.	... para cualquier cosa que no desee que coincida siempre con la interfaz de usuario de la barra de comandos.
... al crear un control de barra de comandos que es similar a un cuadro combinado.	... cuando tiene sin tener acceso a un cuadro combinado con estilo.

Campo de entrada del cuadro combinado de la barra de comandos: estado predeterminado



Campo de entrada del cuadro combinado de la barra de comandos

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxText</code>
Borde	<code>Environment.ComboBoxBorder</code>
Separador	Sin separador

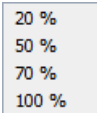
Botón desplegable de la barra de comandos: estado predeterminado



Botón desplegable de la barra de comandos

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D (se hereda del fondo de la barra de comandos)
Primer plano (glifo)	<code>Environment.ComboBoxGlyph</code>

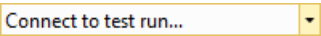
Lista desplegable de la barra de comandos: estado predeterminado



Lista desplegable de la barra de comandos

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxPopupBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.ComboBoxItemText</code>
Borde	<code>Environment.ComboBoxPopupBorder</code>

Campo de entrada del cuadro combinado de la barra de comandos: estado de desplazamiento



Campo de entrada del cuadro combinado de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.ComboBoxMouseOverText</code>
Borde	<code>Environment.ComboBoxMouseOverBorder</code>
Separador	<code>Environment.ComboBoxMouseOverSeparator</code>

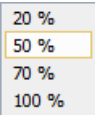
Botón desplegable de la barra de comandos: estado de desplazamiento



Botón desplegable de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxButtonMouseOverBackground</code>
Primer plano (glifo)	<code>Environment.ComboBoxMouseOverGlyph</code>

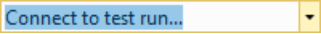
Lista desplegable de la barra de comandos: estado de desplazamiento



Lista desplegable de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo (elemento de menú)	<code>Environment.ComboBoxItemMouseOverBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxItemMouseOverText</code>
Borde (elemento de menú)	<code>Environment.ComboBoxItemMouseOverBorder</code>

Campo de entrada del cuadro combinado de la barra de comandos: estado enfocado



Campo de entrada de cuadro combinado de barra de comandos enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxFocusedBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxFocusedText</code>
Borde	<code>Environment.ComboBoxFocusedBorder</code>
Separador	<code>Environment.ComboBoxFocusedButtonSeparator</code>

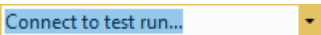
Botón desplegable de la barra de comandos: estado enfocado



Botón desplegable de la barra de comandos enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxFocusedButtonBackground</code>
Primer plano (glifo)	<code>Environment.ComboBoxFocusedGlyph</code>

Campo de entrada del cuadro combinado de la barra de comandos: estado presionado



Campo de entrada del cuadro combinado de la barra de comandos presionada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxMouseDownBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxMouseDownText</code>
Borde	<code>Environment.ComboBoxMouseDownBorder</code>
Separador	<code>Environment.ComboBoxMouseDownSeparator</code>

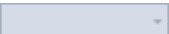
Botón desplegable de la barra de comandos: estado pulsado



Botón desplegable de la barra de comandos pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxButtonMouseDownBackground</code>
Primer plano (glifo)	<code>Environment.ComboBoxMouseDownGlyph</code>

Campo de entrada del cuadro combinado de la barra de comandos: estado deshabilitado



Campo de entrada del cuadro combinado de la barra de comandos desactivada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ComboBoxDisabledBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxDisabledText</code>
Borde	<code>Environment.ComboBoxDisabledBorder</code>
Separador	Sin separador

Botón desplegable de la barra de comandos: estado deshabilitado



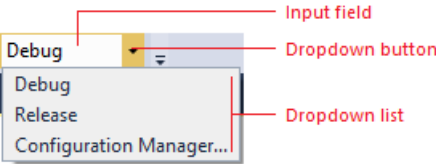
Botón desplegable de la barra de comandos desactivada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (glifo)	<code>Environment.ComboBoxDisabledGlyph</code>

Menús desplegables de la barra de comandos

IMPORTANT

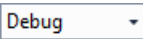
Las listas desplegables son similares a los cuadros combinados, pero carecen de áreas de texto editable. Si el menú desplegable incluye una región de texto editable, utilice los tokens de color para los **cuadros combinados** de la barra de comandos.



Lista desplegable de la barra de comandos (línea roja)

USO...	NO UTILICE ...
... cuando se crean controles de lista desplegable personalizados.	... para cualquier cosa que no sea similar a una lista desplegable.
	... para cuadros combinados o botones divididos.

Campo de selección desplegable de la barra de comandos: estado predeterminado



Campo de selección desplegable de la barra de comandos predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownBackground</code>
Primer plano (texto)	<code>DropDownText</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Borde	<code>DropDownBorder</code>
Separador	Sin separador

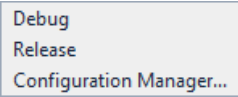
Botón desplegable de la barra de comandos: estado predeterminado



Botón desplegable de la barra de comandos predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (glifo)	<code>Environment.DropDownGlyph</code>

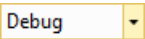
Lista desplegable de la barra de comandos: estado predeterminado



Lista desplegable de la barra de comandos predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownPopupBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.ComboBoxItemText</code>
Borde	<code>Environment.DropDownPopupBorder</code>
Shadow	<code>Environment.DropShadowBackground</code>

Campo de selección desplegable de la barra de comandos: estado de desplazamiento



Campo de selección desplegable de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.DropDownMouseOverText</code>
Borde	<code>Environment.DropDownMouseOverBorder</code>
Separador	<code>Environment.DropDownButtonMouseOverSeparator</code>

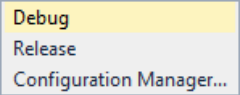
Botón desplegable de la barra de comandos: estado de desplazamiento



Botón desplegable de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownButtonMouseOverBackground</code>
Primer plano (glifo)	<code>Environment.DropDownMouseOverGlyph</code>

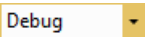
Lista desplegable de la barra de comandos: estado de desplazamiento



Lista desplegable de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo (elemento de menú)	<code>Environment.ComboBoxItemMouseOverBackground</code>
Primer plano (texto)	<code>Environment.ComboBoxItemMouseOverText</code>
Borde (elemento de menú)	<code>Environment.ComboBoxItemMouseOverBorder</code>

Campo de selección desplegable de la barra de comandos: estado pulsado



Campo de selección desplegable de la barra de comandos pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownMouseDownBackground</code>
Primer plano (texto)	<code>Environment.DropDownMouseDownText</code>
Borde	<code>Environment.DropDownMouseDownBorder</code>
Separador	<code>Environment.DropDownButtonMouseDownSeparator</code>

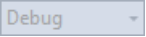
Botón desplegable de la barra de comandos: estado pulsado



Botón desplegable de la barra de comandos pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownButtonMouseDownBackground</code>
Primer plano (glifo)	<code>Environment.DropDownMouseDownGlyph</code>

Campo de selección desplegable de la barra de comandos: estado deshabilitado



Campo de selección desplegable de la barra de comandos desactivada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DropDownDisabledBackground</code>
Primer plano (texto)	<code>Environment.DropDownDisabledText</code>
Borde	<code>Environment.DropDownDisabledBorder</code>
Separador	Sin separador

Botón desplegable de la barra de comandos: estado deshabilitado

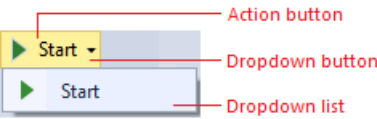


Botón desplegable de la barra de comandos desactivada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (glifo)	<code>Environment.DropDownDisabledGlyph</code>

Botones divididos en la barra de comandos

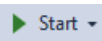
Los botones de expansión comparten muchos nombres de token con otros controles de barra de comandos como, por ejemplo, botones, menús y texto de la barra de comandos. Todos los nombres de token de acciones y botones desplegables necesarios se repiten aquí para su comodidad. Las listas desplegables de botones divididos son implementaciones de menús de la barra de [comandos](#).



Botón de división de la barra de comandos (línea roja)

USO...	NO UTILICE ...
... cuando se crea un botón de división personalizado.	... para otros tipos de botones.
	... en cualquier combinación de fondo/primer plano que no sea especificada.

Botón de división de la barra de comandos: estado predeterminado

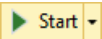


Botón de división de la barra de comandos por defecto

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	None
Primer plano (texto)	<code>Environment.CommandBarTextActive</code>
Primer plano (glifo)	<code>Environment.CommandBarSplitButtonGlyph</code>
Borde	N/D

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Separador	N/D

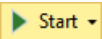
Botón de división de la barra de comandos: estado de desplazamiento



Botón de división de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.CommandBarTextHover</code>
Primer plano (glifo)	<code>Environment.CommandBarSplitButtonMouseOverGlyph</code>
Borde	<code>Environment.CommandBarBorder</code>
Separador	<code>Environment.CommandBarSplitButtonSeparator</code>

Botón de división de la barra de comandos: estado pulsado



Botón de división de la barra de comandos pulsada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarMouseDownBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.CommandBarTextMouseDown</code>
Primer plano (glifo)	<code>Environment.CommandBarSplitButtonMouseDownGlyph</code>
Borde	<code>Environment.CommandBarBorder</code>
Separador	N/D

Botón de división de la barra de comandos: estado deshabilitado



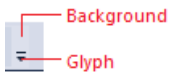
Botón de división de la barra de comandos desactivada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (texto)	<code>Environment.ComboBoxItemTextInactive</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (glifo)	<code>Environment.CommandBarTextInactive</code>
Borde	N/D
Separador	N/D

Barra de comandos 'Más opciones' y 'Desbordamiento' botones

El botón "Más opciones" se usa cuando un grupo de la barra de comandos se puede personalizar al agregar o quitar botones relacionados de barra de comandos. El botón "Desbordamiento" aparece cuando una barra de comandos se trunca debido a la falta de espacio horizontal y, al hacer clic en él, muestra un menú que contiene los botones de la barra de comandos que no pueden mostrarse. Los colores para estos dos botones están controlados por el mismo conjunto de nombres de token.



Barra de comandos 'Más opciones' botón (línea roja)

USO...	NO UTILICE ...
... para los botones personalizados "Más opciones" o "Desbordamiento".	... para botones que no tienen una funcionalidad similar a un botón "Más opciones" o "Desbordamiento".

Botones de la barra de comandos 'Más opciones' y 'Desbordamiento': estado predeterminado



Botón de la barra de comandos predeterminada 'Más opciones'



Botón 'Desbordamiento' de la barra de comandos predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarOptionsBackground</code>
Primer plano (glifo)	<code>Environment.CommandBarOptionsGlyph</code>

Botones de la barra de comandos 'Más opciones' y 'Desbordamiento': estado de desplazamiento



Barra de comandos 'Más opciones' al pasar el ratón



Botón 'Desbordamiento' de la barra de comandos al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarOptionsMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (glifo)	<code>Environment.CommandBarOptionsMouseDownGlyph</code>

Botones de la barra de comandos 'Más opciones' y 'Desbordamiento': estado pulsado



Botón de la barra de comandos pulsada 'Más opciones'



Botón de la barra de comandos pulsada 'Desbordamiento'

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.CommandBarOptionsMouseDownBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (glifo)	<code>Environment.CommandBarOptionsMouseDownGlyph</code>

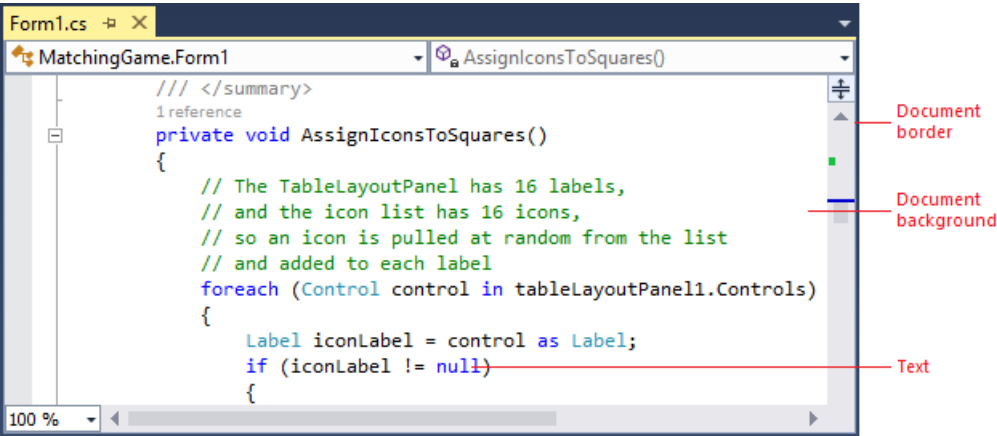
Ventanas de documento

No es necesario replicar las ventanas de documento, porque las proporciona el entorno de Visual Studio. Sin embargo, quizá prefiera aprovechar los colores que se usan en las ventanas de documento para que la interfaz de usuario se muestre siempre coherente con esta parte del entorno de Visual Studio.

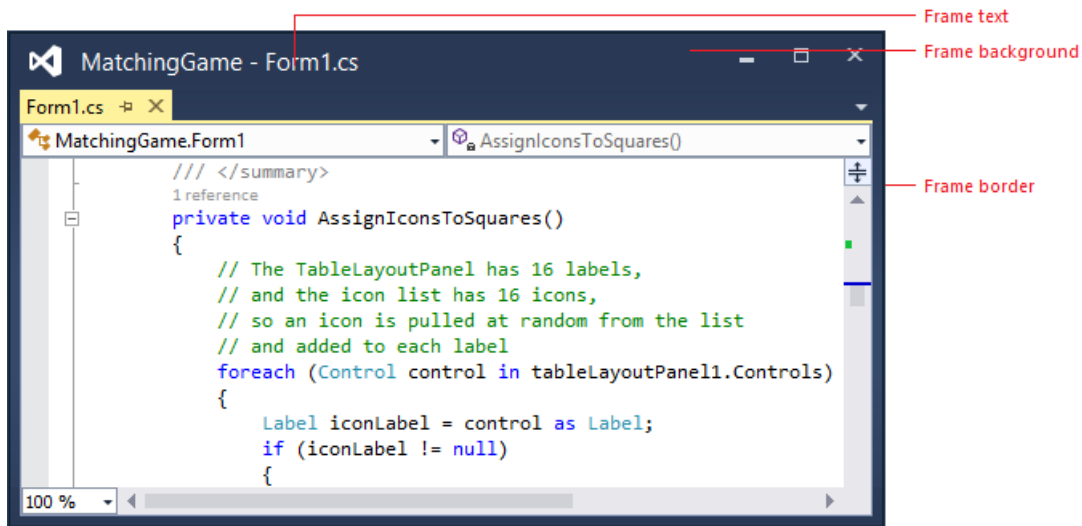
Cuando utilice tokens de color de ventana de documento, tenga cuidado de usarlos solo para elementos similares y siempre en pares. Si no lo hace, es posible que obtenga resultados inesperados en la interfaz de usuario.

Marcos de ventanas de documentos

Las ventanas de documento pueden estar acopladas en el IDE o flotantes como una ventana independiente. Cuando una ventana de documento está flotando fuera del IDE, todavía se encuentra en un documento bien y tiene colores de fondo, borde, texto y tabulación que son los mismos que cuando forma parte del IDE. Sin embargo, el documento se encuentra dentro de un marco que tiene su propio fondo, borde y colores de texto. Cuando las ventanas de herramientas se acoplan en el cuadro del documento, heredan el comportamiento y el color de los nombres de token de ventana de documento para sus pestañas.



Ventana de documento acoplado (línea roja)



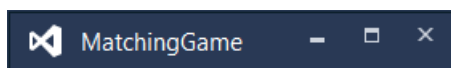
Ventana de documento flotante (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con la ventana del documento.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

Ventana de documento acoplada o flotante: estado predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Depende del tipo de documento
Primer plano (texto)	Depende del tipo de documento
Borde	Environment.ToolWindowBorder

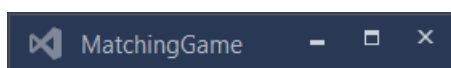
Marco de ventana de documento flotante y enfocado: estado predeterminado



Marco de ventana de documento flotante y enfocado por defecto

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Environment.ToolWindowFloatingFrame
Primer plano (texto)	Environment.ToolWindowFloatingFrame
Primer plano (glifo)	Environment.RaftedWindowButtonActiveGlyph
Borde	Environment.MainWindowActiveDefaultBorder
Borde (glifo)	Environment.RaftedWindowButtonActiveBorder (Establecer en transparente)

Marco de ventana de documento flotante y desenfocado: estado predeterminado



Marco de ventana de documento flotante y desenfocado por defecto

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowFloatingFrameInactive</code>
Primer plano (texto)	<code>Environment.ToolWindowFloatingFrameInactive</code>
Primer plano (glifo)	<code>Environment.RaftedWindowButtonInactiveGlyph</code>
Borde	<code>Environment.MainWindowInactiveBorder</code>
Borde (glifo)	<code>Environment.RaftedWindowButtonInactiveBorder</code> (Establecer en transparente)

Marco de ventana de documento flotante y enfocado: estado de desplazamiento



Marco de ventana de documento flotante y enfocado al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo (glifo)	<code>Environment.RaftedWindowButtonHoverActive</code>
Primer plano (glifo)	<code>Environment.RaftedWindowButtonHoverActiveGlyph</code>
Borde (glifo)	<code>Environment.RaftedWindowButtonHoverActiveBorder</code>

Marco de ventana de documento flotante y desenfocado: estado de desplazamiento



Marco de ventana de documento flotante y desenfocado al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo (glifo)	<code>EnvironmentRaftedWindowButtonHoverInactive</code>
Primer plano (glifo)	<code>Environment.RaftedWindowButtonHoverInactiveGlyph</code>
Borde (glifo)	<code>Environment.RaftedWindowButtonHoverInactiveBorder</code>

Marco de ventana de documento flotante y enfocado: estado pulsado



Marco de ventana de documento flotante y enfocado en la prensa

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Fondo (glifo)	<code>Environment.RaftedWindowButtonDown</code>
Primer plano (glifo)	<code>Environment.RaftedWindowButtonDownGlyph</code>

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Borde (glifo)	<code>Environment.RaftedWindowButtonDownBorder</code>

Pestañas de documentos

Las pestañas de documentos se colocan en el canal de pestañas para indicar los documentos que están abiertos actualmente, así como el documento actual activo o seleccionado. Las ventanas de herramientas también se pueden acoplar en el canal de pestañas de documentos si el usuario las coloca allí. En este caso, usan los mismos colores de pestaña que las ventanas de documento. Si va a crear interfaz de usuario que desea que siempre coincida con los colores de las ventanas de documento (lo que incluye las actualizaciones de temas o si se instalan nuevos temas), entonces haga referencia a estos tokens de color.



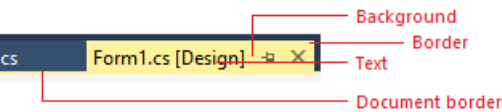
Fichas Documento (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las pestañas del documento y recoger automáticamente las actualizaciones del tema o los nuevos colores del tema.	... para cualquier interfaz de usuario que no desee cambiar automáticamente cuando el shell tiene una actualización del tema.

Pestañas de documentos abiertos

Cada documento abierto tiene una pestaña en el canal de pestañas de documentos que muestra su nombre. Los documentos pueden estar seleccionados o abiertos en segundo plano y sus pestañas reflejan los siguientes estados:

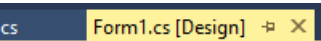
- La pestaña seleccionada representa el documento que se muestra actualmente en el cuadro de documento. Una pestaña seleccionada tiene un borde de documento que se extiende por todo el borde superior del cuadro de documento.
- Las pestañas de fondo son las fichas de documento que no son la pestaña seleccionada actualmente. Una vez hecho clic, se convierten en la pestaña seleccionada y adquieren todos los colores de fondo, borde y texto de esos nombres de token.



Ficha Abrir documento (línea roja)

USO...	NO UTILICE ...
... cuando crea pestañas de documentos personalizadas.	... para las pestañas provisionales (versión preliminar).
	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

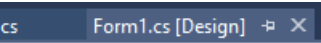
Pestaña Documento seleccionado y enfocado



Pestaña Documento seleccionado y enfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabSelectedGradientTop</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.FileTabSelectedText</code>
Borde	<code>Environment.FileTabSelectedBorder</code> (Establecer en el mismo color que el fondo.)
Borde de documento	<code>Environment.FileTabDocumentBorderBackground</code>

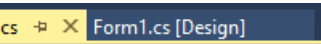
Pestaña Documento seleccionado y desenfocado



Pestaña Documento seleccionado y desenfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabInactiveGradientTop</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.FileTabInactiveText</code>
Borde	<code>Environment.FileTabInactiveBorder</code> (Establecer en el mismo color que el fondo.)
Borde de documento	<code>Environment.FileTabInactiveDocumentBorderBackground</code>

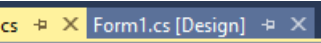
Ficha Documento de fondo: estado predeterminado



Ficha Documento de fondo predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabBackground</code>
Primer plano (texto)	<code>Environment.FileTabText</code>
Borde	<code>Environment.FileTabBorder</code> (Establecer en el mismo color que el fondo.)

Ficha Documento de fondo: estado de desplazamiento



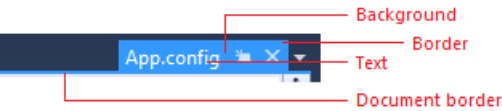
Ficha Documento de fondo al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabHotGradientTop</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.FileTabHotText</code>
Borde	<code>Environment.FileTabHotBorder</code> (Establecer en el mismo color que el fondo.)

Pestaña de vista previa

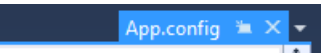
También se denomina pestaña "provisional". La pestaña de vista previa aparece en el lado derecho del canal de ficha del documento cuando el usuario hace clic en un elemento de la ventana de herramientas del Explorador de soluciones. Funciona como una vista previa del documento y también proporciona al usuario la posibilidad de mantener el documento abierto en la parte izquierda del canal de pestañas de documentos. Solo se puede abrir una pestaña de vista previa a la vez. Las pestañas de vista previa tienen tanto el estado seleccionado como en segundo plano al igual que las pestañas abiertas y, además, pueden estar con y sin foco en su estado activo.



Ficha Vista previa (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando una vista previa provisional y desee que algún elemento coincida con el color actual de la pestaña de vista previa.	... para cualquier tipo de documento o pestaña que no sea provisional (versión preliminar).
	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

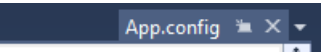
Pestaña de vista previa seleccionada y enfocada



Pestaña de vista previa seleccionada y enfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabProvisionalSelectedActive</code>
Primer plano (texto)	<code>Environment.FileTabProvisionalSelectedActiveForeground</code>
Borde	<code>Environment.FileTabProvisionalSelectedActiveBorder</code> (Establecer en el mismo color que el fondo.)
Borde de documento	<code>Environment.FileTabProvisionalSelectedActiveBorder</code>

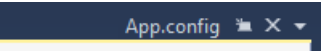
Pestaña de vista previa seleccionada y desenfocada



Pestaña de vista previa seleccionada y desenfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabProvisionalSelectedInactive</code>
Primer plano (texto)	<code>Environment.FileTabProvisionalSelectedInactiveForeground</code>
Borde	<code>Environment.FileTabProvisionalSelectedInactiveBorder</code>
Borde de documento	<code>Environment.FileTabProvisionalSelectedInactiveBorder</code>

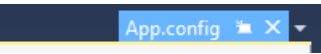
Pestaña de vista previa de fondo: estado predeterminado



Pestaña de vista previa de fondo predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabProvisionalInactive</code>
Primer plano (texto)	<code>Environment.FileTabProvisionalInactiveForeground</code>
Borde	<code>Environment.FileTabProvisionalInactiveBorder</code> (Establecer en el mismo color que el fondo.)

Pestaña de vista previa de fondo: estado de desplazamiento



Pestaña de vista previa de fondo al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.FileTabProvisionalHover</code>
Primer plano (texto)	<code>Environment.FileTabProvisionalHoverForeground</code>
Borde	<code>Environment.FileTabProvisionalHoverBorder</code> (Establecer en el mismo color que el fondo.)

Botón de desbordamiento de documento

El botón de desbordamiento de documento se muestra si hay uno o varios documentos abiertos, independientemente de si hay espacio vertical en la configuración actual para que quepan todas las pestañas de documentos. El menú desplegable de desbordamiento de documentos, que se controla mediante los colores del menú de la barra de [comandos](#), muestra una lista de todos los documentos abiertos, tanto visibles como ocultos, y el glifo de desbordamiento cambia en función de si todos los documentos abiertos se muestran en el canal de ficha.



Botón de desbordamiento de documento (línea roja)

USO...	NO UTILICE ...
... cuando se crea un botón de desbordamiento de documento personalizado.	... para la interfaz de usuario que no es similar a un botón de desbordamiento.
	... para los botones de desbordamiento de la barra de comandos.

Botón de desbordamiento de documento: estado predeterminado



Botón de desbordamiento de documento predeterminado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DocWellOverflowButtonBackground</code>
Primer plano (glifo)	<code>Environment.DocWellOverflowButtonGlyph</code>
Borde	N/D

Botón de desbordamiento de documento: estado de desplazamiento



Botón de desbordamiento de documento al mantener el puntero

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DocWellOverflowButtonMouseOverBackground</code>
Primer plano (glifo)	<code>Environment.DocWellOverflowButtonMouseOverGlyph</code>
Borde	<code>Environment.DocWellOverflowButtonMouseOverBorder</code>

Botón de desbordamiento del documento: estado pulsado

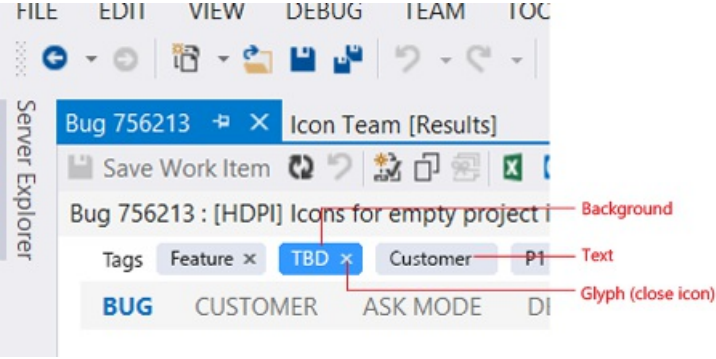


Botón de desbordamiento de documentos al presionar

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.DocWellOverflowButtonMouseDownBackground</code>
Primer plano (glifo)	<code>Environment.DocWellOverflowButtonMouseDownGlyph</code>
Borde	<code>Environment.DocWellOverflowButtonMouseDownBorder</code>

Etiquetado

Visual Studio admite el etiquetado, lo que permite al usuario declarar palabras clave de búsqueda con fines de seguimiento. Por ejemplo, los desarrolladores y los administradores de proyectos pueden usar Team Foundation Server (TFS) para etiquetar elementos de trabajo. Las tablas siguientes proporcionan nombres de colores para la etiqueta en sí y el glifo del "icono de cerrar" que aparecen en los estados Al desplazar el puntero y Seleccionado.

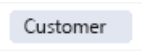


Etiquetado en Visual Studio (línea roja)

USO...	NO UTILICE ...
... para la interfaz de usuario que admite el etiquetado.	... para cualquier otro tipo de interfaz de usuario.

Etiquetas

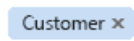
Etiqueta: estado predeterminado



Etiqueta predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.Background
Primer plano (texto)	Tag.Background

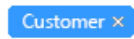
Etiqueta: estado de desplazamiento



Etiqueta al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.HoverBackground
Primer plano (texto)	Tag.HoverBackgroundText

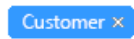
Etiqueta: estado presionado



Etiqueta presionada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.PressedBackground
Primer plano (texto)	Tag.PressedBackgroundText

Etiqueta: estado seleccionado

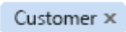


Etiqueta seleccionada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.SelectedBackground
Primer plano (texto)	Tag.SelectedBackgroundText

Cerrar×glifo de etiqueta ()

Cerrar×() glifo de etiqueta: estado predeterminado



Glifo de×etiqueta De cierre predeterminado ()

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (glifo)	Tag.TagHoverGlyph

Cerrar×() glifo de etiqueta: estado de desplazamiento



Cerrar×() glifo de etiqueta al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.TagHoverGlyphHoverBackground
Primer plano (glifo)	Tag.TagHoverGlyphHover
Borde	Tag.TagHoverGlyphHoverBorder

Cerrar×() glifo de etiqueta: estado presionado



Pictograma de etiqueta De cierre presionado (×)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Tag.TagHoverGlyphPressedBackground
Primer plano (glifo)	Tag.TagHoverGlyphPressed
Borde	Tag.TagHoverGlyphPressedBorder

Etiqueta seleccionada con×glifo Cerrar (): estado predeterminado



Etiqueta seleccionada predeterminada×con glifo Cerrar ()

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (glifo)	<code>Tag.TagSelectedGlyph</code>

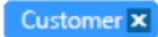
Etiqueta seleccionada con×el glifo Cerrar (): estado de desplazamiento: estado de desplazamiento



Etiqueta seleccionada con×el glifo Cerrar () al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Tag.TagSelectedGlyphHoverBackground</code>
Primer plano (glifo)	<code>Tag.TagSelectedGlyphHover</code>
Borde	<code>Tag.TagSelectedGlyphHoverBorder</code>

Etiqueta seleccionada con×glifo Cerrar (): estado presionado

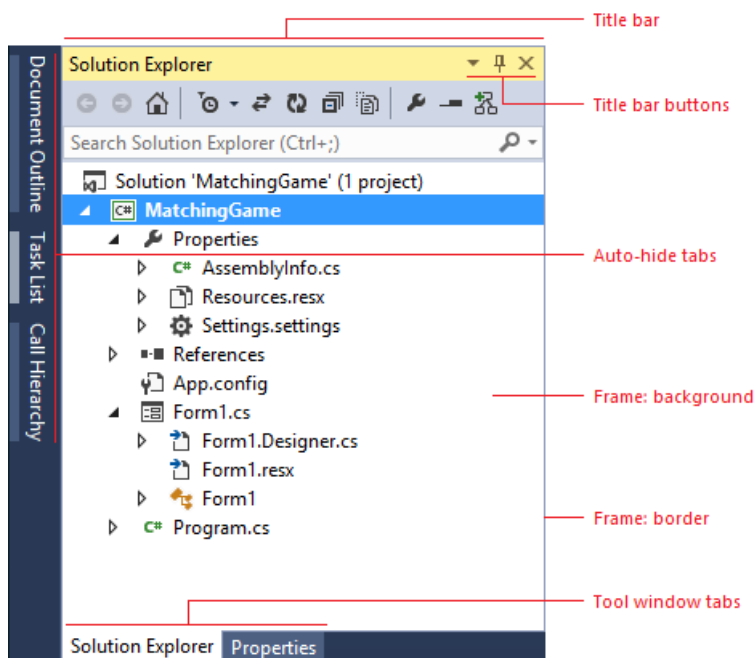


Etiqueta seleccionada y presionada×con el glifo Cerrar ()

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Tag.TagSelectedGlyphPressedBackground</code>
Primer plano (glifo)	<code>Tag.TagSelectedGlyphPressed</code>
Borde	<code>Tag.TagSelectedGlyphPressedBorder</code>

Ventanas de herramientas

No es necesario replicar las ventanas de herramientas, ya que las proporciona el entorno de Visual Studio. Sin embargo, quizá prefiera aprovechar los colores que se usan en las ventanas de herramientas para que la interfaz de usuario se muestre siempre coherente con esta parte del entorno de Visual Studio.

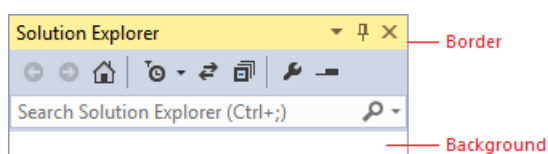


Ventana de herramientas (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las ventanas de herramientas.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

Marco de ventana de herramientas

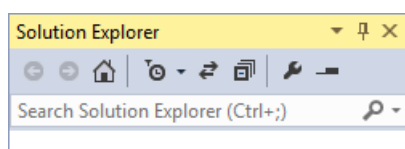
Las ventanas de herramientas de Visual Studio se usan para diversas tareas y pueden estar en uno de varios estados diferentes. Si una ventana de herramientas está abierta, se puede asignar a cualquiera de los cuatro lados del área del documento. Estas ventanas también pueden flotar fuera del IDE, lo que permite reubicarlas en cualquier lugar dentro de la pantalla del usuario. Las ventanas flotantes siempre se colocan arriba del IDE. Por último, las ventanas de herramientas se pueden acoplar como ventanas de documento y aparecen como una pestaña en el cuadro de documento. Las ventanas de herramientas que se han acoplado como ventanas de documento se colorean en parte mediante los nombres de token de ventana de documento.



Marco de la ventana de la herramienta (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las ventanas de herramientas.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

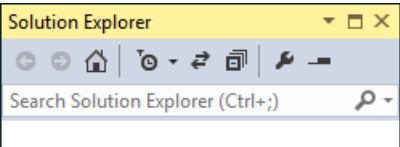
Ventana de herramientas acoplada



Ventana de herramientas acoplada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowBackground</code>
Borde	<code>Environment.ToolWindowBorder</code>

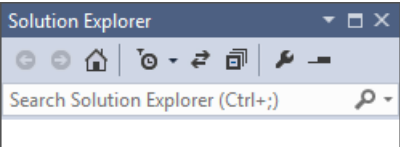
Ventana de herramientas flotante y enfocada



Ventana de herramientas flotante y enfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowBackground</code>
Borde	<code>Environment.MainWindowActiveDefaultBorder</code>

Ventana de herramientas flotante y desenfocada

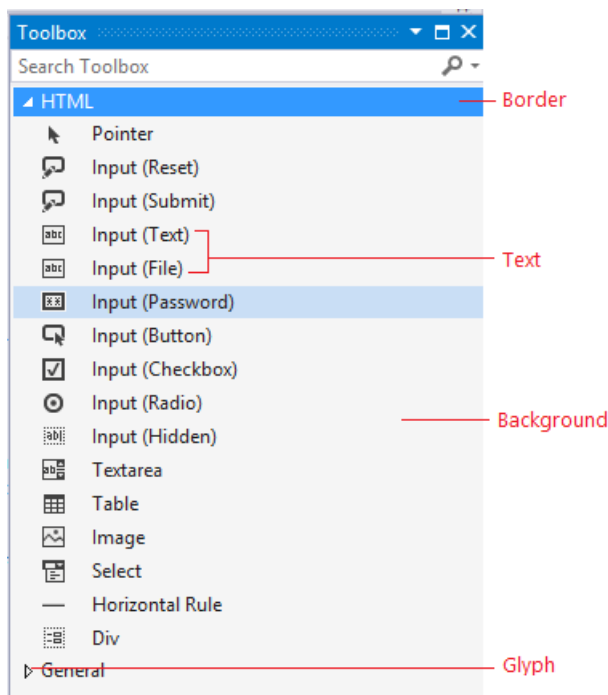


Ventana de herramientas flotante y desenfocada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowBackground</code>
Borde	<code>Environment.MainWindowInactiveBorder</code>

Ventanas tipo caja de herramientas

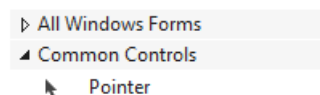
El cuadro de herramientas es una de las ventanas de herramientas comunes más utilizadas en Visual Studio. Es esencialmente un control de árbol con un tema especial y estilo aplicado.



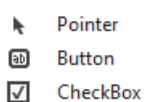
Ventana de tipo Caja de herramientas (línea roja)

USO...	NO UTILICE ...
... cuando está diseñando una ventana de herramientas que desea que siempre sea coherente con la caja de herramientas del shell.	... para cualquier cosa que no sea similar a la interfaz de usuario del cuadro de herramientas, o si no está seguro de si la interfaz de usuario tendrá problemas si cambian los colores del cuadro de herramientas del shell.

Nodos de Toolbox: estado predeterminado



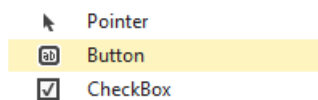
Nodo primario de la caja de herramientas predeterminada



Nodo secundario de la caja de herramientas predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolboxContent</code> (Encabezados)
Información previa	<code>Environment.ToolWindowBackground</code> (Elementos individuales o ventana completa si no hay controles disponibles)
Borde	None
Primer plano (glifo)	<code>Environment.ToolboxContent</code>
Primer plano (texto)	<code>Environment.ToolboxContent</code>

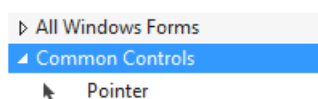
Nodos secundarios de Toolbox: estado de desplazamiento



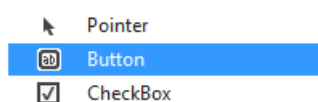
Nodo secundario de cuadro de herramientas al mantener el mouse

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	Environment.ToolboxContentMouseOver (Sólo artículos individuales)
Borde	None
Primer plano (texto)	Environment.ToolboxContentMouseOver (Sólo artículos individuales)

Nodos de caja de herramientas seleccionados: estado enfocado



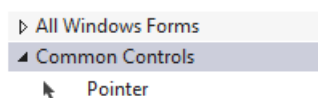
Nodo principal de la caja de herramientas seleccionado y centrado



Nodo secundario de la caja de herramientas centrado y seleccionado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemActive De categoría Tree view
Borde	TreeView.FocusVisualBorder De categoría Tree view
Primer plano (glifo)	TreeView.SelectedItemActive De categoría Tree view
Primer plano (texto)	TreeView.SelectedItemActive De categoría Tree view

Nodos de caja de herramientas seleccionados: estado desenfocado



Nodo primario de la caja de herramientas seleccionado y desenfocado



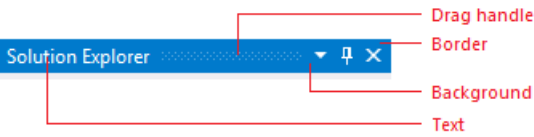
Nodo secundario de la caja de herramientas seleccionado y desenfocado

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	TreeView.SelectedItemInactive De categoría Tree view

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Borde	None
Primer plano (glifo)	<code>TreeView.SelectedItemInactive</code> De categoría Tree view
Primer plano (texto)	<code>TreeView.SelectedItemInactive</code> De categoría Tree view

Barra de título de la ventana de herramientas

El borde de la barra de título no es un borde verdadero, es una línea gruesa en la parte superior de la barra de título. No tiene un nombre de token para su estado desenfocado.



Barra de título de la ventana de herramientas (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las ventanas de herramientas.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

Barra de título con foco



Barra de título con foco

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.TitleBarActiveGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.TitleBarActiveText</code>
Borde	<code>Environment.TitleBarActiveBorder</code> (Establecer en el mismo color que el fondo.)
Controlador de arrastre	<code>Environment.TitleBarDragHandleActive</code>

Barra de título sin foco

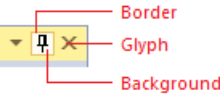


Barra de título sin foco

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.TitleBarInactiveGradientBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.TitleBarInactiveText</code>
Borde	N/D
Controlador de arrastre	<code>Environment.TitleBarDragHandle</code>

Botones de la barra de título de la ventana de herramientas



Botón de la barra de título (línea roja)

USO...	NO UTILICE ...
... para los botones que aparecen en la interfaz de usuario que usa tokens de color de las barras de título de la ventana de herramientas.	... para botones que aparecen en otras ubicaciones.
	... en cualquier combinación de fondo/primer plano que no sea especificada.

Botones de la barra de título enfocados: estado predeterminado



Botones de barra de título predeterminados y enfocados

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (glifo)	<code>Environment.ToolWindowButtonActiveGlyph</code>
Borde	N/D

Botones de la barra de título desenfocados: estado predeterminado



Botones de barra de título predeterminados y desenfocados

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	N/D
Primer plano (glifo)	<code>Environment.ToolWindowButtonInactiveGlyph</code>
Borde	N/D

Botones de la barra de título enfocados: estado de desplazamiento



Botones de la barra de título enfocados al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowButtonHoverActive</code>
Primer plano (glifo)	<code>Environment.ToolWindowButtonHoverActiveGlyph</code>
Borde	<code>Environment.ToolWindowButtonHoverActiveBorder</code>

Botones de la barra de título desenfocados: estado de desplazamiento



Botones de la barra de título desenfocados al pasar el ratón

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowButtonHoverInactive</code>
Primer plano (glifo)	<code>Environment.ToolWindowButtonHoverInactiveGlyph</code>
Borde	<code>Environment.ToolWindowButtonHoverInactiveBorder</code>

Botones de la barra de título enfocados: estado presionado



Botones de la barra de título enfocados en la prensa

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowButtonDown</code>
Primer plano (glifo)	<code>Environment.ToolWindowButtonDownActiveGlyph</code>
Borde	<code>Environment.ToolWindowButtonDownBorder</code>

Botones de la barra de título desenfocados: estado pulsado



Botones de barra de título desenfocados al presionar

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowButtonDown</code>
Primer plano (glifo)	<code>Environment.ToolWindowButtonDownInactiveGlyph</code>
Borde	<code>Environment.ToolWindowButtonDownBorder</code>

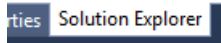
Pestañas de ventana de herramientas



Pestaña Ventana de herramientas (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las ventanas de herramientas.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

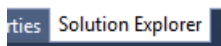
Pestaña de ventana de herramienta seleccionada, con foco



Pestaña de ventana de herramienta seleccionada, con foco

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowTabSelectedTab</code>
Primer plano (texto)	<code>Environment.ToolWindowTabSelectedActiveText</code>
Borde	<code>Environment.ToolWindowTabSelectedBorder</code> (Establecer en el mismo color que el fondo.)

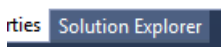
Pestaña de ventana de herramienta seleccionada, sin foco



Pestaña de ventana de herramienta seleccionada, sin foco

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowTabSelectedTab</code>
Primer plano (texto)	<code>Environment.ToolWindowTabSelectedText</code>
Borde	<code>Environment.ToolWindowTabSelectedBorder</code> (Establecer en el mismo color que el fondo.)

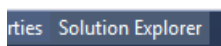
Pestaña de la ventana de la herramienta de fondo: estado predeterminado



Ficha predeterminada de la ventana de la ventana de la herramienta de fondo

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowTabGradientBegin</code> <code>Environment.ToolWindowTabGradientEnd</code> (Gradient se detiene establecido en el mismo valor de color en Visual Studio 2013.)
Primer plano (texto)	<code>Environment.ToolWindowTabText</code>
Borde	<code>Environment.ToolWindowTabBorder</code>

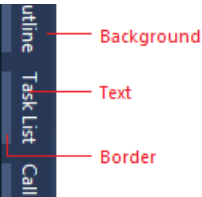
Pestaña de la ventana de la herramienta de fondo: estado de desplazamiento



Pestaña de ventana de herramientas en segundo plano al mantener el puntero

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.ToolWindowTabMouseOverBackgroundBegin</code> <code>Environment.ToolWindowTabMouseOverBackgroundEnd</code> (Gradient se detiene establecido en el mismo valor de color en Visual Studio 2013.)
Primer plano (texto)	<code>Environment.ToolWindowTabMouseOverText</code>
Borde	<code>Environment.ToolWindowTabMouseOverBorder</code> (Establecer en el mismo color que el fondo.)

Pestaña de ocultación automática



Ocultar automáticamente pestañas (línea roja)

USO...	NO UTILICE ...
... en cualquier lugar en el que esté creando la interfaz de usuario que desea que coincida con las pestañas de la ventana de herramientas ocultas automáticamente.	... para cualquier interfaz de usuario que no desee cambiar automáticamente si el shell tiene una actualización del tema.

Ocultar pestañas automáticamente: estado predeterminado



Pestaña de ocultación automática predeterminada

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.AutoHideTabBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)
Primer plano (texto)	<code>Environment.AutoHideTabText</code>
Borde	<code>Environment.AutoHideTabBorder</code>

Ocultar pestañas automáticamente: estado de desplazamiento



Pestaña de ocultación automática al mantener el puntero

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Información previa	<code>Environment.AutoHideTabMouseOverBackgroundBegin</code> (Se detiene el degradado para este token no utilizado en la interfaz de usuario temática.)

ELEMENTO	NOMBRE DEL TOKEN: CATEGORY.COLOR
Primer plano (texto)	<code>Environment.AutoHideTabMouseOverText</code>
Borde	<code>Environment.AutoHideTabMouseOverBorder</code>

Referencia de valor de color para Visual Studio

02/12/2019 • 33 minutes to read • [Edit Online](#)

En este tema es una referencia para la creación de nuevos tokens de color. Al crear la nueva interfaz de usuario, puede usar colores similares de la interfaz de usuario existente para elegir los colores para cada tema principal y el nombre del color del sistema para ese elemento para los temas de contraste alto.

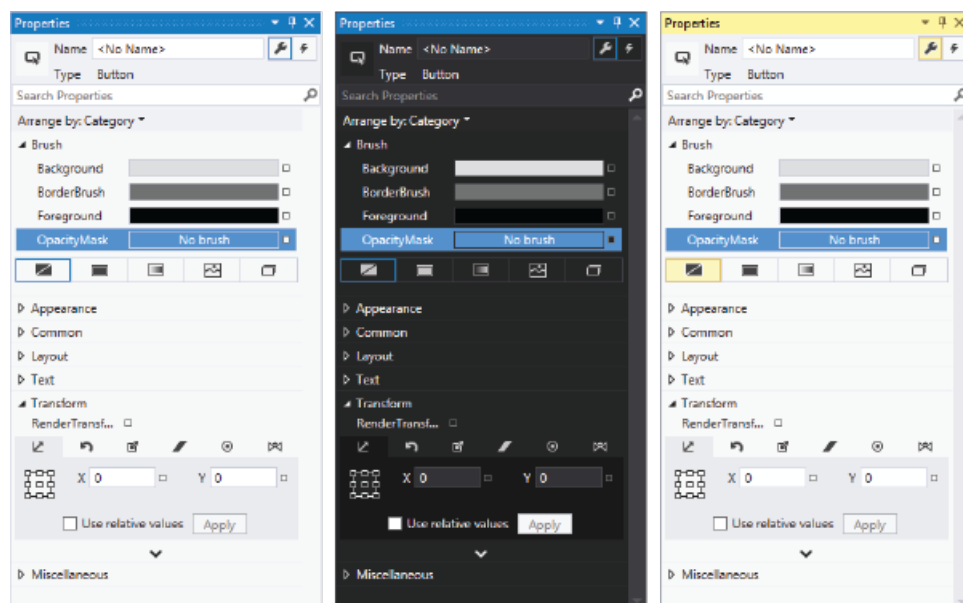
Para obtener más información sobre la creación de nuevos tokens de color, vea [herramientas de creación de temas de Color](#).

Para que los tokens de color usar con la interfaz de usuario existente, consulte [compartido colores para Visual Studio](#).

Al usar a esta guía de referencia, recuerde:

- **Copiar siempre los matices de primer plano y fondo juntos.** No elija un primer plano de un elemento de interfaz de usuario y un fondo de otro solo porque le gusta los matices.
- **Estos son los códigos hexadecimales, no los tokens de color.** No se puede usar los nombres de token de color real de estos elementos de interfaz de usuario porque no se puede controlar cómo se puede cambiar los tokens en el futuro. En su lugar, cree tokens dentro de su categoría y use los valores de matiz que se muestra en las tablas. Los tokens de color de la interfaz de usuario existente se puede encontrar en [compartido colores para Visual Studio](#).
- **Muestras de contraste altos se muestran para la comparación solo.** Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.
- **Para obtener mejores resultados, el color de par establece con la interfaz de usuario similar del ejemplo.**

Propiedades (ventana)



Ventana de propiedades en los temas claro, oscuro y azul

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Líneas divisorias

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Default	 #FFEEEEF2	 #FF2D2D30	 #FFEEEEF2	 ControlDark

Glifo de expansión

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Window
Primer plano: al mantener el mouse	 #FF007ACC	 #FF007ACC	 #FF007ACC	 HighlightText
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFFFCF4	 Highlight
Borde	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Window
Borde: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFE5C365	 WindowFrame

Los controles de ficha insertados: cuerpo

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #FFE7E8EC	 #FF1B1B1C	 #FFE7E8EC	 Control
Borde	 #FFE7E8EC	 #FF1B1B1C	 #FFE7E8EC	 WindowFrame

Los controles de ficha insertados: pestaña activa

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo	 #FFE7E8EC	 #FF1B1B1C	 #FFE7E8EC	 Highlight
Borde	 #FFE7E8EC	 #FF1B1B1C	 #FFE7E8EC	 WindowFrame

Los controles de ficha insertados: ficha inactiva

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control
Primer plano: al mantener el mouse	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 HighlightText
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFEFEFE	 Highlight
Primer plano: deshabilitada	 #FFA2A4A5	 #FF656565	 #FFA2A4A5	 GrayText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
En segundo plano: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control
Borde	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 WindowFrame
Borde: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFEFEFE	 WindowFrame
Borde: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 WindowFrame

Encabezado de sección

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Primer plano: al mantener el mouse	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 HighlightText
Fondo	 #FFEEEEF2	 #FF292929	 #FFEEEEF2	 Control
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFFFCF4	 Highlight
Borde	Ninguna	Ninguna	Ninguna	Ninguna
Borde: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFE5C365	 WindowFrame

Grupo de categorías

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano del cuerpo	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo del cuerpo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control
Primer plano de glifo	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 WindowText
Primer plano del glifo: al mantener el mouse	 #FF007ACC	 #FF007ACC	 #FF007ACC	 WindowText

Botones de alternancia

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 WindowText
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Window
Primer plano: al mantener el mouse	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 HighlightText
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFFFCF4	 Highlight
Primer plano: seleccionado	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 HighlightText
En segundo plano: seleccionado	 #FFF5F5F5	 #FF252526	 #FFFD4BF	 Highlight
Primer plano: deshabilitada	 #FFA2A4A5	 #FF656565	 #FFA2A4A5	 ControlText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
En segundo plano: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Window
Borde	 #FFCCCEDB	 #FF3F3F46	 #FFCCCEDB	 Window
Borde: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFE5C365	 WindowFrame
Borde: seleccionado	 #FF007ACC	 #FF007ACC	 #FFE5C365	 WindowFrame
Borde: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 WindowFrame

Campos de entrada

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo	 #FFFFFFF5	 #FF333337	 #FFF6F6F6	 Control
Primer plano: deshabilitada	 #FFA2A4A5	 #FF656565	 #FFA2A4A5	 GrayText
En segundo plano: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control
Borde	 #FFCCCEDB	 #FF434346	 #FFCCCEDB	 ControlDark
Borde: deshabilitada	 #FFCCCEDB	 #FF434346	 #FFC6C6C6	 ControlDark

Elementos de lista

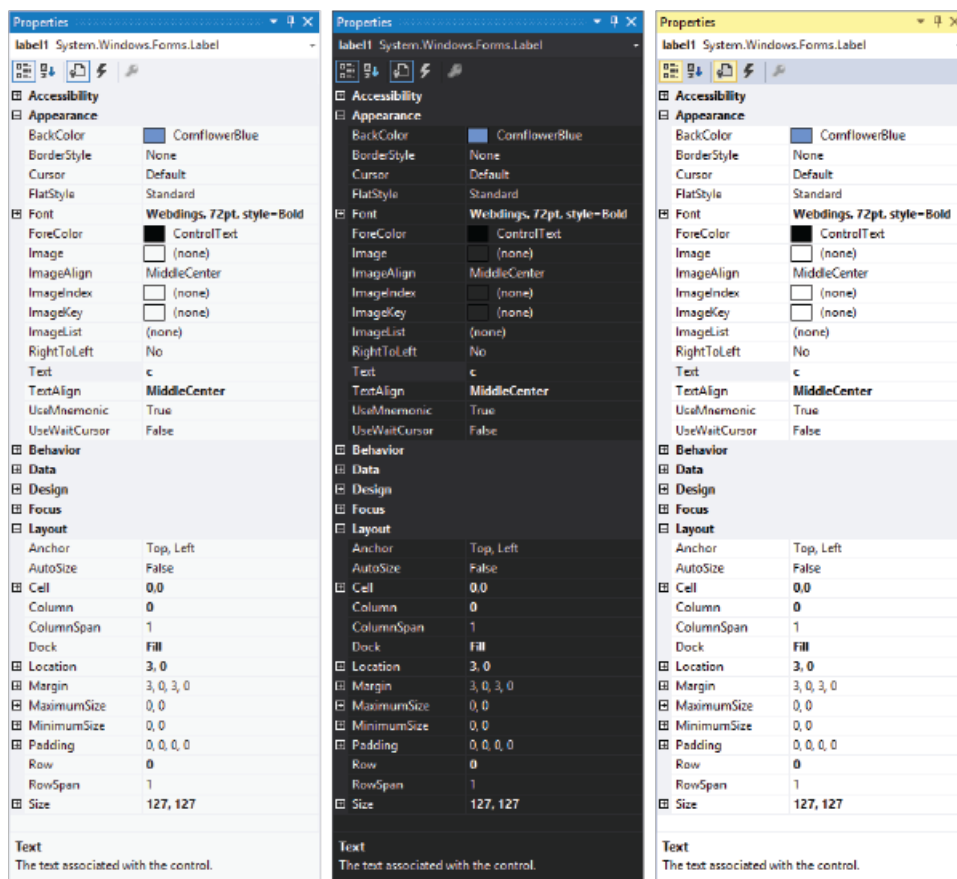
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Primer plano: al mantener el mouse	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 HighlightText
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #72555555	 #FFFD4BF	 Highlight
Primer plano: seleccionado	 #FFFFFFF	 #FFFFFFF	 #FFFFFFF	 HighlightText
En segundo plano: seleccionado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight
Primer plano: deshabilitada	 #FFA2A4A5	 #FF656565	 #FFA2A4A5	 GrayText
Borde: al mantener el mouse	 #FFC9DEF5	 #72555555	 #FFFD4BF	 WindowFrame
Borde: seleccionado	 #FFCCCEDB	 #FF3399FF	 #FF3399FF	 WindowFrame

Botones de comando

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFEFEFE	 Highlight

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano: seleccionado	 #FFFFFF	 #FFFFFF	 #FFFFFF	 HighlightText
En segundo plano: presionado	 #FF007ACC	 #FF007ACC	 #FF007ACC	 Highlight
Primer plano: deshabilitada	 #FFA2A4A5	 #FF656565	 #FFA2A4A5	 GrayText
En segundo plano: deshabilitada	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Window
Borde: al mantener el mouse	 #FFC9DEF5	 #FF3E3E40	 #FFFEFEFE	 HighlightText
Borde: presionado	 #FF007ACC	 #FF007ACC	 #FF007ACC	 HighlightText
Borde: deshabilitada	 #FFCCCEDB	 #FF434346	 #FFC6C6C6	 WindowFrame

Elementos de control de cuadrícula de propiedades



Cuadrícula de propiedades en los temas claro, oscuro y azul

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Contenido

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF000000	 WindowText
Fondo	 #FFF5F5F5	 #FF252526	 #FFFFFFF5	 Control
De primer plano: se selecciona, centrado	 #FFFFFFF5	 #FFFFFFF5	 #FFFFFFF5	 HighlightText
En segundo plano: se selecciona, centrado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
De primer plano: seleccionada, sin foco	 #FF1E1E1E	 #FFF1F1F1	 #FF000000	 WindowText
En segundo plano: seleccionada, sin foco	 #FFEEEEF2	 #FF2D2D30	 #FFEEEEF2	 ControlDark
Líneas divisorias	 #FFEEEEF2	 #FF2D2D30	 #FFEEEEF2	 ControlDark
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF000000	 WindowText
Fondo	 #FFEEEEF2	 #FF2D2D30	 #FFEEEEF2	 ControlDark

IU de CodeLens

```

        int randomNumber = random.Next(icons.Count);
        iconLabel.Text = icons[randomNumber];
    }

    MatchingGame\Form1.cs\Form1.Designer.cs (16)
    98: this.label16.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    110: this.label15.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    122: this.label14.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    134: this.label13.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    146: this.label12.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    158: this.label11.MouseClick += new System.Windows.Forms.MouseEventHandler(this.label_Click);
    Show on Code Map | Collapse All
    <param name="e"></param>
    16 references
    private void label_Click(object sender, MouseEventArgs e)
    {
        // The timer is only on after two non-matching
        // icons have been shown to the player,
        // so ignore any clicks if the timer is running
        if (timer1.Enabled == true)
            return;
    }

```

NOTE

Colores de CodeLens UI son los mismos para los temas claro, oscuro y azul.

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

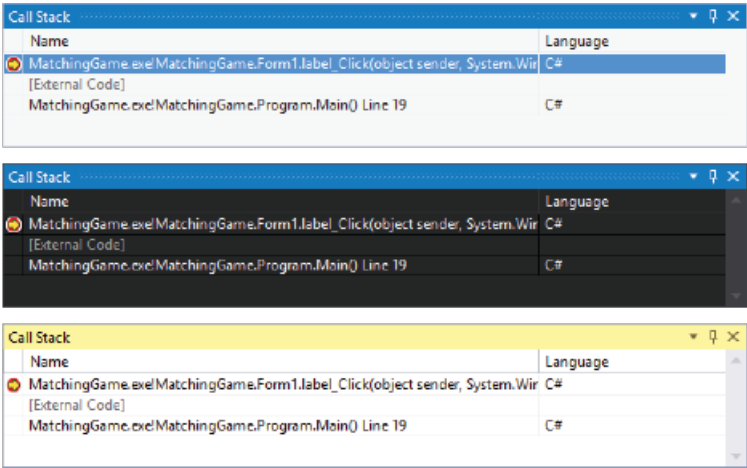
Cuerpo

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FF1E1E1E	 #FF1E1E1E	 WindowText
Fondo	 #FFFCFCFC	 #FFFCFCFC	 #FFFCFCFC	 Window
Primer plano: seleccionado	 #FFFFFFF	 #FFFFFFF	 #FFFFFFF	 HighlightText
En segundo plano: seleccionado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight
En segundo plano: resaltado	 #FFFDFBAC	 #FFFDFBAC	 #FFFDFBAC	 HotTrack
Borde	 #FF6FC2E9	 #FF6FC2E9	 #FF6FC2E9	 WindowFrame

Hipervínculo y separador

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano de hipervínculo	 #FF0E70C0	 #FF0E70C0	 #FF0E70C0	 HotTrack
Primer plano del separador	 #FF9C9C9C	 #FF9C9C9C	 #FF9C9C9C	 HotTrack

Grid (control)



Control de cuadrícula de luz (superior), oscuro (centro) y temas de color azul (inferior)

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Cuerpo

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF000000	 WindowText
Primer plano: seleccionado	 #FFFFFFFF	 #FFFFFFFF	 #FFFFFFFF	 HighlightText

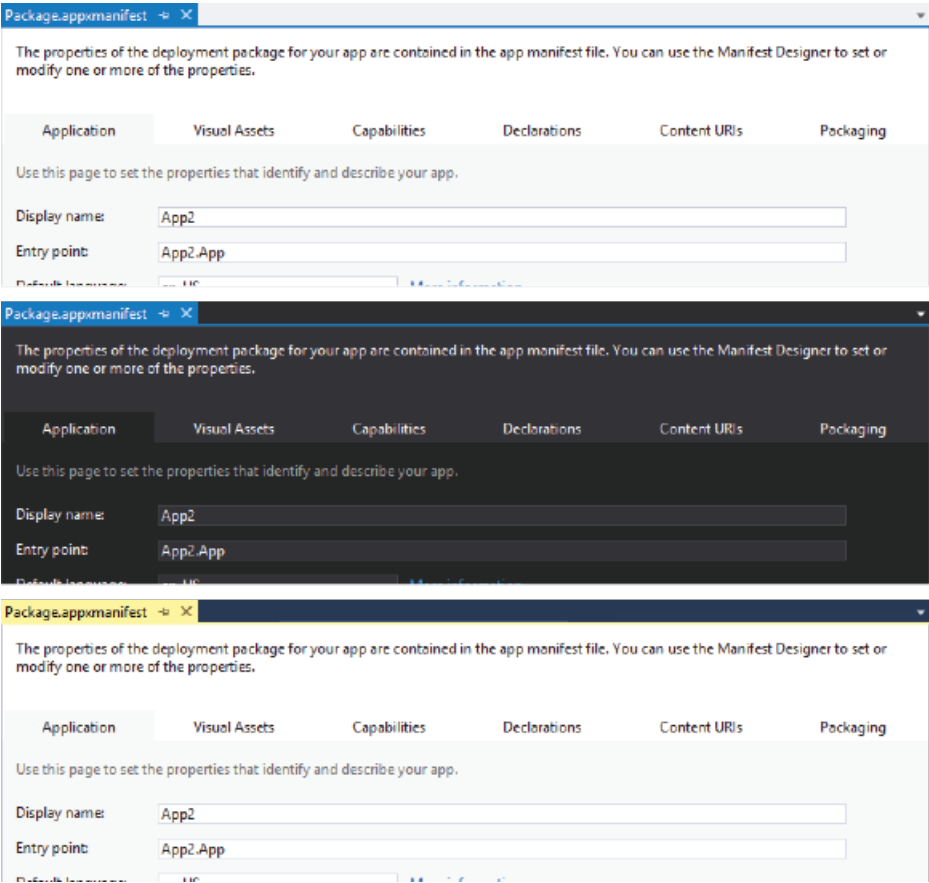
Encabezado de la subsección

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF717171	 #FF999999	 #FF717171	 GrayText
Fondo	 #FFF5F5F5	 #FF252526	 #FFFFFFFF	 Window
En segundo plano: seleccionado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight

Líneas de cuadrícula

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Elementos (primer plano)	 #FFF0F0F0	 #FF000000	 #FFF0F0F0	 ScrollBar
Encabezado (primer plano)	 #FFE0E3E6	 #FF333337	 #FFBEC3CB	 ControlDark

Diseñador de manifiestos



Diseñador de manifiestos en claro (superior), oscuro (centro) y los temas de color azul (inferior)

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Cuerpo

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control

Pestaña activa

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ActiveCaptionText
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 ActiveCaption

Pestaña inactiva

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 InactiveCaptionText
Fondo	 #FFFEFEFE	 #FF333337	 #FFFEFEFE	 HighlightText
Primer plano: al mantener el mouse	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 InactiveCaption
En segundo plano: al mantener el mouse	 #FFCCCEDB	 #FF3F3F46	 #FFCCCEDB	 Highlight

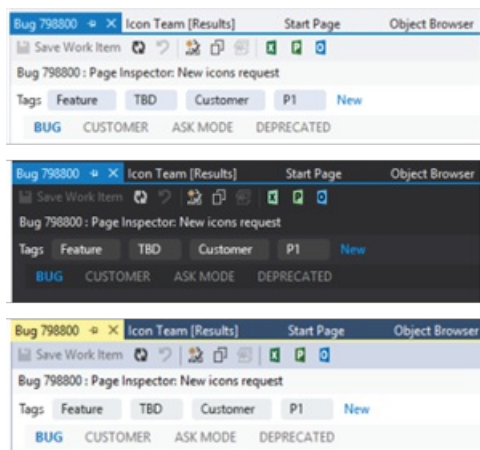
Texto de marca de agua

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF999999	 #FF999999	 #FF717171	 GrayText
Fondo	 #FFF5F5F5	 #FF252526	 #FFF6F6F6	 Control

Panel de descripción

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano	 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	 ControlText
Fondo	 #FFEEEE2	 #FF252526	 #FFFFFFF7	 Control

Etiquetado TFS



Etiquetado TFS en Light (superior), oscuro (centro) y los temas de color azul (inferior)

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Etiquetas

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Primer plano (texto)	 #FF1E1E1E	 #FFFFFFF7	 #FF1E1E1E	 ActiveCaptionText
Fondo	 #FFE1E6F1	 #FF424242	 #FFE9ECEE	 ActiveCaption
Primer plano (texto): al mantener el mouse	 #FF1E1E1E	 #FFFFFFF7	 #FF1E1E1E	 HighlightText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
En segundo plano: al mantener el mouse	 #FFC9DEF5	 #FF606060	 #FFFD4BF	 Highlight
Primer plano (texto): presionado	 #FFFFFF	 #FFFFFF	 #FFFFFF	 HighlightText
En segundo plano: presionado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight
Primer plano (texto): seleccionado	 #FFFFFF	 #FFFFFF	 #FFFFFF	 HighlightText
En segundo plano: seleccionado	 #FF3399FF	 #FF3399FF	 #FF3399FF	 Highlight

Glifo al mantener el mouse

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Glifo	 #FF1E1E1E	 #FFFFFF	 #FF1E1E1E	 HighlightText
Fondo	 #FFF7F7F9	 #FF393939	 #FFFFCF4	 HighlightText
Borde	 #FFF7F7F9	 #FF393939	 #FFE5C365	 HighlightText
Glifo (presione y mantenga el puntero)	 #FFFFFF	 #FFFFFF	 #FFFFFF	 Highlight
En segundo plano (presione y mantenga el puntero)	 #FF0E6198	 #FF007ACC	 #FFFD4BF	 HighlightText
Borde (presione y mantenga el puntero)	 #FF0E6198	 #FF007ACC	 #FFE5C365	 HighlightText

Glifo seleccionado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Glifo	#FFFFFF	#FFFFFF	#FFFFFF	HighlightText
Fondo	#FF52B0EF	#FF52B0EF	#FF52B0EF	HighlightText
Borde	#FF52B0EF	#FF52B0EF	#FF52B0EF	HighlightText
Glifo (presione y mantenga el puntero)	#FFFFFF	#FFFFFF	#FFFFFF	Highlight
En segundo plano (presione y mantenga el puntero)	#FF0E6198	#FF0E6198	#FF0E6198	HighlightText
Borde (presione y mantenga el puntero)	#FF0E6198	#FF0E6198	#FF0E6198	HighlightText

Controles de botón

NOTE

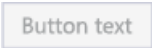
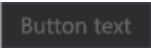
Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Default

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	Button text	Button text	Button text	Button text
Primer plano	#1E1E1E	#F1F1F1	#1E1E1E	ControlText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #ECECF0	 #3F3F46	 #ECECF0	 Control
Borde	 #CCCEDB	 #555555	 #CCCEDB	 ControlDark

Deshabilitado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	 Button text	 Button text	 Button text	 Button text
Primer plano	 #A2A4A5	 #656565	 #A2A4A5	 InactiveCaptionText
Fondo	 #F5F5F5	 #2D2D30	 #F5F5F5	 InactiveCaption
Borde	 #CCCEDB	 #3F3F46	 #CCCEDB	 InactiveBorder

Al mantener el puntero

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	 Button text	 Button text	 Button text	 Button text
Primer plano	 #1E1E1E	 #F1F1F1	 #1E1E1E	 ActiveCaptionText
Fondo	 #C9DEF5	 #3F3F46	 #C9DEF5	 ActiveCaption
Borde	 #CCCEDB	 #007ACC	 #3399FF	 ActiveBorder

Presionado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	Button text	Button text	Button text	Button text
Primer plano	#FFFFFF	#F1F1F1	#FFFFFF	HighlightText
Fondo	#007ACC	#007ACC	#007ACC	Highlight
Borde	#007ACC	#007ACC	#007ACC	HighlightText

Con foco

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	Button text	Button text	Button text	Button text
Primer plano	#1E1E1E	#F1F1F1	#1E1E1E	HighlightText
Fondo	#C9DEF5	#3F3F46	#C9DEF5	Highlight
Borde	#3399FF	#007ACC	#3399FF	HighlightText

Controles de casilla

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

Default

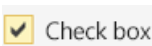
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	☒ Check box	☒ Check box	☒ Check box	☒ Check box

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #FEFEFE	 #252526	 #FEFEFE	 Control
Borde	 #717171	 #999999	 #717171	 ControlDark
Texto	 #1E1E1E	 #F1F1F1	 #1E1E1E	 ControlText
Glifo	 #1E1E1E	 #F1F1F1	 #1E1E1E	 ControlText

Deshabilitado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				
Fondo	 #F6F6F6	 #2D2D30	 #F6F6F6	 InactiveCaption
Borde	 #C6C6C6	 #434346	 #C6C6C6	 InactiveBorder
Texto	 #A2A4A5	 #656565	 #A2A4A5	 InactiveCaption
Glifo	 #A2A4A5	 #656565	 #A2A4A5	 InactiveCaptionText

Al mantener el puntero

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #F3F9FF	 #1F1F20	 #FDF4BF	 Highlight
Borde	 #3399FF	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #F1F1F1	 #1E1E1E	 HighlightText
Glifo	 #1E1E1E	 #F1F1F1	 #424242	 HighlightText

Presionado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	 Check box	 Check box	 Check box	 Check box
Fondo	 #007ACC	 #007ACC	 #E5C365	 Highlight
Borde	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #F1F1F1	 #1E1E1E	 HighlightText
Glifo	 #FFFFFF	 #F1F1F1	 #1E1E1E	 HighlightText

Con foco

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	 Check box	 Check box	 Check box	 Check box

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #FDF4BF	 #1F1F20	 #FDF4BF	 Highlight
Borde	 #3399FF	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #F1F1F1	 #1E1E1E	 HighlightText
Glifo	 #1E1E1E	 #F1F1F1	 #424242	 HighlightText

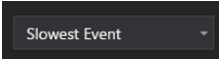
Controles de lista desplegable o cuadro combinado

NOTE

Muestras de contraste altos se muestran para la comparación solo. Use el nombre del color del sistema para la interfaz de usuario de contraste alto y no use un selector de color o Cuentagotas en estas muestras. Muestras de contraste altos en este tema reflejan el tema de alto contraste #1 en Windows. Los colores tendrá un aspecto diferentes en otros temas de contraste alto.

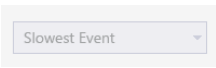
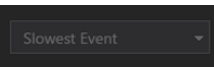
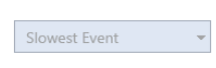
Apariencia inicial

Default

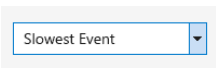
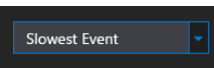
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia	 Slowest Event	 Slowest Event	 Slowest Event	 Slowest Event
Fondo	 #007ACC	 #333337	 #FCFCFC	 Control
Borde	 #CCCEDB	 #434346	 #8591A2	 ControlLight
Separador	 #FFFFFF	 #333337	 #FCFCFC	 ControlLight

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Glifo	 #717171	 #999999	 #1B293E	 ControlText
Fondo de glifo	 #FFFFFF	 #333337	 #FCFCFC	 #Control

Deshabilitado

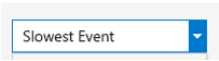
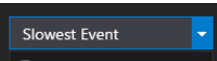
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				
Fondo	 #EEEEF2	 #2D2D30	 #DFE7F3	 InactiveCaption
Borde	 #CCCEDB	 #434346	 #A4ADBA	 InactiveBorder
Texto	 #A2A4A5	 #656565	 #A2A4A5	 InactiveCaptionText
Separador	 #EEEEF2	 #2D2D30	 #D5DCE8	 InactiveBorder
Glifo	 #CCCEDB	 #656565	 #A2A4A5	 InactiveCaptionText
Fondo de glifo	 #EEEEF2	 #2D2D30	 #D5DCE8	 InactiveCaption

Al mantener el puntero

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				

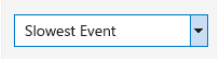
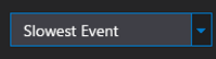
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #007ACC	 #3F3F46	 #FCFCFC	 Highlight
Borde	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #007ACC	 #000000	 HighlightText
Separador	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Glifo	 #1E1E1E	 #007ACC	 #000000	 HighlightText
Fondo de glifo	 #C9DEF5	 #1F1F20	 #FDF4BF	 Highlight

Presionado

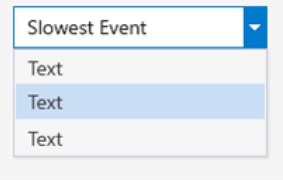
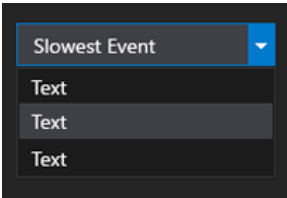
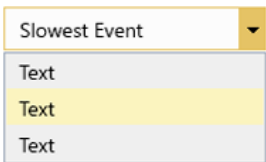
	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				
Fondo	 #FFFFFF	 #3F3F46	 #FCFCFC	 Highlight
Borde	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #FFFFFF	 #000000	 HighlightText
Separador	 #007ACC	 #007ACC	 #E5C365	 HighlightText

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Glifo	 #FFFFFF	 #FFFFFF	 #000000	 HighlightText
Fondo de glifo	 #007ACC	 #007ACC	 #E5C365	 Highlight

Con foco

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				
Fondo	 #FFFFFF	 #3F3F46	 #FCFCFC	 Highlight
Borde	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Texto	 #1E1E1E	 #FFFFFF	 #000000	 HighlightText
Separador	 #007ACC	 #007ACC	 #E5C365	 HighlightText
Glifo	 #1E1E1E	 #007ACC	 #000000	 HighlightText
Fondo de glifo	 #C9DEF5	 #3F3F46	 #FCFCFC	 Highlight

Lista (que aparece al presionar)

TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
			

Default

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	#F6F6F6	#1B1B1C	#EFEFEF	Menu
Borde	#CCCEDB	#3F3F46	#98A7B7	MenuText
Texto	#1E1E1E	#FFFFFF	#000000	MenuText
Sombra de fondo	#19000000	#19000000	#19000000	ControlDark

Al mantener el puntero

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	#C9DEF5	#3F3F46	#FDF4BF	Highlight
Borde	#CCCEDB	#3F3F46	#FDF4BF	HighlightText
Texto	#1E1E1E	#FFFFFF	#000000	HighlightText

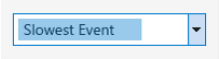
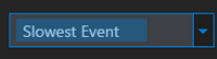
Presionado

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	#C9DEF5	#3F3F46	#FDF4BF	Highlight
Borde	#CCCEDB	#3F3F46	#FDF4BF	HighlightText
Texto	#1E1E1E	#FFFFFF	#000000	HighlightText

Con foco

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Fondo	 #F6F6F6	 #3F3F46	 #FDF4BF	 Highlight
Borde	 #C9DEF5	 #3F3F46	 #FDF4BF	 HighlightText
Texto	 #1E1E1E	 #FFFFFF	 #000000	 HighlightText

Selección de entrada de texto

	TEMA CLARO	TEMA OSCURO	TEMA AZUL	CONTRASTE ALTO
Apariencia				
Selección de entrada de texto	 #66007ACC	 #66007ACC	 #66007ACC	 HighlightText

Muestras de color por función

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FFC9DEF5	 #FF2D2D30	 #FFFDF4BF	Los elementos de menú, elementos del cuadro combinado, botones de barra de comandos, botones desplegables y botones de acción: en segundo plano
 #FFCCCEDB	 #FF3F3F46	 #FF465A7D	Borde de pestaña de ocultación automática
 #FFEEEEF2	 #FF2D2D30	 #FF293955	Fondos de predeterminado y mantenga el puntero de pestaña de ocultación automática
 #FF444444	 #FFD0D0D0	 #FFFFFFFF	Texto de la pestaña de ocultación automática, texto de barra de título de ventana de herramientas no enfocado y texto de ficha de ventana de herramientas

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FF0E70C0	 #FF0097FB	 #FFFFFF	Texto de la pestaña de ocultar automáticamente: al mantener el mouse
 #FF1C97EA	 #FF1C97EA	 #FF5B7199	Las pestañas de documentos en segundo plano: fondo y el borde al mantener el mouse
 #FFFFFF	 #FF333337	 #FFFFFF	Campo de entrada de cuadro combinado y cuadro de búsqueda: fondo predeterminado
 #FFFFFF	 #FF333337	 #FFFFFF	Cuadro combinado y algunos elementos de menú de lista desplegable: en segundo plano al mantener el mouse y mueva el mouse hacia abajo
 #FF007ACC	 #FF1F1F20	 #FFFDF4BF	Botones del cuadro combinado: en segundo plano al mantener el mouse
 #FFCCCEDB	 #FF434346	 #FFA4ADBA	Cuadro combinado, control de búsqueda, lista desplegable y cuadro de texto: borde deshabilitado
 #FFEEEEF2	 #FF2D2D30	 #FFD5DCE8	Cuadro combinado desplegable y cuadro de texto: deshabilitado fondos
 #FF717171	 #FFF1F1F1	 #FF1B293E	Cuadro combinado: valor predeterminado de glifo
 #FFC9DEF5	 #72555555	 #FFFDF4BF	Opciones de la barra de comandos: en segundo plano al mantener el mouse
 #FF717171	 #FF999999	 #FF1B293E	Opciones de la barra de comandos: valor predeterminado de glifo
 #FF007ACC	 #FF007ACC	 #FF1B293E	Opciones de la barra de comandos: mantenga el puntero de glifo

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FFFFFF	 #FFFFFF	 #FF1B293E	Opciones de la barra de comandos: mouse hacia abajo de glifo
 #FF1E1E1E	 #FFF1F1F1	 #FF1B293E	Texto de la barra de comandos, la lista desplegable y el texto de elemento de menú de cuadro combinado
 #FF1E1E1E	 #FFF1F1F1	 #FF000000	Barra de comandos: mantenga el mouse, del mouse hacia abajo y el texto seleccionado
 #FFCCCEDB	 #FF222222	 #FF8591A2	Barra de comandos: separador de barra de herramientas
 #FFF5F5F5	 #FF46464A	 #FFD6DBE9	Barra de comandos: separador resaltado de la barra de herramientas
 #FFEEEEF2	 #FF2D2D30	 #FFD6DBE9	Fondo del estante de comandos
 #FFCCCEDB	 #FF434346	 #FF8591A2	Bordes predeterminados y los separadores de barra de comandos
 #FFCCCEDB	 #FF434346	 #FF8591A2	Lista desplegable lista y combinado cuadro selección/input campo borde predeterminado
 #FF717171	 #FF999999	 #FF000000	Glifos predeterminados de menú y submenú
 #FFF6F6F6	 #FF1B1B1C	 #FFEAF0FF	Fondo de menú predeterminado
 #FF007ACC	 #FF007ACC	 #FF007ACC	Fondo predeterminado de la barra de estado

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FFA2A4A5	 #FF656565	 #FFA2A4A5	Texto deshabilitado
 #FFE7E8EC	 #FF1B1B1C	 #FFFFFF	Fondo del destino de acoplamiento
 #FFCCEDB	 #FF333337	 #FF636871	Borde del destino de acoplamiento
 #FFF5F5F5	 #FF252526	 #FFF5F8FB	Fondo de botón de destino de acoplamiento
 #FFF5F5F5	 #FF252526	 #FFFDE8A7	Fondo del glifo de destino de acoplamiento
 #FF52B0EF	 #FF52B0EF	 #FFFFFCF4	Botones de la ventana de documento y herramientas: fondos activos al mantener el mouse
 #FF007ACC	 #FF007ACC	 #FFE5C365	Botón desplegable y fondos de botón de mouse hacia abajo la barra de comandos
 #FFC9DEF5	 #FF1F1F20	 #FFFDF4BF	Botones desplegables: en segundo plano al mantener el mouse
 #FFF6F6F6	 #FF1B1B1C	 #FFEFEFEF	Fondo del menú desplegable del cuadro de lista desplegable y cuadro combinado
 #FFFFFF	 #FF3F3F46	 #FFFCFCFC	Lista desplegable, cuadro combinado y menú desplegable de control de búsqueda: en segundo plano al mantener el mouse y mueva el mouse hacia abajo
 #FF1E1E1E	 #FF007ACC	 #FF000000	Lista desplegable, cuadro combinado, control de búsqueda, menú y los glifos de submenú al mantener el mouse

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FF717171	 #FF999999	 #FF1B293E	Lista desplegable, menú y botón de expansión: default glifos
 #FF007ACC	 #FF434346	 #FFE5C365	Lista desplegable: borde al mantener el mouse y mueva el mouse hacia abajo
 #FFFFFF	 #FF333337	 #FFFCFCFC	Lista desplegable: fondo predeterminado
 #FF0E70C0	 #FF0097FB	 #FF0E70C0	Texto de vínculo del menú desplegable
 #FFC9DEF5	 #FF1F1F20	 #FFFDF4BF	Fondo de botón de lista desplegable del cuadro combinado enfocado
 #FF007ACC	 #FF007ACC	 #FF293955	Centra el título de la ventana de herramienta de barras y fondo de ficha de ventana de documento
 #FF007ACC	 #FF007ACC	 #FFFFF29D	Centra la barra de título de ventana de herramientas y documentos borde de ficha de ventana
 #FFFFFF	 #FFFFFF	 #FFFFFF	Glifos que aparecen en las superficies de oscuras
 #FFEEEEF2	 #FF2D2D30	 #FF293955	Fondos de pestaña de IDE y vista previa no seleccionado
 #FFEEEEF2	 #FF2D2D30	 #FF35496A	Degradado de fondo IDE
 #FFA2A4A5	 #FF656565	 #FF808080	Barra de comandos inactiva y menú texto

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #D8FFFFFF	 #72555555	 #FFFFFFCF4	Ventana principal y botones de la ventana rafted: en segundo plano al mantener el mouse
 #D8FFFFFF	 #72555555	 #FFE5C365	Ventana principal y botones de la ventana rafted: borde al mantener el mouse
 #FFE0E3E6	 #FF333337	 #FFCCCEDB	Separadores de grupo del menú de la barra de menús
 #FFF6F6F6	 #FF1B1B1C	 #FFF2F4FE	Fondo de canal de icono de menú
 #FFC9DEF5	 #FF333334	 #FFFD4BF	Los elementos de menú: en segundo plano al mantener el mouse
 #FF1E1E1E	 #FFF1F1F1	 #FF1B293E	Menú: mouse hacia abajo de glifo
 #FFC9DEF5	 #FF3E3E40	 #FFFD4BF	La mayoría de fondos y bordes al mantener el mouse
 #FF007ACC	 #FF007ACC	 #FFE5C365	La mayoría de los bordes y separadores de mouse hacia abajo y mantenga el mouse
 #FF1E1E1E	 #FFF1F1F1	 #FF1E1E1E	La mayoría de texto
 #FF6A6A6A	 #FFEFEBEF	 #FF6A6A6A	Presiona la barra de desplazamiento
 #FF442359	 #FF442359	 #FFFD4BF	Botones de ficha de documento provisional, seleccionados o en segundo plano: en segundo plano de mouse hacia abajo

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FF68217A	 #FF68217A	 #FF3399FF	Pestaña de documento provisional: fondo activo y el borde
 #FFF5F5F5	 #FF3E3E42	 #FFE8E8EC	Barra de desplazamiento y flechas de la barra de desplazamiento: de forma predeterminada, deshabilitado, al mantener el mouse y fondos de presionar el mouse
 #FF1C97EA	 #FF1C97EA	 #FF1C97EA	Glifos flecha mantener el mouse en la barra de desplazamiento
 #FF007ACC	 #FF007ACC	 #FF007ACC	Las flechas de la barra de desplazamiento: glifo de mouse hacia abajo
 #FF888888	 #FF9E9E9E	 #FF888888	Control de posición de la barra de desplazamiento: fondo y el borde al mantener el mouse
 #FF6A6A6A	 #FFEFEBEF	 #FF6A6A6A	Control de posición de la barra de desplazamiento: fondo y el borde de presionar el mouse
 #FFDEDFE7	 #FF686868	 #FFD0D1D7	Control de posición de la barra de desplazamiento: predeterminado en segundo plano y el borde
 #FF0E70C0	 #FF0097FB	 #FF0E70C0	Glifo "Borrar" del control de búsqueda
 #FFC9DEF5	 #FF2D2D30	 #FFFDF4BF	Botón de lista desplegable del control de búsqueda: centrado en segundo plano
 #FFC9DEF5	 #72555555	 #FFFDF4BF	Buscar elementos de menú desplegable: fondo y el borde al mantener el mouse y del mouse hacia abajo
 #FFC9DEF5	 #FF3E3E40	 #FFFFFFCF4	Selecciona los iconos: en segundo plano al mantener el mouse

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FF3399FF	 #FF3399FF	 #FF3399FF	Los elementos seleccionados: fondos o bordes
 #FF9B4F96	 #FF9B4F96	 #FFFFCF4	Selecciona, centra los botones de la ficha de documento provisional: en segundo plano al mantener el mouse
 #FFB7B9C5	 #FF1B1B1C	 #FFDF4BF	Seleccionado, el documento no enfocado y botones de pestaña provisional: en segundo plano de mouse hacia abajo
 #FFE6E7ED	 #FF555555	 #FFFFCF4	Seleccionado, el documento no enfocado y pestañas provisionales: en segundo plano al mantener el mouse
 #FFE6E7ED	 #FF555555	 #FE5C365	Seleccionado, el documento no enfocado y pestañas provisionales: borde al mantener el mouse
 #FFCCEDB	 #FF3F3F46	 #FFCCEDB	Fondo de la vista de árbol seleccionada, sin foco
 #FE0E3E6	 #FF333337	 #FFCCEDB	Separadores: grupo de menús, botón de expansión y sección de menú desplegable del control de búsqueda, la barra de menús
 #FF717171	 #FF999999	 #FF717171	Algunas instancias de texto sin foco
 #FFFFFFF	 #FFFFFFF	 #FFFFFFF	Barra de estado, seleccionado y texto activo
 #FE51400	 #FE51400	 #FE51400	Error grave
 #FF999999	 #FF46464A	 #FF4D6082	Controlador de arrastre la barra de título

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FF525252	 #FF999999	 #FF000000	Texto de la barra de título de ventana principal y las ventanas de documento flotante
 #FFEEEEF2	 #FF2D2D30	 #FFD6DBE9	Barra de título, barra de menús y fondos de la barra de comandos
 #FFFFFFF5	 #FFFFFFF5	 #FF000000	Botones de la ficha documentos y ventanas de herramientas: de forma predeterminada, al mantener el mouse y del mouse hacia abajo de glifos
 #FF0E6198	 #FF0E6198	 #FFFFE8A6	Ventana de herramientas y botones de la ventana de documento seleccionado: fondo y el borde de presionar el mouse
 #FFF5F5F5	 #FF252526	 #FFFFFFF5	Fondos de la ventana de herramienta
 #FFCCCEDB	 #FF3F3F46	 #FF8E9BBC	Borde de ventana de herramientas
 #FF0E70C0	 #FF0097FB	 #FF000000	Texto de ficha seleccionada de ventana de herramientas
 #FF007ACC	 #FF007ACC	 #FFFFF29D	Las barras de título de ventana de herramientas y fichas de la ventana de documento: centrado en segundo plano y el borde
 #FFC9DEF5	 #FF3E3E40	 #FF4B5C74	Ficha no seleccionada de ventana de herramienta: fondo predeterminado y mantener el mouse
 #FFEEEEF2	 #FF2D2D30	 #FF4B5C74	Ficha no seleccionada de ventana de herramienta: borde predeterminado
 #FFC9DEF5	 #FF3E3E40	 #FF4B5C74	Ficha no seleccionada de ventana de herramienta: borde al mantener el mouse

CLARO	OSCURO	AZUL	SE UTILIZA PARA...
 #FFCCCEDB	 #FF3F3F46	 #FF4B5C74	Ficha no seleccionada de ventana de herramienta: separador
 #FFEEEEF2	 #FF2D2D30	 #FFCFD6E5	Fondo de barra de herramientas
 #FFEEEEF2	 #FF2D2D30	 #FFDCE0EC	Borde de la barra de herramientas
 #FFEEEEF2	 #FF2D2D30	 #FFDCE0EC	En segundo plano de las opciones de barra de herramientas
 #FF007ACC	 #FF007ACC	 #FF007ACC	Vista de árbol: valor predeterminado o un glifo seleccionado, no enfocado al mantener el mouse
 #FFEEEEF2	 #FF2D2D30	 #FF4D6082	Fondo de barra de título de ventana de herramientas, pestaña de vista previa y pestaña del documento sin foco
 #FFCCCEDB	 #FF3F3F46	 #FF4D6082	Borde de barra de título de ventana de herramientas, pestaña de vista previa y pestaña del documento sin foco
 #FFF7F7F9	 #FF393939	 #FFFFCF4	Botones de la ventana de herramientas no enfocado: en segundo plano al mantener el mouse
 #FFEEEEF2	 #FF2D2D30	 #FF364E6F	Fondo de ficha de ventana de documento no seleccionado
 #FFB064AB	 #FFB064AB	 #FFFFCF4	Sin seleccionar botones de la ficha de documento provisional: en segundo plano al mantener el mouse
 #FF68217A	 #FFFFFFF	 #FF68217A	Logotipo de Visual Studio

Imágenes e iconos para Visual Studio

06/04/2020 • 31 minutes to read • [Edit Online](#)

Uso de imágenes en Visual Studio

Antes de crear ilustraciones, considere la posibilidad de usar más de 1.000 imágenes en la biblioteca de imágenes de [Visual Studio](#).

Tipos de imágenes

- **Iconos.** Imágenes pequeñas que aparecen en comandos, jerarquías, plantillas, etc. El tamaño de icono predeterminado utilizado en Visual Studio es un PNG de 16x16. Los iconos generados por el servicio de imágenes generan automáticamente el formato XAML para la compatibilidad con HDPI.

NOTE

Mientras que las imágenes se utilizan en el sistema de menús, no debe crear un icono para cada comando. Consulte [Menús y comandos de Visual Studio](#) para ver si el comando debe obtener un icono.

- **Miniaturas.** Imágenes utilizadas en el área de vista previa de un cuadro de diálogo, como el cuadro de diálogo Nuevo proyecto.
- **Imágenes de cuadro de diálogo.** Imágenes que aparecen en cuadros de diálogo o asistentes, ya sea como gráficos descriptivos o indicadores de mensajes. Utilice con poca frecuencia y solo cuando sea necesario para ilustrar un concepto difícil o llamar la atención del usuario (alerta, advertencia).
- **Imágenes animadas.** Se utiliza en indicadores de progreso, barras de estado y cuadros de diálogo de operación.
- **Cursores.** Se utiliza para indicar si se permite una operación con el mouse, donde se puede quitar un objeto, etc.

Diseño de icono

Información general

Visual Studio usa iconos de estilo moderno, que tienen geometría limpia y un equilibrio 50/50 de positivo/negativo (luz/oscuridad) y usan metáforas directas y comprensibles. Los puntos de diseño de iconos cruciales se centran en la claridad, la simplificación y el contexto.

- **Claridad:** concéntrese en la metáfora central que da a un icono su significado e individualidad.
- **Simplificación:** reduzca el icono a su significado central - obtener el tema a través con sólo los elementos necesarios y sin lujos.
- **Contexto:** considerar todos los aspectos del papel de un icono durante el desarrollo del concepto, que es crucial a la hora de decidir qué elementos constituyen la metáfora central del icono.

Con los iconos, hay una serie de puntos de diseño para evitar:

- No use iconos que signifiquen elementos de interfaz de usuario, excepto cuando corresponda. Elija un enfoque más abstracto o simbólico cuando el elemento de interfaz de usuario no es común, evidente ni único.
- No utilice en exceso elementos comunes como documentos, carpetas, flechas y la lupa. Utilice estos

elementos sólo cuando sea esencial para el significado del icono. Por ejemplo, la lupa orientada a la derecha debe indicar solo Buscar, Examinar y Buscar.

- Aunque algunos elementos de icono heredados mantienen el uso de la perspectiva, no cree nuevos iconos con perspectiva a menos que el elemento carezca de claridad sin él.
- No abarques demasiada información en un icono. Una imagen simple que se puede reconocer o aprender fácilmente como un símbolo reconocible es mucho más útil que una imagen demasiado compleja. Un icono no puede contar toda la historia.

Creación de iconos

Desarrollo de conceptos

Visual Studio tiene dentro de su interfaz de usuario una amplia variedad de tipos de icono. Considere cuidadosamente el tipo de icono durante el desarrollo. No utilice objetos de interfaz de usuario poco claros o poco comunes para los elementos de icono. Opte por el símbolo en estos casos, como con el icono de etiqueta inteligente. Tenga en cuenta que el significado de la etiqueta abstracta de la izquierda es más obvio que la vaga versión basada en la interfaz de usuario de la derecha:

Uso correcto de imágenes simbólicas	Uso incorrecto de imágenes simbólicas
	

Hay casos en los que los elementos de interfaz de usuario estándar y fácilmente reconocibles funcionan bien para los iconos. Agregar ventana es uno de estos ejemplos:

Elemento de interfaz de usuario correcto en un icono	Elemento de interfaz de usuario incorrecto en un icono
	

No utilice un documento como elemento base a menos que sea esencial para el significado del icono. Sin el elemento de documento en Agregar documento (abajo) se pierde el significado, mientras que con Actualizar el elemento de documento no es necesario comunicar el significado.

Uso correcto del icono del documento	Uso incorrecto del icono del documento
	

El concepto de "show" debe estar representado por el icono que mejor ilustra lo que se muestra, como con el ejemplo Mostrar todos los archivos. Una metáfora de lente se puede utilizar para indicar el concepto de "vista" si es necesario, como con el ejemplo de la vista de recursos.

"Espectáculo"	"Vista"
	

El icono de lupa orientado a la derecha solo debe representar Buscar, Buscar y Examinar. La variante orientada a la izquierda con el signo más o signo menos debe representar solo acercar/alejar.

"Buscar"	"Zoom"
	

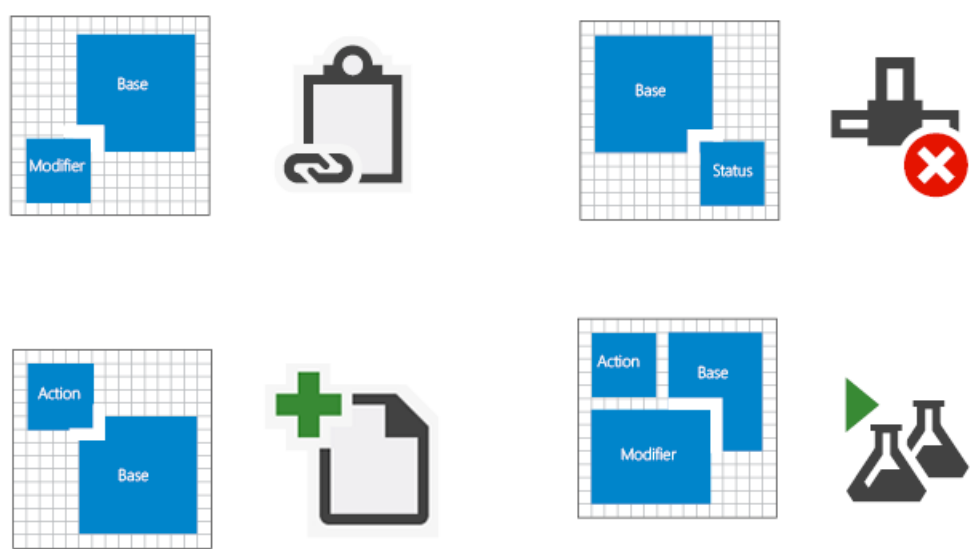
En las vistas de árbol, no utilice el icono de carpeta ni un modificador. Cuando esté disponible, utilice solo el modificador.

Iconos de vista de árbol correctos	Iconos de vista de árbol incorrectos
  icono Corregir vista de árbol (2)	  de vista de árbol incorrecto (1)Icono de vista de árbol incorrecto (2)

Detalles de estilo

Diseño

Elementos de pila como se muestra para los iconos estándar de 16x16:



Los elementos de notificación de estado se utilizan mejor como iconos independientes. Sin embargo, hay contextos en los que se debe apilar una notificación en el elemento base, como con el icono Tarea completada:

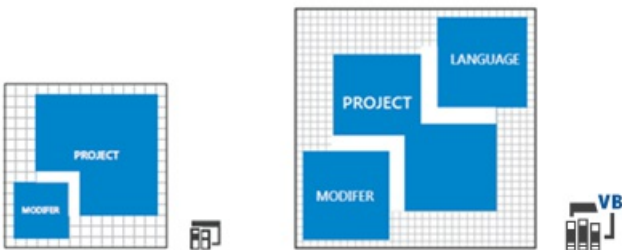


Iconos de notificación independientes



Icono Tarea completada

Los iconos de proyecto suelen ser archivos .ico que contienen varios tamaños. La mayoría de los iconos 16x16 contienen los mismos elementos. Las versiones 32x32 tienen más detalles, incluido el tipo de proyecto cuando corresponda.



Iconos del proyecto de la biblioteca de control de Windows de VB, 16x16 y 32x32

Centra un icono dentro de su marco de píxeles. Si eso no es posible, alinee el icono con la parte superior y/o derecha del marco.



Icono centrado en el fotograma de píxeles



Icono alineado a la parte superior derecha del marco

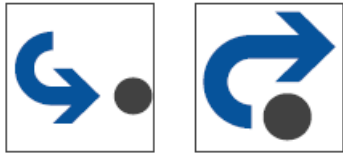


Icono centrado y alineado a la parte superior del marco

Para lograr una alineación y un equilibrio ideales, evite obstruir el elemento base del icono con glifos de acción. Coloque el glifo cerca de la parte superior izquierda del elemento base. Al agregar un elemento adicional, tenga en cuenta la alineación y el equilibrio del icono.

Alineación y equilibrio correctos	Alineación y equilibrio incorrectos
	

Asegúrese de la paridad de tamaño para los iconos que comparten elementos y se utilizan en conjuntos. Tenga en cuenta que en el emparejamiento incorrecto, el círculo y la flecha están sobredimensionados y no coinciden.

Paridad de tamaño correcta	Paridad de tamaño incorrecta
	

Utilice grosores visuales y de línea coherentes. Evalúe cómo se compara el icono que está creando con otros iconos mediante una comparación en paralelo. Nunca utilice toda la trama de 16x16, utilice 15x15 o más pequeño. La relación negativo-positivo (oscuro-claro) debe ser 50/50.

Relación negativa-positiva correcta	Relación negativa-positiva incorrecta
	

Utilice formas simples y comparables y ángulos complementarios para construir sus elementos sin sacrificar la integridad de los elementos. Utilice ángulos de 45o o 90o siempre que sea posible.



Perspectiva

Mantenga el icono claro y comprensible. Utilice la perspectiva y una fuente de luz solo cuando sea necesario. Aunque se debe evitar el uso de perspectiva en elementos de icono, algunos elementos son irreconocibles sin él. En tales casos, una perspectiva estilizada comunica la claridad del elemento.



Perspectiva de 3 puntos



Perspectiva de 1 punto

La mayoría de los elementos deben estar orientados o en ángulo a la derecha:



Utilice fuentes de luz solo cuando añada la claridad necesaria a un objeto.

Fuente de luz correcta	Fuente de luz incorrecta

Utilice contornos solo para mejorar la legibilidad o para comunicar mejor la metáfora. El balance negativo-positivo (luz oscura) debe ser 50/50.

Uso correcto de esquemas	Uso incorrecto de contornos

Tipos de icono

Los iconos de shell y barra de **comandos** constan de no más de tres de los siguientes elementos: una base, un modificador, una acción o un estado.



Ejemplos de iconos de shell y barra de comandos

Los iconos de la barra de **comandos** de la ventana de herramientas constan de no más de tres de los siguientes elementos: una base, un modificador, una acción o un estado.



Ejemplos de iconos de la barra de comandos de la ventana de herramientas

Los iconos de **desambiguator** de vista de árbol constan de no más de tres de los siguientes elementos: una base, un modificador, una acción o un estado.



Ejemplos de iconos de desambiguadores de vista de árbol

Los iconos de taxonomía de **valores basados** en estado existen en los siguientes estados: activo, deshabilitado activo e inactivo deshabilitado.



Ejemplos de iconos de taxonomía de valor basados en el estado

Los iconos de **IntelliSense** constan de no más de tres de los siguientes elementos: una base, un modificador y un estado.



Ejemplos de iconos de IntelliSense

Los iconos de **proyecto pequeños (16x16)** no deben tener más de dos elementos: una base y un modificador.

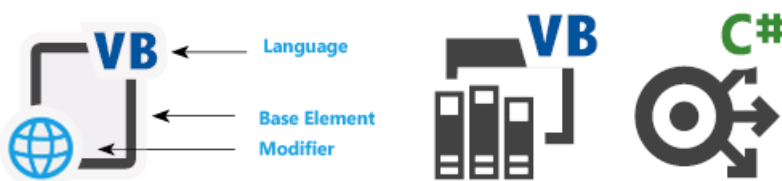


icono de

icono de proyecto de

Ejemplos de iconos de proyectos pequeños (16x16)

Los iconos de **proyecto grandes (32x32)** constan de no más de cuatro de los siguientes elementos: una base, uno a dos modificadores y una superposición de idioma.



Ejemplos de iconos de proyecto grandes (32x32)

Detalles de la producción

Todos los nuevos elementos de interfaz de usuario se deben crear con Windows Presentation Foundation (WPF) y todos los iconos nuevos para WPF deben estar en formato PNG de 32 bits. El PNG de 24 bits es un formato heredado que no admite transparencia y, por lo tanto, no se recomienda para iconos.

Guarde la resolución a 96 DPI.

Tipos de archivo

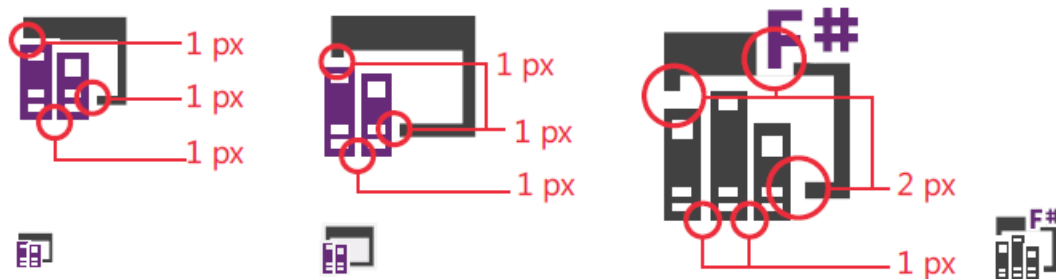
- **PNG de 32 bits:** el formato preferido para los iconos. Formato de archivo de compresión de datos sin pérdidas que puede almacenar una sola imagen ráster (píxel). Los archivos PNG de 32 bits admiten transparencia de canal alfa, corrección gamma y entrelazado.

- **BMP de 32 bits:** para controles que no son DEWPF. También llamado XP o de color alto, BMP de 32 bits es un formato de imagen RGB/A, una imagen de color verdadero con una transparencia de canal alfa. El canal alfa es una capa de transparencia designada en Adobe Photoshop que, a continuación, se guarda dentro del mapa de bits como un canal de color adicional (cuarto). Se añade un fondo negro durante la producción de ilustraciones a todos los archivos BMP de 32 bits para proporcionar una indicación visual rápida sobre la profundidad de color. Este fondo negro representa el área que se va a enmascarar en la interfaz de usuario.
- **ICO de 32 bits:** para iconos de proyecto y Agregar elemento. Todos los archivos ICO son de color verdadero de 32 bits con transparencia de canal alfa (RGB/A). Dado que los archivos ICO pueden almacenar varios tamaños y profundidades de color, los iconos de Vista suelen estar en un formato ICO que contiene tamaños de imagen de 16x16, 32x32, 48x48 y 256x256. Para que se muestren correctamente en el Explorador de Windows, los archivos ICO deben guardarse en profundidades de color de 24 bits y 8 bits para cada tamaño de imagen.
- **XAML:** para superficies de diseño y adornos de Windows. Los iconos XAML son archivos de imagen basados en vectores que admiten escalado, rotación, archivado y transparencia. No son comunes en Visual Studio hoy en día, pero se están volviendo más populares debido a su flexibilidad.
- **SVG**
- **BMP de 24 bits:** para la barra de comandos de Visual Studio. Un formato de imagen RGB de color verdadero, BMP de 24 bits es una convención de icono que crea una capa de transparencia mediante el uso de magenta (R-255, G-0, B-255) como una clave de color para una capa de transparencia knock-out. En un BMP de 24 bits, todas las superficies magenta se muestran utilizando el color de fondo.
- **GIF de 24 bits:** para la barra de comandos de Visual Studio. Un formato de imagen RGB de color verdadero que admite transparencia. Los archivos GIF se utilizan a menudo en ilustraciones del asistente y animaciones GIF.

Construcción de icono

El tamaño de icono más pequeño de Visual Studio es 16x16. El más grande en uso común es 32x32. Tenga en cuenta no llenar todo el marco de 16x16, 24x24 o 32x32 al diseñar un icono. La construcción de iconos legibles y uniformes es esencial para el reconocimiento del usuario. Adherirse a los siguientes puntos al crear iconos.

- Los iconos deben ser claros, comprensibles y coherentes.
- Es mejor utilizar los elementos de notificación de estado como iconos únicos y no apilarlos encima de un elemento base de icono. En determinados contextos, la interfaz de usuario puede requerir que el elemento de estado se empareje con un elemento base.
- Los iconos de proyecto suelen ser archivos .ico que contienen varios tamaños. Solo se actualizan los iconos 16x16, 24x24 y 32x32. La mayoría de los iconos 16x16 y 24x24 contendrán los mismos elementos. Los iconos de 32x32 contienen más detalles, incluido el tipo de idioma del proyecto cuando corresponda.
- Para los iconos de 32x32, los elementos base generalmente tienen un grosor de línea de 2 píxeles. Se puede utilizar un grosor de línea de 1 o 2 píxeles para los elementos de detalle. Utilice su mejor criterio para determinar cuál es más adecuado.
- Tener al menos un espaciado de 1 píxel entre los elementos para los iconos 16x16 y 24x24. Para iconos de 32x32, utilice el espaciado de 2 píxeles entre los elementos y entre el modificador y el elemento base.



Espaciado de elementos para iconos de tamaño 16x16, 24x24 y 32x32

Color y accesibilidad

Las directrices de cumplimiento de Visual Studio requieren que todos los iconos del producto superen los requisitos de accesibilidad para el color y el contraste. Esto se logra a través de la inversión de icono, y cuando usted está diseñando, usted debe ser consciente de que se invertirán mediante programación en el producto.

Para obtener más información sobre el uso del color en los iconos de Visual Studio, vea [Uso del color en imágenes](#).

Uso del color en imágenes

Información general

Los iconos de Visual Studio son principalmente monocromáticos. El color está reservado para transmitir información específica y nunca para la decoración. Se utiliza el color:

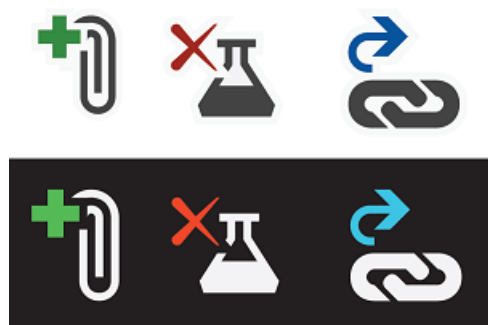
- para indicar una acción
- para alertar al usuario de una notificación de estado
- designar afiliación al idioma
- para diferenciar elementos dentro de IntelliSense

Accesibilidad

Las directrices de cumplimiento de Visual Studio requieren que todos los iconos registrados en el producto superen los requisitos de accesibilidad para el color y el contraste. Los colores de la paleta de idiomas visuales se han probado y cumplen estos requisitos.

Inversión de color para temas oscuros

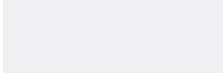
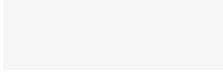
Para que los iconos aparezcan con la relación de contraste correcta en el tema oscuro de Visual Studio, se aplica una inversión mediante programación. Los colores de esta guía se han elegido en parte para que se inviertan correctamente. Restringe el uso del color a esta paleta, o obtendrá resultados impredecibles cuando se aplique la inversión.



Ejemplos de iconos que han invertido sus colores

Paleta base

Todos los iconos estándar contienen tres colores base. Los iconos no contienen degradados ni sombras paralelas, con una o dos excepciones para los iconos de herramientas 3D.

USO	NOMBRE	VALOR (TEMA DE LUZ)	SWATCH	EJEMPLO
Fondo/Oscuro	VS BG	424242 / 66,66,66		
Primer plano/luz	VS FG	F0EFF1 / 240.239.241		
Esquema	VS Out	F6F6F6 / 246.246.246		

Además de los colores base, cada icono puede contener un color adicional de la paleta extendida.

Paleta extendida

Modificadores de acción

Los cuatro colores siguientes indican los tipos de acciones requeridas por los modificadores de acción:

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
Positive	VS Acción Verde	388A34 / 56,138,52	
Negative	VS Acción Rojo	A1260D / 161,38,13	
Neutra	VS Action Blue	00539C / 0,83,156	
Crear/Nuevo	VS Acción Naranja	C27D1A / 194.156,26	

Ejemplos

El verde se utiliza para modificadores de acción positiva como "Agregar", "Ejecutar", "Reproducir" y "Validar".

			
Ejecute	Ejecutar consulta	Reproducir todos los pasos	Añadir control

El rojo se utiliza para modificadores de acción negativos como "Eliminar", "Detener", "Cancelar" y "Cerrar."

			
Eliminar relación	Eliminar columna	Detener consulta	Conexión sin conexión

El azul se aplica a los modificadores de acción neutral más comúnmente representados como flechas, como "Abrir", "Siguiente", "Anterior", "Importar" y "Exportar."

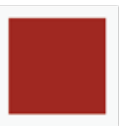
			
Ir al campo	Check-In por lotes	Editor de direcciones	Editor de la Asociación

El oro oscuro se utiliza principalmente para el modificador "Nuevo".

			
Nuevo proyecto	Crear nuevo gráfico	Nueva prueba unitaria	Nuevo elemento de lista

Casos especiales

En casos especiales, un modificador de acción de color se puede utilizar de forma independiente como un icono independiente. El color utilizado para el icono refleja las acciones a las que está asociado el icono. Este uso se limita a un pequeño subconjunto de iconos, incluyendo:

				
Ejecute	Stop	Eliminar	Save	Navegar hacia atrás

Paleta de jerarquía de código

Carpeta

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH	EJEMPLO
Carpetas	Carpeta	DCB67A / 220,182,122		

Idiomas de Visual Studio

Cada uno de los lenguajes comunes o plataformas disponibles en Visual Studio tiene un color asociado. Estos colores se utilizan en el icono base o en los modificadores de idioma que aparecen en la esquina superior derecha de los iconos compuestos.

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
ASP, HTML, WPF	ASP HTML WPF Azul	0095D7 / 0,149,215	
C++	CPP Purple	9B4F96 / 155,79,150	
C#	CS Verde (VS Action Green)	388A34 / 56,138,52	
CSS	CSS Rojo	BD1E2D / 189,30,45	
F#	FS Purple	672878 / 103,40,120	
JavaScript	JS Orange	F16421 / 241,100,33	
VB	AZUL VB (azul de acción VS)	00539C / 0,83,156	
TypeScript	TS Orange	E04C06 / 224,76,6	
Python	PY Green	879636 / 135,150,54	

Ejemplos de iconos con modificadores de idioma

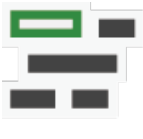
					
VB	C#	C++	F#	JavaScript	Python
					
HTML	WPF	ASP	CSS	TypeScript	

IntelliSense

Los iconos de IntelliSense utilizan una paleta de colores exclusiva. Estos colores se utilizan para ayudar a los usuarios a distinguir rápidamente entre los diferentes elementos de la lista emergente de IntelliSense.

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
Clase, Evento	VS Acción Naranja	C27D1A / 194.125,26	
Método de extensión, método, módulo, delegado	VS Acción Púrpura	652D90 / 101,45,144	
Campo, Elemento de enumeración, Macro, Estructura, Tipo de valor de unión, Operador, Interfaz	VS Action Blue	00539C / 0,83,156	
Object	VS Acción Verde	388A34 / 56,138,52	
Constante, Excepción, Elemento de enumeración, Mapa, Elemento de mapa, Espacio de nombres, Plantilla, Definición de tipo	Fondo (VS BG)	424242 / 66,66,66	

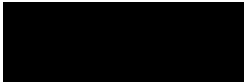
Ejemplos de iconos de IntelliSense

				
Clase	Evento privado	Delegar	Amigo del método	Campo
				
Artículo de enum protegido	Object	Plantilla	Acceso directo a excepciones	

Notificaciones

Las notificaciones en Visual Studio se usan para indicar el estado. La paleta de notificaciones utiliza los cuatro colores siguientes, así como las opciones de relleno de primer plano en blanco o negro, para definir notificaciones con los siguientes niveles de estado.

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
Estado: neutral	Azul de notificación (AZUL VS)	1BA1E2 / 27.161.226	
Estado: positivo	Notificación verde (verde VS)	339933 / 51,153,51	

USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
Estado: negativo	Notificación roja (VS Rojo)	E51400 / 229,20,0	
Estado: advertencia	Notificación amarilla (VS naranja)	FFCC00 / 255,204,0	
Relleno de primer plano	Notificación Negro (Negro)	000000 / 0,0,0	
Relleno de primer plano	Notificación blanco (blanco)	FFFFFF / 255.255.255	

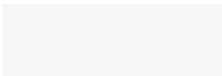
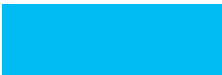
Ejemplos de iconos de notificación

			
Alerta	Advertencia	Operación completada	Stop

Visual Studio Online

En general, Visual Studio Online consta de características hospedadas en un explorador. El color varía en diferentes ambientes, pero el estilo sigue siendo el mismo.

GRUPO	USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
TFS	Información previa	TFSO BG	656565/ 101, 101, 101	
TFS	Esquema	TFSO OUT	FFFFFF / 255, 255, 255	
Napa	Información previa	Blanco	FFFFFF / 255, 255, 255	
Mónaco	Información previa	Blanco	FFFFFF / 255, 255, 255	
F12	Información previa	Blanco	FFFFFF / 255, 255, 255	
F12	Normal	F12 Grey_Primary	555555 / 85, 85, 85	

GRUPO	USO	NOMBRE	VALOR (TODOS LOS TEMAS)	SWATCH
F12	Al mantener el puntero	F12 Blue_Hover	2279BF / 34,121,191	
F12	Disabled	F12 LtGrey_Disabled	ABABAC / 171.171.172	
F12	Pase el ratón por el fondo	Hover bg	D9EBF7 / 217.235.247	
F12	Fondo presado	Presionado bg	B2D7F0 / 178.215.240	
F12	Esquema	VS OUT	F6F6F6 / 246.246.246	
F12	Information	Information	00BCF2 / 0,188,242	
F12	Advertencia	Advertencia	F28300 / 242.131,0	
F12	Error / Negativo	Error_Negative	E81123 / 232,17,35	
F12	Inicio / Positivo	Start_Positive	009E49 / 0,158,73	
F12	Tipo de rotura	Tipo de rotura	9B4F96 / 155,79,150	
F12	Marca del evento	Marca del evento	A51F00 / 165,31,0	
F12	Marca de usuario	Marca de usuario	F16220 / 241,98,32	

Ejemplos de iconos de Visual Studio Online

TFS ONLINE			
 Equipo en línea	 Information	 Historial	 Rama

NAPA			
 Contenido	 Correo de oficina	 SharePoint	 Panel de tareas
MÓNACO			
 Archivos	 Git	 Search	 Texto
F12			
 Pretty Code	 Advertencia	 Emulación	

Diccionario del lenguaje Visual

02/12/2019 • 26 minutes to read • [Edit Online](#)

Visual Studio usa un lenguaje visual para ilustrar los conceptos que son los bloques de creación de su iconografía y otras imágenes. Al diseñar nuevos iconos, utilice este diccionario para identificar los conceptos que desea transmitir en su propia imagen.

Instrucciones generales

- Estos conceptos visuales deben utilizarse de manera coherente con su definición de diccionario. De lo contrario, corre el riesgo de confundir a los usuarios, lo que esperan elementos comunes que tienen un significado coherente.
- Elija solo los conceptos más importantes o relevantes para su uso en los iconos, para que las imágenes son nítidas y fáciles de distinguir.
- Use solo uno o dos conceptos de cada imagen.
- Para obtener más estrategias para diseñar iconos, consulte [diseño de los iconos](#).

Saltar a la sección:

1. [Conceptos a Z](#)
2. [Productos y plataformas](#)
3. [Conceptos comunes con flechas](#)
4. [Indicadores de estado](#)

Conceptos a Z

A

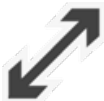
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Accesibilidad			
Acción		 Registro de acciones	El acción icono también puede utilizarse como una anotación para indicar que se realizará la acción de base mediante la configuración automática.
Actividad		 Actividad de código	
Actor			

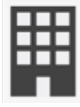
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Add		 Agregar todas las pruebas del controlador  Agregar dato adjunto  Agregar comportamiento	
Dirección			
Alert			
Canal alfa			
Animación		 Error de animación  Siguiendo animación	
Aplicación Proyecto		 Error de rol de aplicación  Proyecto C++ MFC	
Área		 Gráfico de áreas de spline  Serie de áreas apiladas	
Arrange		 Organizar formas  Organizar formas automáticamente	
Assign			

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Asociación		 Agregar asociación  Editar asociación	
Asymmetric			El asimétrica concepto está representado por la juxtaposition de imágenes opuesta (por ejemplo, claro/oscurο o izquierda o derecha).
Asociar		 Agregar dato adjunto  Abrir datos adjuntos	
Attribute Variable, parámetro		 Atributo de cuenta  Agregar parámetro  Variable de miembro	
Audio		 Silenciar audio  Reproducción de audio  Archivo de sonido	
Grabación de audio		 Silencio de registro  Voz y pantalla de registro	
Autosum			

B

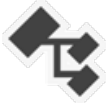
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Bidireccional			
Binary		 Iniciar binario de destino de instrumentación	
Marcador		 Proyecto de marcador  Eliminar marcador  Ir al marcador anterior	
Rama		 Rama cargado  Rama remota	
Brillo		 Reducir brillo  Aumentar brillo	
Examinar		 Examinar siguiente  Examinar anterior	
Bug		 Crear nuevo error	
Compilación		 Compilar selección  Instancia de compilación privada  Compilación creada por el usuario	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Builder		 Generador de cubos  Generador de dimensiones	
Business		 Plantilla de modelo de negocio  Origen de datos de objetos de negocios  Negocio en línea	
Button		 Botón Agregar  Botón de imagen  Botón vinculado	

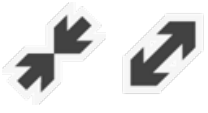
C

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Caché		 Cache OK  Actualización de caché	
Calendar		 Eje de fecha y hora  Calendario global	
Call		 Explorador de llamadas  Configuración del explorador de llamadas  Vista llamador o destinatario	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Cancelar Igual que "Borrar" y "Eliminar"		 Cancelar compilación	
Capture		 Capturar fotograma  Captura de pantalla completa  Empezar a capturar diagnósticos gráficos	
CD		 Unidad de CD	
Certificate		 Error de certificado  Advertencia de certificado	
Changeset		 Conjuntos de cambios de grupo  Nuevo conjunto de cambios	
Alternancia de elección		 Columna de campo de casilla de verificación  Lista de casillas	
Clase		 Agregar clase  Detalles de clase de ventana de herramientas	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Borrar Igual que "Cancelar" y "Eliminar"		 Borrar todos los puntos de interrupción  Eliminar marcador  Eliminar colección	
Cerrar Terminar		 Cerrar todo  Cerrar resultados  Terminar proceso	
Cloud		 Paquete en la nube  Servicio en la nube	
Código		 Prueba de IU codificada  Convertir a prueba web codificada	
Contraer		 Contraer todo  Volver al valor contraído	
Colección		 Eliminar colección  Colección de recuento	
Columna		 Columna de la propiedad AutoSize  Detalles de columna  Error de la columna	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Comentario Comentarios, anotaciones		 Agregar comentario  Llamada	
Comentario (código)		 quitar comentarios	
Compare		 Comparar datos  Comparar carpetas  Comparar informes de rendimiento	
Componente		 Agregar componente  Diagrama de componentes  Componente de Windows Phone RT	
Conectar		 Agregar conexión  Conectar el plan de pruebas  Conectarse al entorno	
Consola		 Prueba de consola	
Contrato (nombre)		 Error de contrato  Advertencia de contrato	El contrato (nombre) icono se utiliza para indicar un acuerdo entre la aplicación y el sistema operativo.

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Contrato o expandir			El contrato o expandir iconos se usan en situaciones de diseño para indicar que se ocultará el espacio entre los dos elementos.
Contraste			
Control		 Agregar control  Agregar control personalizado  Agregar control heredado	
Convertir		 Convertir rama en carpeta  Convertir partición  Convertir a prueba web codificada	
Copiar		 Copiar agregación  Copiar sitio Web	Duplicar y apilamiento el objeto que representa el copia acción.
Contador			
Crear		 Crear nuevo error  Crear nuevo gráfico  Crear nueva variable	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Cursor Puntero		 Inspeccionar el botón de modo  Herramienta de punto de conexión de conjunto	
Expresión personalizada Predicción		 Predicción de minería de datos	
Cortar			

D

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Tema oscuro		 En el tema oscuro	
Data		 Comparar ambas bases de datos  Especificación de auditoría de base de datos  Proyecto de base de datos	
Minería de datos		 Columna de minería de datos  Predicción de minería de datos  Estructura de minería de datos	
Delegate		 Invocar al delegado	

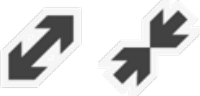
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Eliminar <i>Igual que "Cancelar" y "Clear"</i>		 Cancelar compilación  Eliminar columna  Eliminar conjunto de datos  Eliminar carpeta	
dependencia			
Implementar		 Extensiones de configuración de implementación  Configuraciones de implementación	
Diagram Flujo de trabajo		 Nuevo diagrama  Mostrar panel de diagrama	
Cuadro de diálogo		 Grupo de cuadro de diálogo de prueba  Probar cuadro de diálogo	
Dictionary		 Agregar nuevo diccionario  Agregar al diccionario  Borrar diccionario	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Dimensión Plan		 Vista de explorador de dimensiones  Vista Generador de dimensiones  Vista de traducción de dimensiones	
Pantalla Monitor		 Configuración de la visualización  Captura de pantalla completa	
Document Archivo		 Biblioteca de documentos  Descarga de archivos  Dar formato al documento	
Drive		 Unidad de CD	
Driver		 Plantilla de paquete de controladores  Explorador de grupo de pruebas de controladores  Plantilla de controlador de red NDIS	
Dynamic		 Control dinámico de ASP.NET  Sitio Web de datos dinámicos de C#  Validador dinámico	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Valor dinámico		 Copiar valor dinámico  Vaciar valor dinámico	

E

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Editar		 Editor de direcciones  Editar consulta  Editar relación	El editar anotación se enfrenta a izquierda o derecha según el aspecto que tiene el icono de base. Elija la orientación que quede mejor con el diseño general de los iconos.
Efectos <i>Igual que "Function", "Variable (global)"</i>			
vacío		 Página en blanco  Proyecto vacío de C++  F#proyecto de sitio Web vacío	
Enumerator		 Crear enumerador	
Entorno		 Conectarse al entorno  Crear entorno de biblioteca  Entorno de biblioteca	
Event Desencadenador		 Agregar evento  Registro de eventos	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Expanda		 Devolver valor expandido	
Expandir/contraer			El expandir/contraer iconos se usan en situaciones de diseño para indicar que se ocultará el espacio entre los dos elementos.
Export <i>Igual que "Abierto"</i>		 Exportar filtro  Exportar datos de informe  Exportar filtro de informe	

F

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Favorito Protegido, la clasificación		 Agregar a Favoritos  Calificación	
Campo		 Agregar campo  Editar campo	
Field (elemento de código)		 Vaya al campo  Nuevo campo	
Filtrar		 Autofiltro  Filtro del gráfico  Exportar filtro	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Finanzas		 Editor de divisas	
Flagged		 Marcar subproceso marcado  Marcar subproceso no marcado  Mostrar sólo los subprocesos marcados	
Flow			
Carpeta		 Biblioteca de documentos  Buscar en archivos  Carpeta abierta  Carpeta vinculada abierta	
Fuente		 Color de fuente  Tamaño de fuente  Serif	
Frame			
Friend			
Function Expresión <i>Igual que "Efectos" y "Variable (global)"</i>		 Advertencia de función  Función de la página WPF	

G

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Get Descargar		 Obtener la carpeta de descarga  Obtener el valor del diccionario  Obtener elemento de colección	
Graph Gráfico de barras		 Sesión de rendimiento HPC	
Gráficos (3D)			
Grid		 Control de cuadrícula de datos de activos  Aplicación de cuadrícula C++  Alternar la cuadrícula	
Diseño de cuadrícula web		 Elemento de cuadrícula  Elemento del divisor de cuadrícula	
Agrupación		 Grupo de cuadro de diálogo  Equipo  Máquinas virtuales	Apilar objetos múltiples uno sobre otro representa el concepto de agrupación o multiplicidad.

H

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Hierarchy		 Variable de jerarquía  Jerarquía de llamadas	
Historial			
Página principal		 Pantalla principal de LightSwitch	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Idea			
Image Activos, recursos		 Cargador de imágenes  Botón de imagen  Control de lista de imágenes	
Importante Atención, ruta de acceso activa		 Mostrar líneas activas  Zona activa	
Indizador		 Agregar indizador	
Herencia		 Agregar control heredado  Agregar formulario heredado  Control heredado	El concepto de derivado herencia representado por una esquina con puntos. Vea también "Nonsubstantive (oculto o una plantilla)" y "Virtual".

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Herencia (flecha)			El herencia (flecha) icono representa un control de cuadro de herramientas para una superficie de modelado.
Interface		 Implementación de interfaz	
Item		 Copiar elemento  Crear elemento de lista  Compruebe en el elemento	

K

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Key Permiso, Id.		 Nueva clave  Permiso  Obtener Id. de elemento actual	
Indicador clave de rendimiento (KPI)		 Eliminar KPI  Vista de explorador KPI  KPI con error	

L

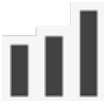
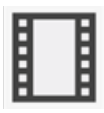
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Library		 Proyecto de biblioteca de clases de C++  Biblioteca de exportaciones  F#Biblioteca de controles Windows Forms	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Vínculo		 Convertir en hipervínculo  Crear tabla vinculada  Editar vínculo	
Bloqueo Private, permiso		 Bloquear eje X  Permisos de rama  cola privada	
Log		 Propiedades del catálogo  Registro de acciones  Conectar el plan de pruebas	
Loop		 Agregar o Insertar bucle condicional  Bucle condicional	

M

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Correo electrónico Mensaje		 Cola de mensajes  Seguimiento de mensajes recibidos  Visor de direcciones de correo electrónico	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Manifest		 Editar manifiesto  Manifiesto de administrador de la extensión  Nuevo manifiesto de aplicación	
Marcado		 Error de marcado  Etiqueta de marcado  Fragmento de código de evento o etiqueta  Coincide con la etiqueta  Editar etiqueta	El marcado concepto también se utiliza como un elemento en varios conceptos compuestos.
Marcado: HTML (web)		 Archivo HTML	
Marcado: XAML (WPF)		 aplicación WPF  Aplicación de explorador WPF VB  Biblioteca de controles WPF	
Marcado: XML		 Volver a cargar XML  Archivo XML	
Página maestra		 Desde el nuevo patrón de archivos	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Medida		 Medida calcula  Expresión de medida	
Media Película		 Mi colección de películas	
Miembro Editor de materiales, especular para 3D		 Agregar miembro  Miembro calculado  Variable de miembro	
Memoria (chip)		 Configuración de memoria  Matriz de memoria  Archivo de paginación	
Memoria Ventana de herramientas de memoria			
Merge		 Combinar automáticamente todos los  Combinar los cambios con la herramienta  Referencia de módulo de combinación	
Mensaje Chat			

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Método		 Extraer método  Invoke (método)  Vista Generador de cubos	
Teléfono móvil		 Servicios móviles  Componentes de juegos para móviles  Proyecto de web móvil de VB	
Módulo		 Proyecto de módulo de combinación  Agregar módulo  Exclusión de módulo de combinación	
Mover			

N

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Espacio de nombres		 Quitar mediante el espacio de nombres  Orden de acuerdo con el espacio de nombres  Agregar mediante espacio de nombres	

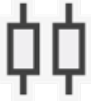
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Network Conexión		 Plantilla de paquete de controladores de red  Conexión sin conexión  Conexión desconocida	
nodo Forma de cruz			
Nonsubstantive (oculto o una plantilla) <i>Igual que "Virtuales"</i>		 Campo oculto  Carpeta oculta  Plantilla dinámica	Un contorno punteado de un objeto indica que es oculto o con plantilla .
Nota:			

O

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Abrir <i>Igual que "Export"</i>		 Abrir datos adjuntos  Abrir resultado de comparación  Cuadro de diálogo Abrir archivo	
Salida			

P

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Paquete		 Paquete en la nube  Crear paquete  Plantilla de paquete de controladores	
Parallel		 Paralelo para cada uno	
partición		 Esquema de partición  Nueva partición  Partition (función)	
Partes		 Elemento asignado  Elemento sin asignar	
Rendimiento			
PIN		 Desanclar	
Planeación			El Planning icono se utiliza para indicar la parte de un flujo de trabajo en una superficie de diseño.
Directiva		 Caché de directiva de Visual Studio  Lenguaje de descripción de directiva de Visual Studio	
Pantalla completa		 ENTRAR	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Power		 Fuente de alimentación ininterrumpida	
Anterior		 Marcador anterior  Buscar anterior	
Imprimir		 Impresión directa  Cuadro de diálogo Imprimir  Control de documento de impresión	
Procedimiento Ventana interactiva		 Ventana interactiva de Python  Advertencia de procedimiento  Procedimiento almacenado	
Propiedad Configuración de las configuraciones		 Agregar propiedad  Propiedad extendida—advertencia  Nueva propiedad	

Q

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Consulta		 Editar consulta  Extensor de la consulta  Consulta rápida	

R

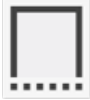
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Record		 Grabar pantalla  Vaya a la sesión de prueba registrada	
Redo			
Referencia		 Referencia errónea  Ir a referencia  Referencia de módulo de combinación	
Actualizar		 Actualizar el servicio en la nube  Actualización de caché	
Relación Bifurcar		 Definir relación  Agregar relación  Elementos primarios y secundarios	
Remote		 Rama remota  Escritorio remoto	
Remove			
Cambiar nombre			

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Reorder		 Reordenar parámetros	
Reparación			
Informe		 Informe de cuadro de diálogo de archivo  Informe de rendimiento de instrumentación  Asistente para proyectos de informe	
Rule		 Nueva regla de extracción  Nueva regla de umbral  Nueva regla de validación	
Ruler Guía de medida		 Modo de medición activado  Unidades de medida	

S

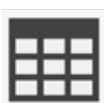
CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Guardar		 Guardar todo  Guardar y cerrar  Guardar el control de cuadro de diálogo de archivo	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Esquema		 Crear esquema  Esquema de base de datos  Esquema XML	
Script		 Exportar a script  Generar script de cambio	
Búsqueda Buscar, buscar		 Buscar en archivos  Resultados de la búsqueda	
Servidor (local)		 Agregar SQL server  Rol de servidor de inicio de sesión  Configurar equipos	
Servidor (remoto)		 Agregar servidor  Ejecutar pruebas de servidor	
Configuración Servicios de proceso		 Configuración del explorador de llamadas  Configuración de pruebas  Error del proceso de preparación	
Compartir			

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Shortcut			
Fragmento de código		 Fragmento revisado  Fragmento de código de evento o etiqueta	
Control de código fuente		 Cambiar control de código fuente	
Iniciar Run		 Ejecutar la generación de perfiles de simultaneidad  Se inició el servidor  Inicie el diagnóstico de gráficos	
Pasos Fase de fase		 Crear pasos compartidos  Insertar pasos compartidos  Reproducir todos los pasos	
estructura Struct			
Hoja de estilos		 Nueva hoja de estilos  Adjuntar hoja de estilos	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Sincronizar Actualizar		 JavaScript actualizado  Actualiza los elementos de base de datos  Sincronización	

T

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Table		 Advertencia de tabla  Colección de la tabla de almacenamiento en la nube	
Tablet		 Configuración de Tablet PC  Advertencia de tableta	
Tag Sistema de etiquetado			
Task		 Tarea vinculada  Lista de tareas	
Equipo		 Nuevo proyecto  Agregar proyecto  Team Foundation Server	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Texto		 Imagen y texto  Cuadro de texto enriquecido  Texto—primera sangría activada	
Subproceso		 Muchos subprocesos detenidos  Ocultar todos salvo los subprocesos seleccionados  Ocultar subprocesos seleccionados	
Tiempo Pendiente		 Hora hacia arriba o hacia abajo  Selector de hora activado	
Temporizador		 Fin de tiempo  Inicio de tiempo	
Toggle		 Activar o desactivar todos los puntos de interrupción  Alternar	
Cuadro de herramientas		 Control de cuadro de herramientas WinForm  Control de cuadro de herramientas WPF	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Undo Revertir, restaurar		 Restauración de la imagen  Deshacer la desprotección del elemento	
Arriba Cargar		 Parámetro de carga de archivo  Sube un nivel	
User Role, Profile		 Agregar usuario  Agregar control de usuario web  Tipos definidos por el usuario SQL	
Interfaz de usuario IU			

V

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Variable (local)		 Nueva variable  variable local  Variable SQL	
Variable (global) Igual que "Efectos" y "Function"		 Propiedades de la variable  Variable global  Eliminar variable	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Vista		 Vista avanzada  Vista del origen de datos	
Vista Auditoría		 Vista de recursos  Ver en explorador  Vista de rayos x	
Virtual <i>Igual que "Nonsubstantive (oculto o una plantilla)"</i>		 Entorno virtual	
Visible		 Ocultar u ocultar	

W

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Watch		 Monitor de sistema de archivos	
Web		 Publicar en web  Descargar configuración web  Nuevo sitio Web	
Wizard		 Asistente para capas de mapa  Asistente para clases MFC  Asistente para mapas	

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Elemento de trabajo		 Ir al elemento de trabajo  Consulta de elementos de trabajo	

Y

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Yield			

Z

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS	NOTAS
Zoom		 Acercar  Alejar	

Productos y plataformas

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
App Insights		
ATL		 Componente de páginas active server ATL  Clase de control ATL  Servidor ATL
Blend		 Aplicación SketchFlow de Blend  Aplicación de Windows Phone de Blend  Aplicación Pivot Blend Windows Phone

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
CoffeeScript		
Informes Crystal		 Informes Crystal
GitHub		
LightSwitch		 Biblioteca de extensiones de C# LightSwitch
Mercury		
Microsoft Foundation Class MFC		 Proyecto de control ActiveX de MFC  Proyecto de aplicación MFC  Proyecto de DLL de MFC de C++
WCF		 Data services de WCF  Proyecto WCF#  Proyecto WCF VB
WF		 C# del proyecto WF  Proyecto WF VB

Conceptos comunes con flechas

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Línea de asociación		

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Rama		
Cámara orbital		
conector		
Contexto actual		
Ubicación actual		
Fila actual		
Cursor		
Herramienta de delegación		
dependencia		
Selección directa		
Abajo Descargar		 Descargar configuración web  Carpeta de descarga  Orden ascendente

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Dynamic		 Control dinámico de ASP.NET  Validador dinámico  Sitio Web de datos dinámicos de C#
Expanda		
Export		 Exportar filtro  Exportar filtro de informe (datos)  Exportar filtro de informe
(Glifo) de		
(Glifo) presionada		
(Glifo) siguiente		
(Glifo) anterior		
Selector de archivos contraído (glifo)		
Selector de archivo expandido (glifo)		
Import		 Importar elemento de catálogo  Filtro de importación

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Herencia		
Puntero de instrucción		 Puntero actual  Puntero histórico  Puntero de devolución de llamada
Merge		 Combinar automáticamente todos los  Combinar los cambios con la herramienta  Referencia de módulo de combinación
Mover al final		
Mover arriba		
Navegar hacia atrás		 Parte posterior del hipervínculo
Navegar hacia delante		 Hiperínculo hacia delante
Siguiente Generar, vaya a		 Error siguiente  Generar tabla  Ir a código fuente

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Abrir		 Abrir datos adjuntos  Abrir resultado de comparación  Cuadro de diálogo Abrir archivo
Anterior		 Marcador anterior  Buscar anterior
Redo		
Reorder		 Reordenar parámetros
Sync		 Contenido de sincronización
Undo Revertir, restaurar		 Restauración de la imagen  quitar comentarios
Arriba Cargar		 Parámetro de carga de archivo  Sube un nivel

Indicadores de estado

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Estado de la acción: en pausa		 Servidor en pausa

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Estado de la acción: reproducción en curso, ejecute		 Se inició el servidor  No ejecutar nunca
Estado de la acción: detenido		 Servidor detenido
Alert		 Tabla vinculada con la indicación de la pestaña de error
Punto de interrupción: avanzado deshabilitado		
Punto de interrupción: avanzado habilitado		
Punto de interrupción: enlazado		
Punto de interrupción: deshabilitado		
Punto de interrupción: habilitado		 ventana Puntos de interrupción  Borrar todos los puntos de interrupción  Activar o desactivar todos los puntos de interrupción
Punto de interrupción: asignado deshabilitado		
Punto de interrupción: asignado habilitado		

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Completado y correcto		 Tarea completada  Proceso terminado  Servicio Aceptar en la nube
Error		 Error de punto de interrupción  Error de certificado  restricción CHECK
Ayuda No concluyentes		 Archivo de Ayuda de MS compilado  Archivo del índice de ayuda de MS
Información		 Relación de información rígida  Información del sistema
Información sobre herramientas de información		
Invalid		
No Silenciar, no es aplicable		 Silencio de micrófono  Silenciar audio
Remove Sin conexión, crítico		 Conexión sin conexión  Carpeta sin conexión  Detener consulta

CONCEPTO	ICONO PRINCIPAL	OTROS EJEMPLOS
Obligatoria		 Requisito del conjunto de pruebas
Alerta del escudo de seguridad		
Escudo de seguridad completado y correcto		
Escudo de seguridad crítica		
Punto de seguimiento: avanzado deshabilitado		
Punto de seguimiento: avanzado habilitado		
TracePoint: bound		
TracePoint: disabled		 Punto de seguimiento con error  Punto de seguimiento con advertencia
TracePoint: enabled		
Advertencia		 Rol de aplicación que faltan  No se encontró el ensamblado  Clave no encontrada

Menús y comandos para Visual Studio

06/04/2020 • 30 minutes to read • [Edit Online](#)

Uso de comandos

Información general

A diferencia de Microsoft Office, que es un conjunto que comprende muchos productos independientes, Visual Studio contiene muchos productos que cada uno aporta sus conjuntos de comandos al IDE global de Visual Studio. El IDE administra la complejidad de miles de comandos filtrando la funcionalidad disponible para el usuario en función del contexto.

Cuando cambia el contexto de un usuario, como cambiar de una ventana de diseño a una ventana de edición de código, desaparece la funcionalidad no relacionada con el nuevo contexto. Al mismo tiempo, la nueva funcionalidad aparece junto con la información dinámica relacionada, como las opciones Propiedades y Cuadro de herramientas. El usuario no debe notar el intercambio del conjunto de comandos disponible. Si el usuario está distraído o confundido por los comandos que aparecen o desaparecen, el diseño de la interfaz de usuario necesita un ajuste. El contexto actual del usuario siempre se indica de una o varias maneras, como en la barra de título del IDE, la ventana Propiedades o el cuadro de diálogo Páginas de propiedades.

Las barras de comandos permiten flexibilidad en la interfaz de usuario. Las únicas estructuras de comandos inherentes al entorno de Visual Studio son el menú principal y la barra de comandos principal, que se pueden personalizar e incluso ocultar. Otras barras de comandos aparecen y desaparecen en función del estado de la aplicación. Las ventanas de herramientas y los editores de documentos también pueden contener barras de herramientas incrustadas dentro de sus bordes de ventana.

Directrices básicas

Utilice comandos compartidos, grupos de comandos y menús existentes siempre que sea posible.

Puesto que los comandos se muestran normalmente en función del contexto, el uso de los menús compartidos existentes y los grupos de comandos garantiza que la estructura de comandos permanezca relativamente estable entre los cambios en el contexto. Reutilizar comandos compartidos y colocar nuevos comandos cerca de comandos compartidos relacionados también reduce la complejidad del IDE y crea una experiencia más fácil de usar. Si es necesario definir un nuevo comando, intente colocarlo en un grupo de comandos compartido existente. Si es necesario definir un nuevo grupo, colóquelo en un menú compartido existente cerca de un grupo de comandos relacionado antes de crear un nuevo menú de nivel superior.

No cree iconos para cada comando.

Piense detenidamente antes de crear un icono de comando. Los iconos se deben crear solamente para los comandos que:

- aparecen en una barra de herramientas predeterminada.
- es probable que los usuarios agreguen a una barra de herramientas a través del cuadro de diálogo **Personalizar...**
- tienen un icono asociado a la misma acción en otro producto de Microsoft.

Limitar la adición de atajos de teclado

La gran mayoría de los usuarios emplean una pequeña fracción de todos los accesos directos disponibles. En caso de duda, no vincule la función a un método abreviado de teclado. Trabaje con su equipo de experiencia de usuario antes de agregar nuevos accesos directos.

Proporcione a los comandos una ubicación de menú predeterminada.

Tenga en cuenta que sus comandos serán personalizados por otros y diseñarlos en consecuencia. No existe tal cosa como un comando oculto. Todos los comandos de Visual Studio aparecen en el cuadro de diálogo

Herramientas > personalizar, la ventana de comandos, autocompletar, el cuadro de diálogo **Herramientas > opciones > teclado** y el entorno de herramientas de desarrollo (DTE). Asegúrese de asignar a sus comandos un nombre y información sobre herramientas en el archivo .ctc para que los usuarios puedan encontrarlos fácilmente.

No duplique comandos compartidos en una barra de herramientas incrustada.

Es útil colocar comandos cerca del área del foco del usuario. Una forma de hacerlo es crear una barra de herramientas incrustada en la parte superior de la ventana de herramientas o el editor de documentos. Los comandos colocados en la barra de herramientas deben ser específicos de la región de contenido dentro de la ventana. No duplique los comandos compartidos en estas barras de herramientas. Por ejemplo, nunca coloque un icono "Guardar" dentro de una barra de herramientas incrustada.

Visibilidad de contenido y comandos

Los comandos existen en los ámbitos siguientes: **Entorno**, **Jerarquía** y **Documento**. Conozca cada ámbito para tener confianza en la colocación del comando.

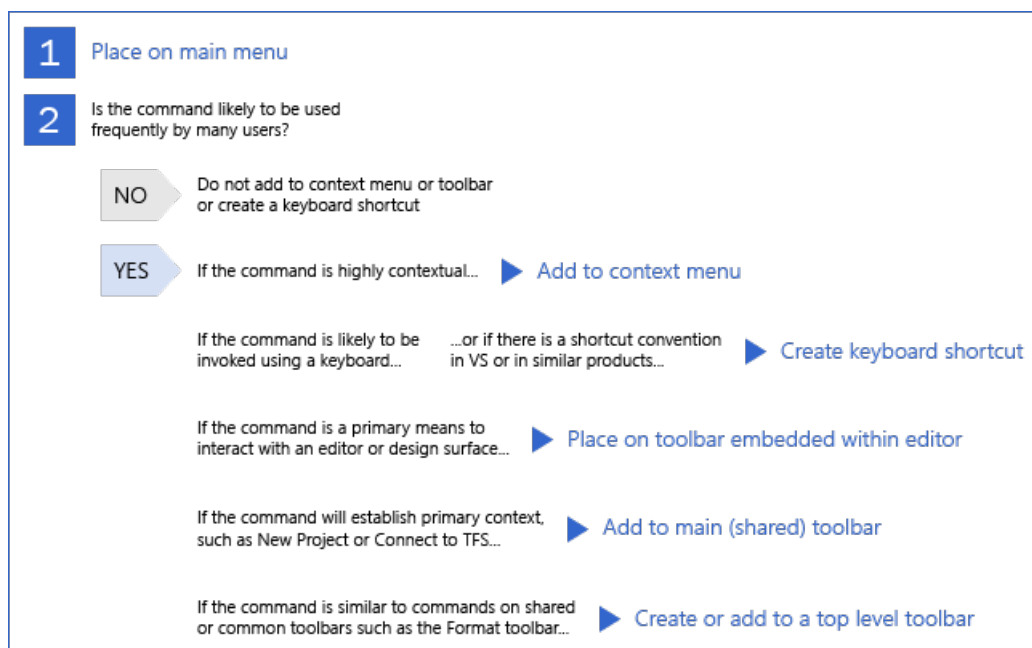
Los comandos del ámbito **Entorno** establecen el contexto principal y se comparten entre varios contextos. Modifican la visibilidad o disposición de documentos y ventanas de herramientas. Entre los comandos del ámbito de entorno se encuentran **Nuevo proyecto**, **Conectar al servidor**, **Adjuntar proceso**, **Cortar**, **Copiar**, **Pegar**, **Buscar**, **Opciones**, **Personalizar**, **Nueva ventana** y **Ver ayuda**.

Los comandos del ámbito **Jerarquía** administran jerarquías en Visual Studio, incluidos **Proyecto**, **Equipo** y **Datos**. Se relacionan con el subcontexto de un proyecto, por ejemplo, **Depurar**, **Compilar**, **Probar**, **Arquitectura** o **Analizar**. Entre los comandos del ámbito Jerarquía se encuentran **Agregar nuevo elemento**, **Nueva consulta**, **Configuración del proyecto**, **Agregar nuevo origen de datos**, **Iniciar asistente de rendimiento** y **Nuevo diagrama**.

Los comandos del ámbito **Documento** actúan sobre el contenido de un documento, como código, diseño o una consulta de elemento de trabajo (WIQ). También actúan en la vista de una ventana de herramientas o son específicos de esa ventana de herramientas. Los comandos de ámbito de documento también actúan en los objetos de archivo que son específicos de la jerarquía, como **Quitar del proyecto**. Entre los comandos del ámbito del documento se encuentran **Refactorizar > cambiar nombre**, **Crear copia del elemento de trabajo**, **Expandir todo**, **Contraer todo** y **Crear tarea de usuario**.

Decisiones de colocación de comandos

Una vez que haya decidido crear un comando, tendrá que determinar su ubicación adecuada y si desea crear un método abreviado de teclado. Siga esta ruta de decisión para establecer dónde colocar el comando:



Ruta de decisión para la colocación de comandos en Visual Studio

Colocación de comandos en los menús

Barra de menú principal

La barra de menú principal debe ser la ubicación estándar para los comandos de cualquier paquete de menú específico del contexto que contribuya a la interfaz de usuario. La barra de menús principal difiere de otras estructuras de comandos en que el entorno la utiliza para controlar qué comandos son visibles. Todas las demás barras de comandos simplemente deshabilitan los comandos que están fuera de contexto, ya sea que se coloquen en un menú o en una barra de herramientas.

El entorno define un conjunto de comandos integrados en la barra de menús principal que son comunes en todo el IDE y varios dominios de tareas. Estos comandos siempre están visibles independientemente de qué VSPackages se cargan en el entorno. Aunque VSPackages puede extender este conjunto de comandos, el conjunto de comandos de cada producto y la colocación de sus comandos es responsabilidad de cada equipo.

La estructura del menú principal de Visual Studio se puede desglosar en las siguientes categorías de menú:

Menús principales

- Archivo
- Editar
- Ver
- Herramientas
- Periodo
- Ayuda

Menús específicos del proyecto

- proyecto
- Build
- Depurar

Menús específicos del contexto

- Team
- data
- Prueba
- Architecture
- Analizar

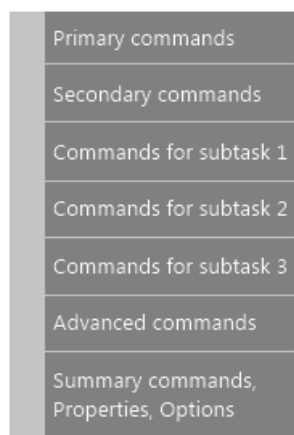
Menús específicos del documento

- Formato
- Tabla

Al diseñar menús principales, siga estas reglas:

- No exceda los 25 elementos de nivel superior en un contexto determinado
- Los menús nunca deben superar los 600 píxeles de altura.
- Evalúe un menú principal en varios contextos, como en la SKU definitiva y el perfil general.
- Los menús desplegables son aceptables.
- Los menús desplegables deben contener al menos tres elementos y no más de siete.

- Los menús desplegables deben ir solo un nivel de profundidad: algunos elementos de menú de Visual Studio tienen submenús en cascada, pero no se recomienda este patrón.
- No utilice más de seis separadores. Las agrupaciones deben adherirse a la siguiente ilustración:



- Aunque no es necesario tener cada agrupación en la figura, la adición de agrupaciones adicionales está restringida.
- Cada agrupación debe tener de dos a siete elementos de menú.

Ordenación del menú principal

Antes de agregar un nuevo elemento de nivel superior, considere la posibilidad de colocar el comando en un menú de nivel superior existente. Al agregar un nuevo menú de nivel superior, asegúrese de colocarlo en la ubicación correcta. Decida si el menú es específico del proyecto, el contexto o el documento. Mantenga el nombre del menú de nivel superior conciso y utilice una sola palabra.

Los menús principales deben reservar el resto de los comandos. Archivo, Editar y Ver siempre deben estar a la izquierda, y Herramientas, Ventana y Ayuda siempre deben estar a la derecha.

Menús contextuales

Colocar demasiada funcionalidad dentro de los menús contextuales da como resultado una interfaz difícil de aprender. Todas las funciones principales deben estar disponibles a través de la barra de menú principal. La colocación de comandos debe conciliarse con los comandos existentes para evitar comandos duplicados. Para los menús contextuales, el shell define los grupos de menú estándar que se deben incluir en función de si el menú contextual es para la solución, un nodo de proyecto o un elemento de proyecto.

Al diseñar menús contextuales, siga las mismas reglas que para el menú principal y, además:

- No exceda los 25 elementos de menú de nivel superior.
- Los menús desplegables son aceptables, pero no deben superar un nivel de profundidad - nunca utilice flyouts en cascada.
- No utilice más de seis separadores.

Colocación de comandos en las barras de herramientas

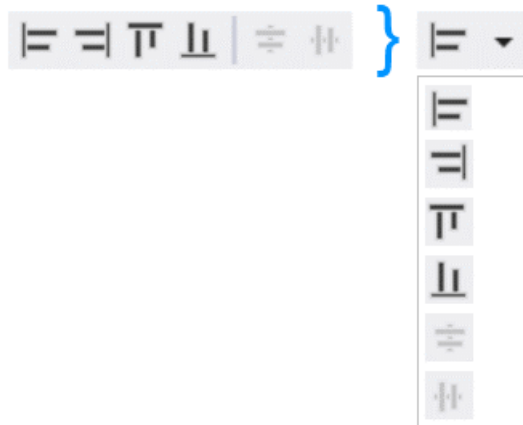
Barras de herramientas generales

Al diseñar y organizar barras de herramientas, siga estos estándares:

- No uses más de un verbo por botón. Un botón: una acción.
- Utilice texto junto al icono solo si necesita reforzarse con la etiqueta.
- Utilice un cuadro combinado exclusivamente para las propiedades que se cambiarán varias veces en una sesión. De lo contrario, exponga la propiedad en otro lugar.
- El ancho de un cuadro combinado debe ser igual al ancho del elemento más largo dentro del cuadro +

30%. Por ejemplo, si el elemento más largo es de 200 píxeles, el cuadro combinado debe tener 260 píxeles de ancho.

- Limite el uso de separadores. El uso de un separador junto a una lista desplegable es un antipatrón, porque la forma de la lista desplegable actúa como un separador visual.
- Los grupos de iconos deben contener de tres a seis iconos.
- Si los calificadores dan como resultado varios comandos útiles, utilice un botón de división que almacene la última configuración:



Ejemplo de un botón de división. Los seis comandos de la izquierda pueden caber en un solo botón.

Barras de herramientas específicas del producto

Cada producto puede proporcionar una barra de herramientas predeterminada que contiene comandos importantes y de uso frecuente, y la barra de herramientas predeterminada de cada producto debe aparecer la primera vez que visual Studio se inicia después de instalar el producto.

Los productos también deben aprovechar los menús y grupos de comandos compartidos proporcionados por el IDE. Cada grupo de comandos compartidos se coloca en un menú compartido destinado a organizar los comandos relacionados de una manera significativa para el usuario. Es importante aprovechar esta estructura de comandos compartidos para reducir la complejidad.

Barras de herramientas globales

Las barras de herramientas globales deben caber en una fila de inmediato. Al crear una nueva barra de herramientas global, siga las directrices para ese tipo de barra de herramientas.

Directrices generales de la barra de herramientas:

- Cada barra de herramientas tiene 24 píxeles en controles comunes (agarrador, desbordamiento).
- Cada botón de la barra de herramientas tiene 22 píxeles de ancho, incluido el relleno. Hacer que el icono sea un botón de división añade otros 11 píxeles de ancho.
- Se permite la duplicación de comandos en las barras de herramientas.

Las **barras de herramientas específicas** del documento aparecen cuando un determinado tipo de archivo está activo y desaparecen cuando se activa un tipo de archivo diferente.

- Es posible que las barras de herramientas específicas del documento no tengan más de 12 botones.
- El ancho total de la barra de herramientas no puede superar los 300 píxeles.
- Cada tipo de archivo puede tener una barra de herramientas incrustada o una barra de herramientas global específica del documento, pero no ambas.

Las **barras de herramientas específicas** del contexto aparecen cuando se establece un contexto

determinado y tienden a permanecer activas durante períodos prolongados.

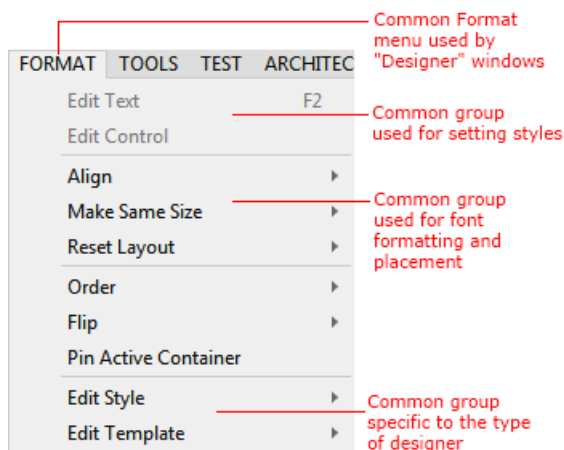
- El límite de botones para todas las barras de herramientas específicas del contexto es 18.
- Si la mayoría de los usuarios no emplean constantemente los comandos de esta barra de herramientas cuando el contexto está activo, no asocie esta barra de herramientas con un contexto.
- Asegúrese de que la barra de herramientas desaparece al salir del contexto. Ninguna de estas barras de herramientas debe aparecer al inicio.

Las **barras de herramientas sin contexto** nunca aparecen automáticamente. Estos se muestran solamente cuando el usuario los activa. Mantenga el ancho máximo por debajo de 200 píxeles.

Organización general y grupos definidos por shell

Utilice comandos compartidos, grupos de comandos y menús existentes. Si es necesario definir un nuevo comando, intente colocarlo en un grupo de comandos compartido existente. Si es necesario definir un nuevo grupo, intente colocarlo en un menú compartido existente cerca de un grupo de comandos relacionado antes de crear un nuevo menú de nivel superior. Esto reduce la complejidad de los comandos al tiempo que garantiza la colocación coherente de los comandos en el IDE.

El menú **Formato** compartido, que normalmente se muestra en el contexto de las ventanas de documento de estilo diseñador, se ilustra en la siguiente imagen:



Grupos de menús en Visual Studio

Reducción y reutilización de comandos

Los comandos se muestran típicamente basados en el contexto, para reducir el número de comandos que el usuario ve en un momento dado. Sin embargo, también debe reutilizar los menús compartidos existentes y los grupos de comandos para asegurarse de que la estructura de comandos permanece relativamente estable entre los cambios en el contexto.

Reutilizar comandos compartidos y colocar nuevos comandos cerca de comandos compartidos relacionados reduce la complejidad del IDE y crea una experiencia más fácil de usar.

Comandos de nomenclatura

Convenciones de nomenclatura

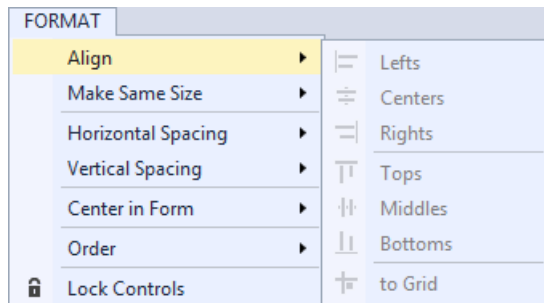
La nomenclatura de comandos coherente es fundamental para que los usuarios puedan buscar y ejecutar comandos, ya sea mediante la línea de comandos o enlazando a un método abreviado de teclado. Los nombres de comando también ayudan al usuario a comprender qué propósito sirve un comando cuando se muestra en una barra de herramientas o en un menú contextual o en cascada.

Al asignar nombres a los comandos:

- Construya texto para que sea fácilmente localizable. Para obtener más información sobre la localización de

texto, consulte [Prácticas recomendadas](#) de localización .

- Sé conciso. Los comandos no deben usar más de tres palabras.
- Usar mayúsculas y minúsculas de título: la primera letra de cada palabra debe estar en mayúsculas. Para obtener más información sobre el formato de texto en Visual Studio, vea [Estilo de texto](#) .
- Tenga en cuenta dónde se colocará el comando. ¿Está en un menú de nivel superior o en un control flotante? Por ejemplo, al agrupar comandos de alineación en un control flotante, el comando de nivel superior debe ser "Alinear" y los comandos desplegables deben ser "Izquierda", "Derecha", "Centro", "Justificar", etc. Sería redundante nombrar los comandos desplegables "Alinear a la izquierda" o "Alinear a la derecha."



Uso de iconos con comandos

Estar ahorrando en el uso de emparejamiento de iconos con comandos. Aunque asociar una imagen única con un comando acelera la capacidad del usuario para identificar ese comando, el desorden visual y la ineficiencia se producen con el uso excesivo de la imagen. Las siguientes reglas ayudan a decidir si se debe crear un icono de comando.

Utilice un icono con un comando solo si:

- El mismo comando tiene un icono asociado en otro producto de Microsoft prominente, como una de las aplicaciones de Microsoft Office.
- El comando se colocará en una barra de herramientas predeterminada.
- El comando es un comando de especialidad que es probable que los usuarios agreguen a una barra de herramientas mediante el cuadro de diálogo "Personalizar..." .

Teclas de acceso y acceso directo

Información general

Hay dos tipos de asignaciones de teclas de teclado:

- Las **teclas de acceso** (también conocidas como aceleradores) permiten el acceso del teclado a través de los menús para los comandos y a cada etiqueta en la interfaz de usuario del cuadro de diálogo. Las teclas de acceso son principalmente para fines de accesibilidad, se asignan a todos los menús y la mayoría de los controles de cuadro de diálogo, no están diseñadas para memorizarse, afectan solo a la ventana actual y están localizadas.
- Las **teclas de método abreviado** utilizan principalmente secuencias de teclas Control (Ctrl) y Función (Fn). Están diseñados más para usuarios avanzados y ayudan en la productividad. Se asignan sólo a los comandos más utilizados y permiten un acceso rápido mientras se omite el menú principal. Las teclas de método abreviado están diseñadas para memorizarse y, por ese motivo, deben asignarse de forma coherente con el esquema de perfil. Los esquemas de teclas de acceso directo pueden variar de un perfil a un perfil. Un usuario puede personalizar las teclas de método abreviado a través de **Herramientas > Opciones > teclado**.

Asignación de claves de acceso

Las claves de acceso constan de Alt más claves alfanuméricas. Asigne una clave de acceso a cada elemento de menú sin excepción. Siga Windows y las convenciones comunes para asignar claves de acceso. por ejemplo, la clave de acceso para **file > New** siempre debe ser **Alt, F, N**.

No utilice letras de un solo píxel de ancho como 'i' (en mayúsculas o minúsculas) o una 'l' minúscula, y evite usar caracteres con descendientes (g, j, p, q e y) ya que son difíciles de distinguir.

Evite el uso de claves duplicadas cuando sea posible. En los casos en los que la duplicación es inevitable, el sistema de menús controla los conflictos recorriendo todos los comandos que utilizan la clave. Por ejemplo, para un comando hipotético "Número" en el menú Archivo que duplica la tecla de acceso "N", **Alt, F, N** crearía un nuevo archivo, y **Alt, F, N, N** realizaría el comando "Número".

Asignación de teclas de método abreviado

Evite asignar nuevas teclas de método abreviado, ya que no son necesarias para cada comando e impuestos del sistema (y la memoria del usuario) si se utilizan en exceso. Los datos del Programa de mejora de la experiencia del usuario (CEIP) indican que los usuarios de Visual Studio solo usan un pequeño subconjunto de los accesos directos integrados.

Al definir accesos directos, siga estas reglas:

- **Utilice las secuencias de teclas Control (Ctrl) y Función (Fn).**
- **Conservar accesos directos utilizados con frecuencia.** Mantenga los accesos directos más populares.
- **Haga que los accesos directos del editor sean fáciles de escribir.** Enlazar accesos directos fáciles de escribir a comandos que los desarrolladores necesitan más al escribir código. Por ejemplo, `Edit.InvokeSmartTag` debe tener una tecla de método abreviado rápido como `Ctrl+/` y no `Alt+Mayús+F10`.
- **Esfuézate por obtener accesos directos con temas consistentes.**
- **Siga las directrices de Windows para determinar qué claves modificadoras emplear.** Utilice las combinaciones de teclas `Ctrl` para los comandos que tienen efectos a gran escala, como los comandos que se aplican a todo un documento. Utilice las combinaciones de teclas `Mayús` para los comandos que amplían o complementan las acciones de la tecla de método abreviado estándar. No utilice las combinaciones `Ctrl+Alt`.
- **Eliminar accesos directos extraños.** Si usted tiene una característica heredada, considere quitar los accesos directos que se utilizan con la infrecuencia extrema (menos de 10 veces de los datos CEIP) o la infrecuencia moderada (menos de 100 veces de los datos CEIP) si una clave de acceso proporciona el acceso rápido al mismo comando. Por ejemplo: `Alt, H, C` abrirá Ayuda/Contenido.

No hay una manera sencilla de comprobar la disponibilidad de accesos directos. Si desea agregar un acceso directo, siga estos pasos:

1. Compruebe la lista de [accesos directos de Visual Studio 2013](#) para determinar si hay comandos similares con los que agrupar el suyo.
2. Vaya a **Herramientas > Opciones > Entorno > teclado** y pruebe el acceso directo. Compruebe cada esquema de asignación de teclado que aparece en "Aplicar el siguiente esquema de asignación de teclado adicional." Marque los perfiles General, C, VB y C++, ya que comparten accesos directos únicos. El acceso directo está disponible si no está asignado en ninguno de esos lugares.

Patrones de interacción para Visual Studio

06/04/2020 • 5 minutes to read • [Edit Online](#)

Información general

Un patrón de diseño, en general, es el núcleo de un diseño que se puede aplicar en situaciones específicas para resolver problemas con conjuntos similares de restricciones. Los diseñadores de características y sistemas utilizan estos patrones de diseño como puntos de partida, que luego se pueden adaptar a su situación específica.

Visual Studio tiene una biblioteca de patrones de interacción comunes que se deben tener en cuenta al crear nuevas características. Hay dos contextos principales para nuestros patrones de diseño: Visual Studio Client (devenv) y Visual Studio Online. Para algunos problemas de diseño, hay un patrón ubicuo que funciona bien en todas las situaciones. En muchos casos, sin embargo, la solución puede ser diferente para la interfaz de usuario que se presenta en un explorador y la que se hospeda en una aplicación cliente.

Tipos de patrones de cliente de Visual Studio

TIPO DE PATRÓN	DESCRIPCIÓN	EJEMPLOS
Patrones de nivel de aplicación	Patrones de alto nivel comunes a la aplicación, determinando o mostrando el contexto de la aplicación y conteniendo patrones compuestos y de control dentro de ellos	- Ventanas de herramientas - Ventanas de documentos
Patrones compuestos	Patrones comunes que pueden abarcar patrones de aplicación, o un patrón reconocido compuesto por varios controles en una configuración distinta	- Conmutación de vista - Constructores de listas - Visualización de datos - Notificaciones - Validación - Modelos de selección
Patrones de control	Detalles sobre cómo se espera que se comporten los controles de bajo nivel	- Vistas a los árboles - Edición dentro de un control de cuadrícula

Patrones de aplicación

En un nivel alto, la interfaz de Visual Studio consta de varias ventanas, cuadros de diálogo, comandos y barras de herramientas dentro de un único IDE. La jerarquía de Visual Studio determina los menús de contexto e unidades. Los puntos clave de integración en la interfaz de usuario del IDE son ventanas de documento, ventanas de herramientas, proyectos, la estructura de comandos, el editor de texto, el cuadro de herramientas, la ventana Propiedades y Herramientas > Opciones.

Hay patrones de uso básicos para cada uno de los puntos de integración clave en la interfaz de usuario del IDE:

- [Menús y comandos para Visual Studio](#)
- [Patrones de aplicaciones para Visual Studio](#)
 - [Interacciones de ventanas](#)
 - [Ventanas de herramientas](#)

- [Convenciones del editor de documentos](#)
- [Diálogos](#)
- [Proyectos](#)

Patrones de control comunes

Los patrones de control se basan principalmente en cómo se espera que se comporten los controles individuales. Este es un área en la que la consistencia es más crítica.

Los controles más comunes de Visual Studio deben seguir las directrices de Desktop Windows. Nuestras directrices solo incluyen áreas en las que necesitamos aumentar las convenciones comunes con interacciones específicas de Visual Studio, o lugares en los que reemplazamos las directrices por completo con el fin de adaptar Visual Studio para satisfacer las necesidades de nuestros usuarios sofisticados.

- [Patrones de control comunes para Visual Studio](#)
 - [Controles comunes](#)
 - [Controles de texto](#)
 - [Botones e hipervínculos](#)

Patrones compuestos

Hay varias maneras en que los usuarios esperan realizar tareas. Siempre que sea posible, las características deben diseñarse para utilizar esos patrones tanto para la interacción como para el diseño visual.

Aunque hay muchos patrones compuestos en Visual Studio, algunos de los más importantes con respecto a la coherencia son:

- [Patrones compuestos para Visual Studio](#)
 - [Interfaz de usuario en el objeto y miradas](#)
 - [Modelos de selección](#)
 - [Configuración de persistencia y ahorro](#)
 - [Entrada táctil](#)

Patrones de aplicaciones para Visual Studio

06/04/2020 • 80 minutes to read • [Edit Online](#)

Interacciones de ventanas

Información general

Los dos tipos de ventana principal utilizados en Visual Studio son editores de documentos y ventanas de herramientas. Raros, pero posibles, son grandes diálogos no iniciados. Aunque todos estos son no esmimosos en el shell, sus patrones son fundamentalmente diferentes. En esta sección se trata la diferencia entre las ventanas de documento, las ventanas de herramientas y los cuadros de diálogo no más mideteros. Los patrones de diálogo modal se tratan en [Diálogos](#).

Comparación de patrones de uso de ventanas

Las **ventanas** de documento casi siempre se muestran dentro del pozo del documento. Esto le da al editor de documentos un "escenario central" para organizar ventanas de herramientas suplementarias alrededor.

Una ventana de **herramientas** se muestra con mayor frecuencia como una ventana separada y más pequeña contraída contra el borde del IDE. Esto puede ser visible, oculto u auto-oculto. Sin embargo, a veces las ventanas de herramientas se presentan dentro del documento bien, desmarcando la **propiedad Ventana/Acoplamiento** en la ventana. Esto da como resultado más bienes raíces, pero también una decisión de diseño común: al intentar integrarse en Visual Studio, debe decidir si la característica debe mostrar una ventana de herramientas o una ventana de documento.

Se desaconsejan los cuadros de diálogo no especificados en Visual Studio. La mayoría de los cuadros de diálogo no iniciados son, por definición, ventanas de herramientas flotantes y deben implementarse de esa manera. Los cuadros de diálogo no iniciados se permiten en los casos en que el tamaño de una ventana de herramientas normal acoplada al lado del vaciado sería demasiado limitador. También se permiten en los casos en que el usuario sería probable que mueva el cuadro de diálogo a un monitor secundario.

Piense cuidadosamente en qué tipo de contenedor necesita. Las consideraciones comunes del patrón de uso para el diseño de la interfaz de usuario se encuentran en la tabla siguiente.

	VENTANA DEL DOCUMENTO	VENTANA DE HERRAMIENTAS	DIÁLOGO NO MIDETES
Posición	Siempre posicionado dentro del documento bien y no se acopla alrededor de los bordes del IDE. Se puede "tirar" para que flote por separado de la cáscara principal.	Por lo general, se acopla la pestaña alrededor de los bordes del IDE, pero se puede personalizar para que sea flotante, oculto automáticamente (sin anclar) o acoplado dentro del documento.	Ventana flotante grande separada del IDE.

	VENTANA DEL DOCUMENTO	VENTANA DE HERRAMIENTAS	DIÁLOGO NO MIDETES
Confirmar modelo	<i>Compromiso retrasado</i> Para guardar los datos en un documento, el usuario debe publicar el comando **Guardar archivo > **, Guardar como Guardar todo. Una ventana de documento tiene el concepto de que los datos dentro de ella se "suciedad" y luego se compromete a uno de los comandos save. Al cerrar una ventana de documento, todo el contenido se guarda en el disco o se pierde.	<i>Compromiso inmediato</i> No hay ningún modelo de guardado. Para las ventanas de herramientas del inspector que ayudan a editar un archivo, el archivo debe estar abierto en el editor o diseñador activo, y el editor o diseñador es el propietario de la salvaguardia.	<i>Compromiso retrasado o inmediato</i> La mayoría de las veces, un cuadro de diálogo no es posible requiere una acción para confirmar los cambios y permite una operación "Cancelar", que revierte los cambios realizados en la sesión de diálogo. Esto diferencia un cuadro de diálogo no quede de una ventana de herramientas en que las ventanas de herramientas siempre tienen un modelo de confirmación inmediata.
Visibilidad	<i>Abrir/Crear (archivo) y Cerrar</i> La apertura de una ventana de documento se realiza abriendo un documento existente o utilizando una plantilla para crear un nuevo documento. No hay ningún <comando "Abrir editor específico">".	<i>Ocultar y mostrar</i> Las ventanas de herramientas de instancia única se pueden ocultar o mostrar. El contenido y los estados de la ventana de herramientas persisten, ya sea a la vista u ocultos. Las ventanas de herramientas de varias instancias se pueden cerrar y ocultar. Cuando se cierra una ventana de herramientas de varias instancias, se descarta el contenido y el estado de la ventana de herramientas.	<i>Lanzado desde un comando</i> Los cuadros de diálogo se inician desde un comando basado en tareas.
Instancias	<i>Multi-instancia</i> Varios editores pueden estar abiertos al mismo tiempo y editar diferentes archivos, mientras que algunos editores también permiten que el mismo archivo esté abierto en más de un editor (mediante el comando Ventana > Nueva ventana). Un solo editor puede estar editando uno o varios archivos al mismo tiempo (Diseñador de proyectos).	<i>Instancia única o múltiple</i> El contenido cambia para reflejar el contexto (como en el Explorador de propiedades) o insertar el foco/contexto en otras ventanas (Lista de tareas, Explorador de soluciones). Las ventanas de herramientas de instancia única y de varias instancias deben asociarse a la ventana de documento activa a menos que haya una razón convincente para no hacerlo.	<i>Instancia única</i>

	VENTANA DEL DOCUMENTO	VENTANA DE HERRAMIENTAS	DIÁLOGO NO MIDETES
Ejemplos	Editores de texto, como el editor de código Superficies de diseño, como un diseñador de formularios o una superficie de modelado Diseños de control similares a los cuadros de diálogo, como el Diseñador de manifiestos	El Explorador de soluciones proporciona una solución y proyectos contenidos en la solución El Explorador de servidores proporciona una vista jerárquica de los servidores y las conexiones de datos que el usuario elige abrir en la ventana. Al abrir un objeto desde la jerarquía de base de datos, como una consulta, se abre una ventana de documento y se permite al usuario editar la consulta. El Navegador de propiedades muestra las propiedades del objeto seleccionado en una ventana de documento u otra ventana de herramientas. Las propiedades se presentan en una vista de cuadrícula jerárquica o en controles complejos de tipo cuadro de diálogo y permiten al usuario establecer los valores de esas propiedades.	

Ventanas de herramientas

Información general

Las ventanas de herramientas admiten el trabajo del usuario que se produce en las ventanas de documento. Se pueden usar para mostrar una jerarquía que representa un objeto raíz fundamental que Visual Studio proporciona y puede manipular.

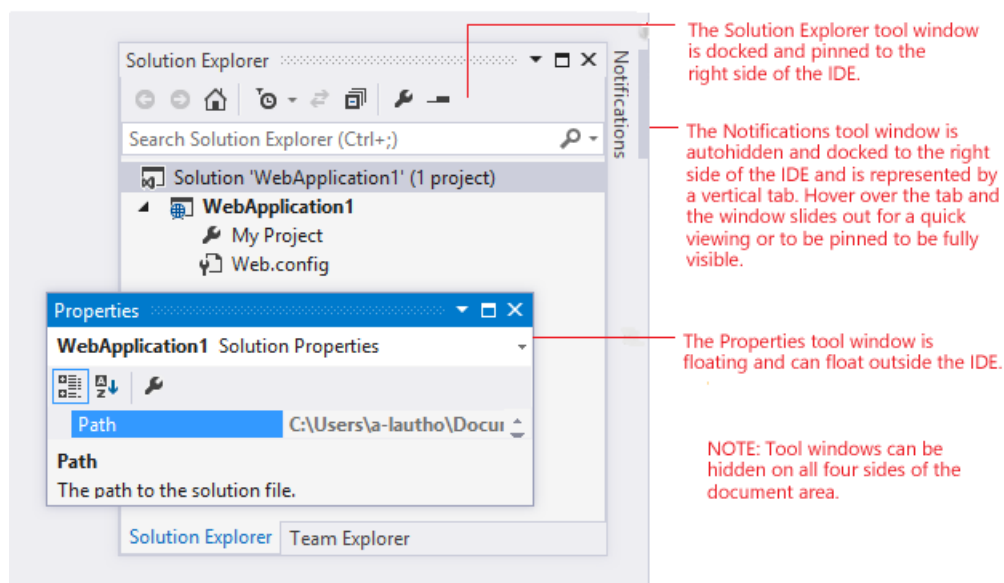
Al considerar una nueva ventana de herramientas en el IDE, los autores deben:

- Utilice ventanas de herramientas existentes apropiadas para tareas y no cree otras nuevas con una funcionalidad similar. Las nuevas ventanas de herramientas solo se deben crear si ofrecen una "herramienta" o funcionalidad significativamente diferente que no se puede integrar en una ventana similar, o convirtiendo una ventana existente en un concentrador dinámico.
- Utilice una barra de comandos estándar, si es necesario, en la parte superior de la ventana de herramientas.
- Sea coherente con los patrones ya presentes en otras ventanas de herramientas para la presentación de control y la navegación por teclado.
- Sea coherente con la presentación de control en otras ventanas de herramientas.
- Haga que las ventanas de herramientas específicas del documento sean visibles automáticamente cuando sea posible, de modo que aparezcan solo cuando se active el documento principal.
- Asegúrese de que el contenido de la ventana sea navegable por el teclado (teclas de flecha de soporte).

Estados de la ventana de herramientas

Las ventanas de herramientas de Visual Studio tienen diferentes estados, algunos de los cuales están activados por el usuario (como la característica de ocultación automática). Otros estados, como la visible automática, permiten que las ventanas de herramientas aparezcan en el contexto correcto y se oculten cuando no sea necesario. Hay cinco estados de ventana de herramientas en total.

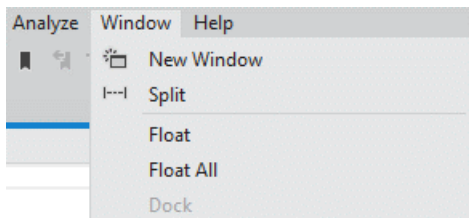
- Las ventanas de herramientas **acopladas/ancladas** se pueden adjuntar a cualquiera de los cuatro lados del área del documento. El icono de pasador aparece en la barra de título de la ventana de herramientas. La ventana de herramientas se puede acoplar horizontal o verticalmente a lo largo del borde del vaciado y otras ventanas de herramientas, y también se puede vincular a pestañas.
- Las ventanas de herramientas **ocultas automáticamente** no están ancladas. La ventana se puede deslizar fuera de la vista, dejando una pestaña (con el nombre de la ventana de herramientas y su icono) en el borde del área del documento. La ventana de herramientas se desliza hacia fuera cuando un usuario pasa el cursor sobre la pestaña.
- Las ventanas de herramientas **visibles** automáticamente aparecen automáticamente cuando se inicia otra parte de la interfaz de usuario, como un editor, o gana el foco.
- Las ventanas de herramientas **flotantes** se sitúan fuera del IDE. Esto es útil para configuraciones de varios monitores.
- Las ventanas de herramientas de **documentos con pestañas** se pueden acoplar dentro del pozo del documento. Esto es útil para ventanas de herramientas grandes, como el Examinador de objetos, que necesitan más bienes raíces de lo que permite el acoplamiento a los bordes del marco.



Estados de la ventana de herramientas en Visual Studio

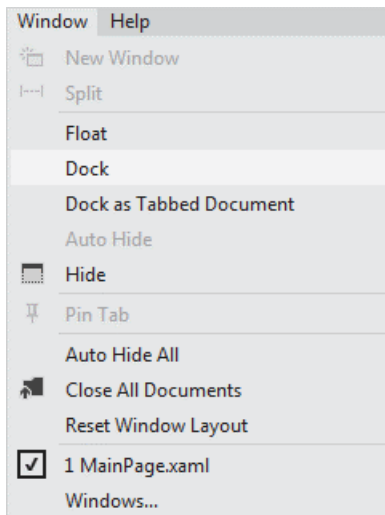
Instancia única y multiinstancia

Las ventanas de herramientas son de una sola instancia o de varias instancias. Es posible que algunas ventanas de herramientas de instancia única estén asociadas a la ventana de documento activa, mientras que las ventanas de herramientas de varias instancias podrían no estar asociadas. Las ventanas de herramientas de varias instancias responden al comando **Ventana > nueva ventana** creando una nueva instancia de la ventana. La siguiente imagen ilustra una ventana de herramientas que habilita el comando Nueva ventana cuando una instancia de la ventana está activa:



Ventana de herramientas que habilita el comando 'Nueva ventana' cuando una instancia de la ventana está activa

Las ventanas de herramientas de una sola instancia se pueden ocultar o mostrar, mientras que las ventanas de herramientas de varias instancias se pueden cerrar y ocultar. Todas las ventanas de herramientas se pueden acoplar, vincular a tabulaciones, flotar o establecerse como una ventana secundaria de interfaz de varios documentos (MDI) (similar a una ventana de documento). Todas las ventanas de herramientas deben responder a los comandos de administración de ventanas adecuados en el menú Ventana:



Comandos de administración de ventanas en el menú Ventana de Visual Studio

Ventanas de herramientas específicas del documento

Algunas ventanas de herramientas están diseñadas para cambiar en función de un tipo determinado de documento. Estas ventanas se actualizan continuamente para reflejar la funcionalidad aplicable a la ventana de documento activa en el IDE.

Ejemplos de ventanas de herramientas cuyo contenido cambia para reflejar el editor seleccionado son el Cuadro de herramientas y el Esquema del documento. Estas ventanas muestran una marca de agua cuando un editor tiene el foco que no ofrece contexto a la ventana.

Ventanas de herramientas de lista navegables

Algunas ventanas de herramientas muestran una lista de elementos navegables con los que el usuario puede interactuar. En este tipo de ventana, siempre debe haber comentarios para el elemento actual en la lista, incluso si la ventana está inactiva. La lista debe responder a los comandos **GoToNextLocation** y **GoToPrevLocation** cambiando también el elemento seleccionado actualmente en la ventana

Ejemplos de ventanas de herramientas de lista navegables son el Explorador de soluciones y la ventana Buscar resultados.

Tipos de ventanas de herramientas

Ventanas de herramientas comunes y sus funciones

Ventanas de herramientas jerárquicas

VENTANA DE HERRAMIENTAS	FUNCIÓN

VENTANA DE HERRAMIENTAS	FUNCIÓN
Explorador de soluciones	Un árbol jerárquico que muestra una lista de documentos contenidos en proyectos, archivos varios y elementos de solución. La visualización de los elementos dentro de los proyectos se define mediante el paquete que posee el tipo de proyecto (por ejemplo, tipos basados en referencias, basados en directorios o en modo mixto).
Vista de clases	Un árbol jerárquico de las clases y varios elementos en el conjunto de trabajo de documentos, independientemente de los propios archivos.
Explorador de servidores	Un árbol jerárquico que muestra todos los servidores y conexiones de datos de la solución.
Esquema de documento	La estructura jerárquica del documento activo.

Ventanas de herramientas de cuadrícula

VENTANA DE HERRAMIENTAS	FUNCIÓN
Propiedades	Cuadrícula que muestra una lista de propiedades para el objeto seleccionado, junto con selectores de valor para editar esas propiedades.
Lista de tareas	Una cuadrícula que permite al usuario crear/editar/eliminar tareas y comentarios.

Ventanas de herramientas de contenido

VENTANA DE HERRAMIENTAS	FUNCIÓN
Ayuda	Una ventana que permite a los usuarios acceder a varios métodos para obtener ayuda, desde "¿Cómo puedo?" vídeos a foros de MSDN.
Ayuda dinámica	Una ventana de herramientas que muestra vínculos para ayudar a los temas aplicables a la selección actual.
Examinador de objetos	Conjunto de marcos de dos columnas con una lista de componentes de objetos jerárquicos en el panel izquierdo y las propiedades y métodos del objeto en la columna derecha.

Ventanas de herramientas de diálogo

VENTANA DE HERRAMIENTAS	FUNCIÓN
Buscar	Un cuadro de diálogo que permite al usuario buscar o buscar y reemplazar en varios archivos dentro de la solución.
Búsqueda avanzada	Un cuadro de diálogo que permite al usuario buscar o buscar y reemplazar en varios archivos dentro de la solución.

Otras ventanas de herramientas

VENTANA DE HERRAMIENTAS	FUNCIÓN
Cuadro de herramientas	La ventana de herramientas utilizada para almacenar elementos que se colocarán en superficies de diseño, lo que proporciona una fuente de arrastre coherente para todos los diseñadores.
Página de inicio	Portal del usuario a Visual Studio, con acceso a fuentes de noticias para desarrolladores, ayuda de Visual Studio y proyectos recientes. Los usuarios también pueden crear páginas de inicio personalizadas copiando el archivo "StartPage.xaml" desde el directorio de archivos de programa de Visual Studio "Common7-IDE-StartPages" a la carpeta StartPages del directorio de documentos de Visual Studio y, a continuación, editando el XAML a mano o abriéndolo en Visual Studio u otro editor de código.
VENTANA DE HERRAMIENTAS	FUNCIÓN
Cuadro de herramientas	La ventana de herramientas utilizada para almacenar elementos que se colocarán en superficies de diseño, lo que proporciona una fuente de arrastre coherente para todos los diseñadores.

Ventanas de herramientas del depurador

VENTANA DE HERRAMIENTAS	FUNCIÓN
Autos	
Inmediata	
Salida	La ventana de salida se puede utilizar siempre que tenga eventos textuales o estado para declarar.
Memoria	
Puntos de interrupción	
En ejecución	
Documentos	
Pila de llamadas	
Locals	
Reloj	
Desensamblado	
Registros	
Subprocesos	

Convenciones del editor de documentos

Interacciones de documentos

El "documento bien" es el espacio más grande dentro del IDE y es donde el usuario generalmente ha centrado su atención con el fin de completar sus tareas, asistido por ventanas de herramientas suplementarias. Los editores de documentos representan las unidades fundamentales de trabajo que el usuario abre y guarda en Visual Studio. Conservan un fuerte sentido de selección vinculado al Explorador de soluciones u otras ventanas de jerarquía activas. El usuario debe ser capaz de apuntar a una de esas ventanas de jerarquía y saber dónde está contenido el documento y su relación con la solución, el proyecto u otro objeto raíz proporcionado por un paquete de Visual Studio.

La edición de documentos requiere una experiencia de usuario coherente. Para permitir que el usuario se centre en la tarea en cuestión en lugar de en la administración de ventanas y la búsqueda de comandos, seleccione una estrategia de vista de documento que mejor se adapte a las tareas del usuario para editar ese tipo de documento.

Interacciones comunes para el documento bien

- Mantenga un modelo de interacción coherente en las experiencias comunes de **Nuevo archivo** y **Abrir archivo**.
- Actualice la funcionalidad relacionada en ventanas y menús relacionados cuando se abra la ventana del documento.
- Los comandos de menú se integran adecuadamente en menús comunes como los menús **Editar**, **Formato** y **Ver**. Si hay una cantidad sustancial de comandos especializados disponibles, se puede crear un nuevo menú. Este nuevo menú debe estar visible solo cuando el documento tiene el foco.
- Una barra de herramientas incrustada se puede colocar en la parte superior del editor. Esto es preferible a tener una barra de herramientas independiente que aparece fuera del editor.
- Mantenga siempre una selección en el Explorador de soluciones o en una ventana de jerarquía activa similar.
- Al hacer doble clic en un documento en el Explorador de soluciones, se debe realizar la misma acción que **Abrir**.
- Si se puede utilizar más de un editor en un tipo de documento, el usuario debe poder anular o restablecer la acción predeterminada en un tipo de documento determinado mediante el cuadro de diálogo **Abrir con** haciendo clic con el botón derecho en el archivo y seleccionando **Abrir con** en el menú contextual.
- No cree un asistente en un documento bien.

Expectativas de usuario para tipos de documentos específicos

Hay varios tipos básicos diferentes de editores de documentos y cada uno tiene un conjunto de interacciones que son coherentes con otros del mismo tipo.

- **Editor basado en texto:** editor de código, archivos de registro
- **Superficie de diseño:** WPF WPF Forms designer, Windows Forms
- **Editor de estilo de diálogo:** Diseñador de manifiestos, propiedades del proyecto
- **Diseñador de modelos:** diseñador de flujo de trabajo, mapa de código, diagrama de arquitectura, progresión

También hay varios tipos que no son de editor que utilizan bien el documento. Aunque no editan documentos por sí mismos, sí deben seguir las interacciones estándar para las ventanas de documentos.

- **Informes:** Informe IntelliTrace, informe de Hyper-V, informe del generador de perfiles
- **Panel de control:** Centro de Diagnóstico

Editores basados en texto

- El documento participa en el modelo de pestaña de vista previa, lo que permite obtener una vista previa del documento sin abrirlo.
- La estructura del documento se puede representar dentro de una ventana de herramientas complementaria, como un esquema de documento.
- IntelliSense (si procede) se comportará de forma coherente con otros editores de código.
- Las ventanas emergentes o la interfaz de usuario de asistencia siguen estilos y patrones similares para la interfaz de usuario similar existente, como CodeLens.
- Los mensajes relativos al estado del documento se presentarán en un control de la barra de información en la parte superior del documento o en la barra de estado.
- El usuario debe ser capaz de personalizar la apariencia de las fuentes y colores mediante una página **Herramientas > Opciones**, ya sea la página Fuentes y colores compartida o una específica del editor.

Superficies de diseño

- Un diseñador vacío debe tener una marca de agua en la superficie que indique cómo empezar.
- Los mecanismos de cambio de vista seguirán los patrones existentes, como hacer doble clic para abrir un editor de código o pestañas dentro de la ventana del documento que permiten la interacción con ambos paneles.
- La adición de elementos a la superficie de diseño debe realizarse a través del cuadro de herramientas, a menos que se requiera una ventana de herramientas muy específica.
- Los elementos de la superficie seguirán un modelo de selección coherente.
- Las barras de herramientas incrustadas solo contienen comandos específicos del documento, no comandos comunes como **Guardar**.

Editores de estilo de diálogo

- El diseño del control debe seguir las convenciones normales del diseño del cuadro de diálogo.
- Las pestañas dentro del editor no deben coincidir con la apariencia de las pestañas del documento, deben coincidir con uno de los dos estilos de pestañas interiores permitidos.
- Los usuarios deben poder interactuar con los controles utilizando solo el teclado; ya sea activando el editor y tabulando a través de controles o usando mnemotécnicas estándar.
- El diseñador debe usar el modelo Save común. No se deben colocar botones generales Guardar o confirmar en la superficie, aunque otros botones pueden ser adecuados.

Diseñadores de modelos

- Un diseñador vacío debe tener una marca de agua en la superficie que indique cómo empezar.
- La adición de elementos a la superficie de diseño debe realizarse a través del cuadro de herramientas.
- Los elementos de la superficie seguirán un modelo de selección coherente.
- Las barras de herramientas incrustadas solo contienen comandos específicos del documento, no comandos comunes como **Guardar**.
- Una leyenda puede aparecer en la superficie, ya sea indicativa o una marca de agua.
- El usuario debe ser capaz de personalizar la apariencia de las fuentes/colores mediante una página **Herramientas > Opciones**, ya sea la página Fuentes y colores compartida o una específica del editor.

Informes

- Los informes suelen ser de solo información y no participan en el modelo Guardar. Sin embargo, pueden incluir interacción, como vínculos a otra información relevante o secciones que se expanden y contraen.
- La mayoría de los comandos en la superficie deben ser hipervínculos, no botones.
- El diseño debe incluir un encabezado y seguir las directrices de diseño de informe estándar.

Paneles

- Los paneles no tienen un modelo de interacción en sí mismos, pero sirven como un medio para ofrecer una variedad de otras herramientas.
- No participan en el modelo Guardar.
- Los usuarios deben poder interactuar con los controles utilizando solo el teclado, ya sea activando el editor y tabulando a través de los controles o utilizando mnemotécnicas estándar.

Diálogos

Introducción

Los cuadros de diálogo de Visual Studio normalmente deben admitir una unidad discreta del trabajo del usuario y, a continuación, se descartan.

Si ha determinado que necesita un diálogo, tiene tres opciones, en orden de preferencia:

1. Integre las características en uno de los cuadros de diálogo compartidos de Visual Studio.
2. Cree su propio cuadro de diálogo utilizando un patrón que se encuentra en un cuadro de diálogo similar existente.
3. Cree un nuevo cuadro de diálogo, siguiendo las directrices de interacción y diseño.

En esta sección se describe cómo elegir el patrón de cuadro de diálogo correcto en los flujos de trabajo de Visual Studio y las convenciones comunes para el diseño de cuadros de diálogo.

Temas

Los cuadros de diálogo de Visual Studio siguen uno de los dos estilos básicos:

Estándar (sin tema)

La mayoría de los cuadros de diálogo son cuadros de diálogo de utilidad estándar y deben ser destemáticos. No vuelva a crear plantillas de controles comunes ni intente crear botones o controles "modernos" estilizados. Los controles y la apariencia de chrome siguen [las directrices de interacción estándar](#) de Escritorio de Windows para los cuadros de diálogo.

Temática

Los cuadros de diálogo de "firma" de especialidad pueden ser temáticos. Los cuadros de diálogo temáticos tienen una apariencia distinta, que también tiene algunos patrones de interacción especiales asociados con el estilo. Tema de su diálogo sólo si cumple con estos requisitos:

- El cuadro de diálogo es una experiencia común que se verá y usará a menudo o por muchos usuarios (por ejemplo, el cuadro de diálogo **Nuevo proyecto**).
- El cuadro de diálogo contiene elementos destacados de la marca del producto (por ejemplo, el cuadro de diálogo Configuración de la **cuenta**).
- El cuadro de diálogo aparece como una parte integral de un flujo más grande que incluye otros cuadros de diálogo temáticos (por ejemplo, el cuadro de diálogo **Agregar servicio conectado**).
- El diálogo es una parte importante de una experiencia que desempeña un papel estratégico en la promoción o diferenciación de una versión del producto.

Al crear un cuadro de diálogo temático, utilice los colores de entorno adecuados y siga los patrones de diseño e interacción correctos. (Consulte [Diseño para Visual Studio](#).)

Diseño de cuadros de diálogo

Los cuadros de diálogo bien diseñados tienen en cuenta los siguientes elementos:

- La tarea de usuario que se admite
- El estilo de texto del cuadro de diálogo, el idioma y la terminología
- Opciones de control y convenciones de interfaz de usuario
- Especificación de diseño visual y alineación de control
- Acceso mediante el teclado

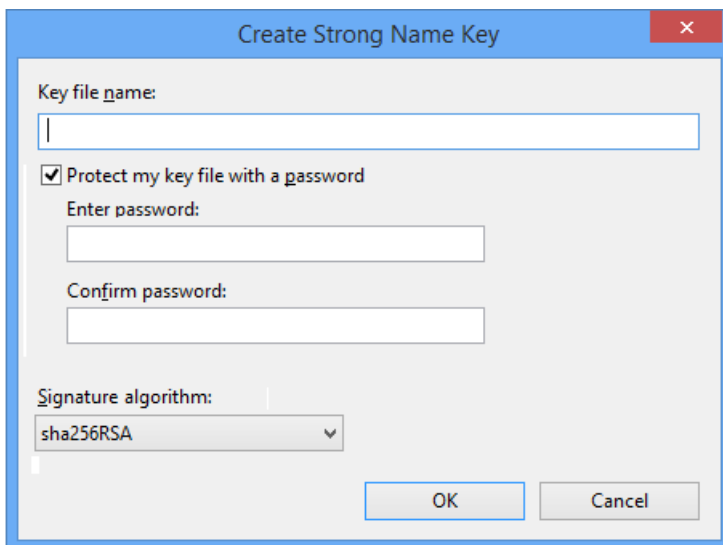
Organización de contenido

Considere las diferencias entre estos tipos básicos de diálogos:

- [Los cuadros de diálogo simples](#) presentan controles en una sola ventana modal. La presentación puede incluir variaciones de patrones de control complejos, incluido un selector de campos o una barra de iconos.
- Los [cuadros](#) de diálogo en capas se utilizan para aprovechar al máximo los bienes inmuebles de la pantalla cuando una sola parte de la interfaz de usuario comprende varios grupos de controles. Las agrupaciones del cuadro de diálogo se "capan" a través de controles de pestaña, controles de lista de navegación o botones para que el usuario pueda elegir qué agrupación ver en un momento dado.
- [Los asistentes](#) son útiles para dirigir al usuario a través de una secuencia lógica de pasos hacia la finalización de una tarea. Una serie de opciones se ofrecen en paneles secuenciales, a veces introduciendo diferentes flujos de trabajo ("ramas") dependiendo de una elección realizada en el panel anterior.

Diálogos sencillos

Un cuadro de diálogo simple es una presentación de controles en una sola ventana modal. Esta presentación puede incluir variaciones de patrones de control complejos, como un selector de campos. Para cuadros de diálogo simples, siga el diseño general estándar, así como cualquier diseño específico necesario para agrupaciones de control complejas.



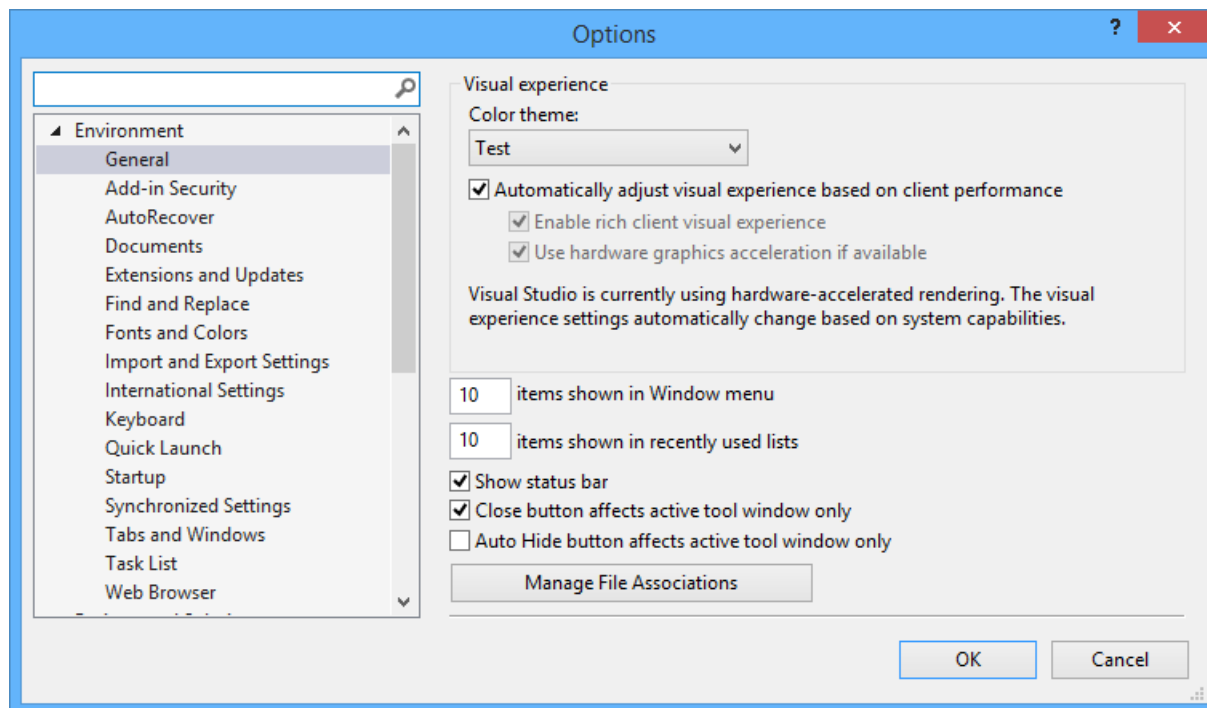
Crear clave de nombre seguro es un ejemplo de un cuadro de diálogo simple en Visual Studio.

Diálogos en capas

Los cuadros de diálogo en capas incluyen pestañas, paneles y árboles incrustados. Se utilizan para maximizar los bienes raíces cuando hay varios grupos de controles ofrecidos en una sola parte de la interfaz de usuario. Las agrupaciones están en capas para que el usuario pueda elegir qué agrupación ver en cualquier momento.

En el caso más sencillo, el mecanismo para cambiar entre agrupaciones es un control de pestañas. Hay varias alternativas disponibles. Consulte Priorización y capas para obtener información sobre cómo elegir el estilo más adecuado.

El cuadro de diálogo **Herramientas > Opciones** de herramientas es un ejemplo de un cuadro de diálogo en capas que utiliza un árbol incrustado:



Herramientas > Opciones es un ejemplo de un cuadro de diálogo en capas en Visual Studio.

Asistentes

Los asistentes son útiles para dirigir al usuario a través de una secuencia lógica de pasos en la finalización de una tarea. Se ofrecen una serie de opciones en paneles secuenciales y el usuario debe continuar a través de cada paso antes de continuar con el siguiente. Una vez que hay suficientes valores predeterminados disponibles, el botón **Finalizar** está habilitado.

Los asistentes modales se utilizan para tareas que:

- Contienen bifurcación, donde se ofrecen diferentes rutas dependiendo de las opciones del usuario
- Contienen dependencias entre pasos, donde los pasos subsiguientes dependen de la entrada del usuario de los pasos anteriores
- Son lo suficientemente complejos como para que la interfaz de usuario se use para explicar las opciones ofrecidas y los posibles resultados en cada paso
- Son transaccionales, lo que requiere un conjunto de pasos que deben completarse en su totalidad antes de que se confirmen los cambios

Convenciones comunes

Para lograr un diseño y funcionalidad óptimos con los cuadros de diálogo, siga estas convenciones sobre el tamaño del cuadro de diálogo, la posición, los estándares, la configuración y alineación del control, el texto de la interfaz de usuario, las barras de título, los botones de control y las teclas de acceso.

Para obtener instrucciones específicas del diseño, vea [Diseño para Visual Studio](#).

Tamaño

Los cuadros de diálogo deben ajustarse a una resolución de pantalla mínima de 1024 x 768, y el tamaño del cuadro de diálogo inicial no debe superar los 900 x 700 píxeles. Los cuadros de diálogo pueden ser redimensionables, pero no es un requisito.

Hay dos recomendaciones para los cuadros de diálogo redimensionables:

1. Que un tamaño mínimo es un definido para el cuadro de diálogo que optimizará para el conjunto de controles sin recorte y se ajustará para adaptarse al crecimiento razonable de la localización.
2. Que el tamaño escalado por el usuario persiste de una sesión a otra. Por ejemplo, si el usuario escala un cuadro de diálogo al 150%, se mostrará un inicio posterior del cuadro de diálogo al 150%.

Posición

Los cuadros de diálogo deben aparecer centrados en el IDE en el primer lanzamiento. No es necesario conservar la última posición de los cuadros de diálogo no redimensionables, por lo que aparecerán centrados en los lanzamientos posteriores.

Para los cuadros de diálogo de tamaño variable, el tamaño debe conservarse en los lanzamientos posteriores. Para los cuadros de diálogo modales de tamaño variable, no es necesario conservar la posición. Mostrarlos centrados en el IDE evita la posibilidad de que el cuadro de diálogo aparezca en una posición impredecible o inutilizable cuando la configuración de visualización del usuario ha cambiado.

Para los cuadros de diálogo no posicionados que se pueden cambiar de posición, la posición del usuario debe mantenerse en los lanzamientos posteriores, ya que el cuadro de diálogo se puede usar con frecuencia como parte integral de un flujo de trabajo más grande.

Cuando los cuadros de diálogo deben generar otros cuadros de diálogo, el cuadro de diálogo superior debe conectarse en cascada a la derecha y hacia abajo desde el elemento primario para que sea obvio para el usuario que ha navegado a un nuevo lugar.

Modalidad

Ser modal significa que los usuarios deben completar o cancelar el cuadro de diálogo antes de continuar. Dado que los cuadros de diálogo modales impiden que el usuario interactúe con otras partes del entorno, el flujo de tareas de la entidad debe usarlos con la mayor moderación posible. Cuando es necesaria una operación modal, Visual Studio tiene una serie de cuadros de diálogo compartidos en los que puede integrar las características. Si debe crear un nuevo cuadro de diálogo, siga el patrón de interacción de un cuadro de diálogo existente con una funcionalidad similar.

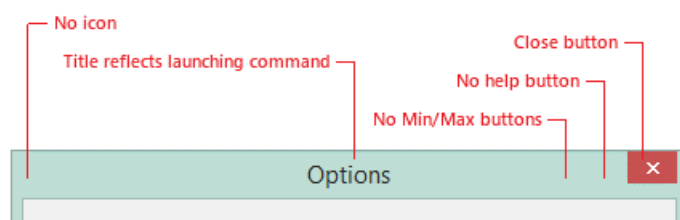
Cuando los usuarios necesitan realizar dos actividades a la vez, como **Buscar** y **reemplazar** al escribir código nuevo, el cuadro de diálogo debe ser no codificador para que el usuario pueda cambiar fácilmente entre ellos. Visual Studio generalmente usa ventanas de herramientas para este tipo de tarea vinculada compatible con el editor.

Configuración de control

Sea coherente con las configuraciones de control existentes que logran lo mismo en Visual Studio.

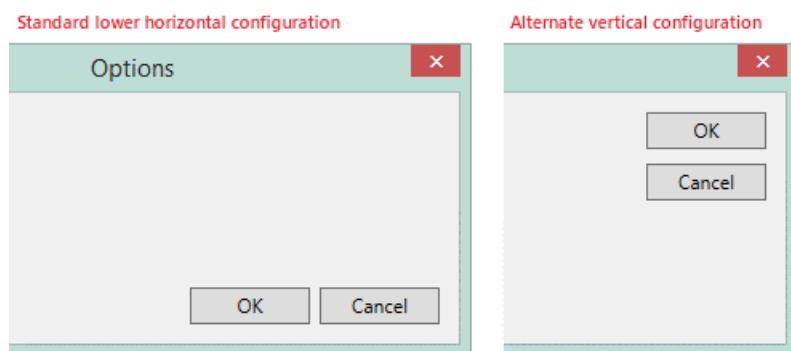
Barras de título

- El texto de la barra de título debe reflejar el nombre del comando que lo inició.
- No se debe utilizar ningún icono en las barras de título del cuadro de diálogo. En los casos en que el sistema lo requiera, use el logotipo de Visual Studio.
- Los cuadros de diálogo no deben tener botones de minimizar o maximizar.
- Los botones de ayuda de la barra de título han quedado obsoletos. No los agregue a nuevos cuadros de diálogo. Cuando existen, deben iniciar un tema de Ayuda que sea conceptualmente relevante para la tarea.



Botones de control

En general, los botones **Aceptar**, **Cancelar** y **Ayuda** deben organizarse horizontalmente en la esquina inferior derecha del cuadro de diálogo. La pila vertical alternativa se permite si un cuadro de diálogo tiene varios otros botones en la parte inferior del cuadro de diálogo que presentarían confusión visual con los botones de control.



Configuraciones aceptables para botones de control en los cuadros de diálogo de Visual Studio

El cuadro de diálogo debe incluir un botón de control predeterminado. Para determinar el mejor comando que se utilizará como valor predeterminado, elija una de las siguientes opciones (listadas en orden de prioridad):

- Elija el comando más seguro y seguro como valor predeterminado. Esto significa elegir el comando más probable para evitar la pérdida de datos y evitar el acceso involuntario al sistema.
- Si la pérdida de datos y la seguridad no son factores, elija el comando predeterminado en función de la comodidad. Incluir el comando más probable como predeterminado mejorará el flujo de trabajo del usuario cuando el cuadro de diálogo admita tareas frecuentes o repetitivas.

Evite elegir una acción destructiva permanente para el comando predeterminado. Si tal comando está presente, elija un comando más seguro como el valor por defecto en su lugar.

Claves de acceso

No utilice las teclas de acceso para los botones **Aceptar**, **Cancelar** o **Ayuda**. Estos botones se asignan a las teclas de método abreviado de forma predeterminada:

NOMBRE DEL BOTÓN	MÉTODOS ABREVIADOS DE TECLADO
Aceptar	Escriba
Cancelar	Esc
Ayuda	F1

Imágenes

Utilice imágenes con moderación en los cuadros de diálogo. No utilice iconos grandes en los cuadros de diálogo simplemente para usar espacio. Utilice imágenes solo si son una parte importante de la transmisión del mensaje al usuario, como iconos de advertencia o animaciones de estado.

Priorización y estratificación

Priorizar la interfaz de usuario

Puede ser necesario llevar ciertos elementos de la interfaz de usuario a la vanguardia y colocar un comportamiento y opciones más avanzados (incluidos los comandos oscuros) en los cuadros de diálogo. Lleve la funcionalidad de uso común a la vanguardia haciendo espacio para ella y haciéndola visible de forma predeterminada en la interfaz de usuario con una etiqueta de texto cuando se muestra el cuadro de diálogo.

Capas de la interfaz de usuario

Si ha determinado que un cuadro de diálogo es necesario, pero la funcionalidad relacionada que desea presentar al usuario va más allá de lo que se puede mostrar en un cuadro de diálogo simple, entonces debe poner la capa de la interfaz de usuario. Los métodos de capas más comunes que usa Visual Studio son pestañas y pasillos o paneles. En algunos casos, las regiones que pueden expandirse y contraerse pueden ser adecuadas. Por lo general, no se recomienda la interfaz de usuario adaptable en Visual Studio.

Hay ventajas y desventajas para diferentes métodos de creación de capas de la interfaz de usuario a través de controles de tipo pestaña. Revise la lista siguiente para asegurarse de que está eligiendo una técnica de capas que sea adecuada a su situación.

Tabulación

MECANISMO DE CONMUTACIÓN	VENTAJAS Y USO ADECUADO	DESVENTAJAS Y USO INAPROPIADO
Control Tab	Agrupar lógicamente páginas de diálogo en conjuntos relacionados Es útil para menos de cinco (o el número de pestañas que caben en una fila en el cuadro de diálogo) páginas de controles relacionados en el cuadro de diálogo Las etiquetas de tabulación deben ser cortas: una o dos palabras que pueden identificar fácilmente el contenido Un estilo de diálogo común del sistema Ejemplo: **Propiedades > ** del elemento del Explorador de archivos	Hacer etiquetas cortas descriptivas puede ser difícil Generalmente no escala más allá de cinco pestañas en un cuadro de diálogo Inapropiado si tiene demasiadas pestañas para una fila (utilice una técnica de capas alternativa) No extensible
Navegación de la barra lateral	Dispositivo de conmutación simple que puede acomodar más categorías que pestañas Lista plana de categorías (sin jerarquía) Extensible Ejemplo: Personalizar... Añadir > comando	No es un buen uso del espacio horizontal si hay menos de tres grupos La tarea podría ser más adecuada para una lista desplegable
Tree (control)	Permite categorías ilimitadas Permite agrupar y/o jerarquizar categorías Extensible Ejemplo: ** > Opciones de herramientas**	Las jerarquías muy anidadas pueden causar un desplazamiento horizontal excesivo Visual Studio tiene una sobreabundancia de vistas de árbol

MECANISMO DE CONMUTACIÓN	VENTAJAS Y USO ADECUADO	DESVENTAJAS Y USO INAPROPIADO
Asistente	Ayuda con la finalización de tareas guiando al usuario a través de pasos secuenciales basados en tareas: el asistente representa una tarea de alto nivel y los paneles individuales representan las subtarefas necesarias para realizar la tarea general Resulta útil cuando la tarea cruza los límites de Ui, como cuando el usuario tendría que usar varios editores y ventanas de herramientas para completar la tarea Resulta útil cuando la tarea requiere bifurcación Resulta útil cuando la tarea contiene dependencias entre pasos Resulta útil cuando varias tareas similares con una bifurcación de decisión se pueden presentar en un cuadro de diálogo para reducir el número de cuadros de diálogo similares diferentes	Inapropiado para cualquier tarea que no requiera un flujo de trabajo secuencial Los usuarios pueden sentirse abrumados y confundidos por un asistente con demasiados pasos Los magos han limitado intrínsecamente los bienes raíces de la pantalla

Pasillos o tableros

Los pasillos y paneles son cuadros de diálogo o paneles que sirven como puntos de inicio a otros cuadros de diálogo y ventanas. El "hallway" bien diseñado inmediatamente expone solo las opciones, comandos y configuraciones más comunes, lo que permite al usuario realizar fácilmente tareas comunes. Al igual que un pasillo del mundo real proporciona puertas para acceder a las habitaciones detrás de ellos, aquí la interfaz de usuario menos común se recoge en "habitaciones" separadas (a menudo otros diálogos) de funcionalidad relacionada a la que se puede acceder desde el pasillo principal.

Como alternativa, una interfaz de usuario que ofrece toda la funcionalidad disponible en una sola colección en lugar de refactorizar la funcionalidad menos común en ubicaciones independientes es simplemente un panel.



Change the settings for messages you create and receive.

Compose messages



Change the editing settings for messages.

Editor Options...

Compose messages in this format: HTML



☐ Always check spelling before sending

Spelling and Autocorrect...

☒ Ignore original message text in reply or forward



Create or modify signatures for messages.

Signatures...



Use stationery to change default fonts and styles, colors, and backgrounds.

Stationery and Fonts...

Outlook panes



Customize how items are marked as read when using the Reading Pane.

Reading Pane...

Concepto de pasillo para exponer la interfaz de usuario adicional en Outlook

Mostrar u ocultar la interfaz de usuario en función del uso o de la experiencia autoinformada de un usuario es otra forma de presentar la interfaz de usuario necesaria mientras se ocultan otras partes. Esto no se recomienda en Visual Studio, ya que los algoritmos para decidir cuándo mostrar u ocultar la interfaz de usuario pueden ser complicados y las reglas siempre serán incorrectas para algún conjunto de casos.

Proyectos

Proyectos en el Explorador de soluciones

La mayoría de los proyectos se clasifican como basados en referencias, basados en directorios o mixtos. Los tres tipos de proyectos se admiten simultáneamente en el Explorador de soluciones. La raíz de la experiencia del usuario en el trabajo con proyectos tiene lugar dentro de esta ventana. Aunque los nodos de proyecto diferentes son proyectos de tipo de referencia, directorio o modo mixto, hay un patrón de interacción común que se debe aplicar como punto de partida antes de divergir en patrones de usuario específicos del proyecto.

Los proyectos siempre deben:

- Admite la capacidad de agregar carpetas de proyecto para organizar el contenido del proyecto
- Mantener un modelo coherente para la persistencia del proyecto

Los proyectos también deben mantener modelos de interacción coherentes para:

- Eliminación de elementos de proyecto
- Guardar documentos
- Edición de propiedades del proyecto
- Edición del proyecto en una vista alternativa
- Operaciones de arrastrar y soltar

Modelo de interacción de arrastrar y soltar

Normalmente, los proyectos se clasifican como basados en referencias (capaces de conservar solo las referencias a elementos de proyecto en el almacenamiento), basados en directorios (capaces de conservar solo los elementos del proyecto almacenados físicamente en la jerarquía de un proyecto) o mixtos (capaces de conservar referencias o elementos físicos). El IDE da cabida a los tres tipos de proyectos simultáneamente en el Explorador de **soluciones**.

Desde una perspectiva de arrastrar y colocar, las siguientes características deben aplicarse a cada tipo de proyecto en el Explorador de **soluciones**:

- **Proyecto basado en referencias:** El punto clave es que el proyecto está arrastrando alrededor de una referencia a un elemento en el almacenamiento. Cuando un proyecto basado en referencias actúa como origen para una operación de movimiento, solo debe quitar la referencia al elemento del proyecto. El elemento no debe eliminarse realmente del disco duro. Cuando un proyecto basado en referencias actúa como destino para una operación de movimiento (o copia), debe agregar una referencia al elemento de origen original sin realizar una copia privada del elemento.
- **Proyecto basado en directorios:** Desde un punto de vista de arrastrar y colocar, el proyecto se arrastra alrededor del elemento físico en lugar de una referencia. Cuando un proyecto basado en directorios actúa como origen para una operación de movimiento, debe terminar eliminando el elemento físico del disco duro, así como quitándolo del proyecto. Cuando un proyecto basado en directorios actúa como destino para una operación de movimiento (o copia), debe realizar una copia del elemento de origen en su ubicación de destino.
- **Proyecto mixto:** Desde un punto de vista de arrastrar y colocar, el comportamiento de este tipo de proyecto se basa en la naturaleza del elemento que se arrastra (ya sea una referencia a un elemento en el almacenamiento o el propio elemento). El comportamiento correcto para las referencias y los elementos

físicos se describe anteriormente.

Si solo hubiera un tipo de proyecto en el **Explorador** de soluciones, las operaciones de arrastrar y colocar serían sencillas. Dado que cada sistema de proyecto tiene la capacidad de definir su propio comportamiento de arrastrar y colocar, se deben seguir ciertas directrices (basadas en el comportamiento de arrastrar y colocar del Explorador de Windows) para garantizar una experiencia de usuario predecible:

- Una operación de arrastre sin modificar en el **Explorador** de soluciones (cuando no se mantienen pulsadas las teclas Ctrl ni Mayús) debe dar lugar a una operación de movimiento.
- La operación de arrastrar y mayúsculas también debería dar lugar a una operación de movimiento.
- La operación Ctrl-arrastrar debe dar lugar a una operación de copia.
- Los sistemas de proyectos mixtos y basados en referencias admiten la noción de agregar un vínculo (o referencia) al elemento de origen. Cuando estos proyectos son el destino de una operación de arrastrar y colocar (cuando Ctrl + **Mayús** se mantiene presionada), debe dar lugar a una referencia al elemento que se agrega al proyecto

No todas las operaciones de arrastrar y colocar son razonables en combinaciones de proyectos basados en referencias, basados en directorios y mixtos. En particular, es problemático pretender permitir una operación de movimiento entre un proyecto de origen basado en directorios y un proyecto de destino basado en referencias porque el proyecto basado en directorios de origen tendrá que eliminar el elemento de origen al finalizar el movimiento. El proyecto basado en referencias de destino terminaría con una referencia a un elemento eliminado.

También es engañoso pretender permitir una operación de copia entre estos tipos de proyectos porque el proyecto basado en referencias de destino no debe hacer una copia independiente del elemento de origen. De forma similar, No se debe permitir que se aplase Ctrl + Mayús a un proyecto de destino basado en directorios porque un proyecto basado en directorios no puede conservar las referencias. En los casos en los que no se admite la operación de arrastrar y colocar, el IDE debe no permitir la colocación y mostrar al usuario el cursor sin colocar (que se muestra en la tabla de punteros siguiente).

Para implementar correctamente el comportamiento de arrastrar y colocar, el proyecto de origen del arrastre debe comunicar su naturaleza al proyecto de destino. (Por ejemplo, ¿está basado en referencias o directorios?) Esta información se indica mediante el formato del portapapeles que ofrece el origen. Como origen de una operación de arrastrar (o `CF_VSREFPROJECTITEMS` `CF_VSSTGPROJECTITEMS` copia del portapapeles) un proyecto debe ofrecer uno o respectivamente, dependiendo de si el proyecto está basado en referencias o en directorios. Ambos formatos tienen el mismo contenido de datos, que es similar al formato de `CF_HDROP` Windows, `NULL` excepto que `Projref` las listas de `IVsSolution::GetProjrefOfItem` `::GetProjrefOfProject` cadenas, en lugar de ser nombres de archivo, son una lista de cadenas de doble terminación (según se devuelva desde o según corresponda).

Como destino de una operación de colocación (o pegar portapapeles), un proyecto debe aceptar ambos `CF_VSREFPROJECTITEMS` y `CF_VSSTGPROJECTITEMS`, aunque el control exacto de la operación de arrastrar y colocar varía en función de la naturaleza del proyecto de destino y el proyecto de origen. El proyecto fuente declara su naturaleza `CF_VSREFPROJECTITEMS` `CF_VSSTGPROJECTITEMS` por si ofrece o . El objetivo de la caída entiende su propia naturaleza y, por lo tanto, tiene suficiente información para tomar decisiones sobre si se debe realizar un movimiento, una copia o un vínculo. El usuario también modifica qué operación de arrastrar y colocar se debe realizar presionando las teclas Ctrl, Mayús o Ctrl y Mayús. Es importante que el destino de colocación indique correctamente `DragEnter` qué `DragOver` operación se realizará con antelación en sus métodos y. El **Explorador** de soluciones sabe automáticamente si el proyecto de origen y el proyecto de destino son el mismo proyecto.

No se admite específicamente arrastrar elementos de proyecto entre instancias de Visual Studio (por ejemplo, de una instancia de devenv.exe a otra). El **Explorador** de soluciones también deshabilita directamente esto.

El usuario siempre debe ser capaz de determinar el efecto de una operación de arrastrar y colocar seleccionando un elemento, arrastrándolo a la ubicación de destino y observando cuál de los siguientes punteros del mouse

aparece antes de quitar el elemento:

PUNTERO	GET-HELP	DESCRIPCIÓN
	Sin caída	El elemento no se puede quitar en la ubicación especificada.
	Copiar	El elemento se copiará en la ubicación de destino.
	Move	El elemento se moverá a la ubicación de destino.
	Agregar referencia	Se agregará una referencia al elemento seleccionado a la ubicación de destino.

Proyectos basados en referencias

En la tabla siguiente se resumen las operaciones de arrastrar y colocar (así como cortar/copiar/pegar) que se deben realizar en función de la naturaleza del elemento de origen y las teclas modificadoras presionadas para los proyectos de destino basados en referencia:

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Sin modificador	Acción	Move	Vínculo
Sin modificador	Destino	Añade referencia al elemento original	Añade referencia al elemento original
Sin modificador	Source	Elimina la referencia al elemento original	Conserva el artículo original
Sin modificador	Resultado	DROPEFFECT_MOVE se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_LINK se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Mayús+Arrastrar	Acción	Move	Sin caída
Mayús+Arrastrar	Destino	Añade referencia al elemento original	Sin caída
Mayús+Arrastrar	Source	Elimina la referencia al elemento original	Sin caída
Mayús+Arrastrar	Resultado	DROPEFFECT_MOVE se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	Sin caída
Ctrl+Arrastrar	Acción	Copiar	Sin caída

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Ctrl+Arrastrar	Destino	Añade referencia al elemento original	Sin caída
Ctrl+Arrastrar	Source	Conserva la referencia al artículo original	Sin caída
Ctrl+Arrastrar	Resultado	DROPEFFECT_COPY se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	Sin caída
Ctrl+Mayús+Arrastrar	Acción	Vínculo	Vínculo
Ctrl+Mayús+Arrastrar	Destino	Añade referencia al elemento original	Añade referencia al elemento original
Ctrl+Mayús+Arrastrar	Source	Conserva la referencia al artículo original	Conserva el artículo original
Ctrl+Mayús+Arrastrar	Resultado	DROPEFFECT_LINK se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_LINK se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Ctrl+Mayús+Arrastrar	Nota:	Igual que el comportamiento de arrastrar y colocar para los accesos directos en el Explorador de Windows.	
Cortar/Pegar	Acción	Move	Vínculo
Cortar/Pegar	Destino	Añade referencia al elemento original	Añade referencia al elemento original
Cortar/Pegar	Source	Conserva la referencia al artículo original	Conserva el artículo original
Cortar/Pegar	Resultado	El artículo permanece en la ubicación original en el almacenamiento	El artículo permanece en la ubicación original en el almacenamiento
Copiar/Pegar	Acción	Copiar	Vínculo
Copiar/Pegar	Source	Añade referencia al elemento original	Añade referencia al elemento original
Copiar/Pegar	Resultado	Conserva la referencia al artículo original	Conserva el artículo original

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Copiar/Pegar	Acción	El artículo permanece en la ubicación original en el almacenamiento	El artículo permanece en la ubicación original en el almacenamiento

Proyectos basados en directorios

En la tabla siguiente se resumen las operaciones de arrastrar y colocar (así como cortar/copiar/pegar) que se deben realizar en función de la naturaleza del elemento de origen y las teclas modificadoras presionadas para los proyectos de destino basados en directorios:

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Sin modificador	Acción	Move	Move
Sin modificador	Destino	Copia el elemento en la ubicación de destino	Copia el elemento en la ubicación de destino
Sin modificador	Source	Elimina la referencia al elemento original	Elimina la referencia al elemento original
Mayús+Arrastrar	Acción	Move	Move
Mayús+Arrastrar	Destino	Copia el elemento en la ubicación de destino	Copia el elemento en la ubicación de destino
Mayús+Arrastrar	Source	Elimina la referencia al elemento original	Elimina el elemento de la ubicación original
Mayús+Arrastrar	Resultado	DROPEFFECT_MOVE se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_MOVE se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Ctrl+Arrastrar	Acción	Copiar	Copiar
Ctrl+Arrastrar	Destino	Copia el elemento en la ubicación de destino	Copia el elemento en la ubicación de destino
Ctrl+Arrastrar	Source	Conserva la referencia al artículo original	Conserva la referencia al artículo original
Ctrl+Arrastrar	Resultado	DROPEFFECT_COPY se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_COPY se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Ctrl+Mayús+Arrastrar		Sin caída	Sin caída

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Cortar/Pegar	Acción	Move	Move
Cortar/Pegar	Destino	Copia el elemento en la ubicación de destino	Copia el elemento en la ubicación de destino
Cortar/Pegar	Source	Elimina la referencia al elemento original	Elimina el elemento de la ubicación original
Cortar/Pegar	Resultado	El artículo permanece en la ubicación original en el almacenamiento	El artículo se elimina de la ubicación original en el almacenamiento
Copiar/Pegar	Acción	Copiar	Copiar
Copiar/Pegar	Destino	Añade referencia al elemento original	Copia el elemento en la ubicación de destino
Copiar/Pegar	Source	Conserva el artículo original	Conserva el artículo original
Copiar/Pegar	Resultado	El artículo permanece en la ubicación original en el almacenamiento	El artículo permanece en el almacenamiento original de ins de ubicación

Proyectos mixtos

En la tabla siguiente se resumen las operaciones de arrastrar y colocar (así como cortar/copiar/pegar) que se deben realizar en función de la naturaleza del elemento de origen y las teclas modificadoras presionadas para proyectos de destino mixto:

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Sin modificador	Acción	Move	Move
Sin modificador	Destino	Añade referencia al elemento original	Copia el elemento en la ubicación de destino
Sin modificador	Source	Elimina la referencia al elemento original	Elimina la referencia al elemento original
Sin modificador	Resultado	<code>DROPEFFECT_ MOVE</code> se devuelve como <code>::Drop</code> acción y el artículo permanece en la ubicación original en el almacenamiento	<code>DROPEFFECT_ MOVE</code> se devuelve como <code>::Drop</code> acción y el elemento se elimina de la ubicación original en el almacenamiento
Mayús+Arrastrar	Acción	Move	Move
Mayús+Arrastrar	Destino	Añade referencia al elemento original	Copia el elemento en la ubicación de destino

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Mayús+Arrastrar	Source	Elimina la referencia al elemento original	Elimina el elemento de la ubicación original
Mayús+Arrastrar	Resultado	DROPEFFECT_ MOVE se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_ MOVE se devuelve como ::Drop acción y el elemento se elimina de la ubicación original en el almacenamiento
Ctrl+Arrastrar	Acción	Copiar	Copiar
Ctrl+Arrastrar	Destino	Añade referencia al elemento original	Copia el elemento en la ubicación de destino
Ctrl+Arrastrar	Source	Conserva la referencia al artículo original	Conserva el artículo original
Ctrl+Arrastrar	Resultado	DROPEFFECT_ COPY se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_ COPY se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Ctrl+Mayús+Arrastrar	Acción	Vínculo	Vínculo
Ctrl+Mayús+Arrastrar	Destino	Añade referencia al elemento original	Agrega referencia al elemento de origen original
Ctrl+Mayús+Arrastrar	Source	Conserva la referencia al artículo original	Conserva el artículo original
Ctrl+Mayús+Arrastrar	Resultado	DROPEFFECT_ LINK se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento	DROPEFFECT_ LINK se devuelve como ::Drop acción y el artículo permanece en la ubicación original en el almacenamiento
Cortar/Pegar	Acción	Move	Move
Cortar/Pegar	Destino	Copia el elemento en la ubicación de destino	Copia el elemento en la ubicación de destino
Cortar/Pegar	Source	Elimina la referencia al elemento original	Elimina el elemento de la ubicación original
Cortar/Pegar	Resultado	El artículo permanece en la ubicación original en el almacenamiento	El artículo se elimina de la ubicación original en el almacenamiento
Copiar/Pegar	Acción	Copiar	Copiar

MODIFICADOR	CATEGORY	ELEMENTO FUENTE: REFERENCIA/ENLACE	ELEMENTO DE ORIGEN: ELEMENTO CF_HDROP FÍSICO O SISTEMA DE ARCHIVOS ()
Copiar/Pegar	Destino	Añade referencia al elemento original	Copia el elemento en la ubicación de destino
Copiar/Pegar	Source	Conserva el artículo original	Conserva el artículo original
Copiar/Pegar	Resultado	El artículo permanece en la ubicación original en el almacenamiento	El artículo permanece en la ubicación original en el almacenamiento

Estos detalles deben tenerse en cuenta al implementar el arrastre en el Explorador de **soluciones**:

- Diseño para varios escenarios de selección.
- Los nombres de archivo (ruta de acceso completa) deben ser únicos en todo el proyecto de destino o no se debe permitir la colocación.
- Los nombres de carpeta deben ser únicos (sin distinción entre mayúsculas y minúsculas) en el nivel en el que se quitan.
- Hay diferencias de comportamiento entre los archivos que están abiertos o cerrados en el momento de arrastrar (no se menciona en los escenarios anteriores).
- Los archivos de nivel superior se comportan de forma ligeramente diferente que los archivos de las carpetas.

Otro problema a tener en cuenta es cómo controlar las operaciones de movimiento en elementos que tienen diseñadores o editores abiertos. El comportamiento esperado es el siguiente (esto se aplica a todos los tipos de proyecto):

1. Si el editor/diseñador abierto no tiene ningún cambio no guardado, la ventana del editor/diseñador debe cerrarse silenciosamente.
2. Si el editor/diseñador abierto tiene cambios no guardados, el origen del arrastre debe esperar a que se produzca la colocación y, a continuación, pedir al usuario que guarde los cambios no confirmados en los documentos abiertos antes de cerrar la ventana con un mensaje similar al siguiente:

```
=====
One or more open documents have unsaved changes.
Do you want to save uncommitted changes before proceeding?
[Yes] [No] [Cancel]
=====
```

Esto da al usuario la oportunidad de guardar el trabajo en curso antes de que el destino realice sus copias. Se ha `IVsHierarchyDropDataSource2::OnBeforeDropNotify` agregado un nuevo método para habilitar este control.

A continuación, el destino copiará el estado del elemento tal como está en el almacenamiento (sin incluir los cambios no guardados en el editor si el usuario eligió **No**). Una vez que el destino

`IVsHierarchyDropDataSource::Drop` ha completado su copia (en), el origen tiene `IVsHierarchyDropDataSource::OnDropNotify` la oportunidad de completar la parte de eliminación de la operación de movimiento (en).

Los editores con cambios no guardados deben dejarse abiertos. Para aquellos documentos con cambios no guardados, esto significa que se realizará la parte de copia de la operación de movimiento, pero se anulará la parte de eliminación. En un escenario de selección múltiple cuando el usuario elige **No**, los documentos con cambios no

guardados no deben cerrarse ni quitarse, pero los que no tienen cambios no guardados deben cerrarse y quitarse.

Patrones de control comunes para Visual Studio

06/04/2020 • 33 minutes to read • [Edit Online](#)

Controles comunes

Información general

Los controles comunes conforman la mayoría de la interfaz de usuario en Visual Studio. Los controles más comunes utilizados en la interfaz de Visual Studio deben seguir las directrices de interacción de escritorio de [Windows](#). Este tema es específico de Visual Studio y cubre situaciones especiales o detalles que aumentan esas directrices de Windows.

Controles comunes en este tema

- [Barras de desplazamiento](#)
- [Campos de entrada](#)
- [Cuadros combinados y listas desplegables](#)
- [Casillas](#)
- [Botones de radio](#)
- [Marcos de grupo](#)
- [Controles de texto](#)
- [Botones e hipervínculos](#)
- [Vistas de árbol](#)

Estilo visual

Lo primero que hay que tener en cuenta al aplicar estilos a los controles es si los controles se usarán en la interfaz de usuario temática. Los controles de la interfaz de usuario estándar no son de interfaz de usuario y deben seguir el estilo normal de escritorio de [Windows](#), lo que significa que no se vuelven a crear plantillas y deben aparecer en su apariencia de control predeterminada.

- **Diálogos estándar (utilidad):** no temáticos. No vuelvas a plantillar. Utilice los valores predeterminados de estilo de control básico.
- **Ventanas de herramientas, editores de documentos, superficies de diseño y cuadros de diálogo temáticos:** Utilice la apariencia temática especializada utilizando el servicio de color.

Barras de desplazamiento

Las barras de desplazamiento deben seguir patrones de [interacción comunes para las barras](#) de desplazamiento de Windows a menos que se aumenten con información de contenido, como en el editor de código.

Campos de entrada

Para el comportamiento de interacción típico, siga las directrices de escritorio de [Windows para cuadros de texto](#).

Estilo visual

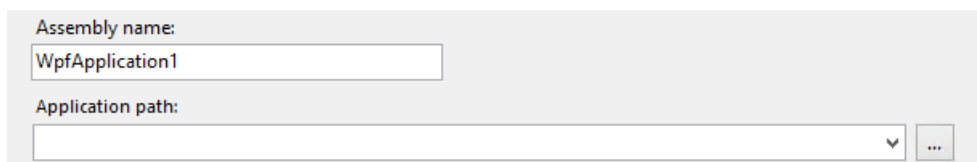
- Los campos de entrada no deben aplicar estilo en los cuadros de diálogo de utilidad. Utilice el estilo básico intrínseco al control.
- Los campos de entrada temáticos solo deben utilizarse en cuadros de diálogo temáticos y ventanas de herramientas.

Interacciones especializadas

- Los campos de solo lectura tendrán un fondo gris (deshabilitado) pero un primer plano predeterminado (activo).
- Los campos obligatorios deben tener **<>obligatorios<>** como marcas de agua dentro de ellos. No debe cambiar el color del fondo excepto en situaciones raras.
- Validación de errores: consulte [Notificaciones y progreso para Visual Studio](#)
- Los campos de entrada deben tener un tamaño que se ajuste al contenido, no para ajustarse al ancho de la ventana en la que se muestran, ni para que coincidan arbitrariamente con la longitud de un campo largo, como una ruta de acceso. La longitud puede ser una indicación para el usuario de las limitaciones en cuanto a cuántos caracteres se permiten en el campo.

A screenshot of a dialog box with a light gray background. It contains two labels: "Assembly name:" and "Application path:". The "Assembly name:" label is followed by a text input field containing "WpfApplication1". The "Application path:" label is followed by a dropdown menu showing "(Default Icon)" and a button with three dots "...". The input field for "Assembly name:" is wider than the text it contains.

Longitud de campo de entrada incorrecta: es poco probable que el nombre sea tan largo.

A screenshot of a dialog box similar to the one above, but the "Assembly name:" input field is narrower, matching the width of the text "WpfApplication1". The "Application path:" dropdown and button remain the same.

Longitud correcta del campo de entrada: el campo de entrada es un ancho razonable para el contenido esperado.

Cuadros combinados y listas desplegables

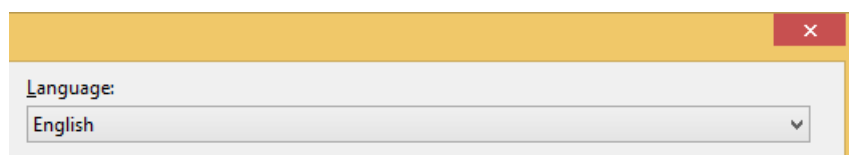
Para el comportamiento de interacción típico, siga las directrices de escritorio de [Windows para listas desplegables y cuadros combinados](#).

Estilo visual

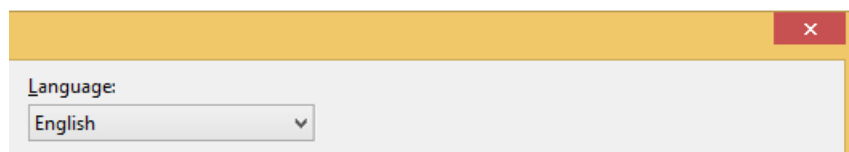
- En los cuadros de diálogo de utilidad, no vuelva a crear una plantilla del control. Utilice el estilo básico intrínseco al control.
- En la interfaz de usuario temática, los cuadros combinados y los menús desplegables siguen el tema estándar de los controles.

Diseño

Los cuadros combinados y los menús desplegables deben tener un tamaño que se ajuste al contenido, no para ajustarse al ancho de la ventana en la que se muestran, ni para que coincidan arbitrariamente con la longitud de un campo largo, como una ruta de acceso.

A screenshot of a dialog box with a yellow header bar and a red close button. It contains a label "Language:" followed by a dropdown menu showing "English". The dropdown menu is wider than the text "English", which is a design error.

Error: el ancho de la lista desplegable es demasiado largo para el contenido que se mostrará.

A screenshot of a dialog box similar to the one above, but the dropdown menu for "Language:" is narrower, matching the width of the text "English". This is the correct design.

Correcto: el tamaño de la lista desplegable permite el crecimiento de la traducción, pero no es innecesariamente largo.

Casillas de verificación

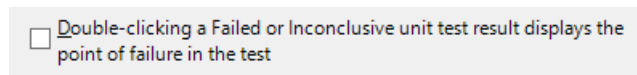
Para el comportamiento de interacción típico, siga las directrices de escritorio de [Windows para las casillas de verificación](#).

Estilo visual

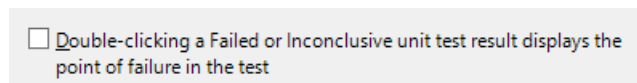
- En los cuadros de diálogo de utilidad, no vuelva a crear una plantilla del control. Utilice el estilo básico intrínseco al control.
- En la interfaz de usuario temática, las casillas de verificación siguen el tema estándar para los controles.

Interacciones especializadas

- La interacción con una casilla de verificación nunca debe abrir un cuadro de diálogo ni navegar a otra área.
- Alinee las casillas de verificación con la línea base de la primera línea de texto.



Error: la casilla de verificación se centra en el texto.



Correcto: la casilla de verificación está alineada con la primera línea del texto.

Botones de radio

Para el comportamiento de interacción típico, siga las directrices de escritorio de [Windows para botones de opción](#).

Estilo visual

En los cuadros de diálogo de utilidad, no aplicar estilo a los botones de opción. Utilice el estilo básico intrínseco al control.

Interacciones especializadas

No es necesario utilizar una trama de grupo para incluir opciones de radio, a menos que necesite mantener la distinción de grupo en un diseño ajustado.

Marcos de grupo

Para el comportamiento de interacción típico, siga las [directrices de escritorio](#) de Windows para marcos de grupo.

Estilo visual

En los cuadros de diálogo de utilidad, no estilizar marcos de grupo. Utilice el estilo básico intrínseco al control.

Diseño

- No es necesario utilizar una trama de grupo para incluir opciones de radio, a menos que necesite mantener la distinción de grupo en un diseño ajustado.
- Nunca utilice un marco de grupo para un solo control.
- A veces es aceptable usar una regla horizontal en lugar de un contenedor de marco de grupo.

Controles de texto

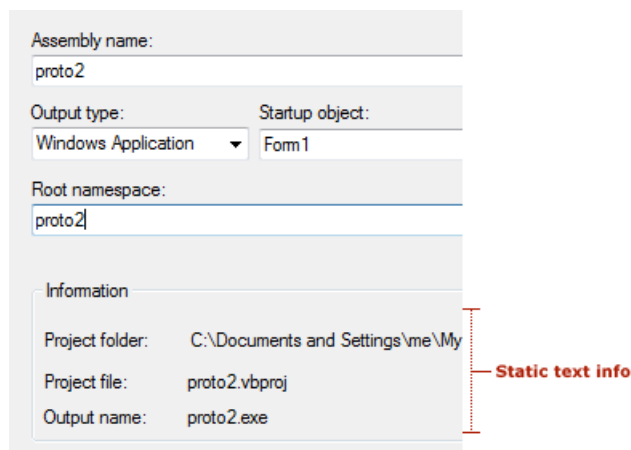
Campos de texto estático

Un campo de texto estático presenta información de solo lectura y el usuario no puede seleccionarla. Evite usarlo para cualquier texto que el usuario desee copiar en el portapapeles. Sin embargo, el texto estático de solo lectura puede cambiar para reflejar un cambio de estado. En el ejemplo siguiente, el texto estático Nombre de salida en el grupo Información cambia para reflejar los cambios realizados en el cuadro de texto Espacio de nombres raíz situado encima de él.

Hay dos maneras de mostrar información de texto estático.

El texto estático puede estar solo en un cuadro de diálogo sin ninguna contención cuando no hay conflicto de

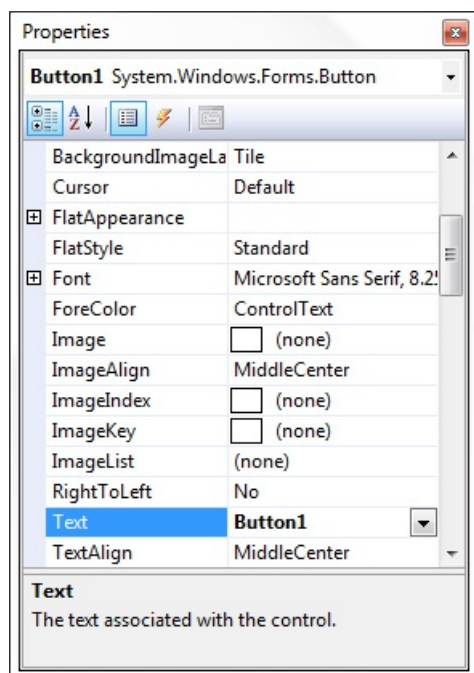
agrupación. Decida si las líneas adicionales de una caja son realmente necesarias. Un ejemplo es la visualización de una ruta de directorio bajo una sección creada por una línea de grupo, como se muestra a continuación:



The screenshot shows a 'Properties' window for a project named 'proto2'. It has sections for 'Assembly name', 'Output type' (Windows Application), 'Startup object' (Form1), and 'Root namespace' (proto2). Below these is an 'Information' section with a red bracket on the right labeled 'Static text info'. This section contains three lines of text: 'Project folder: C:\Documents and Settings\me\My', 'Project file: proto2.vbproj', and 'Output name: proto2.exe'.

Información de texto estático en controles de texto

En un cuadro de diálogo donde existen otras áreas agrupadas y la contención de la información ayudaría a la legibilidad, y cuando una sección se puede ocultar o mostrar (como en el panel de descripción de la **ventana Propiedades**) o desea ser coherente con una interfaz de usuario similar, coloque el texto estático dentro de un cuadro. Este cuadro de grupo debe ser `ButtonShadow` una sola regla y coloreado con:



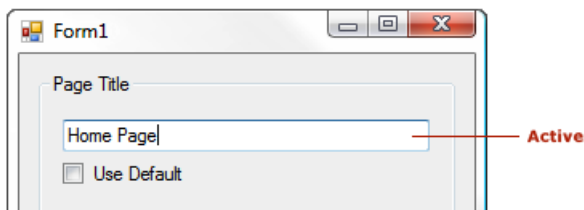
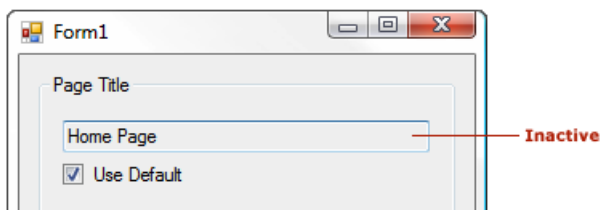
The screenshot shows the 'Properties' window for a 'Button1' control. It lists various properties like 'BackgroundImage', 'Cursor', 'FlatAppearance', 'Font', 'ForeColor', 'Image', 'ImageAlign', 'ImageIndex', 'ImageKey', 'ImageList', 'RightToLeft', 'Text', and 'TextAlign'. The 'Text' property is selected and shows the value 'Button1'. Below the list is a 'Text' section with the description 'The text associated with the control.'.

Texto estático en la ventana Propiedades

Cuadro de texto de solo lectura

Esto permite al usuario seleccionar el texto dentro del campo pero no editarlo. Estos cuadros de texto están bordeados por `ButtonShadow` el cinkel 3D habitual con un relleno.

Un cuadro de texto puede activarse (editable) cuando un usuario modifica un control asociado, como activar/desmarcar una casilla de verificación o seleccionar o anular la selección de un botón de opción. Por ejemplo, en la página ** > Opciones** de herramientas que se muestra a continuación, el cuadro de texto **Página principal** se activa cuando la casilla Usar **predeterminado** está desactivada.



Cuadro de texto de solo lectura, que muestra los estados inactivos y activos

Uso de texto en cuadros de diálogo

Directrices clave para el texto en los cuadros de diálogo:

- Las etiquetas de cuadros de texto, cuadros de lista y marcos en cuadros de diálogo sin tema comienzan con un verbo, tienen un mayúscula inicial solo en la primera palabra y terminan con dos puntos.

Los controles de texto de los cuadros de diálogo temáticos siguen [las directrices](#) de la experiencia de usuario de escritorio de Windows y no toman la puntuación final, con la excepción de los signos de interrogación en los vínculos de ayuda.

- Las etiquetas para las casillas de verificación y los botones de opción comienzan con un verbo, un mayúscula inicial solo en la primera palabra y no tienen puntuación final.
- Las etiquetas de botones, menús, elementos de menú y pestañas tienen mayúsculas iniciales en cada palabra (caso de título).
- La terminología de etiquetas debe ser coherente con etiquetas similares en otros cuadros de diálogo.
- Si es posible, pida a un escritor/editor que escriba o apruebe el texto antes de que vaya al desarrollador para su implementación.
- Todos los controles deben tener etiquetas, excepto en circunstancias especiales en las que la tabulación sea suficiente. Utilice texto auxiliar cuando sea apropiado.

Texto auxiliar

Incluido en los cuadros de diálogo para ayudar al usuario a entender el propósito del cuadro de diálogo o para indicar qué acción realizar. El texto auxiliar solo se debe usar cuando sea necesario para evitar el desorden de los cuadros de diálogo simples. Las dos variaciones del texto auxiliar son diálogo y marca de agua.

Siga las ubicaciones comunes para el texto auxiliar y sea selectivo en la introducción de nuevas áreas. Los escenarios comunes para el texto auxiliar son:

- Texto auxiliar en los cuadros de diálogo, para dar instrucciones adicionales sobre cómo interactuar con un cuadro de diálogo complejo.
- Texto de marca de agua en ventanas de herramientas o cuadros de diálogo vacíos, para explicar por qué no hay contenido visible.
- Un panel de descripción, como en la parte inferior de la **ventana Propiedades**.
- Texto de marca de agua en un editor vacío, para explicar qué acción debe realizar el usuario para empezar.

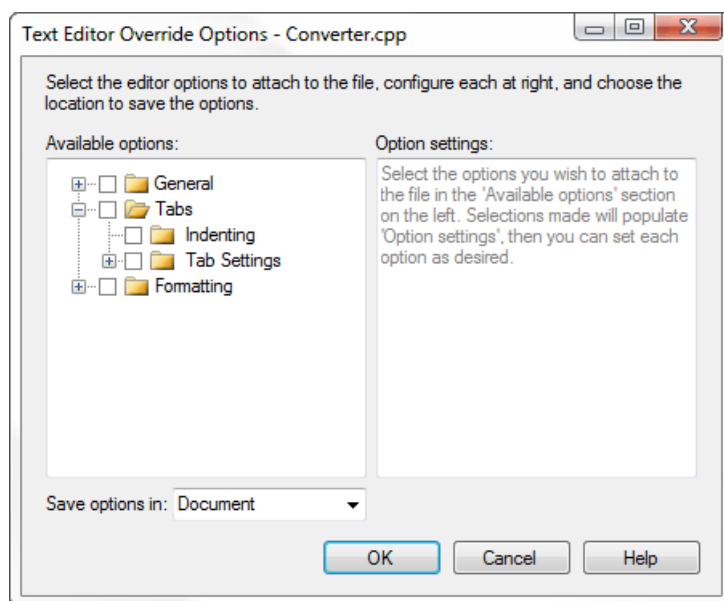
Texto del asistente del cuadro de diálogo

Un diseñador de experiencia de usuario puede ayudar a determinar cuándo el texto auxiliar es adecuado. El diseñador puede definir dónde aparece el texto auxiliar, así como su contenido general. La asistencia del usuario puede escribir/editar el texto real.

Marcas de agua

Los diálogos se benefician de pautas de marca de agua ligeramente diferentes. Dado que un cuadro de diálogo puede aparecer ocupado con muchos elementos de la interfaz de usuario (etiquetas, texto de `ButtonShadow` sugerencia, botones y otros controles de contenedor con texto), especialmente cuando aparecen en negro, las marcas de agua se destacan mejor en gris oscuro (`VSColor:`). Normalmente, una marca de agua aparece dentro de un `Window` control como un cuadro de lista con un fondo blanco (`VSColor:`).

- El texto aparece en gris `ButtonShadow` oscuro (`VSColor:`). Sin embargo, si la marca de agua aparece `ButtonFace` en un fondo gris medio u otro color (`VSColor:`) y hay preocupación sobre su legibilidad, vaya con texto negro (`VSColor: WindowText`).
- Las marcas de agua se pueden centrar o a la izquierda. Aplique reglas de diseño estándar al tomar decisiones de alineación. La marca de agua no se puede seleccionar en el fondo.



Ejemplo de texto de marca de agua

Texto específico del contexto (dinámico)

El texto dinámico se puede utilizar de dos maneras en un cuadro de diálogo o una interfaz de usuario no modificada: como una etiqueta dinámica o como contenido dinámico.

- Etiqueta dinámica: un uso común del texto dinámico se encuentra en paneles descriptivos que ofrecen más información para el elemento seleccionado, como en un cuadro de diálogo que contiene una lista de elementos y propiedades para los elementos que se muestran en una cuadrícula a la derecha. La etiqueta de la cuadrícula de propiedades puede ser dinámica para que cuando se selecciona un elemento a la izquierda, la cuadrícula de la derecha muestre información para ese elemento específico.
- Texto dinámico: puede ser útil en casos en los que necesita mostrar información específica y no información general de esta manera, pero se debe tener cuidado de no hacer uso excesivo.

Si desea que los usuarios tengan la capacidad de copiar la información, el texto dinámico debe estar en un campo de texto de solo lectura.

Botones e hipervínculos

Información general

Los botones y controles de vínculo (hipervínculos) deben seguir las instrucciones básicas de Escritorio de [Windows sobre hipervínculos](#) para el uso, la redacción, el tamaño y el espaciado.

Elegir entre botones y enlaces

Tradicionalmente, los botones se han utilizado para acciones y los hipervínculos se han reservado para la navegación. Los botones se pueden usar en todos los casos, pero el rol de vínculos se ha expandido en Visual Studio para que los botones y vínculos sean más intercambiables en algunas condiciones.

Cuándo utilizar los botones de comando:

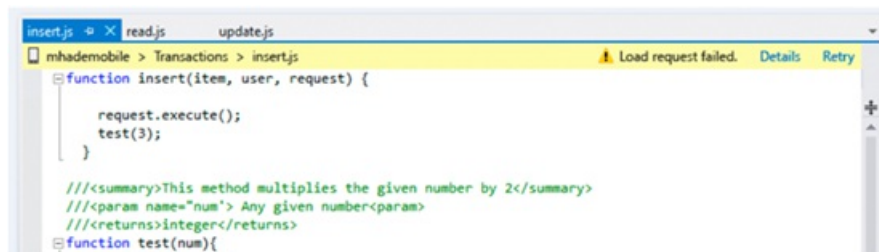
- Comandos primarios
- Visualización de ventanas utilizadas para recopilar entradas o tomar decisiones, incluso si son comandos secundarios
- Acciones destructivas o irreversibles
- Botones de compromiso dentro de los asistentes y flujos de página

Evite los botones de comando en las ventanas de herramientas o si necesita más de dos palabras para la etiqueta. Los vínculos pueden tener etiquetas más largas.

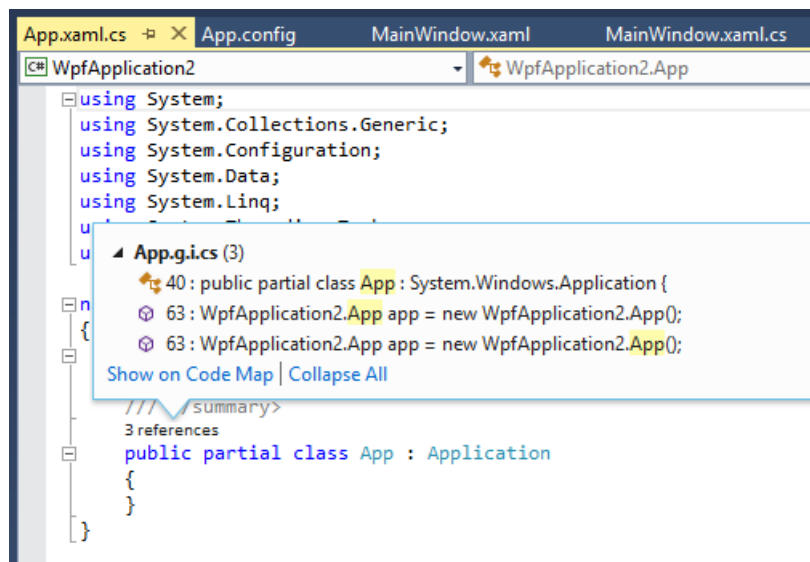
Cuándo utilizar enlaces:

- Navegación a otra ventana, documento o página web
- Situaciones que requieren una etiqueta más larga o una frase corta para describir la intención de la acción
- Espacios estrechos donde un botón abrumaría a la interfaz de usuario, siempre que la acción no sea destructiva o irreversible
- Desenfatar los comandos secundarios en situaciones en las que hay muchos comandos

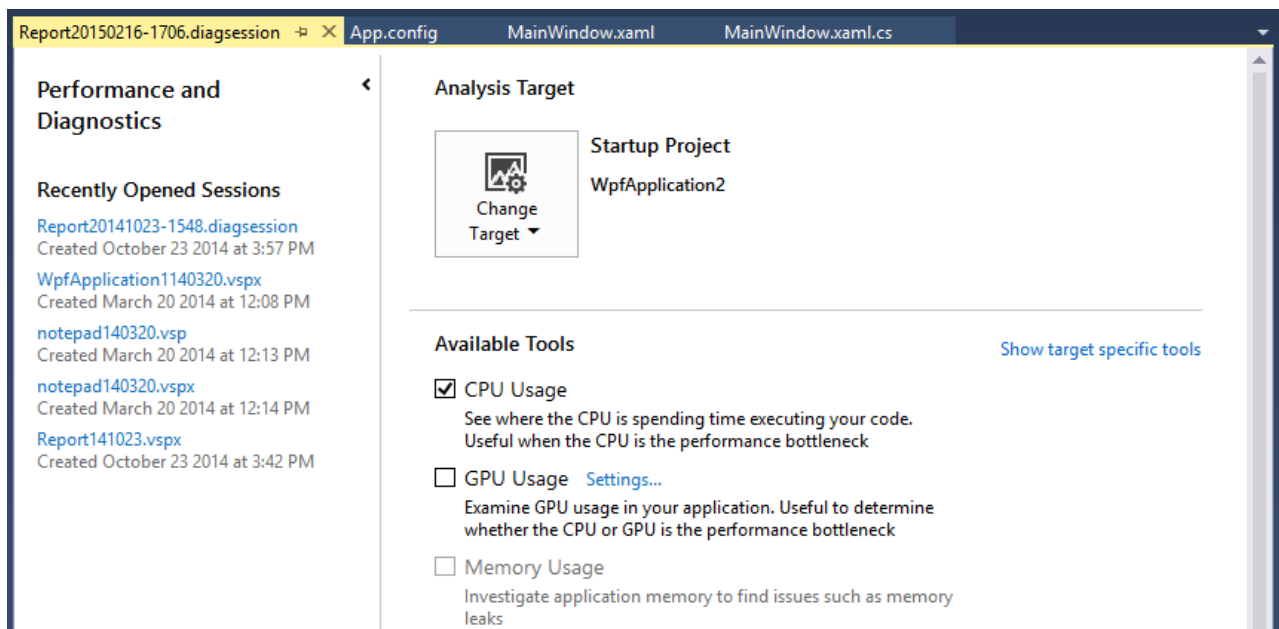
Ejemplos



Enlaces de comandos utilizados en la barra de información después de un mensaje de estado



Vínculos que se usan en el menú emergente de CodeLens



Enlaces utilizados para comandos secundarios donde los botones atraerían demasiada atención

Botones comunes

Texto

Siga las directrices de escritura en el texto y la terminología de la interfaz de [usuario](#).

Estilo visual

Estándar (sin tema)

La mayoría de los botones de Visual Studio aparecerán en los cuadros de diálogo de la utilidad y no se deben aplicar estilos. Deben reflejar la apariencia estándar de los botones según lo dictado por el sistema operativo.

Temática

En algunos casos, los botones se pueden usar dentro de la interfaz de usuario con estilo y estos botones deben tener un estilo adecuado. Consulte [Diálogos](#) para obtener información sobre los controles temáticos.

Botones especiales

Navega... Botones

[Examinar...] botones se utilizan en cuadrículas, cuadros de diálogo y ventanas de herramientas y otros elementos de interfaz de usuario no esmimoso. Muestran un selector que ayuda al usuario a rellenar un valor en un control. Hay dos variaciones de este botón, largo y corto.



El botón largo [Examinar...]



El botón corto de puntos suspensivos [...]

Cuándo utilizar el botón corto de solo puntos suspensivos:

- Si hay más de un botón largo [Examinar...] en un cuadro de diálogo, como cuando varios campos permiten la navegación. Utilice el botón corto [...] para cada uno para evitar las claves de acceso confusas creadas por esta situación** (&Examinar** y B&rowse en el mismo cuadro de diálogo).
- En un diálogo apretado, o cuando no hay un lugar razonable para colocar el botón largo.
- Si el botón aparecerá en un control de cuadrícula.

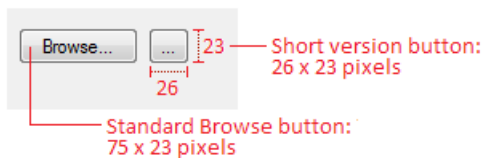
Directrices para el uso del botón:

- No utilice una clave de acceso. Para acceder a él mediante el teclado, el usuario debe tabular desde el control

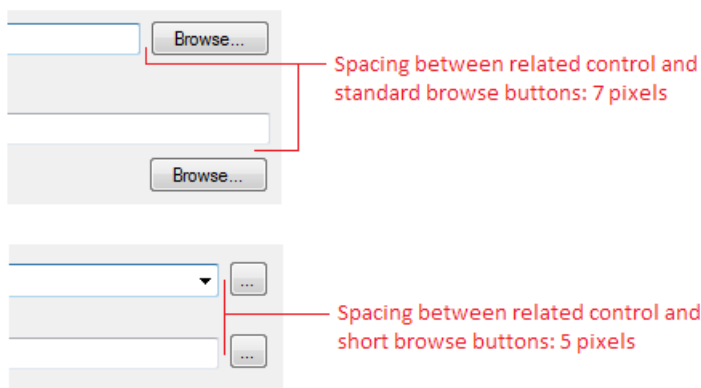
adyacente. Asegúrese de que el orden de tabulación es tal que cualquier botón de exploración cae inmediatamente después del campo que se rellenará. Nunca utilice un carácter de subrayado por debajo del primer período.

- Establezca la propiedad **Nombre** de microsoft Active Accessibility (MSAA) en **Examinar...** (incluidos los puntos suspensivos) para que los lectores de pantalla la lean como "Examinar" y no "punto-punto-punto" o "período-período-período." Para los controles administrados, esto significa establecer el **AccessibleName** propiedad.
- Nunca utilice un botón de puntos suspensivos [...] para nada excepto una acción de exploración. Por ejemplo, si necesita un botón [**Nuevo...**] pero no tiene suficiente espacio para el texto, el cuadro de diálogo debe rediseñarse.

Tamaño y espaciado



Tamaño de los botones [Examinar...]



Espaciado de los botones [Examinar...]

Botones gráficos

Algunos botones siempre deben usar una imagen gráfica y nunca incluir texto para ahorrar espacio y evitar problemas de localización. Estos se utilizan a menudo en selectores de campo y otras listas que se pueden ordenar.

NOTE

Los usuarios tienen que tabular estos botones (no hay claves de acceso), así que colóquelos en un orden razonable. Asigne `name` la propiedad del botón a la acción que realiza para que los lectores de pantalla interpreten correctamente la acción del botón.

FUNCIÓN	BOTÓN
Sumar	>
Remove	<
Agregar todo	>>
Quitar todo	<<

FUNCIÓN	BOTÓN
Subir	
Bajar	
Eliminar	

Tamaño y espaciado

El tamaño de los botones gráficos es el mismo que para la versión corta del botón **[Examinar...]** (26x23 píxeles):



Apariencia de una imagen gráfica en el botón, con y sin color transparente que se muestra

Hipervínculos

Los hipervínculos son adecuados para acciones basadas en navegación, como abrir un tema de Ayuda, un cuadro de diálogo modal o un asistente. Si se utiliza un hipervínculo para un comando, siempre debe mostrar un cambio visible y notable en la interfaz de usuario. En general, las acciones que se confirman en una acción (como Guardar, Cancelar y Eliminar) se comunican mejor mediante un botón.

Estilo de escritura

Siga las instrucciones de escritorio de Windows para el texto de la interfaz de [usuario](#). No uses frases sobre "Más información", "Cuéntame más" o "Obtén ayuda con esto". En su lugar, frase Ayuda vincular texto en términos de la pregunta principal respondida por el contenido de la Ayuda. Por ejemplo, "¿Cómo **agrego un servidor al Explorador** de servidores? "

Estilo visual

- Los hipervínculos siempre deben usar [el servicio VSColor](#). Si un hipervínculo no tiene un estilo correcto, parpadea en rojo cuando está activo o muestra un color diferente después de ser visitado.
- No incluya subrayados en el estado de reposo del control a menos que el vínculo sea un fragmento de oración dentro de una oración completa, como en una marca de agua.
- Los subrayados no deben aparecer al pasar el ratón. En su lugar, los comentarios al usuario de que el vínculo está activo es un ligero cambio de color y el cursor de vínculo adecuado.

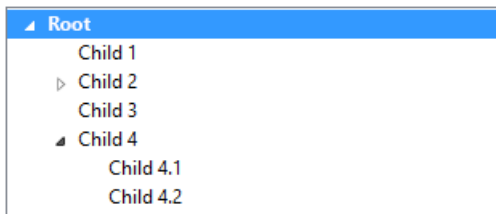
Vistas a los árboles

Las vistas de árbol proporcionan una manera de organizar listas complejas en grupos primarios y secundarios. Un usuario puede expandir o contraer grupos primarios para mostrar u ocultar los elementos secundarios subyacentes. Cada elemento de una vista de árbol se puede seleccionar para proporcionar más acción.

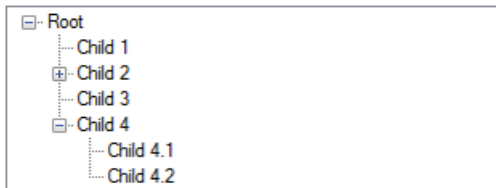
Estilo visual de vista de árbol

Expansores

Los controles de vista de árbol deben ajustarse al diseño del expansor utilizado por Windows y Visual Studio. Cada nodo utiliza un control de expansión para mostrar u ocultar los elementos subyacentes. El uso de un control de expansión proporciona coherencia a los usuarios que pueden encontrar diferentes vistas de árbol dentro de Windows y Visual Studio.



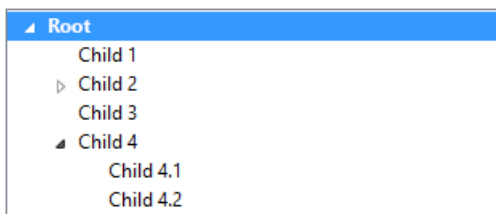
Correcto: estilo adecuado del nodo de vista de árbol mediante un control de expansión



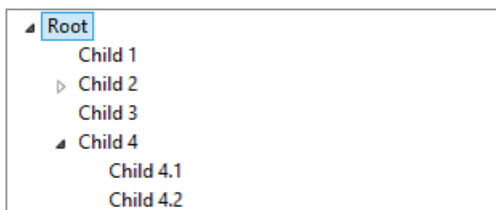
Error: estilo incorrecto del nodo de vista de árbol

Número de selección

Cuando se selecciona un nodo dentro de la vista de árbol, el resaltado debe expandirse al ancho completo del control de vista de árbol. Esto ayuda a los usuarios a identificar claramente qué elemento han seleccionado. Los colores de selección deben reflejar el tema actual de Visual Studio.



Correcto: el resaltado del nodo seleccionado se ajusta a todo el ancho del control de vista de árbol.



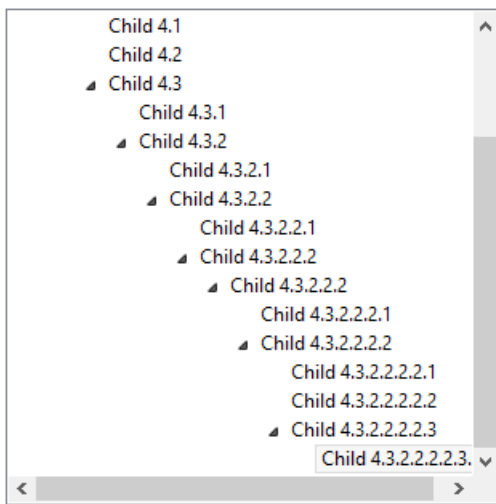
Error: el resaltado del nodo seleccionado no se ajusta a todo el ancho del control de vista de árbol.

Iconos

Los iconos solo se deben usar en los controles de vista de árbol si ayudan a identificar visualmente las diferencias entre los elementos. En general, los iconos deben utilizarse únicamente en listas heterogéneas en las que los iconos llevan información para diferenciar los tipos de elementos. En una lista homogénea el uso de iconos se puede ver con frecuencia como ruido y debe evitarse. En ese caso, el icono de grupo (principal) puede transmitir el tipo de elementos dentro de él. La excepción a esta regla sería si el icono es dinámico y se utiliza para indicar el estado.

Barras de desplazamiento

Las barras de desplazamiento siempre deben estar ocultas si el contenido se ajusta al control de vista de árbol. Es aceptable que las barras de desplazamiento estén ocultas o semitransparentes en una ventana desplazable y aparezcan cuando la ventana que contiene la vista de árbol tenga el foco o al pasar el cursor por la propia vista de árbol.

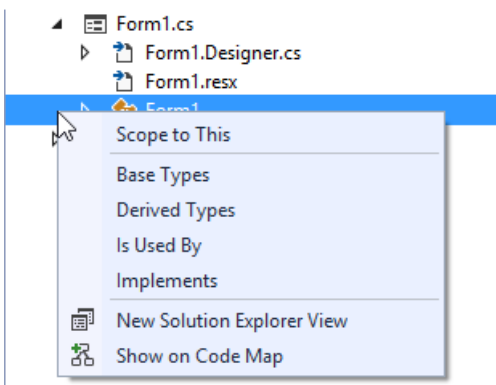


Las barras de desplazamiento verticales y horizontales se muestran porque el contenido ha superado los límites del control de vista de árbol.

Interacciones con la vista de árbol

Menús contextuales

Un nodo de vista de árbol puede mostrar opciones de submenú en un menú contextual. Normalmente, esto ocurre cuando un usuario ha hecho clic con el botón derecho en un elemento o ha presionado la tecla Menú en un teclado de Windows con el elemento seleccionado. Es importante que el nodo gane el foco y esté seleccionado. Esto ayuda al usuario a identificar a qué elemento pertenece el submenú.



El elemento que ha generado el menú contextual gana el foco para notificar al usuario qué elemento se ha seleccionado.

Teclado

La vista de árbol debe proporcionar la capacidad de seleccionar elementos y expandir o contraer nodos con el teclado. Esto garantiza que la navegación cumpla con nuestros requisitos de accesibilidad.

Control de vista de árbol

Los controles de árbol de Visual Studio deben seguir la navegación de teclado común:

- **Flecha arriba:** Seleccione elementos subiendo el árbol
- **Flecha abajo:** Seleccione elementos bajando el árbol
- **Flecha derecha:** Expandir un nodo en el árbol
- **Flecha izquierda:** Contraer un nodo en el árbol
- **Introduzca la clave:** Iniciar, cargar, ejecutar el elemento seleccionado

Trid (vista de árbol y vista de cuadrícula)

Un control trid es un control complejo que contiene una vista de árbol dentro de una cuadrícula. Expandir, contraer y navegar por el árbol debe respetar los mismos comandos de teclado que una vista de árbol, con las siguientes adiciones:

- **Flecha derecha:** Expanda un nodo. Después de expandir el nodo, debe continuar navegando a la columna más cercana a la derecha. La navegación debe detenerse al final de la fila.
- **Pestaña:** Navega a la celda más cercana a la derecha. Al final de la fila, la navegación continúa a la siguiente fila.
- **Mayús + Tabulador:** Navega a la celda más cercana a la izquierda. Al principio de la fila, la navegación continúa a la celda más a la derecha de la fila anterior.

ID	Name	Action
▶ ☒ Microsoft....		Warning
▲ ☒ Microsoft....		Warning
☒ CA1301	Avoid duplicate accelerators	Warning
☒ CA2101	Specify marshaling for P/Invoke string argumen...	Warning
▲ ☒ Microsoft.I...		Warning
☒ CA1400	P/Invoke entry points should exist	Warning
☒		
☒ CA1403	Auto layout types should not be COM visible	Warning
☒ CA1404	Call GetLastError immediately after P/Invoke	Warning
☒ CA1405	COM visible type base types should be COM vi...	Warning
☒ CA1410	COM registration methods should be matched	Warning
☒ CA1415	Declare P/Invokes correctly	Warning
▲ ☒ Microsoft.P...		Warning
☒ CA1821	Remove empty finalizers	Warning
▲ ☐ Microsoft.P...		None
☐ CA1900	Value type fields should be portable	None

Un control trid en Visual Studio

Patrones compuestos para Visual Studio

06/04/2020 • 66 minutes to read • [Edit Online](#)

Los patrones compuestos combinan elementos de interacción y diseño en configuraciones distintas. Algunos de los patrones compuestos más importantes de Visual Studio con respecto a la coherencia incluyen:

- [Visualización de datos](#)
- [Interfaz de usuario en el objeto y miradas](#)
- [Modelos de selección](#)
- [Configuración de persistencia y ahorro](#)
- [Entrada táctil](#)

Visualización de datos

Información general

Los gráficos son una forma visual de agregar y visualizar datos con el fin de mejorar la toma de decisiones. Pueden ayudar a los usuarios que se enfrentan a una gran cantidad de datos, pero poco significado ver lo que merece atención y lo que podría necesitar una acción.

El usuario se beneficiará de un gráfico si se cumple alguna de las siguientes condiciones:

- ¿El gráfico ayudará a los usuarios a identificar las tareas en las que pueden actuar?
- ¿Permitirá el gráfico a los usuarios pronosticar las consecuencias de posibles cambios?
- ¿El gráfico ayudará a los usuarios a descubrir tendencias e identificar patrones?
- ¿Permitirá el gráfico a los usuarios tomar mejores decisiones?
- ¿Ayudará el gráfico a responder a una pregunta específica que los usuarios puedan tener en el contexto dado?

Reglas generales para gráficos

- Etiquete claramente los datos. Las ilustraciones sin explicación son sólo imágenes bonitas.
- Arranque los ejes en cero para evitar proporciones sesgadas. La longitud de línea y el tamaño de la barra son indicaciones visuales importantes para comprender las relaciones entre los puntos de datos.
- Crear gráficos, no infografías. Las infografías son representaciones artísticas de los datos, y su objetivo principal es la narración visual. Los gráficos pueden (y deben) ser visualmente atractivos, pero deje que los datos hablen por sí mismos.
- Evite el eskeumorfismo, los gráficos de barras pictóricas, las marcas hash de contraste y otros toques infográficos.
- No utilice efectos 3D como elemento decorativo. Utilícelos sólo si realmente son parte integral de la capacidad del usuario para comprender la información.
- Evite el uso de varias líneas y rellenos, ya que más de dos colores pueden hacer que este tipo de gráfico sea difícil de leer e interpretar correctamente.
- No utilice un gráfico (o cualquier ilustración) como único medio para entender un concepto o interactuar

con los datos. Esto presenta dificultades para los usuarios con discapacidades visuales.

- No utilice gráficos como elementos gratuitos o decorativos en una página. En otras palabras, si un gráfico no agrega ningún valor o ayuda a los usuarios a resolver un problema, no lo use.

Tipos de gráficos

Los tipos de gráficos utilizados en Visual Studio incluyen gráficos de barras, gráficos de líneas, un gráfico circular modificado conocido como gráfico circular o "gráfico de anillos", líneas de tiempo, gráficos de dispersión (también denominados "gráficos de clúster") y gráficos de Gantt. Cada tipo de gráfico es útil para comunicar un tipo diferente de información.

Otras consideraciones de gráficos

Color

Hay una paleta específica de colores de gráficos definidas para su uso en Visual Studio. La paleta es accesible para los principales tipos de daltonismo, y los colores se pueden diferenciar incluso cuando se utilizan como rebanadas de color muy estrechas. Puede usar estos colores en cualquier combinación para cualquier tipo de gráfico o gráfico en la interfaz de usuario. No es necesario utilizar los siete colores si no necesita tantos colores distintos. Estos colores no se diseñaron para ser utilizados con ningún elemento de primer plano, por lo que no coloque texto o glifos encima de estos colores. Estos tonos deben estar codificados de forma rígida y exponerse a la personalización del usuario en **Herramientas > Opciones** (consulte Exposición de [colores para los usuarios finales](#)).

SWATCH	HEX	RGB
	#71B252	113,178,82
	#BF3F00	191,63,0
	#FCB714	252,183,20
	#903F8B	144,63,139
	#117AD1	17,122,209
	#79D7F2	121,215,242
	#B5B5B5	181,181,181

Interfaz de usuario en el objeto y miradas

En esta sección se proporciona contexto a los peeking, también conocido como vista de vista de vista de código, un tipo de interfaz de usuario en el objeto exclusiva de Visual Studio.

Información general

- La interfaz de usuario en el objeto debe proporcionar al usuario más información o interactividad sin menoscabo de su tarea principal.
- El patrón principal para la interfaz de usuario en el objeto en Visual Studio se conoce como "información en el punto de atención."
- La interfaz de usuario en el objeto en Visual Studio es en línea o flotante y es duradera o transitoria.
 - Vista de vista de código, un tipo de interfaz de usuario en el objeto en Visual Studio, es en línea y duradero.
 - CodeLens, un tipo de interfaz de usuario en el objeto en Visual Studio, es flotante y transitorio

Comprender cómo funciona un fragmento de código o encontrar detalles sobre ese código, a menudo requiere que un desarrollador cambie de contexto e vaya a otro contenido u otra ventana. Estos cambios de contexto pueden ser perjudiciales, ya que los usuarios pueden perder el foco en su tarea original si abandonan su ventana principal. Además, recuperar ese contexto original puede ser difícil, especialmente si cambiar las ventanas hizo que su código original quedara oscurecido por otra interfaz de usuario.

La interfaz de usuario en el objeto sigue un patrón denominado "información en el punto de atención." Estos mensajes, ventanas emergentes y cuadros de diálogo proporcionan a los usuarios información adicional y relevante que agrega aclaración o interactividad sin perder el foco en su tarea principal. Ejemplos de interfaz de usuario en el objeto incluyen ventanas emergentes que aparecen cuando un usuario pasa el puntero sobre un icono en el área de notificación, el ondulador rojo bajo una palabra mal escrita y la vista peek introducida en Visual Studio 2013.

Puntos de decisión

En Visual Studio, hay varias maneras de usar este patrón de información en el punto de atención. Elegir el mecanismo adecuado e implementarlo de una manera coherente y predecible es esencial para la experiencia general. De lo contrario, es posible que se presente a los usuarios una experiencia confusa o incoherente que resta atención al propio contenido.

Relaciones entre el contenido maestro y el contenido detallado

La información en el punto de atención se utiliza para mostrar una relación entre el contenido en el que el usuario se centra (el contenido "maestro") y el contenido relacionado adicional (el contenido "detalle"). En este patrón, el contenido detallado está claramente relacionado con el contenido con el que el usuario está trabajando y se puede mostrar cerca del contenido maestro. La información suplementaria o la información que no se puede resumir sin sobrecargar el contenido maestro debe seguir otro patrón, como una ventana de herramientas.

- **Muestre siempre** el contenido detallado cerca del contenido maestro.
- **Asegúrese siempre** de que el contenido detallado todavía permite a un usuario permanecer centrado en el contenido maestro. A menudo, la mejor manera de lograr esto es representar el contenido de detalle lo más cerca posible del contenido maestro. Esto se puede hacer representando el contenido de detalle en una ventana emergente junto al contenido maestro, o representando el contenido de detalle en línea debajo del contenido maestro.
- **Nunca** utilice la información en el punto de atención que aleja al usuario del contenido maestro. Si los usuarios necesitan ver el contenido detallado por separado, exponga una acción explícita que permita al usuario hacerlo.

Detalles de diseño

Una vez que haya determinado que la interfaz de usuario en el objeto es la opción correcta, hay cuatro consideraciones de diseño principales:

1. **Persistencia:** ¿se espera que el contenido sea duradero o transitorio? ¿Desearán los usuarios mantener la información visible para hacer referencia o interactuar con ella? ¿O los usuarios querrán echar un vistazo rápidamente a la información y luego continuar con su tarea principal?

2. **Tipo de contenido:** ¿el contenido será informativo, procesable o navegable? ¿Necesita el usuario detalles adicionales sobre el contenido maestro? ¿Es necesario que el usuario complete una tarea que afecte al contenido maestro? ¿O es necesario dirigir al usuario a otro recurso?
3. **Tipo de indicador:** ¿tiene sentido un indicador ambiental? ¿Se puede resumir la información de una manera útil y mostrarse sin agobiar el contenido maestro?
4. **Gestos:** ¿qué gestos se usarán para invocar y descartar la interfaz de usuario? ¿Cómo mostrará el usuario el contenido detallado y lo enviará? ¿Hay valor en agregar un gesto como anclar para cambiar entre estados transitorios y duraderos?

Cada uno de estos cuatro puntos de decisión tendrá un impacto en los componentes principales de la interfaz de usuario en el objeto.

Componentes de interfaz de usuario en el objeto

1. Tipo de contenedor (presentador de contenido)
 - Flotante
 - En línea
2. Tipo de contenido
 - Informativo: datos que pueden ser estáticos o dinámicos
 - Accionable: comandos que cambian el contenido maestro
 - Navegación: vínculos que llevan al usuario a otra ventana o aplicación, como MSDN
3. Gestos
 - Invocación
 - Despido
 - Anclaje
 - Otras interacciones
4. Modelo de persistencia y confirmación
 - Transitorio
 - Duradero
 - Automático
 - A petición
5. Indicadores ambientales (opcional)
 - Subrayado squiggle
 - Icono de etiqueta inteligente
 - Otros indicadores ambientales

Tipo de contenedor (presentador de contenido)

Hay dos opciones principales disponibles para presentar contenido en el punto de atención:

1. **En línea:** un presentador en línea, como la vista de vista que se introdujo en el Editor de código de Visual Studio 2013, hace espacio para el nuevo contenido cambiando el contenido existente.
 - **Prefiere** los presentadores en línea si espera que los usuarios quieran pasar una cantidad

significativa de tiempo haciendo referencia o interactuando con el contenido que presenta.

- **Evite los** presentadores en línea si espera que los usuarios quieran echar un vistazo a la información que presenta y, a continuación, continúe con su tarea principal con una interrupción mínima.

2. **Flotante:** un presentador flotante se coloca lo más cerca posible del contenido seleccionado, pero no altera el diseño del contenido existente. Se pueden emplear varias estrategias, como mostrar un panel de contenido flotante sobre el espacio en blanco disponible más cercano al símbolo seleccionado.

- **Prefiere** los presentadores flotantes si espera que los usuarios quieran echar un vistazo a la información que presenta y, a continuación, continuar con su tarea principal con una interrupción mínima.
- **Evite los** presentadores flotantes si espera que los usuarios quieran pasar una cantidad significativa de tiempo haciendo referencia o interactuando con el contenido que presenta.

Tipo de contenido

Hay tres tipos principales de contenido que se pueden mostrar dentro de cualquier contenedor de interfaz de usuario en el objeto. Se puede mostrar cualquier combinación de estos tipos de información. Los tres tipos son:

1. **Informativo:** la mayoría de los contenedores de interfaz de usuario en el objeto mostrarán algún tipo de contenido informativo. El contenido puede representar información sobre el estado actual del entorno o puede representar información sobre un posible estado futuro del entorno. Por ejemplo, podría utilizarse para mostrar el efecto de un comando determinado, como una refactorización, en el código existente.

- **Utilice siempre** la representación canónica de la información que muestra. Por ejemplo, el código debe parecerse al código, con resaltado de sintaxis, y debe respetar cualquier fuente y otra configuración de entorno que el usuario haya establecido.
- **Considere siempre** admitir cualquier acción sobre el contenido informativo que sería posible si esa misma información se presenta como contenido maestro. Por ejemplo, si presenta código existente dentro de un contenedor de interfaz de usuario en el objeto, considere la posibilidad de admitir la capacidad de examinar y modificar ese código.
- **Considere siempre** el uso de un color de fondo diferente si presenta contenido informativo que representa un estado futuro potencial.

2. **Accionable:** algunos contenedores de interfaz de usuario en el objeto proporcionarán la capacidad de realizar alguna acción sobre el contenido maestro, como realizar una operación de refactorización.

- **Coloque siempre** los comandos accionables por separado del contenido informativo.
- **Habilite y deshabilite siempre** las acciones cuando corresponda.
- **Consulte siempre** las directrices estándar para representar comandos dentro de los cuadros de diálogo.
- **Mantenga siempre** el número de acciones que se exponen en un contenedor de interfaz de usuario en el objeto a un mínimo absoluto. Interactuar con la interfaz de usuario en el objeto debe ser una experiencia ligera y rápida. El usuario no debe tener que gastar energía en la administración del propio contenedor de interfaz de usuario en el objeto.
- **Considere siempre** cómo y cuándo se cerrará o descartará un contenedor de interfaz de usuario en el objeto. Como práctica recomendada, cualquier acción que concluya el cuadro de diálogo entre el contenido maestro y el contenido de detalle también debe cerrar el contenedor de interfaz de usuario en el objeto cuando se invoca esa acción.

3. **Navegación:** algunos contenedores de interfaz de usuario en el objeto incluyen vínculos que llevan al usuario a otra ventana o aplicación, como abrir un artículo de MSDN en el explorador web del usuario.

- **Siempre** anteponga cualquier enlace de navegación con "Abrir" para que los usuarios no se sorprendan al ser navegados a algún otro contenido.
- **Separe siempre** los vínculos de navegación de los vínculos procesables.

Indicadores ambientales (opcional)

Los indicadores ambientales pueden ser sutiles, incluido el texto presentado en un color de contraste del resto del código, o obvio, incluidos símbolos de tickler como subrayados ondulados e iconos de etiquetas inteligentes. Los indicadores ambientales comunican la disponibilidad de información adicional y relevante. Idealmente, proporcionan información útil incluso sin necesidad de que el usuario interactúe con ellos.

- **Coloque siempre** un indicador ambiental para que no distraiga ni abrume al usuario. Si es imposible colocar un indicador ambiental de tal manera, considere otra solución.
- **Coloque siempre** el indicador ambiental lo más cerca posible del contenido con el que está relacionado.
- **Trate siempre** de crear un indicador que resuma la información que pone a disposición. Considere la posibilidad de proporcionar un recuento del número de elementos de datos disponibles (por ejemplo, "3 referencias" en lugar de simplemente "Referencias") o pensar en alguna otra manera de resumir los datos.
 - En los casos en que los datos de un indicador no siempre se pueden calcular y mostrar, considere inmediatamente proporcionar comentarios progresivos a medida que se calculan los valores. Por ejemplo, considere la posibilidad de animar los cambios que reflejan las actualizaciones de los datos disponibles, de forma similar a la forma en que se actualiza el icono de correo electrónico en Windows Phone a medida que aumenta el número de correos electrónicos no leídos.
- **Nunca** agregue más indicadores de los que un usuario puede aceptar razonablemente para un determinado contenido. Los indicadores ambientales deben ser útiles sin necesidad de ninguna interacción del usuario. Los indicadores pierden su ambiente si requieren desbordamiento y otros controles de gestión para llevarlos a la vista.

Gestos

Un aspecto clave de permitir al usuario mantener el enfoque en el contenido maestro es apoyar los gestos correctos para abrir y descartar el contenido de detalle adicional.

- **Siempre** requiere que el usuario realice algún gesto explícito para abrir el contenido adicional. Los gestos abiertos comunes incluyen:
 - **Hover**: información sobre herramientas o contenido informativo no interactivo
 - **Comando explícito**: presentador en línea
 - **Haga doble clic en el indicador ambiental**: Ventana emergente CodeLens
- **Descarte siempre** el contenido detallado cada vez que el usuario presione la tecla Esc.
- **Considere siempre** el contexto de la interfaz de usuario en el objeto. Para los presentadores de contenido que permiten la interacción dentro del contenedor, considere cuidadosamente si desea mostrar información adicional al mantener el mouse, lo que probablemente sea perjudicial para el flujo de trabajo del usuario.
- **Nunca** muestre contenido al mantener el mouse que parezca editable o invite a la interacción del usuario. Este comportamiento puede frustrar a los usuarios si intentan mover el cursor sobre el contenido de detalle, ya que el comportamiento estándar de una información sobre herramientas es descartar inmediatamente cuando el cursor ya no está sobre el contenido maestro que lo produjo.

Modelos de selección

Información general

Un modelo de selección es el mecanismo utilizado para indicar y confirmar operaciones en uno o varios objetos de

interés dentro de la interfaz de usuario. En este tema se describen los patrones de interacción de selección en los editores de documentos de Visual Studio: editores de texto, superficies de diseño y superficies de modelado.

Los usuarios deben tener una forma de indicar a Visual Studio en qué están trabajando y Visual Studio debe responder predeciblemente con comentarios a los usuarios sobre lo que está operando. Las diferencias o una comunicación incorrecta entre el usuario y la interfaz de usuario pueden dar lugar a que el usuario no se dé cuenta de una acción, que puede tener consecuencias no deseadas. A menudo, el error pasa desapercibido hasta que el usuario ve que falta algo o ha cambiado. Por lo tanto, los modelos de selección son una de las piezas más críticas del diseño de la interfaz de usuario. Aunque los modelos de selección en Visual Studio son coherentes con Windows, hay ligeras variaciones.

En Visual Studio, como en Windows, los modelos de selección difieren en función del contexto en el que se produce la interacción. Las selecciones pueden producirse en cuatro tipos de objetos:

- Texto
- Objetos gráficos
- Listas y árboles
- Cuadrículas

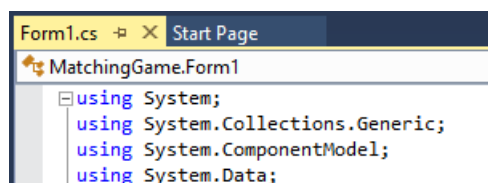
Dentro de estos objetos, hay tres tipos de selecciones:

- Contiguos
- Desunido
- Region

Ámbito

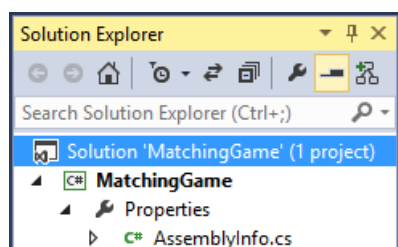
El componente más importante de la selección es asegurarse de que el usuario sabe en qué ventana está trabajando (activación) y dónde se encuentra el foco (selección). Visual Studio amplía la funcionalidad de administración de ventanas en Windows, pero el esquema de activación es el mismo: interactuar con una ventana pone el foco en la ventana. Visual Studio tiene dos indicadores para la activación: uno para las ventanas de documento y otro para las ventanas de herramientas.

Para las ventanas de documento, la ventana activa se indica mediante una pestaña de ventana de documento que viene al frente y cambia su color de fondo:

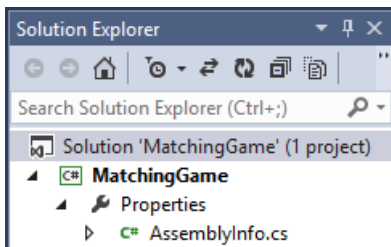


Selección activa de pestañas

Para las ventanas de herramientas, la ventana activa se indica mediante un cambio en el color del área de la barra de título de la ventana de herramientas:



Ventana de herramientas activa que muestra la selección principal de un nodo



Ventana de herramientas inactiva, que muestra la selección latente del nodo

Una vez que una ventana está activa, su enfoque se indica de acuerdo con los modelos de selección descritos en esta sección de las directrices.

Context

Visual Studio se diseñó para conservar un concepto sólido de contexto, realizar un seguimiento de dónde está trabajando el usuario. Solo hay una ventana activa, ya sea una ventana de herramienta o de documento. Sin embargo, la ventana de documento superior siempre conserva una selección latente. Aunque el foco puede estar en una ventana de herramientas, la ventana de documento que estuvo activa por última vez muestra una selección, incluso en un estado inactivo. Esto se hace para conservar el contexto del usuario en el documento que estaban editando, mostrándoles que Visual Studio ha conservado su estado para que puedan volver y cambiar sin problemas entre las ventanas de herramientas y las ventanas de documento.

Selección de texto

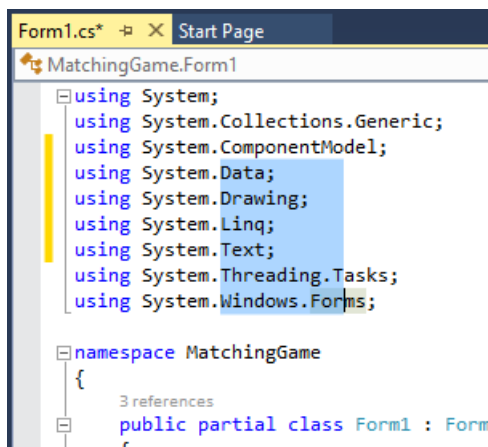
Los editores de Visual Studio que son estrictamente textuales, como el editor de texto integrado, usan el mismo modelo de selección de texto y apariencia que se describe en la página [Mouse y punteros](#) de las directrices de interacción de la experiencia del usuario de Windows en MSDN. El foco de entrada en el editor de texto se indica mediante una barra vertical denominada punto de inserción. El punto de inserción es un solo píxel grueso y coloreado como el inverso de lo que aparece detrás de él. Parpadea según la velocidad establecida por el ajuste velocidad de **parpadeo del cursor** en la pestaña **Velocidad** del applet **Teclado** del Panel de control.

Selección contigua y desarticulada

La selección dentro del editor de texto es solo contigua. Las selecciones de texto desarticuladas no están permitidas, pero deben abordarse en editores de objetos gráficos. Cuando el puntero del ratón del usuario está sobre un área de texto, el cursor cambia a una viga I. Un solo clic coloca el punto de inserción en el editor de texto en la ubicación de clic. Al mantener pulsado el botón del ratón se inicia un resaltado de selección y al soltar el botón del ratón se finaliza el resaltado de selección.

Selección de región (selección de cuadro)

Visual Studio admite selecciones de región en el editor de texto y esto se denomina selección de cuadro. La selección de cuadro permite al usuario seleccionar una región de texto que no sigue la secuencia de texto normal. Al igual que con la selección de texto estándar, la selección debe ser contigua. La selección de cuadros se inicia manteniendo pulsada la tecla Alt mientras arrastra con el ratón. La selección de cuadros también se puede iniciar manteniendo pulsadas las teclas Alt y Mayús mientras se utilizan las teclas de flecha para indicar la región de selección. La selección de cuadro utiliza el resaltado de selección normal y muestra el cursor del punto de inserción parpadeando al final del área de selección.



Selección de región (cuadro) en Visual Studio

Apariencia de selección de texto

Los colores utilizados para la selección activa e inactiva en el editor se pueden personalizar. Para personalizar la apariencia visual del editor, un usuario puede ir a **Herramientas > Opciones**, a continuación, buscar en Entorno > Fuentes y colores > Editor de texto .

Selección gráfica

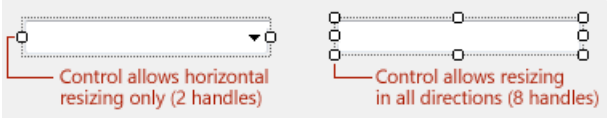
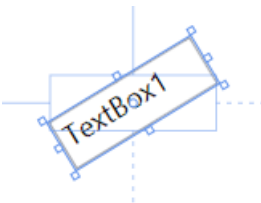
Interacción

La selección gráfica de objetos puede ser compleja y depende de una serie de factores:

- **Modelo de selección principal del editor.** Los editores que contienen objetos gráficos también se pueden utilizar para editar texto o cuadrículas. Por ejemplo, el editor podría ser un editor basado en texto que también admite la colocación de objetos gráficos, como el diseñador XAML de Visual Studio. La compatibilidad con varios tipos de objetos puede afectar a la forma en que el usuario selecciona grupos formados por diferentes tipos de objetos.
- **Compatibilidad con los estados de selección primarios y secundarios.** Un editor puede proporcionar estados de selección primarios y secundarios para que los objetos se puedan editar al unísono, alinearse entre sí, cambiar el tamaño entre sí, etc.
- **Soporte de edición in situ.** Los editores también pueden permitir que se edite el contenido de sus objetos gráficos. Por ejemplo, una forma de rectángulo también puede contener texto en el interior que el usuario puede cambiar. Además, ese texto podría centrarse o justificarse. La edición in situ implica un nivel más detallado de interacción del usuario y, por lo tanto, requiere un conjunto adecuado de indicaciones visuales para presentar información de estado al usuario.

Interacción con el ratón

ENTRADA	RESULTADO
Haga clic en un objeto no seleccionado	Selecciona el objeto y muestra una línea discontinua y los identificadores de selección, si el objeto es redimensionable.
Haga clic en un objeto seleccionado	Activa la edición in situ si el objeto lo admite. Al hacer clic fuera del objeto se desactiva el modo de edición in situ.
Haga doble clic en un objeto	Abre el código detrás del objeto para su edición y podría insertar un controlador de eventos predeterminado, si procede.
Apuntar a un objeto	Cambia el puntero al cursor de movimiento. La apariencia del objeto, como su luminosidad o color, puede cambiar.

ENTRADA	RESULTADO
Señale un controlador de selección	Cambia el puntero al cursor de cambio de tamaño. Para los objetos que admiten la rotación, algunos identificadores de selección pueden cambiar el puntero a un cursor de rotación a medida que el puntero se coloca de forma diferente (por ejemplo, se mueve más lejos) con respecto al identificador de selección.
Arrastrar	Incluso si el objeto no está seleccionado previamente, cambia el puntero al cursor de movimiento y mueve el objeto.
Editor pierde el foco	Desactiva el modo de edición in situ, aunque el objeto conserva el contenido y la apariencia que tuvo durante su último estado de operación/selección.
Selección de objetos	Indicado por un borde, una línea punteada u otro tratamiento visualmente distinto para resaltar el límite del objeto.
Cambiar el tamaño de un objeto seleccionado	Indicado por los controladores de selección. Un objeto redimensionable tiene ocho identificadores, que representan cada dirección en la que se puede cambiar el tamaño. Se pueden utilizar menos identificadores si el objeto solo se puede cambiar de tamaño en determinadas direcciones. Cuando el usuario ajusta el tamaño de un objeto hasta donde ocho identificadores no serían interactivos, se pueden usar cuatro identificadores. Los tamaños de los identificadores deben estar vinculados a las métricas de borde y borde de la ventana con la función de API GetSystemMetrics para que el tamaño sea proporcional a la resolución de pantalla. 
Girar un objeto seleccionado	

Interacción con el teclado

ENTRADA	RESULTADO
Pestaña	Mueve el indicador de enfoque entre el orden lógico de los objetos en el editor. Esto puede ser de izquierda a derecha o de arriba a abajo dependiendo del valor de la propiedad TabIndex (o equivalente), el orden de creación del objeto y el propósito general del editor. Mayús+Tab invierte la dirección del indicador de enfoque.
Barra espaciadora	Activa el modo de panorámica mientras se mantiene la pulsación de tecla. Se requiere una entrada adicional del ratón para desplazar la posición de la ventana gráfica.

ENTRADA	RESULTADO
Ctrl+Barra espaciadora	Activa el modo de zoom mientras se mantiene la pulsación de tecla. Se requiere una entrada adicional del ratón para aumentar y disminuir el factor de zoom.
Ctrl+Alt+Signo menos	Disminuye el factor de zoom en un nivel.
Ctrl+Alt+Signo Más	Aumenta el factor de zoom en un nivel.
Mayús O Ctrl	Agrega el objeto al grupo de selección. Ctrl también le permite eliminar objetos individualmente del grupo de selección.
Escriba	Realiza el comando predeterminado para el objeto (normalmente Abrir o Editar).
F2	Activa la edición in situ para el objeto.
Teclas de dirección	Mueve los objetos seleccionados en la dirección de la tecla de flecha pulsada, en pequeños incrementos (por ejemplo, 1 píxel a la vez)
Ctrl+teclas de flecha	Mueve los objetos seleccionados en la dirección de la tecla de flecha pulsada, en incrementos más grandes (por ejemplo, 10 píxeles a la vez)
Teclas Mayús+flecha	Cambia el tamaño de los objetos seleccionados en la dirección respectiva, en pequeños incrementos (por ejemplo, 1 píxel a la vez)
Ctrl+Mayús+teclas de flecha	Cambia el tamaño de los objetos seleccionados en la dirección respectiva, en incrementos más grandes (por ejemplo, 10 píxeles a la vez)

Cuando los usuarios editan los controles en su lugar, puede tener sentido que los objetos cambie de tamaño automáticamente con la entrada del usuario. Por ejemplo, si el usuario edita un control de etiqueta, la etiqueta debe crecer para mostrar el texto que el usuario acaba de escribir. Si esto no se hace, el usuario debe cambiar el tamaño del control manualmente después de editar el texto. Si el usuario tiene muchos controles, esto se convierte en una tarea de sorción e improductiva.

Contenedores gráficos

En algunos casos, los editores gráficos proporcionan contenedores para otros objetos gráficos, como el control Panel de formularios Windows Forms o el control Diseño de cuadrícula en el diseñador HTML. Si el editor proporciona contenedores para otros objetos gráficos, se debe utilizar el siguiente modelo de selección solo para el contenedor (los objetos dentro del contenedor siguen el modelo estándar como se ha descrito anteriormente):

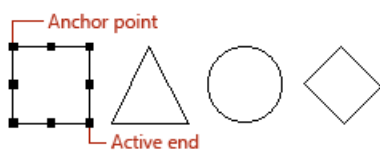
ENTRADA	RESULTADO
Un solo clic en el contenedor	Selecciona el objeto contenedor sin seleccionar directamente ninguno de los objetos contenidos. El contenedor se puede mover y/o cambiar de tamaño con la entrada estándar del ratón y el teclado (como se describió anteriormente). Los objetos contenidos se mueven en relación con el contenedor, pero los objetos contenidos no se redimensionan a menos que también se seleccionen directamente.

ENTRADA	RESULTADO
Pase el cursor sobre la región límite del contenedor	Convierte el ratón en el cursor de movimiento, lo que indica que se puede mover el contenedor.
Arrastre la región de límite del contenedor	Cambia el ratón al cursor de movimiento y mueve el contenedor (y los objetos contenidos dentro). El contenedor no se puede mover sin seleccionarse primero con un solo clic.
Haga un solo clic en un objeto dentro del contenedor	Anula la selección del contenedor (si está seleccionado) y selecciona solo el objeto en el que se ha clic.
Mayús+clic o pulse O Ctrl+clic en un objeto y/o contenedor contenidos	Agrega el objeto en el que se ha clic a una selección o grupo de selección existente. Si el objeto en el que se ha hecho clic ya es miembro del grupo de selección, se elimina del grupo de selección.

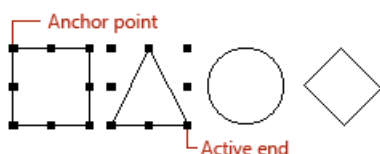
Los objetos contenidos deben adherirse al modelo de selección básico como se describe en la sección anterior. A partir de las pruebas de usabilidad del diseñador de Windows Forms, los usuarios esperaban un acceso sin problemas a los objetos contenidos sin pasos intermedios (impuestos por el objeto de contención).

Selección desarticulada y de región

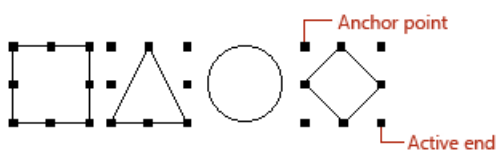
Los editores de objetos gráficos deben admitir selecciones desarticuladas. Tenga en cuenta que este gráfico no muestra la apariencia del control para Visual Studio. Consulte Apariencia de selección de [objetos gráficos](#) para obtener especificaciones visuales detalladas.



1. The user makes an initial selection by clicking the first item.



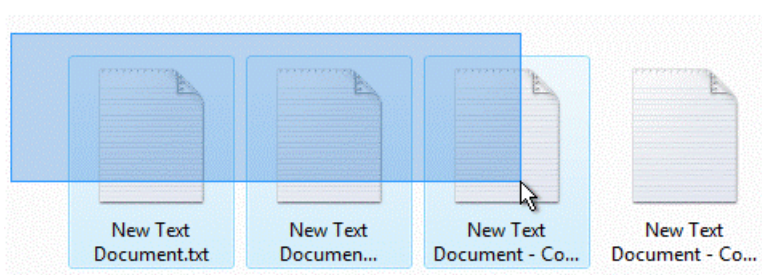
2. The user extends the selection by holding down the SHIFT key and clicking the next item.



3. The user makes a disjoint selection by holding down the CTRL key and clicking another item.

Selección desarticulada

Los editores gráficos también deben proporcionar selecciones de región con un indicador de selección de tipo marco. Si el editor gráfico admite otros tipos de objetos (como texto), es posible que las selecciones de región no sean posibles en función de las restricciones de esos otros tipos de objeto.



Selección de recuadro

Selecciones primarias y secundarias

Algunos editores gráficos de objetos permiten al usuario editar o alinear objetos en grupos. En este caso, es necesario introducir el concepto de selecciones primarias y secundarias. La selección principal es el objeto al que responden todos los demás objetos para las operaciones de grupo. El objeto que el usuario selecciona primero se convierte en el control principal y las selecciones posteriores se convierten en las selecciones secundarias. La selección primaria tiene un tratamiento visual distinto de la(s) selección(es) secundaria(s) para indicar qué objeto es primario:



Selección primaria con dos selecciones secundarias

Apariencia de selección de objetos gráficos

Los controladores de selección son cuadrados dibujados en un patrón rectangular alrededor del cuadro delimitador del objeto. El gráfico siguiente muestra ejemplos de los distintos estados que un objeto gráfico puede tener con el identificador, el tamaño y la apariencia de edición in situ. El tamaño de los identificadores debe estar vinculado a las métricas de borde y borde de la ventana mediante la API **GetSystemMetrics**.

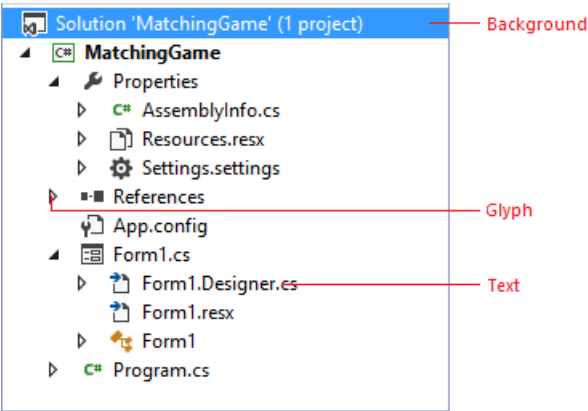
STATE	APARIENCIA	DETALLES VISUALES
No seleccionado	Valor predeterminado	
Selección primaria	Redimensionable	
Selección primaria	No redimensionable	
Selección primaria	Bloqueado	
Selección secundaria	Redimensionable	
Selección secundaria	No redimensionable	
Selección secundaria	Bloqueado	
Interfaz de usuario activa	Valor predeterminado	

Ver modelos de selección

Vista de árbol

La selección en una vista de árbol se muestra con un resaltado simple. Si el usuario hace clic en un nombre de nodo o un icono de nodo, el nodo se selecciona. Los glifos triangulares a la izquierda del nodo expanden o

contraen el control de árbol, pero no afectan a la selección del usuario, con una excepción: al contraer un nodo primario cuando la selección está en un elemento secundario de ese nodo, la selección se mueve al elemento primario.



Vista de árbol típica de Visual Studio

Las vistas de árbol pueden admitir selecciones contiguas y desarticuladas, incluso en varios niveles del árbol. Se deben realizar selecciones múltiples contiguas o desarticuladas en nodos de árbol visibles. Si se contrae un nodo, se pierde la selección desarticulada y el nodo contraído obtiene la selección. De esta manera, el usuario puede ver los nodos que se verán afectados por una operación. Cuando los nodos se contraen, no está claro qué nodos podrían verse afectados.

Cuando se selecciona un nodo primario, la operación debe aplicarse al elemento primario, aunque puede haber casos en los que tenga sentido que una operación se aplique al elemento primario y a todos sus elementos secundarios. En este caso, proporcione una interfaz de usuario adicional durante la operación, como una casilla de verificación o un cuadro de diálogo de confirmación para que la opción "aplicar a todos los elementos secundarios" sea explícita para el usuario.

Cambiar nombre

Si los nodos del árbol admiten el cambio de nombre, el cambio de nombre debe realizarse en su lugar. La operación en el lugar debe ser el estándar en todos los controles de árbol en Visual Studio. Proporcione un comando de cambio de nombre que active inmediatamente el modo de edición in situ, con la selección de texto que cubre todo el nombre del nodo, listo para aceptar la entrada del usuario. Si el nodo representa un archivo, el nombre de archivo debe contener la extensión. El resaltado de selección debe incluir solo el cuerpo del nombre de archivo y no la extensión.

ENTRADA	RESULTADO
Entrar (tecla)	Confirma la operación de cambio de nombre
Tecla Esc	Cancela la operación de cambio de nombre
Hacer clic fuera de la región de edición in situ	Confirma la operación de cambio de nombre
Deshacer	Proporcionar fácil deshacer para cancelar la operación de cambio de nombre

Selección dentro de listas y controles de cuadrícula

El concepto clave en la selección de lista es que se basa en filas, lo que significa que cuando se realiza una selección, toda la fila se selecciona como una unidad. Por el contrario, las cuadrículas pueden permitir que se seleccionen celdas específicas sin afectar a ningún otro aspecto de la fila. Las cuadrículas también pueden contener una jerarquía de filas anidadas (como en un TreeGrid) que permiten seleccionar y anular la selección de ramas enteras de la jerarquía mediante la interacción con las filas primarias. La selección en las listas se muestra con un color de resaltado simple en toda la fila de datos. El foco se muestra mediante un borde punteado de un solo píxel

alrededor de la fila o celda editable actual (fila si todas las celdas son de solo lectura).

NOTE

El **enfoque** y la **selección** son conceptos diferentes. *Focus* es una indicación de qué elemento de interfaz de usuario está destinado a recibir entradas no dirigidas explícitamente a otro objeto, mientras que la *selección* hace referencia al estado de la inclusión de un objeto en un conjunto de objetos en el que pueden tener lugar operaciones posteriores.

Las selecciones en las listas pueden ser contiguas, desarticuladas o regiones. Cuando se permiten varias selecciones, siempre se debe admitir la selección contigua y desarticulada, mientras que la compatibilidad con selecciones de región (cuadro) es opcional. Las selecciones de región se inician arrastrando en el espacio en blanco del cuerpo de la lista.

OBJECT	NÚMERO DE SELECCIÓN
List	Contiguos
List	Desunido
List	Region

Al hacer clic una vez en una lista, se selecciona la fila en la que se produjo el clic. Si el usuario hace clic en una celda de lista que admite la edición in situ, la celda también se activa inmediatamente para la edición in situ. De lo contrario, toda la fila se selecciona inmediatamente y muestra un resaltado.

Arrastrar en el cuerpo de la lista hace una de tres cosas:

- Inicia una selección de región si la lista lo admite y el ratón hacia abajo está en el espacio en blanco
- Inicia una operación de arrastrar/soltar si la celda o fila de lista admite ser un origen de arrastre
- Selecciona la fila actual

Edición in situ

Cuando se permite la edición in situ, hay dos modelos básicos: control de edición simple y selector de propiedades. Con un control de edición simple, el contenido se resalta y está listo para la entrada del usuario tan pronto como se activa la edición in situ. Cuando se implementa un selector de propiedades, el botón que invoca el selector de propiedades se muestra una vez que se activa el modo de edición en contexto y la selección actual no se resalta. El botón del selector debe estar justificado a la derecha en la celda. Para obtener ejemplos de edición in situ, vea la **ventana Propiedades y la lista** de tareas en Visual Studio.

Compatibilidad con el teclado

La compatibilidad con teclado para la selección en listas y cuadrículas sigue las convenciones estándar de Windows:

- Las teclas de flecha navegan por la lista, seleccionando cada fila/celda a medida que se mueve el foco.
- Mayús + flecha realiza una selección contigua en la dirección de las teclas de flecha.
- Ctrl + flecha seguida de barra espaciadora alterna entre agregar y eliminar elementos de lista de la selección, creando una selección desarticulada.
- Para las cuadrículas que contienen jerarquías anidadas, la tecla Flecha derecha expande una fila primaria y la tecla Flecha izquierda contrae una.
- La tecla Tabulador mueve el foco entre las celdas de la fila actual si las celdas son editables.
- La tecla Intro realiza el comando predeterminado en el elemento de la lista (a menudo **Abrir**).

- La tecla F2 activa la edición in situ para la celda seleccionada actualmente.

Configuración de persistencia y ahorro

Información general

Aunque cada componente de software de Visual Studio suele ser responsable de su propio estado y persistencia, Visual Studio guarda automáticamente la configuración en algunos casos, como con los tamaños y las posiciones de las ventanas. La siguiente tabla es una combinación de ajustes guardados automáticamente y ajustes que requieren que se realice una acción explícita del usuario o programada.

OBJECT	QUÉ AHORRAR	CUÁNDO GUARDAR	DÓNDE AHORRAR
Objeto seleccionable (por ejemplo, una línea de código)	Un punto de interrupción en una línea de código Un acceso directo de usuario asociado a la línea de código	Cuando se guarda el proyecto	El archivo de opciones de usuario (.suo) para el proyecto
Diálogo	La ubicación del diálogo, si se hubiera movido La vista que el usuario utilizó por última vez en el cuadro de diálogo	Cuando se cierra el cuadro de diálogo Cuando finaliza la sesión de Visual Studio	En la memoria Registro en HKEY_Current_User
Periodo	El tamaño y la ubicación de la ventana	Cuando se cierra la ventana Cuando cambia el modo de Visual Studio Cuando finaliza la sesión de Visual Studio	El archivo de opciones de usuario (.suo) para el proyecto Archivo de opciones personalizadas para la configuración de la ventana
Documento	La selección actual en el documento La vista del documento Los últimos lugares que el usuario visitó	Cuando se guarda el documento	El archivo de opciones de usuario (.suo) para el proyecto
proyecto	Referencias a archivos Referencias a directorios en disco Referencias a otro software Componentes Información del estado sobre el propio proyecto	Cuando se guarda el proyecto	El archivo del proyecto
Solución	Referencias a proyectos Referencias a archivos	Cuando se guarda el proyecto o la solución	El archivo de solución (.sln)

OBJECT	QUÉ AHORRAR	CUÁNDO GUARDAR	DÓNDE AHORRAR
Configuración en Herramientas > Opciones	Personalizaciones del teclado Personalizaciones de la barra de herramientas Esquemas de color	Cuando se cierra el cuadro de diálogo Herramientas > Opciones Cuando finaliza la sesión de Visual Studio	Registro en HKEY_Current_User

Lo que el usuario está haciendo, y cuando lo está haciendo, dicta si una configuración se está guardando en la memoria (durante la sesión), se guarda en el disco (entre sesiones como una configuración del Registro), como parte del propio archivo de proyecto o solución, como parte del archivo de opciones de **solución (.suo)** o como un archivo de configuración personalizado que solo ese componente de software conoce. La tabla anterior muestra varios eventos en los que se puede guardar la configuración. Sin embargo, hay otras veces en las que es posible que desee guardar el estado:

- Cuando el usuario cambia de ubicación dentro de un cuadro de diálogo o ventana
- Cuando el usuario transfiere el foco a otra ventana
- Cuando el usuario cambia de diseño a modo de depuración
- Cuando el usuario cierra la sesión de su cuenta
- Cuando el equipo entra en hibernación o se apaga
- Cuando el ordenador/disco duro está a punto de ser reformateado y configurado de nuevo

Configuraciones de ventanas

Una configuración de ventana es la presentación básica del entorno de desarrollo - es un esquema que consiste en la lista de ventanas de herramientas presentes y la forma en que se organizan. Para las ventanas administradas por el IDE (ventanas IDE), la información de diseño se conserva por usuario, por lo que cuando un usuario inicia el IDE, el diseño de ventana aparece igual que cuando salió por última vez de Visual Studio. El estado y la posición de las ventanas IDE se conservan en un archivo de opciones personalizado en formato XML. Las ventanas de herramientas creadas por paquetes cargados en el IDE conservan su información de estado en el registro y pueden o no ser por usuario.

Diseños específicos del perfil

Cada perfil incluye diseños de ventana de herramientas, organizados de una manera familiar para personas de desarrolladores específicos (los desarrolladores de Visual C++ esperan ver el **Explorador** de soluciones en el lado izquierdo del IDE, mientras que los desarrolladores de C- esperan ver el **Explorador** de soluciones a la derecha). Los diseños de ventana específicos del perfil se cargan después de que el usuario elija un perfil al iniciarse. Un autor de paquete debe determinar el diseño de ventana más adecuado para la experiencia de su cliente, sabiendo que los cambios que el usuario realice en la configuración de la ventana se conservarán.

Entrada táctil

Los usuarios utilizan cada vez más productos de desarrollo de Microsoft en dispositivos táctiles. Sin embargo, hay barreras que dificultan el uso de herramientas de desarrollo en dispositivos táctiles. Los usuarios esperarán que nuestros productos proporcionen una experiencia táctil fiable y precisa. La intención de estas directrices es informar a las decisiones sobre qué capacidades táctiles incorporar y fomentar una experiencia táctil coherente en Visual Studio y productos relacionados.

Niveles de experiencia

Los siguientes niveles de experiencia están destinados a servir como una guía para ayudar a los equipos a decidir qué capacidades táctiles ofrecer en función del nivel deseado de interés de inversión en contacto.

- La **experiencia básica** es para los equipos que quieren proporcionar capacidades táctiles para que no haya callejones sin salida a lo largo de su trabajo.
- La **experiencia optimizada** es para los equipos que desean proporcionar las capacidades táctiles más comunes (por ejemplo, las que normalmente están disponibles en aplicaciones de explorador de Internet).
- La **experiencia elevada** es para los equipos que desean agregar capacidades como gestos u otras capacidades opcionales que pueden hacer que su aplicación sea amigable con el tacto primero.

	EXPERIENCIA BÁSICA	EXPERIENCIA OPTIMIZADA	EXPERIENCIA ELEVADA
Permite a los usuarios ...	Corregir código y lectura a nivel de solución/proyecto sin callejones sin salida	Realizar tareas de mantenimiento, refactores y navegación	Opere con una experiencia consistente, intuitiva y fluida con confianza
Editor	Toque panorámica y selección Desplazamiento de la barra de desplazamiento para saltar y pulse +arrastrar	Zoom de pellizco Desplazamiento rápido Número de selección Fácil uso del menú contextual	
Ventanas de herramientas superiores	Lista de panoramización Selección de elementos Desplazamiento de la barra de desplazamiento para saltar y pulse +arrastrar	Fácil desplazamiento y selección de elementos	
Basado en ventanas		Ventana de cambio de tamaño Acceso rápido	
Documento bien		Fácil navegación entre archivos abiertos	
Gestos		Asegúrese de que los gestos comunes funcionen en todo el IDE	Acciones basadas en gestos Soporte de arrastrar y soltar y diseñadores
Otras consideraciones			Teclado en pantalla personalizado

Gestos

Los gestos proporcionan a los usuarios un acceso directo a los comandos que, de lo contrario, podrían requerir una interacción más complicada. Consulte las directrices de Windows sobre [gestos táctiles comunes para aplicaciones](#) de escritorio y siga esta guía para la mayoría de los gestos, incluidos los gestos simples, como el desplazamiento panorámico y el zoom.

Texto de la interfaz de usuario y ayuda para Visual Studio

06/04/2020 • 34 minutes to read • [Edit Online](#)

Texto y terminología de la interfaz de usuario

El texto comprensible es crucial para una interfaz de usuario eficaz. Los usuarios de software tienden a leer las etiquetas primero, a saber, las más relevantes para completar la tarea en cuestión. El texto estático se lee con menos frecuencia. Planifique que los usuarios inicien sus sesiones de trabajo con un análisis rápido de toda la ventana, seguido de una lectura de la interfaz de usuario en este orden aproximado:

1. Controles interactivos en el centro
2. Botones de confirmación
3. Controles interactivos encontrados en otros lugares
4. Principales instrucciones
5. Explicaciones suplementarias
6. Título de la ventana
7. Otro texto estático en el cuerpo principal

Patrones de uso para el texto de la interfaz de usuario

Texto de la barra de título

El texto de la barra de título debe coincidir con el comando que generó la interfaz de usuario.

Texto instructivo (texto auxiliar)

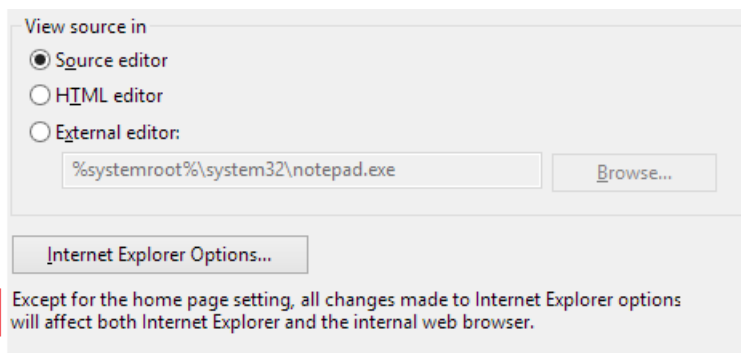
En algunos cuadros de diálogo, es útil proporcionar instrucciones principales prominentes para explicar qué hacer en la ventana o en la página. Esto se conoce a veces como "texto auxiliar".

Reglas de estilo de escritura para texto auxiliar

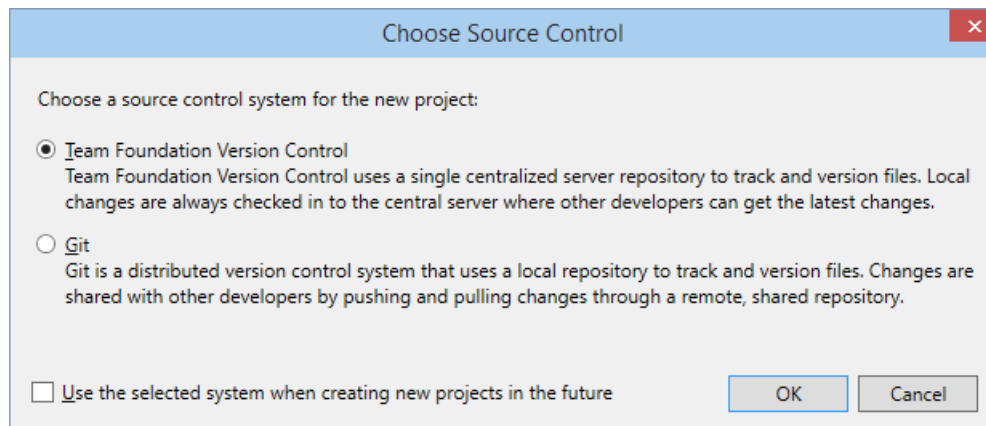
- No expliques lo obvio. A menos que sea absolutamente necesario, no incluya texto instructivo.
- El texto instructivo siempre se coloca en la parte superior del cuadro de diálogo y debe hacer referencia a la tarea que se está realizando.
- Explicar con precisión a los usuarios lo que tienen que hacer. Evite la comunicación excesiva y la redundancia.
- Revise cada ventana y elimine las palabras y las declaraciones duplicadas.
- Mantenga el texto instructivo corto. Si se necesita más información para determinados usuarios o escenarios, proporcione un vínculo a un tema conceptual en línea detallado.
- Escribe tu texto para que cada palabra tenga peso y sea necesaria.
- Siga las instrucciones existentes de Microsoft para el texto y el estilo y el [tono](#) de la interfaz de [usuario](#).

Instrucciones suplementarias

Las instrucciones complementarias proporcionan información adicional que ayuda al usuario a comprender los controles o las agrupaciones de control. Esto también podría incluir el texto de sugerencia necesario para comprender qué formato espera el control de entrada. Utilice instrucciones suplementarias con moderación. Reservarlos para los casos en los que es probable que el usuario no comprenderá completamente las ramificaciones de la elección que están haciendo.



Texto complementario en Visual Studio

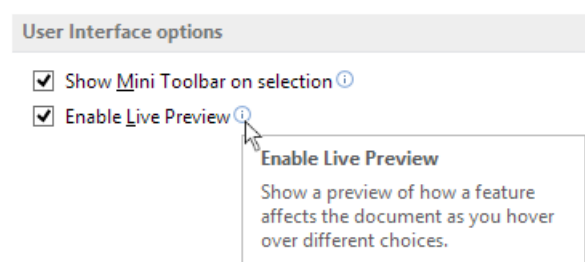


Texto complementario en Visual Studio

Consejos de información

A menudo, el texto instructivo puede ser demasiado largo para colocar en el lugar en la interfaz de usuario o puede ser útil sólo para los nuevos usuarios, sintiéndose como desorden para los usuarios experimentados. En este caso, el texto instructivo/informativo debe colocarse como información sobre herramientas en una información sobre información.

Las sugerencias de información deben colocarse cerca de los controles con los que están relacionados y deben usar el icono de información específica, que es discreto pero perceptible.



Ejemplo de información en Visual Studio

Reglas de estilo de escritura para InfoTips

- Escriba InfoTips como oraciones completas. Requieren verbos específicos, mayúsculas y minúsculas y puntuación final.
- Utilice InfoTips para complementar su instrucción o información principal. Si solo estás usando palabras diferentes para volver a decir la idea principal, no necesitas una información.
- Mantenga InfoTips cortos y dulces. Utilice palabras pequeñas y un lenguaje sencillo y cotidiano que apoye y anime al usuario.
- Siga las instrucciones existentes de Microsoft para el texto y el estilo y el [tono](#) de la interfaz de [usuario](#).

Etiquetas de control

Las etiquetas de control deben ser cortas, concisas y seguir las instrucciones de escritorio de [Windows para controles](#).

Para obtener más información sobre el formato de etiqueta de control y la ubicación dentro de la interfaz de usuario, consulte [Diseño para Visual Studio](#).

Vínculos a la Ayuda

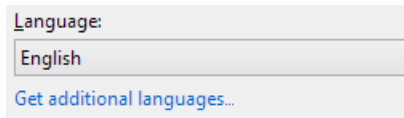
Los vínculos de ayuda se pueden colocar en el texto de instrucción o en el cuerpo de la interfaz de usuario. Pueden ser vínculos a Ayuda o iniciar diálogos internos.

Reglas de estilo visual para vínculos de ayuda

- Utilice los colores de entorno correctos para los hipervínculos. Un hipervínculo con el estilo correcto no parpadeará brevemente en rojo cuando se haga clic en él. Si ve esto, entonces es una indicación de que los colores del entorno no se están utilizando.
- Los subrayados solo se deben usar al mantener el mouse o cuando el vínculo está incrustado en un párrafo.
- Para obtener información más detallada sobre los estilos visuales y de interacción de hipervínculos, consulte Botones e hipervínculos.

Reglas de estilo de escritura para vínculos de ayuda

- Al iniciar cuadros de diálogo, mantenga los estándares para los puntos suspensivos: no hay puntos suspensivos para la navegación, puntos suspensivos si la tarea requiere interfaz de usuario adicional.



Los puntos suspensivos (...) en un vínculo de Ayuda indican que la tarea requerirá una interfaz de usuario adicional.

- Los vínculos no deben comenzar con "Aprender", ya que esa no es la intención del usuario. El usuario quiere responder a una pregunta específica, no recibir una educación general.
- Enlaces de ayuda de frase para que hagan la pregunta que el tema responderá.

Error: "Más información sobre los precios de Windows Azure Mobile Services"

Correcto: "¿Qué opciones de precios están disponibles para Windows Azure Mobile Services?"

- Nunca utilice *Haga clic...* en el texto del enlace.
- Nunca vincules sólo la palabra "aquí". Esto es problemático para algunos lectores de pantalla, que solo expresarán la palabra hiperenlace.

Error: "Buscar información en Windows Azure Mobile Services aquí"

Correcto: "¿Qué opciones de precios están disponibles para Windows Azure Mobile Services?"

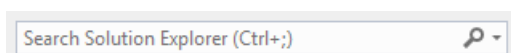
- Para obtener más información sobre el estilo de escritura correcto para los vínculos de Ayuda, consulte las instrucciones de Escritorio de [Windows para Obtener ayuda](#).

Texto de sugerencia

El texto de sugerencia aparece como una marca de agua dentro de un control o debajo del control. El formato correcto se aplicará mediante el `Environment.GrayText` token VSColors adecuado, .

Puede aparecer en una serie de formas.

- En lugar de la etiqueta de control:

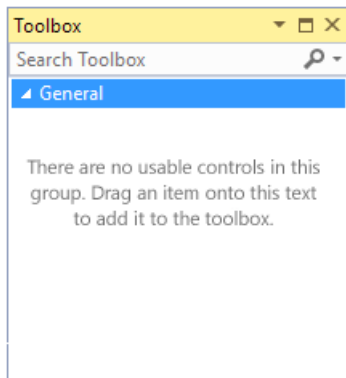


- Con un verbo, dando instrucciones:

- Con texto que indica una entrada obligatoria:

Texto de marca de agua

En una superficie de diseño vacía, el texto debe indicar qué hacer, así como proporcionar vínculos para abrir otras ventanas relacionadas, si procede:



Ejemplo de texto de marca de agua en Visual Studio

Terminología común

TÉRMINO	EXPLICACIÓN	COMENTARIO
Iniciar sesión / Cerrar sesión	Verbos utilizados como sinónimos de la web para representar la autenticación en una propiedad web. Dentro de los clientes, usamos esta única vez como una noción de nivel superior para iniciar y cerrar sesión en la conexión de usuario IDE, que representa una identidad de nivel superior que proporciona capacidades de nivel superior, como itinerancia y licencias que no están disponibles con todas las demás conexiones.	El usuario IDE es la única característica que debe representar un verbo de inicio /cierre de sesión, ya que representa el usuario IDE de nivel superior.
Conectar / Desconectar	Utilícelo en lugares donde una característica mantiene una sola conexión a un servicio en línea.	El Explorador de servidores, donde solo puede tener una conexión de Azure activa a la vez, es un ejemplo de Conectar/Desconectar.
Añadir / Eliminar	No destructivo. Utilícelo al agregar o quitar algo de una lista.	El cuadro de diálogo de lista de servidores de TFS Connection Manager es un ejemplo de Agregar o quitar.
Eliminar	Destructivo. Utilícelo solo cuando el elemento que se va a quitar se descarte o elimine permanentemente del disco.	"Eliminar" generalmente requiere un mensaje si el resultado es eliminar un archivo del disco.

Mensajes de error

Información general

Se producen errores. Establecer limitaciones en lo que el usuario puede hacer es un primer paso sensato para evitar mensajes de error evitables. Sin embargo, cuando se produce un error, un mensaje de error bien escrito puede ir un largo camino hacia la mitigación del problema. Los mensajes de error son posiblemente uno de los tipos más importantes de notificación que el usuario ve, porque son sincrónicos e indican un problema que debe resolverse. Los mensajes de error mal escritos dejan a los usuarios por su cuenta para decidir la causa de los errores y cualquier posible solución.

Los usuarios pueden dejar de prestar atención a los mensajes de error sobreutilizados o confusos, por lo que escribir sólo los mensajes necesarios que agregan valor a la experiencia del usuario. Si el mensaje es simplemente una notificación, utilice una presentación alternativa.

Reglas para crear un mensaje de error

- Al construir mensajes de error, elija el nivel de error adecuado para la audiencia. Apunte a resúmenes sencillos que proporcionen una acción que el usuario pueda realizar, si corresponde. No indique nada que el usuario no necesite saber.
- Proporcionar asistencia constructiva. Es más fácil leer y actuar en un mensaje de error que contiene instrucciones.
- No uses dobles negativos.
- Realice una comprobación automática y manual de gramática y ortografía en cualquier mensaje de error que escriba.
- Para mensajes de error complejos, evite las comunicaciones secuenciales. Nunca utilice una conexión F1 para el mensaje de error. El mensaje en sí debe ser suficiente.
- Utilice el icono correcto.
- Haga que las preguntas sean fáciles de entender y use botones que tengan opciones claras, como "Eliminar" y "Cancelar".
- En el caso de las advertencias, tenga claro cuál será la consecuencia del procedimiento. Los botones deben indicar la consecuencia.
- En caso de errores, describa lo que el usuario puede hacer para solucionar el problema. Los botones deben ser acciones o decir "Cerrar". No utilice un botón "OK" para un mensaje de error.
- Algunas preguntas que debe hacerse al construir un mensaje de error:
 - ¿Puede el usuario averiguar cómo resolver el problema con este error solo?
 - ¿Utiliza el usuario el mismo vocabulario que este error?
 - ¿Este error es ambiguo o compartido en varias situaciones? Si es así, ¿cómo guía a los usuarios a la solución que necesitan?

Errores de compilación

Dado que Visual Studio es una herramienta de desarrollo de software, muchos de sus componentes tienen un paso de compilación, conversión o codificación para convertir el trabajo del desarrollador a formato binario. Estas conversiones pueden provocar errores cuando el compilador no puede procesar archivos creados incorrectamente o cuando las opciones del compilador no se establecieron correctamente.

Los usuarios de Visual Studio pueden pasar un gran número de horas de desarrollo resolviendo errores de compilación. Este tiempo de resolución aumenta cuando los errores tienen dependencias o cuando los mensajes de error están mal escritos, lo que puede dificultar la detección del origen del error.

Los mejores errores de compilación son los que no se producen en primer lugar, por lo que Visual Studio proporciona Autocompletar e IntelliSense ondea. Los validadores de esquema y herramientas similares

proporcionan el mismo tipo de comentarios. Estos mecanismos guían proactivamente al usuario para construir código bien formado, lo que disminuye la posibilidad de errores de compilación.

Visual Studio proporciona una ventana de herramientas donde los usuarios pueden leer y navegar a través de los errores que se produjeron en sus ventanas de documento. Se proporcionan métodos abreviados de teclado para que el usuario pueda navegar rápidamente por grandes cantidades de código e ir directamente a la ubicación del problema. Visual Studio también permite que cada error de compilación se ata a un identificador de contexto o palabra clave de Ayuda determinado para que el usuario pueda ir directamente a un tema de Ayuda que proporciona información más detallada sobre el error.

Escriba errores de compilación claros y concisos:

- **Utilice un lenguaje sencillo** que explique el problema con poca o ninguna jerga del compilador. El texto de un error de compilación no debe ser demasiado técnico.
- **Esbozar posibles causas.** Por ejemplo, "Falta un signo de dos puntos entre la propiedad y el valor en la declaración '(property) : (value)'."
- Proporcione detalles sobre posibles correcciones. Si no hay suficiente espacio, se pueden poner detalles adicionales en el tema de Ayuda correspondiente.

Componentes de un mensaje de error bien escrito

Utilice el servicio de diálogo de shell para los mensajes de error.

El uso del servicio de diálogo de shell le permite controlar el aspecto del mensaje, las fuentes en particular, sin cambios importantes en elementos individuales. Use los mecanismos **IErrorInfo** e infórmelos mediante **IVsUIShell::SetErrorInfo/ReportErrorInfo**.

Elija una presentación de notificación eficaz y adecuada.

Utilice un cuadro de diálogo modal con una advertencia crítica si se requiere una acción inmediata para evitar la pérdida de datos (notificación sincrónica). Los iconos críticos están reservados para situaciones en las que cerrar el mensaje sin leerlo puede dar lugar a consecuencias negativas. La pérdida de datos es una situación crítica que requiere una respuesta a nivel de alarma. El uso excesivo del icono crítico desensibiliza a los usuarios a su importancia. Si el mensaje de error es de naturaleza informativa, considere alternativas a un cuadro de diálogo modal (notificación asincrónica).

Proporcione una explicación limpia y sucinta de por qué ocurrió el problema en lugar de una explicación técnica.

Sobrecargar a los usuarios con detalles técnicos en la explicación hará que sean más propensos a ignorar los mensajes de error. Ejemplos de buenos mensajes:

- "No se puede abrir el archivo solicitado."
- "No se puede conectar a Internet."

Proporcione información sobre cómo solucionar el problema.

Ofrezca al usuario sugerencias de cómo solucionar el problema. Sea honesto con el usuario si no hay sugerencias. Proporcionar enlaces directos a fuentes alternativas en línea, como soporte técnico o soporte de la comunidad. Intente apuntar a los usuarios a información específica en línea pertinente al problema. Para un identificador de error, considere la posibilidad de vincular a los usuarios a un subproceso de discusión sobre ese error específico. Ejemplos de buenos mensajes:

- "Asegúrese de que está conectado a Internet e intente esta operación de nuevo."
- "Asegúrese de que el archivo existe y que tiene permiso para abrirlo."

Escriba un mensaje que sea corto y al grano.

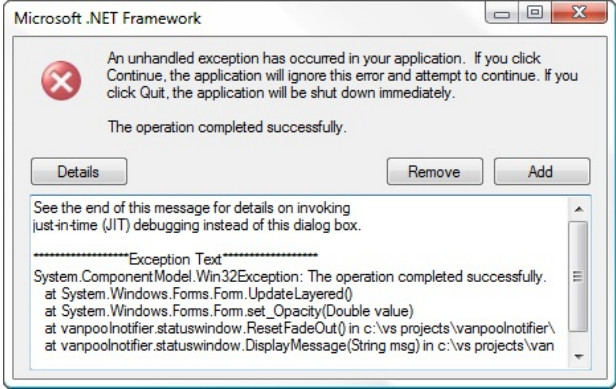
Un mensaje de error puede notificar, explicar y ofrecer una solución, pero aún así se omite si es demasiado wordy. Una solución es usar la divulgación progresiva con un botón de detalles. Por ejemplo, proporcione una breve descripción/solución y, a continuación, coloque más detalles en un botón de detalles. Si los usuarios eligen leer más

información sobre el error, pueden hacerlo.

El idioma del mensaje debe ser:

- **Apropiado para el dominio.** Utilice el idioma que el usuario entenderá. A pesar de que nuestros clientes son desarrolladores, a menudo no tienen el contexto y la terminología que tenemos.
- **Específico.** Evite la redacción vaga y dé nombres y ubicaciones específicos de los objetos involucrados. Por ejemplo, un mensaje de error como "carácter no es válido" no es útil. ¿Qué personaje? "Archivo no encontrado." ¿Qué archivo?
- **Cortés.** No culpes al usuario ni lo hagas sentir estúpido. Evite el lenguaje hostil u ofensivo (matar, ejecutar, terminar, fatal, ilegal). Evite el texto en mayúsculas, que a menudo se ve como gritos y no es tan legible. No uses humor.
- **Correcto.** Utilice la ortografía y la gramática correctas (incluso en alfas). Los typos no son profesionales y embarazosos.
- **Contextualmente apropiado.** Utilice el texto del botón adecuado. Evite el botón "Aceptar" y en su lugar use "Continuar" o "Sí/No."

Ejemplos de mensajes de error

BUENO	INCORRECTO
"El número que marcó ya no está en servicio. Compruebe el número y marque de nuevo o marque 0 para el operador."	- "Error (449): Número ilegal" - "Este error de excepción no controlado indica que la operación se completó correctamente." 

Obtener acceso a la Ayuda

Información general

Además de la documentación de MSDN, un usuario de Visual Studio tiene varios puntos de acceso para ayudar al usuario mientras está en la interfaz de usuario. Para garantizar que estos puntos de acceso estén disponibles de forma coherente, los equipos de entidades deben aprovechar el sistema de ayuda que ofrece el entorno. Estos puntos de acceso son:

- **Texto instructivo y suplementario en diálogos.** Texto estático que proporciona dirección o explicación, ya sea en la superficie de la interfaz de usuario o disponible al pasar el cursor sobre un icono de información.
- **Ayuda de F1 (solo editor).** En el editor de Visual Studio, un usuario puede confiar en que, en cualquier momento, al presionar F1 aparecerá un tema de Ayuda específico para la selección actual. Asegúrese de que los temas asociados con F1 sean adecuados e informativos.
- **Hipervínculos a temas de ayuda.** Un hipervínculo dentro de un cuadro de diálogo, una ventana de

herramientas o una superficie de diseño que inicia un tema para ayudar al usuario a obtener más información sobre una tecnología, capacidad o información sobre cómo realizar una tarea.

- **Mecanismos de interfaz de usuario auxiliares, como etiquetas inteligentes y cuadros de diálogo de creación.** Estos mecanismos ayudan al usuario a comprender un elemento de interfaz de usuario o facilitan una tarea, como etiquetas inteligentes o cuadros de diálogo del generador.
- **Botones de ayuda de la interfaz de usuario (obsoleto).** Un indicador visible en la barra de título que da acceso al tema de Ayuda de F1 relacionado.

Texto

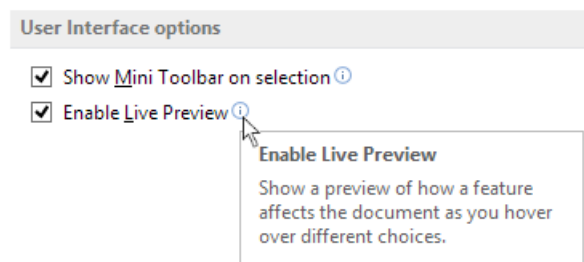
Texto instructivo y suplementario en diálogos

En los cuadros de diálogo que admiten tareas complejas, puede ser necesario proporcionar texto instructivo dentro de la interfaz de usuario, a menudo en la parte superior del cuadro de diálogo o cerca de controles complejos. Consulte texto y terminología de la [interfaz](#) de usuario para obtener más información sobre el estilo de escritura.

Consejos de información

A menudo, el texto instructivo puede ser demasiado largo para colocar en su lugar en la interfaz de usuario o puede ser útil sólo para los nuevos usuarios, sintiéndose como desorden para los usuarios experimentados. En este caso, el texto instructivo/informativo debe colocarse como información sobre herramientas en una información sobre información.

Las sugerencias de información deben colocarse cerca de los controles con los que están relacionados y deben usar el icono de información específica, que es discreto pero perceptible.



Ejemplo de información en Visual Studio

Mecanismos de ayuda interactiva

Ayuda F1

La Ayuda de F1 es necesaria dentro de un editor o una superficie de diseño, pero no en ningún otro lugar del entorno de Visual Studio.

Hipervínculos a temas de ayuda

Los hipervínculos se pueden usar para realizar una acción, navegar dentro del IDE o iniciar la Ayuda en un explorador. Consulte texto y terminología de la [interfaz](#) de usuario para obtener más información sobre el idioma y 07.10.01 Botones e hipervínculos para obtener instrucciones visuales y de diseño.

Botones de ayuda [?] en las barras de título del cuadro de diálogo (obsoleto)

En su mayor parte, los botones Ayuda [?] de la barra de título de los cuadros de diálogo están en desuso. Los temas de la interfaz de usuario ya no forman parte de nuestro modelo de documento y, por lo tanto, es posible que no haya un tema relevante al que vincularse. Esencialmente, el botón de la barra de título era lo mismo que la Ayuda de F1, y eso ya no es necesario en los cuadros de diálogo. En algunos casos, esto todavía se puede utilizar como un indicador de que hay más información conceptual o de procedimiento disponible, aunque los hipervínculos se utilizan más comúnmente en la interfaz de usuario más reciente.

Diálogos creados a través del entorno

Muchos cuadros de diálogo de shell se crean a través de la función **VBDialogBoxParam**. Esta función compartida se actualizó para ayudar a mover el botón **Ayuda** desde el cuadro de diálogo al ? conserve una arquitectura compatible con versiones anteriores y extensible.

En concreto, la función **VBDialogBoxParam** examina la plantilla de cuadro de diálogo de un botón cuyo identificador es **IDHELP** (9) o la etiqueta es **Ayuda** o **&Ayuda**. Si se encuentra un botón Ayuda, se oculta y el estilo **WS_EX_CONTEXTHELP** se agrega al cuadro de diálogo, que coloca el ? en la barra de título del cuadro de diálogo.

Cuando se crea el cuadro de diálogo, inserta el proceso de diálogo en una pila e invoca el cuadro de diálogo con un proceso de diálogo de preprocesamiento denominado **DialogPreProc**. Cuando el ? se hace clic en el botón, envía una **WM_SYSCOMMAND** de **SC_CONTEXTHELP** al cuadro de diálogo. El **DialogPreProc** captura este comando y lo cambia a un mensaje de **WM_HELP**, que se pasa al proceso de diálogo original.

La mayoría de los cuadros de diálogo creados por el entorno tienen un botón Ayuda en el cuadro de diálogo. Cuando se muestra el cuadro de diálogo, el botón Ayuda se oculta automáticamente y solo el ? botón funciona. Si el ? botón se elimina o cambia en Windows, esta solución le permite volver rápidamente a los botones de Ayuda originales.

Esta solución hace cuatro suposiciones que podrían causar errores:

- El botón de ayuda del cuadro de diálogo es **IDHELP** (9).
- El cuadro de diálogo parece correcto cuando el botón Ayuda está oculto.
- El cuadro de diálogo no sustituye su winproc.
- El cuadro de diálogo no está incrustado dentro de otro cuadro de diálogo.

Si el cuadro de diálogo reside en **msenv** y no usa **VBDialogBoxParam**, investigue el aprovechamiento de **VBDialogBoxParam** antes de implementar su propio controlador.

Diálogos creados a través de otros paquetes

Puede implementar su propia solución para los cuadros de diálogo que residen fuera de **msenv**. Para una clase de cuadro de diálogo compartido en el **VSPackage**, considere la posibilidad de mover el botón a la barra de título o implementar un controlador en cada cuadro de diálogo. El código siguiente es un esqueleto de una implementación para ayudarle a empezar:

```

struct DLGPROCITEM
{
    FARPROC proc; // The info used to create the dialog.
    DLGPROCITEM* procPrev;
};

DLGPROCITEM* g_dlgProcStack = NULL;

// A dialog starter/wrapper function is used to push the new
// dialog proc to the top of our dialog proc stack.

int SomeDialogStarterFunction(hinst, id, proc, etc)
{
    if (g_dlgProcStack == NULL)
    {
        g_dlgProcStack = new DLGPROCITEM;
        g_dlgProcStack->procPrev = NULL;
    }
    else
    {
        DLGPROCITEM* procItem = new DLGPROCITEM;
        g_dlgProcStack->procPrev = g_dlgProcStack;
        g_dlgProcStack = procItem;
    }
}

// Pop this dialog proc off the dialog proc stack.

DialogBoxIndirectParam...(...)
{
    DLGPROCITEM* procItem = g_dlgProcStack->procPrev;
    delete g_dlgProcStack;
    g_dlgProcStack = procItem;
}

// A wrapper dialog procedure will allow us to capture the
// SC_CONTEXTHELP button on the title bar from Windows and
// forward it as a simple WM_HELP message back to the dialog.

INT_PTR CALLBACK DialogPreProc(HWND hwndDlg, UINT uMsg,
    WPARAM wParam, LPARAM lParam)
{
    if (uMsg == WM_SYSCOMMAND && wParam == SC_CONTEXTHELP)
    {
        uMsg = WM_HELP;
        wParam = 0;
        lParam = 0;
    }
    return CallWindowProc((WNDPROC)g_dlgProcStack->proc,
        hwndDlg, uMsg, wParam, lParam);
}

```

Botones de ayuda en código administrado

Reemplazar el comportamiento predeterminado del botón Ayuda de la barra de título de la ventana es fácil en el código administrado. A continuación se muestra una aplicación de demostración completa que muestra este comportamiento. En esencia, debe invalidar el método **WndProc** del formulario y, a continuación, activar las solicitudes de ayuda F1 cuando se intercepta un mensaje de **SC_CONTEXTHELP**.

```

using System;
using System.Windows.Forms;

public class HelpForm : Form
{
    private const int SC_CONTEXTHELP = 0xF180;
    private const int WM_SYSCOMMAND = 0x0112;

    public HelpForm()
    {
        this.ClientSize = new System.Drawing.Size(300, 250);
        this.HelpButton = true;
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "HelpForm";
        this.Text = "Help Form";
    }

    protected override void WndProc(ref Message m)
    {
        if (m.Msg == WM_SYSCOMMAND && SC_CONTEXTHELP == (int)m.WParam)
            ShowHelp();
        else
            base.WndProc(ref m);
    }

    private void ShowHelp()
    {
        MessageBox.Show("F1 Help goes here.");
    }

    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.EnableRTLMirroring();
        Application.Run(new HelpForm());
    }
}

```

Vea también

- [Fuentes y formato de Visual Studio](#)
- [Diseño para Visual Studio](#)
- [Notificaciones y progreso para Visual Studio](#)

Diseño para Visual Studio

06/04/2020 • 11 minutes to read • [Edit Online](#)

La mayoría de los cuadros de diálogo de Visual Studio son Diseño de cuadro de [diálogo Utilidad](#), que son los cuadros de diálogo sin tema que siguen los principios de diseño de cuadro de diálogo estándar de Windows [Desktop](#). A medida que Visual Studio se mueve para actualizar su interfaz de usuario, algunos de los cuadros de diálogo más destacados tienen un nuevo diseño que los establece como experiencias de definición de productos. Estos diseños de cuadro de [diálogo temáticos](#) tienen una apariencia temática.

Diseño del cuadro de diálogo de la utilidad

- Todos los controles dentro de un cuadro de diálogo de utilidad deben comenzar en la parte superior /izquierda y fluir hacia abajo.
- Nunca centre los controles en un cuadro de diálogo para rellenar un área grande.
- Utilice la fuente de entorno para todo el texto del cuadro de diálogo. Al escribir una especificación visual, especifique la fuente del entorno en lugar de seleccionar una fuente y un tamaño determinados. Consulte [La fuente del entorno](#).
- Utilice un espaciado de control y una colocación uniformes para apoyar el objetivo de calidad en la artesanía.
- Los cuadros de diálogo pueden volverse más complejos a partir de un mayor número de controles, una yuxtaposición única de controles o ambos. Para esas situaciones complejas, permita un espacio adecuado entre las agrupaciones de control para proporcionar al usuario un flujo lógico para analizar.

Ejemplos de diseño de cuadro de diálogo de utilidad

Todas las dimensiones se expresan como píxeles.

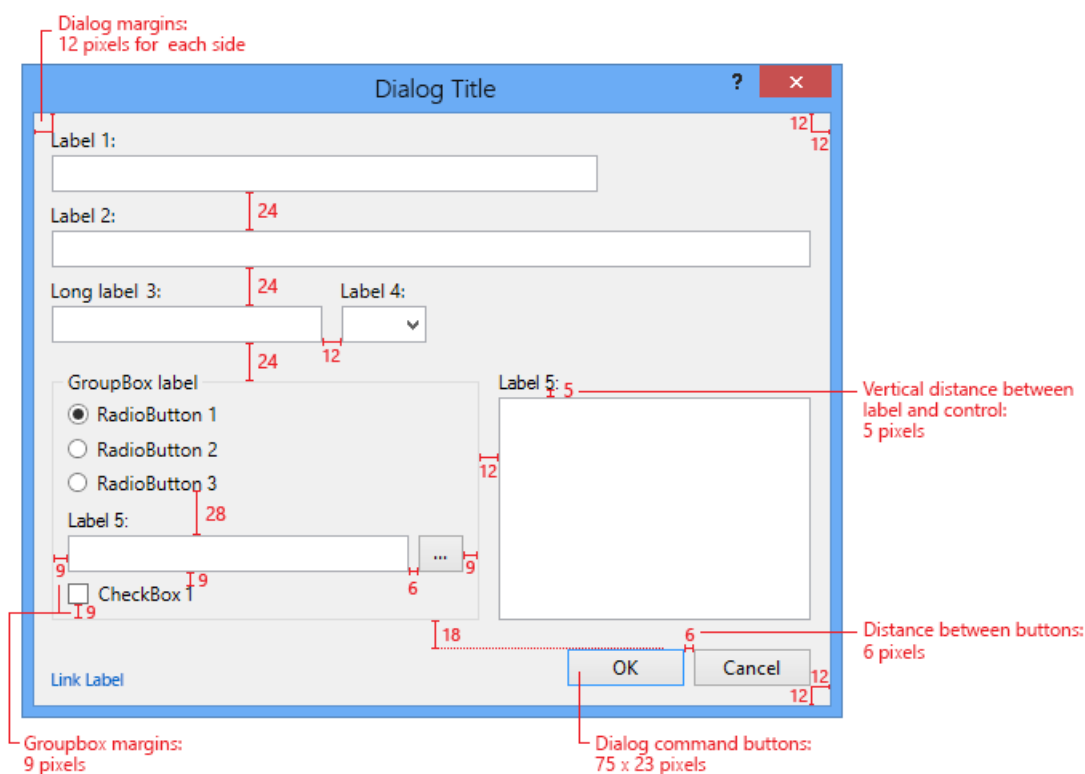


Figura 08.01-a: Directrices de espaciado para cuadros de diálogo de utilidades con etiquetas por

encima de los controles

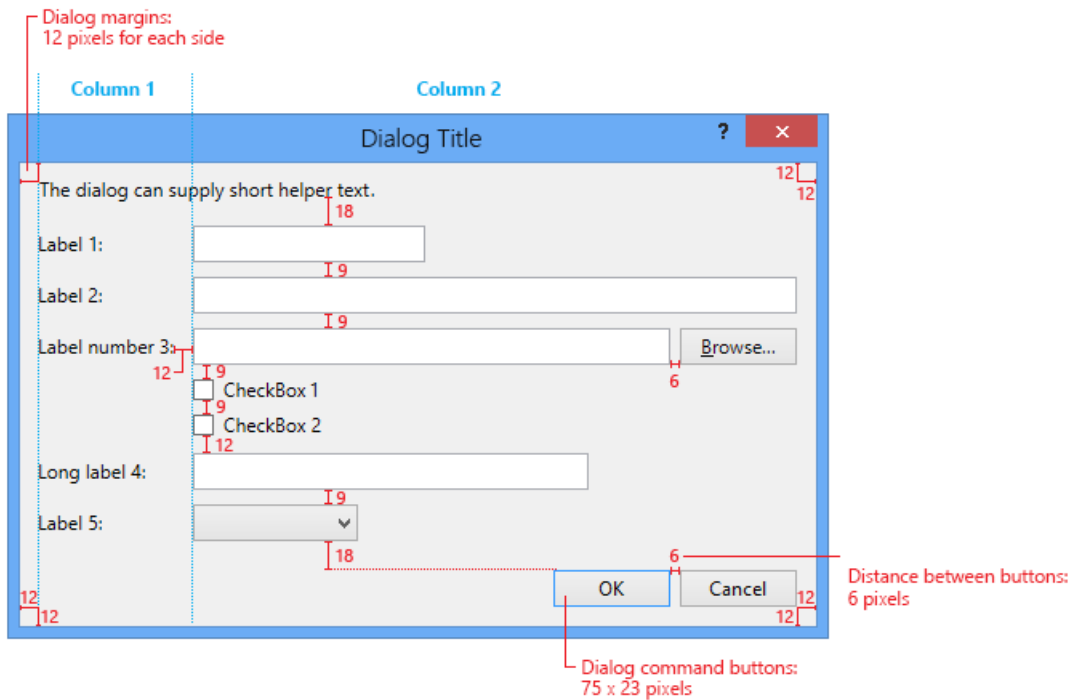


Figura 08.01-b: Directrices de espaciado para diálogos de utilidades con etiquetas a la izquierda de los controles

Detalles del diseño

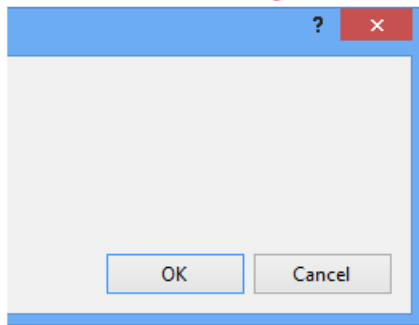
Márgenes

- Todos los cuadros de diálogo deben tener un borde de 12 píxeles alrededor de todos los bordes.
- Los márgenes dentro de un marco de grupo deben estar a 9 píxeles del borde del marco.
- Los márgenes dentro de un control de ficha deben estar a 6 píxeles del borde del control de ficha.

Botones de comando

- Los botones de comando funcionan en el marco de diálogo, no en el contenido. Deben colocarse en la parte inferior derecha y deben tener suficiente espacio variable arriba para establecer los botones claramente separados.
- Si hay botones horizontales que funcionan dentro del cuadro de diálogo, la configuración del botón de comando alternativo es una pila vertical en la parte superior derecha. Consulte los [botones](#) de comando Interior a continuación.
- El espacio a la izquierda de los botones de comando (inferior izquierda/centro del cuadro de diálogo) se considera parte de la "banda" de los controles de operación del cuadro de diálogo. Lo único que debe inmiscuirse en ese espacio es un vínculo de Ayuda que sea relevante para la tarea o el cuadro de diálogo general.
- Los botones de comando deben ser de 75x23 píxeles.
- Los botones de comando deben estar separados por 6 píxeles.

Standard lower horizontal configuration



Alternate vertical configuration

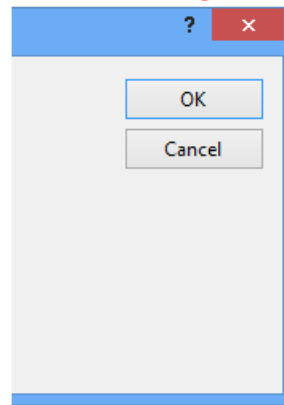


Figura 08.01-c: Alineación básica de botones

Etiquetas

- Alinee a la izquierda todas las etiquetas.
- Para las etiquetas que se encuentran encima de un control, deben alinearse a la izquierda con precisión con el control debajo de él y la parte inferior de la etiqueta debe estar 5 píxeles por encima de la parte superior del otro control (por ejemplo, un cuadro combinado).
- Para las etiquetas que se sentan a la izquierda de los controles, el ancho mínimo entre la etiqueta y el control de entrada es de 10 píxeles. Se debe establecer una segunda columna implícita para alinear los cuadros de texto, cuadros combinados u otros controles.
- Las etiquetas son mayúsculas y minúsculas y van seguidas de dos puntos. Consulte [Estilo de texto](#).

Distancia entre los controles

Controles de pila razonablemente. No hay ninguna directriz absoluta para el espaciado entre los controles apilados. La tensión entre los controles puede variar ligeramente entre los cuadros de diálogo. El espaciado recomendado es de 20 píxeles para los pares verticales de control/etiqueta y de 9 píxeles para los pares horizontales de control/etiqueta. El espaciado de control mínimo para los pares horizontales es de 6 píxeles.

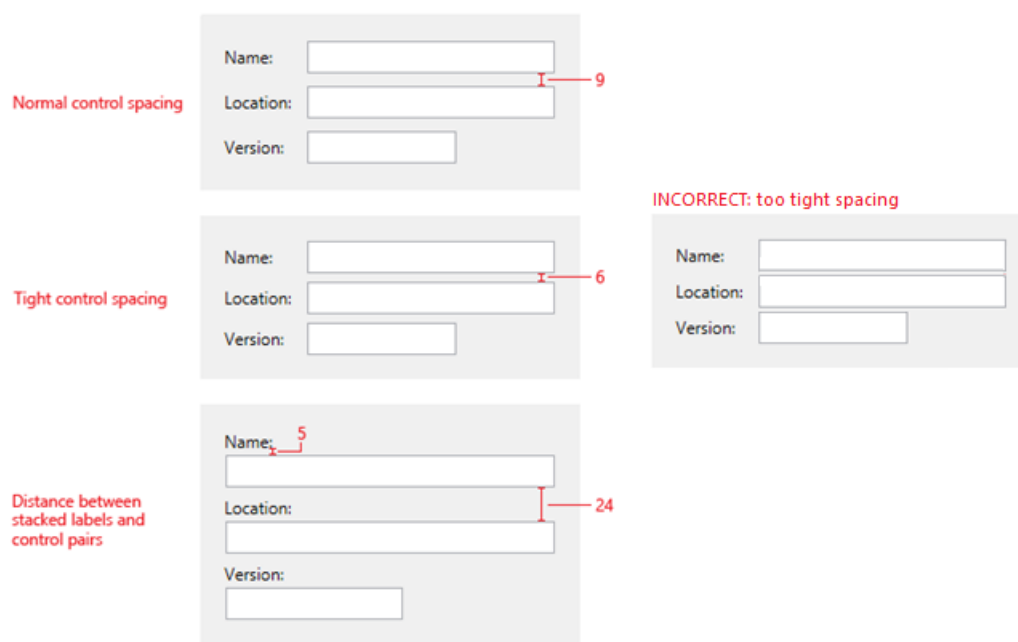


Figura 08.01-d: Recomendaciones para la distancia entre los controles

Sangría de control

Cuando los controles están anidados, alinee los controles internos horizontalmente con el borde izquierdo del control anterior, normalmente la etiqueta.

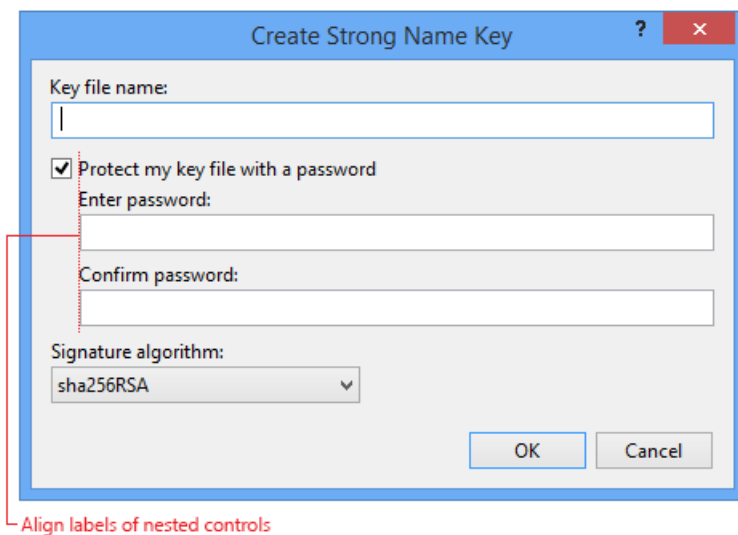


Figura 08.01-e: Alineación de control anidado

Ancho de control

El ancho de un cuadro de texto u otros controles similares no debe ser mayor que la entrada promedio para el campo. La palabra en inglés promedio es de cinco caracteres. Por ejemplo, un cuadro de texto que requiere un nombre de ruta de acceso largo debe ser siempre y cuando el diseño horizontal lo permita, mientras que una lista desplegable para los nombres de plataforma solo debe ser una longitud que permita la entrada más larga.

Texto auxiliar

- Un cuadro de diálogo puede mostrar texto auxiliar que proporciona más información sobre el propósito del cuadro de diálogo. Esto normalmente se sienta en la parte superior y puede ser 1-2 oraciones.
- La longitud de la línea debe ser un ancho cómodo para que un usuario la analice y lea. Un cuadro de diálogo medio no debe tener más de 550 píxeles de ancho.

Botones de comando interiores

En cuadros de diálogo más complejos, un control interno puede tener sus propios botones relacionados, que pueden afectar a dónde se encuentran los botones de confirmación del cuadro de diálogo.

- Utilice una alineación vertical (columna) de botones interiores cuando **OK/Cancel** estén orientados horizontalmente en la esquina inferior derecha.
- Utilice una alineación horizontal (fila) de botones interiores cuando **OK/Cancel** estén orientados verticalmente en la esquina superior derecha. Esta situación es menos común.
- El tamaño del botón interior debe tener como objetivo el tamaño estándar del botón de 75x23 píxeles, que coincida con el tamaño de los botones **Ok/Cancel** cuando sea posible. Si una etiqueta de botón hace que el botón supere el tamaño del botón estándar, los demás botones de ese conjunto deben alinearse con ese tamaño más amplio.

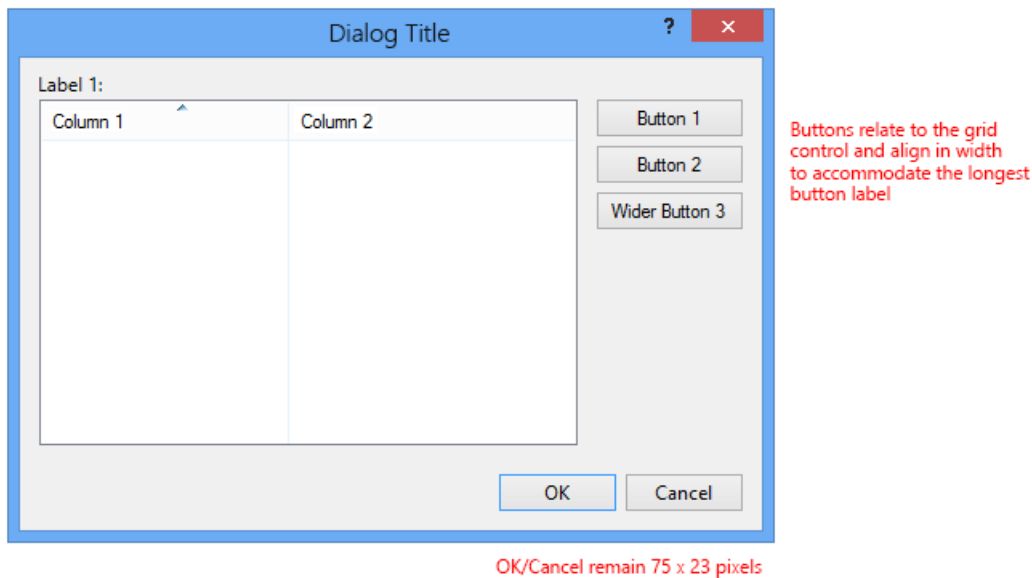


Figura 08.01-f: Botones interiores verticales con OK/Cancelar horizontal

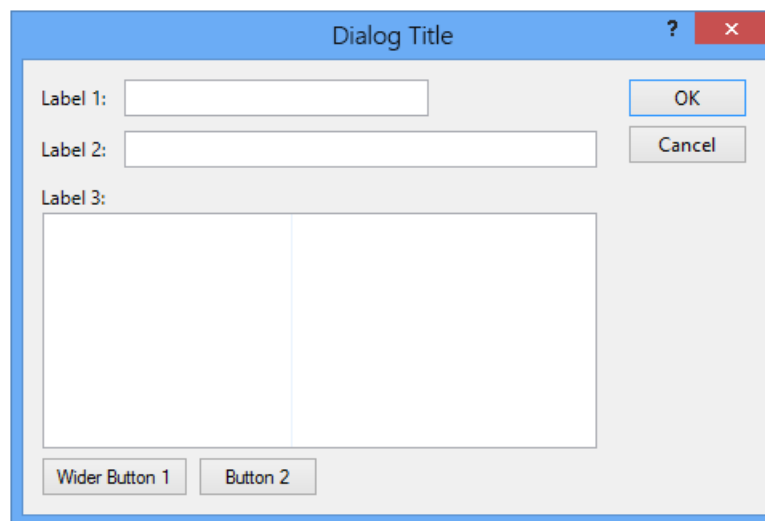


Figura 08.01-g: Botones interiores horizontales con OK/Cancelar vertical

[Examinar...] Botón

[Examinar...] botones que siguen a un cuadro de texto deben deletrear "Examinar..." en su totalidad, incluyendo los puntos suspensivos. Si el espacio es reducido o hay varios botones [Examinar...] en la pantalla, el botón se puede reducir a solo los puntos suspensivos.

Diseño de cuadro de diálogo temático

Los cuadros de diálogo temáticos de Visual Studio tienen un aspecto más claro y ofrecen más espacio en blanco. La tipografía proporciona más énfasis e interés, ofreciendo un espaciado de línea más abierto y una variación de tamaños y pesos de fuente. Siempre que sea posible, las barras de cromo y título se han reducido o eliminado. El diseño de estos cuadros de diálogo debe seguir este patrón básico:

1. El fondo del cuadro de diálogo es blanco.
2. Hay un borde de regla de 1 píxel en un gris de valor medio.
3. El título del cuadro de diálogo ya no se encuentra en una barra de título, pero proporciona interés visual y énfasis en un tamaño de punto más grande. (Consulte la sección de tamaño de fuente en Estilo de [texto](#).)
4. Las etiquetas junto con texto adicional, como una descripción, deben ser Fuente de entorno + **Negrita**.
5. Las columnas interiores están separadas por una regla de 1 píxel en gris claro.

- Los vínculos predeterminados no tienen guiones bajos. Los estados de desplazamiento y presionado tienen un cambio de color más subrayado.
- Los botones de confirmación (como **Ok/Cancel**) se establecen en la esquina inferior derecha.

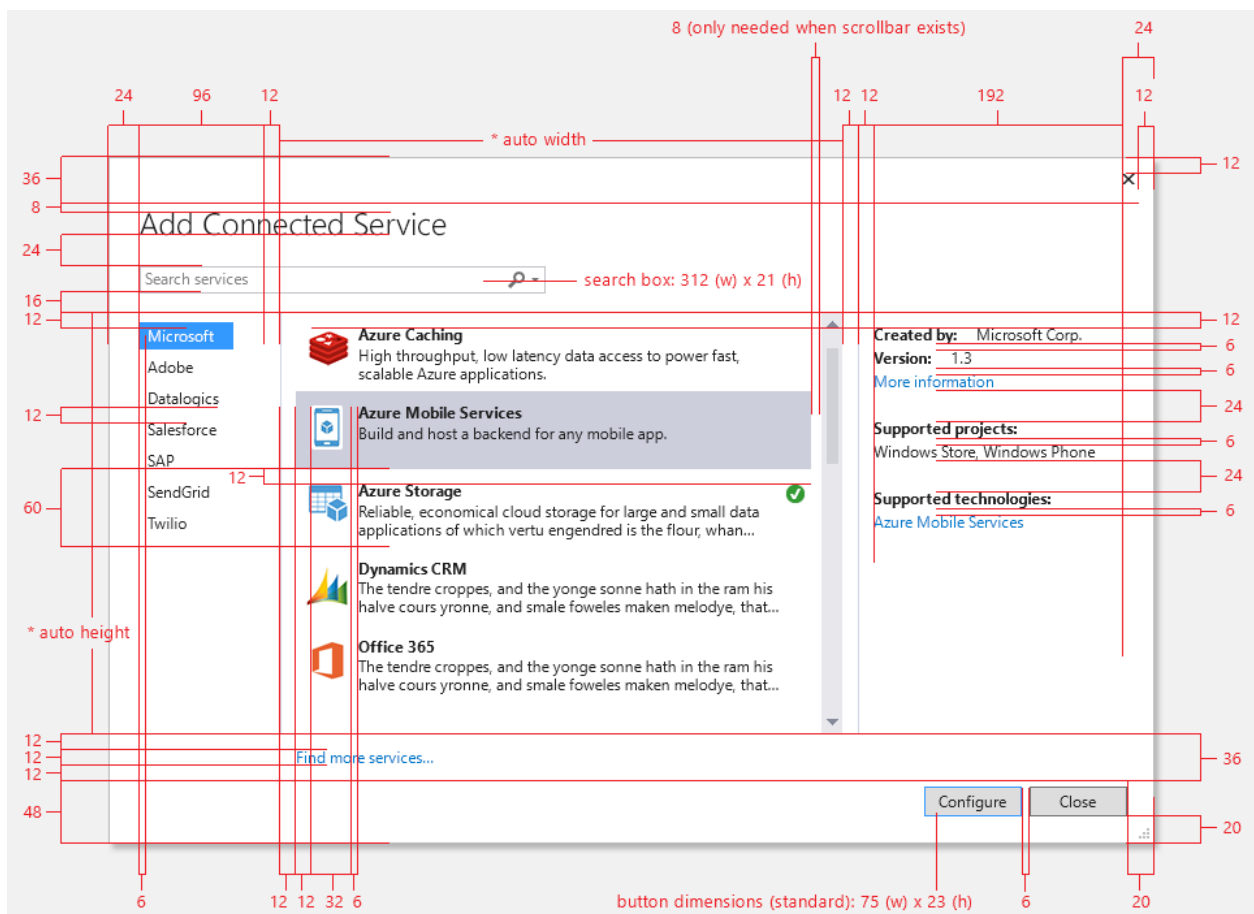
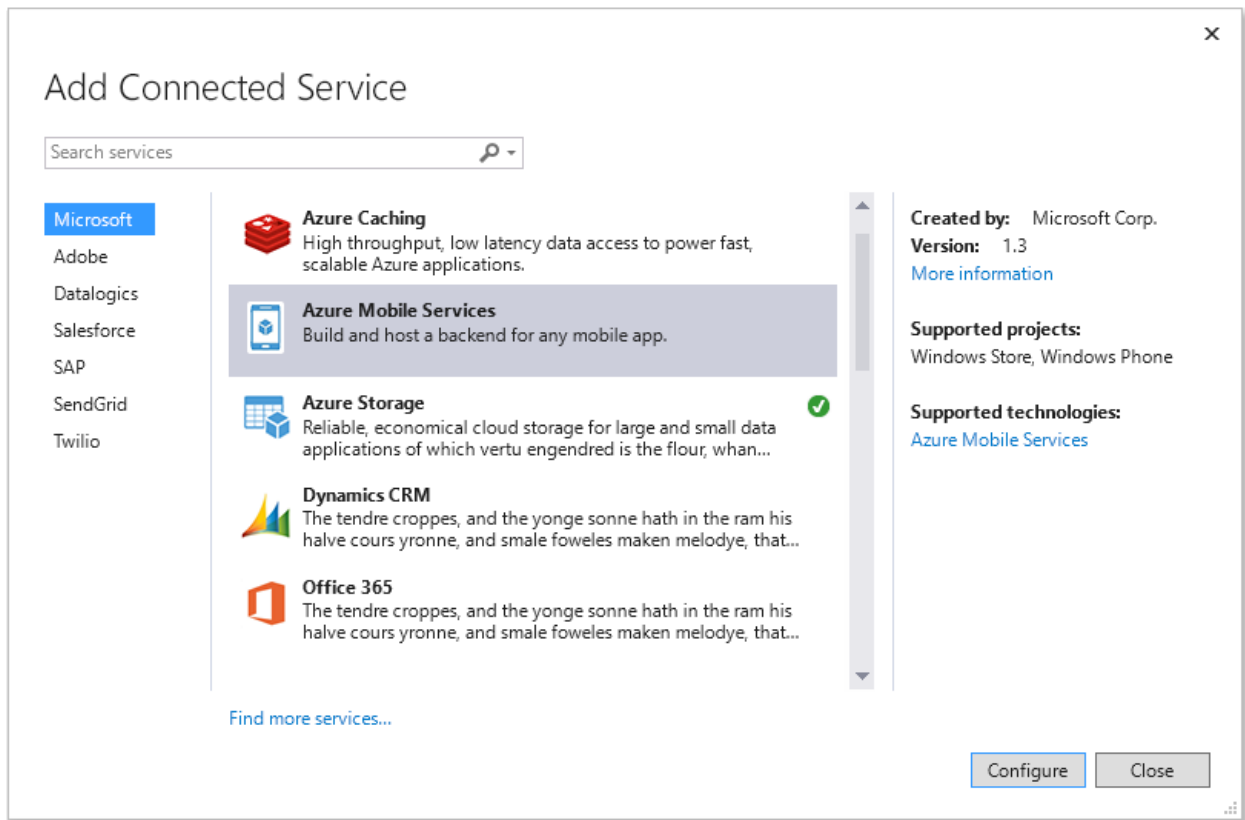


Figura 08.01-i: Diálogo temático - Dimensiones

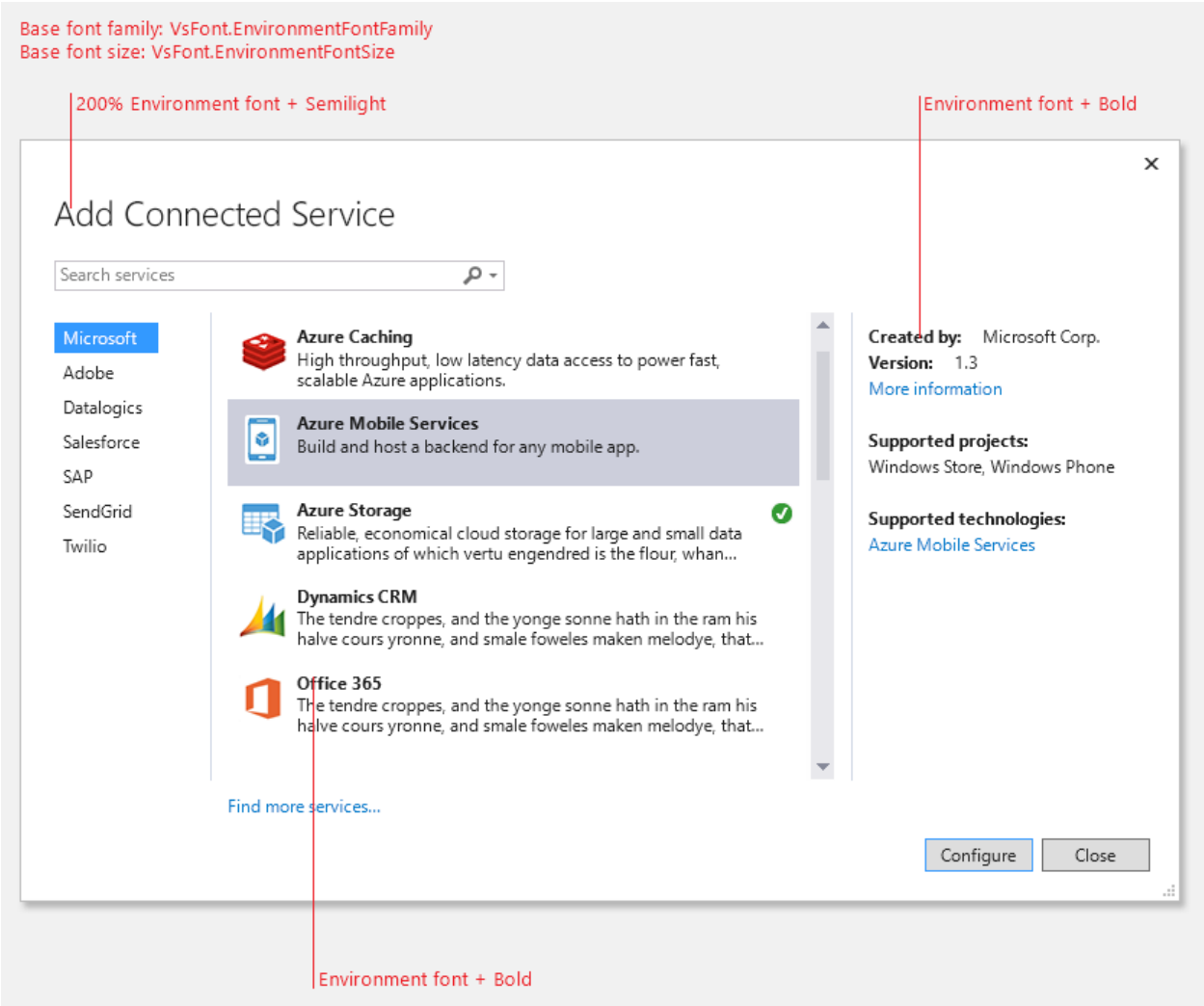


Figura 08.01-j: Diálogo temático - Fuentes

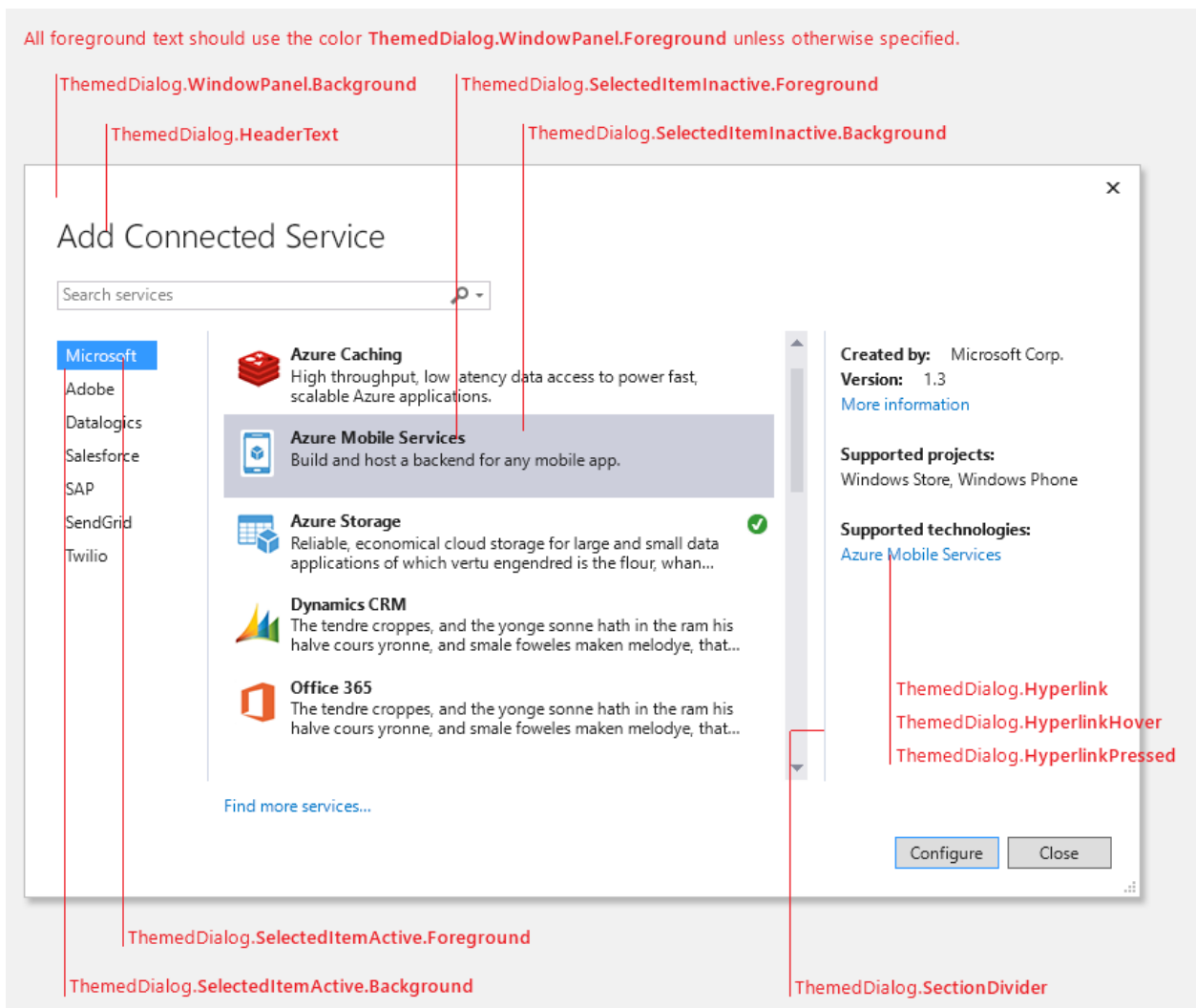


Figura 08.01-k: Diálogo temático - Colores

Vea también

- [Patrones de aplicaciones para Visual Studio](#)
- [Controles \(Windows\)](#)
- [Cuadros de diálogo \(Windows\)](#)

Notificaciones y progreso para Visual Studio

06/04/2020 • 48 minutes to read • [Edit Online](#)

Sistemas de notificación

Información general

Hay varias maneras de informar al usuario lo que está sucediendo en Visual Studio con respecto a sus tareas de desarrollo de software.

Al implementar cualquier tipo de notificación:

- **Mantenga el número de notificaciones al número mínimo efectivo.** Los mensajes de notificación deben aplicarse a la mayoría de los usuarios de Visual Studio o a los usuarios de un área de características o características específica. El uso excesivo de notificaciones puede desviar al usuario o disminuir la facilidad de uso percibida del sistema.
- **Asegúrese** de que está presentando mensajes claros y procesables que el usuario puede usar para invocar el contexto adecuado para tomar decisiones más complejas y tomar medidas adicionales.
- **Presente mensajes sincrónicos y asincrónicos de forma adecuada.** Las notificaciones sincrónicas indican que algo necesita atención inmediata, como cuando se bloquea un servicio web o se produce una excepción de código. El usuario debe ser informado de esas situaciones de inmediato de una manera que requiera su entrada, como en un cuadro de diálogo modal. Las notificaciones asincrónicas son las que el usuario debe conocer pero que no deben tener que actuar de inmediato, como cuando finaliza una operación de compilación o finaliza una implementación de sitio web. Esos mensajes deben ser más ambiente y no interrumpir el flujo de tareas del usuario.
- **Utilice los cuadros de diálogo modales solo cuando sea necesario para evitar que el usuario realice más acciones** antes de reconocer el mensaje o tomar una decisión presentada en el cuadro de diálogo.
- **Elimine las notificaciones ambientales cuando ya no sean válidas.** No exija al usuario que descarte una notificación si ya ha tomado medidas para abordar el problema sobre el que se le notificó.
- **Tenga en cuenta que las notificaciones pueden dar lugar a correlaciones falsas.** Los usuarios pueden creer que una o más de sus acciones ha activado una notificación cuando en realidad no había ninguna relación causal. Sea claro en el mensaje de notificación sobre el contexto, el desencadenador y el origen de la notificación.

Elegir el método correcto

Utilice esta tabla para ayudarle a elegir el método adecuado para notificar al usuario de su mensaje.

MÉTODO	USO	NO DEBE USARSE
Diálogos de mensajes de error modales	Se usa cuando se requiere una respuesta del usuario antes de continuar.	No lo utilice cuando no sea necesario bloquear al usuario e interrumpir su flujo. Evite utilizar cuadros de diálogo modales si es posible mostrar el mensaje de otra manera menos intrusiva.

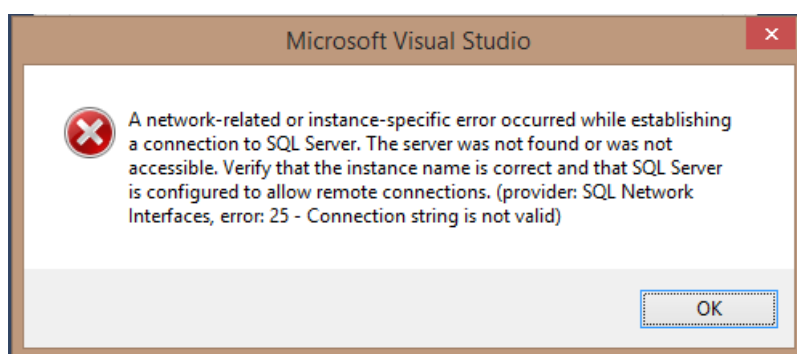
MÉTODO	USO	NO DEBE USARSE
Barra de estado IDE	Se utiliza cuando hay información textual ambiental sobre el estado de un proceso.	No lo use solo. Se utiliza mejor junto con otro mecanismo de retroalimentación.
Barra de información incrustada	En una ventana de herramientas o una ventana de documento, utilice para notificar el progreso, el estado de error, los resultados y/o la información procesable.	No utilizar si la información no es relevante para la ubicación donde se coloca la barra de información. No utilice fuera de una ventana de documento/herramienta.
Cambios en el cursor del ratón	Se puede utilizar para notificar que un proceso está en curso. También se utiliza para notificar que hay un cambio de estado en el mouse, como cuando arrastrar/soltar está en curso o que el cursor del ratón está en un modo determinado, como el modo de dibujo.	No lo utilice para cambios de progreso cortos o si es probable que se alejee el cursor (por ejemplo, cuando está vinculado a partes de un proceso de ejecución más largo en lugar de a todo el proceso).
Indicadores de progreso	Utilícelo cuando necesite informar de progreso (ya sea determinado o indeterminado). Hay una variedad de tipos de indicadores de progreso y uso específico para cada uno. Consulte Indicadores de progreso .	
Ventana de notificaciones de Visual Studio	La ventana Notificaciones no es extensible públicamente. Sin embargo, se usa para comunicar una serie de mensajes sobre Visual Studio, incluidos problemas críticos con la licencia y notificaciones informativas de actualizaciones a Visual Studio o a paquetes.	No usar para otros tipos de notificaciones.
Lista de errores	Cuando el problema se relaciona directamente con la solución actualmente abierta del usuario que tiene un problema (error/advertencia/información), es posible que deba tomar medidas en el código. Esto incluiría, por ejemplo: - Mensajes del compilador (error/advertencia/info) - Analizador de código / Mensajes de diagnóstico sobre el código - Construir mensajes Puede ser adecuado para problemas relacionados con los archivos de proyecto o solución, pero considere primero una indicación del Explorador de soluciones.	No usar para elementos que no tienen ninguna relación con el código de solución abierto del usuario.

MÉTODO	USO	NO DEBE USARSE
Notificaciones del editor: Bombilla de luz	Se usa cuando tiene una corrección disponible para solucionar un problema que existe en el archivo abierto. Tenga en cuenta que la bombilla también se debe usar para hospedar acciones rápidas que se toman en el código del usuario a petición, como las refactorizaciones, pero en ese caso no aparecerá "estilo de notificación."	No utilizar para elementos que no tienen ninguna relación con el archivo abierto.
Notificaciones del editor: Squiggles	Se utiliza para alertar al usuario de un problema con un intervalo determinado de su código abierto (por ejemplo, un ondulado rojo para errores).	No usar para elementos que no se relacionan con un intervalo determinado de su código abierto.
Barras de estado integradas	Se utiliza para proporcionar el estado relacionado con el contenido o el proceso en el contexto de una ventana de herramientas, una ventana de documento o una ventana de diálogo específicas.	No utilice para notificaciones generales de productos, procesos o elementos que no tengan ninguna relación con el contenido dentro de la ventana específica.
Notificaciones de bandejas de Windows	Se utiliza para exponer notificaciones para procesos fuera de proceso o aplicaciones complementarias.	No usar para las notificaciones que son relevantes para el IDE.
Burbujas de notificación	Se usa para notificar un proceso remoto o cambiar fuera del IDE.	No utilizar como medio para notificar al usuario de los procesos dentro del IDE.

Métodos de notificación

Diálogos de mensajes de error modales

Se utiliza un cuadro de diálogo de mensaje de error modal para mostrar un mensaje de error que requiere la confirmación o la acción del usuario.



Un cuadro de diálogo de mensaje de error modal que alerta al usuario de una cadena de conexión no válida a una base de datos

Barra de estado IDE

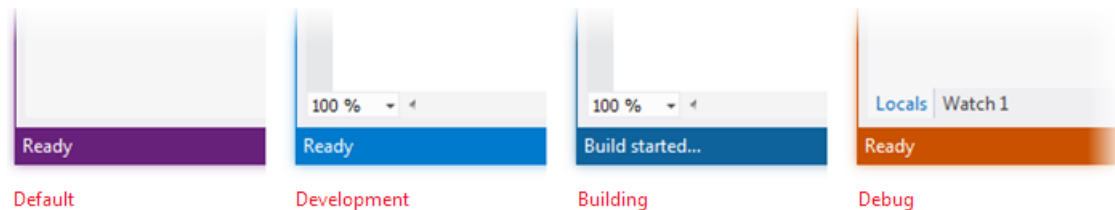
La probabilidad de que los usuarios noten el texto de la barra de estado se correlaciona con su experiencia informática integral y experiencia específica con la plataforma Windows. La base de clientes de Visual Studio tiende a experimentarse en ambas áreas, aunque incluso los usuarios de Windows bien informados podrían perder los cambios en la barra de estado. Por lo tanto, la barra de estado se utiliza mejor con fines informativos o como una señal redundante para la información presentada en otro lugar. Cualquier tipo de información crítica que el usuario debe resolver inmediatamente debe proporcionarse en un cuadro de diálogo o en la ventana de la

herramienta Notificaciones.

La barra de estado de Visual Studio está diseñada para permitir que se muestren varios tipos de información. Se divide en regiones para comentarios, diseñador, barra de progreso, animación y cliente.

La región de comentarios y la región del diseñador siempre están visibles. La barra de progreso y las regiones de animación siempre son dinámicas y se basan en el contexto del usuario. La región del diseñador tiene un ancho estático determinado por la longitud de la cadena que se extrae de un recurso adjunto para el mensaje de texto. Esto permite que la localización cambie el tamaño del ancho sin necesidad de un cambio de código. Para inglés, el ancho de esta cadena es de aproximadamente 220 píxeles. La región del diseñador se comportará normalmente y la región de comentarios absorberá el espacio restante.

La barra de estado también se colorea para agregar interés visual y valor funcional mediante la comunicación de varios cambios de estado IDE, como cuando el IDE está en modo de depuración.



Colores de la barra de estado del IDE

Barra de información integrada

Una barra de información se puede utilizar en la parte superior de una ventana de documento o ventana de herramientas para informar al usuario de un estado o condición. También puede ofrecer comandos para que el usuario pueda tener una manera de tomar medidas fácilmente. La barra de información es un control de vaciado estándar. Evite crear el suyo propio, que actuará y parecerá incoherente con otros en el IDE. Consulte [Barras de información](#) para obtener detalles de implementación y orientación sobre el uso.



Una barra de información incrustada en una ventana de documento, alertando al usuario de que el IDE está en modo de depuración histórica y el editor no responderá de la misma manera que lo hace en el modo de depuración estándar.

Cambios en el cursor del ratón

Al cambiar el cursor del mouse, utilice colores que estén vinculados al servicio VSColor y que ya estén asociados con el cursor. Los cambios del cursor se pueden utilizar para indicar una operación en curso, así como zonas de posicionamiento donde el usuario está pasando el cursor sobre un destino que se puede arrastrar, colocar o utilizar para seleccionar un objeto.

Utilice el cursor del ratón ocupado/espera solo cuando se debe reservar todo el tiempo de CPU disponible para una operación, lo que impide que el usuario exprese cualquier otra entrada. En la mayoría de los casos con aplicaciones bien escritas que utilizan multithreading, las horas en las que se impide a los usuarios realizar otras operaciones deben ser poco frecuentes.

Tenga en cuenta que los cambios de cursor son útiles como una señal redundante para la información presentada en otro lugar. No confíe en un cambio de cursor como la única forma de comunicarse con el usuario, especialmente cuando intenta transmitir algo que es crítico que el usuario debe abordar.

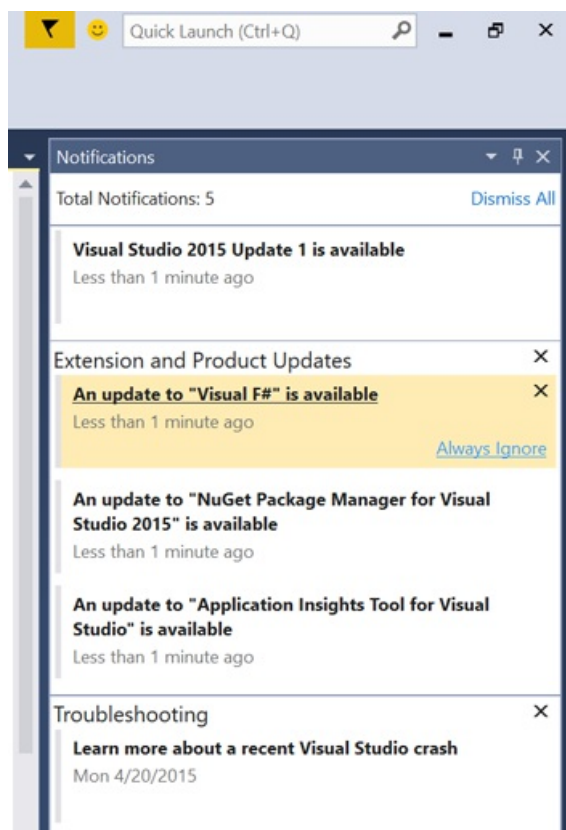
Indicadores de progreso

Los indicadores de progreso son importantes para proporcionar a los usuarios comentarios durante los procesos que tardan más de unos segundos en completarse. Los indicadores de progreso se pueden mostrar in situ (cerca del punto de inicio de la acción en curso), en una barra de estado incrustada, en un cuadro de diálogo modal o en la barra de estado de Visual Studio. Siga las instrucciones en [Los indicadores de progreso](#) con respecto a su uso e implementación.

Ventana Notificaciones de Visual Studio

La ventana Notificaciones de Visual Studio notifica a los desarrolladores acerca de las licencias, el entorno (Visual Studio), las extensiones y las actualizaciones. Los usuarios pueden descartar notificaciones individuales o pueden optar por ignorar ciertos tipos de notificaciones. La lista de notificaciones omitidas se administra en una página **Herramientas > opciones**.

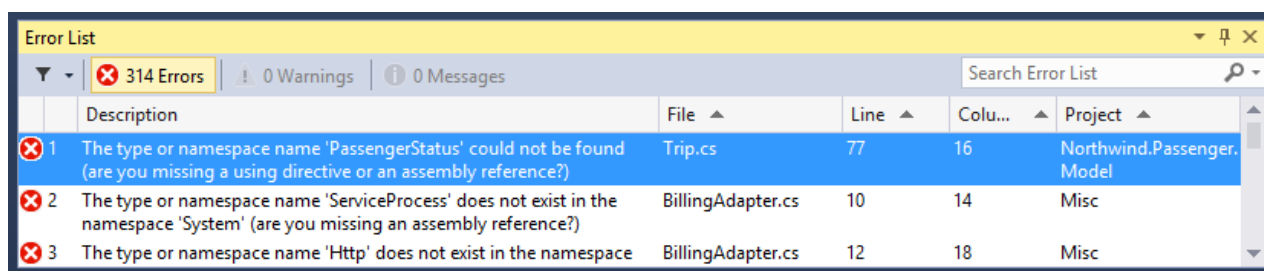
La ventana Notificaciones no es actualmente extensible.



Ventana de la herramienta Notificaciones de Visual Studio

Lista de errores

Una notificación dentro de la lista de errores indica errores y advertencias que se produjeron durante la compilación y el proceso de compilación y permite al usuario navegar en el código a ese error de código específico.

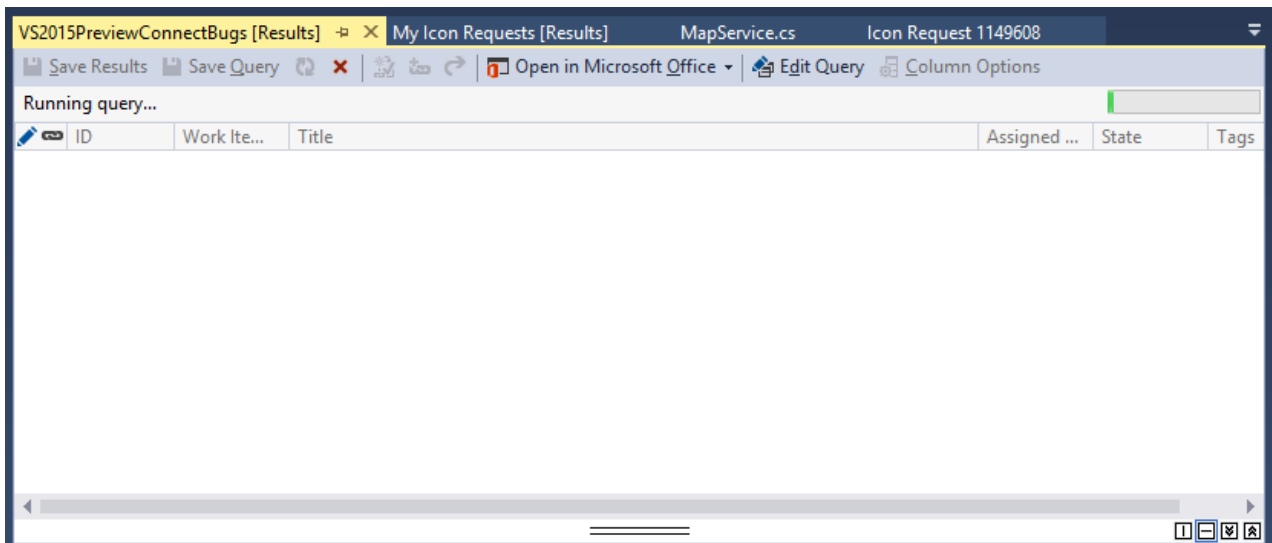


Lista de errores en Visual Studio

Barras de estado integradas

Dado que la barra de estado IDE es dinámica, con su contexto de región de cliente establecido en la ventana de documento activo y la actualización de información sobre el contexto del usuario o las respuestas del sistema, es

difícil mantener una visualización continua de la información o dar estado en los procesos asincrónicos a largo plazo. Por ejemplo, la barra de estado IDE no es adecuada para las notificaciones de resultados de ejecución de pruebas para varias ejecuciones o selecciones de elementos inmediatamente procesables. Es importante conservar dicha información de estado en el contexto del documento o ventana de herramientas donde el usuario realiza una selección o inicia un proceso.



Barra de estado incrustada en Visual Studio

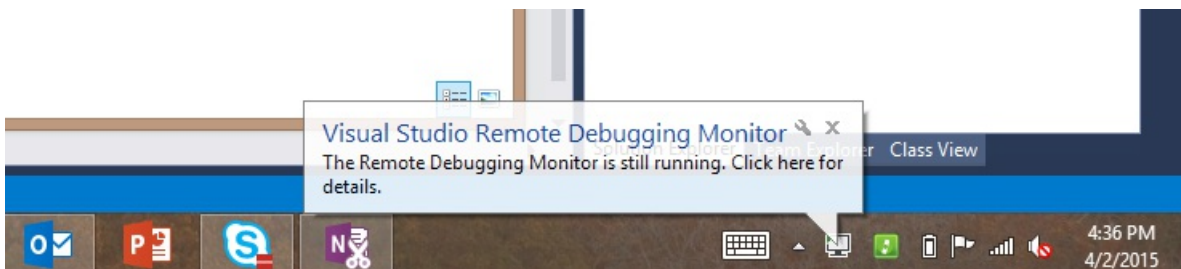
Notificaciones de bandejas de Windows

El área de notificación de Windows está junto al reloj del sistema en la barra de tareas de Windows. Muchas utilidades y componentes de software proporcionan iconos en esta área para que el usuario pueda obtener un menú contextual para las tareas de todo el sistema, como cambiar la resolución de pantalla u obtener actualizaciones de software.

Las notificaciones de nivel de entorno deben aparecer en el centro de notificaciones de Visual Studio, no en el área de notificación de Windows.

Burbujas de notificación

Las burbujas de notificación pueden aparecer como informativas dentro de un editor/diseñador o como parte del área Notificación de Windows. El usuario percibe estas burbujas como problemas que pueden resolver más adelante, lo que es un beneficio para las notificaciones no críticas. Las burbujas son inapropiadas para la información crítica que el usuario debe resolver de inmediato. Si usa burbujas de notificación en Visual Studio, siga las instrucciones de Escritorio de Windows para las burbujas de [notificación](#).



Burbuja de notificación en el área Notificación de Windows utilizada para Visual Studio

Indicadores de progreso

Información general

Los indicadores de progreso son una parte importante de un sistema de notificación para dar comentarios a los usuarios. Indican al usuario cuándo se completarán los procesos y las operaciones. Los tipos de indicadores conocidos incluyen barras de progreso, cursores giratorios e iconos animados. El tipo y la colocación de un indicador de progreso depende del contexto, incluyendo lo que se está notificando y cuánto tiempo tardará el

proceso u operación en completarse.

Factores

Para determinar qué tipo de indicador es apropiado, debe determinar los siguientes factores.

1. **Tiempo:** duración de la operación
2. **Modalidad:** si la operación es modal al entorno (bloquea la interfaz de usuario hasta que se completa el proceso)
3. **Persistente/Transitorio:** si el resultado final del progreso debe ser reportado y/o visible en un momento posterior
4. **Determinado/Indeterminado:** si se puede calcular la hora de finalización y el progreso de la operación
5. **Ubicación gráfica/textual:** si el progreso o el proceso se captura en línea, en el cuerpo de un mensaje o en un control específico, como el control Tree
6. **Proximidad:** si el progreso debe estar muy cerca de la interfaz de usuario con la que está relacionado. (Por ejemplo, ¿puede estar en la barra de estado, que podría estar lejos, o tiene que estar cerca del botón que inició el proceso?)

Progreso determinado

TIPO DE PROGRESO	CUÁNDO Y CÓMO USAR	NOTAS
Barra de progreso (determinada)	Duración prevista de >5 segundos. Puede incluir una descripción textual de los detalles del proceso.	NO inserte texto en la animación.
Barra de información	Mensajería asociada a la interfaz de usuario contextual. Consulte Barras de información . Puede incluir una descripción textual de los detalles del proceso.	NO utilice varias barras de información cuando necesite indicar varios procesos. Utilice barras de progreso apiladas en su lugar.
Ventana de salida	Notificación transitoria: proceso de nivel de aplicación del que el usuario desea revisar los detalles después de la finalización.	NO utilice si el usuario tendrá que hacer referencia a los datos más adelante.
Archivo de registro	Emparejado con notificación intransitoria en los casos en que es importante guardar los detalles después de la finalización.	
Barra de estado	Notificación transitoria: proceso de nivel de aplicación que el usuario no necesitará detalles de después de la finalización. Incluye una barra de progreso incrustada. Puede incluir una descripción textual de los detalles del proceso.	

Progreso indeterminado

TIPO DE PROGRESO	CUÁNDO Y CÓMO USAR	NOTAS
Barra de progreso (indeterminada)	Duración prevista de >5 segundos. Puede incluir una descripción textual de los detalles del proceso.	NO inserte texto en la animación.
Hormigas (puntos horizontales animados)	Viaje de ida y vuelta al servidor. Se coloca cerca del punto de contexto en la parte superior del contenedor principal.	NO lo use si no es padre de todo el contenedor.
Spinner (anillo de progreso)	Proceso asociado con la interfaz de usuario contextual o donde el espacio es una consideración. Puede incluir una descripción textual de los detalles del proceso.	
Barra de información	Mensajería asociada a la interfaz de usuario contextual. Consulte Barras de información .	NO utilice varias barras de información cuando necesite indicar varios procesos. Utilice barras de progreso apiladas en su lugar.
Ventana de salida	Notificación transitoria: proceso de nivel de aplicación que el usuario querrá revisar los detalles de después de la finalización.	NO se use para obtener información que desee conservar entre sesiones.
Archivo de registro	Emparejado con notificación intransitoria en los casos en que es importante guardar los detalles después de la finalización.	
Barra de estado	Notificación transitoria: proceso de nivel de aplicación que el usuario no necesitará detalles de después de la finalización. Incluye barra de progreso incrustada. Puede incluir una descripción textual de los detalles del proceso.	

Tipos de indicadores de progreso

Barras de progreso

Indeterminado



Barra de progreso indeterminado

"Indeterminado" significa que no se puede determinar el progreso general de una operación o proceso. Utilice barras de progreso indeterminadas para operaciones que requieran una cantidad de tiempo sin límites o que tienen acceso a un número desconocido de objetos. Utilice una descripción textual para acompañar lo que está sucediendo. Use tiempos de espera para dar límites a operaciones basadas en tiempo. Las barras de progreso indeterminadas utilizan animaciones para mostrar que se está progresando, pero no proporcionan otra información. No elija una barra de progreso indeterminada basada únicamente en la posible falta de precisión por

sí sola.

Determinados



Barra de progreso determinada

"Determinado" significa que una operación o proceso requiere una cantidad limitada de tiempo, incluso si esa cantidad de tiempo no se puede predecir con precisión. Indique claramente la finalización. No permita que una barra de progreso vaya al 100 por ciento a menos que la operación se haya completado. La animación de la barra de progreso determinada se mueve de izquierda a derecha de 0 a 100%.

Nunca mueva el indicador de progreso hacia atrás durante una operación. La barra debe avanzar hacia adelante constantemente cuando la operación comienza y alcanzar el 100% cuando termina. El punto de la barra de progreso es dar al usuario una idea de cuánto tiempo tarda toda la operación, independientemente de cuántos pasos estén involucrados.

Informes simultáneos (barras de progreso apiladas)

Si una operación tarda mucho tiempo - tal vez varios minutos - entonces se pueden utilizar dos barras de progreso, una que muestra el progreso general de una operación y otra para la progresión del paso actual. Por ejemplo, si un programa de instalación está copiando muchos archivos, se puede utilizar una barra de progreso para indicar cuánto tiempo tarda todo el proceso, mientras que un segundo puede indicar qué porcentaje del archivo o directorio actual se está copiando. No informe de más de cinco operaciones o procesos simultáneos mediante barras de progreso apiladas. Si tiene más de cinco operaciones o procesos simultáneos para informar, utilice un cuadro de diálogo modal con un botón Cancelar e informe de los detalles de progreso a la ventana de salida.

Descripciones textuales

Utilice una descripción textual para acompañar lo que está sucediendo y el tiempo estimado hasta su finalización. Si es imposible determinar cuánto tiempo tardará una operación, entonces una mejor opción para dar comentarios podría ser un icono animado en lugar de una barra de progreso.

Visual Studio proporciona una barra de progreso estándar en la barra de estado que puede usar cualquier producto integrado en Visual Studio. Para las descripciones textuales de lo que está sucediendo mientras se anima la barra de progreso, el texto de la barra de estado se puede actualizar.

Otros indicadores de progreso

Hormigas (puntos horizontales animados)



"Ants", los puntos horizontales animados, proporcionan una referencia visual para un proceso de servidor de ida y vuelta indeterminado.

Spinner (anillo de progreso)



El spinner (también conocido como "anillo de progreso") es un indicador de progreso indeterminado utilizado principalmente en relación con la interfaz de usuario contextual. Mostrar un spinner muy cerca de su contenido relacionado, como un encabezado de categoría textual, mensajería o control.

Retroalimentación del cursor

Para las operaciones que tardan entre 2 y 7 segundos, proporcione comentarios del cursor. Normalmente, esto significa usar el cursor de espera proporcionado por el sistema operativo. Para obtener instrucciones, vea el artículo de MSDN [Cursors.Wait \(Propiedad\)](#).

Ubicaciones de los indicadores de progreso

Barra de estado

La barra de estado proporciona a la aplicación un lugar para mostrar mensajes e información útil al usuario sin

interrumpir el trabajo del usuario. Normalmente se muestra en la parte inferior de una ventana, el estado del progreso será un panel de información sobre herramientas que incluye un mensaje sobre la medida del progreso en combinación con un indicador de barra de progreso.



Barra de estado con barra de progreso



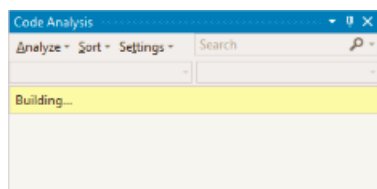
Barra de estado con descripción textual

Barra de información

Al igual que la barra de estado, la barra de información proporciona notificación contextual y mensajería, que también se puede emparejar con indicadores de progreso indeterminados, como la barra de progreso o el spinner. La barra de información no debe proporcionar un progreso de nivel granular ni una indicación de progreso determinada. Consulte [Barras de información](#).

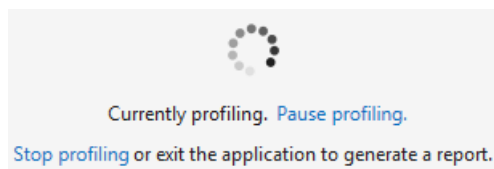


Barra de información con barra de progreso y descripción textual

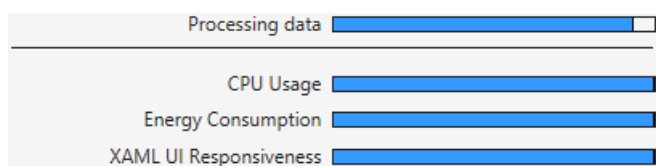


En línea

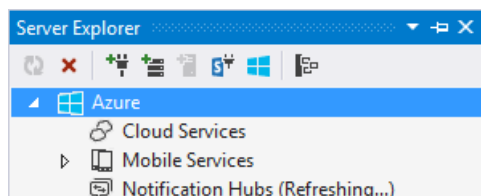
La indicación de progreso en línea se puede representar mediante cualquiera de los tipos de cargador de progreso. Normalmente, el indicador de progreso se empareja con la mensajería, pero esto no es un requisito.



Spinner combinado con descripción textual



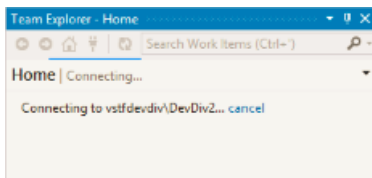
Barra de progreso apiladas determinadas



Texto en línea del Explorador de servidores: Actualizar...

Ventanas de herramientas

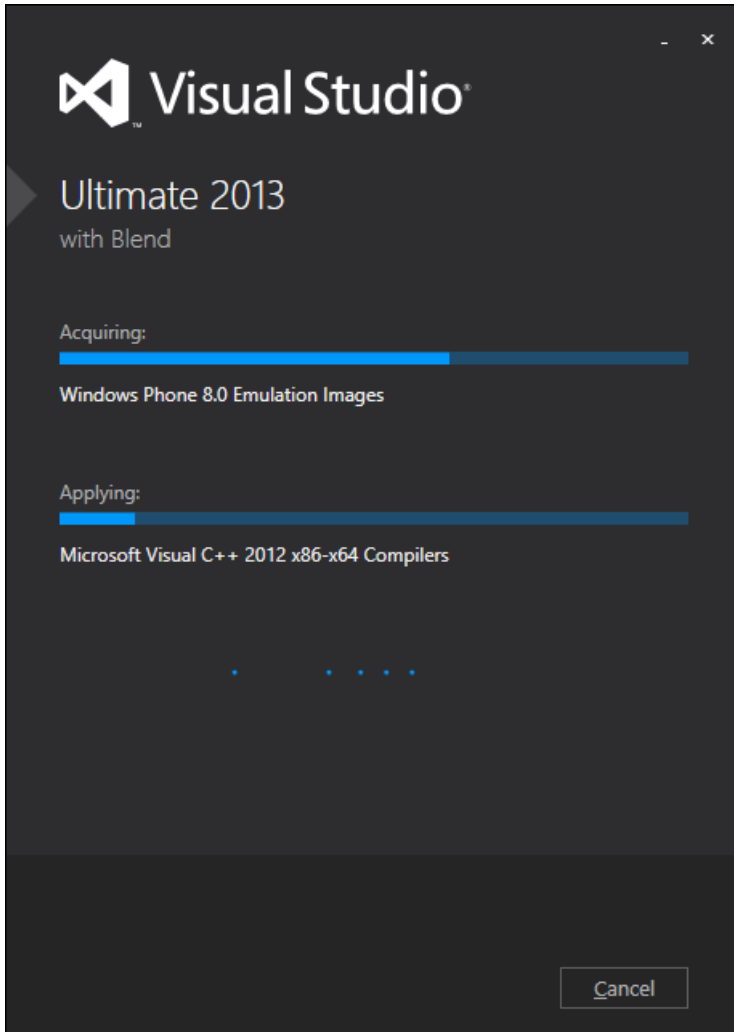
La indicación de progreso global se representa mediante una barra de progreso indeterminada situada directamente debajo de la barra de herramientas.



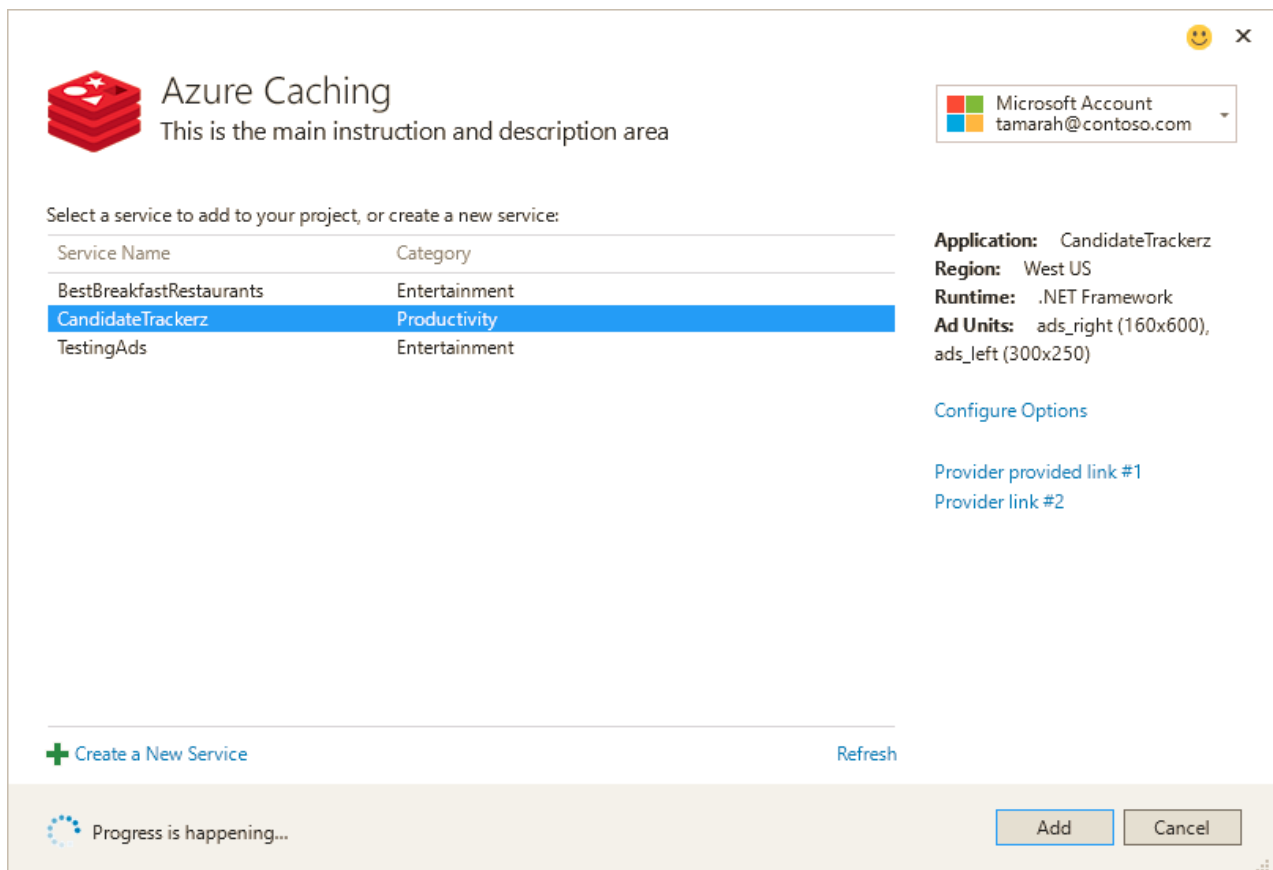
Barra de progreso indeterminada global de Team Explorer

Cuadros de diálogo

Los cuadros de diálogo pueden contener cualquiera de los tipos de cargador de progreso. Los indicadores de progreso se pueden emparejar con mensajería, así como combinarse con múltiples niveles de indicación de progreso para representar procesos granulares y sub-procesos.



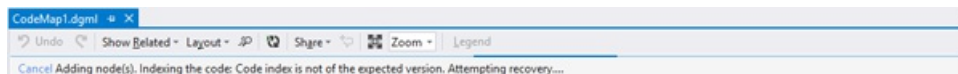
Cuadro de diálogo de Visual Studio con procesos simultáneos y varios tipos de indicadores de progreso



Diálogo de Visual Studio con cargador de progreso y comandos en línea de mensajería

Documento bien

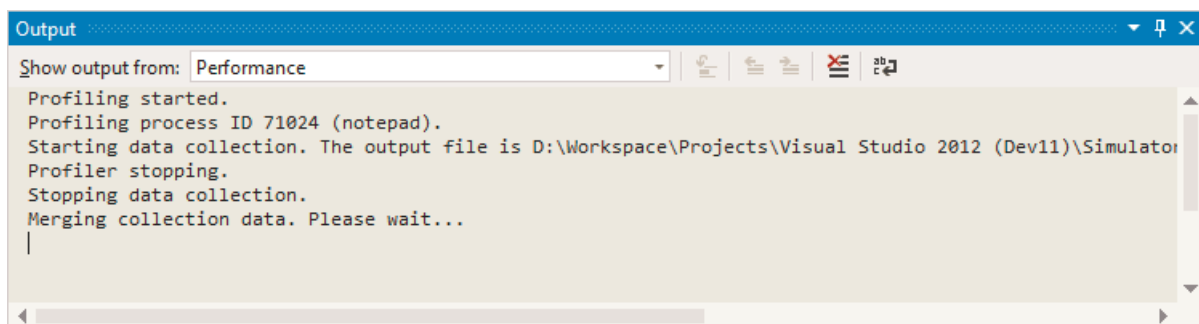
El pozo de documentos puede mostrar varios tipos de cargadores de progreso en combinación con controles.



Barra de progreso indeterminada debajo de la barra de herramientas

Ventana de salida

La ventana Salida es adecuada para controlar la progresión del proceso y el estado de progreso continuo a través de la mensajería textual en línea. Debe utilizar la barra Estado junto con cualquier informe de progreso de la ventana de salida.



Ventana de salida con estado de proceso continuo y mensajería de espera

Barras de información

Información general

Las barras de información proporcionan al usuario un indicador cercano a su punto de atención y el uso del control de la barra de información compartida garantiza la coherencia en la apariencia visual y la interacción.



Infobars en Visual Studio

Usos adecuados para una barra de información

- Para dar al usuario un mensaje importante y sin bloqueo relevante para el contexto actual
- Para indicar que la interfaz de usuario está en un determinado estado o condición que contiene algunas implicaciones de interacción, como la depuración histórica
- Notificar al usuario que el sistema ha detectado problemas, como cuando una extensión está causando problemas de rendimiento
- Proporcionar al usuario una forma de tomar medidas fácilmente, como cuando el editor detecta que un archivo tiene pestañas y espacios mixtos

Sí:

- Mantenga el texto del mensaje de la barra de información corto y al punto.
- Mantenga el texto en enlaces y botones sucintos.
- Asegúrese de que las opciones de "acción" que proporcione a los usuarios sean mínimas, lo que muestra solo las acciones necesarias.

No:

- Utilice una barra de información para ofrecer comandos estándar que se deben colocar en una barra de herramientas.
- Utilice una barra de información en lugar de un cuadro de diálogo modal.
- Cree un mensaje flotante fuera de una ventana.
- Utilice varias barras de información en varias ubicaciones dentro de la misma ventana.

¿Pueden mostrarse varias barras de información al mismo tiempo?

Sí, varias barras de información pueden mostrarse al mismo tiempo. Se mostrarán por orden de llegada con la primera barra de información que aparece en la parte superior y barras de información adicionales que se muestran a continuación.

El usuario verá un máximo de tres barras de información a la vez, después de lo cual, si hay más barras de información disponibles, la región de la barra de información se convertirá en desplazable.

Creación de una barra de información

La barra de información tiene cuatro secciones, de izquierda a derecha:

- **Icono:** Aquí es donde agregarías cualquier icono que quieras mostrar para la barra de información, como un icono de advertencia.
- **Texto:** Puede agregar texto para describir el escenario o la situación en la que se encuentra el usuario, junto con vínculos dentro del texto, si es necesario. Recuerde mantener el texto sucinto.
- **Acciones:** Esta sección debe contener enlaces y botones para las acciones que el usuario puede realizar en la barra de información.
- **Botón Cerrar:** La última sección a la derecha puede tener un botón de cierre.

Creación de una barra de información estándar en código administrado

La clase InfoBarModel se puede utilizar para crear un origen de datos para una barra de información. Utilice uno de estos cuatro constructores:

```
public InfoBarModel(IEnumerable<IVsInfoBarTextSpan> textSpans, ImageMoniker image = default(ImageMoniker),
    bool isCloseButtonVisible = true);
```

```
public InfoBarModel(string text, ImageMoniker image = default(ImageMoniker), bool isCloseButtonVisible =
    true);
```

```
public InfoBarModel(IEnumerable<IVsInfoBarTextSpan> textSpans, IEnumerable<IVsInfoBarActionItem> actionItems,
    ImageMoniker image = default(ImageMoniker), bool isCloseButtonVisible = true);
```

```
public InfoBarModel(string text, IEnumerable<IVsInfoBarActionItem> actionItems, ImageMoniker image =
    default(ImageMoniker), bool isCloseButtonVisible = true);
```

Este es un ejemplo que crea un InfoBarModel con texto con un hipervínculo, un botón de acción y un icono.



```
var infoBar = new InfoBarModel(
    textSpans: new[]
    {
        new InfoBarTextSpan("This is a "),
        new InfoBarHyperlink("hyperlink"),
        new InfoBarTextSpan(" InfoBar.")
    },
    actionItems: new[]
    {
        new InfoBarButton("Click Me")
    },
    image: KnownMonikers.StatusInformation,
    isCloseButtonVisible: true);
```

Creación de una barra de información estándar en código nativo

Implemente la interfaz IVsInfoBar para proporcionar una barra de información desde código nativo.

```
public interface IVsInfoBar
{
    IVsInfoBarActionItemCollection ActionItems { get; }
    ImageMoniker Image { get; }
    bool IsCloseButtonVisible { get; }
    IVsInfoBarTextSpanCollection TextSpans { get; }
}
```

Obtener una barra de información UIElement desde una barra de información

El InfoBarModel o IVsInfoBar implementación son modelos de datos que se deben convertir en un UIElement para que se muestre en la interfaz de usuario. Un UIElement se puede recuperar con el SVsInfoBarUIFactory/IVsInfoBarUIFactory servicio.

```
private bool TryCreateInfoBarUI(IVsInfoBar infoBar, out IVsInfoBarUIElement uiElement)
{
    IVsInfoBarUIFactory infoBarUIFactory = serviceProvider.GetService(typeof(SVsInfoBarUIFactory)) as
IVsInfoBarUIFactory;
    if (infoBarUIFactory == null)
    {
        uiElement = null;
        return false;
    }

    uiElement = infoBarUIFactory.CreateInfoBar(infoBar);
    return uiElement != null;
}
```

Ubicación

Las barras de información se pueden mostrar en una o más de las siguientes ubicaciones:

- Ventanas de herramientas
- Dentro de una pestaña de documento

IMPORTANT

Es posible colocar una barra de información para dar un mensaje sobre el contexto global. Esto aparecería entre las barras de herramientas y el documento bien. Esto no se recomienda porque causa problemas con "salto y tirón" del IDE y debe evitarse a menos que sea absolutamente necesario y apropiado.

Colocar una barra de información en un ToolWindowPane

El ToolWindowPane.AddInfoBar(IVsInfoBar) método se puede utilizar para agregar una barra de información a una ventana de herramientas. Esta API puede agregar un IVsInfoBar (de los cuales InfoBarModel es una implementación predeterminada) o un IVsUIElement.

Colocar una barra de información en un documento o en un panel que no sea ToolWindowPane

Para colocar una barra de información en cualquier IVsWindowFrame, use la propiedad VSFPROPID_InfoBarHost para obtener el IVsInfoBarHost para el marco y, a continuación, agregue la barra de información UIElement.

```
private void AddInfoBar(IVsWindowFrame frame, IVsUIElement uiElement)
{
    IVsInfoBarHost infoBarHost;
    if (TryGetInfoBarHost(frame, out infoBarHost))
    {
        infoBarHost.AddInfoBar(uiElement);
    }
}
private bool TryGetInfoBarHost(IVsWindowFrame frame, out IVsInfoBarHost infoBarHost)
{
    object infoBarHostObj;
    if (ErrorHandler.Failed(frame.GetProperty((int)__VSFPROPID7.VSFPROPID_InfoBarHost, out infoBarHostObj)))
    {
        infoBarHost = null;
        return false;
    }

    infoBarHost = infoBarHostObj as IVsInfoBarHost;
    return infoBarHost != null;
}
```

Colocación de una barra de información en la ventana principal

Para colocar una barra de información en la ventana principal, use la VSSPROPID_MainWindowInfoBarHost del

servicio IVsShell para obtener IVsInfoBarHost de la ventana principal y, a continuación, agregue la barra de información UIElement a ella.

¿Sabré cuándo el usuario realiza una acción en mi barra de información?

Sí, devolveremos cada acción de evento al autor de la barra de información. A continuación, depende del autor de la barra de información para realizar una acción en el IDE en función de la selección del usuario en la barra de información. Las barras de información se eliminarán automáticamente del host en el que se hizo clic en el botón Cerrar, pero se requiere trabajo adicional si es necesario eliminar otras barras de información después de cerrar. La telemetría también tendrá que ser registrada independientemente por cada barra de información.

Recepción de eventos de barra de información en un ToolWindowPane

ToolWindowPane tiene dos eventos para las barras de información. El evento InfoBarClosed se produce cuando se cierra una barra de información en El ToolWindowPane. El evento InfoBarActionItemClicked se produce cuando se hace clic en un hipervínculo o botón dentro de la barra de información.

Recepción de eventos de barra de información directamente desde UIElement

IVsInfoBarUIElement.Advise se puede utilizar para suscribirse a eventos directamente desde UIElement de una barra de información. Implementar IVsInfoBarUIEvents permitirá al autor recibir eventos de cierre y clic.

```
public interface IVsInfoBarUIEvents
{
    void OnActionItemClicked(IVsInfoBarUIElement infoBarUIElement, IVsInfoBarActionItem actionItem);
    void OnClosed(IVsInfoBarUIElement infoBarUIElement);
}
```

Validación de errores

Cuando un usuario escribe información que no es aceptable, como cuando se omite un campo obligatorio o cuando se introducen datos en el formato incorrecto, es mejor usar la validación del control o comentarios cerca del control en lugar de usar un cuadro de diálogo de error emergente de bloqueo.

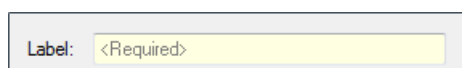
Validación de campos

La validación de formularios y campos consta de tres componentes: un control, un icono y una información sobre herramientas. Aunque varios tipos de controles pueden usar esto, se usará un cuadro de texto como ejemplo.



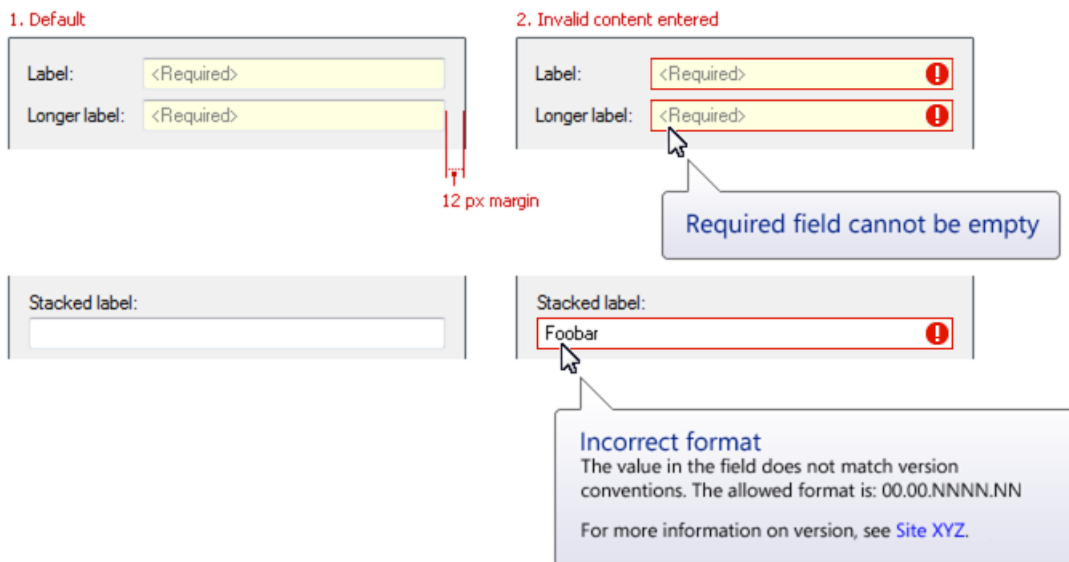
Si el campo es obligatorio, debe ** <** haber texto de marca de agua que

`Environment.ControlEditRequiredBackground` indique Requerido>y el `Environment.ControlEditRequiredHintText` fondo del campo debe ser amarillo claro (VSColor:) y el primer plano debe ser gris (VSColor:):



El programa puede determinar que el control está en un estado de *contenido no válido introducido* cuando el foco se mueve a otro control o cuando el usuario hace clic en un botón de confirmación [OK] o cuando el usuario guarda el documento o formulario.

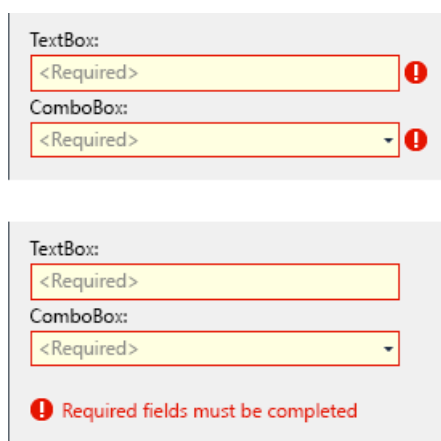
Cuando se determina el estado de contenido no válido, aparece un icono dentro del control o justo al lado de él. Una información sobre herramientas que describe el error debe aparecer al mantener el mouse del icono o del control. Además, debe aparecer un borde de 1 píxel alrededor del control que está creando el estado no válido.



Especificaciones de diseño para la validación de campo

Variaciones aceptables para la ubicación del icono

Hay innumerables casos únicos en los que los usuarios necesitan ser informados sobre los errores de validación. Teniendo en cuenta el tipo de control y la configuración de la interfaz de usuario, elija la ubicación del icono adecuada a su situación.



Variaciones aceptables para ubicaciones de icono de validación de campo

Validación que requiere un viaje de ida y vuelta a un servidor o conexión de red

En algunos casos, se requiere un viaje de ida y vuelta al servidor para verificar el contenido y sería importante mostrar el progreso del usuario, verificado y los estados de error. La figura siguiente muestra un ejemplo de este caso y la interfaz de usuario recomendada.

Default

User account:

In progress

User account:  Verifying...

Verified

User account: 

Error - Validation

User account: 

Invalid user account

Error - Couldn't connect to server

User account:  [Retry](#)

Error verifying user account
An unexpected error was encountered while attempting to contact the server.

Validación que implica un viaje de ida y vuelta a un servidor

Tenga en cuenta que se debe proporcionar un espacio disponible adecuado a la derecha del control para dar cabida a la "Verificación..." y texto "Reintentar".

Texto de advertencia in situ

Cuando hay espacio disponible para colocar el mensaje de error cerca del control en un estado de error, esto es preferible a usar la información sobre herramientas solo.

TextBox:

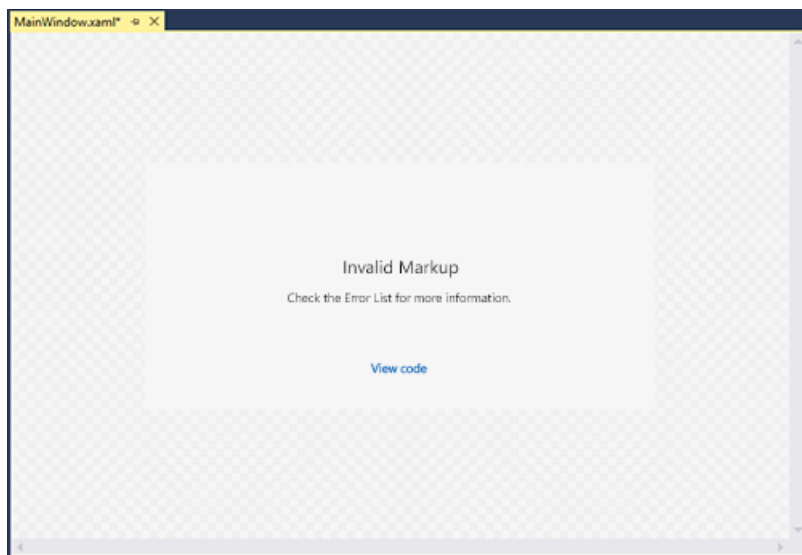
 Required fields must be completed

ComboBox:

Texto de advertencia in situ

Marcas de agua

A veces, un control o ventana completo está en un estado de error. En esta situación, utilice una marca de agua para indicar el error.



Validación del campo de marca de agua

Animaciones para Visual Studio

06/04/2020 • 21 minutes to read • [Edit Online](#)

Fundamentos de la animación

Prácticas recomendadas de animación en Visual Studio

Siga estas reglas para garantizar estilos de animación coherentes y fáciles de usar en el IDE de Visual Studio.

- **Sé selectivo.** Limite las animaciones a aquellas que sirvan a propósitos específicos.
- **El tiempo y la velocidad son importantes** para garantizar que las transiciones se sientan rápidas y naturales:
 - Completa transiciones animadas en medio segundo (500 milisegundos).
 - Las animaciones que se producirían con frecuencia deben ser lo suficientemente rápidas como para que no interrumpan el flujo de trabajo del usuario. Mira la animación en un bucle y ajusta el tiempo hasta que se sienta bien.
 - Las animaciones no deben ser tan rápidas o inquietantes que es difícil de entender, pero no tan lentas que hace que un impaciente para la transición termine.
 - Utilice el tiempo variable para enfatizar la importancia. Por ejemplo, al navegar por una secuencia de elementos en un diagrama de clases, acelere las transiciones entre elementos y, a continuación, reduzca la velocidad para centrarse en los elementos importantes.
- Utilice la **flexión no lineal gradual** de un estado a otro, dando una sensación de calma y movimiento natural.
- Cuando sea posible, **utilice una animación sutil al pasar el ratón** para indicar elementos interactivos debajo del ratón.
- Si depende en gran medida de las animaciones de las entidades, **proporcione un medio para desactivarlas** localmente (para todas las entidades) como una opción en el cuadro de diálogo Herramientas > opciones.
- **Solo debe producirse una animación** a la vez y transmitir solo una información. Más de un objeto que se mueve o intenta transmitir varias cosas puede ser confuso.
- **La sutileza es importante.** En la mayoría de los casos, la animación no tiene que exigir la atención del usuario para cumplir su propósito. Los cambios sutiles en el tiempo, la secuenciación y el comportamiento pueden afectar significativamente a la percepción y pueden marcar la diferencia entre una animación eficaz e ineficaz.
- Cuando utilice animación para llamar la atención sobre algo, asegúrese de que vale la pena interrumpir el tren de pensamiento del **usuario**.
- **Al mostrar el progreso o el estado** a través de la animación:
 - Deje de mostrar el movimiento de progreso cuando el proceso subyacente no avance.
 - Distinguir los procesos indeterminados de los procesos determinados.
 - Asegúrese de que una animación tiene estados identificables de finalización y error.
 - Minimice el uso de animaciones de efectos que muestren el estado y asegúrese de que tienen valor

real proporcionando información adicional de uso real. Algunos ejemplos son los cambios de estado transitorios y las emergencias

Animación no:

- No utilice movimientos pequeños (movimiento en una huella pequeña). Prefiere fundidos y cambios sobre objetos en movimiento.
- No utilice animaciones que tienen lugar en una gran área de bienes raíces de pantalla. Independientemente del tamaño, este estilo de animación distrae al usuario.
- No utilice animaciones no relacionadas con el objeto en el que el usuario está centrado o con el que interactúa.
- No utilice animaciones que requieran la interacción del usuario para restablecer el estado, como forzar al usuario a responder a una notificación parpadeante para que deje de parpadear. Interactuar con ellos de cualquier manera debería ser suficiente para descartarlos.

Para obtener más información sobre las aplicaciones para estas [prácticas recomendadas](#), consulte Patrones de animación .

Métricas de animación

- El sistema debe reaccionar visiblemente a los gestos del usuario en menos de 10 milisegundos.
- Las transiciones animadas no deben tardar más de 500 milisegundos en completarse.
- Una forma de compensar las transiciones que requieren tiempos más largos es separarlas en dos partes. Por ejemplo, la primera parte de una animación podría ser el contenedor de contenido vacío (hasta 500 milisegundos), seguido del contenido que se desvanece en el contenedor (hasta 500 milisegundos).
- Para los tiempos de carga que se pueden calcular, se prefiere un indicador de progreso determinante (indicador de progreso porcentual hecho).
- Para los tiempos de carga que no se pueden calcular, un indicador ocupado como un cursor o una animación de giro incrustada (indicador de carga o de trabajo) es adecuado.

Animación como comunicador

En la interfaz de usuario de Visual Studio, la animación solo funciona como una herramienta de comunicación. Se usa para comunicar una variedad de información, como cambios estructurales en la interfaz de usuario (por ejemplo, cuando se abre o se cierra un menú). La animación puede ayudar a visualizar el comportamiento dependiente del tiempo de sistemas complejos, como la visualización del progreso de la instalación. Las animaciones también se pueden utilizar para atraer la atención con alertas y notificaciones.

Las animaciones de la interfaz de usuario suelen funcionar de cuatro maneras: visualizar, atraer la atención, simular y los indicadores de tiempos de respuesta/progreso.

Visualizar

La animación puede enfatizar la naturaleza tridimensional de los objetos y facilitar a los usuarios la visualización de su estructura espacial. Para lograr esto, la animación puede necesitar girar el objeto en un círculo completo, girarlo lentamente hacia adelante y hacia atrás, o acercar el objeto y aumentar ligeramente su tamaño para enfatizar el rollover o el enfoque.

Aunque los objetos tridimensionales se pueden mover con el control de usuario, el diseñador debe determinar de antemano (mediante programación o manualmente) cómo animar mejor un movimiento que proporciona una comprensión óptima del objeto. Esta animación programada puede ser activada por el usuario colocando el cursor sobre el objeto, mientras que los movimientos controlados por el usuario requieren que el usuario entienda cómo manipular el objeto. Limitar el movimiento a un solo eje u orientación a la vez; escala, gira o traduce, pero no haga más de uno simultáneamente.

La categoría Visualizar incluye los aspectos de los datos, las relaciones, el estado, la estructura, la secuencia y el tiempo.

data

Ilustrar información compleja y variable:

- Moverse a través de visualizaciones de información como gráficos y gráficos
- Paso a través de una secuencia, visita guiada y paginación
- Llamar a los detalles, señalar y resaltar información específica
- Superposición de detalles e información adicional sobre un elemento enfocado
- Morphing de una representación estructural o organizativa a otra
- Representación de los cambios a lo largo del tiempo mediante controles deslizantes de tiempo, ruedas jog-and-shuttle y controles de transporte (reproducir, detener y pausar)

Relaciones

- Ilustrar cómo se relacionan entre sí los elementos o qué elementos se relacionan con un elemento determinado
- Mostrar jerarquías y relaciones padre-hijo o hermano
- Un elemento genera otro
- Un elemento se minimiza a otro elemento
- Un elemento atado a otro

State

- Actualizaciones de contenido
- Enfoque y selección del usuario
- Progreso
- Errors

Estructura

- Pivotar la estructura en un nodo
- Reorientar
- Minimizar y maximizar, o expandir y contraer

Secuencia

- Secuencia de presentación de diapositivas
- Voltar a través de las imágenes

Time

- Mostrar cambio con el tiempo, lapso de tiempo y screencast
- Muévete a la basura, rehacer y rehacer
- Restaurar estado histórico

Atraer la atención

Si el objetivo es atraer la atención del usuario a un solo elemento de varios o alertar al usuario de información actualizada, entonces una animación podría ser adecuada. Por ejemplo, la página de inicio de la aplicación podría emplear un botón de introducción que se desliza en su lugar después de que se cargue la página.

Como regla general, el último elemento en movimiento en la pantalla atrae la atención del usuario. En una serie de

elementos animados, la atención del usuario seguirá el último objeto en movimiento.

Alerta

- Alertar al usuario, llamar la atención, mostrar el progreso
- Mostrar que algo se está haciendo correctamente o incorrectamente o mostrar cambios de progreso o progreso
- Preguntar a los usuarios durante una tarea, como encontrar más información en línea o aprender sobre la tarea actual

Notificaciones

- Alertar al usuario sobre una condición de error
- Interrumpir al usuario para ver si le gustaría atender a otra cosa
- Informar suavemente al usuario de que un proceso se ha completado o cambiado, como cuando se completa una descarga.

Simular

Esta categoría abarca la física y la dimensionalidad.

- Ilustrar de dónde vienen los objetos o a dónde van
- Expandir y contraer o abrir y cerrar
- Panorámica, desplazamiento y vueltas de página
- Apilamiento y ordenamiento en z
- Carrusel y acordeón
- Volver y rotar la interfaz de usuario

Indicadores de respuesta y progreso

Los indicadores de progreso tienen un par de ventajas notables:

- Los indicadores de progreso determinados e indeterminados aseguran al usuario que el sistema no se ha bloqueado y está trabajando en el problema.
- Los indicadores determinados dan al usuario una idea de lo lejos que avanza la acción, así como una sensación de acercarse al final.

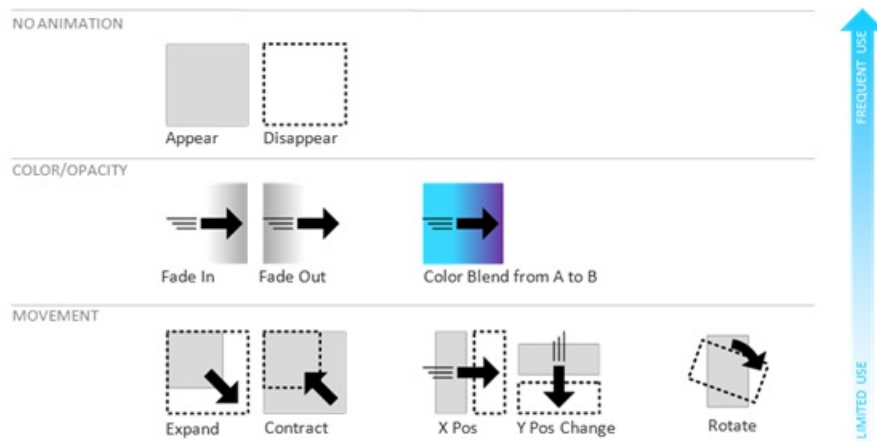
Patrones de animación

Información general

Las animaciones en Visual Studio están diseñadas para servir a una función específica sin obstaculizar la productividad del usuario. Por lo general, las animaciones en Visual Studio deben ser:

- Pequeño y discreto
- Natural y realista
- Sutil y tenue
- Rápido y eficiente
- Relajado, no apresurado

En esta ilustración se muestran los estilos de animación que se recomiendan para Visual Studio. Ninguna animación o animaciones sutiles como fade in/fade out son las más utilizadas. Hay una aplicación limitada de animaciones de movimiento como expandir y contraer, cambio de posición X e Y, y rotación.



Estilos de animación recomendados para Visual Studio

Aparecen y desaparecen

Con este patrón, un elemento cambia de visible a fuera de vista y hacia atrás sin una animación de transición.



Aparece y desaparece la animación

Uso correcto

Elementos de interfaz de usuario frescos que necesitan aparecer o desaparecer al instante para que el usuario no esté distraído ni obstruido. Además, las animaciones de movimiento lento pueden percibirse como un arrastre de rendimiento, lo que no se producirá con el estilo de aparecer y desaparecer.

Uso incorrecto

Casos en los que la interfaz de usuario aparece tan abruptamente el usuario no tiene idea de lo que sucedió, y agregar una animación ayudaría con la comprensión contextual.

Propiedades de animación

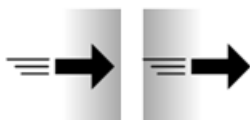
El retardo de tiempo es generalmente cero segundos.

Ejemplos

- Ocultar automáticamente las ventanas de herramientas
- Interfaz de usuario del editor activado por teclado, como IntelliSense y la Ayuda de parámetros
- Expandir y contraer regiones de código

Fade-in y fade-out

Con este patrón, un elemento de interfaz de usuario pasa de no visible (0% de opacidad) a visible (100% de opacidad) o viceversa.



Animación fade-in y fade-out

Uso correcto

Esta es la animación de interfaz de usuario más comúnmente recomendada. Es un efecto sutil que añade interés sin interrumpir el flujo. En algunos casos, es posible que el usuario ni siquiera se dé cuenta de que hay una animación, al percibir un sistema de interfaz de usuario fluido y fluido.

Propiedades de animación

- Opacidad inicial: 0% para fade-in, 100% para fade-out
- Opacidad final: 100% para fade-in, 0% para fade-out

- Duración: 200 milisegundos independientes, 100 milisegundos cuando se utiliza como parte de una secuencia de animación de combinación
- Estilo de adelanto: Sine InOut

Ejemplos

- Ocultar automáticamente las ventanas de herramientas
- Menú abierto y cerrado
- Transiciones de pestañas en primer plano y en segundo plano

Mezcla de colores de A a B

Con este patrón, un elemento de interfaz de usuario cambia de color A a color B.



Animación de mezcla de colores

Uso correcto

Como transición animada cuando un elemento de interfaz de usuario cambia de color de un contexto o estado a otro.

Propiedades de animación

- Color inicial: específico de la interfaz de usuario
- Color final: específico de la interfaz de usuario
- Duración: 200 milisegundos independientes, 100 milisegundos cuando se utiliza como parte de una secuencia de animación de combinación
- Estilo de adelanto: Sine InOut

Ejemplos

- Transiciones de estado de la ventana de documento (activa, última activa e inactiva)
- Transiciones de estado de la ventana de herramientas (enfocadas y desenfocadas)

Expandir y contrar

Con este patrón, un elemento de interfaz de usuario se expande en las direcciones X, Y o ambas direcciones.



Expandir y contraer animación

Uso correcto

Como transición animada cuando un elemento de interfaz de usuario cambia de tamaño de un contexto a otro.

Propiedades de animación

- Escala X: % o dimensión específica (en píxeles)
- Escala Y: % o dimensión específica (en píxeles)
- Posición de anclaje: generalmente superior izquierda (para idiomas de izquierda a derecha) o superior derecha (para idiomas de derecha a izquierda)
- Duración: 200 milisegundos independientes, 100 milisegundos cuando se utiliza como parte de una secuencia de animación de combinación

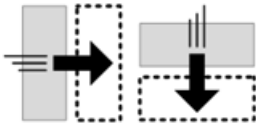
Ejemplos

- El panel Explorador de arquitectura se expande y contrae

- Visual Studio 2017 Elemento de página de inicio expandir y contraer

Cambio de posición X-Y

Con este patrón, un elemento de interfaz de usuario cambia su posición X o Y o ambas.



Animación de cambio de posición X-Y

Uso correcto

Como transición animada cuando un elemento de interfaz de usuario cambia de posición de un contexto a otro.

Propiedades de animación

- Inicio de la posición X e Y: específico de la interfaz de usuario
- Posición De terminación X e Y: específica de la interfaz de usuario
- Ruta de movimiento: ninguno
- Duración: 200 milisegundos independientes, 100 milisegundos cuando se utiliza como parte de una secuencia de animación de combinación
- Estilo de adelanto: Sine InOut

Ejemplo

Reordenación de pestañas

Girar

Con este patrón, el elemento de interfaz de usuario gira.



Animación de rotación de elementos de interfaz de usuario

Uso correcto

Sólo para el indicador de progreso de giro indeterminado.

Propiedades de animación

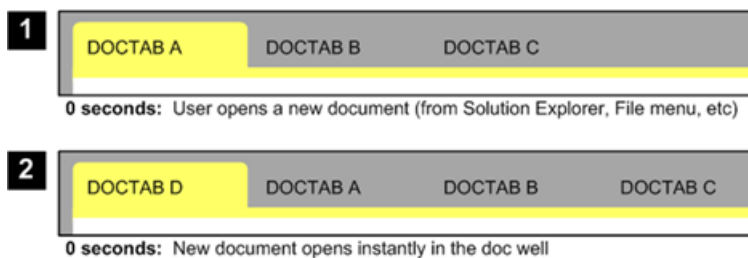
- Grado de rotación: 360
- Centro de rotación: centro del objeto
- Duración: continua

Ejemplo

Indicador de progreso indeterminado (giro)

Acciones comunes de la interfaz de usuario del shell y animaciones recomendadas

Pestaña abierta

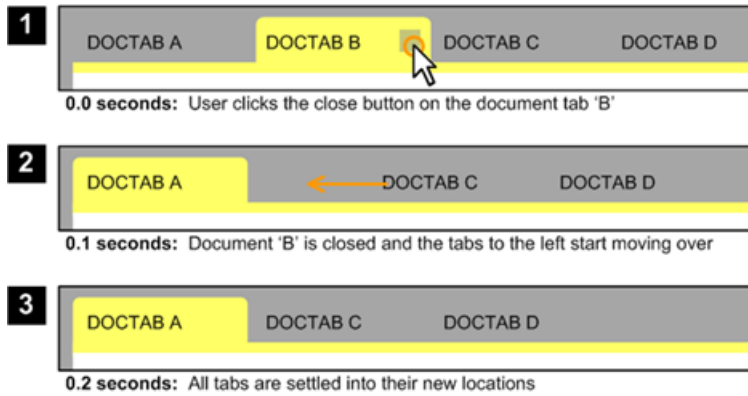


Animación abierta de pestañas

- Estilo: aparece

- Duración: cero segundos

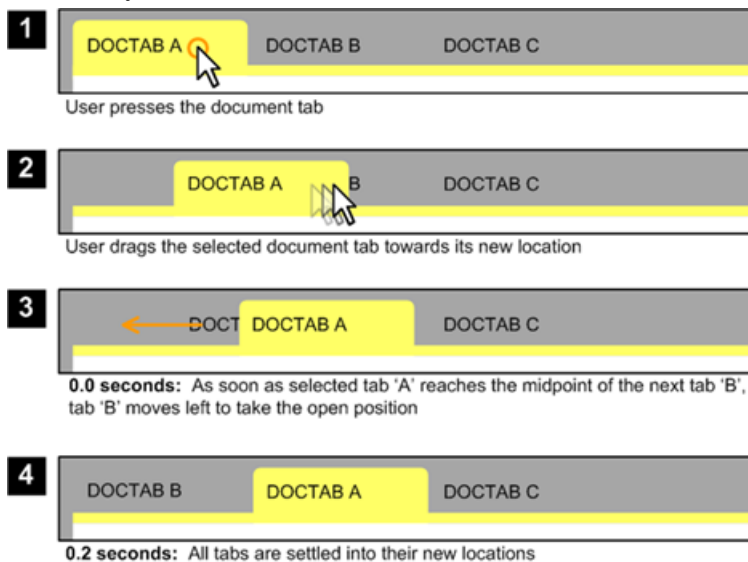
Cierre de pestañas



Animación de cierre de pestaña

- Estilo: Cambio de posición X
- Duración: 200 milisegundos

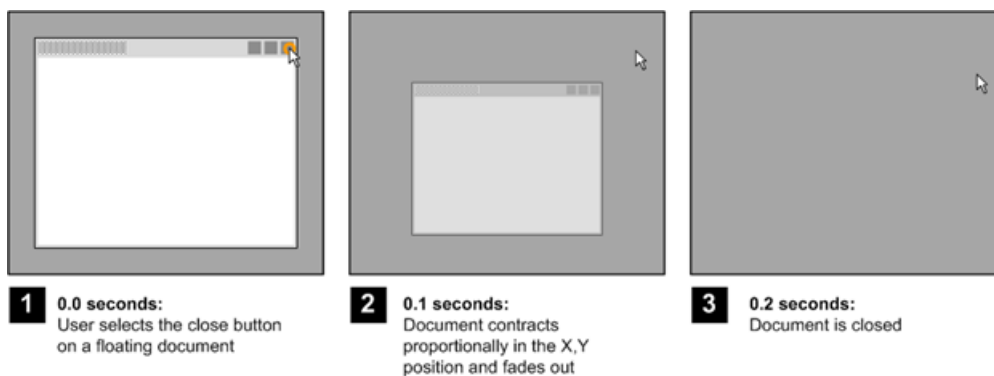
Reorden de pestañas



Animación de reordenación de pestañas

- Estilo: Cambio de posición X
- Duración: 200 milisegundos

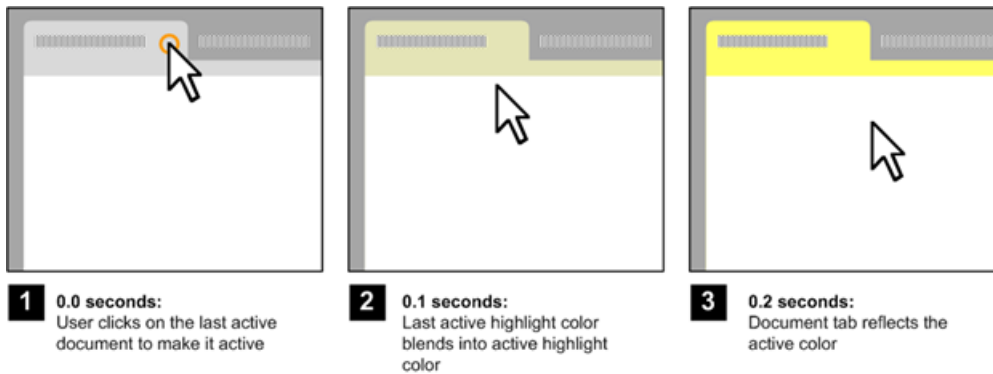
Cerrar documento flotante



Cerrar animación de documento flotante

- Estilo: aparece
- Duración: 200 milisegundos

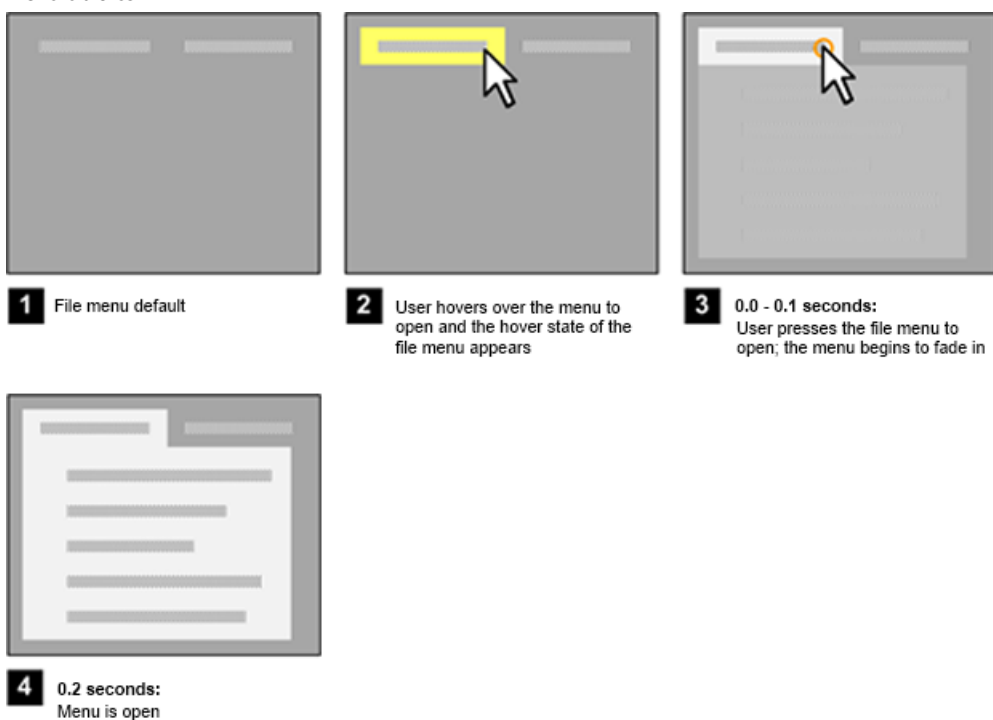
Transición del estado de la ventana



Animación de transición de estado de ventana

- Estilo: para ser coherente con otras ventanas, deje que el sistema operativo actual defina la animación de cierre del documento.
- Duración: 200 milisegundos

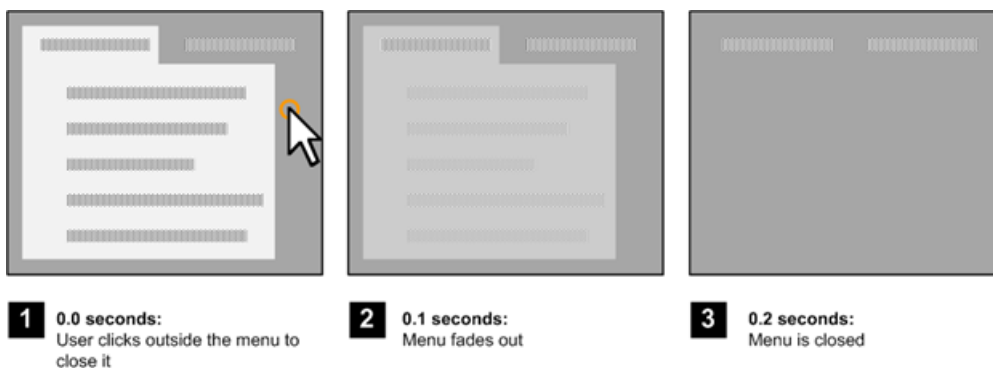
Menú abierto



Menú de animación abierta

- Estilo: fade-in
- Duración: 200 milisegundos

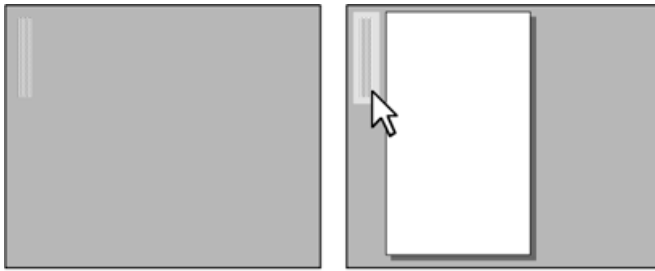
Menú cerrado



Menú cerrar animación

- Estilo: fade-out
- Duración: 200 milisegundos

La ventana de herramientas de ocultación automática revela



1 Default state of autohidden tool window tab

2 0.0 seconds:
User hovers over the tab and the toolwindow appears immediately

La ventana de herramientas de ocultación automática revela la animación

- Estilo: aparece
- Duración: cero segundos

Herramientas de evaluación para Visual Studio

06/04/2020 • 17 minutes to read • [Edit Online](#)

Lista de verificación de artesanía para Visual Studio

Utilice esta lista de comprobación para evaluar la calidad de la experiencia del usuario para obtener detalles visuales y de interacción.

Información general

- Compruebe que todos los comandos dan como resultado comentarios que indican a los usuarios que sus comandos se han llevado a cabo.
- Compruebe que todos los elementos y controles de la interfaz de usuario están visibles en todos los temas y en el modo de contraste alto.
- Compruebe que la selección inactiva y activa siempre esté diferenciada, tanto en el modo estándar como en el modo de contraste alto.
- Compruebe que el enfoque siempre es visible y aparente.

Rendimiento

- Verifique que se muestre algún tipo de indicador "ocupado" si un comando tarda más de un segundo en completarse.
- Compruebe que si un comando tarda más de 10 segundos en completarse, se muestra una barra de progreso explícita, ya sea determinada (preferida) o indeterminada.

Texto de la interfaz de usuario

- Compruebe que todas las etiquetas son mayúsculas y minúsculas y que no hay texto en minúsculas.

	CORRECTO	INCORRECTO
Texto de comando (todos)	Caso de sentencia: Nombre del directorio:	Nombre del directorio:
Texto del botón (cliente)	Caso de título: [Establecer como predeterminado]	ESTABLECER COMO PREDETERMINADO
Texto del botón (en línea)	Caso de sentencia: [Establecer como predeterminado]	

- Compruebe que todas las etiquetas, excepto los encabezados y botones de grupo, terminan con dos puntos y preceden al control con el que están emparejados.
- Compruebe que los botones, comandos y vínculos de comandos que inician la interfaz de usuario para capturar la entrada del usuario terminan en puntos suspensivos [...].

Ejemplos:

- Un botón [**Avanzado...**] en un cuadro de diálogo.
- Las opciones de comando en el menú Herramientas (**Herramientas > Opciones**) no deben obtener puntos suspensivos, porque iniciar el propio cuadro de diálogo es la intención del comando.
- Compruebe que la interfaz de usuario no contiene abreviaturas, excepto los términos estándar del sector. Por ejemplo, ni HTML ni TCP/IP deben deletrearse, aunque OOM (fuera de memoria) y PII (información de identificación personal) deben ser deletreados.

Acceso mediante el teclado

- Compruebe que hay una manera de realizar cada tarea con el teclado. Generalmente esto se logra a través del acceso al teclado para cada control, pero para algunas áreas altamente visuales, una solución alternativa como ir a la vista de código es aceptable.
- Compruebe que puede desplazarse por los controles en un orden lógico (de izquierda a derecha y de arriba a abajo). Aunque esta es una práctica recomendada para la mayoría de los controles, no todos los controles requieren este enfoque. Por ejemplo, compruebe que los controles de botón de opción están en un grupo con una sola tabulación.
- Compruebe que todos los controles tienen etiquetas y que cada etiqueta tiene un mnemotécnico (las excepciones incluyen algunos controles no etiquetados que podrían seguir un control etiquetado en la pestaña).
- Verifique que no haya conflictos mnemotécnicos.

Fuentes

- Compruebe que todas las fuentes (cara, tamaño, color) se utilizan de forma coherente y mantenga la jerarquía.
- Compruebe que todos los elementos de la interfaz de usuario utilizan el servicio de fuentes de entorno. (Consulte [Fuentes y formato para Visual Studio](#))

Para comprobar si se está utilizando el servicio, vaya a **Herramientas > Opciones > Fuentes y colores**. En el menú desplegable Configuración, elija Fuente de entorno y cambie la cara de fuente a algo estilísticamente diferente (como Harrington o Comic Sans) y establezca el tamaño en 12 pt. A continuación, haga clic en Aceptar. Es posible que deba reiniciar el IDE, pero la mayoría de la interfaz de usuario cambiará inmediatamente. Las áreas que no detectan el cambio de fuente incluso al reiniciar no utilizan la fuente de entorno.

- Compruebe que las fuentes derivadas del servicio (por ejemplo, texto en negrita o ampliado) conservan su tamaño y formato en relación con el texto "normal" cuando se cambia el tamaño de fuente del entorno.
- Compruebe que no hay errores de recorte debido a fuentes ampliadas. Las fuentes que se recortan son probablemente el resultado de controles de altura fija o contenedores de altura fija.

Cuadros de diálogo

- Compruebe que el título del cuadro de diálogo es el mismo que el comando que lo inició.
- Compruebe que todos los controles estándar son coherentes con el sistema operativo: el color de fondo es estándar y que ningún control debe tener un estilo especial re-plantillado que los haga parecer diferentes de los controles estándar.
- Compruebe que los márgenes del formulario deben ser de 12 píxeles y deben aparecer uniformes y coherentes.
- Compruebe que los cuadros de diálogo aparecen centrados en el shell IDE o en la ventana que los generó.
- Cuando es útil, los cuadros de diálogo deben ser redimensionables. Para los cuadros de diálogo que son

redimensionables, compruebe que al cambiar el tamaño, los controles adecuados deben cambiar el tamaño mientras que otras partes del cuadro de diálogo permanecen constantes.

- Compruebe que los cuadros de diálogo redimensionables conservan cualquier tamaño ajustado por el usuario (tamaño, ubicación, expansión de los controles de cuadro de diálogo, etc.).
- Compruebe que no hay ningún icono en la barra de título.
- Compruebe que no hay botones Minimizar y Maximizar en la barra de título.

Botones de operación de cuadro de diálogo (solo cliente VS)

- Compruebe que los botones de operación están en este orden: **Aceptar**, **Cancelar**, **Aplicar**.
- Compruebe que los botones **Aceptar** y **Cancelar** tienen el tamaño estándar: 75x23 píxeles.
- Compruebe que los botones **Aceptar** y **Cancelar** tienen el mismo tamaño, independientemente de la longitud de la cadena.
- Si la etiqueta de un botón de operación requiere que el botón sea más ancho que el estándar, compruebe que el botón **Cancelar** correspondiente es del mismo tamaño.
- Compruebe que hay un relleno de 6 píxeles entre los botones y los controles asociados.
- Compruebe que los botones **Aceptar** y **Cancelar** no tienen mnemotécnicas (claves de acceso definidas por una letra subrayada).
- Compruebe que un botón (normalmente **Aceptar**) tiene el foco de forma predeterminada.
- Compruebe que **Esc** cancela el cuadro de diálogo
- Compruebe que **Enter** ejecuta el botón predeterminado si el foco no está en un control que procesa **Enter**.
- Compruebe que los botones **Aceptar** y **Cancelar** están situados en la esquina inferior derecha del cuadro de diálogo. En raras excepciones, es aceptable que se apilen verticalmente en la parte superior derecha.
- Compruebe que la configuración vertical solo se utiliza si otros botones están en una alineación horizontal dentro del cuadro de diálogo.

Normas de control

General

- Compruebe que, cuando sea posible, hay buenos valores predeterminados para acelerar la interacción del usuario y dirigir a los usuarios hacia un resultado seguro o común.
- Compruebe que los controles estándar se comportan de la misma manera para que los usuarios sepan lo que sucederá en función de la experiencia anterior.

Controles de etiquetas

- Compruebe que cada control tiene una etiqueta y que cada etiqueta está emparejada visualmente con su control (generalmente dentro de un intervalo de 4-6 píxeles) y está más cerca de su control correspondiente que de otros controles.
- Compruebe que las etiquetas se colocan al ras a la izquierda con el borde izquierdo del control si se colocan encima y se centran horizontalmente, de modo que la línea base de la etiqueta se alinee con la línea base del texto de entrada si se coloca a la izquierda.
- Compruebe que si varios controles de entrada y etiqueta apilados se colocan a la izquierda de un control, las etiquetas se vacían a la izquierda y a la misma distancia del borde del cuadro de diálogo, nunca se vacían a la derecha y a la misma distancia de los controles. Los pares deben distribuirse uniformemente a menos que necesiten espaciado adicional para indicar la agrupación.

Controles de entrada (cuadros de texto y cuadros combinados)

- Compruebe que, al utilizar la fuente de entorno predeterminada, la altura de visualización de los cuadros de texto, cuadros combinados y botones es de 23 píxeles.
- Si se utiliza texto de sugerencia, `Environment.ControlEditHintText` compruebe que el color está configurado para usar el servicio de color.
- Si el campo es un campo obligatorio que se debe identificar como tal, verifique:
 - que el fondo `Environment.ControlEditRequiredBackground` se establece en y el primer plano se establece en `Environment.ControlEditRequiredHintText`
 - que hay texto de sugerencia dentro del control que aparece como **<"> requerido"**

Controles de botón

- Compruebe que los botones tienen un tamaño mínimo de 75x23 píxeles, a menos que se acomode texto más largo.
- Compruebe que los botones tienen márgenes izquierdo y derecho de 3-5 píxeles, así como relleno para el contenido.
- Es aceptable utilizar un pequeño botón cuadrado con sólo un punto suspensivo [...] en él en lugar de un botón [Examinar...] (o funcionalidad similar). Si se utiliza, compruebe que el botón tiene un tamaño de 23x23.
- Si hay más de un botón [Examinar...] en un cuadro de diálogo, compruebe que la versión abreviada (solo **ellipsis** [...]) se utiliza para todos.
- Compruebe que los botones de puntos suspensivos [...] no tienen un mnemotécnico. Cuando el foco está en el control de entrada junto a él, una pestaña debe mover el foco al botón de puntos suspensivos.
- Compruebe que los botones, comandos y vínculos de comandos que inician la interfaz de usuario secundaria que captura más entradas de usuario deben terminar en puntos suspensivos [...].

Hipervínculos

- Compruebe que un control de hipervínculo nunca parpadea en rojo cuando está activo. Este es un indicador de que el servicio de color no se está utilizando
- Verifique que los colores VS usados sean:
 - `Environment.ControlLinkText`
 - `Environment.ControlLinkTextHover`
 - `Environment.ControlLinkTextPressed`
- Compruebe que los hipervínculos aparecen en azul sin subrayado a menos que esté incrustado en un párrafo.

Casillas

- Si una casilla de verificación tiene texto de varias líneas, compruebe que la casilla se alinea con la primera línea de texto, no centrada verticalmente en todas las líneas.
- Compruebe que las casillas de verificación siempre indican una opción binaria y no navegue por el usuario ni abra nuevas ventanas o páginas.
- Si una casilla de verificación presenta una opción relacionada con un control de entrada, compruebe que se coloca al ras a la izquierda y muy cerca bajo ese control para indicar su relación.
- Compruebe que **nunca** se utilice una casilla de verificación como medio para habilitar todo el contenido de un cuadro de diálogo o página.

Cajas de grupo

- Compruebe que un cuadro de diálogo no contiene un único cuadro de grupo que contenga todo el contenido del cuadro de diálogo.
- Compruebe que hay al menos dos controles dentro de cada cuadro de grupo.
- Rara vez debe haber más de dos cuadros de grupo en un cuadro de diálogo.
- Compruebe que no hay cuadros de grupo anidados.

Iconos

- Compruebe que los iconos aparecen correctamente invertidos cuando está en el tema oscuro.
- Compruebe que todos los iconos se basan en conceptos básicos.
- Compruebe que cada icono es distinto, fácil de reconocer y no contiene más de dos conceptos (sin modificador de estado/idioma).
- Compruebe que el icono base aparece centrado en el espacio.
- Compruebe que todos los iconos aparecen legibles en el modo de contraste alto.
- Compruebe que cualquier color utilizado se alinee con los estándares de uso de color.
- Compruebe que no haya halos (bordes) alrededor de los iconos. (Si está presente, el halo debe coincidir con el color de fondo de la interfaz de usuario adyacente).

Interfaz de usuario táctil

- Compruebe que los controles interactivos son lo suficientemente grandes como para ser fácilmente táctiles - mínimo **23x23 píxeles** de tamaño
- Compruebe que los controles utilizados con más frecuencia tienen un tamaño mínimo de **40 x 40 píxeles**.
- Compruebe que los controles interactivos tengan al menos **5 píxeles de espaciado** entre ellos

Compatibilidad de reconocimiento por monitor para extensores de Visual Studio

02/12/2019 • 27 minutes to read • [Edit Online](#)

Las versiones anteriores a Visual Studio 2019 tenían su contexto de reconocimiento de PPP establecido en compatible con el sistema, en lugar de con el reconocimiento de PPP por monitor (PMA). La ejecución en el reconocimiento del sistema provocó una experiencia visual degradada (por ejemplo, fuentes borrosas o iconos) siempre que Visual Studio tuviera que representarse en monitores con factores de escala diferentes o en equipos remotos con diferentes configuraciones de pantalla (por ejemplo, diferentes Escalado de Windows).

El contexto de reconocimiento de PPP de Visual Studio 2019 se establece como PMA, cuando el entorno lo admite, lo que permite que Visual Studio se represente según la configuración de la pantalla en la que se hospeda, en lugar de una única configuración definida por el sistema. Trasladar en última instancia a una interfaz de usuario siempre nítida para áreas de superficie que admiten el modo PMA.

Consulte la documentación sobre el [desarrollo de aplicaciones de escritorio de PPP alta en Windows](#) para obtener más información acerca de los términos y el escenario general descritos en este documento.

Guía de inicio rápido

- Asegúrese de que Visual Studio se está ejecutando en modo PMA (consulte [habilitación de PMA](#))
- Validar que la extensión funciona correctamente en un conjunto de escenarios comunes (consulte [comprobación de las extensiones para problemas de PMA](#))
- Si encuentra problemas, puede usar las estrategias/recomendaciones descritas en este documento para diagnosticar y corregir dichos problemas. También necesitará agregar el nuevo paquete de NuGet [Microsoft.VisualStudio.DpiAwareness](#) al proyecto para tener acceso a las API necesarias.

Habilitar PMA

Para habilitar PMA en Visual Studio, deben cumplirse los siguientes requisitos:

- Actualización 2018 de abril de Windows 10 (v1803, RS4) o posterior
- .NET Framework 4,8 RTM o superior
- Visual Studio 2019 con la opción "[optimizar representación para pantallas con densidades de píxeles diferentes](#)" habilitada

Una vez cumplidos estos requisitos, Visual Studio habilita automáticamente el modo PMA en todo el proceso.

NOTE

Windows Forms contenido de Visual Studio (por ejemplo, el explorador de propiedades) solo es compatible con PMA si tiene Visual Studio 2019 versión 16,1 o posterior.

Prueba de las extensiones para problemas de PMA

Visual Studio es compatible oficialmente con los marcos de interfaz de usuario de WPF, Windows Forms, Win32 y HTML/JS. Cuando Visual Studio se coloca en el modo PMA, cada pila de la interfaz de usuario se comporta de forma diferente. Por lo tanto, independientemente del marco de interfaz de usuario, se recomienda que se realice

un paso de prueba para asegurarse de que toda la interfaz de usuario sea compatible con el modo PMA.

Se recomienda validar los siguientes escenarios comunes:

- Cambiar el factor de escala de un entorno de monitor único mientras la aplicación se está ejecutando.

Este escenario ayuda a probar que la interfaz de usuario responde al cambio dinámico de PPP de Windows.

- Acoplar y desacoplar un equipo portátil en el que un monitor conectado esté establecido en el servidor principal y el monitor conectado tenga un factor de escala diferente del portátil mientras se ejecuta la aplicación.

Este escenario ayuda a probar que la interfaz de usuario responde al cambio de PPP de la pantalla, así como al control de las pantallas que se agregan o se quitan dinámicamente.

- Tener varios monitores con diferentes factores de escala y mover la aplicación entre ellos.

Este escenario ayuda a probar que la interfaz de usuario responde al cambio de PPP de la pantalla.

- Comunicación remota en un equipo cuando los equipos locales y remotos tienen factores de escala diferentes para el monitor principal.

Este escenario ayuda a probar que la interfaz de usuario responde al cambio dinámico de PPP de Windows.

Una buena prueba preliminar de si la interfaz de usuario puede tener problemas es si el código utiliza las clases *Microsoft. VisualStudio. Utilities. dpi. DpiHelper*, *Microsoft. VisualStudio. PlatformUI. DpiHelper* o *VsUI::CDpiHelper*. Estas clases *DpiHelper* antiguas solo admiten el reconocimiento de PPP del sistema y no siempre funcionarán correctamente cuando el proceso sea PMA.

El uso típico de estos *DpiHelpers* tendrá el siguiente aspecto:

```
Point screenTopRight = logicalBounds.TopRight.LogicalToDeviceUnits();

POINT screenIntTopRight = new POINT
{
    x = (int)screenTopRight.X,
    y = (int)screenTopRight.Y
}

// Declared via P/Invoke
IntPtr monitor = MonitorFromPoint(screenIntTopRight, MONITOR_DEFAULTTONEAREST);
```

En el ejemplo anterior, un rectángulo que representa los límites lógicos de una ventana se convierte en unidades de dispositivo para que se pueda pasar al método nativo *MonitorFromPoint* que espera las coordenadas del dispositivo para devolver un puntero de monitor preciso.

Clases de problemas

Cuando el modo PMA está habilitado para Visual Studio, la interfaz de usuario podría replicar problemas de varias maneras comunes. La mayoría de estos problemas, si no todos, pueden producirse en cualquiera de los marcos de interfaz de usuario compatibles con Visual Studio. Además, estos problemas también pueden producirse cuando una parte de la interfaz de usuario se hospeda en escenarios de escala de PPP en modo mixto (consulte la [documentación](#) de Windows para obtener más información).

Creación de ventanas Win32

Al crear ventanas con *CreateWindow ()* o *CreateWindowEx ()*, un patrón común es crear la ventana en coordenadas (0,0) (la esquina superior izquierda de la pantalla principal) y, a continuación, moverla a su posición final. Sin embargo, si lo hace, la ventana desencadenará un evento o mensaje de cambio de PPP, que puede redesencadenar otros eventos o mensajes de la interfaz de usuario y, finalmente, provocar un comportamiento o una representación no deseados.

Colocación de elementos de WPF

Al mover elementos de WPF con las antiguas Microsoft. VisualStudio. Utilities. ppp. DpiHelper, las coordenadas superior izquierda podrían no calcularse correctamente cuando los elementos están en un valor de PPP no primario.

Serialización de los tamaños o posiciones de los elementos de la interfaz de usuario

Cuando el tamaño o la posición de la interfaz de usuario (si se guarda como unidades de dispositivo) se restaura en un contexto de PPP diferente del que se almacenó, se colocará y se dimensionará incorrectamente. Esto sucede porque las unidades de dispositivo tienen una relación PPP inherente.

Escala incorrecta

Los elementos de la interfaz de usuario creados en los PPP primarios se escalan correctamente; sin embargo, cuando se mueven a una pantalla con un valor de PPP diferente, no cambian de escala y su contenido es demasiado grande o demasiado pequeño.

Límite incorrecto

Del mismo modo que el problema de escalado, los elementos de la interfaz de usuario calculan sus límites correctamente en su contexto de PPP primario, sin embargo, cuando se mueven a un valor de PPP no primario, no calcularán correctamente los nuevos límites. Como tal, la ventana de contenido es demasiado pequeña o demasiado grande en comparación con la interfaz de usuario de hospedaje, lo que produce un espacio vacío o un recorte.

Arrastrar & colocar

Siempre que estén dentro de los escenarios de PPP en modo mixto (por ejemplo, distintos elementos de la interfaz de usuario en modos de reconocimiento de PPP diferentes), las coordenadas de arrastrar y colocar podrían ser erróneas, lo que da lugar a que la posición de colocación final sea incorrecta.

Interfaz de usuario fuera de proceso

Algunas interfaces de usuario se crean fuera de proceso y, si el proceso externo de creación está en un modo de reconocimiento de PPP diferente que Visual Studio, esto puede presentar cualquiera de los problemas de representación anteriores.

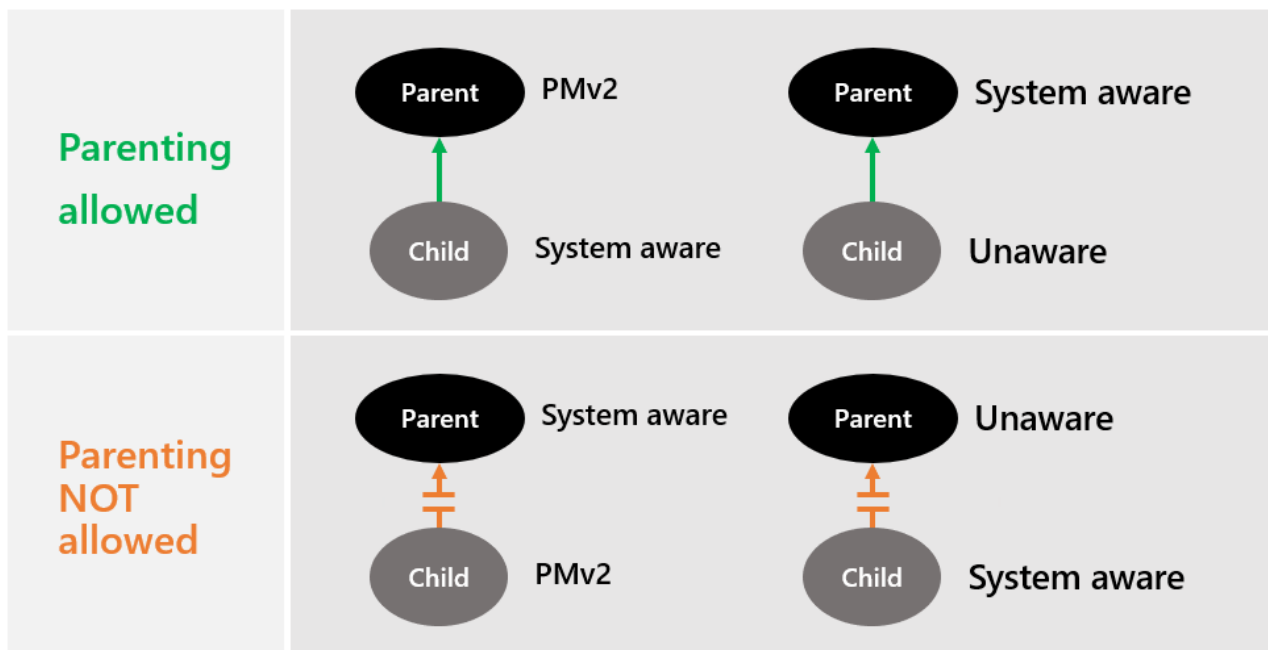
Windows Forms los controles, las imágenes o los diseños se representan incorrectamente

No todo el contenido de la Windows Forms admite el modo PMA. Como resultado, puede ver el problema de representación con diseños o escalas incorrectos. En este caso, una posible solución consiste en representar explícitamente Windows Forms contenido en "DpiAwarenessContext de sistema" (consulte [forzar un control en un DpiAwarenessContext específico](#)).

Windows Forms controles o ventanas que no se muestran

Una de las principales causas de este problema es que los desarrolladores intentan volver a controlar un control o una ventana con un DpiAwarenessContext en una ventana con un DpiAwarenessContext diferente.

En las siguientes imágenes se muestran las restricciones actuales del sistema operativo Windows en las ventanas parentables:



NOTE

Puede cambiar este comportamiento estableciendo el comportamiento de hospedaje de subprocesos (consulte la [enumeración `Dpi\_Hosting\_Behavior`](#)).

Como resultado, si establece una relación de elementos primarios y secundarios entre los modos no admitidos, se producirá un error y es posible que el control o la ventana no se representen según lo esperado.

Diagnóstico de problemas

Hay muchos factores que se deben tener en cuenta a la hora de identificar problemas relacionados con PMA:

- ¿La interfaz de usuario o la API esperan valores lógicos o de dispositivo?
 - La interfaz de usuario y las API de WPF suelen usar valores lógicos (pero no siempre)
 - La interfaz de usuario y las API de Win32 suelen usar valores de dispositivo
- ¿De dónde proceden los valores?
 - Si recibe valores de otra interfaz de usuario o de una API, está pasando valores lógicos o de dispositivo.
 - Si recibe valores de varios orígenes, ¿todos ellos usan/esperan los mismos tipos de valores o se debe mezclar y hacer coincidir las conversiones?
- ¿Se usan constantes de interfaz de usuario y en qué formulario están?
- ¿Es el subproceso en el contexto de PPP correcto para los valores que está recibiendo?

Los cambios para habilitar el hospedaje de PPP mixto deben poner normalmente las rutas de acceso de código en el contexto derecho; sin embargo, el trabajo realizado fuera del bucle de mensajes principal o el flujo de eventos podría ejecutarse en un contexto de PPP incorrecto.

- ¿Los valores son límites de contexto entre PPP?

Arrastrar & Drop es una situación común en la que las coordenadas pueden cruzar contextos de PPP. La ventana intenta hacer lo correcto, pero en algunos casos, es posible que la interfaz de usuario del host tenga que realizar el trabajo de conversión para asegurarse de que coincidan los límites del contexto.

Paquete NuGet de PMA

Las nuevas bibliotecas de `DpiAwareness` se pueden encontrar en el paquete NuGet [Microsoft.VisualStudio.DpiAwareness](#).

Herramientas recomendadas

Las herramientas siguientes pueden ayudarle a depurar problemas relacionados con el PMA en algunas de las distintas pilas de interfaz de usuario compatibles con Visual Studio.

Espionaje

Snoop es una herramienta de depuración de XAML que tiene alguna funcionalidad adicional que las herramientas de XAML de Visual Studio integradas no tienen. Además, Snoop no necesita depurar activamente Visual Studio para poder ver y retocar su interfaz de usuario de WPF. Las dos formas principales de Snoop pueden ser útiles para diagnosticar problemas de PMA es para validar coordenadas de ubicación lógicas o límites de tamaño, y para validar la interfaz de usuario tiene los PPP correctos.

Herramientas XAML de Visual Studio

Como Snoop, las herramientas de XAML en Visual Studio pueden ayudar a diagnosticar problemas de PMA. Una vez que se encuentra un posible culpable, puede establecer puntos de interrupción y usar la ventana del árbol visual activo, así como las ventanas de depuración, para inspeccionar los límites de la interfaz de usuario y los PPP actuales.

Estrategias para corregir problemas de PMA

Reemplazar llamadas DpiHelper

En la mayoría de los casos, la corrección de problemas de la interfaz de usuario en el modo PMA reduce el reemplazo de llamadas en código administrado a las clases antiguas *Microsoft. VisualStudio. Utilities. ppp. DpiHelper* y *Microsoft. VisualStudio. PlatformUI. DpiHelper*, con llamadas a la nueva *Clase auxiliar Microsoft. VisualStudio. Utilities. DpiAwareness*.

```
// Remove this kind of use:
Point deviceTopLeft = new Point(window.Left, window.Top).LogicalToDeviceUnits();

// Replace with this use:
Point deviceTopLeft = window.LogicalToDevicePoint(new Point(window.Left, window.Top));
```

En el caso de código nativo, se implica el reemplazo de llamadas a la clase antigua *VsUI:: CDpiHelper* con llamadas a la nueva clase *VsUI:: CDpiAwareness*.

```
// Remove this kind of use:
int cx = VsUI::DpiHelper::LogicalToDeviceUnitsX(m_cxS);
int cy = VsUI::DpiHelper::LogicalToDeviceUnitsY(m_cyS);

// Replace with this use:
int cx = m_cxS;
int cy = m_cyS;
VsUI::CDpiAwareness::LogicalToDeviceUnitsX(m_hwnd, &cx);
VsUI::CDpiAwareness::LogicalToDeviceUnitsY(m_hwnd, &cy);
```

Las nuevas clases *DpiAwareness* y *CDpiAwareness* ofrecen las mismas aplicaciones auxiliares de conversión de unidades que las clases *DpiHelper* pero requieren un parámetro de entrada adicional: el elemento de la interfaz de usuario que se va a usar como referencia para la operación de conversión. Es importante tener en cuenta que las aplicaciones auxiliares de escalado de imágenes no existen en las nuevas aplicaciones auxiliares de *DpiAwareness/CDpiAwareness* y, si es necesario, se debe usar la [ImageService](#) en su lugar.

La clase *DpiAwareness* administrada proporciona aplicaciones auxiliares para objetos visuales de WPF, controles de Windows Forms y HWND de Win32 y HMONITORS (ambos en forma de *IntPtrs*), mientras que la clase de *CDpiAwareness* nativa ofrece aplicaciones auxiliares HWND y HMONITOR.

Windows Forms cuadros de diálogo, ventanas o controles que se muestran en el DpiAwarenessContext incorrecto

Incluso después de una protección correcta de Windows con diferentes `DpiAwarenessContexts` (debido a un comportamiento predeterminado de Windows), los usuarios pueden seguir viendo problemas de escalado como ventanas con diferentes escalas de `DpiAwarenessContexts` de forma diferente. Como resultado, es posible que los usuarios vean problemas de alineación/texto borroso o imagen en la interfaz de usuario.

La solución consiste en establecer el ámbito de `DpiAwarenessContext` correcto para todas las ventanas y controles de la aplicación.

Cuadros de diálogo de modo mixto de nivel superior (TLMM)

Al crear ventanas de nivel superior como cuadros de diálogo modales, es importante asegurarse de que el subproceso está en el estado correcto antes de que se cree la ventana (y su identificador). El subproceso se puede incluir en el reconocimiento del sistema mediante el uso de la aplicación auxiliar `CDpiScope` en la aplicación auxiliar nativa o `DpiAwareness`. `EnterDpiScope` en `Managed`. (TLMM generalmente debe usarse en cuadros de diálogo o ventanas que no son de WPF).

Modo mixto de nivel secundario (CLMM)

De forma predeterminada, las ventanas secundarias reciben el contexto de reconocimiento de PPP del subproceso actual si se crean sin un elemento primario o el contexto de reconocimiento de PPP del elemento primario cuando se crean con un elemento primario. Para crear un elemento secundario con un contexto de reconocimiento de PPP distinto al de su elemento primario, el subproceso se puede colocar en el contexto de reconocimiento de PPP deseado. Después, el elemento secundario se puede crear sin un elemento primario y cambiar manualmente de primario a la ventana primaria.

Problemas de CLMM

La mayor parte del trabajo de cálculo de la interfaz de usuario que se produce como parte del bucle de mensajería principal o de la cadena de eventos debe estar ya en ejecución en el contexto de reconocimiento de PPP adecuado. Sin embargo, si los cálculos de coordenadas o de ajuste de tamaño se realizan fuera de estos flujos de trabajo principales (por ejemplo, durante una tarea de tiempo de inactividad o fuera del subproceso de la interfaz de usuario, el contexto de reconocimiento de PPP actual podría ser incorrecto, lo que provocaría un error de la interfaz de usuario o problemas de cambio de tamaño. En general, poner el subproceso en el estado correcto para el trabajo de la interfaz de usuario corrige el problema.

No participar en CLMM

Si una ventana de herramientas que no es de WPF se está migrando a una compatibilidad total con el PMA, deberá optar por no participar en CLMM. Para ello, es necesario implementar una nueva interfaz: `IVsDpiAware`.

```
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IVsDpiAware
{
    [ComAliasName("Microsoft.VisualStudio.Shell.Interop.VSDPIMode")]
    uint Mode {get;}
}
```

```
IVsDpiAware : public IUnknown
{
    public:
        HRESULT STDMETHODCALLTYPE get_Mode(__RPC__out VSDPIMODE *dwMode);
};
```

En el caso de los lenguajes administrados, el mejor lugar para implementar esta interfaz está en la misma clase que deriva de *Microsoft.VisualStudio.Shell.ToolWindowPane*. Para C++, el mejor lugar para implementar esta interfaz está en la misma clase que implementa *IVsWindowPane* desde `vsshell.h`.

El valor devuelto por la propiedad `Mode` en la interfaz es `__VSDPIMODE` (y se convierte en un valor `uint` en `Managed`):

```
enum __VSDPIMODE
{
    VSDM_Unaware    = 0x01,
    VSDM_System     = 0x02,
    VSDM_PerMonitor = 0x03,
}
```

- No compatible significa que la ventana de herramientas debe controlar 96 PPP, Windows controlará el escalado para el resto de PPP. La causa de que el contenido sea ligeramente borroso.
- Sistema significa que la ventana de herramientas debe controlar el PPP del PPP de la pantalla principal. Cualquier presentación con un PPP coincidente tendrá un aspecto nítido, pero si el PPP es diferente o cambia durante la sesión, Windows controlará el escalado y será ligeramente borroso.
- Permonitor significa que la ventana de herramientas debe administrar todos los PPP en todas las pantallas y cada vez que cambie el PPP.

NOTE

Visual Studio solo admite el reconocimiento de PerMonitorV2, por lo que el valor de enumeración de permonitor se traduce en el valor de Windows de DPI_AWARENESS_CONTEXT_PER_MONITOR_AWARE_V2.

Forzar un control en un DpiAwarenessContext específico

La interfaz de usuario heredada que no se está actualizando para admitir el modo PMA puede seguir necesitando pequeños ajustes para trabajar mientras Visual Studio se ejecuta en modo PMA. Una de estas correcciones implica asegurarse de que la interfaz de usuario se crea en el DpiAwarenessContext derecho. Para forzar la interfaz de usuario en un DpiAwarenessContext determinado, puede especificar un ámbito de PPP con el código siguiente:

```
using (DpiAwareness.EnterDpiScope(DpiAwarenessContext.SystemAware))
{
    Form form = new MyForm();
    form.ShowDialog();
}
```

```
void MyClass::ShowDialog()
{
    VsUI::CDpiScope dpiScope(DPI_AWARENESS_CONTEXT_SYSTEM_AWARE);
    HWND hwnd = ::CreateWindow(...);
}
```

NOTE

Forzar el DpiAwarenessContext solo funciona en los cuadros de diálogo que no son de WPF y de nivel superior. Al crear una interfaz de usuario de WPF que se hospedará dentro de las ventanas de herramientas o los diseñadores, en cuanto el contenido se inserte en el árbol de la interfaz de usuario de WPF, se convertirá en el proceso actual DpiAwarenessContext.

Problemas conocidos

Windows Forms

Para optimizar los nuevos escenarios en modo mixto, Windows Forms ha cambiado la forma en que crea controles y ventanas cuando su elemento primario no se estableció explícitamente. Anteriormente, los controles sin un elemento primario explícito usaban una "ventana de estacionamiento" interna como un elemento primario temporal en el control o la ventana que se va a crear.

Antes de .NET 4,8, había una única "ventana de estacionamiento" que obtiene su `DpiAwarenessContext` del contexto de reconocimiento de PPP del subproceso actual en el momento de creación de la ventana. Cualquier control no primario hereda el mismo `DpiAwarenessContext` que la ventana de estacionamiento cuando se crea el identificador del control y el desarrollador de la aplicación puede cambiar el elemento primario al elemento primario esperado. Esto produciría errores basados en el tiempo si la "ventana de estacionamiento" tenía un `DpiAwarenessContext` superior al de la ventana primaria final.

A partir de .NET 4,8, ahora hay una "ventana de aparcamiento" para cada `DpiAwarenessContext` que se ha encontrado. La otra diferencia principal es que el `DpiAwarenessContext` que se usa para el control se almacena en caché cuando se crea el control, no cuando se crea el identificador. Esto significa que el comportamiento de final general es el mismo, pero puede convertir lo que solía ser un problema basado en el tiempo en un problema coherente. También proporciona al desarrollador de la aplicación un comportamiento más determinista para escribir el código de la interfaz de usuario y el ámbito correcto.

Abordar problemas de PPP

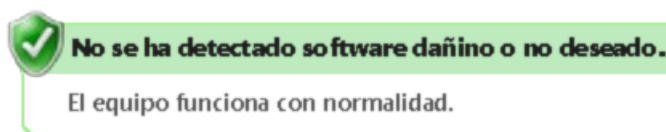
06/04/2020 • 24 minutes to read • [Edit Online](#)

Un número cada vez mayor de dispositivos se envían con pantallas de "alta resolución". Estas pantallas suelen tener más de 200 píxeles por pulgada (ppi). Trabajar con una aplicación en estos equipos requerirá que el contenido se escale verticalmente para satisfacer las necesidades de ver el contenido a una distancia de visualización normal para el dispositivo. A partir de 2014, el objetivo principal para las pantallas de alta densidad son los dispositivos informáticos móviles (tabletas, computadoras portátiles y teléfonos).

Windows 8.1 y superior contiene varias características para permitir que estas máquinas funcionen con pantallas y entornos donde la máquina está conectada a pantallas de alta densidad y densidad estándar al mismo tiempo.

- Windows puede permitirle escalar contenido al dispositivo mediante la opción "Hacer texto y otros elementos más grandes o más pequeños" (disponible desde Windows XP).
- Windows 8.1 y versiones posteriores escalarán automáticamente el contenido de la mayoría de las aplicaciones cuando se muevan entre pantallas de densidades de píxeles diferentes. Cuando la pantalla principal es de alta densidad (200% de escala) y la pantalla secundaria es de densidad estándar (100%), Windows reducirá automáticamente el contenido de la ventana de la aplicación en la pantalla secundaria (1 píxel mostrado por cada 4 píxeles representados por la aplicación).
- Windows establecerá de forma predeterminada la escala correcta para la densidad de píxeles y la distancia de visualización de la pantalla (Windows 7 y versiones posteriores, configurables por EL OEM).
- Windows puede escalar automáticamente contenido hasta un 250 % en dispositivos nuevos que superen los 280 ppp (a partir de Windows 8.1 S14).

Windows tiene una manera de tratar con el escalado vertical de la interfaz de usuario para aprovechar el aumento del recuento de píxeles. Una aplicación opta por este sistema declarándose "consciente del sistema DPI." Las aplicaciones que no lo hacen son escaladas verticalmente por el sistema. Esto puede dar lugar a una experiencia de usuario "difusa" en la que toda la aplicación está uniformemente estirada en píxeles. Por ejemplo:



Visual Studio opta por ser compatible con el escalado de PPP y, por lo tanto, no está "virtualizado".

Windows (y Visual Studio) aprovechan varias tecnologías de interfaz de usuario, que tienen diferentes formas de tratar con los factores de escalado establecidos por el sistema. Por ejemplo:

- WPF/WPF mide los controles de forma independiente del dispositivo (unidades, no píxeles). WPF/WPF UI escala automáticamente para el PPP actual.
- Todos los tamaños de texto, independientemente del marco de la interfaz de usuario, se expresan en puntos, por lo que el sistema trata como independiente de PPP. El texto de Win32, WinForms y WPF ya se escala correctamente cuando se dibuja en el dispositivo de visualización.
- Los cuadros de diálogo y ventanas de Win32/WinForms tienen medios para habilitar el diseño que cambia de tamaño con texto (por ejemplo, a través de paneles de cuadrícula, flujo y diseño de tabla). Esto permite evitar ubicaciones de píxeles codificadas de forma rígida que no se escalan cuando se aumentan los tamaños de fuente.

- Los iconos proporcionados por el sistema o los recursos basados en métricas del sistema (por ejemplo, SM_CXICON y SM_CXSMICON) ya se han ampliado.

Win32 más antiguo (GDI, GDI+) y la interfaz de usuario basada en WinForms

Aunque WPF/WPF ya es compatible con PPP, gran parte de nuestro código basado en Win32/GDI no se escribió originalmente teniendo en cuenta el conocimiento de PPP. Windows ha proporcionado API de escalado de PPP. Las correcciones para los problemas de Win32 deben usarlos de forma coherente en todo el producto. Visual Studio ha proporcionado una biblioteca de clases auxiliares para evitar duplicar la funcionalidad y garantizar la coherencia en todo el producto.

Imágenes de alta resolución

Esta sección es principalmente para los desarrolladores que extienden Visual Studio 2013. Para Visual Studio 2015, use el servicio de imágenes integrado en Visual Studio. También puede encontrar que necesita admitir o apuntar a muchas versiones de Visual Studio y, por lo tanto, el uso del servicio de imágenes en 2015 no es una opción, ya que no existe en versiones anteriores. Esta sección también es para ti entonces.

Escalar imágenes demasiado pequeñas

Las imágenes que son demasiado pequeñas se pueden escalar verticalmente y representarse en GDI y WPF mediante algunos métodos comunes. Las clases auxiliares de PPP administrados están disponibles para los integradores internos y externos de Visual Studio para abordar iconos de escalado, mapas de bits, imágenes e listas de imágenes. Los c/C++ helpers nativos basados en Win32 están disponibles para escalar HICON, HBITMAP, HIMAGELIST y VsUI::GdiplusImage. El escalado de un mapa de bits normalmente solo requiere un cambio de una línea después de incluir una referencia a la biblioteca auxiliar. Por ejemplo:

```
(Unmanaged) VsUI::DpiHelper::LogicalToDeviceUnits(&hBitmap);
```

```
(WinForms) DpiHelper.LogicalToDeviceUnits(ref image);
```

El escalado de una lista de imágenes depende de si la lista de imágenes se ha completado en tiempo de carga o se anexa en tiempo de ejecución. Si se completa en `LogicalToDeviceUnits()` tiempo de carga, llame con la lista de imágenes como lo haría con un mapa de bits. Cuando el código necesite cargar un mapa de bits individual antes de componer la lista de imágenes, asegúrese de escalar el tamaño de imagen de la lista de imágenes:

```
imagelist.ImageSize = DpiHelper.LogicalToDeviceUnits(imagelist.ImageSize);
```

En el código nativo, las dimensiones se pueden escalar al crear la lista de imágenes de la siguiente manera:

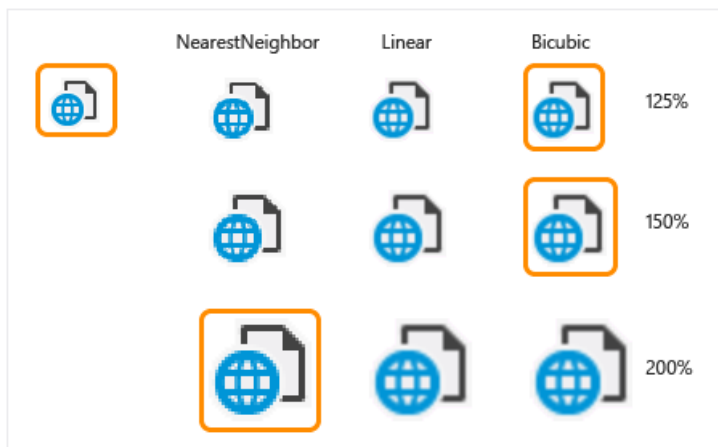
```
ImageList_Create(VsUI::DpiHelper::LogicalToDeviceUnitsX(16), VsUI::DpiHelper::LogicalToDeviceUnitsY(16),  
ILC_COLOR32|ILC_MASK, nCount, 1);
```

Las funciones de la biblioteca permiten especificar el algoritmo de cambio de tamaño. Al escalar las imágenes que se van a colocar en las listas de imágenes, asegúrese de especificar el color de fondo que se utiliza para la transparencia, o utilice la escala `NearestNeighbor` (que provocará distorsiones al 125% y 150%).

Consulte [DpiHelper](#) la documentación en MSDN.

En la tabla siguiente se muestran ejemplos de cómo se deben escalar las imágenes en los factores de escala de PPP

correspondientes. Las imágenes descritas en naranja denotan nuestra práctica recomendada a partir de Visual Studio 2013 (escalado de 100%-200% PPP):



Problemas de diseño

Los problemas de diseño comunes se pueden evitar principalmente manteniendo los puntos en la interfaz de usuario escalados y relativos entre sí en lugar de mediante el uso de ubicaciones absolutas (específicamente, en unidades de píxeles). Por ejemplo:

- Las posiciones de diseño/texto deben ajustarse para tener en cuenta las imágenes escaladas.
- Las columnas de las cuadrículas deben tener anchos ajustados para el texto escalado vertical.
- Los tamaños codificados de forma rígida o el espacio entre los elementos también tendrán que ampliarse. Los tamaños que se basan solo en dimensiones de texto suelen estar bien, ya que las fuentes se escalan automáticamente.

Las funciones auxiliares están disponibles en la [DpiHelper](#) clase para permitir el escalado en los ejes X e Y:

- LogicalToDeviceUnitsX/LogicalToDeviceUnitsY (las funciones permiten escalar en el eje X/Y)
- espacio int á DpiHelper.LogicalToDeviceUnitsX (10);
- int height á VsUI::DpiHelper::LogicalToDeviceUnitsY(5);

Hay sobrecargas LogicalToDeviceUnits para permitir el escalado de objetos como Rect, Point y Size.

Uso de la biblioteca/clase DPIHelper para escalar imágenes y diseño

La biblioteca auxiliar de PPP de Visual Studio está disponible en formularios nativos y administrados y otras aplicaciones pueden usarla fuera del shell de Visual Studio.

Para usar la biblioteca, vaya a los ejemplos de [extensibilidad de VSSDK](#) de Visual Studio y clone el ejemplo de alta DPI_Images_Icons.

En los archivos de origen, incluya *VsUIDpiHelper.h* y llame a las funciones estáticas de `VsUI::DpiHelper` la clase:

```
#include "VsUIDpiHelper.h"

int cxScaled = VsUI::DpiHelper::LogicalToDeviceUnitsX(cx);
VsUI::DpiHelper::LogicalToDeviceUnits(&hBitmap);
```


NOTE

No utilice las funciones auxiliares en variables estáticas de nivel de módulo o de clase. La biblioteca también utiliza estáticas para la sincronización de subprocesos y es posible que se encuentre con problemas de inicialización de pedidos. Convierta esas estáticas en variables miembro no estáticas o envuélvalas en una función (para que se construyan en el primer acceso).

Para tener acceso a las funciones auxiliares de PPP desde código administrado que se ejecutará dentro del entorno de Visual Studio:

- El proyecto de consumo debe hacer referencia a la versión más reciente de Shell MPF. Por ejemplo:

```
<Reference Include="Microsoft.VisualStudio.Shell.14.0.dll" />
```

- Asegúrese de que el proyecto tiene referencias a **System.Windows.Forms**, **PresentationCore** y **PresentationUI**.
- En el código, use el **Microsoft.VisualStudio.PlatformUI** espacio de nombres y llame a las funciones estáticas de **DpiHelper** clase. Para los tipos admitidos (puntos, tamaños, rectángulos, etc.), se proporcionan funciones de extensión que devuelven nuevos objetos escalados. Por ejemplo:

```
using Microsoft.VisualStudio.PlatformUI;
double x = DpiHelper.LogicalToDeviceUnitsX(posX);
Point ptScaled = ptOriginal.LogicalToDeviceUnits();
DpiHelper.LogicalToDeviceUnits(ref bitmap);
```

Tratar con el difuso de imágenes de WPF en la interfaz de usuario ampliable

En WPFWPF, WPFWPF cambia el tamaño automáticamente de los mapas de bits para el nivel de zoom de PPP actual mediante un algoritmo bicúbico de alta calidad (predeterminado), que funciona bien para imágenes o capturas de pantalla grandes, pero es inapropiado para los iconos de elementos de menú porque introduce el difusor percibido.

Recomendaciones:

- Para la ilustración de **BitmapScalingMode** imágenes de logotipos y banners, se podría utilizar el modo de cambio de tamaño predeterminado.
- Para los elementos de menú **BitmapScalingMode** y las imágenes de iconografía, se debe utilizar cuando no causa que otros artefactos de distorsión eliminen el mareo (al 200% y al 300%).
- Para niveles de zoom grandes no múltiplos del 100% (por ejemplo, 250% o 350%), escalado de imágenes de iconografía con resultados bicúbicos en una interfaz de usuario difusa y lavada. Un mejor resultado se obtiene escalando primero la imagen con **NearestNeighbor** al múltiplo más grande del 100% (por ejemplo, 200% o 300%) y escalando con bicúbico desde allí. Consulte Caso especial: preescalado de imágenes de WPF para grandes niveles de PPP para obtener más información.

El **DpiHelper** clase en el **Microsoft.VisualStudio.PlatformUI** espacio de nombres proporciona un miembro **BitmapScalingMode** que se puede usar para el enlace. Permitirá que el shell de Visual Studio controle el modo de escalado de mapa de bits en todo el producto de forma uniforme, en función del factor de escalado de PPP.

Para usarlo en XAML, agrega:

```
xmlns:vsui="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Shell.14.0"

<Setter Property="RenderOptions.BitmapScalingMode" Value="{x:Static vs:DpiHelper.BitmapScalingMode}" />
```

El shell de Visual Studio ya establece esta propiedad en ventanas y cuadros de diálogo de nivel superior. La interfaz de usuario basada en WPF que se ejecuta en Visual Studio ya la heredará. Si la configuración no se propaga a los elementos concretos de la interfaz de usuario, se puede establecer en el elemento raíz de la interfaz de usuario XAML/WPF. Los lugares donde esto sucede incluyen ventanas emergentes, en elementos con padres de Win32 y ventanas de diseño que se ejecutan fuera de proceso, como Blend.

Algunas interfaces de usuario pueden escalar independientemente del nivel de zoom de PPP del conjunto de sistemas, como el editor de texto de Visual Studio y los diseñadores basados en WPF (WPF Desktop y Windows Store). En estos casos, `DpiHelper.BitmapScalingMode` no debe utilizarse. Para corregir este problema en el editor, el equipo IDE creó una propiedad personalizada denominada `RenderOptions.BitmapScalingMode`. Establezca ese valor de propiedad en `HighQuality` o `NearestNeighbor` en función del nivel de zoom combinado del sistema y la interfaz de usuario.

Caso especial: preescalado de imágenes WPF para grandes niveles de PPP

Para niveles de zoom muy grandes que no son múltiplos del 100% (por ejemplo, 250%, 350%, etc.), escalar imágenes de iconografía con resultados bicúbicos en una interfaz de usuario difusa y lavada. La impresión de estas imágenes junto con texto nítido es casi como la de una ilusión óptica. Las imágenes parecen estar más cerca del ojo y desenfocadas en relación con el texto. El resultado de la escala en este tamaño aumentado se puede mejorar primero escalando la imagen con `NearestNeighbor` al múltiplo más grande del 100% (por ejemplo, 200% o 300%) y escalando con bicúbico al resto (un 50% adicional).

A continuación se muestra un ejemplo de las diferencias en los resultados, donde la primera imagen se escala con el algoritmo de doble escala mejorado 100%->200%->250%, y la segunda solo con bicúbico 100%->250%.



Para permitir que la interfaz de usuario use este doble escalado, el marcado XAML para mostrar cada `Image` elemento tendrá que modificarse. En los ejemplos siguientes se muestra cómo usar el doble escalado en WPF/WPF en Visual Studio mediante la biblioteca `DpiHelper` y `Shell.12/14`.

Paso 1: Preescale la imagen al 200%, 300%, y así sucesivamente usando `NearestNeighbor`.

Preajuste la escala de la imagen mediante un convertidor aplicado en un enlace o con una extensión de marcado XAML. Por ejemplo:

```
<vsui:DpiPrescaleImageSourceConverter x:Key="DpiPrescaleImageSourceConverter" />

<Image Source="{Binding Path=SelectedImage, Converter={StaticResource DpiPrescaleImageSourceConverter}}"
Width="16" Height="16" />

<Image Source="{vsui:DpiPrescaledImage Images/Help.png}" Width="16" Height="16" />
```

Si la imagen también necesita ser temática (la mayoría, si no todos, debería), el marcado puede usar un convertidor diferente que primero realiza el tema de la imagen y, a continuación, el escalado previo. El marcado puede `DpiPrescaleThemedImageConverter` `DpiPrescaleThemedImageSourceConverter` usar o , dependiendo de la

salida de conversión deseada.

```
<vsui:DpiPrescaleThemedImageSourceConverter x:Key="DpiPrescaleThemedImageSourceConverter" />

<Image Width="16" Height="16">
  <Image.Source>
    <MultiBinding Converter="{StaticResource DpiPrescaleThemedImageSourceConverter}">
      <Binding Path="Icon" />
      <Binding Path="(vsui:ImageThemingUtilities.ImageBackgroundColor)"
        RelativeSource="{RelativeSource Self}" />
      <Binding Source="{x:Static vsui:Boxes.BooleanTrue}" />
    </MultiBinding>
  </Image.Source>
</Image>
```

Paso 2: Asegúrese de que el tamaño final es correcto para el DPI actual.

Dado que WPFWPF escalará la interfaz de usuario para el PPP actual mediante el `BitmapScalingMode` propiedad establecida en el `UIElement`, un `Image` control con una imagen preescalado como su origen se verá dos o tres veces mayor de lo que debería. Las siguientes son un par de maneras de contrarrestar este efecto:

- Si conoce la dimensión de la imagen original al 100%, puede especificar el tamaño exacto del control `Image`. Estos tamaños reflejarán el tamaño de la interfaz de usuario antes de aplicar el escalado.

```
<Image Source="{Binding Path=SelectedImage, Converter={StaticResource
DpiPrescaleImageSourceConverter}}" Width="16" Height="16" />
```

- Si no se conoce el tamaño de la imagen original, un `LayoutTransform` se puede utilizar para reducir la escala del objeto `Image` final. Por ejemplo:

```
<Image Source="{Binding Path=SelectedImage, Converter={StaticResource
DpiPrescaleImageSourceConverter}}" >
  <Image.LayoutTransform>
    <ScaleTransform
      ScaleX="{x:Static vsui:DpiHelper.PreScaledImageLayoutTransformScale}"
      ScaleY="{x:Static vsui:DpiHelper.PreScaledImageLayoutTransformScale}" />
  </Image.LayoutTransform>
</Image>
```

Habilitación de la compatibilidad con HDPI en el WebOC

De forma predeterminada, los controles WebOC (como el control `WebBrowser` en WPFWPF o la interfaz `IWebBrowser2`) no habilitan la detección y compatibilidad con HDPI. El resultado será un control incrustado con contenido de pantalla que es demasiado pequeño en una pantalla de alta resolución. A continuación se describe cómo habilitar la compatibilidad con PPP altos en una instancia WebOC específica.

Implemente la interfaz `IDocHostUIHandler` (consulte el artículo de MSDN en [el IDocHostUIHandler](#):

```
[ComImport, InterfaceType(ComInterfaceType.InterfaceIsIUnknown),
Guid("BD3F23C0-D43E-11CF-893B-00AA00BDCE1A")]
public interface IDocHostUIHandler
{
    [return: MarshalAs(UnmanagedType.I4)]
    [PreserveSig]
    int ShowContextMenu(
        [In, MarshalAs(UnmanagedType.U4)] int dwID,
        [In] POINT pt,
        [In, MarshalAs(UnmanagedType.Interface)] object pcmdtReserved,
        [In, MarshalAs(UnmanagedType.IDispatch)] object pdispReserved);
```

```

[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int GetHostInfo([In, Out] DOCHOSTUIINFO info);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int ShowUI(
    [In, MarshalAs(UnmanagedType.I4)] int dwID,
    [In, MarshalAs(UnmanagedType.Interface)] object activeObject,
    [In, MarshalAs(UnmanagedType.Interface)] object commandTarget,
    [In, MarshalAs(UnmanagedType.Interface)] object frame,
    [In, MarshalAs(UnmanagedType.Interface)] object doc);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int HideUI();
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int UpdateUI();
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int EnableModeless([In, MarshalAs(UnmanagedType.Bool)] bool fEnable);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int OnDocWindowActivate([In, MarshalAs(UnmanagedType.Bool)] bool fActivate);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int OnFrameWindowActivate([In, MarshalAs(UnmanagedType.Bool)] bool fActivate);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int ResizeBorder(
    [In] COMRECT rect,
    [In, MarshalAs(UnmanagedType.Interface)] object doc,
    bool fFrameWindow);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int TranslateAccelerator(
    [In] ref MSG msg,
    [In] ref Guid group,
    [In, MarshalAs(UnmanagedType.I4)] int nCmdID);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int GetOptionKeyPath(
    [Out, MarshalAs(UnmanagedType.LPArray)] string[] pbstrKey,
    [In, MarshalAs(UnmanagedType.U4)] int dw);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int GetDropTarget(
    [In, MarshalAs(UnmanagedType.Interface)] IOleDropTarget pDropTarget,
    [MarshalAs(UnmanagedType.Interface)] out IOleDropTarget ppDropTarget);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int GetExternal([MarshalAs(UnmanagedType.IDispatch)] out object ppDispatch);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int TranslateUrl(
    [In, MarshalAs(UnmanagedType.U4)] int dwTranslate,
    [In, MarshalAs(UnmanagedType.LPWStr)] string strURLIn,
    [MarshalAs(UnmanagedType.LPWStr)] out string pstrURLOut);
[return: MarshalAs(UnmanagedType.I4)]
[PreserveSig]
int FilterDataObject(
    IDataObject pDO,
    out IDataObject ppDORet);
}

```

Opcionalmente, implemente la interfaz ICustomDoc (consulte el artículo de MSDN en [el ICustomDoc](#):

```
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown),
 Guid("3050F3F0-98B5-11CF-BB82-00AA00BDCE0B")]
public interface ICustomDoc
{
    void SetUIHandler(IDocHostUIHandler pUIHandler);
}
```

Asocie la clase que implementa IDocHostUIHandler con el documento de WebOC. Si ha implementado la interfaz ICustomDoc anterior, tan pronto como la propiedad de documento de WebOC sea válida, conviértela en un ICustomDoc y llame al método SetUIHandler, pasando la clase que implementa IDocHostUIHandler.

```
// "this" references that class that owns the WebOC control and in this case also implements the
IDocHostUIHandler interface
ICustomDoc customDoc = (ICustomDoc)webBrowser.Document;
customDoc.SetUIHandler(this);
```

Si NO implementó la interfaz ICustomDoc, tan pronto como la propiedad de documento de WebOC sea válida, `SetClientSite` deberá convertirla en un IOleObject y llamar al método, pasando la clase que implementa IDocHostUIHandler. Establezca el indicador DOCHOSTUIFLAG_DPI_AWARE en el DOCHOSTUIINFO pasado a la `GetHostInfo` llamada al método:

```
public int GetHostInfo(DOCHOSTUIINFO info)
{
    // This is what the default site provides.
    info.dwFlags = (DOCHOSTUIFLAG)0x5a74012;
    // Add the DPI flag to the defaults
    info.dwFlags |= DOCHOSTUIFLAG.DOCHOSTUIFLAG_DPI_AWARE;
    return S_OK;
}
```

Esto debe ser todo lo que necesita para obtener el control WebOC para admitir HPDI.

Sugerencias

1. Si cambia la propiedad document del control WebOC, es posible que deba volver a asociar el documento con la clase IDocHostUIHandler.
2. Si lo anterior no funciona, hay un problema conocido con el WebOC no recoger el cambio al indicador DPI. La forma más confiable de arreglar esto es alternar el zoom óptico del WebOC, lo que significa dos llamadas con dos valores diferentes para el porcentaje de zoom. Además, si se requiere esta solución alternativa, puede ser necesario realizarla en cada llamada de navegación.

```
// browser2 is a SHDocVw.IWebBrowser2 in this case
// EX: Call the Exec twice with DPI%-1 and then DPI% as the zoomPercent values
IOleCommandTarget cmdTarget = browser2.Document as IOleCommandTarget;
if (cmdTarget != null)
{
    object commandInput = zoomPercent;
    cmdTarget.Exec(IntPtr.Zero,
        OLECMDID_OPTICAL_ZOOM,
        OLECMDEXECOPT_DONTPROMPTUSER,
        ref commandInput,
        ref commandOutput);
}
```

Servicio y catálogo de imágenes

06/04/2020 • 45 minutes to read • [Edit Online](#)

Este libro de recetas contiene instrucciones y procedimientos recomendados para adoptar el servicio de imágenes de Visual Studio y el catálogo de imágenes introducidos en Visual Studio 2015.

El servicio de imágenes introducido en Visual Studio 2015 permite a los desarrolladores obtener las mejores imágenes para el dispositivo y el tema elegido por el usuario para mostrar la imagen, incluido el tema correcto para el contexto en el que se muestran. La adopción del servicio de imágenes ayudará a eliminar los principales puntos débiles relacionados con el mantenimiento de activos, el escalado de HDPI y el tema.

Problemas de hoy	Soluciones
Mezcla de color de fondo	Mezcla alfa incorporada
Imágenes temáticas (algunas)	Metadatos temáticos
Modo de contraste alto	Recursos alternativos de alto contraste
Necesita varios recursos para diferentes modos de PPP	Recursos seleccionables con reserva basada en vectores
Imágenes duplicadas	Un identificador por concepto de imagen

¿Por qué adoptar el servicio de imágenes?

- Obtenga siempre la última imagen "pixel-perfect" de Visual Studio
- Puede enviar y utilizar sus propias imágenes
- No es necesario probar las imágenes cuando Windows agrega nuevo escalado de PPP
- Aborde los antiguos obstáculos arquitectónicos en sus implementaciones

La barra de herramientas del shell de Visual Studio antes y después de usar el servicio de imágenes:

Antes



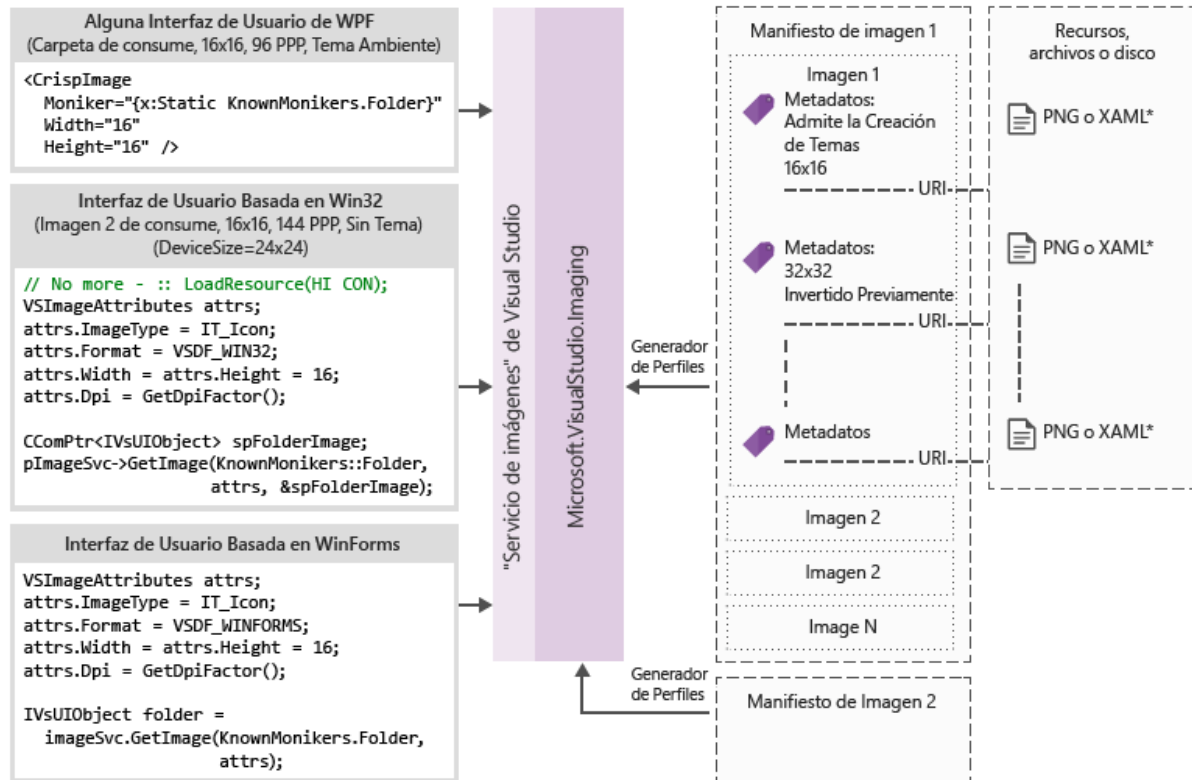
Después

Funcionamiento

El servicio de imágenes puede proporcionar una imagen de mapa de bits adecuada para cualquier marco de interfaz de usuario compatible:

- WPF: BitmapSource
- WinForms: System.Drawing.Bitmap
- Win32: HBITMAP

Diagrama de flujo de servicio de imágenes



Monikers de imagen

Un moniker de imagen (o moniker para abreviar) es un par GUID/ID que identifica de forma única un recurso de imagen o un recurso de lista de imágenes en la biblioteca de imágenes.

Monikers conocidos

Conjunto de monikers de imagen contenidos en el catálogo de imágenes de Visual Studio y consumible públicamente por cualquier componente o extensión de Visual Studio.

Archivos de manifiesto de imagen

Los archivos de manifiesto de imagen (*.imagemanifest*) son archivos XML que definen un conjunto de activos de imagen, los monikers que representan esos activos y la imagen o imágenes reales que representan cada recurso. Los manifiestos de imagen pueden definir imágenes independientes o listas de imágenes para la compatibilidad con la interfaz de usuario heredada. Además, hay atributos que se pueden establecer en el activo o en las imágenes individuales detrás de cada activo para cambiar cuándo y cómo se muestran esos activos.

Esquema de manifiesto de imagen

Un manifiesto de imagen completo tiene este aspecto:

```

<ImageManifest>
  <!-- zero or one Symbols elements -->
  <Symbols>
    <!-- zero or more Import, Guid, ID, or String elements -->
  </Symbols>
  <!-- zero or one Images elements -->
  <Images>
    <!-- zero or more Image elements -->
  </Images>
  <!-- zero or one ImageLists elements -->
  <ImageLists>
    <!-- zero or more ImageList elements -->
  </ImageLists>
</ImageManifest>

```

Symbols

Como ayuda de legibilidad y mantenimiento, el manifiesto de imagen puede utilizar símbolos para los valores de atributo. Los símbolos se definen así:

```

<Symbols>
  <Import Manifest="manifest" />
  <Guid Name="ShellCommandGuid" Value="8ee4f65d-bab4-4cde-b8e7-ac412abbda8a" />
  <ID Name="cmdidSaveAll" Value="1000" />
  <String Name="AssemblyName" Value="Microsoft.VisualStudio.Shell.UI.Internal" />
</Symbols>

```

Subelemento	Definición
Importar	Importa los símbolos del archivo de manifiesto dado para su uso en el manifiesto actual
Guid	El símbolo representa un GUID y debe coincidir con el formato GUID
id	El símbolo representa un identificador y debe ser un entero no negativo
String	El símbolo representa un valor de cadena arbitrario

Los símbolos distinguen entre mayúsculas y minúsculas y se hace referencia a ellos mediante la sintaxis \$(nombre-símbolo):

```

<Image Guid="$(ShellCommandGuid)" ID="$(cmdidSaveAll)" >
  <Source Uri="$(AssemblyName);Component/Resources/image.xaml" />
</Image>

```

Algunos símbolos están predefinidos para todos los manifiestos. Se pueden utilizar en el <atributo Uri <del elemento Source> o Import> para hacer referencia a rutas de acceso en el equipo local.

Símbolo	Descripción
CommonProgramFiles	El valor de la variable de entorno %CommonProgramFiles%

LocalAppData	El valor de la variable de entorno %LocalAppData%
ManifestFolder	La carpeta que contiene el archivo de manifiesto
MyDocuments	La ruta de acceso completa de la carpeta Mis documentos del usuario actual
ProgramFiles	El valor de la variable de entorno %ProgramFiles%
Sistema	La carpeta <i>Windows-System32</i>
WinDir	El valor de la variable de entorno %WinDir%

Imagen

El <elemento Image> define una imagen a la que puede hacer referencia un moniker. El GUID y el identificador tomados juntos forman el moniker de imagen. El moniker de la imagen debe ser único en toda la biblioteca de imágenes. Si más de una imagen tiene un moniker determinado, la primera encontrada al compilar la biblioteca es la que se conserva.

Debe contener al menos una fuente. Las fuentes de tamaño neutro darán los mejores resultados en una amplia gama de tamaños, pero no son necesarias. Si se solicita al servicio una imagen de <un tamaño no definido en el elemento Image> y no hay ningún origen de tamaño neutro, el servicio elegirá el mejor origen específico del tamaño y lo escalará al tamaño solicitado.

```
<Image Guid="guid" ID="int" AllowColorInversion="true/false">
  <Source ... />
  <!-- optional additional Source elements -->
</Image>
```

Atributo	Definición
Guid	[Requerido] La parte GUID del moniker de imagen
id	[Requerido] La parte id del moniker de imagen
AllowColorInversion	[Opcional, predeterminado true] Indica si la imagen puede tener sus colores invertidos mediante programación cuando se usa en un fondo oscuro.

Origen

El <elemento Source> define un único recurso de origen de imagen (XAML y PNG).

```
<Source Uri="uri" Background="background">
  <!-- optional NativeResource element -->
</Source>
```

Atributo	Definición

Identificador URI	[Requerido] URI que define desde dónde se puede cargar la imagen. Puede tener uno de los valores siguientes: - Un URI de paquete con la autoridad application:/// - Una referencia de recursos de componentes absoluta - Una ruta de acceso a un archivo que contiene un recurso nativo
Información previa	[Opcional] Indica qué tipo de fondo está destinado a utilizar se va a utilizar el origen. Puede tener uno de los valores siguientes: <i>Luz</i>: La fuente se puede utilizar en un fondo claro. <i>Oscuro</i>: La fuente se puede utilizar sobre un fondo oscuro. <i>HighContrast</i>: La fuente se puede utilizar en cualquier fondo en el modo de contraste alto. <i>HighContrastLight</i>: La fuente se puede utilizar en un fondo claro en el modo de contraste alto. <i>HighContrastDark</i>: La fuente se puede utilizar en un fondo oscuro en el modo de contraste alto. Si se omite el atributo Background, el origen se puede utilizar en cualquier fondo. Si Background es <i>Light</i>, <i>Dark</i>, <i>HighContrastLight</i> o <i>HighContrastDark</i>, los colores del origen nunca se invierten. Si Background se omite o se establece en <i>HighContrast</i>, la inversión de los colores del origen se controla mediante el atributo AllowColorInversion de la imagen.

Un <elemento Source> puede tener exactamente uno de los siguientes subelementos opcionales:

Elemento	Atributos (todos requeridos)	Definición
<Tamaño>	Value	La fuente se utilizará para imágenes del tamaño dado (en unidades de dispositivo). La imagen será cuadrada.
<> sizeRange	MinSize, MaxSize	El origen se utilizará para las imágenes de MinSize a MaxSize (en unidades de dispositivo) de forma inclusiva. La imagen será cuadrada.
<Dimensiones>	Ancho, alto	La fuente se utilizará para imágenes de la anchura y altura dadas (en unidades de dispositivo).
<> De DimensionRange	MinWidth, MinHeight, MaxWidth, MaxHeight	La fuente se utilizará para imágenes desde la anchura/altura mínima hasta la anchura/altura máxima (en unidades de dispositivo) inclusive.

Un <elemento source> también <puede tener un subelemento <nativoopcional>, que define un> de origen que

se carga desde un ensamblado nativo en lugar de un ensamblado administrado.

```
<NativeResource Type="type" ID="int" />
```

Atributo	Definición
Tipo	[Requerido] El tipo del recurso nativo, XAML o PNG
id	[Requerido] La parte de ID entero del recurso nativo

Imagelist

El <Elemento de> ImageList define una colección de imágenes que se pueden devolver en una sola tira. La tira se construye bajo demanda, según sea necesario.

```
<ImageList>
  <ContainedImage Guid="guid" ID="int" External="true/false" />
  <!-- optional additional ContainedImage elements -->
</ImageList>
```

Atributo	Definición
Guid	[Requerido] La parte GUID del moniker de imagen
id	[Requerido] La parte id del moniker de imagen
Externo	[Opcional, predeterminado false] Indica si el moniker de imagen hace referencia a una imagen en el manifiesto actual.

El moniker de la imagen contenida no tiene que hacer referencia a una imagen definida en el manifiesto actual. Si la imagen contenida no se encuentra en la biblioteca de imágenes, se utilizará una imagen de marcador de posición en blanco en su lugar.

Uso del servicio de imágenes

Primeros pasos (administrados)

Para usar el servicio de imágenes, debe agregar referencias a algunos o todos los ensamblados siguientes al proyecto:

- *Microsoft.VisualStudio.ImageCatalog.dll*
 - Necesario si utiliza el catálogo de imágenes integrado **KnownMonikers**.
- *Microsoft.VisualStudio.Imaging.dll*
 - Necesario si usa **CrispImage** e **ImageThemingUtilities** en la interfaz de usuario de WPF.
- *Microsoft.VisualStudio.Imaging.Interop.14.0.DesignTime.dll*
 - Necesario si utiliza los tipos **ImageMoniker** y **ImageAttributes**.
 - **EmbedInteropTypes** debe establecerse en true.
- *Microsoft.VisualStudio.Shell.Interop.14.0.DesignTime*

- Necesario si usa el **IVsImageService2** tipo.
- **EmbedInteropTypes** debe establecerse en true.
- *Microsoft.VisualStudio.Utilities.dll*
 - Necesario si usa el **BrushToColorConverter** para el **ImageThemingUtilities.ImageBackgroundColor** en la interfaz de usuario de WPF.
- *Microsoft.VisualStudio.Shell. <VSVersion>.0*
 - Necesario si utiliza el **Tipo IVsUIObject**.
- *Microsoft.VisualStudio.Shell.Interop.10.0.dll*
 - Necesario si utiliza las aplicaciones auxiliares de interfaz de usuario relacionadas con WinForms.
 - **EmbedInteropTypes** debe establecerse en true

Primeros pasos (nativo)

Para utilizar el servicio de imágenes, debe incluir algunos o todos los encabezados siguientes en el proyecto:

- **KnownImageIds.h**
 - Necesario si usa el catálogo de imágenes integrado **KnownMonikers**, pero no puede usar el tipo **ImageMoniker**, como al devolver valores de **IVsHierarchy GetGuidProperty** o **GetProperty** llamadas.
- **KnownMonikers.h**
 - Necesario si utiliza el catálogo de imágenes integrado **KnownMonikers**.
- **ImageParameters140.h**
 - Necesario si utiliza los tipos **ImageMoniker** y **ImageAttributes**.
- **VSShell140.h**
 - Necesario si usa el **IVsImageService2** tipo.
- **ImageThemingUtilities.h**
 - Necesario si no puede permitir que el servicio de imágenes se encargue de la segmentación por usted.
 - No utilice este encabezado si el servicio de imágenes puede controlar el tema de la imagen.
- **VSUIDPIHelper.h**
 - Necesario si utiliza las aplicaciones auxiliares de PPP para obtener el PPP actual.
- **VsDpiAwareness.h**
 - Necesario si utiliza las aplicaciones auxiliares de reconocimiento de PPP para obtener el PPP actual.

¿Cómo escribo una nueva interfaz de usuario de WPF?

1. Comience agregando las referencias de ensamblado necesarias en la sección de los primeros pasos anteriores al proyecto. No es necesario agregarlos todos, así que agregue solo las referencias que necesita. (Nota: si está utilizando o tiene acceso a **Colores** en lugar de **Pinceles**, entonces puede omitir la referencia a **Utilidades**, ya que no necesitará el convertidor.)
2. Seleccione la imagen deseada y obtenga su apodo. Utilice un **KnownMoniker** o utilice el suyo propio si tiene sus propias imágenes y monikers personalizados.
3. Agrega **CrispImages** a tu XAML. (Véase el siguiente ejemplo.)

4. Establezca la propiedad **ImageThemingUtilities.ImageBackgroundColor** en la jerarquía de la interfaz de usuario. (Esto debe establecerse en la ubicación donde se conoce el color de fondo, no necesariamente en el **CrispImage**.) (Véase el siguiente ejemplo.)

```
<Window
  x:Class="WpfApplication.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:imaging="clr-namespace:Microsoft.VisualStudio.Imaging;assembly=Microsoft.VisualStudio.Imaging"
  xmlns:theming="clr-namespace:Microsoft.VisualStudio.PlatformUI;assembly=Microsoft.VisualStudio.Imaging"
  xmlns:utilities="clr-namespace:Microsoft.VisualStudio.Imaging;assembly=Microsoft.VisualStudio.Imaging"
  xmlns:catalog="clr-namespace:Microsoft.VisualStudio.Imaging;assembly=Microsoft.VisualStudio.ImageCatalog"
  Title="MainWindow" Height="350" Width="525" UseLayoutRounding="True">
  <Window.Resources>
    <utilities:BrushToColorConverter x:Key="BrushToColorConverter"/>
  </Window.Resources>
  <StackPanel Background="White" VerticalAlignment="Center"
    theming:ImageThemingUtilities.ImageBackgroundColor="{Binding Background, RelativeSource={RelativeSource
    Self}, Converter={StaticResource BrushToColorConverter}}">
    <imaging:CrispImage Width="16" Height="16" Moniker="{x:Static catalog:KnownMonikers.MoveUp}" />
  </StackPanel>
</Window>
```

¿Cómo actualizo la interfaz de usuario de WPF existente?

La actualización de la interfaz de usuario de WPF existente es un proceso relativamente sencillo que consta de tres pasos básicos:

1. Reemplace <todos los elementos <Image> de la interfaz de usuario por elementos de> CrispImage.
2. Cambie todos los atributos Source a Moniker attributes.
 - Si la imagen nunca cambia y está utilizando **KnownMonikers**, vincule estáticamente esa propiedad al **KnownMoniker**. (Véase el ejemplo anterior.)
 - Si la imagen nunca cambia y está utilizando su propia imagen personalizada, entonces enlace estáticamente a su propio apodo.
 - Si la imagen puede cambiar, enlace el Moniker atributo a una propiedad de código que notifica los cambios de propiedad.
3. En algún lugar de la jerarquía de la interfaz de usuario, establezca **ImageThemingUtilities.ImageBackgroundColor** para asegurarse de que la inversión de color funciona correctamente.
 - Esto puede requerir el uso de la **BrushToColorConverter** clase. (Véase el ejemplo anterior.)

¿Cómo actualizo la interfaz de usuario de Win32?

Agregue lo siguiente al código cuando corresponda para reemplazar la carga sin procesar de imágenes. Cambiar valores para devolver HBITMAPs frente a HICONs frente a HIMAGELIST según sea necesario.

Obtenga el servicio de imágenes

```
COMPtr<IVsImageService2> spImgSvc;
GlobalServiceProvid::HrQueryService(SID_SVsImageService, &spImgSvc);
```

Solicitar la imagen

```

ImageAttributes attr = { 0 };
attr.StructSize      = sizeof(attributes);
attr.Format          = DF_Win32;
// IT_Bitmap for HBITMAP, IT_Icon for HICON, IT_ImageList for HIMAGELIST
attr.ImageType       = IT_Bitmap;
attr.LogicalWidth    = 16;
attr.LogicalHeight   = 16;
attr.Dpi             = VsUI::DpiHelper::GetDeviceDpiX();
// Desired RGBA color, if you don't use this, don't set IAF_Background below
attr.Background      = 0xFFFFFFFF;
attr.Flags           = IAF_RequiredFlags | IAF_Background;

CComPtr<IVsUIObject> spImg;
// Replace this KnownMoniker with your desired ImageMoniker
spImgSvc->GetImage(KnownMonikers::Blank, attributes, &spImg);

```

```

UINT dpiX, dpiY;
HWND hwnd = // get the HWND where the image will be displayed
VsUI::CDpiAwareness::GetDpiForWindow(hwnd, &dpiX, &dpiY);

ImageAttributes attr = { 0 };
attr.StructSize      = sizeof(attributes);
attr.Format          = DF_Win32;
// IT_Bitmap for HBITMAP, IT_Icon for HICON, IT_ImageList for HIMAGELIST
attr.ImageType       = IT_Bitmap;
attr.LogicalWidth    = 16;
attr.LogicalHeight   = 16;
attr.Dpi             = dpiX;
// Desired RGBA color, if you don't use this, don't set IAF_Background below
attr.Background      = 0xFFFFFFFF;
attr.Flags           = IAF_RequiredFlags | IAF_Background;

CComPtr<IVsUIObject> spImg;
// Replace this KnownMoniker with your desired ImageMoniker
spImgSvc->GetImage(KnownMonikers::Blank, attributes, &spImg);

```

¿Cómo actualizo la interfaz de usuario de WinForms?

Agregue lo siguiente al código cuando corresponda para reemplazar la carga sin procesar de imágenes. Cambie los valores para devolver mapas de bits frente a iconos según sea necesario.

Útil uso de la declaración

```
using GelUtilities = Microsoft.Internal.VisualStudio.PlatformUI.Utilities;
```

Obtenga el servicio de imágenes

```
// This or your preferred way of querying for Visual Studio services
IVsImageService2 imageService = (IVsImageService2)Package.GetGlobalService(typeof(SVsImageService));
```

Solicitar la imagen

```

ImageAttributes attributes = new ImageAttributes
{
    StructSize    = Marshal.SizeOf(typeof(ImageAttributes)),
    // IT_Bitmap for Bitmap, IT_Icon for Icon, IT_ImageList for ImageList
    ImageType     = (uint)_UIImageType.IT_Bitmap,
    Format        = (uint)_UIDataFormat.DF_WinForms,
    LogicalWidth  = 16,
    LogicalHeight = 16,
    Dpi           = (int)DpiHelper.DeviceDpiX;
    // Desired RGBA color, if you don't use this, don't set IAF_Background below
    Background    = 0xFFFFFFFF,
    Flags         = unchecked((uint)_ImageAttributesFlags.IAF_RequiredFlags |
    _ImageAttributesFlags.IAF_Background),
};

// Replace this KnownMoniker with your desired ImageMoniker
IVsUIObject uiObj = imageService.GetImage(KnownMonikers.Blank, attributes);

Bitmap bitmap = (Bitmap)GelUtilities.GetObjectData(uiObj); // Use this if you need a bitmap
// Icon icon = (Icon)GelUtilities.GetObjectData(uiObj);    // Use this if you need an icon

```

```

Control control = // get the control where the image will be displayed

ImageAttributes attributes = new ImageAttributes
{
    StructSize    = Marshal.SizeOf(typeof(ImageAttributes)),
    // IT_Bitmap for Bitmap, IT_Icon for Icon, IT_ImageList for ImageList
    ImageType     = (uint)_UIImageType.IT_Bitmap,
    Format        = (uint)_UIDataFormat.DF_WinForms,
    LogicalWidth  = 16,
    LogicalHeight = 16,
    Dpi           = (int)DpiAwareness.GetWindowDpi(control.Handle);
    // Desired RGBA color, if you don't use this, don't set IAF_Background below
    Background    = 0xFFFFFFFF,
    Flags         = unchecked((uint)_ImageAttributesFlags.IAF_RequiredFlags |
    _ImageAttributesFlags.IAF_Background),
};

// Replace this KnownMoniker with your desired ImageMoniker
IVsUIObject uiObj = imageService.GetImage(KnownMonikers.Blank, attributes);

Bitmap bitmap = (Bitmap)GelUtilities.GetObjectData(uiObj); // Use this if you need a bitmap
// Icon icon = (Icon)GelUtilities.GetObjectData(uiObj);    // Use this if you need an icon

```

¿Cómo utilizo monikers de imagen en una nueva ventana de herramientas?

La plantilla de proyecto de paquete VSIX se actualizó para Visual Studio 2015. Para crear una nueva ventana de herramientas, haga clic con el botón derecho en el proyecto VSIX y seleccione **Agregar > nuevo elemento** (**Ctrl+Shift+A**). En el nodo Extensibilidad para el idioma del proyecto, seleccione Ventana de **herramientas personalizadas**, asigne un nombre a la ventana de herramientas y pulse el botón **Agregar**.

Estos son los lugares clave para usar monikers en una ventana de herramientas. Siga las instrucciones para cada uno:

1. La pestaña de la ventana de herramientas cuando las pestañas se vuelven lo suficientemente pequeñas (también se utiliza en el modificador de ventana **Ctrl+Tab**).

Agregue esta línea al constructor de la clase que deriva del tipo **ToolWindowPane**:

```
// Replace this KnownMoniker with your desired ImageMoniker
this.BitmapImageMoniker = KnownMonikers.Blank;
```

2. El comando para abrir la ventana de herramientas.

En el archivo `.vsct` del paquete, edite el botón de comando de la ventana de herramientas:

```
<Button guid="guidPackageCmdSet" id="CommandId" priority="0x0100" type="Button">
  <Parent guid="guidSHLMainMenu" id="IDG_VS_WNDO_OTRWNDWS1"/>
  <!-- Replace this KnownMoniker with your desired ImageMoniker -->
  <Icon guid="ImageCatalogGuid" id="Blank" />
  <!-- Add this -->
  <CommandFlag>IconIsMoniker</CommandFlag>
  <Strings>
    <ButtonText>MyToolWindow</ButtonText>
  </Strings>
</Button>
```

¿Cómo utilizo monikers de imagen en una ventana de herramientas existente?

La actualización de una ventana de herramientas existente para utilizar monikers de imagen es similar a los pasos para crear una nueva ventana de herramientas.

Estos son los lugares clave para usar monikers en una ventana de herramientas. Siga las instrucciones para cada uno:

3. La pestaña de la ventana de herramientas cuando las pestañas se vuelven lo suficientemente pequeñas (también se utiliza en el modificador de ventana **Ctrl + Tab**).

a. Quite estas líneas (si existen) en el constructor de la clase que deriva del tipo **ToolWindowPane**:

```
this.BitmapResourceID = <Value>;
this.BitmapIndex = <Value>;
```

b. Consulte el paso #1 de la "¿Cómo uso los monikers de imagen en una nueva ventana de herramientas?" sección anterior.

4. El comando para abrir la ventana de herramientas.

- Vea el paso #2 de la "¿Cómo uso los monikers de imagen en una nueva ventana de herramientas?" sección anterior.

¿Cómo uso los monikers de imagen en un archivo `.vsct`?

Actualice su archivo `.vsct` como se indica en las líneas comentadas a continuación:


```

<?xml version="1.0" encoding="utf-8"?>
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Include the definitions for images included in the VS image catalog -->
  <Include href="KnownImageIds.vscx" />
  <Commands package="guidMyPackage">
    <Buttons>
      <Button guid="guidMyCommandSet" id="cmdidMyCommand" priority="0x0000" type="Button">
        <!-- Add an Icon element, changing the attributes to match the image moniker you want to use.
              In this case, we're using the Guid for the VS image catalog.
              Change the id attribute to be the ID of the desired image moniker. -->
        <Icon guid="ImageCatalogGuid" id="OpenFolder" />
        <CommandFlag>DynamicVisibility</CommandFlag>
        <CommandFlag>DefaultInvisible</CommandFlag>
        <CommandFlag>DefaultDisabled</CommandFlag>
        <CommandFlag>CommandWellOnly</CommandFlag>
        <CommandFlag>IconAndText</CommandFlag>
        <!-- Add the IconIsMoniker CommandFlag -->
        <CommandFlag>IconIsMoniker</CommandFlag>
        <Strings>
          <ButtonText>Quick Fixes...</ButtonText>
          <CommandName>Show Quick Fixes</CommandName>
          <CanonicalName>ShowQuickFixes</CanonicalName>
          <LocCanonicalName>ShowQuickFixes</LocCanonicalName>
        </Strings>
      </Button>
    </Buttons>
  </Commands>
  <!-- It is recommended that you remove <Bitmap> elements that are no longer used in the vscx file -->
  <Symbols>
    <GuidSymbol name="guidMyPackage" value="{1491e936-6ffe-474e-8371-30e5920d8fdd}" />
    <GuidSymbol name="guidMyCommandSet" value="{10347de4-69a9-47f4-a950-d3301f6d2bc7}">
      <IDSymbol name="cmdidMyCommand" value="0x9437" />
    </GuidSymbol>
  </Symbols>
</CommandTable>

```

¿Qué sucede si mi archivo .vscx también necesita ser leído por versiones anteriores de Visual Studio?

Las versiones anteriores de Visual Studio no reconocen la marca de comando **IconIsMoniker**. Puede usar imágenes del servicio de imágenes en versiones de Visual Studio que lo admiten, pero seguir usando imágenes de estilo antiguo en versiones anteriores de Visual Studio. Para ello, dejaría el archivo .vscx sin cambios (y, por lo tanto, compatible con versiones anteriores de Visual Studio) y crearía <un archivo CSV (valores separados por comas) que se asigna a partir de pares GUID/ID definidos en el elemento de> Bitmaps de un archivo .vscx a los pares GUID/ID de moniker de imagen.

El formato del archivo CSV de asignación es:

```

Icon guid, Icon id, Moniker guid, Moniker id
b714fcf7-855e-4e4c-802a-1fd87144ccad,1,fda30684-682d-421c-8be4-650a2967058e,100
b714fcf7-855e-4e4c-802a-1fd87144ccad,2,fda30684-682d-421c-8be4-650a2967058e,200

```

El archivo CSV se implementa con el paquete y su ubicación se especifica mediante la propiedad **IconMappingFilename** del atributo de paquete **ProvideMenuResource**:

```

[ProvideMenuResource("MyPackage.ctmenu", 1, IconMappingFilename="IconMappings.csv")]

```

El **IconMappingFilename** es una ruta de acceso relativa implícitamente arraigada en **\$PackageFolder\$** (como en el ejemplo anterior) o una ruta de acceso absoluta explícitamente arraigada en un directorio definido por una

¿Cómo puedo convertir de una tira de imagen real a una tira de imagen basada en moniker?

Necesito apoyar a HIMAGELISTS

Si ya hay una tira de imágenes existente para el código que desea actualizar para usar el servicio de imágenes, pero está limitado por las API que requieren pasar listas de imágenes, todavía puede obtener las ventajas del servicio de imágenes. Para crear una tira de imagen basada en moniker, siga los pasos que se indican a continuación para crear un manifiesto a partir de monikers existentes.

1. Ejecute la herramienta **ManifestFromResources** y pasándole la tira de imagen. Esto generará un manifiesto para la tira.
 - Recomendado: proporcione un nombre no predeterminado para que el manifiesto se adapte a su uso.
2. Si solo utiliza **KnownMonikers**, haga lo siguiente:
 - Reemplace <la sección Imágenes <> del manifiesto por Imágenes/>.
 - Elimine todos los ID de subimagen (cualquier cosa con <el nombre de la tira de imágenes>).
 - Recomendado: cambie el nombre del símbolo AssetsGuid y del símbolo de la tira de imagen para que se adapte a su uso.
 - Reemplace cada GUID de **ContainedImage** por \$(ImageCatalogGuid), reemplace cada ID<de **ContainedImage** por \$(moniker>) y agregue el atributo External-"true" a cada **ContainedImage**
 - <> de apodos deben reemplazarse con el **ConocidoMoniker** que coincida con la imagen pero con el "ConocidoMonikers". del nombre.
 - Agregue <Manifiesto de importación\ "\$(ManifestFolder)> <Ruta de acceso de dir de* instalación relativa a <* .Microsoft.VisualStudio.ImageCatalog.imagemanifest" /> en la parte superior de la> sección Símbolos.
 - La ruta de acceso relativa viene determinada por la ubicación de implementación definida en la creación de la configuración para el manifiesto.
3. Ejecute la herramienta **ManifestToCode** para generar contenedores de modo que el código existente tenga un moniker que pueda usar para consultar el servicio de imágenes para la tira de imágenes.
 - Recomendado: proporcione nombres no predeterminados para los contenedores y espacios de nombres para adaptarse a su uso.
4. Realice todos los complementos, la creación/implementación de la configuración y otros cambios de código para trabajar con el servicio de imágenes y los nuevos archivos.

Manifiesto de ejemplo que incluye imágenes internas y externas para ver cómo debería ser:

```

<?xml version="1.0"?>
<ImageManifest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.microsoft.com/VisualStudio/ImageManifestSchema/2014">

  <Symbols>
    <!-- This needs to be the relative path from your manifest to the ImageCatalog's manifest
         where $(ManifestFolder) is the deployed location of this manifest. -->
    <Import Manifest="$(ManifestFolder)\<RelPath>\Microsoft.VisualStudio.ImageCatalog.imagemanifest" />

    <String Name="Resources" Value="/My.Assembly.Name;Component/Resources/ImageStrip" />
    <Guid Name="ImageGuid" Value="{fb41b7ef-6587-480c-aa27-5b559d42cfc9}" />
    <Guid Name="ImageStripGuid" Value="{9c84a570-d9a7-4052-a340-188fb276f973}" />
    <ID Name="MyImage_0" Value="100" />
    <ID Name="MyImage_1" Value="101" />
    <ID Name="InternalList" Value="1001" />
    <ID Name="ExternalList" Value="1002" />
  </Symbols>

  <Images>
    <Image Guid="$(ImageGuid)" ID="$(MyImage_0)">
      <Source Uri="$(Resources)/MyImage_0.png">
        <Size Value="16" />
      </Source>
    </Image>
    <Image Guid="$(ImageGuid)" ID="$(MyImage_1)">
      <Source Uri="$(Resources)/MyImage_1.png">
        <Size Value="16" />
      </Source>
    </Image>
  </Images>

  <ImageLists>
    <ImageList Guid="$(ImageStripGuid)" ID="$(InternalList)">
      <ContainedImage Guid="$(ImageGuid)" ID="$(MyImage_0)" />
      <ContainedImage Guid="$(ImageGuid)" ID="$(MyImage_1)" />
    </ImageList>
    <ImageList Guid="$(ImageStripGuid)" ID="$(ExternalList)">
      <ContainedImage Guid="$(ImageCatalogGuid)" ID="$(StatusError)" External="true" />
      <ContainedImage Guid="$(ImageCatalogGuid)" ID="$(StatusWarning)" External="true" />
      <ContainedImage Guid="$(ImageCatalogGuid)" ID="$(StatusInformation)" External="true" />
    </ImageList>
  </ImageLists>

</ImageManifest>

```

No necesito apoyar a HIMAGELISTS

1. Determine el conjunto de **KnownMonikers** que coincidan con las imágenes de la tira de imágenes o cree sus propios monikers para las imágenes de la tira de imágenes.
2. Actualice cualquier asignación que haya utilizado para obtener la imagen en el índice requerido en la tira de imagen para usar los monikers en su lugar.
3. Actualice el código para usar el servicio de imágenes para solicitar monikers a través de la asignación actualizada. (Esto podría significar actualizar a **CrispImages** para código administrado, o solicitar HBITMAPs o HICONs del servicio de imágenes y pasarlos para código nativo.)

Prueba de tus imágenes

Puede utilizar la herramienta Visor de biblioteca de imágenes para probar los manifiestos de imagen para asegurarse de que todo está creado correctamente. Puede encontrar la herramienta en el SDK de [Visual Studio](#)

2015. La documentación de esta herramienta y otras se pueden encontrar [aquí](#).

Recursos adicionales

Ejemplos

Varios de los ejemplos de Visual Studio en GitHub se han actualizado para mostrar cómo usar el servicio de imágenes como parte de varios puntos de extensibilidad de Visual Studio.

Compruebe <http://github.com/Microsoft/VSSDK-Extensibility-Samples> las muestras más recientes.

Tooling

Se creó un conjunto de herramientas de soporte técnico para el servicio de imágenes para ayudar a crear o actualizar la interfaz de usuario que funciona con el servicio de imágenes. Para obtener más información acerca de cada herramienta, consulte la documentación que viene con las herramientas. Las herramientas se incluyen como parte del SDK de [Visual Studio 2015](#).

ManifestFromResources

La herramienta Manifiesto de recursos toma una lista de recursos de imagen (PNG o XAML) y genera un archivo de manifiesto de imagen para usar esas imágenes con el servicio de imágenes.

ManifestToCode

La herramienta Manifiesto en código toma un archivo de manifiesto de imagen y genera un archivo contenedor para hacer referencia a los valores de manifiesto en código (c++ , C o VB) o archivos *.vsct*.

ImageLibraryViewer

La herramienta Visor de biblioteca de imágenes puede cargar manifiestos de imagen y permite al usuario manipularlos de la misma manera que Visual Studio para asegurarse de que el manifiesto se crea correctamente. El usuario puede modificar el fondo, los tamaños, la configuración de PPP, el contraste alto y otros ajustes. También muestra información de carga para buscar errores en los manifiestos y muestra información de origen para cada imagen del manifiesto.

Preguntas más frecuentes

- ¿Hay alguna dependencia que debe <incluir al cargar Reference Include"Microsoft.VisualStudio.*.Interop.14.0.DesignTime" />?
 - Establezca `EmbedInteropTypes="true"` en todos los archivos DLL de interoperabilidad.
- ¿Cómo implemento un manifiesto de imagen con mi extensión?
 - Agregue el archivo *.imagemanifest* al proyecto.
 - Establezca "Incluir en VSIX" en True.
- Estoy actualizando mi sistema de proyectos CPS. ¿Qué pasó con **ImageName** y **StockIconService**?
 - Estos se eliminaron cuando CPS se actualizó para usar monikers. Ya no es necesario llamar a **StockIconService**, simplemente pasar el **KnownMoniker** deseado al método o propiedad mediante el método de extensión **ToProjectSystemType()** en las utilidades cpS. Puede encontrar una asignación de **ImageName** a **KnownMonikers** a continuación:

ImageName	KnownMoniker
ImageName.OfflineWebApp	KnownImageIds.Web

ImageName.WebReferencesFolder	KnownImagelds.Web
ImageName.OpenReferenceFolder	KnownImagelds.FolderOpened
ImageName.ReferenceFolder	KnownImagelds.Reference
ImageName.Reference	KnownImagelds.Reference
ImageName.SdlWebReference	KnownImagelds.WebReferenceFolder
ImageName.DiscoWebReference	KnownImagelds.DynamicDiscoveryDocument
ImageName.Folder	KnownImagelds.FolderClosed
ImageName.OpenFolder	KnownImagelds.FolderOpened
ImageName.ExcludedFolder	KnownImagelds.HiddenFolderClosed
ImageName.OpenExcludedFolder	KnownImagelds.HiddenFolderOpened
ImageName.ExcludedFile	KnownImagelds.HiddenFile
ImageName.DependentFile	KnownImagelds.GenerateFile
ImageName.MissingFile	KnownImagelds.DocumentWarning
ImageName.WindowsForm	KnownImagelds.WindowsForm
ImageName.WindowsUserControl	KnownImagelds.UserControl
ImageName.WindowsComponent	KnownImagelds.ComponentFile
ImageName.XmlSchema	KnownImagelds.XMLSchema
ImageName.XmlFile	KnownImagelds.XMLFile
ImageName.WebForm	KnownImagelds.Web
ImageName.WebService	KnownImagelds.WebService
ImageName.WebUserControl	KnownImagelds.WebUserControl
ImageName.WebCustomUserControl	KnownImagelds.WebCustomControl
ImageName.AspPage	KnownImagelds.ASPFile
ImageName.GlobalApplicationClass	KnownImagelds.SettingsFile
ImageName.WebConfig	KnownImagelds.ConfigurationFile
ImageName.HtmlPage	KnownImagelds.HTMLFile

ImageName.StyleSheet	KnownImagelds.StyleSheet
ImageName.ScriptFile	KnownImagelds.JSScript
ImageName.TextFile	KnownImagelds.Document
ImageName.SettingsFile	KnownImagelds.Settings
ImageName.Resources	KnownImagelds.DocumentGroup
ImageName.Bitmap	KnownImagelds.Image
ImageName.Icon	KnownImagelds.IconFile
ImageName.Image	KnownImagelds.Image
ImageName.ImageMap	KnownImagelds.ImageMapFile
ImageName.XWorld	KnownImagelds.XWorldFile
ImageName.Audio	KnownImagelds.Sound
ImageName.Video	KnownImagelds.Media
ImageName.Cab	KnownImagelds.CABProject
ImageName.Jar	KnownImagelds.JARFile
ImageName.DataEnvironment	KnownImagelds.DataTable
ImageName.PreviewFile	KnownImagelds.Report
ImageName.DanglingReference	KnownImagelds.ReferenceWarning
ImageName.XsltFile	KnownImagelds.XSLTransform
ImageName.Cursor	KnownImagelds.CursorFile
ImageName.AppDesignerFolder	KnownImagelds.Property
ImageName.Data	KnownImagelds.Database
ImageName.Application	KnownImagelds.Application
ImageName.DataSet	KnownImagelds.DatabaseGroup
ImageName.Pfx	KnownImagelds.Certificate
ImageName.Snk	KnownImagelds.Rule
ImageName.VisualBasicProject	KnownImagelds.VBProjectNode

ImageName.CSharpProject	KnownImageIds.CSProjectNode
ImageName.Empty	KnownImageIds.Blank
ImageName.MissingFolder	KnownImageIds.FolderOffline
ImageName.SharedImportReference	KnownImageIds.SharedProject
ImageName.SharedProjectCs	KnownImageIds.CSSharedProject
ImageName.SharedProjectVc	KnownImageIds.CPPSharedProject
ImageName.SharedProjectJs	KnownImageIds.JSSharedProject
ImageName.CSharpCodeFile	KnownImageIds.CSFileNode
ImageName.VisualBasicCodeFile	KnownImageIds.VBFileNode

- o Estoy actualizando mi proveedor de lista de finalización. ¿Qué **KnownMonikers** coinciden con los antiguos **estándarStandardGlyphGroup** y **StandardGlyph** valores?

GlyphGroupClass	GlyphItemPublic	ClassPublic
GlyphGroupClass	GlyphItemInternal	ClassInternal
GlyphGroupClass	GlyphItemFriend	ClassInternal
GlyphGroupClass	GlyphItemProtected	ClassProtected
GlyphGroupClass	GlyphItemPrivate	ClassPrivate
GlyphGroupClass	GlyphItemShortcut	ClassShortcut
GlyphGroupConstant	GlyphItemPublic	ConstantPublic
GlyphGroupConstant	GlyphItemInternal	ConstantInternal
GlyphGroupConstant	GlyphItemFriend	ConstantInternal
GlyphGroupConstant	GlyphItemProtected	ConstantProtected
GlyphGroupConstant	GlyphItemPrivate	ConstantPrivate
GlyphGroupConstant	GlyphItemShortcut	ConstantShortcut
GlyphGroupDelegate	GlyphItemPublic	DelegadoPúblico
GlyphGroupDelegate	GlyphItemInternal	DelegadoInterno
GlyphGroupDelegate	GlyphItemFriend	DelegadoInterno

GlyphGroupDelegate	GlyphItemProtected	DelegadoProtegido
GlyphGroupDelegate	GlyphItemPrivate	DelegatePrivate
GlyphGroupDelegate	GlyphItemShortcut	DelegateShortcut
GlyphGroupEnum	GlyphItemPublic	EnumerationPublic
GlyphGroupEnum	GlyphItemInternal	EnumerationInternal
GlyphGroupEnum	GlyphItemFriend	EnumerationInternal
GlyphGroupEnum	GlyphItemProtected	EnumerationProtected
GlyphGroupEnum	GlyphItemPrivate	EnumerationPrivate
GlyphGroupEnum	GlyphItemShortcut	EnumerationShortcut
GlyphGroupEnumMember	GlyphItemPublic	EnumerationItemPublic
GlyphGroupEnumMember	GlyphItemInternal	EnumerationItemInternal
GlyphGroupEnumMember	GlyphItemFriend	EnumerationItemInternal
GlyphGroupEnumMember	GlyphItemProtected	EnumerationItemProtected
GlyphGroupEnumMember	GlyphItemPrivate	EnumerationItemPrivate
GlyphGroupEnumMember	GlyphItemShortcut	EnumerationItemShortcut
GlyphGroupEvent	GlyphItemPublic	EventPublic
GlyphGroupEvent	GlyphItemInternal	EventInternal
GlyphGroupEvent	GlyphItemFriend	EventInternal
GlyphGroupEvent	GlyphItemProtected	EventProtected
GlyphGroupEvent	GlyphItemPrivate	EventPrivate
GlyphGroupEvent	GlyphItemShortcut	EventShortcut
GlyphGroupException	GlyphItemPublic	ExceptionPublic
GlyphGroupException	GlyphItemInternal	ExceptionInternal
GlyphGroupException	GlyphItemFriend	ExceptionInternal
GlyphGroupException	GlyphItemProtected	ExceptionProtected

GlyphGroupException	GlyphItemPrivate	ExceptionPrivate
GlyphGroupException	GlyphItemShortcut	ExceptionShortcut
GlyphGroupField	GlyphItemPublic	FieldPublic
GlyphGroupField	GlyphItemInternal	FieldInternal
GlyphGroupField	GlyphItemFriend	FieldInternal
GlyphGroupField	GlyphItemProtected	FieldProtected
GlyphGroupField	GlyphItemPrivate	FieldPrivate
GlyphGroupField	GlyphItemShortcut	FieldShortcut
GlyphGroupInterface	GlyphItemPublic	InterfacePublic
GlyphGroupInterface	GlyphItemInternal	InterfaceInternal
GlyphGroupInterface	GlyphItemFriend	InterfaceInternal
GlyphGroupInterface	GlyphItemProtected	InterfaceProtected
GlyphGroupInterface	GlyphItemPrivate	InterfacePrivate
GlyphGroupInterface	GlyphItemShortcut	InterfaceShortcut
GlyphGroupMacro	GlyphItemPublic	MacroPublic
GlyphGroupMacro	GlyphItemInternal	MacroInternal
GlyphGroupMacro	GlyphItemFriend	MacroInternal
GlyphGroupMacro	GlyphItemProtected	MacroProtegido
GlyphGroupMacro	GlyphItemPrivate	MacroPrivate
GlyphGroupMacro	GlyphItemShortcut	MacroShortcut
GlyphGroupMap	GlyphItemPublic	MapPublic
GlyphGroupMap	GlyphItemInternal	MapInternal
GlyphGroupMap	GlyphItemFriend	MapInternal
GlyphGroupMap	GlyphItemProtected	MapProtected
GlyphGroupMap	GlyphItemPrivate	MapPrivate
GlyphGroupMap	GlyphItemShortcut	MapShortcut

GlyphGroupMapItem	GlyphItemPublic	MapItemPublic
GlyphGroupMapItem	GlyphItemInternal	MapItemInternal
GlyphGroupMapItem	GlyphItemFriend	MapItemInternal
GlyphGroupMapItem	GlyphItemProtected	MapItemProtected
GlyphGroupMapItem	GlyphItemPrivate	MapItemPrivate
GlyphGroupMapItem	GlyphItemShortcut	MapItemShortcut
GlyphGroupMethod	GlyphItemPublic	MethodPublic
GlyphGroupMethod	GlyphItemInternal	MethodInternal
GlyphGroupMethod	GlyphItemFriend	MethodInternal
GlyphGroupMethod	GlyphItemProtected	MethodProtected
GlyphGroupMethod	GlyphItemPrivate	MethodPrivate
GlyphGroupMethod	GlyphItemShortcut	MethodShortcut
GlyphGroupOverload	GlyphItemPublic	MethodPublic
GlyphGroupOverload	GlyphItemInternal	MethodInternal
GlyphGroupOverload	GlyphItemFriend	MethodInternal
GlyphGroupOverload	GlyphItemProtected	MethodProtected
GlyphGroupOverload	GlyphItemPrivate	MethodPrivate
GlyphGroupOverload	GlyphItemShortcut	MethodShortcut
GlyphGroupModule	GlyphItemPublic	ModulePublic
GlyphGroupModule	GlyphItemInternal	ModuleInternal
GlyphGroupModule	GlyphItemFriend	ModuleInternal
GlyphGroupModule	GlyphItemProtected	ModuleProtected
GlyphGroupModule	GlyphItemPrivate	ModulePrivate
GlyphGroupModule	GlyphItemShortcut	ModuleShortcut
GlyphGroupNamespace	GlyphItemPublic	NamespacePublic
GlyphGroupNamespace	GlyphItemInternal	NamespaceInternal

GlyphGroupNamespace	GlyphItemFriend	NamespaceInternal
GlyphGroupNamespace	GlyphItemProtected	NamespaceProtected
GlyphGroupNamespace	GlyphItemPrivate	NamespacePrivate
GlyphGroupNamespace	GlyphItemShortcut	NamespaceShortcut
GlyphGroupOperator	GlyphItemPublic	OperatorPublic
GlyphGroupOperator	GlyphItemInternal	OperatorInternal
GlyphGroupOperator	GlyphItemFriend	OperatorInternal
GlyphGroupOperator	GlyphItemProtected	OperadorProtegido
GlyphGroupOperator	GlyphItemPrivate	OperatorPrivate
GlyphGroupOperator	GlyphItemShortcut	OperatorShortcut
GlyphGroupProperty	GlyphItemPublic	PropertyPublic
GlyphGroupProperty	GlyphItemInternal	PropertyInternal
GlyphGroupProperty	GlyphItemFriend	PropertyInternal
GlyphGroupProperty	GlyphItemProtected	PropertyProtected
GlyphGroupProperty	GlyphItemPrivate	PropertyPrivate
GlyphGroupProperty	GlyphItemShortcut	PropertyShortcut
GlyphGroupStruct	GlyphItemPublic	EstructuraPúblico
GlyphGroupStruct	GlyphItemInternal	EstructuralInterno
GlyphGroupStruct	GlyphItemFriend	EstructuralInterno
GlyphGroupStruct	GlyphItemProtected	EstructuraProtegido
GlyphGroupStruct	GlyphItemPrivate	StructurePrivate
GlyphGroupStruct	GlyphItemShortcut	EstructuraAcceso atravesado
GlyphGroupTemplate	GlyphItemPublic	TemplatePublic
GlyphGroupTemplate	GlyphItemInternal	TemplateInternal
GlyphGroupTemplate	GlyphItemFriend	TemplateInternal
GlyphGroupTemplate	GlyphItemProtected	TemplateProtected

GlyphGroupTemplate	GlyphItemPrivate	TemplatePrivate
GlyphGroupTemplate	GlyphItemShortcut	TemplateShortcut
GlyphGroupTypedef	GlyphItemPublic	TypeDefinitionPublic
GlyphGroupTypedef	GlyphItemInternal	TypeDefinitionInternal
GlyphGroupTypedef	GlyphItemFriend	TypeDefinitionInternal
GlyphGroupTypedef	GlyphItemProtected	TypeDefinitionProtected
GlyphGroupTypedef	GlyphItemPrivate	TypeDefinitionPrivate
GlyphGroupTypedef	GlyphItemShortcut	TypeDefinitionShortcut
GlyphGroupType	GlyphItemPublic	TypePublic
GlyphGroupType	GlyphItemInternal	TypeInternal
GlyphGroupType	GlyphItemFriend	TypeInternal
GlyphGroupType	GlyphItemProtected	TypeProtected
GlyphGroupType	GlyphItemPrivate	TypePrivate
GlyphGroupType	GlyphItemShortcut	TypeShortcut
GlyphGroupUnion	GlyphItemPublic	UnionPublic
GlyphGroupUnion	GlyphItemInternal	UnionInternal
GlyphGroupUnion	GlyphItemFriend	UnionInternal
GlyphGroupUnion	GlyphItemProtected	UnionProtected
GlyphGroupUnion	GlyphItemPrivate	UnionPrivate
GlyphGroupUnion	GlyphItemShortcut	UnionShortcut
GlyphGroupVariable	GlyphItemPublic	FieldPublic
GlyphGroupVariable	GlyphItemInternal	FieldInternal
GlyphGroupVariable	GlyphItemFriend	FieldInternal
GlyphGroupVariable	GlyphItemProtected	FieldProtected
GlyphGroupVariable	GlyphItemPrivate	FieldPrivate
GlyphGroupVariable	GlyphItemShortcut	FieldShortcut

GlyphGroupValueType	GlyphItemPublic	ValueTypePublic
GlyphGroupValueType	GlyphItemInternal	ValueTypeInternal
GlyphGroupValueType	GlyphItemFriend	ValueTypeInternal
GlyphGroupValueType	GlyphItemProtected	ValueTypeProtected
GlyphGroupValueType	GlyphItemPrivate	ValueTypePrivate
GlyphGroupValueType	GlyphItemShortcut	ValueTypeShortcut
GlyphGroupIntrinsic	GlyphItemPublic	ObjectPublic
GlyphGroupIntrinsic	GlyphItemInternal	ObjectInternal
GlyphGroupIntrinsic	GlyphItemFriend	ObjectInternal
GlyphGroupIntrinsic	GlyphItemProtected	ObjectProtected
GlyphGroupIntrinsic	GlyphItemPrivate	ObjectPrivate
GlyphGroupIntrinsic	GlyphItemShortcut	ObjectShortcut
GlyphGroupJSharpMethod	GlyphItemPublic	MethodPublic
GlyphGroupJSharpMethod	GlyphItemInternal	MethodInternal
GlyphGroupJSharpMethod	GlyphItemFriend	MethodInternal
GlyphGroupJSharpMethod	GlyphItemProtected	MethodProtected
GlyphGroupJSharpMethod	GlyphItemPrivate	MethodPrivate
GlyphGroupJSharpMethod	GlyphItemShortcut	MethodShortcut
GlyphGroupJSharpField	GlyphItemPublic	FieldPublic
GlyphGroupJSharpField	GlyphItemInternal	FieldInternal
GlyphGroupJSharpField	GlyphItemFriend	FieldInternal
GlyphGroupJSharpField	GlyphItemProtected	FieldProtected
GlyphGroupJSharpField	GlyphItemPrivate	FieldPrivate
GlyphGroupJSharpField	GlyphItemShortcut	FieldShortcut
GlyphGroupJSharpClass	GlyphItemPublic	ClassPublic
GlyphGroupJSharpClass	GlyphItemInternal	ClassInternal

GlyphGroupJSharpClass	GlyphItemFriend	ClassInternal
GlyphGroupJSharpClass	GlyphItemProtected	ClassProtected
GlyphGroupJSharpClass	GlyphItemPrivate	ClassPrivate
GlyphGroupJSharpClass	GlyphItemShortcut	ClassShortcut
GlyphGroupJSharpNamespace	GlyphItemPublic	NamespacePublic
GlyphGroupJSharpNamespace	GlyphItemInternal	NamespaceInternal
GlyphGroupJSharpNamespace	GlyphItemFriend	NamespaceInternal
GlyphGroupJSharpNamespace	GlyphItemProtected	NamespaceProtected
GlyphGroupJSharpNamespace	GlyphItemPrivate	NamespacePrivate
GlyphGroupJSharpNamespace	GlyphItemShortcut	NamespaceShortcut
GlyphGroupJSharpInterface	GlyphItemPublic	InterfacePublic
GlyphGroupJSharpInterface	GlyphItemInternal	InterfaceInternal
GlyphGroupJSharpInterface	GlyphItemFriend	InterfaceInternal
GlyphGroupJSharpInterface	GlyphItemProtected	InterfaceProtected
GlyphGroupJSharpInterface	GlyphItemPrivate	InterfacePrivate
GlyphGroupJSharpInterface	GlyphItemShortcut	InterfaceShortcut
GlyphGroupError		StatusError
GlyphBscFile		ClassFile
GlyphAssembly		Referencia
GlyphLibrary		Biblioteca
GlyphVBProject		VBProjectNode
GlyphCoolProject		CSPProjectNode
GlyphCppProject		CPPProjectNode
GlyphDialogId		Diálogo
GlyphOpenFolder		CarpetaAbierta
GlyphClosedFolder		CarpetaCerrada

GlyphArrow		GoToNext
GlyphCSharpFile		CSFileNode
GlyphCSharpExpansion		Fragmento de código
GlyphKeyword		IntellisenseKeyword
GlyphInformation		StatusInformation
GlyphReference		ClassMethodReference
GlyphRecursion		Recursividad
GlyphXmlItem		Etiqueta
GlyphJSharpProject		DocumentCollection
GlyphJSharpDocument		Documento
GlyphForwardType		GoToNext
GlyphCallersGraph		CallTo
GlyphCallGraph		Llamada
GlyphWarning		StatusWarning
GlyphMaybeReference		QuestionMark
GlyphMaybeCaller		CallTo
GlyphMaybeCall		Llamada
GlyphExtensionMethod		ExtensionMethod
GlyphExtensionMethodInternal		ExtensionMethod
GlyphExtensionMethodFriend		ExtensionMethod
GlyphExtensionMethodProtected		ExtensionMethod
GlyphExtensionMethodPrivate		ExtensionMethod
GlyphExtensionMethodShortcut		ExtensionMethod
GlyphXmlAttribute		XmlAttribute
GlyphXmlChild		XmlElement
GlyphXmlDescendant		XmlDescendant

GlyphXmlNamespace		XmlNamespace
GlyphXmlAttributeQuestion		XmlAttributeLowConfidence
GlyphXmlAttributeCheck		XmlAttributeHighConfidence
GlyphXmlChildQuestion		XmlElementLowConfidence
GlyphXmlChildCheck		XmlElementHighConfidence
GlyphXmlDescendantQuestion		XmlDescendantLowConfidence
GlyphXmlDescendantCheck		XmlDescendantHighConfidence
GlyphCompletionWarning		IntellisenseWarning

Uso y prestación de servicios

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un servicio es un contrato entre dos VSPackages. Un VSPackage ofrece un conjunto específico de interfaces para otro VSPackage para consumir. Por ejemplo, Visual Studio [SVsActivityLog](#) ofrece el servicio a cualquier VSPackage que cargue. Este servicio [IVsActivityLog](#) proporciona la interfaz, que se puede utilizar para escribir en el registro de actividad. Para obtener más información, consulte [Cómo: Usar el registro de actividad](#).

VSPackages puede ofrecer servicios propios [IProfferService](#) mediante el uso de la interfaz..

Visual Studio ofrece servicios importantes, como los siguientes:

SERVICIO IDE	DESCRIPCIÓN
SVsShell	Proporciona acceso a los servicios IDE que se ocupan de la funcionalidad básica, VSPackages y el registro.
SVsUIShell	Proporciona ventanas básicas y funcionalidad relacionada con la interfaz de usuario en el IDE, como la capacidad de crear herramientas y ventanas de documentos.
SVsSolution	Proporciona funcionalidad básica relacionada con la solución, como la capacidad de enumerar proyectos, crear nuevos proyectos y supervisar los cambios de proyecto.

En esta sección

- [Service Essentials](#) Presenta los elementos importantes de un servicio de Visual Studio.
- [Cómo: Obtener un servicio](#) Describe cómo solicitar (consumir) un servicio.
- [Cómo: Proporcionar un servicio](#) Describe cómo proporcionar un servicio.
- [Cómo: Proporcionar un servicio de Visual Studio asincrónico](#) Describe cómo proporcionar un servicio asincrónico.
- [Cómo: Solucionar problemas de servicios](#) Discute problemas comunes y les presenta soluciones.

Secciones relacionadas

- [Visual Studio SDK](#)

Cómo: Obtener un servicio

06/04/2020 • 5 minutes to read • [Edit Online](#)

A menudo necesita obtener servicios de Visual Studio para tener acceso a diferentes características. En general, un servicio de Visual Studio proporciona una o varias interfaces que puede usar. Puede obtener la mayoría de los servicios de un VSPackage.

Cualquier VSPackage que [Package](#) deriva y que se ha colocado correctamente puede solicitar cualquier servicio global. Dado `Package` que la `IServiceProvider` clase implementa, cualquier `Package` VSPackage que deriva también es un proveedor de servicios.

Cuando Visual Studio [Package](#) carga un `IServiceProvider`, pasa [SetSite](#) un objeto al método durante la inicialización. Esto se denomina *sentar* el VSPackage. La `Package` clase ajusta este proveedor [GetService](#) de servicios y proporciona el método para obtener servicios.

Obtener un servicio de un VSPackage inicializado

1. Cada extensión de Visual Studio comienza con un proyecto de implementación de VSIX, que contendrá los activos de extensión. Cree Visual Studio un proyecto `GetServiceExtension` VSIX denominado . Puede encontrar la plantilla de proyecto VSIX en el cuadro de diálogo **Nuevo proyecto** buscando "vsix".
2. Ahora agregue una plantilla de elemento de comando personalizada denominada `GetServiceCommand`. En el cuadro de diálogo **Agregar nuevo elemento**, vaya a **Extensibilidad de Visual C-** > y seleccione **Comando personalizado**. En el campo **Nombre** en la parte inferior de la ventana, cambie el nombre del archivo de comandos a `GetServiceCommand.cs`. Para obtener más información acerca de cómo crear un comando personalizado, [Crear una extensión con un comando](#) de menú
3. En `GetServiceCommand.cs`, quite `MenuItemCommand` el cuerpo del método y agregue el código siguiente:

```
IVsActivityLog activityLog = ServiceProvider.GetService(typeof(SVsActivityLog)) as IVsActivityLog;
if (activityLog == null) return;
System.Windows.Forms.MessageBox.Show("Found the activity log service.");
```

Este código obtiene un servicio `SVsActivityLog` y `IVsActivityLog` lo convierte en una interfaz, que se puede usar para escribir en el registro de actividad. Para obtener un ejemplo, consulte [Cómo: Usar el registro de actividad](#).

4. Compile la solución y comience la depuración. Aparece la instancia experimental.
5. En el menú **Herramientas** de la instancia experimental, busque el botón **Invocar GetServiceCommand**. Al hacer clic en este botón, debería ver un cuadro de mensaje que dice **Encontrado el servicio de registro de actividad**.

Obtener un servicio desde una ventana de herramientas o un contenedor de control

A veces es posible que necesite obtener un servicio desde una ventana de herramientas o un contenedor de control que no se ha ubicado, o bien se ha ubicado con un proveedor de servicios que no conoce el servicio que desea. Por ejemplo, es posible que desee escribir en el registro de actividad desde dentro de un control.

El [GetGlobalService](#) método estático se basa en un proveedor de servicios en caché [Package](#) que se inicializa la

primera vez que se encuentra cualquier VSPackage derivado de.

Dado que el VSPackage se llama antes de que el VSPackage esté sitio, los servicios globales normalmente no están disponibles desde dentro del constructor de VSPackage. Consulte [Cómo: Solucionar problemas de servicios](#) para obtener una solución alternativa.

Este es un ejemplo de la forma de obtener un servicio en una ventana de herramientas u otro elemento que no sea VSPackage.

```
IVsActivityLog log = Package.GetService(typeof(SVsActivityLog)) as IVsActivityLog;
if (log == null) return;
```

Obtener un servicio desde el objeto DTE

También puede obtener [DTEClass](#) servicios de objeto. Sin embargo, debe obtener el objeto DTE como un [GetGlobalService](#) servicio de un VSPackage o mediante una llamada al método estático.

El objeto DTE [IServiceProvider](#) implementa, que puede usar para [GetService](#) consultar un servicio mediante .

A continuación se muestra cómo obtener un servicio desde el objeto DTE.

```
// Start with the DTE object, for example:
// using EnvDTE;
// DTE dte = (DTE)GetService(typeof(DTE));

ServiceProvider sp = new ServiceProvider((Microsoft.VisualStudio.OLE.Interop.IServiceProvider)dte);
if (sp != null)
{
    IVsActivityLog log = sp.GetService(typeof(SVsActivityLog)) as IVsActivityLog;
    if (log != null)
    {
        System.Windows.Forms.MessageBox.Show("Found the activity log service.");
    }
}
```

Vea también

- [Cómo: Proporcionar un servicio](#)
- [Usar y proporcionar servicios](#)
- [Elementos esenciales del servicio](#)

Cómo: Proporcionar un servicio

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un VSPackage puede proporcionar servicios que otros VSPackages pueden usar. Para proporcionar un servicio, un VSPackage debe registrar el servicio con Visual Studio y agregar el servicio.

La [Package](#) clase implementa [IServiceProvider](#) [IServiceContainer](#) tanto y [IServiceContainer](#) contiene métodos de devolución de llamada que proporcionan servicios a petición.

Para obtener más información acerca de los servicios, consulte [Elementos esenciales](#) del servicio .

NOTE

Cuando un VSPackage está a punto de descargarse, Visual Studio espera hasta que se hayan entregado todas las solicitudes de servicios que proporciona un VSPackage. No permite nuevas solicitudes para estos servicios. No debe llamar explícitamente al [RevokeService](#) método para revocar un servicio al descargar.

Implementar un servicio

1. Crear un proyecto VSIX (**Archivo > Nuevo > proyecto > Proyecto > deextensibilidad > VSIX Project**).
2. Agregue un VSPackage al proyecto. Seleccione el nodo del proyecto en el **Explorador** de soluciones y haga clic en **Agregar > nuevo elemento > Paquete deextensibilidad > deelementos > deVisual Studio**.
3. Para implementar un servicio, debe crear tres tipos:
 - Interfaz que describe el servicio. Muchas de estas interfaces están vacías, es decir, no tienen métodos.
 - Interfaz que describe la interfaz de servicio. Esta interfaz incluye los métodos que se van a implementar.
 - Una clase que implementa el servicio y la interfaz de servicio.

En el ejemplo siguiente se muestra una implementación básica de los tres tipos. El constructor de la clase de servicio debe establecer el proveedor de servicios.

```

public class MyService : SMyService, IMyService
{
    private Microsoft.VisualStudio.OLE.Interop.IServiceProvider serviceProvider;
    private string myString;
    public MyService(Microsoft.VisualStudio.OLE.Interop.IServiceProvider sp)
    {
        Trace.WriteLine(
            "Constructing a new instance of MyService");
        serviceProvider = sp;
    }
    public void Hello()
    {
        myString = "hello";
    }
    public string Goodbye()
    {
        return "goodbye";
    }
}

public interface SMyService
{
}

public interface IMyService
{
    void Hello();
    string Goodbye();
}

```

Registrar un servicio

1. Para registrar un servicio, agregue el [ProvideServiceAttribute](#) VSPackage que proporciona el servicio. Este es un ejemplo:

```

[ProvideService(typeof(SMyService))]
[PackageRegistration(UseManagedResourcesOnly = true)]
[Guid(VSPackage1.PackageGuidString)]
public sealed class VSPackage1 : Package
{
    . . .
}

```

Este atributo `SMyService` se registra con Visual Studio.

NOTE

Para registrar un servicio que reemplaza otro servicio [ProvideServiceOverrideAttribute](#) con el mismo nombre, utilice el archivo . Tenga en cuenta que solo se permite una invalidación de un servicio.

Agregar un servicio

1. En el inicializador de VSPackage, agregue el servicio y agregue un método de devolución de llamada para crear los servicios. Aquí está el cambio [Initialize](#) a realizar en el método:

```

protected override void Initialize()
{
    ServiceCreatorCallback callback = new ServiceCreatorCallback(CreateService);

    ((IServiceContainer)this).AddService(typeof(SMyService), callback);
    . . .
}

```

2. Implemente el método de devolución de llamada, que debe crear y devolver el servicio, o null si no se

puede crear.

```
private object CreateService(IServiceContainer container, Type serviceType)
{
    if (typeof(SMyService) == serviceType)
        return new MyService(this);
    return null;
}
```

NOTE

Visual Studio puede rechazar una solicitud para proporcionar un servicio. Lo hace si otro VSPackage ya proporciona el servicio.

3. Ahora puede obtener el servicio y utilizar sus métodos. En el ejemplo siguiente se muestra el uso del servicio en el inicializador, pero puede obtener el servicio en cualquier lugar donde desee usar el servicio.

```
protected override void Initialize()
{
    ServiceCreatorCallback callback =new ServiceCreatorCallback(CreateService);

    ((IServiceContainer)this).AddService(typeof(SMyService), callback);

    MyService myService = (MyService) this.GetService(typeof(SMyService));
    myService.Hello();
    string helloString = myService.Goodbye();

    base.Initialize();
}
```

El valor `helloString` debe ser "Hola".

Vea también

- [Cómo: Obtener un servicio](#)
- [Usar y proporcionar servicios](#)
- [Elementos esenciales del servicio](#)

Cómo: Proporcionar un servicio asincrónico de Visual Studio

06/04/2020 • 9 minutes to read • [Edit Online](#)

Si desea obtener un servicio sin bloquear el subproceso de interfaz de usuario, debe crear un servicio asincrónico y cargar el paquete en un subproceso en segundo plano. Para este propósito, [AsyncPackage](#) puede usar [Package](#) un valor, en lugar de un, y agregar el servicio con los métodos asincrónicos especiales del paquete asincrónico.

Para obtener información acerca de cómo proporcionar servicios de Visual Studio sincrónicos, vea [Cómo: proporcionar un servicio](#).

Implementar un servicio asincrónico

1. Crear un proyecto VSIX (**Archivo > Nuevo > proyecto > Proyecto de VisualC- > Extensibility > VSIX Project**). Asigne al proyecto el nombre **TestAsync**.
2. Agregue un VSPackage al proyecto. Seleccione el nodo del proyecto en el **Explorador** de soluciones y haga clic en **Agregar > nuevo elemento > Paquete de extensibilidad > de elementos > de Visual Studio**. Asigne a este archivo el *nombre* `TestAsyncPackage.cs`.
3. En `TestAsyncPackage.cs`, cambie el `AsyncPackage` paquete `Package` para heredar de en lugar de:

```
public sealed class TestAsyncPackage : AsyncPackage
```

4. Para implementar un servicio, debe crear tres tipos:
 - Interfaz que identifica el servicio. Muchas de estas interfaces están vacías, es decir, no tienen métodos, ya que solo se utilizan para consultar el servicio.
 - Interfaz que describe la interfaz de servicio. Esta interfaz incluye los métodos que se van a implementar.
 - Una clase que implementa el servicio y la interfaz de servicio.
5. En el ejemplo siguiente se muestra una implementación muy básica de los tres tipos. El constructor de la clase de servicio debe establecer el proveedor de servicios. En este ejemplo, solo agregaremos el servicio al archivo de código del paquete.
6. Agregue las siguientes directivas using al archivo de paquete:

```
using System.Threading;  
using System.Threading.Tasks;  
using System.Runtime.CompilerServices;  
using System.IO;  
using Microsoft.VisualStudio.Threading;  
using IAsyncServiceProvider = Microsoft.VisualStudio.Shell.IAsyncServiceProvider;  
using Task = System.Threading.Tasks.Task;
```

7. Aquí está la implementación del servicio asincrónico. Tenga en cuenta que debe establecer el proveedor de servicios asincrónicos en lugar del proveedor de servicios sincrónico en el constructor:


```

public class TextWriterService : STextWriterService, ITextWriterService
{
    private IAsyncServiceProvider asyncServiceProvider;

    public TextWriterService(IAsyncServiceProvider provider)
    {
        // constructor should only be used for simple initialization
        // any usage of Visual Studio service, expensive background operations should happen in the
        // asynchronous InitializeAsync method for best performance
        asyncServiceProvider = provider;
    }

    public async Task InitializeAsync(CancellationTokens cancellationTokens)
    {
        await TaskScheduler.Default;
        // do background operations that involve IO or other async methods

        await ThreadHelper.JoinableTaskFactory.SwitchToMainThreadAsync(cancellationTokens);
        // query Visual Studio services on main thread unless they are documented as free threaded
        explicitly.
        // The reason for this is the final cast to service interface (such as IVsShell) may involve
        COM operations to add/release references.

        IVsShell vsShell = this.asyncServiceProvider.GetServiceAsync(typeof(SVsShell)) as IVsShell;
        // use Visual Studio services to continue initialization
    }

    public async Task WriteLineAsync(string path, string line)
    {
        StreamWriter writer = new StreamWriter(path);
        await writer.WriteLineAsync(line);
        writer.Close();
    }
}

public interface STextWriterService
{
}

public interface ITextWriterService
{
    System.Threading.Tasks.Task WriteLineAsync(string path, string line);
}

```

Registrar un servicio

Para registrar un servicio, agregue el [ProvideServiceAttribute](#) paquete que proporciona el servicio. A diferencia del registro de un servicio sincrónico, debe asegurarse de que tanto el paquete como el servicio admiten la carga asíncrona:

- Debe agregar el campo **AllowsBackgroundLoading** [PackageRegistrationAttribute](#) - true al para asegurarse de que el paquete se puede inicializar de forma asíncrona Para obtener más información acerca de PackageRegistrationAttribute, vea [Registrar y anular el registro de VSPackages](#).
- Debe agregar el campo **IsAsyncQueryable** [ProvideServiceAttribute](#) - true al para asegurarse de que la instancia de servicio se puede inicializar de forma asíncrona.

A continuación se `AsyncPackage` muestra un ejemplo de un registro de servicio asíncrono:

```
[ProvideService((typeof(STextWriterService)), IsAsyncQueryable = true)]
[ProvideAutoLoad(UIContextGuids80.SolutionExists, PackageAutoLoadFlags.BackgroundLoad)]
[PackageRegistration(UseManagedResourcesOnly = true, AllowsBackgroundLoading = true)]
[Guid(TestAsyncPackage.PackageGuidString)]
public sealed class TestAsyncPackage : AsyncPackage
{. . . }
```

Agregar un servicio

1. En *TestAsyncPackage.cs*, `Initialize()` quite el `InitializeAsync()` método e invalide el método. Agregue el servicio y agregue un método de devolución de llamada para crear los servicios. A continuación se muestra un ejemplo del inicializador asincrónico que agrega un servicio:

```
protected override async Task InitializeAsync(CancellationTokens cancellationTokens,
IPProgress<ServiceProgressData> progress)
{
    await base.InitializeAsync(cancellationTokens, progress);
    this.AddService(typeof(STextWriterService), CreateTextWriterService);
}
```

Para que este servicio sea visible fuera de este paquete, establezca el valor de marca de promoción en *true* como último parámetro: `this.AddService(typeof(STextWriterService), CreateTextWriterService, true);`

2. Agregue una referencia a *Microsoft.VisualStudio.Shell.Interop.14.0.DesignTime.dll*.
3. Implemente el método de devolución de llamada como un método asincrónico que crea y devuelve el servicio.

```
public async Task<object> CreateTextWriterService(IAsyncServiceContainer container, CancellationTokens
cancellationTokens, Type serviceType)
{
    TextWriterService service = new TextWriterService(this);
    await service.InitializeAsync(cancellationTokens);
    return service;
}
```

Usar un servicio

Ahora puede obtener el servicio y utilizar sus métodos.

1. Lo mostraremos en el inicializador, pero puede obtener el servicio en cualquier lugar que desee usar el servicio.

```
protected override async System.Threading.Tasks.Task InitializeAsync(CancellationTokens
cancellationTokens, IPProgress<ServiceProgressData> progress)
{
    await base.InitializeAsync(cancellationTokens, progress);
    this.AddService(typeof(STextWriterService), CreateTextWriterService);

    ITextWriterService textService = await this.GetServiceAsync(typeof(STextWriterService)) as
ITextWriterService;
    string userpath = @"C:\MyDir\MyFile.txt";
    await textService.WriteLineAsync(userpath, "this is a test");
}
```

No se olvide `userpath` de cambiar a un nombre de archivo y ruta que tiene sentido en su máquina!

2. Compile y ejecute el código. Cuando aparezca la instancia experimental de Visual Studio, abra una solución. Esto hace `AsyncPackage` que la carga automática. Cuando se haya ejecutado el inicializador, debe buscar un archivo en la ubicación especificada.

Usar un servicio asíncrono en un controlador de comandos

Este es un ejemplo de cómo utilizar un servicio asíncrono en un comando de menú. Puede usar el procedimiento que se muestra aquí para utilizar el servicio en otros métodos no asíncronos.

1. Agregue un comando de menú al proyecto. (En el **Explorador** de soluciones, seleccione el nodo del proyecto, haga clic con el botón derecho y seleccione **Agregar** > nuevo comando **personalizado de extensibilidad** > **de elementos** > .) Asigne al archivo de comandos el *nombre* `TestAsyncCommand.cs`.
2. La plantilla de comando personalizada `Initialize()` vuelve a agregar el método al archivo `TestAsyncPackage.cs` para inicializar el comando. En `Initialize()` el método, copie la línea que inicializa el comando. Debería ser parecido a este:

```
TestAsyncCommand.Initialize(this);
```

Mueva esta línea `InitializeAsync()` al método del archivo `AsyncPackageForService.cs`. Puesto que esto se encuentra en una inicialización asíncrona, debe `SwitchToMainThreadAsync` cambiar al subproceso principal antes de inicializar el comando mediante . Debería ser parecido a este:

```
protected override async System.Threading.Tasks.Task InitializeAsync(CancellationTokencancellationTokentoken, IProgress<ServiceProgressData> progress)
{
    await base.InitializeAsync(cancellationTokentoken, progress);
    this.AddService(typeof(STextWriterService), CreateTextWriterService);

    ITextWriterService textService =
        await this.GetServiceAsync(typeof(STextWriterService)) as ITextWriterService;

    string userpath = @"C:\MyDir\MyFile.txt";
    await textService.WriteLineAsync(userpath, "this is a test");

    await this.JoinableTaskFactory.SwitchToMainThreadAsync(cancellationTokentoken);
    TestAsyncCommand.Initialize(this);
}
```

3. Elimine `Initialize()` el método.
4. En el archivo `TestAsyncCommand.cs`, busque el `MenuItemCallback()` método. Elimine el cuerpo del método.
5. Agregue una directiva using:

```
using System.IO;
```

6. Agregue un método `UseTextWriterAsync()` asíncrono denominado , que obtiene el servicio y utiliza sus métodos:

```
private async System.Threading.Tasks.Task UseTextWriterAsync()
{
    // Query text writer service asynchronously to avoid a blocking call.
    ITextWriterService textService =
        await AsyncServiceProvider.GlobalProvider.GetServiceAsync(typeof(STextWriterService))
        as ITextWriterService;

    string userpath = @"C:\MyDir\MyFile.txt";
    await textService.WriteLineAsync(userpath, "this is a test");
}
```

7. Llame a este `MenuItemCallback()` método desde el método:

```
private void MenuItemCallback(object sender, EventArgs e)
{
    UseTextWriterAsync();
}
```

8. Compile la solución y comience la depuración. Cuando aparezca la instancia experimental de Visual Studio, vaya al menú **Herramientas** y busque el elemento de menú **Invocar TestAsyncCommand**. Al hacer clic en él, `TextWriterService` escribe en el archivo especificado. (No es necesario abrir una solución, porque invocar el comando también hace que el paquete se cargue.)

Vea también

- [Usar y proporcionar servicios](#)

Cómo: Solucionar problemas de servicios

06/04/2020 • 4 minutes to read • [Edit Online](#)

Hay varios problemas comunes que pueden ocurrir cuando intenta obtener un servicio:

- El servicio no Visual Studio está registrado con .
- El servicio se solicita por tipo de interfaz y no por tipo de servicio.
- El VSPackage que solicita el servicio no se ha sitio.
- Se utiliza el proveedor de servicios incorrecto.

Si no se puede obtener el [GetService](#) servicio solicitado, la llamada a devuelve null. Siempre debe probar si es null después de solicitar un servicio:

```
IVsActivityLog log =  
    GetService(typeof(SVsActivityLog)) as IVsActivityLog;  
if (log == null) return;
```

Para solucionar problemas de un servicio

1. Examine el registro del sistema para ver si el servicio se ha registrado correctamente. Para obtener más información, consulte [Cómo: proporcionar un servicio](#).

El siguiente fragmento de archivo *.reg* muestra cómo se puede registrar el servicio SVsTextManager:

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\<version number>\Services\{F5E7E71D-1401-11d1-883B-0000F87579D2}]  
@="{F5E7E720-1401-11d1-883B-0000F87579D2}"  
"Name"="SVsTextManager"
```

En el ejemplo anterior, el Visual Studio número de versión es la versión de , como 12.0 o 14.0, la clave .F5E7E71D-1401-11d1-883B-0000F87579D2 es el identificador de servicio (SID) del servicio, SVsTextManager, y el valor predeterminado de f5E7E720-1401-11d1-883B-0000F87579D2 es el GUID de paquete del administrador de texto VSPackage, que proporciona el servicio.

2. Utilice el tipo de servicio y no el tipo de interfaz cuando llame a GetService. Al solicitar un Visual Studio servicio [Package](#) de , extrae el GUID del tipo. No se encontrará un servicio si existen las siguientes condiciones:
 - a. Un tipo de interfaz se pasa a GetService en lugar del tipo de servicio.
 - b. No se asigna ningún GUID explícitamente a la interfaz. Por lo tanto, el sistema crea un GUID predeterminado para un objeto según sea necesario.
3. Asegúrese de que el VSPackage que solicita el servicio se ha sitio. Visual Studio sitios de un VSPackage después [Initialize](#) de construirlo y antes de llamar a .

Si tiene código en un constructor de VSPackage que `Initialize` necesita un servicio, muévelo al método.

4. Asegúrese de que está utilizando el proveedor de servicios correcto.

No todos los proveedores de servicios son iguales. El proveedor Visual Studio de servicios que pasa a una

ventana de herramientas difiere de la que pasa a un VSPackage. El proveedor de servicios [STrackSelection](#) de ventana de [SVsRunningDocumentTable](#) herramientas conoce , pero no conoce . Puede llamar [GetGlobalService](#) para obtener un proveedor de servicios VSPackage desde dentro de una ventana de herramientas.

Si una ventana de herramientas hospeda un control de usuario o cualquier otro contenedor de control, Visual Studio el modelo de componentes de Windows ubicará el contenedor y no tendrá acceso a ningún servicio. Puede llamar [GetGlobalService](#) para obtener un proveedor de servicios VSPackage desde dentro de un contenedor de control.

Vea también

- [Lista de servicios disponibles](#)
- [Usar y proporcionar servicios](#)
- [Elementos esenciales del servicio](#)

Administración de VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la mayoría de los casos, no es necesario preocuparse por la administración de VSPackages, ya que las plantillas de proyecto y elemento registran y cargan el paquete automáticamente. Sin embargo, en algunas circunstancias es posible que necesite aprender un poco más para administrar su paquete.

Utilice la instancia experimental

Para obtener más información sobre la instancia experimental, consulte [La instancia experimental](#).

Registrar y anular el registro de VSPackages

Para averiguar cómo registrar y anular el registro de VSPackages y otros tipos de extensión, vea Registrar y anular el registro de [VSPackages](#).

Cargar un VSPackage

VSPackages se pueden establecer en autoload cuando un GUID CMDUICONTEXT determinado está activado. Para obtener más información, vea [Cargar VSPackages](#).

Use AsyncPackage para cargar VSPackages en segundo plano

La `AsyncPackage` clase permite la carga de paquetes en un subproceso en segundo plano para una mejor capacidad de respuesta de la interfaz de usuario en Visual Studio. Para obtener más información, vea [Cómo: usar AsyncPackage para cargar VSPackages en segundo plano](#).

Contexto de interfaz de usuario basado en reglas para extensiones

Contextos de interfaz de usuario basados en reglas permite a los autores de extensiones definir las condiciones precisas en las que se activa un contexto de interfaz de usuario y se cargan vsPackages asociados. Para obtener más información, vea [Cómo: Usar el contexto de interfaz de usuario basado en reglas para extensiones](#) de Visual Studio .

Diagnosticar el rendimiento de la extensión

Las extensiones pueden afectar al rendimiento de la carga de la solución y el inicio. Obtenga información sobre cómo se calcula el impacto de la extensión de Visual Studio y cómo se puede analizar localmente para probar si una extensión puede mostrarse como una extensión que afecta al rendimiento. Para obtener más información, consulte [Cómo: diagnosticar el rendimiento de la extensión](#) .

Solución de problemas de VSPackages

Descubra las técnicas para solucionar problemas de VSPackages que no se cargan o experimentan errores: [Solucionar problemas de VSPackages](#)

Vea también

- [VSPackages](#)

Instancia experimental

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para proteger el entorno de desarrollo de Visual Studio de aplicaciones no probadas que podrían cambiarlo, VSSDK proporciona un espacio experimental que puede usar para experimentar. Las nuevas aplicaciones se desarrollan mediante Visual Studio como de costumbre, pero se ejecutan mediante esta instancia experimental.

Cada aplicación que tiene un paquete VSIX inicia la instancia experimental de Visual Studio en modo de depuración.

Si desea iniciar la instancia experimental de Visual Studio fuera de una solución específica, ejecute el siguiente comando en la ventana de comandos:

" Ruta de instalación de* <Visual Studio> * de la ruta de instalación de Visual Studio> de la ruta de instalación de Visual Studio .

NOTE

La instancia experimental se escribe `<version number>Exp` en `<version number>Exp_Config` el registro bajo el y nodos. Por ejemplo, el área de registro experimental de Visual Studio 2015 es

```
HKCU\Software\Microsoft\VisualStudio\14.0Exp y HKCU\Software\Microsoft\VisualStudio\14.0Exp_Config
```

Le recomendamos que ejecute la extensión en la instancia experimental mientras la está desarrollando. Al implementar la extensión, se ejecuta en la instancia de desarrollo. Para obtener más información sobre el registro de aplicaciones, vea Registro de [VSPackages](#).

Registrar y anular el registro de VSPackages

06/04/2020 • 4 minutes to read • [Edit Online](#)

Utilice atributos para registrar un VSPackage, pero

Registrar un VSPackage

Puede usar atributos para controlar el registro de VSPackages administrados. Toda la información de registro está contenida en un archivo *.pkgdef*. Para obtener más información sobre el registro basado en archivos, consulte [Utilidad CreatePkgDef](#).

El código siguiente muestra cómo usar los atributos de registro estándar para registrar el VSPackage.

```
[PackageRegistration(UseManagedResourcesOnly = true)]
[Guid("0B81D86C-0A85-4f30-9B26-DD2616447F95")]
public sealed class BasicPackage : Package
{
    // ...
}
```

Cancelar el registro de una extensión

Si ha estado experimentando con una gran cantidad de VSPackages diferentes y desea quitarlos de la instancia experimental, puede ejecutar el **comando Restablecer**. Busque **Restablecer la instancia experimental** de Visual Studio en la página de inicio del equipo o ejecute este comando desde la línea de comandos:

```
<location of Visual Studio 2015 install>\\"Microsoft Visual Studio
14.0\VSSDK\VisualStudioIntegration\Tools\Bin\CreateExpInstance.exe" /Reset /VSInstance=14.0 /RootSuffix=Exp
```

Si desea desinstalar una extensión que ha instalado en la instancia de desarrollo de Visual Studio, vaya a > **Extensiones y actualizaciones** de herramientas, busque la extensión y haga clic en **Desinstalar**.

Si por alguna razón ninguno de estos métodos se realiza correctamente al desinstalar la extensión, puede anular el registro del ensamblado VSPackage desde la línea de comandos de la siguiente manera:

```
<location of Visual Studio 2015 install>\\"Microsoft Visual Studio
14.0\VSSDK\VisualStudioIntegration\Tools\Bin\regpkg" /unregister <pathToVSPackage assembly>
```

Utilice un atributo de registro personalizado para registrar una extensión

En algunos casos, es posible que deba crear un nuevo atributo de registro para la extensión. Puede usar atributos de registro para agregar nuevas claves del Registro o para agregar nuevos valores a las claves existentes. El nuevo atributo [RegistrationAttribute](#) debe derivar de [Register](#), [Unregister](#) y debe invalidar los métodos y.

Creación de un atributo personalizado

El código siguiente muestra cómo crear un nuevo atributo de registro.

```
[AttributeUsage(AttributeTargets.Class, Inherited = true, AllowMultiple = false)]
public class CustomRegistrationAttribute : RegistrationAttribute
{
}
```

Se [AttributeUsageAttribute](#) utiliza en las clases de atributo para especificar el elemento de programa (clase, método, etc.) al que pertenece el atributo, si se puede utilizar más de una vez y si se puede heredar.

Crear una clave del Registro

En el código siguiente, el atributo personalizado crea una subclave **personalizada** bajo la clave para el VSPackage que se está registrando.

```
public override void Register(RegistrationAttribute.RegistrationContext context)
{
    Key packageKey = null;
    try
    {
        packageKey = context.CreateKey(@"Packages\" + context.ComponentType.GUID + @"\Custom");
        packageKey.SetValue("NewCustom", 1);
    }
    finally
    {
        if (packageKey != null)
            packageKey.Close();
    }
}

public override void Unregister(RegistrationContext context)
{
    context.RemoveKey(@"Packages\" + context.ComponentType.GUID + @"\Custom");
}
```

Cree un nuevo valor bajo una clave de registro existente

Puede agregar valores personalizados a una clave existente. El código siguiente muestra cómo agregar un nuevo valor a una clave de registro de VSPackage.

```
public override void Register(RegistrationAttribute.RegistrationContext context)
{
    Key packageKey = null;
    try
    {
        packageKey = context.CreateKey(@"Packages\" + context.ComponentType.GUID + @"\");
        packageKey.SetValue("NewCustom", 1);
    }
    finally
    {
        if (packageKey != null)
            packageKey.Close();
    }
}

public override void Unregister(RegistrationContext context)
{
    context.RemoveValue(@"Packages\" + context.ComponentType.GUID, "NewCustom");
}
```

Vea también

- [VSPackages](#)

Cargar VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

VSPackages se cargan en Visual Studio solo cuando se requiere su funcionalidad. Por ejemplo, un VSPackage se carga cuando Visual Studio usa un generador de proyectos o un servicio que implementa el VSPackage. Esta característica se denomina carga retrasada, que se utiliza siempre que sea posible para mejorar el rendimiento.

NOTE

Visual Studio puede determinar cierta información de VSPackage, como los comandos que ofrece un VSPackage, sin cargar el VSPackage.

VSPackages se puede establecer para cargar automáticamente en un contexto de interfaz de usuario (UI) determinado, por ejemplo, cuando una solución está abierta. El [ProvideAutoLoadAttribute](#) atributo establece este contexto.

Cargar automáticamente un VSPackage en un contexto específico

- Agregue `ProvideAutoLoad` el atributo a los atributos de VSPackage:

```
[DefaultRegistryRoot(@"Software\Microsoft\VisualStudio\14.0")]
[PackageRegistration(UseManagedResourcesOnly = true)]
[ProvideAutoLoad(UIContextGuids80.SolutionExists)]
[Guid("00000000-0000-0000-0000-000000000000")] // your specific package GUID
public class MyAutoloadedPackage : Package
{
    . . .
}
```

Consulte los campos [UIContextGuids80](#) enumerados de para obtener una lista de los contextos de interfaz de usuario y sus valores GUID.

- Establezca un punto [Initialize](#) de interrupción en el método.
- Compile el VSPackage e inicie la depuración.
- Cargue una solución o cree una.

El VSPackage se carga y se detiene en el punto de interrupción.

Forzar un VSPackage para cargar

En algunas circunstancias, un VSPackage puede tener que forzar otro VSPackage para cargar. Por ejemplo, un VSPackage ligero podría cargar un VSPackage más grande en un contexto que no está disponible como `CMDUIContext`.

Puede usar [LoadPackage](#) el método para forzar un VSPackage para cargar.

- Inserte este código [Initialize](#) en el método del VSPackage que obliga a otro VSPackage a cargar:

```
IVsShell shell = GetService(typeof(SVsShell)) as IVsShell;
if (shell == null) return;

IVsPackage package = null;
Guid PackageToBeLoadedGuid =
    new Guid(Microsoft.PackageToBeLoaded.GuidList.guidPackageToBeLoadedPkgString);
shell.LoadPackage(ref PackageToBeLoadedGuid, out package);
```

Cuando se inicializa el VSPackage, `PackageToBeLoaded` forzará a cargar.

Forzar la carga no se debe usar para la comunicación de VSPackage. Utilice [Usar y proporcionar servicios](#) en su lugar.

Vea también

- [VSPackages](#)

Cómo: Administrar varios subprocesos en código administrado

06/04/2020 • 3 minutes to read • [Edit Online](#)

Si tiene una extensión VSPackage administrada que llama a métodos asíncronos o tiene operaciones que se ejecutan en subprocesos que no sean el subproceso de interfaz de usuario de Visual Studio, debe seguir las instrucciones que se indican a continuación. Puede mantener la capacidad de respuesta del subproceso de interfaz de usuario porque no necesita esperar a que se complete el trabajo en otro subproceso. Puede hacer que el código sea más eficaz, ya que no tiene subprocesos adicionales que ocupa espacio de pila y puede hacer que sea más confiable y más fácil de depurar porque evita interbloqueos y bloqueos.

En general, puede cambiar del subproceso de interfaz de usuario a otro subproceso, o viceversa. Cuando se devuelve el método, el subproceso actual es el subproceso desde el que se llamó originalmente.

IMPORTANT

Las siguientes directrices usan [Microsoft.VisualStudio.Threading](#) las API del [JoinableTaskFactory](#) espacio de nombres, en particular, la clase. Las API de este espacio Visual Studio 2013 de nombres son nuevas en . Puede obtener una instancia [JoinableTaskFactory](#) de [ThreadHelper](#) `ThreadHelper.JoinableTaskFactory` a desde la propiedad .

Cambiar del subproceso de interfaz de usuario a un subproceso en segundo plano

1. Si está en el subproceso de interfaz de usuario y `Task.Run()` desea realizar un trabajo asíncrono en un subproceso en segundo plano, utilice:

```
await Task.Run(async delegate{
    // Now you're on a separate thread.
});
// Now you're back on the UI thread.
```

2. Si está en el subproceso de interfaz de usuario y desea bloquear sincrónicamente mientras realiza el trabajo en un subproceso en segundo plano, utilice la [TaskScheduler](#) propiedad `TaskScheduler.Default` dentro [Runde](#):

```
// using Microsoft.VisualStudio.Threading;
ThreadHelper.JoinableTaskFactory.Run(async delegate {
    await TaskScheduler.Default;
    // You're now on a separate thread.
    DoSomethingSynchronous();
    await OrSomethingAsynchronous();
});
```

Cambiar de un subproceso en segundo plano al subproceso de interfaz de usuario

1. Si está en un subproceso en segundo plano y desea [SwitchToMainThreadAsync](#) hacer algo en el subproceso

de interfaz de usuario, utilice:

```
// Switch to main thread
await ThreadHelper.JoinableTaskFactory.SwitchToMainThreadAsync();
```

Puede usar [SwitchToMainThreadAsync](#) el método para cambiar al subproceso de interfaz de usuario. Este método publica un mensaje en el subproceso de interfaz de usuario con la continuación del método asíncrono actual y también se comunica con el resto del marco de subprocesos para establecer la prioridad correcta y evitar interbloqueos.

Si el método de subproceso en segundo plano no es asíncrono `await` y no puede hacerlo asíncrono, puede seguir usando la sintaxis para cambiar al subproceso de interfaz de usuario ajustando el trabajo con [Run](#), como en este ejemplo:

```
ThreadHelper.JoinableTaskFactory.Run(async delegate {
    // Switch to main thread
    await ThreadHelper.JoinableTaskFactory.SwitchToMainThreadAsync();
    // Do your work on the main thread here.
});
```

Cómo: Usar AsyncPackage para cargar VSPackages en segundo plano

06/04/2020 • 7 minutes to read • [Edit Online](#)

Cargar e inicializar un paquete VS puede dar lugar a E/S de disco. Si dicha E/S se produce en el subproceso de interfaz de usuario, puede provocar problemas de capacidad de respuesta. Para solucionar este tema, Visual Studio [AsyncPackage](#) 2015 introdujo la clase que permite la carga de paquetes en un subproceso en segundo plano.

Crear un AsyncPackage

Puede empezar por crear un proyecto VSIX (**Proyecto** > **New** > **Project** > **VSIX** de extensión de proyecto de archivo de VisualC) > **Extensibility** > y agregar un VSPackage al proyecto (haga clic con el botón derecho en el proyecto y **agregar** > **nuevo** > **elemento** > de paquete de Visual Studio de la extensibilidad >). A continuación, puede crear los servicios y agregarlos al paquete.

1. Derive el [AsyncPackage](#) paquete de .
2. Si proporciona servicios cuyas consultas pueden hacer que el paquete se cargue:

Para indicar a Visual Studio que el paquete es seguro para [PackageRegistrationAttribute](#) la carga en segundo plano y para optar por este comportamiento, debe establecer **AllowsBackgroundLoading** propiedad true en el constructor de atributos.

```
[PackageRegistration(UseManagedResourcesOnly = true, AllowsBackgroundLoading = true)]
```

Para indicar a Visual Studio que es seguro crear instancias del [IsAsyncQueryable](#) servicio en [ProvideServiceAttribute](#) un subproceso en segundo plano, debe establecer la propiedad en true en el constructor.

```
[ProvideService(typeof(SMyTestService), IsAsyncQueryable = true)]
```

3. Si va a cargar a través de contextos de interfaz [ProvideAutoLoadAttribute](#) de usuario, debe especificar **PackageAutoLoadFlags.BackgroundLoad** para el valor OR (0x2) en las marcas escritas como el valor de la entrada de carga automática del paquete.

```
[ProvideAutoLoad(UIContextGuid, PackageAutoLoadFlags.BackgroundLoad)]
```

4. Si tiene que realizar un trabajo [InitializeAsync](#) de inicialización asincrónica, debe invalidar . Quite `Initialize()` el método proporcionado por la plantilla VSIX. (El `Initialize()` método de **AsyncPackage** está sellado). Puede usar cualquiera [AddService](#) de los métodos para agregar servicios asincrónicos al paquete.

NOTA: Para `base.InitializeAsync()` llamar a , puede cambiar el código fuente a:

```
await base.InitializeAsync(cancellationToken, progress);
```

5. Debe tener cuidado de NO realizar RPC (llamada a procedimiento remoto) a partir del código de inicialización asincrónico (en `InitializeAsync`). Estos pueden ocurrir `GetService` cuando se llama directa o indirectamente. Cuando se requieren cargas de sincronización, el subproceso de interfaz de usuario bloqueará mediante `JoinableTaskFactory`. El modelo de bloqueo predeterminado deshabilita las RPC. Esto significa que si intenta usar un RPC de sus tareas asincrónicas, interbloqueo si el subproceso de interfaz de usuario está esperando a que se cargue el paquete. La alternativa general es calcular las referencias del código al subproceso de `SwitchToMainThreadAsync` interfaz de usuario si es necesario mediante algo como `Unable Task Factory's` o algún otro mecanismo que no use un RPC. NO use `ThreadHelper.Generic.Invoke` o, por lo general, bloquee el subproceso que realiza la llamada en espera para llegar al subproceso de interfaz de usuario.

NOTA: Debe evitar el uso de `InitializeAsync` `GetService` o `QueryService` en el método. Si tiene que usarlos, primero tendrá que cambiar al subproceso de interfaz de usuario. La alternativa es `GetServiceAsync(Type)` usar desde el `AsyncPackage` (mediante la conversión a `IAsyncServiceProvider`.)

C:: Cree un `AsyncPackage`:

```
[PackageRegistration(UseManagedResourcesOnly = true, AllowsBackgroundLoading = true)]
[ProvideService(typeof(SMyTestService), IsAsyncQueryable = true)]
public sealed class TestPackage : AsyncPackage
{
    protected override Task InitializeAsync(System.Threading.CancellationToken cancellationToken,
        IProgress<ServiceProgressData> progress)
    {
        this.AddService(typeof(SMyTestService), CreateService, true);
        return Task.FromResult<object>(null);
    }
}
```

Convertir un `VSPackage` existente en `AsyncPackage`

La mayor parte del trabajo es lo mismo que crear un nuevo `AsyncPackage`. Siga los pasos del 1 al 5 anteriores. También debe tener más cuidado con las siguientes recomendaciones:

1. Recuerde quitar `Initialize` la anulación que tenía en el paquete.
2. Evitar interbloqueos: podría haber RPC ocultos en el código. que ahora suceden en un hilo en segundo plano. Asegúrese de que si va a realizar un RPC (por ejemplo, `GetService`), debe (1) cambiar al subproceso principal o (2) usar la versión asincrónica de la API si existe (por ejemplo, `GetServiceAsync`).
3. No cambie entre roscas con demasiada frecuencia. Intente localizar el trabajo que puede ocurrir en un subproceso en segundo plano para reducir el tiempo de carga.

Consulta de servicios desde `AsyncPackage`

Un `AsyncPackage` puede o no cargar de forma asincrónica dependiendo del llamador. Por ejemplo,

- Si el autor de la llamada llamó a `GetService` o `QueryService` (ambas API sincrónicas) o
- Si el llamador llamó a `IVsShell::LoadPackage` (o `IVsShell5::LoadPackageWithContext`) o
- La carga se desencadena mediante un contexto de interfaz de usuario, pero no se especificó que el mecanismo de contexto de interfaz de usuario puede cargar de forma asincrónica

a continuación, el paquete se cargará sincrónicamente.

El paquete todavía tiene una oportunidad (en su fase de inicialización asincrónica) para realizar el trabajo fuera del subproceso de interfaz de usuario, aunque el subproceso de interfaz de usuario se bloqueará para

la finalización de ese trabajo. Si el autor de la llamada usa **IAsyncServiceProvider** para consultar de forma asincrónica el servicio, la carga y la inicialización se realizarán de forma asincrónica suponiendo que no se bloqueen inmediatamente en el objeto de tarea resultante.

C: Cómo consultar el servicio de forma asincrónica:

```
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.Shell.Interop;

IAsyncServiceProvider asyncServiceProvider = Package.GetService(typeof(SAsyncServiceProvider)) as
IAsyncServiceProvider;
IMyTestService testService = await asyncServiceProvider.GetServiceAsync(typeof(SMyTestService)) as
IMyTestService;
```

Cómo: Usar el contexto de interfaz de usuario basado en reglas para las extensiones de Visual Studio

06/04/2020 • 16 minutes to read • [Edit Online](#)

Visual Studio permite cargar VSPackages cuando [UIContext](#) se activan ciertas s conocidas. Sin embargo, estos contextos de interfaz de usuario no son de grano fino, lo que deja a los autores de la extensión no hay más opción que elegir un contexto de interfaz de usuario disponible que se activa antes del punto que realmente quería que el VSPackage cargar. Para obtener una lista de contextos [KnownUIContexts](#) de interfaz de usuario conocidos, vea .

La carga de paquetes puede tener un impacto en el rendimiento y cargarlos antes de lo necesario no es la mejor práctica. Visual Studio 2015 introdujo el concepto de contextos de interfaz de usuario basados en reglas, un mecanismo que permite a los autores de extensiones definir las condiciones precisas en las que se activa un contexto de interfaz de usuario y se cargan los VSPackages asociados.

Contexto de interfaz de usuario basado en reglas

Una "Regla" consta de un nuevo contexto de interfaz de usuario (un GUID) y una expresión booleana que hace referencia a uno o más "Términos" combinados con operaciones lógicas "y", "o", "no". Los "Términos" se evalúan dinámicamente en tiempo de ejecución y la expresión se vuelve a evaluar cada vez que cambia cualquiera de sus términos. Cuando la expresión se evalúa como true, se activa el contexto de interfaz de usuario asociado. De lo contrario, el contexto de la interfaz de usuario se desactiva.

El contexto de interfaz de usuario basado en reglas se puede usar de varias maneras:

1. Especifique restricciones de visibilidad para comandos y ventanas de herramientas. Puede ocultar las ventanas de comandos/herramientas hasta que se cumpla la regla de contexto de interfaz de usuario.
2. Como restricciones de carga automática: paquetes de carga automática solo cuando se cumple la regla.
3. Como tarea retrasada: retrasar la carga hasta que haya pasado un intervalo especificado y se siga cumpliendo la regla.

El mecanismo puede ser utilizado por cualquier extensión de Visual Studio.

Crear un contexto de interfaz de usuario basado en reglas

Supongamos que tiene una extensión denominada TestPackage, que ofrece un comando de menú que solo se aplica a los archivos con extensión *.config*. Antes de VS2015, la mejor opción [SolutionExistsAndFullyLoadedContext](#) era cargar TestPackage cuando se activó el contexto de interfaz de usuario. Cargar TestPackage de esta manera no es eficaz, ya que la solución cargada ni siquiera puede contener un archivo *.config*. Estos pasos muestran cómo se puede usar el contexto de interfaz de usuario basado en reglas para activar un contexto de interfaz de usuario solo cuando se selecciona un archivo con extensión *.config* y cargar TestPackage cuando se activa ese contexto de interfaz de usuario.

1. Defina un nuevo GUID UIContext y [ProvideAutoLoadAttribute](#) agréguelo a la clase VSPackage y [ProvideUIContextRuleAttribute](#).

Por ejemplo, supongamos que se va a agregar un nuevo UIContext "UIContextGuid". El GUID creado (puede crear un GUID haciendo clic en **Herramientas** > crearGUID) es "8B40D5E2-5626-42AE-99EF-3DD1EFF46E7B". A continuación, agregue la siguiente declaración dentro de la clase de paquete:

```
public const string UIContextGuid = "8B40D5E2-5626-42AE-99EF-3DD1EFF46E7B";
```

Para los atributos, agregue los siguientes valores: (Los detalles de estos atributos se explicarán más adelante)

```
[ProvideAutoLoad(TestPackage.UIContextGuid)]  
[ProvideUIContextRule(TestPackage.UIContextGuid,  
    name: "Test auto load",  
    expression: "DotConfig",  
    termNames: new[] { "DotConfig" },  
    termValues: new[] { "HierSingleSelectionName:.config$" })]
```

Estos metadatos definen el nuevo GUID UIContext (8B40D5E2-5626-42AE-99EF-3DD1EFF46E7B) y una expresión que hace referencia a un único término, "DotConfig". El término "DotConfig" se evalúa como true cada vez que la selección actual en la jerarquía activa tiene un nombre que coincide con el patrón de expresión regular ".config\$" (termina con *.config*). El valor (Predeterminado) define un nombre opcional para la regla útil para la depuración.

Los valores del atributo se agregan a pkgdef generado durante el tiempo de compilación después.

2. En el archivo VSCT para los comandos de TestPackage, agregue el indicador "DynamicVisibility" a los comandos adecuados:

```
<CommandFlag>DynamicVisibility</CommandFlag>
```

3. En la sección Visibilities del VSCT, vincule los comandos adecuados al nuevo GUID de UIContext definido en #1:

```
<VisibilityConstraints>  
    <VisibilityItem guid="guidTestPackageCmdSet" id="TestId" context="UIContextGuid"/>  
</VisibilityConstraints>
```

4. En la sección Símbolos, agregue la definición de UIContext:

```
<GuidSymbol name="UIContextGuid" value="{8B40D5E2-5626-42AE-99EF-3DD1EFF46E7B}" />
```

Ahora, los comandos * ** de menú contextual para los archivos .config solo estarán visibles cuando el elemento seleccionado en el explorador de soluciones sea un archivo *.config* y el paquete no se cargará hasta que se seleccione uno de esos comandos.

A continuación, use un depurador para confirmar que el paquete se carga solo cuando lo espera. Para depurar TestPackage:

5. Establezca un punto [Initialize](#) de interrupción en el método.
6. Compile TestPackage e inicie la depuración.
7. Cree un proyecto o abra uno.
8. Seleccione cualquier archivo con una extensión que no sea *.config*. El punto de interrupción no debe ser alcanzado.
9. Seleccione el archivo *App.Config*.

El TestPackage carga y se detiene en el punto de interrupción.

Agregar más reglas para el contexto de la interfaz de usuario

Dado que las reglas de contexto de interfaz de usuario son expresiones booleanas, puede agregar reglas más restringidas para un contexto de interfaz de usuario. Por ejemplo, en el contexto de interfaz de usuario anterior, puede especificar que la regla solo se aplica cuando se carga una solución con un proyecto. De esta manera, los comandos no aparecerán si abre un archivo *.config* como un archivo independiente, no como parte de un proyecto.

```
[ProvideAutoLoad(TestPackage.UIContextGuid)]
[ProvideUIContextRule(TestPackage.UIContextGuid,
    name: "Test auto load",
    expression: "(SingleProject | MultipleProjects) & DotConfig",
    termNames: new[] { "SingleProject", "MultipleProjects", "DotConfig" },
    termValues: new[] { VSConstants.UICONTEXT_SolutionHasSingleProject_string ,
VSConstants.UICONTEXT_SolutionHasMultipleProjects_string , "HierSingleSelectionName:.config$" }]]
```

Ahora la expresión hace referencia a tres términos. Los dos primeros términos, "SingleProject" y "MultipleProjects", hacen referencia a otros contextos de interfaz de usuario conocidos (por sus GUID). El tercer término, "DotConfig" es el contexto de interfaz de usuario basado en reglas definido anteriormente en este artículo.

Activación retrasada

Las reglas pueden tener un "Retraso" opcional. El retardo se especifica en milisegundos. Si está presente, el retraso hace que la activación o desactivación del contexto de interfaz de usuario de una regla se retrase en ese intervalo de tiempo. Si la regla vuelve a cambiar antes del intervalo de retardo, no sucede nada. Este mecanismo se puede utilizar para "escalonar" los pasos de inicialización, especialmente la inicialización de una sola vez sin depender de temporizadores o registrarse para notificaciones inactivas.

Por ejemplo, puede especificar la regla de carga de prueba para que tenga un retraso de 100 milisegundos:

```
[ProvideAutoLoad(TestPackage.UIContextGuid)]
[ProvideUIContextRule(TestPackage.UIContextGuid,
    name: "Test auto load",
    expression: "DotConfig",
    termNames: new[] { "DotConfig" },
    termValues: new[] { "HierSingleSelectionName:.config$" },
    delay: 100)]
```

Tipos de términos

Aquí están los diversos tipos de término que se soportan:

TÉRMINO	DESCRIPCIÓN
Nnnnnnnnnnn-nnnn-nnnn-nnnnnnnnnnnnnnnnnnnnnnnnn.	El GUID hace referencia a un contexto de interfaz de usuario. El término será true siempre que el contexto de la interfaz de usuario esté activo y false en caso contrario.
HierSingleSelectionName:<> de patrón	El término será true siempre que la selección en la jerarquía activa sea un único elemento y el nombre del elemento seleccionado coincida con la expresión regular .Net dada por "patrón".

TÉRMINO	DESCRIPCIÓN
UserSettingsStoreQuery:<> de consulta	"consulta" representa una ruta de acceso completa en el almacén de configuración de usuario, que debe evaluarse como un valor distinto de cero. La consulta se divide en una "colección" y "propertyName" en la última barra diagonal.
ConfigSettingsStoreQuery:<> de consulta	"consulta" representa una ruta de acceso completa en el almacén de configuración de configuración, que debe evaluarse como un valor distinto de cero. La consulta se divide en una "colección" y "propertyName" en la última barra diagonal.
ActiveProjectFlavor:<projectTypeGuid>	El término será true siempre que el proyecto seleccionado tenga sabor (agregado) y tenga un sabor que coincida con el GUID de tipo de proyecto especificado.
ActiveEditorContentType:<contentType>	El término será true cuando el documento seleccionado sea un editor de texto con el tipo de contenido especificado. Nota: cuando se cambia el nombre del documento seleccionado, este término no se actualiza hasta que el archivo se cierra y se vuelve a abrir.
ActiveProjectCapability:<expresión>	El término es true cuando las capacidades activas del proyecto coinciden con la expresión proporcionada. Una expresión puede ser algo así como VB CSharp.
SolutionHasProjectCapability:<> de expresión	Similar a anterior, pero term es true cuando la solución tiene cualquier proyecto cargado que coincida con la expresión.
SolutionHasProjectFlavor:<projectTypeGuid>	El término será true siempre que una solución tenga un proyecto con sabor (agregado) y tenga un sabor que coincida con el GUID de tipo de proyecto especificado.
ProjectAddedItem:<> de patrón	El término es true cuando se agrega un archivo que coincide con el "patrón" a un proyecto en el solución que se abre.
ActiveProjectOutputType:<outputType>	El término es true cuando el tipo de salida para el proyecto activo coincide exactamente. El outputType podría ser __VSPROJOUTPUTTYPE un entero o un tipo.
ActiveProjectBuildProperty:<> de <compilación de la> de regex	El término es true cuando el proyecto activo tiene la propiedad de compilación especificada y el valor de propiedad coincide con el filtro regex proporcionado. Consulte Persisting Data in MSBuild Project Files para obtener más detalles sobre las propiedades de compilación.
SolutionHasProjectBuildProperty:<buildProperty><> á regex>	El término es true cuando la solución tiene un proyecto cargado con la propiedad de compilación especificada y el valor de propiedad coincide con el filtro regex proporcionado.

Compatibilidad con la extensión entre versiones

Contextos de interfaz de usuario basados en reglas es una nueva característica de Visual Studio 2015 y no se portaría a versiones anteriores. Al no migrar a versiones anteriores, se crea un problema con las extensiones o paquetes que tienen como destino varias versiones de Visual Studio. Esas versiones tendrían que cargarse automáticamente en Visual Studio 2013 y versiones anteriores, pero pueden beneficiarse de los contextos de

interfaz de usuario basados en reglas para evitar que se carguen automáticamente en Visual Studio 2015.

Para admitir estos paquetes, las entradas de `AutoLoadPackages` en el registro ahora pueden proporcionar una marca en su campo `value` para indicar que la entrada debe omitirse en Visual Studio 2015 y versiones posteriores. Esto se puede hacer agregando [PackageAutoLoadFlags](#) una opción de marcas a `.VSPackages` ahora puede agregar `SkipWhenUIContextRulesActive` opción a su [ProvideAutoLoadAttribute](#) atributo para indicar que la entrada debe omitirse en Visual Studio 2015 y versiones posteriores.

Reglas de contexto de interfaz de usuario extensible

A veces, los paquetes no pueden usar reglas de contexto de interfaz de usuario estáticas. Por ejemplo, supongamos que tiene un paquete que admite la extensibilidad, de forma que el estado del comando se basa en tipos de editor admitidos por los proveedores MEF importados. El comando está habilitado si hay una extensión que admita el tipo de edición actual. En tales casos, el propio paquete no puede usar una regla de contexto de interfaz de usuario estática, ya que los términos cambiarían en función de las extensiones MEF disponibles.

Para admitir estos paquetes, los contextos de interfaz de usuario basados en reglas admiten una expresión codificada de forma rígida `"*"` que indica que todos los términos siguientes se unirán con OR. Esto permite que el paquete maestro defina un contexto de interfaz de usuario basado en reglas conocido y vincule su estado de comando a este contexto. Después, cualquier extensión MEF destinada al paquete maestro puede agregar sus términos para los editores que admite sin afectar a otros términos o a la expresión maestra.

La [ProvideExtensibleUIContextRuleAttribute](#) documentación del constructor muestra la sintaxis de las reglas de contexto de interfaz de usuario extensible.

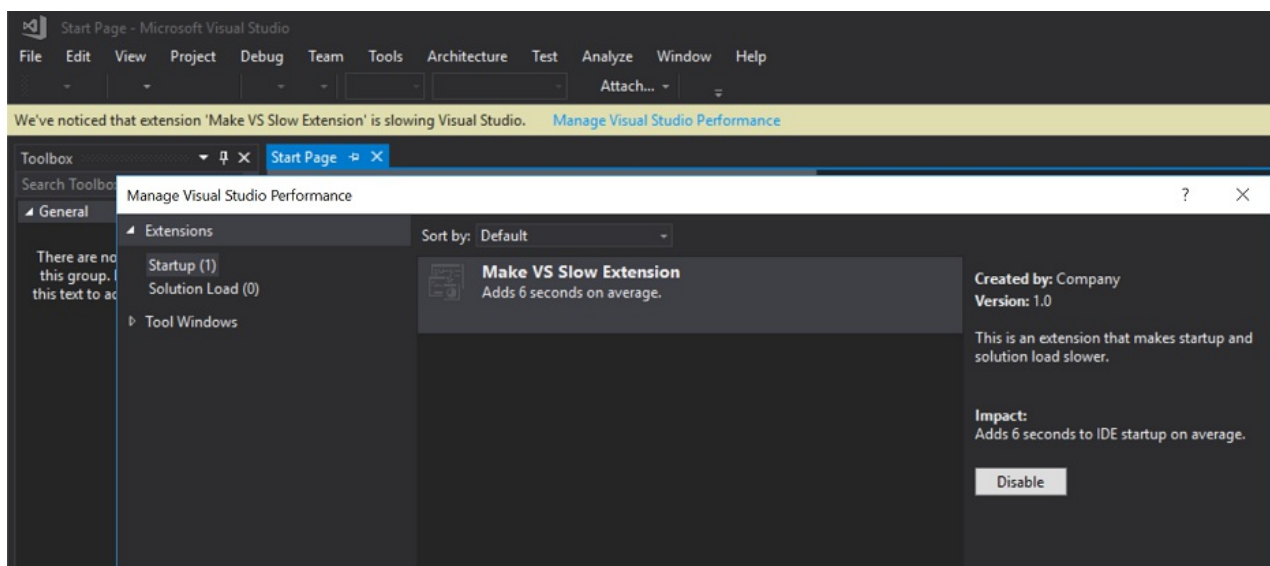
Medir el impacto de la extensión de inicio

02/12/2019 • 20 minutes to read • [Edit Online](#)

Centrarse en el rendimiento de la extensión en Visual Studio 2017

Según los comentarios recibidos, una de las áreas de enfoque para Visual Studio 2017 versión ha sido rendimiento de carga de inicio y la solución. Como el equipo de plataforma de Visual Studio, hemos estado trabajando en mejorar el rendimiento de carga de inicio y la solución. En general, nuestras mediciones sugieren las extensiones instaladas también pueden tener una repercusión importante en estos escenarios.

Para ayudar a los usuarios a comprender mejor este efecto, hemos agregado una característica nueva en Visual Studio para notificar a los usuarios de las extensiones de baja velocidad. A veces, Visual Studio detecta una nueva extensión que se ralentiza la carga de solución o de inicio. Cuando se detecta una ralentización, los usuarios verán una notificación en el IDE que les dirija al nuevo cuadro de diálogo "Administrar el rendimiento de Visual Studio". Este cuadro de diálogo también siempre se puede acceder al menú Ayuda para buscar extensiones detectadas previamente.



El objetivo de este documento es ayudar a los desarrolladores de extensiones con la descripción de cómo se calcula el impacto de la extensión. Este documento también describe cómo se puede analizar el impacto de extensión localmente. Localmente análisis del impacto de la extensión determinará si una extensión se muestren como un rendimiento que afectan a la extensión.

NOTE

Este documento se centra en el impacto de las extensiones de carga de inicio y la solución. Las extensiones también afectan el rendimiento de Visual Studio cuando hacen que la interfaz de usuario deje de responder. Para obtener más información sobre este tema, consulte [Cómo: Interfaz de usuario de diagnosticar retrasos causados por las extensiones](#).

Cómo las extensiones pueden afectar al inicio

Una de las maneras más comunes para las extensiones afectar al rendimiento de inicio es mediante la carga automática en uno de los contextos de interfaz de usuario de inicio conocidos como NoSolutionExists o ShellInitialized. Activar estos contextos de interfaz de usuario durante el inicio. Todos los paquetes que incluyen el `ProvideAutoLoad` atributo en su definición de estos contextos se cargarán y se inicializan en ese momento.

Cuando se mide el impacto de una extensión, nos centramos principalmente en el tiempo invertido por las extensiones que elija para la carga automática en los contextos anteriores. Mide podrían incluir pero no limitarse a veces:

- Carga de ensamblados de extensión para los paquetes sincrónicos
- Tiempo invertido en el constructor de clase de paquete para paquetes sincrónicos
- Tiempo invertido en el método de inicialización (o SetSite) de paquete para los paquetes sincrónicos
- Para paquetes asincrónicos, las operaciones anteriores se ejecutan en el subproceso en segundo plano. Por lo tanto, las operaciones se excluyen de la supervisión.
- Tiempo invertido en ningún trabajo asincrónico programado durante la inicialización del paquete se ejecute en el subproceso principal
- Tiempo invertido en controladores de eventos, específicamente la activación de contexto shell inicializado o el cambio de estado inerte shell
- A partir de Visual Studio 2017 Update 3, también comenzaremos a supervisar el tiempo invertido en llamadas inactivo antes de inicializa el shell de. Las operaciones largas en inactividad controladores también provocar IDE no responde y contribuyen al tiempo de inicio percibido por el usuario.

Hemos agregado muchas características a partir de Visual Studio 2015. Estas características ayudan con la eliminación de la necesidad de paquetes de carga automático. Las características también posponer la necesidad de paquetes para cargar los casos más específicas. Estos casos incluyen ejemplos donde los usuarios podrían más determinados usar la extensión o reducir el impacto de una extensión cuando se cargan automáticamente.

Puede encontrar más detalles sobre estas características en los siguientes documentos:

Contextos de interfaz de usuario basada en reglas: Un motor más completo de están basadas en reglas creado en torno a los contextos de interfaz de usuario permite crear contextos personalizados basadas en atributos, tipos y tipos de proyecto. Contextos personalizados pueden utilizarse para cargar un paquete durante escenarios más específicos. Estos escenarios específicos incluyen la presencia de un proyecto con una función específica en lugar de inicio. También permiten contextos personalizados [comando visibilidad esté vinculada a un contexto personalizado](#) en función de los componentes del proyecto u otros términos disponibles. Esta característica elimina la necesidad de cargar un paquete para registrar un controlador de consulta de estado de comandos.

Compatibilidad con paquetes asincrónica: La nueva clase de base de AsyncPackage en Visual Studio 2015 permite que los paquetes de Visual Studio que se cargue en segundo plano forma asincrónica si se solicitó la carga del paquete mediante un atributo de la carga automática o una consulta de servicio asincrónico. Esta carga en segundo plano permite que el IDE a fin de responder. El IDE está respondiendo incluso mientras se inicializa la extensión en segundo plano y no se verá afectados escenarios críticos, como la carga de inicio y la solución.

Servicios asincrónicos: Con compatibilidad asincrónica de paquete, también hemos agregado compatibilidad para consultar los servicios de forma asincrónica y poder registrar los servicios asincrónicos. Más importante aún estamos trabajando de conversión de los servicios de Visual Studio principales para admitir la consulta asincrónica para que se produzca la mayoría del trabajo en una consulta de async en subprocesos en segundo plano. SComponentModel (host de MEF de Visual Studio) es uno de los servicios principales que ahora es compatible con una consulta asincrónica para permitir las extensiones admitir la carga asincrónica por completo.

Reducir el impacto de auto extensiones cargadas

Si todavía necesita un paquete se carga al inicio de automáticamente, es importante minimizar el trabajo realizado durante la inicialización del paquete. Minimizar el trabajo de inicialización del paquete reduce las posibilidades de la extensión que afectan a inicio.

Algunos ejemplos que podrían provocar que la inicialización costoso paquete son:

Uso de la carga del paquete sincrónico en lugar de la carga del paquete asincrónica

Porque los paquetes sincrónicos se cargan en el subproceso principal de forma predeterminada, le animamos a

los propietarios de extensión que tienen paquetes de carga automáticamente para usar la clase base paquete asincrónica en su lugar como se mencionó anteriormente. Cambiar un paquete de carga automática para admitir cargas asincrónicas también resultará más fácil de resolver los problemas siguientes.

Solicitudes de E/S de archivo sincrónicas/red

Lo ideal es que se debe evitar cualquier solicitud de E/S de archivos o red sincrónico en el subproceso principal. Su impacto dependerá de estado de la máquina y se puede bloquear durante largos períodos de tiempo en algunos casos.

Utilizando la carga del paquete asincrónica y las API de E/S asincrónicas debe asegurarse que la inicialización del paquete no bloquea el subproceso principal en estos casos. Los usuarios también pueden continuar interactuar con Visual Studio, mientras que las solicitudes de E/S que se producen en segundo plano.

Inicialización temprana de servicios, componentes

Uno de los patrones comunes de inicialización del paquete consiste en inicializar los servicios proporcionados por ese paquete en el paquete o utilizado por `constructor` o `initialize` método. Aunque esto garantiza que los servicios están listos para usarse, también puede agregar costos innecesarios para empaquetar la carga si esos servicios no se utilizan inmediatamente. En su lugar se deben inicializar dichos servicios a petición para minimizar el trabajo realizado en la inicialización del paquete.

Proporcionado por un paquete de servicios globales, puede usar `AddService` métodos que toman una función para inicializar el servicio de forma diferida solo cuando se solicita un componente. Para los servicios utilizados dentro del paquete, puede usar `Lazy<T>` o elemento `AsyncLazy<T>` para asegurarse de que los servicios son inicializado ni consultarse en el primer uso.

Medición del impacto de auto carga extensiones mediante el registro de actividad

A partir de Visual Studio 2017 Update 3, registro de actividad de Visual Studio ahora contendrá las entradas de impacto en el rendimiento de los paquetes durante la carga de inicio y la solución. Para ver estas mediciones, tiene que abrir Visual Studio con el modificador/log y abra *ActivityLog.xml* archivo.

En el registro de actividad, las entradas se encontrarán en el origen de "Administrar el rendimiento de Visual Studio" y tendrá un aspecto similar al ejemplo siguiente:

```
Component: 3cd7f5bf-6662-4ff0-ade8-97b5ff12f39c, Inclusive Cost: 2008.9381, Exclusive Cost: 2008.9381, Top Level Inclusive Cost: 2008.9381
```

En este ejemplo muestra que un paquete con el GUID "3cd7f5bf-6662-4ff0-ade8-97b5ff12f39c" dedicado a 2008 ms en el inicio de Visual Studio. Tenga en cuenta que Visual Studio tiene en cuenta el costo de nivel superior como el número principal al calcular el impacto de un paquete, ya que sería que verán los usuarios de ahorro cuando deshabilita la extensión de ese paquete.

Medición del impacto de auto carga extensiones mediante PerfView

Mientras el análisis de código puede ayudar a identificar las rutas de código que pueden ralentizar la inicialización del paquete, también puede utilizar el seguimiento mediante el uso de aplicaciones como PerfView para comprender el impacto de un paquete de carga en el inicio de Visual Studio.

PerfView es una herramienta de seguimiento de todo el sistema. Esta herramienta le ayudará a comprender las rutas de acceso activas en una aplicación, ya sea debido a uso de CPU o bloquear las llamadas del sistema. A continuación es un ejemplo rápido acerca de cómo analizar una extensión de ejemplo con PerfView disponible en el [Microsoft Download Center](#).

Ejemplo de código:

En este ejemplo se basa en el ejemplo de código siguiente, que está diseñado para mostrar algunas causas comunes de retraso de casos:

```
protected override void Initialize()
{
    // Initialize a class from another assembly as an example
    MakeVsSlowServiceImpl service = new MakeVsSlowServiceImpl();

    // Costly work in main thread involving file IO
    string systemPath = Environment.GetFolderPath(Environment.SpecialFolder.Windows);
    foreach (string file in Directory.GetFiles(systemPath))
    {
        DateTime creationDate = File.GetCreationTime(file);
    }

    // Costly work after shell is initialized. This callback executes on main thread
    KnownUIContexts.ShellInitializedContext.WhenActivated(() =>
    {
        DoMoreWork();
    });

    // Start async work on background thread
    DoAsyncWork().Forget();
}

private async Task DoAsyncWork()
{
    // Switch to background thread to do expensive work
    await TaskScheduler.Default;
    System.Threading.Thread.Sleep(500);
}

private void DoMoreWork()
{
    // Costly work
    System.Threading.Thread.Sleep(500);
    // Blocking call to an asynchronous work.
    ThreadHelper.JoinableTaskFactory.Run(async () => { await DoAsyncWork(); });
}
```

Grabación de un objeto trace con PerfView:

Una vez configurado el entorno de Visual Studio con la extensión instalada, puede registrar un seguimiento del inicio, abra PerfView y abra el **recopilar** cuadro de diálogo desde el **recopilar** menú.

Collecting data over a user specified interval

This dialog give displays options for collecting ETW profile data. The only required field the 'Command' field and this is only necessary when using the 'Run' command.

If you wish to analyze on another machine use the Zip option when collecting data. See [Collecting ETW Profile Data](#), for more.

Command: ** Machine Wide **

Data File: PerfViewData.etl

Current Dir: d:\temp

Zip: ☐ Circular: ☒ 2048 Merge: ☐ Thread Time: ☒ Mark Text: Mark 1 Mark Start Collection Log Cancel

Status: Press Start Collection to Start.

Advanced Options

Kernel Base: ☒ CPU Samples: ☒ Page Faults: ☐ File I/O: ☐ Registry: ☐ VirtAlloc: ☐ MemInfo: ☐

Handle: ☐ RefSet: ☐ IIS: ☐ NetMon: ☐ Net Capture: ☐

.NET: ☒ .NET Stress: ☐ Background JIT: ☐ .NET Calls: ☐ JIT Inlining: ☐

GC Collect Only: ☐ GC Only: ☐ .NET Alloc: ☐ .NET SampAlloc: ☐ ETW .NET Alloc: ☐ Dump Heap: ☐ Finalizers: ☐

Additional Providers: Provider Browser ?

CPU Sample Interval Msec: 1 Cpu Ctrs: OS Heap Exe OS Heap Process

.NET Symbol Collection: ☒ No V3.X NGEN Symbols: ☒ Symbol TimeOut: 120

Max Collect Sec: Stop Trigger

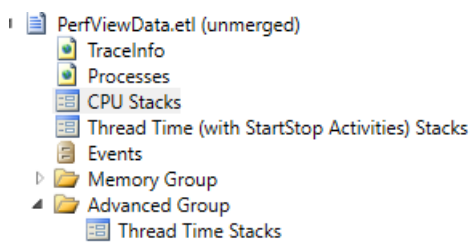
Las opciones predeterminadas proporcionará las pilas de llamadas para el consumo de CPU, pero puesto que estamos interesados en tiempo de bloqueo, también debe habilitar **tiempo subprocesos** pilas. Una vez que la configuración esté lista, puede hacer clic en **iniciar colección** y, a continuación, abra Visual Studio después de la grabación se inicia.

Antes de detener la recopilación, desea asegurarse de que Visual Studio está completamente inicializado, la ventana principal está completamente visible y si la extensión tiene interfaz de usuario que se muestran automáticamente, también están visibles. Cuando Visual Studio se cargó completamente y se inicializa la extensión, puede detener la grabación para analizar el seguimiento.

Analizar un seguimiento con PerfView:

Una vez completada la grabación de PerfView automáticamente se abra el seguimiento y expanda las opciones.

Para los fines de este ejemplo, nos interesa principalmente la **las pilas de subprocesos en tiempo** vista que puede encontrar en **Advanced Group**. Esta vista mostrará el tiempo total invertido en un subproceso por un método como el tiempo de CPU y tiempo de bloqueo, como E/S de disco o en espera en identificadores.



Mientras se abre **las pilas de subprocesos en tiempo** ver, debe elegir el **devenv** proceso para iniciar el análisis.

PerfView ha orientación detallada acerca de cómo se leen las pilas en tiempo en su propio menú de ayuda para un análisis más detallado de subproceso. Para fines de este ejemplo, desea filtrar aún más esta vista solo incluyendo pilas con el subproceso de inicio y el nombre de módulo de paquetes.

1. Establecer **GroupPats** en texto vacío para quitar cualquier agrupación que se agrega de forma predeterminada.
2. Establecer **IncPats** para incluir parte de su nombre de ensamblado y el inicio del subproceso además de filtro de proceso existente. En este caso, debe ser **devenv; Subproceso de inicio; MakeVsSlowExtension**.

Ahora la vista solo mostrará el costo asociado con los ensamblados relacionados con la extensión. En esta vista,

siempre que se muestra bajo el **Inc (costo inclusivo)** columna del subproceso de inicio se relaciona con nuestra extensión filtrado y se pueden influir en el inicio.

Para el ejemplo anterior, algunos interesante llamada pilas sería:

- 1. Uso de E/S `System.IO` clase: Aunque inclusivo costo de estos marcos podría no ser demasiado costosa en el seguimiento, son una posible causa de un problema ya que la velocidad de E/S de archivos puede variar de un equipo a otro.

+ ☒ microsoft.visualstudio.shell.15.0.ni!?
+ ☒ makevsslowextension!MakeVsSlowExtension.MyPackage.Initialize()
+ ☒ mscorlib.ni!System.IO.File.GetCreationTime(System.String)
+ ☒ mscorlib.ni!System.IO.File.InternalGetCreationTimeUtc(System.String, Boolean)
+ ☒ mscorlib.ni!System.IO.File.FillAttributeInfo(System.String, WIN32_FILE_ATTRIBUTE_DATA ByRef, Boolean, Boolean)
+ ☒ mscorlib.ni!System.IO.File.GetCreationTimeUtc(System.String, Boolean)

- 2. Bloqueo de las llamadas a la espera en otro trabajo asincrónico: En este caso, el tiempo inclusivo representaría el tiempo que se bloquea el subproceso principal tras la finalización del trabajo asincrónico.

+ ☒ makevsslowextension!MakeVsSlowExtension.MyPackage.<Initialize>b__2_0()	49.9	1,003.2
+ ☒ makevsslowextension!MakeVsSlowExtension.MyPackage.DoMoreWork()	49.9	1,003.2
+ ☐ mscorlib.ni!System.Threading.Thread.Sleep(Int32)	25.0	501.6
+ ☒ microsoft.visualstudio.threading!Microsoft.VisualStudio.Threading.JoinableTaskFactory.Run()	25.0	501.6
+ ☒ microsoft.visualstudio.threading.ni!?	25.0	501.6
+ ☒ mscorlib.ni!System.Threading.Tasks.Task.Wait(System.TimeSpan)	24.9	500.4
+ ☐ mscorlib.ni!System.Threading.Tasks.Task.Wait(Int32, System.Threading.CancellationToken)	24.9	500.4

Una de las otras vistas en el seguimiento que resultarán útiles para conocer el impacto será el **pilas de carga de imágenes**. Puede aplicar los mismos filtros que se aplican a **las pilas de subprocesos en tiempo** ver y averiguar todos los ensamblados cargados por el código ejecutado por el paquete cargado automáticamente.

Es importante minimizar el número de ensamblados cargados dentro de una rutina de inicialización del paquete como cada ensamblado adicional implicará la E/S de disco adicional que puede ralentizar el inicio considerablemente en equipos más lentos.

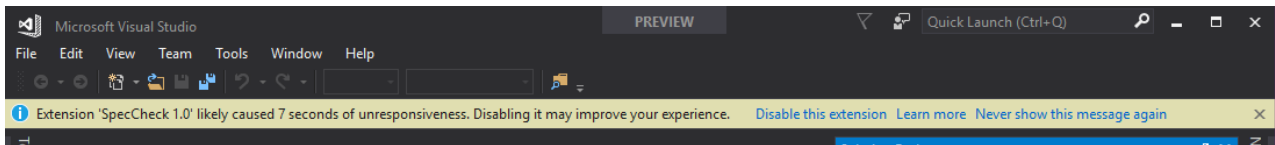
Resumen

Inicio de Visual Studio ha sido una de las áreas en que continuamente recibimos comentarios. Nuestro objetivo, como se indicó anteriormente es para que todos los usuarios tengan un inicio coherente experiencia independientemente de los componentes y extensiones que han instalado. Nos gustaría trabajar con los propietarios de extensión que les ayuden a alcanzar ese objetivo. Las instrucciones anteriores deben ser útil para comprender un impacto de las extensiones en el inicio y o evitar la necesidad de auto carga o cargarlo de forma asincrónica para minimizar el impacto en la productividad del usuario.

Cómo: diagnosticar retrasos de la interfaz de usuario causados por extensiones

10/01/2020 • 16 minutes to read • [Edit Online](#)

Cuando la interfaz de usuario deja de responder, Visual Studio examina la pila de llamadas del subproceso de la interfaz de usuario, empezando por la hoja y trabajando hacia la base. Si Visual Studio determina que un marco de pila de llamadas pertenece a un módulo que forma parte de una extensión instalada y habilitada, muestra una notificación.



La notificación informa al usuario de que el retraso de la interfaz de usuario (es decir, la falta de respuesta en la interfaz de usuario) podría haber sido el resultado del código de una extensión. También proporciona al usuario opciones para deshabilitar la extensión o las notificaciones futuras para esa extensión.

En este documento se describe cómo diagnosticar qué en el código de extensión está causando notificaciones de retraso de la interfaz de usuario.

NOTE

No use la instancia experimental de Visual Studio para diagnosticar los retrasos de la interfaz de usuario. Algunas partes del análisis de la pila de llamadas necesarias para las notificaciones de retraso de la interfaz de usuario se desactivan al usar la instancia experimental, lo que significa que es posible que no se muestren notificaciones de retraso de la interfaz de usuario.

Una introducción al proceso de diagnóstico es la siguiente:

1. Identifique el escenario del desencadenador.
2. Reinicie VS con el inicio de sesión de actividad.
3. Inicie el seguimiento de ETW.
4. Desencadene la notificación para que aparezca de nuevo.
5. Detenga el seguimiento de ETW.
6. Examine el registro de actividad para obtener el identificador de retraso.
7. Analice el seguimiento de ETW con el identificador de retraso del paso 6.

En las secciones siguientes, seguiremos estos pasos con más detalle.

Identificar el escenario de desencadenador

Para diagnosticar un retraso de la interfaz de usuario, primero debe identificar qué (secuencia de acciones) hace que Visual Studio muestre la notificación. Esto está disponible para que pueda desencadenar la notificación más adelante con el registro activado.

Reiniciar VS con el registro de actividades

Visual Studio puede generar un "registro de actividad" que proporciona información útil al depurar un problema. Para activar el registro de actividad en Visual Studio, abra Visual Studio con la opción de línea de comandos `/log`. Una vez que Visual Studio se inicia, el registro de actividad se almacena en la siguiente ubicación:

```
%APPDATA%\Microsoft\VisualStudio\<vs_instance_id>\ActivityLog.xml
```

Para obtener más información sobre cómo encontrar el identificador de la instancia de VS, vea [herramientas para detectar y administrar instancias de Visual Studio](#). Más adelante usaremos este registro de actividad para obtener más información sobre los retrasos de la interfaz de usuario y las notificaciones relacionadas.

Inicio del seguimiento de ETW

Puede usar [PerfView](#) para recopilar un seguimiento de ETW. PerfView proporciona una interfaz fácil de usar para recopilar un seguimiento de ETW y para analizarlo. Use el comando siguiente para recopilar un seguimiento:

```
Perfview.exe collect C:\trace.etl /BufferSizeMB=1024 -CircularMB:2048 -Merge:true -Providers:*Microsoft-VisualStudio:@StacksEnabled=true -NoV2RunDown /kernelEvents=default+FileIOInit+ContextSwitch+Dispatcher
```

Esto habilita el proveedor de "Microsoft-VisualStudio", que es el proveedor que Visual Studio usa para los eventos relacionados con las notificaciones de retraso de la interfaz de usuario. También especifica la palabra clave para el proveedor de kernel que PerfView puede usar para generar la vista de **pilas de tiempo de subproceso**.

Desencadenar la notificación para que aparezca de nuevo

Una vez que PerfView ha iniciado la recopilación de seguimiento, puede usar la secuencia de acción desencadenadora (del paso 1) para que la notificación vuelva a aparecer. Una vez que se muestra la notificación, puede detener la recopilación de seguimiento de PerfView para procesar y generar el archivo de seguimiento de salida.

Detener seguimiento de ETW

Para detener la recopilación de seguimiento, basta con usar el botón **detener colección** de la ventana de PerfView. Después de detener la recopilación de seguimiento, PerfView procesará automáticamente los eventos ETW y generará un archivo de seguimiento de salida.

Examinar el registro de actividad para obtener el identificador de retraso

Como se mencionó anteriormente, puede encontrar el registro de actividad en `%APPDATA%\Microsoft\VisualStudio\<vs_instance_id>\ActivityLog.XML`. Cada vez que Visual Studio detecta un retraso de la interfaz de usuario de la extensión, escribe un nodo en el registro de actividad con `UIDelayNotifications` como origen. Este nodo contiene cuatro fragmentos de información sobre el retraso de la interfaz de usuario:

- El identificador de retraso de la interfaz de usuario, un número secuencial que identifica de forma única un retraso de la interfaz de usuario en una sesión de VS
- El identificador de sesión, que identifica de forma única la sesión de Visual Studio de inicio a cerrar
- Indica si se ha mostrado o no una notificación para el retraso de la interfaz de usuario
- La extensión que probablemente causó el retraso de la interfaz de usuario

```
<entry>
  <record>271</record>
  <time>2018/02/03 12:02:52.867</time>
  <type>Information</type>
  <source>UIDelayNotifications</source>
  <description>A UI delay (Delay ID = 0) has been detected. (Session ID=16e49d4b-26c2-4247-ad1c-488edeb185e0;
Blamed extension="UIDelayR2"; Notification shown? Yes.)</description>
</entry>
```

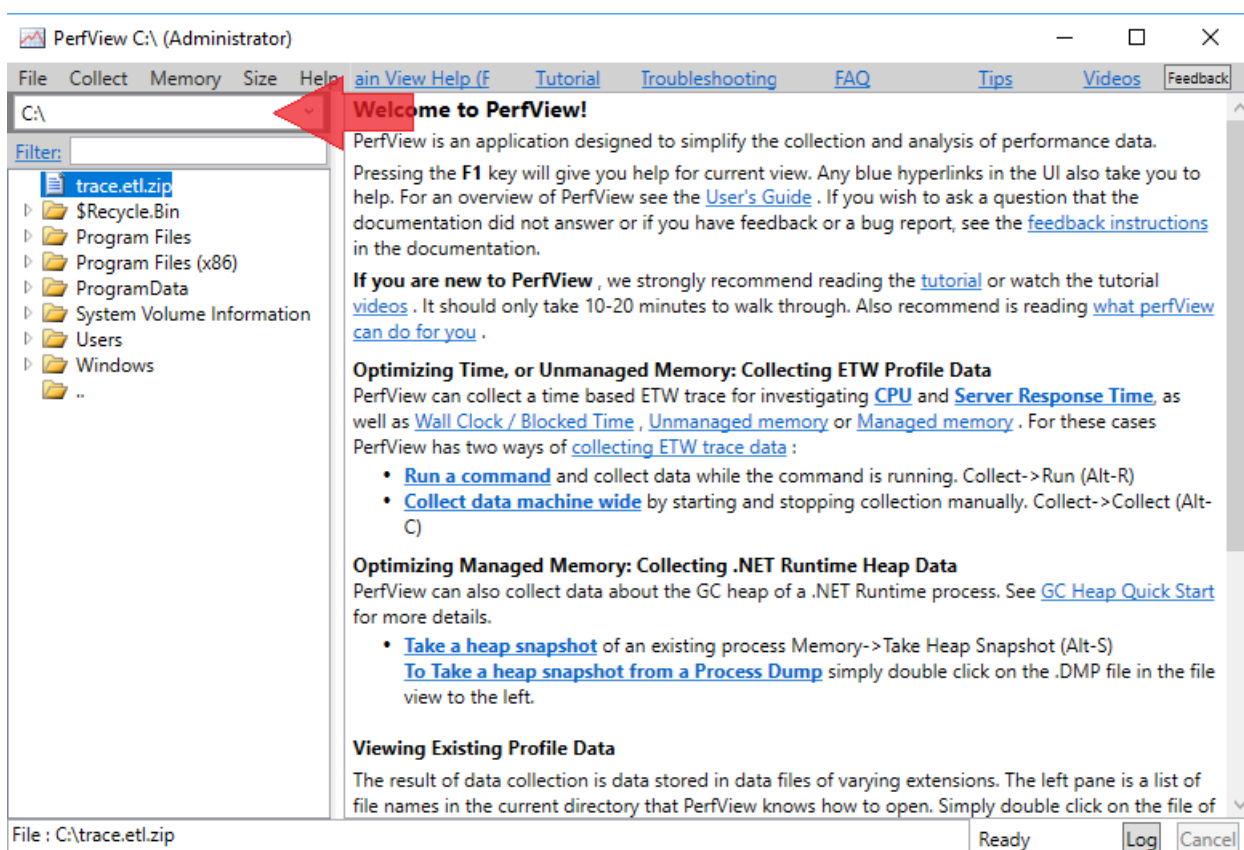
NOTE

No todos los retrasos de la interfaz de usuario generan una notificación. Por lo tanto, siempre debe comprobar la **notificación que se muestra?** valor para identificar correctamente el retraso de la interfaz de usuario correcto.

Después de encontrar el retraso de la interfaz de usuario correcto en el registro de actividad, anote el identificador de retraso de la interfaz de usuario especificado en el nodo. Usará el identificador para buscar el evento ETW correspondiente en el paso siguiente.

Analizar el seguimiento de ETW

A continuación, abra el archivo de seguimiento. Para ello, puede usar la misma instancia de PerfView o iniciar una nueva instancia y establecer la ruta de acceso de la carpeta actual en la parte superior izquierda de la ventana en la ubicación del archivo de seguimiento.

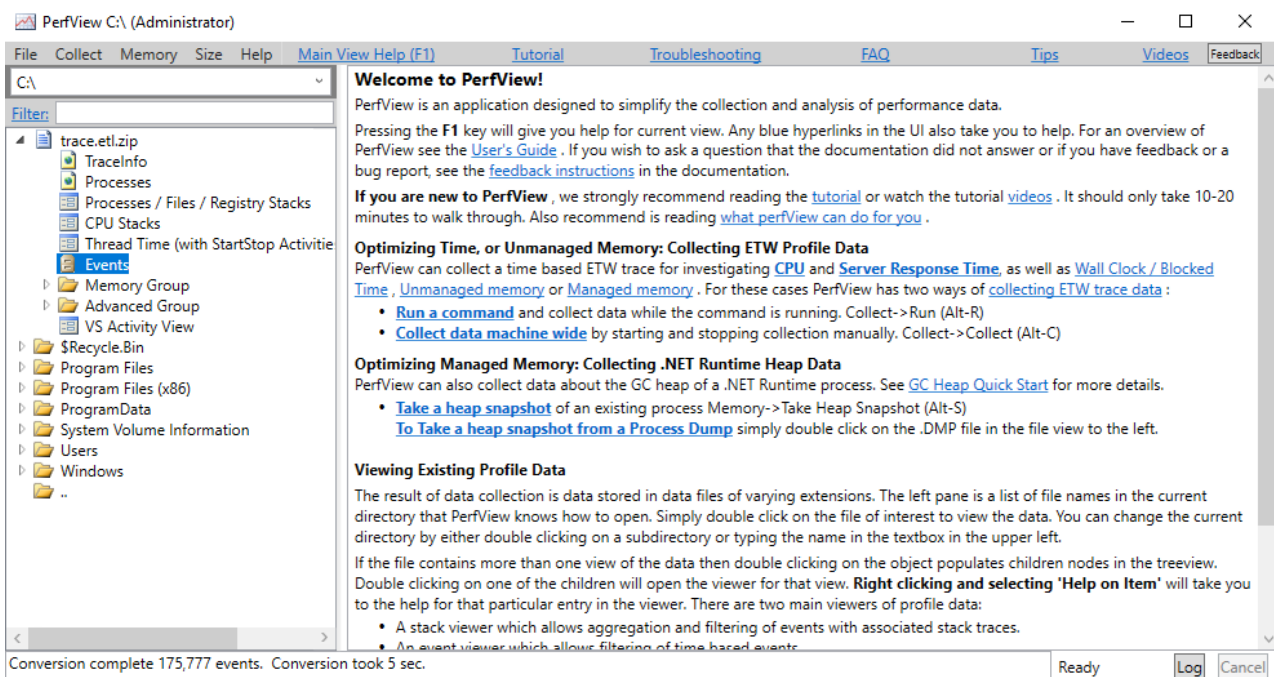


A continuación, seleccione el archivo de seguimiento en el panel izquierdo y ábralo eligiendo **abrir** en el menú contextual.

NOTE

De forma predeterminada, PerfView genera un archivo zip. Al abrir *Trace.zip*, se descomprime automáticamente el archivo y se abre el seguimiento. Para omitir esto, desactive el cuadro **zip** durante la recopilación de seguimiento. Sin embargo, si planea transferir y usar seguimientos en diferentes equipos, se recomienda encarecidamente no desactivar el cuadro **zip**. Sin esta opción, los archivos PDB necesarios para los ensamblados Ngen no se adjuntarán al seguimiento y, por tanto, los símbolos de los ensamblados Ngen no se resolverán en el equipo de destino. (Vea [esta entrada de blog](#) para obtener más información sobre los archivos PDB para los ensamblados Ngen).

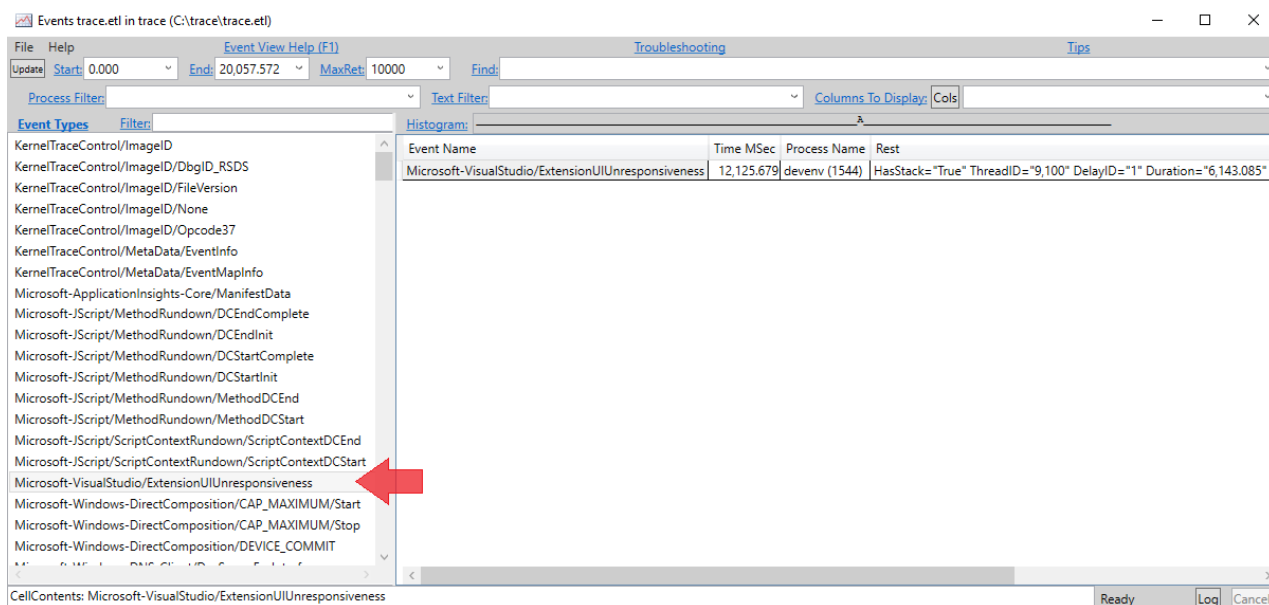
La PerfView puede tardar varios minutos en procesar y abrir el seguimiento. Una vez que el seguimiento está abierto, aparece una lista de las distintas "vistas" en él.



En primer lugar, usaremos la vista de **eventos** para obtener el intervalo de tiempo del retraso de la interfaz de usuario:

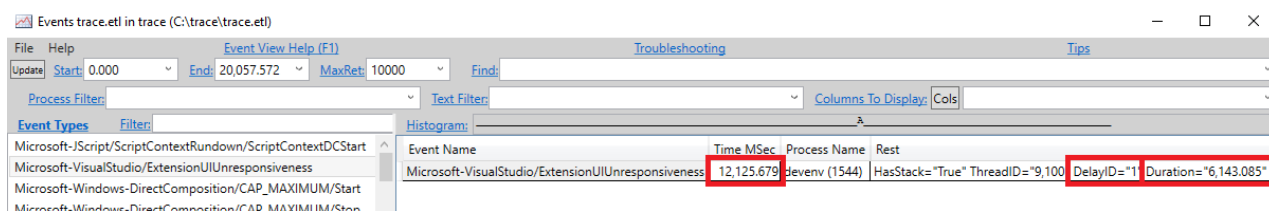
1. Abra la vista **eventos** seleccionando **Events** nodo en el seguimiento y eligiendo **abrir** en el menú contextual.
2. En el panel izquierdo, seleccione "Microsoft-VisualStudio/ExtensionUIUnresponsiveness".
3. Presione entrar.

Se aplica la selección y todos los eventos de **ExtensionUIUnresponsiveness** se muestran en el panel derecho.



Cada fila del panel derecho corresponde a un retraso de la interfaz de usuario. El evento incluye un valor de "ID. de retraso" que debe coincidir con el ID. de retraso del registro de actividad del paso 6. Como

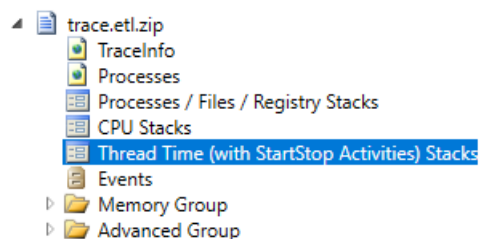
ExtensionUIUnresponsiveness se desencadena al final del retraso de la interfaz de usuario, la marca de tiempo del evento (aproximadamente) marca la hora de finalización del retraso de la interfaz de usuario. El evento también contiene la duración del retraso. Podemos restar la duración de la marca de tiempo final para obtener la marca de tiempo de Cuándo se inició el retraso de la interfaz de usuario.



En la captura de pantalla anterior, por ejemplo, la marca de tiempo del evento es 12.125,679 y la duración del retraso es 6.143,085 (MS). Así,

- El inicio del retraso es $12.125,679 - 6.143,085 = 5.982,594$.
- El intervalo de tiempo de retraso de la interfaz de usuario es de 5.982,594 a 12.125,679.

Una vez que tenemos el intervalo de tiempo, podemos cerrar la vista de **eventos** y abrir la vista de **pilas de tiempo del subproceso (con actividades de StartStop)**. Esta vista es especialmente útil porque, a menudo, las extensiones que están bloqueando el subproceso de la interfaz de usuario simplemente están esperando en otros subprocesos o en una operación enlazada a e/s. Por lo tanto, la vista de pila de la CPU, que es la opción de ir a para la mayoría de los casos, no puede capturar el tiempo que el subproceso invierte en el bloqueo, ya que no está usando la CPU durante ese tiempo. La **pila de tiempo de subprocesos** resuelve este problema mostrando correctamente el tiempo de bloqueo.



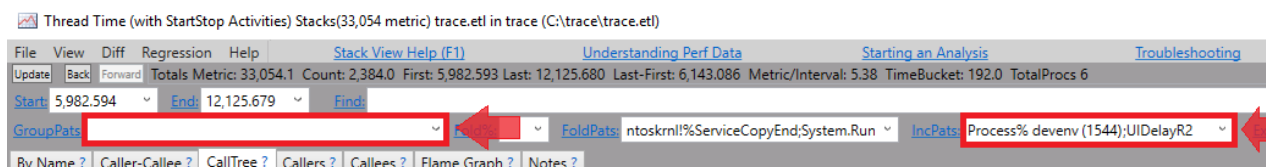
Al abrir la vista **pilas de tiempo de subprocesos**, elija el proceso **devenv** para iniciar el análisis.

[illegible]

NOTE

☒ ROOT
+ ☒ Process32 devenv (1544) Args: /log
+ ☒ (Non-Activities)
+ ☒ Threads
+ ☐ Thread (2976) CPU=28ms
+ ☐ Thread (6304) CPU=23ms
+ ☐ Thread (240) CPU=145ms
+ ☒ Thread (5212) CPU=1213ms
+ ☒ IntlIII?
+ ☒ kernel32!?
+ ☒ devenv!?
+ ☒ msenv!?
+ ☒ user32!?
+ ☒ windowsbase.ni!MS.Win32.HwndSubclass.SubclassWndProc(IntPt
+ ☒ msenv!?

- Establezca **GroupPats** en texto vacío para quitar cualquier agrupación agregada de forma predeterminada.
- Establezca **IncPats** para incluir parte del nombre del ensamblado además del filtro de proceso existente. En este caso, debe ser **devenv; UIDelayR2**.



PerfView tiene instrucciones detalladas en el menú **ayuda** que puede usar para identificar cuellos de botella de rendimiento en el código. Además, los vínculos siguientes proporcionan más información sobre cómo se usan las API de subprocesos de Visual Studio para optimizar el código:

- <https://github.com/Microsoft/vs-threading/blob/master/doc/index.md>
- https://github.com/Microsoft/vs-threading/blob/master/doc/cookbook_vs.md

También puede usar los nuevos Analizadores estáticos de Visual Studio para extensiones (paquete NuGet [aquí](#)), que proporcionan instrucciones sobre los procedimientos recomendados para escribir extensiones eficientes. Vea una lista de [analizadores de vs SDK](#) y [analizadores de subprocesos](#).

NOTE

Si no puede solucionar la falta de respuesta debido a las dependencias en las que no tiene control (por ejemplo, si la extensión tiene que llamar a servicios de VS sincrónicos en el subproceso de la interfaz de usuario), nos gustaría conocerlo. Si es miembro de nuestro programa de Partners de Visual Studio, puede ponerse en contacto con nosotros enviando una solicitud de soporte técnico para desarrolladores. De lo contrario, use la herramienta "Notificar un problema" para enviar sus comentarios e incluir "Extension UI Delay Notifications" en el título. También debe incluir una descripción detallada del análisis.

Solución de problemas de VSPackages

06/04/2020 • 8 minutes to read • [Edit Online](#)

A continuación se muestran los problemas comunes que puede tener con el VSPackage y sugerencias para resolver los problemas.

Para solucionar problemas de un VSPackage que impide que Visual Studio se inicie

- Comience Visual Studio en modo seguro.

Para Visual Studio iniciar en modo seguro, en un símbolo del sistema, escriba **devenv.exe /safemode**.

Durante este proceso no se cargan ningún VSPackages Visual Studio excepto los VSPackages que se incluyen con .

Para solucionar problemas de un VSPackage que no se carga

1. Asegúrese de que está utilizando la raíz del registro en la que el VSPackage está registrado para ejecutarse, normalmente la raíz del registro experimental.

Para obtener más información, consulte [La instancia experimental](#).

2. Si el VSPackage está destinado a ejecutarse en la raíz del registro Visual Studio experimental, asegúrese de que está ejecutando la versión experimental de .

Para ejecutar la versión experimental, escriba lo siguiente en una ventana de comandos: **devenv /rootsuffix exp**.

3. Compruebe las entradas del registro de VSPackage.

Para obtener más información, vea [Registrar VSPackages](#) y [Administrar VSPackages](#).

4. Abra el **output** ventana Visual Studio de la instancia de que no se puede cargar el VSPackage. La información sobre por qué el VSPackage no se puede cargar se puede mostrar en esa ventana.

NOTE

Si está iniciando la Visual Studio versión experimental desde el Visual Studio entorno de desarrollo integrado (IDE), inspeccione la ventana **Salida** de ambas versiones.

5. Examine el registro de actividad.

Para obtener más información, consulte [Cómo: Usar el registro de actividad](#) .

6. Para obtener más información acerca de las excepciones producidas por el IDE, haga clic en **Excepciones** en el **depurar** menú para habilitar las excepciones. En el cuadro de diálogo **Excepciones**, seleccione los tipos de excepciones sobre los que desea obtener más información.

Para solucionar problemas de un VSPackage que no se registra

1. Asegúrese de que el ensamblado VSPackage reside en una ubicación de confianza. RegPkg no puede registrar ensamblados en una ubicación que no sea de confianza o que no sea de confianza parcial, como un recurso compartido de red en la configuración de seguridad predeterminada de .net. Aunque aparece una advertencia cada vez que un usuario crea un proyecto en una ubicación que no es de confianza, la casilla de verificación "no volver a mostrar este mensaje" puede impedir que se repita esta advertencia.

Para solucionar problemas de un comando que no está visible o que genera un error al hacer clic en un

comando

1. Combine los comandos de menú nuevos o modificados y Visual Studio los que ya están en el IDE escribiendo lo siguiente en el símbolo del sistema: **devenv /rootsuffix Exp /setup**.
2. Asegúrese Visual Studio de que puede encontrar UI.dll para el VSPackage.
 - a. Busque el CLSID del VSPackage en la sección Packages del registro:

HKLM, Software, Microsoft\Visual* <Studio, versión> * paquetes

- b. Compruebe que la ruta de acceso proporcionada por la subclave SatelliteDll es correcta.

Para solucionar problemas de un VSPackage que se comporta inesperadamente

1. Establezca puntos de interrupción en el código.

Los buenos puntos de partida para la depuración son el constructor y el método de inicialización. También puede establecer puntos de interrupción en el área que desea evaluar, como un comando de menú. Para habilitar los puntos de interrupción, debe ejecutarse en el depurador.

- a. En el menú **Proyecto** , haga clic en **Propiedades**.
- b. En el cuadro de diálogo Páginas de **propiedades**, seleccione la pestaña **Depurar**.
- c. En el argumentos de línea de **comandos** cuadro, escriba el sufijo raíz del entorno de desarrollo que los destinos de VSPackage. Por ejemplo, para seleccionar la compilación experimental, escriba: **/RootSuffix Exp**.
- d. En el menú **Depurar**, haga clic en **Iniciar depuración** o presione F5.

NOTE

Si está depurando un proyecto, cree o cargue una instancia existente del proyecto ahora.

2. Utilice el registro de actividad.

Seguimiento del comportamiento de VSPackage escribiendo información en el registro de actividad en puntos clave. Esta técnica es especialmente útil cuando se ejecuta un VSPackage en un entorno comercial. Para obtener más información, consulte [Cómo: Usar el registro de actividad](#) .

3. Utilice símbolos públicos.

Para mejorar la legibilidad durante la depuración, puede adjuntar símbolos al depurador.

- a. En el menú **Herramientas/Opciones**, vaya al cuadro de diálogo **Depuración/Símbolos**.
- b. Agregue esta ubicación del **archivo de símbolo (.pdb)**:

<https://msdl.microsoft.com/download/symbols>

- c. Para mejorar el rendimiento, especifique una carpeta de caché de símbolos, por ejemplo:

C:\symbols

Para solucionar problemas de un VSPackage que falta o una de sus dependencias

1. Para el código administrado, asegúrese de que las rutas de acceso de referencia son correctas.
 - a. En el menú **Proyecto** , haga clic en **Propiedades**.
 - b. Seleccione la pestaña **Referencias** en el cuadro de diálogo **Páginas** de propiedades y asegúrese de

que todas las rutas son correctas. Como alternativa, puede utilizar el **Examinador de objetos** para buscar los objetos a los que se hace referencia.

Para el código administrado, puede usar [Fuslogvw.exe \(Assembly Binding Log Viewer\)](#) para mostrar los detalles de las cargas de ensamblado con errores.

2. Para el código no administrado, busque el CLSID del VSPackage en el Visual Studio nodo del Registro CLSID:

*La\<versión de *HKLM-Software-Microsoft-Visual Studio>de la versión de CLSID*

Asegúrese de que la entrada InprocServer32 tiene la ruta de acceso correcta del archivo dll de VSPackage.

Vea también

- [VSPackages](#)

Suministro de extensiones de Visual Studio

06/04/2020 • 6 minutes to read • [Edit Online](#)

Una vez que haya terminado de desarrollar la extensión, puede instalarla en otros equipos, compartirla con sus amigos y compañeros de trabajo o publicarla en Visual Studio Marketplace. En esta sección te explicamos todas las cosas que necesitas hacer para publicar y mantener tu extensión: trabajar con archivos .vsix, publicar, localizar y actualizar.

Trabajar con extensiones VSIX

Puede crear una extensión VSIX creando un proyecto VSIX en blanco y, a continuación, agregándole diferentes plantillas de elementos. Para obtener más información, consulte [Plantilla de proyecto VSIX](#).

Puede usar el formato VSIX para empaquetar plantillas de proyecto, plantillas de elementos, VSPackages, componentes de Managed Extensibility Framework (MEF), controles de **Toolbox**, ensamblados y tipos personalizados (esto incluye páginas de inicio personalizadas para Visual Studio 2017). El formato VSIX utiliza la implementación basada en archivos. Para obtener más información acerca de los paquetes VSIX, consulte [Anatomía de un paquete VSIX](#).

El formato VSIX no admite la instalación de fragmentos de código. Tampoco admite otros escenarios, como escribir en la caché global de ensamblados (GAC) o en el registro del sistema. Si necesita escribir en la GAC o en el registro de la instalación, debe usar Windows Installer. Para obtener más información, consulte [Preparación de extensiones para la implementación de Windows Installer](#).

Publicación de la extensión en Visual Studio Marketplace

Puede distribuir su extensión a otras personas simplemente enviándoles por correo el archivo .vsix o poniéndolo en un servidor. Pero la mejor manera de poner el código en manos de mucha gente es ponerlo en [Visual Studio Marketplace](#). Las extensiones de Visual Studio Marketplace están disponibles para los usuarios de Visual Studio a través de **Extensiones y actualizaciones**. Para obtener más información, vea [Buscar y usar extensiones](#) de Visual Studio.

Para obtener un ejemplo completo que muestra cómo cargar una extensión en Visual Studio Marketplace, vea [Tutorial: publicación](#) de una extensión de Visual Studio.

Private Galleries

A medida que desarrolla controles, plantillas y herramientas, puede compartirlas con su organización publicándolos en una galería privada en la intranet. Para obtener más información, consulte [Galerías privadas](#).

Localizar la extensión

Si está planeando liberar la extensión en diferentes configuraciones regionales, debe considerar la posibilidad de localizarla. Para obtener una explicación de lo que implica, vea [Localizar paquetes VSIX](#).

Actualización y control de versiones de la extensión

Después de publicar la extensión, llegará un momento en el que necesite actualizarla. Para averiguar cómo actualizar una extensión que se ha publicado en Visual Studio Marketplace, vea [Cómo: actualizar una extensión](#).

Puede establecer la extensión para admitir varias versiones de Visual Studio. Para obtener más información, vea

Temas relacionados

TITLE	DESCRIPCIÓN
Introducción a la plantilla de proyecto VSIX	Explica cómo usar la plantilla de proyecto VSIX para instalar una plantilla de proyecto personalizada.
Anatomía de un paquete VSIX	Describe los componentes de un paquete VSIX.
Plantilla de proyecto de VSIX	Proporciona instrucciones paso a paso sobre cómo empaquetar y publicar una extensión.
Adaptación de paquetes VSIX	Explica cómo proporcionar texto localizado para el proceso de instalación mediante archivos extension.vsixlangpack.
Cómo actualizar una extensión	Describe cómo actualizar una extensión en el sistema y cómo implementar una actualización en una extensión de Visual Studio existente.
Adición de una dependencia a un paquete VSIX	Describe cómo agregar referencias a paquetes de implementación VSIX.
Preparación de las extensiones para la implementación de Windows Installer	Explica cómo implementar la extensión con Windows Installer.
Firma de paquetes VSIX	Explica cómo firmar paquetes VSIX.
Galerías privadas	Explica cómo crear galerías privadas para extensiones.
Compatibilidad con varias versiones de Visual Studio	Muestra cómo hacer que la extensión admita varias versiones de Visual Studio.
Búsqueda de Visual Studio	Describe cómo buscar instancias de Visual Studio para la implementación de extensiones personalizadas.

Plantilla de proyecto VSIX

06/04/2020 • 4 minutes to read • [Edit Online](#)

Puede usar la plantilla de proyecto VSIX para ajustar una o varias extensiones de Visual Studio en un proyecto VSIX y, a continuación, publicar el paquete en el sitio Web de [Visual Studio Marketplace](#).

La implementación de VSIX admite VSPackages, ensamblados, componentes MEF, plantillas de proyecto, plantillas de elementos, controles de cuadro de herramientas y tipos de extensión personalizados.

NOTE

Para usar proyectos VSIX, debe instalar el SDK de Visual Studio. Para obtener más información acerca del SDK de Visual Studio, vea SDK de [Visual Studio](#).

Dónde encontrar la plantilla de proyecto VSIX

La plantilla Proyecto VSIX está disponible en el cuadro de diálogo **Nuevo proyecto** buscando "vsix". Hay una versión de Visual Basic y de C.

TIP

Debe asegurarse de que .NET Framework 4.5 o superior se especifica en el cuadro de lista desplegable en la parte superior del cuadro de diálogo **Nuevo proyecto**.

Usos de la plantilla de proyecto VSIX

La plantilla de proyecto VSIX tiene dos usos principales:

- Para implementar plantillas de proyecto, plantillas de elementos y extensiones.
- Para ajustar las salidas de varias extensiones en un paquete de implementación.

Empaquetado de una extensión en un proyecto VSIX vacío

Puede empaquetar una extensión existente o una extensión que aún no tenga compatibilidad con VSIX, envolviéndola en un proyecto VSIX vacío. La extensión que se va a ajustar debe ser de un tipo compatible con el [esquema VSIX](#).

Para empaquetar una extensión mediante un proyecto VSIX

1. Compile los proyectos que componen la extensión.
2. Cree un proyecto VSIX mediante la plantilla **Proyecto VSIX**.

Source.extension.vsixmanifest se abre en el **Diseñador de manifiestos**.

3. En la pestaña **Assets(Assets)**, elija el botón **New (Nuevo)**.

Aparece el cuadro de diálogo **Agregar nuevo recurso**.

4. En la lista **Tipo**, elija el tipo de extensión que desea agregar.
5. Para agregar una extensión o un elemento de contenido que se incluye en la solución actual (por ejemplo,

una plantilla de elemento o un ensamblado compilado), realice los pasos siguientes:

- a. En la lista **Origen**, elija Un proyecto en la **solución actual**.
 - b. En la lista **Proyecto**, elija el nombre de la extensión.
 - c. En el cuadro **Incrustar en esta carpeta**, escriba el nombre de una carpeta en la que desea incrustar el recurso y, a continuación, elija el botón **Aceptar**.
6. Para agregar una extensión o un elemento de contenido que no se incluye en la solución actual, realice los pasos siguientes:
- a. En el cuadro de lista **Origen**, elija **Archivo en el sistema de archivos**.
 - b. En el campo **Ruta** de acceso, escriba la ruta de acceso completa al archivo de extensión compilado o comprimido o utilice el botón **Examinar** para buscar el archivo.
 - c. En el cuadro **Incrustar en esta carpeta**, escriba el nombre de una carpeta en la que desea incrustar el recurso y, a continuación, elija el botón **Aceptar**.
7. Si desea que el paquete incluya extensiones adicionales, agréguelas de la misma manera.
8. Compile la solución.

Visual Studiocompila un archivo `.vsix` que contiene un archivo de manifiesto VSIX, un archivo `.xml` [Content_Types] y todos los activos de extensión que agregó al proyecto.

Vea también

- [Referencia del esquema de extensión VSIX 2.0](#)
- [Búsqueda y uso de extensiones de Visual Studio](#)

Comience con la plantilla de proyecto VSIX

06/04/2020 • 6 minutes to read • [Edit Online](#)

Puede usar la plantilla de proyecto VSIX para crear una extensión o empaquetar una extensión existente para la implementación. La plantilla de proyecto VSIX tiene versiones de Visual Basic y Visual C, y se instala como parte del SDK de Visual Studio.

La plantilla de proyecto VSIX solo consta de un archivo *source.extension.vsixmanifest*, que contiene información sobre la extensión y los activos que envía.

Para buscar la plantilla de proyecto VSIX, debe instalar el SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Implementar una plantilla de proyecto personalizada mediante la plantilla de proyecto VSIX

En los pasos siguientes se muestra cómo usar el proyecto VSIX para empaquetar una plantilla de proyecto que puede compartir con otros desarrolladores o cargar en la Galería de Visual Studio.

1. Cree una plantilla de proyecto.
 - a. Abra el proyecto desde el que desea crear una plantilla. Este proyecto puede ser de cualquier tipo de proyecto.
 - b. En el menú **Proyecto**, haga clic en **Exportar plantilla**. Complete los pasos del asistente.

Se crea un archivo *.zip* en *%USERPROFILE%-Mis documentos-Visual\Studio .versión-Mis plantillas exportadas*.

2. Cree un proyecto VSIX vacío.

Seleccione **File (Archivo) > New (Nuevo) > Project (Proyecto)** . En el cuadro de búsqueda, escriba "vsix" y seleccione la versión de C o Visual Basic del proyecto VSIX.

3. Agregue el archivo *.zip* al proyecto. Establezca su propiedad Copiar Copy Always en directorio de salida en .
4. En el **Explorador de soluciones** , haga doble clic en el archivo *source.extension.vsixmanifest* para abrirlo en el Diseñador de **manifiestos VSIX**, a continuación, realice los siguientes cambios:
 - Establezca el campo **Nombre del producto** en Mi plantilla de **proyecto**.
 - Establezca el campo **ID de producto** en **MyProjectTemplate - 1**.
 - Establezca el campo **Autor** en **Fabrikam**.
 - Establezca el campo **Descripción** en **Mi plantillade proyecto** .
 - En la sección **Assets**, agregue un tipo **Microsoft.VisualStudio.ProjectTemplate** y establezca su ruta de acceso al nombre del archivo *.zip*.
5. Guarde y cierre el archivo *source.extension.vsixmanifest*.
6. Compile el proyecto.
7. En el directorio de salida, haga doble clic en el archivo *.vsix*.
8. Aparece un cuadro de mensaje **del instalador de VSIX**. Siga las instrucciones para instalar la extensión.

9. Cierre Visual Studio y vuelva a abrirlo.
10. Seleccione **Extensiones y actualizaciones** (en el menú **Herramientas**) y seleccione la categoría **Plantillas**. Una de las extensiones disponibles debe ser **Mi plantilla de proyecto**.
10. Seleccione **Administrar extensiones** (en el menú **Extensiones**) y seleccione la categoría **Plantillas**. Una de las extensiones disponibles debe ser **Mi plantilla de proyecto**.
11. La nueva plantilla de proyecto aparece en el cuadro de diálogo **Nuevo proyecto** en el mismo lugar que la plantilla de proyecto original. Por ejemplo, si ha creado una plantilla denominada **Consola de VB** a partir de una aplicación de consola de Visual Basic, la consola de **VB** aparece en el mismo panel que la plantilla Aplicación de **consola** de Visual Basic.

Para especificar la ubicación de la plantilla en el cuadro de diálogo Nuevo proyecto

1. Las carpetas de la plantilla se encuentran en los directorios de la ruta de instalación de *Visual Studio, Common7, IDE, ProjectTemplates* y la ruta de instalación de *Visual Studio, Common7, IDE, ItemTemplates*. Los nombres de las secciones de nivel superior en el cuadro de diálogo **Nuevo proyecto** no coinciden exactamente con los nombres de las carpetas de plantilla. Donde difieren, utilice el nombre de la carpeta de plantilla.

Cambie la extensión de archivo *.vsix* a *.zip*, a continuación, abra el archivo.

2. Cree una nueva carpeta con el mismo nombre que la sección del cuadro de diálogo **Nuevo proyecto** en el que debe aparecer la plantilla.
3. Si la plantilla va a aparecer en una subsección, cree una subcarpeta con el mismo nombre.
4. Mueva el archivo *.zip* de plantilla a la nueva carpeta.
5. Cambie la extensión *.zip* a *.vsix*.
6. Abra el manifiesto VSIX.
7. En el manifiesto VSIX, actualice la ruta de acceso **Asset** de la plantilla para que apunte a la raíz del árbol de directorios que contiene el archivo de plantilla. Por ejemplo, si la plantilla se encuentra en *el archivo .CSharp*, la referencia debe apuntar a *.CSharp*.

Tutorial: Publicar una extensión de Visual Studio

06/04/2020 • 11 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo publicar una extensión de Visual Studio en Visual Studio Marketplace. Al agregar la extensión a Marketplace, los desarrolladores pueden usar **Extensiones y actualizaciones** para buscar extensiones nuevas y actualizadas.

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

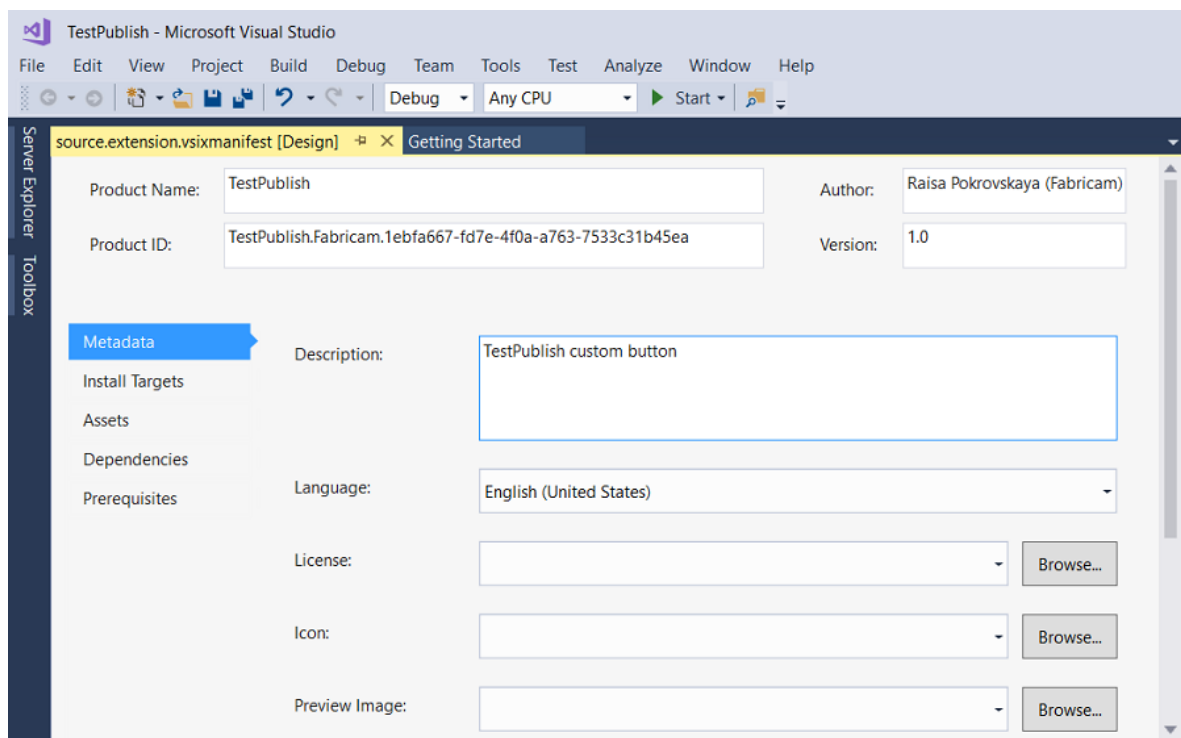
Crear una extensión de Visual Studio

En este artículo se usa una extensión de VSPackage predeterminada, pero los pasos son válidos para cada tipo de extensión.

1. Crear un VSPackage en `TestPublish` C- denominado que tiene un comando de menú. Para obtener más información, consulte [Crear la primera extensión: Hello World](#).

Empaquetado de la extensión

1. Actualice la extensión `.vsixmanifest` con la información correcta sobre el nombre del producto, el autor y la versión.



2. Cree la extensión en el modo **de versión**. Ahora, la extensión se empaqueta como un VSIX en la carpeta `.bin.Release`.
3. Puede hacer doble clic en VSIX para comprobar la instalación.

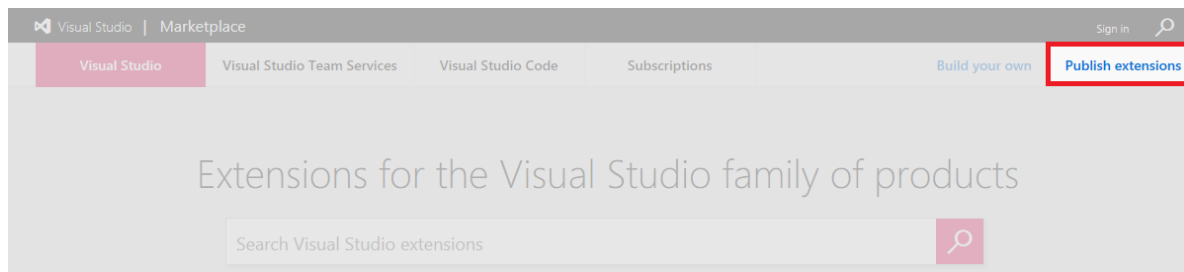
Pruebe la extensión

Antes de distribuir la extensión, compílela y pruébela para asegurarse de que está instalada correctamente en la instancia experimental de Visual Studio.

1. En Visual Studio, inicie la depuración para abrir una instancia experimental de Visual Studio.
2. En la instancia experimental, vaya al menú **Herramientas** y haga clic en **Extensiones y actualizaciones**. La extensión TestPublish debe aparecer en el panel central y estar habilitada.
3. En el menú **Herramientas**, asegúrese de ver el comando test.

Publique la extensión en Visual Studio Marketplace

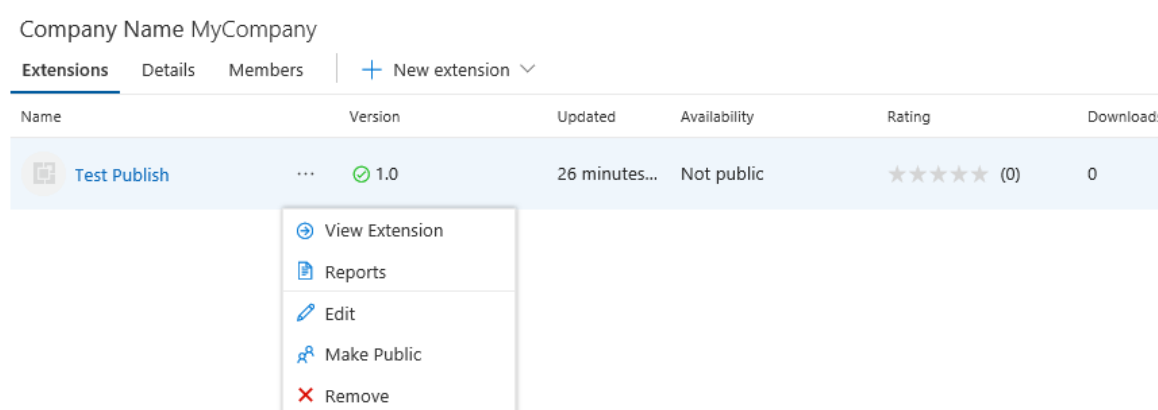
1. Asegúrese de que ha creado la versión de lanzamiento de la extensión y de que está actualizada.
2. En un explorador web, abra el sitio web de [Visual Studio Marketplace](#).
3. En la esquina superior derecha, haga clic en **Iniciar sesión**.
4. Para ello, use la cuenta de Microsoft. Si no tiene una cuenta Microsoft, puede crear una en este momento.
5. Haga clic en **Publicar extensiones**. Esta opción le navega a la página de administración de todas las extensiones. Si no tiene una cuenta de editor, se le pedirá que cree una en este momento.



6. Elija el editor que desea utilizar para cargar la extensión. Puede cambiar los editores haciendo clic en los nombres de los editores que aparecen a la izquierda. Haga clic en **Nueva extensión** y seleccione **Visual Studio**.
7. En **1: Cargar extensión**, puede elegir cargar un archivo VSIX directamente en Visual Studio Marketplace o simplemente agregar un vínculo a su propio sitio web. En este ejemplo, se carga la extensión *TestPublish.vsix*. Arrastre y suelte la extensión o utilice el vínculo de clic para buscar el archivo. Busque la extensión en la carpeta de la versión de la carpeta de la versión de la carpeta. Haga clic en **Continuar**.
8. En **2: Proporcionar detalles de extensión**, algunos campos se rellenan automáticamente desde el archivo *source.extension.vsixmanifest* desde la extensión. Encuentre más detalles sobre cada uno a continuación:
 - **Nombre interno** se utiliza en la dirección URL de la página de detalles de la extensión. Por ejemplo, publicar una extensión bajo el nombre del editor "myname" y especificar el nombre interno para que sea "my extension" da como resultado una dirección URL de "marketplace.visualstudio.com/items?itemName-myname.myextension" para la página de detalles de la extensión.
 - **Mostrar nombre** de la extensión. Este nombre se rellena automáticamente desde el archivo *source.extension.vsixmanifest*.
 - Número de **versión** de la extensión que está cargando. Esta versión se rellena automáticamente desde el archivo *source.extension.vsixmanifest*.
 - **Identificador de VSIX** es el identificador único que Visual Studio usa para la extensión. Este identificador es necesario si desea que la extensión se actualice automáticamente. Este identificador se rellena automáticamente desde el archivo *source.extension.vsixmanifest*.

- **Logotipo** que se utiliza para la extensión. Este logotipo se rellena automáticamente desde el archivo *source.extension.vsixmanifest* si se proporciona.
- **Breve descripción** de lo que hace su extensión. Esta descripción se rellena automáticamente desde el archivo *source.extension.vsixmanifest*.
- **Información general** es un buen lugar para incluir capturas de pantalla e información detallada sobre lo que hace su extensión.
- **Las versiones compatibles** de Visual Studio le permiten elegir en qué versiones de Visual Studio funcionará la extensión. La extensión solo se instala en esas versiones.
- ****La edición de Visual Studio compatible** le permite elegir en qué ediciones de Visual Studio funcionará la extensión. La extensión solo se instala en esas ediciones.
- **Tipo**. Las extensiones más comunes son **Herramientas**.
- **Categorías**. Elige hasta tres que sean los más adecuados para tu extensión.
- **Las etiquetas** son palabras clave que ayudan a los usuarios a encontrar la extensión. Las etiquetas pueden ayudar a aumentar la relevancia de la búsqueda de las extensiones en Marketplace.
- **Categoría de precios** es el costo de su extensión.
- **El repositorio** de código fuente le permite compartir un vínculo a su código fuente con la comunidad.
- **Permitir Q&A para su extensión** permite a los usuarios dejar preguntas en la página de entrada de la extensión.

9. Haga clic en **Guardar & Cargar**. Esta opción le lleva de nuevo a la página de administración del editor. Su extensión aún no se ha publicado. Para publicar la extensión, haga clic con el botón derecho en la extensión y seleccione **Hacer público**. Puede ver cómo se verá la extensión en Marketplace seleccionando **Ver extensión**. Para los números de adquisición, haga clic en **Informes**. Para realizar cambios en su extensión, haga clic en **Editar**.



10. Después de hacer clic en **Hacer público**, la extensión ahora es pública. Busque la extensión en Visual Studio Marketplace.

Agregue usuarios adicionales para administrar su cuenta de editor

Marketplace admite la concesión de permisos de usuarios adicionales para acceder y administrar una cuenta de editor.

1. Vaya a la cuenta de editor a la que desea agregar usuarios adicionales.
2. Seleccione **Miembros** y haga clic en **Agregar**.

User name	Roles
 Stephanie Su	Owner
Email address	Select a role  
	Creator Reader Contributor Owner

3. A continuación, puede especificar la dirección de correo electrónico del usuario que desea agregar y conceder el nivel de acceso adecuado en **Seleccionar un rol**. Puede elegir entre las siguientes opciones:

- **Creador**: El usuario puede publicar extensiones, pero no puede ver ni administrar extensiones publicadas por otros usuarios.
- **Lector**: el usuario puede ver las extensiones, pero no puede publicar ni administrar extensiones.
- **Colaborador**: el usuario puede publicar y administrar extensiones, pero no puede editar la configuración del editor ni administrar el acceso.
- **Propietario**: El usuario puede publicar y administrar extensiones, editar la configuración del editor y administrar el acceso.

Instale la extensión desde Visual Studio Marketplace

Ahora que se ha publicado la extensión, instálela en Visual Studio y pruébela allí.

1. En Visual Studio, en el menú **Herramientas** , haga clic en **Extensiones y actualizaciones**.
2. Haga clic en **En línea** y, a continuación, busque **TestPublish**.
3. Haga clic en **Descargar**. A continuación, la extensión se programa para su instalación.
4. Para completar la instalación, cierre todas las instancias de Visual Studio.

Eliminación de la extensión

Puede quitar la extensión de Visual Studio Marketplace y del equipo.

Para quitar la extensión de Visual Studio Marketplace

1. Abra el sitio web de [Visual Studio Marketplace](#).
2. En la esquina superior derecha, haga clic en **Publicar** extensiones. Elija el publicador que utilizó para publicar **TestPublish**. Aparece la lista de **TestPublish**.
3. Haga clic con el botón derecho en la entrada de extensión y haga clic en **Quitar**. Se le pedirá que confirme si desea eliminar la extensión. Haga clic en **OK**.

Para eliminar la extensión del ordenador

1. En Visual Studio, en el menú **Herramientas** , haga clic en **Extensiones y actualizaciones**.
2. Seleccione **TestPublish** y, a continuación, haga clic en **Desinstalar**. A continuación, la extensión se programa para su desinstalación.
3. Para completar la desinstalación, cierre todas las instancias de Visual Studio.

Tutorial: Publicación de una extensión de Visual Studio a través de la línea de comandos

06/04/2020 • 13 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo publicar una extensión de Visual Studio en Visual Studio Marketplace mediante la línea de comandos. Al agregar la extensión a Marketplace, los desarrolladores pueden usar el cuadro de diálogo [Extensiones y actualizaciones](#) para buscar extensiones nuevas y actualizadas.

VsixPublisher.exe es la herramienta de línea de comandos para publicar extensiones de Visual Studio en Marketplace. Se puede tener acceso a ella desde `$-VSInstallDir-VSSDK-VisualStudioIntegration-Tools-Bin-VsixPublisher.exe`. Los comandos disponibles en esta herramienta son: **publish**, **createPublisher**, **deletePublisher**, **deleteExtension**, **login**, **logout**.

Comandos

Publicar

Publica una extensión en Marketplace. La extensión puede ser un vsix, un archivo exe/msi o un vínculo. Si la extensión ya existe con la misma versión, sobrescribirá la extensión. Si la extensión aún no existe, creará una nueva extensión.

OPCIONES DE COMANDO	DESCRIPCIÓN
carga útil (obligatorio)	Una ruta de acceso a la carga útil para publicar o un vínculo para usar como la "URL de información más".
publishManifest (obligatorio)	Ruta de acceso al archivo de manifiesto de publicación que se va a usar.
ignoreWarnings	Lista de advertencias que se deben omitir al publicar una extensión. Estas advertencias se muestran como mensajes de línea de comandos al publicar una extensión. (por ejemplo, "VSIXValidatorWarning01, VSIXValidatorWarning02")
personalAccessToken	Token de acceso personal (PAT) que se utiliza para autenticar al editor. Si no se proporciona, la PAT se adquiere de los usuarios que han iniciado sesión.

```
VsixPublisher.exe publish -payload "{path to vsix}" -publishManifest "{path to vs-publish.json}" - ignoreWarnings "VSIXValidatorWarning01,VSIXValidatorWarning02"
```

createPublisher

Crea un publicador en Marketplace. También registra el publicador en el equipo para acciones futuras (por ejemplo, eliminar/publicar una extensión).

OPCIONES DE COMANDO	DESCRIPCIÓN
displayName (obligatorio)	Nombre para mostrar del editor.
publisherName (obligatorio)	El nombre del editor (por ejemplo, el identificador).

OPCIONES DE COMANDO	DESCRIPCIÓN

personalAccessToken (obligatorio)	Token de acceso personal que se utiliza para autenticar al editor.
shortDescription	Una breve descripción del editor (no un archivo).
longDescription	Una descripción larga del editor (no un archivo).

```
VsixPublisher.exe createPublisher -publisherName "{Publisher Name}" -displayName "{Publisher Display Name}" -personalAccessToken "{Personal Access Token}"
```

deletePublisher

Elimina un editor en Marketplace.

OPCIONES DE COMANDO	DESCRIPCIÓN
publisherName (obligatorio)	El nombre del editor (por ejemplo, el identificador).
personalAccessToken (obligatorio)	Token de acceso personal que se utiliza para autenticar al editor.

```
VsixPublisher.exe deletePublisher -publisherName "{Publisher Name}" -personalAccessToken "{Personal Access Token}"
```

deleteExtension

Elimina una extensión de Marketplace.

OPCIONES DE COMANDO	DESCRIPCIÓN
extensionName (obligatorio)	El nombre de la extensión que se desea eliminar.
publisherName (obligatorio)	El nombre del editor (por ejemplo, el identificador).
personalAccessToken	Token de acceso personal que se utiliza para autenticar al editor. Si no se proporciona, el pat se adquiere de los usuarios que han iniciado sesión.

```
VsixPublisher.exe deleteExtension -extensionName "{Extension Name}" -publisherName "{Publisher Name}"
```

login

Registra un publicador en la máquina.

OPCIONES DE COMANDO	DESCRIPCIÓN
personalAccessToken (requerido)	Token de acceso personal que se utiliza para autenticar al editor.

OPCIONES DE COMANDO	DESCRIPCIÓN
publisherName (obligatorio)	El nombre del editor (por ejemplo, el identificador).
overwrite	Especifica que cualquier publicador existente debe sobrescribirse con el nuevo token de acceso personal.

```
VsixPublisher.exe login -personalAccessToken "{Personal Access Token}" -publisherName "{Publisher Name}"
```

logout

Cierra la sesión de un editor de la máquina.

OPCIONES DE COMANDO	DESCRIPCIÓN
publisherName (obligatorio)	El nombre del editor (por ejemplo, el identificador).
ignoreMissingPublisher	Especifica que la herramienta no debe error si el publicador especificado aún no ha iniciado sesión.

```
VsixPublisher.exe logout -publisherName "{Publisher Name}"
```

publishManifest archivo

El comando **publish** utiliza un archivo publishManifest. Representa todos los metadatos sobre la extensión que Marketplace necesita saber. Si la extensión que se carga es desde una extensión VSIX, la propiedad "identity" solo debe tener el conjunto "internalName". Esto se debe a que el resto de las propiedades "identity" se pueden generar a partir del archivo vsixmanifest. Si la extensión es un msi/exe o una extensión de vínculo, el usuario debe proporcionar los campos obligatorios en la propiedad "identity". El resto del manifiesto contiene información específica del Marketplace (por ejemplo, categorías, si Q&A está habilitada, etc.).

Ejemplo de archivo publishManifest de extensión VSIX:

```
{
  "$schema": "http://json.schemastore.org/vsix-publish",
  "categories": [ "build", "coding" ], // The categories of the extension. Between 1 and 3 categories are
  required.
  "identity": {
    "internalName": "MyVsixExtension" // If not specified, we try to generate the name from the display
    name of the extension in the vsixmanifest file.
    // Required if the display name is not the actual name of the
  extension.
  },
  "overview": "overview.md", // Path to the "readme" file that gets uploaded to the Marketplace.
  Required.
  "priceCategory": "free", // Either "free", "trial", or "paid". Defaults to "free".
  "publisher": "MyPublisherName", // The name of the publisher. Required.
  "private": false, // Specifies whether or not the extension should be public when
  uploaded. Defaults to false.
  "qna": true, // Specifies whether or not the extension should have a Q&A section.
  Defaults to true.
  "repo": "https://github.com/MyPublisherName/MyVsixExtension" // Not required.
}
```

Ejemplo de archivo publishManifest de MSI/EXE o LINK:

```
{
  "$schema": "http://json.schemastore.org/vsrx-publish",
  "categories": [ "build", "coding" ],
  "identity": {
    "description": "My extension.", // The description of the extension. Required for non-vsrx extensions.
    "displayName": "My Extension", // The display name of the extension. Required for non-vsrx extensions.
    "icon": "\\path\\to\\icon.ico", // The path to an icon file (can be relative to the json file
location). Required for non-vsrx extensions.
    "installTargets": [           // The installation targets for the extension. Required for non-vsrx
extensions.
      {
        "sku": "Microsoft.VisualStudio.Community",
        "version": "[10.0, 16.0)"
      }
    ],
    "internalName": "MyExtension",
    "language": "en-US",           // The default language id of the extension. Can be in the "1033" or
"en-US" format. Required for non-vsrx extensions.
    "tags": [ "tag1", "tag2" ],    // The tags for the extension. Not required.
    "version": "3.7.0",           // The version of the extension. Required for non-vsrx extensions.
    "vsrxId": "MyExtension",       // The vsrx id of the extension. Not required but useful for showing
updates to installed extensions.
  },
  "overview": "overview.md",
  "priceCategory": "free",
  "publisher": "MyPublisherName",
  "private": false,
  "qna": true,
  "repo": "https://github.com/MyPublisherName/MyVsrxExtension"
}
```

Archivos de activos

Se pueden proporcionar archivos de activos para incrustar cosas como imágenes en el archivo Léame. Por ejemplo, si una extensión tiene el siguiente documento de Markdown de "resumen":

```
TestExtension
...
This is test extension.
![Test logo](images/testlogo.png "Test logo")
```

Para resolver "images/testlogo.png" en el ejemplo anterior, un usuario puede proporcionar "assetFiles" en su manifiesto de publicación como se muestra a continuación:

```
{
  "assetFiles": [
    {
      "pathOnDisk": "\\path\\to\\logo.png",
      "targetPath": "images/logo.png"
    }
  ],
  // other required fields
}
```

Tutorial de publicación

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

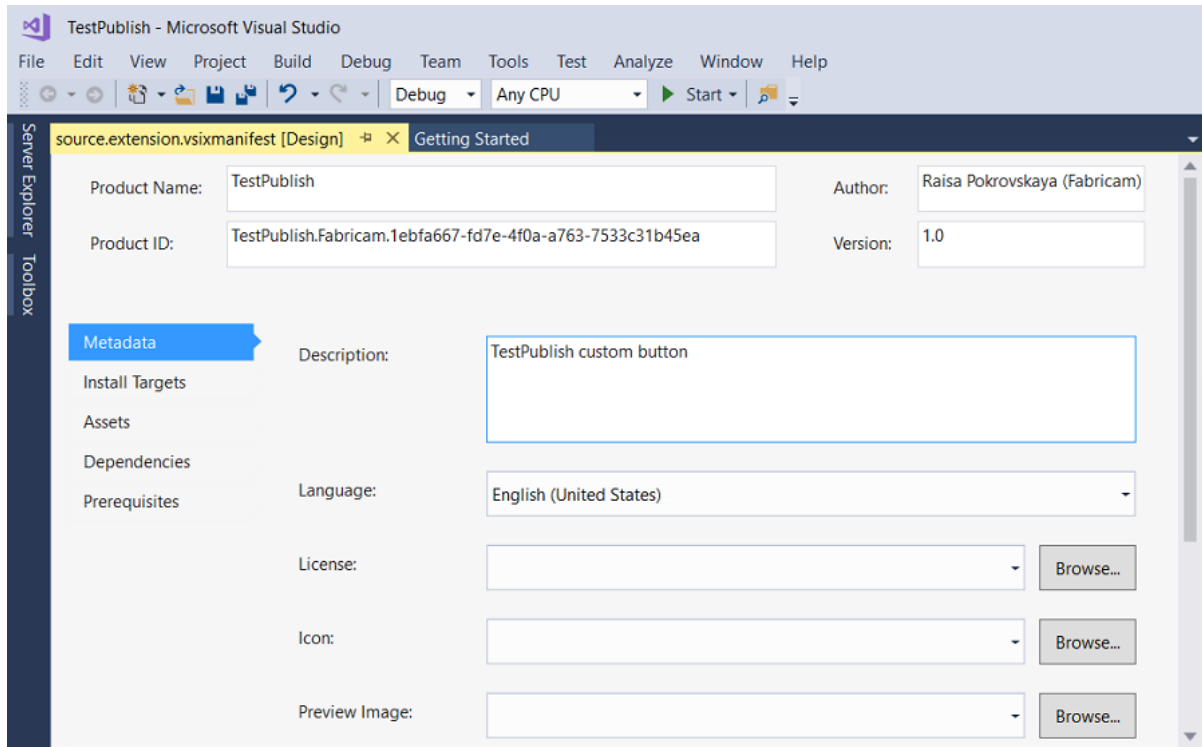
Crear una extensión de Visual Studio

En este caso, usaremos una extensión de VSPackage predeterminada, pero los mismos pasos son válidos para cada tipo de extensión.

1. Crear un VSPackage en C- denominado "TestPublish" que tiene un comando de menú. Para obtener más información, consulte [Creación de la primera extensión: Hello World](#).

Empaquetado de la extensión

1. Actualice la extensión vsixmanifest con la información correcta sobre el nombre, el autor y la versión del producto.



2. Cree la extensión en el modo **de versión**. Ahora, la extensión se empaquetará como un VSIX en la carpeta .bin.Release.
3. Puede hacer doble clic en VSIX para comprobar la instalación.

Pruebe la extensión

Antes de distribuir la extensión, compile y pruébela para asegurarse de que está instalada correctamente en la instancia experimental de Visual Studio.

1. En Visual Studio, inicie la depuración. para abrir una instancia experimental de Visual Studio.
2. En la instancia experimental, vaya al menú **Herramientas** y haga clic en **Extensiones y actualizaciones....** La extensión TestPublish debe aparecer en el panel central y estar habilitada.
3. En el menú **Herramientas**, asegúrese de ver el comando test.

Publique la extensión en Marketplace a través de la línea de comandos

1. Asegúrese de que ha creado la versión Release de la extensión y de que está actualizada.
2. Asegúrese de que ha creado archivos publishmanifest.json y overview.md.
3. Abra la línea de comandos y desplácese hasta el directorio \$-VSInstallDir-VSSDK-VisualStudioIntegration-Tools-Bin.
4. Para crear un nuevo editor, utilice el siguiente comando:

```
VsixPublisher.exe createPublisher -publisherName "TestVSIXPublisher" -displayName "Test VSIX Publisher"
-personalAccessToken "{Personal Access Token that is used to authenticate the publisher. If not
provided, the pat is acquired from the logged-in users.}"
```

5. En la creación correcta del publicador, verá el siguiente mensaje de línea de comandos:

```
Added 'Test VSIX Publisher' as a publisher on the Marketplace.
```

6. Puede comprobar el nuevo editor que creó navegando a [Visual Studio Marketplace](#)
7. Para publicar una nueva extensión, utilice el siguiente comando:

```
VsixPublisher.exe publish -payload "{Path to vsix file}" -publishManifest "{path to publishManifest
file}"
```

8. En la creación correcta del publicador, verá el siguiente mensaje de línea de comandos:

```
Uploaded 'MyVsixExtension' to the Marketplace.
```

9. Puede comprobar la nueva extensión que publicó navegando a [Visual Studio Marketplace](#)

Instale la extensión desde Visual Studio Marketplace

Ahora que se ha publicado la extensión, instálela en Visual Studio y pruébela allí.

1. En Visual Studio, en el menú **Herramientas**, haga clic en **Extensiones y actualizaciones....**
2. Haga clic en **En línea** y, a continuación, busque TestPublish.
3. Haga clic en **Descargar**. La extensión se programará para su instalación.
4. Para completar la instalación, cierre todas las instancias de Visual Studio.

Eliminación de la extensión

Puede quitar la extensión de Visual Studio Marketplace y del equipo.

Para eliminar la extensión del Marketplace a través de la línea de comandos

1. Si desea eliminar una extensión, utilice el siguiente comando:

```
VsixPublisher.exe deleteExtension -publisherName "TestVSIXPublisher" -extensionName "MyVsixExtension"
```

2. En la eliminación correcta de la extensión, verá el siguiente mensaje de línea de comandos:

```
Removed 'MyVsixExtension' from the Marketplace.
```

Para eliminar la extensión del ordenador

1. En Visual Studio, en el menú **Herramientas**, haga clic en **Extensiones y actualizaciones**.
2. Seleccionar "MyVsixExtension" y luego haga clic en **Desinstalar**. La extensión se programará para su desinstalación.
3. Para completar la desinstalación, cierre todas las instancias de Visual Studio.

Tutorial: Creación de un paquete de extensión

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un paquete de extensión es un conjunto de extensiones que se pueden instalar juntas. Los paquetes de extensión le permiten compartir fácilmente sus extensiones favoritas con otros usuarios o agrupar un conjunto de extensiones para un escenario determinado.

Prerrequisitos

A partir de Visual Studio 2015, el SDK de Visual Studio se incluye como una característica opcional en la instalación de Visual Studio. También puede instalar el SDK de VS más adelante. Para obtener más información, vea [Instalar el SDK de Visual Studio](#).

La característica Paquete de extensión está disponible a partir de Visual Studio 15.8 Preview 2.

Cree una extensión con una plantilla de elemento de paquete de extensión

La plantilla de elemento Paquete de extensión crea un paquete de extensión con un conjunto de extensiones que se pueden instalar juntas.

1. En el cuadro de diálogo **Nuevo proyecto**, busque "vsix" y seleccione **Proyecto VSIX**. En **Project name**, escriba "Test Extension Pack". Seleccione **Crear**.
2. En el **Explorador de soluciones**, haga clic con el botón secundario en el nodo del proyecto y seleccione **Agregar > nuevo elemento**. Vaya al nodo **Extensibilidad de Visual C-** y seleccione **Paquete de extensión**. Deje el nombre de archivo predeterminado (ExtensionPack1.cs).
3. ExtensionPack1.vsexst se agrega el archivo que contiene el siguiente código

```
{
  "id": "ExtensionPack1",
  "name": "ExtensionPack1",
  "description": "Read about creating extension packs at https://aka.ms/vsextpack",
  "version": "1.0.0.0",
  "extensions": [ // List of extensions that are included in the Extension Pack.
    {
      "vsixId": "41858b2d-ff0b-4a43-80b0-f1b2d6084935", // The vsix id of the extension you want to
include.
      "name": "AlignAssignments"
    },
    {
      "vsixId": "42374550-426a-400e-96f9-237682e8dea6",
      "name": "CopyAsHtml"
    }
  ]
}
```

4. El vsixid de la extensión que se va a incluir en el paquete de extensión se puede encontrar en [Visual Studio Marketplace](#). Busque la extensión que desea incluir y haga clic en **Copiar ID**. Puede actualizar el vsixid existente en el archivo anterior o agregar otra extensión a la lista.



Copy As Html

Microsoft DevLabs | 374,221 installs | ★★★★★ (9)

Adds support to copy the selected editor text to clipboard in HTML format.

Download

[Overview](#) [Q & A](#) [Rating & Review](#)

HTML Copy

Provides support for the HTML Clipboard format when cutting for copying code from the editor. This means there is no need to fix up the formatting of code when pasting it into a TFS work item form or another HTML based control. It is possible to modify the html format that gets copied to the clipboard. Users can customize the behavior in Tools > Options > Productivity Power Tools > HTML Copy.

Categories

[Tools](#) [Coding](#) [Programming Languages](#)

Tags

[HTML Copy](#) [Power Tools](#) [PPT](#) [Productivity](#)

Works with

Visual Studio 2017

Resources

[License](#)

[Copy ID](#)

5. Compile el proyecto y cargue la extensión en Marketplace. Consulte [Publicación de una extensión](#) de Visual Studio .

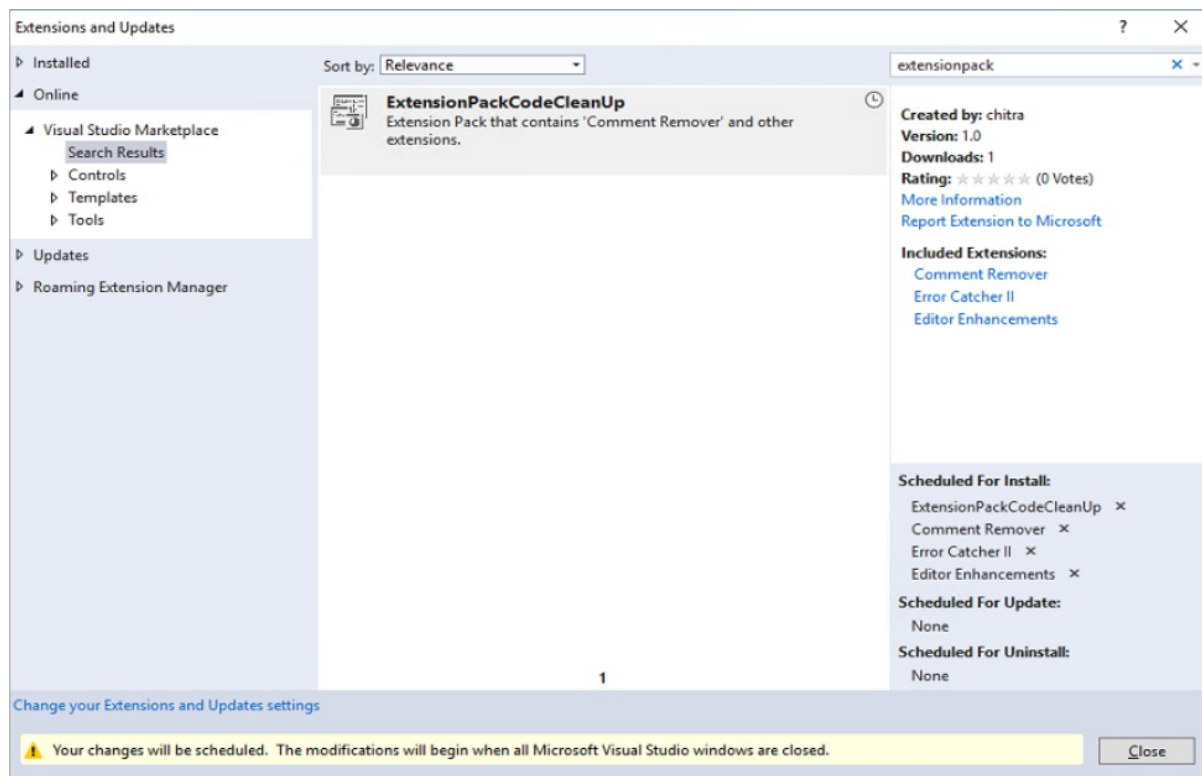
NOTE

Un paquete de extensión solo puede instalar extensiones que estén disponibles en la galería [de Visual Studio Marketplace](#) o [Private](#).

Instale el paquete de extensión desde Visual Studio Marketplace

Ahora que se ha publicado la extensión, instálela en Visual Studio y pruébela allí.

1. En Visual Studio, en el menú **Herramientas** , haga clic en **Extensiones y actualizaciones**.
1. En Visual Studio, en el menú **Extensiones** , haga clic en **Extensiones administradas**.
2. Haga clic en **En línea** y, a continuación, busque "Test Extension Pack".
3. Haga clic en **Descargar**. La extensión y su lista de extensiones incluidas en el paquete de extensión se programarán para su instalación.
4. A continuación se muestra una vista de descarga del paquete de extensión de ejemplo del cuadro de diálogo **Administrar extensiones**. Si prefiere instalar solo algunas de las extensiones incluidas en el paquete de extensiones, puede modificar la lista de extensiones en **Programado para la instalación**.



5. Para completar la instalación, cierre todas las instancias de Visual Studio.

Eliminación de la extensión

Para eliminar la extensión del equipo:

1. En Visual Studio, en el menú **Herramientas** , haga clic en **Extensiones y actualizaciones**.
1. En Visual Studio, en el menú **Extensiones** , haga clic en **Extensiones administradas**.
2. Seleccione **Probar paquete de extensión** y, a continuación, haga clic en **Desinstalar**. La extensión y su lista de extensiones incluidas en el paquete de extensión se programarán para su desinstalación.
3. Para completar la desinstalación, cierre todas las instancias de Visual Studio.

Cómo: Agregar una dependencia a un paquete VSIX

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede configurar una implementación de paquete VSIX que instale las dependencias que aún no estén presentes en el equipo de destino. Para ello, incluya las dependencias VSIX en el archivo *source.extension.vsixmanifest*.

Para agregar una dependencia

1. Abra el archivo *source.extension.vsixmanifest* en la vista **Diseño**. Vaya a la pestaña **Dependencias** y haga clic en **Nuevo**.
2. Para agregar una extensión instalada: en el cuadro de diálogo **Agregar nueva dependencia**, seleccione **Extensión instalada** y, a continuación, en **Nombre**, seleccione una extensión en la lista.
3. Para agregar otro VSIX que no esté instalado: en el cuadro de diálogo **Agregar nueva dependencia**, seleccione **Archivo** en el sistema de archivos y, a continuación, utilice el botón **Examinar** para seleccionar VSIX.

Requerir una versión específica de Visual Studio

Si la extensión requiere una versión específica de Visual Studio 2017, por ejemplo, depende de una característica publicada en 15.3, puede especificar el número de compilación en VSIX **InstallationTarget**. Por ejemplo, la versión 15.3 tiene un número de compilación de '15.0.26730.3'. Puede ver la asignación de versiones para generar números [aquí](#). Tenga en cuenta que el uso del número de versión '15.3' no funcionará correctamente.

Si la extensión requiere 15.3 o superior, declararía la **versión InstallationTarget** como [15.0.26730.3, 16.0):

```
<Installation>
  <InstallationTarget Id="Microsoft.VisualStudio.Community" Version="[15.0.26730.3, 16.0)" />
</Installation>
```

VSIXInstaller detectará versiones anteriores de Visual Studio e informará al usuario de que se requiere una actualización posterior.

Vea también

- [Referencia del esquema de extensión VSIX 1.0](#)
- [Anatomía de un paquete VSIX](#)
- [Preparar extensiones para la implementación de Windows Installer](#)

Anatomía de un paquete VSIX

06/04/2020 • 6 minutes to read • [Edit Online](#)

Un paquete VSIX es un archivo *.vsix* que contiene una o varias extensiones de Visual Studio, junto con los metadatos que Visual Studio usa para clasificar e instalar las extensiones. Esos metadatos se encuentran en el manifiesto VSIX y en el archivo *[Content_Types].xml*. Un paquete VSIX también puede contener uno o varios archivos *Extension.vsixlangpack* para proporcionar texto de instalación localizado y puede contener paquetes VSIX adicionales para instalar dependencias.

El formato de paquete VSIX sigue el estándar Open Packaging Conventions (OPC). El paquete contiene archivos binarios y archivos auxiliares, junto con un archivo *[Content_Types].xml* y un archivo de manifiesto *.vsix*. Un paquete VSIX puede contener la salida de varios proyectos, o incluso varios paquetes que tienen sus propios manifiestos.

NOTE

Los nombres de los archivos incluidos en los paquetes VSIX no deben incluir espacios ni caracteres reservados en identificadores uniformes de recursos (URI), tal como se define en [\[RFC2396\]](#).

El manifiesto VSIX

El manifiesto VSIX contiene información sobre la extensión que se va a instalar y sigue el esquema VSX. Para obtener más información, vea Referencia 1.0 del esquema de [extensión VSIX](#). Para obtener un manifiesto VSIX de ejemplo, vea [Elemento PackageManifest \(elemento raíz, esquema VSX\)](#).

El manifiesto VSIX `extension.vsixmanifest` debe tener nombre cuando se incluye en un archivo *.vsix**.

El contenido

Un paquete VSIX puede contener plantillas, elementos de cuadro de herramientas, VSPackages o cualquier otro tipo de extensión compatible con Visual Studio.

Paquetes de idioma

Un paquete VSIX puede contener una o más archivos *Extension.vsixlangpack* para proporcionar texto localizado durante la instalación. Para obtener más información, consulte Localización de [paquetes VSIX](#).

Dependencias y referencias

Un paquete VSIX puede contener otros paquetes VSIX como referencias. Cada uno de estos otros paquetes debe incluir su propio manifiesto VSIX.

Si un usuario intenta instalar una extensión que tiene dependencias, el instalador comprueba que los ensamblados necesarios están instalados en el sistema de usuario. Si no se encuentran los ensamblados necesarios, **Extensiones y actualizaciones** muestra una lista de los ensamblados que faltan.

Si el manifiesto de extensión incluye uno o varios elementos [Reference](#), **Extensiones y actualizaciones** compara el manifiesto de cada referencia con las extensiones instaladas en el sistema e instala la extensión a la que se hace referencia si aún no está instalada. Si se instala una versión anterior de una extensión a la que se hace referencia, la versión más reciente la reemplaza.

Si un proyecto de una solución de varios proyectos incluye una referencia a otro proyecto de la misma solución, el paquete VSIX incluye las dependencias de ese proyecto. Puede invalidar este comportamiento haciendo clic en la referencia del proyecto interno y, a `BuiltProjectOutputGroup` continuación, en la ventana **Propiedades**, estableciendo la propiedad Grupos de salida incluidos en **VSIX en** .

Para incluir archivos DLL satélite de ensamblados a `SatelliteDllsProjectOutputGroup` los que se hace referencia en el paquete VSIX, agregue a la propiedad Grupos de **salida incluidos en VSIX**.

Ubicación de la instalación

Durante la instalación, **Extensiones y actualizaciones** busca el contenido del paquete VSIX en una carpeta en *%LocalAppData%-Microsoft-VisualStudio-14.0-Extensions*.

De forma predeterminada, la instalación solo se aplica al usuario actual, porque *%LocalAppData%* es un directorio específico del usuario. Sin embargo, si establece el elemento `True` `AllUsers` del manifiesto en , la extensión se instalará en .. \VisualStudioInstallationFolder.Common7\IDE\Extensions y estará disponible para todos los usuarios del equipo.

[Content_Types].xml

El archivo *[Content_Types].xml* identifica los tipos de archivo en el archivo .vsix expandido. Visual Studio usa este archivo durante la instalación del paquete, pero no instala el archivo en sí. Para obtener más información acerca de este archivo, vea [La estructura del archivo \[Content_types\].xml](#).

El estándar Open Packaging Conventions (OPC) requiere un archivo *[Content_Types].xml*. Para obtener más información acerca de OPC, consulte [OPC: Un nuevo estándar para empaquetar](#) los datos en el sitio Web de MSDN.

Referencia del esquema de extensión VSIX 2.0

06/04/2020 • 20 minutes to read • [Edit Online](#)

Un archivo de manifiesto de implementación VSIX describe el contenido de un paquete VSIX. El formato de archivo se rige por un esquema. La versión 2.0 de este esquema admite la adición de tipos y atributos personalizados. El esquema del manifiesto es extensible. El cargador de manifiestos omite los elementos y atributos XML que no entiende.

IMPORTANT

Visual Studio 2015 puede cargar archivos VSIX en los formatos de Visual Studio 2010, Visual Studio 2012 o Visual Studio 2013.

Esquema de manifiesto del paquete

El elemento raíz del archivo `<PackageManifest>` XML de manifiesto es . Tiene un único `Version` atributo, que es la versión del formato de manifiesto. Si se realizan cambios importantes en el formato, se cambia el formato de versión. En este artículo se describe la versión 2.0 del `Version` formato de manifiesto, que se especifica en el manifiesto estableciendo el atributo en el valor de `Version="2.0"`.

Elemento PackageManifest

Dentro `<PackageManifest>` del elemento raíz, puede utilizar los siguientes elementos:

- `<Metadata>` - Metadatos e información publicitaria sobre el propio paquete. Solo `Metadata` se permite un elemento en el manifiesto.
- `<Installation>` - Esta sección define la forma en que se puede instalar este paquete de extensión, incluidas las STU de aplicación en las que se puede instalar. Solo se `Installation` permite un único elemento en el manifiesto. Un manifiesto debe `Installation` tener un elemento o este paquete no se instalará en ninguna SKU.
- `<Dependencies>` - Aquí se define una lista opcional de dependencias para este paquete.
- `<Assets>` - Esta sección contiene todos los activos contenidos en este paquete. Sin esta sección, este paquete no saque ningún contenido.
- `<AnyElement>*` - El esquema de manifiesto es lo suficientemente flexible como para permitir cualquier otro elemento. Los elementos secundarios no reconocidos por el cargador de manifiestos se exponen en la API de Extension Manager como objetos `XmlElement` adicionales. Con estos elementos secundarios, las extensiones VSIX pueden definir datos adicionales en el archivo de manifiesto al que el código que se ejecuta en Visual Studio puede tener acceso en tiempo de ejecución. See [Microsoft.VisualStudio.ExtensionManager.IExtension.AdditionalElements](#).

Elemento metadatos

Esta sección son los metadatos sobre el paquete, su identidad e información publicitaria. `<Metadata>` contiene los siguientes elementos:

- `<Identity>` - Define la información de identificación para este paquete e incluye los siguientes atributos:
 - `Id` - Este atributo debe ser un identificador único para el paquete elegido por su autor. El nombre debe calificarse de la misma manera que los tipos CLR tienen espacio de nombres:

Company.Product.Feature.Name. El `Id` atributo está limitado a 100 caracteres.

- `Version` - Define la versión de este paquete y su contenido. Este atributo sigue el formato de control de versiones del ensamblado CLR: Major.Minor.Build.Revision (1.2.40308.00). Un paquete con un número de versión superior se considera actualizaciones del paquete y se puede instalar sobre la versión instalada existente.
- `Language` - Este atributo es el idioma predeterminado para el paquete y corresponde a los datos textuales de este manifiesto. Este atributo sigue la convención de código de configuración regional de CLR para ensamblados de recursos, por ejemplo: en-us, en, fr-fr. Puede especificar `neutral` que se declare una extensión independiente del lenguaje que se ejecutará en cualquier versión de Visual Studio. El valor predeterminado es `neutral`.
- `Publisher` - Este atributo identifica al editor de este paquete, ya sea una empresa o un nombre individual. El `Publisher` atributo está limitado a 100 caracteres.
- `<DisplayName>` - Este elemento especifica el nombre de paquete fácil de usar que se muestra en la interfaz de usuario del Administrador de extensiones. El `DisplayName` contenido está limitado a 50 caracteres.
- `<Description>` - Este elemento opcional es una breve descripción del paquete y su contenido que se muestra en la interfaz de usuario de Extension Manager. El `Description` contenido puede contener cualquier texto que desee, pero está limitado a 1000 caracteres.
- `<MoreInfo>` - Este elemento opcional es una dirección URL a una página en línea que contiene una descripción completa de este paquete. El protocolo debe especificarse como http.
- `<License>` - Este elemento opcional es una ruta relativa a un archivo de licencia (.txt, .rtf) contenido en el paquete.
- `<ReleaseNotes>` - Este elemento opcional es una ruta relativa a un archivo de notas de la versión contenido en el paquete (.txt, .rtf) o bien una dirección URL a un sitio web que muestra las notas de la versión.
- `<Icon>` - Este elemento opcional es una ruta relativa a un archivo de imagen (png, bmp, jpeg, ico) contenido en el paquete. La imagen del icono debe ser de 32x32 píxeles (o se reducirá a ese tamaño) y aparece en la interfaz de usuario de vista de lista. Si `Icon` no se especifica ningún elemento, la interfaz de usuario usa un valor predeterminado.
- `<PreviewImage>` - Este elemento opcional es una ruta relativa a un archivo de imagen (png, bmp, jpeg) contenido en el paquete. La imagen de vista previa debe ser de 200 x 200 píxeles y se muestra en la interfaz de usuario de detalles. Si `PreviewImage` no se especifica ningún elemento, la interfaz de usuario usa un valor predeterminado.
- `<Tags>` - Este elemento opcional enumera etiquetas de texto delimitadas por punto y coma adicionales que se utilizan para sugerencias de búsqueda. El `Tags` elemento está limitado a 100 caracteres.
- `<GettingStartedGuide>` - Este elemento opcional es una ruta relativa a un archivo HTML o una dirección URL a un sitio web que contiene información sobre cómo utilizar la extensión o el contenido dentro de este paquete. Esta guía se inicia como parte de una instalación.
- `<AnyElement>*` - El esquema de manifiesto es lo suficientemente flexible como para permitir cualquier otro elemento. Los elementos secundarios que no reconoce el cargador de manifiestos se exponen como una lista de objetos XmlElement. Con estos elementos secundarios, las extensiones VSIX pueden definir datos adicionales en el archivo de manifiesto y enumerarlos en tiempo de ejecución.

Elemento de instalación

En esta sección se define la forma en que se puede instalar este paquete y las STU de aplicación en las que se puede instalar. Esta sección contiene los siguientes atributos:

- **Experimental** - Establezca este atributo en true si tiene una extensión que está instalada actualmente para todos los usuarios, pero está desarrollando una versión actualizada en el mismo equipo. Por ejemplo, si ha instalado MyExtension 1.0 para todos los usuarios, pero desea depurar MyExtension 2.0 en el mismo equipo, establezca Experimental "true". Este atributo está disponible en Visual Studio 2015 Update 1 y versiones posteriores.
- **Scope** - Este atributo puede tomar el valor "Global" o "ProductExtension":
 - "Global" especifica que la instalación no tiene el ámbito de una SKU específica. Por ejemplo, este valor se utiliza cuando se instala un SDK de extensión.
 - "ProductExtension" especifica que se instala una extensión VSIX tradicional (versión 1.0) con ámbito a SKUs de Visual Studio individuales. Este es el valor predeterminado.
- **AllUsers** - Este atributo opcional especifica si este paquete se instalará para todos los usuarios. De forma predeterminada, este atributo es false, que especifica que el paquete es por usuario. (Al establecer este valor en true, el usuario de instalación debe elevar al nivel de privilegios administrativos para instalar el VSIX resultante.
- **InstalledByMsi** - Este atributo opcional especifica si este paquete está instalado por un MSI. Los paquetes instalados por un MSI son instalados y administrados por MSI (Programas y características) y no por el Administrador de extensiones de Visual Studio. De forma predeterminada, este atributo es false, que especifica que un MSI no instala el paquete.
- **SystemComponent** - Este atributo opcional especifica si este paquete debe considerarse un componente del sistema. Los componentes del sistema no se muestran en la interfaz de usuario de Extension Manager y no se pueden actualizar. De forma predeterminada, este atributo es false, que especifica que el paquete no es un componente del sistema.
- **AnyAttribute*** - **Installation** El elemento acepta un conjunto abierto de atributos que se expondrán en tiempo de ejecución como un diccionario de pares nombre-valor.
- **<InstallationTarget>** -Este elemento controla la ubicación donde el instalador de VSIX instala el paquete. Si el valor **Scope** del atributo es "ProductExtension", el paquete debe tener como destino una SKU, que ha instalado un archivo de manifiesto como parte de su contenido para anunciar su disponibilidad a las extensiones. El **<InstallationTarget>** elemento tiene los **Scope** siguientes atributos cuando el atributo tiene el valor explícito o predeterminado "ProductExtension":
 - **Id** - Este atributo identifica el paquete. El atributo sigue la convención de espacio de nombres: Company.Product.Feature.Name. El **Id** atributo solo puede contener caracteres alfanuméricos y está limitado a 100 caracteres. Valores esperados:
 - Microsoft.VisualStudio.IntegratedShell
 - Microsoft.VisualStudio.Pro
 - Microsoft.VisualStudio.Premium
 - Microsoft.VisualStudio.Ultimate
 - Microsoft.VisualStudio.VWDExpress
 - Microsoft.VisualStudio.VPDEExpress
 - Microsoft.VisualStudio.VSWinExpress
 - Microsoft.VisualStudio.VSLS
 - My.Shell.App

- **Version** - Este atributo especifica un intervalo de versiones con las versiones mínimas y máximas admitidas de esta SKU. Un paquete puede detallar las versiones de las STU que admite. La notación del rango de versiones es [10.0 - 11.0], donde
 - [- versión mínima incluida.
 -] - versión máxima incluida.
 - (- versión mínima exclusiva.
 -) - versión máxima exclusiva.
 - Versión única: solo la versión especificada.

IMPORTANT

La versión 2.0 del esquema VSIX se introdujo en Visual Studio 2012. Para usar este esquema, debe tener Visual Studio 2012 o posterior instalado en el equipo y usar VSIXInstaller.exe que forma parte de ese producto. Puede tener como destino versiones anteriores de Visual Studio con un VSIXInstaller de Visual Studio 2012 o posterior, pero solo mediante las versiones posteriores del instalador.

Los números de versión de Visual Studio 2017 se pueden encontrar en los números de compilación y las fechas de lanzamiento de [Visual Studio](#).

Al expresar la versión de las versiones de Visual Studio 2017, la versión secundaria siempre debe ser 0. Por ejemplo, Visual Studio 2017 versión 15.3.26730.0 debe expresarse como [15.0.26730.0,16.0). Esto solo es necesario para Visual Studio 2017 y números de versión posteriores.

- **AnyAttribute*** - **<InstallationTarget>** El elemento permite un conjunto abierto de atributos que se expone en tiempo de ejecución como un diccionario de pares nombre-valor.

Elemento Dependencias

Este elemento contiene una lista de dependencias que declara este paquete. Si se especifica alguna dependencia, esos paquetes (identificados por su **Id**) deben haberse instalado antes.

- **<Dependency>** elemento: este elemento secundario tiene los siguientes atributos:
 - **Id** - Este atributo debe ser un identificador único para el paquete dependiente. Este valor de **<Metadata><Identity>Id** identidad debe coincidir con el atributo de un paquete del que depende este paquete. El **Id** atributo sigue la convención de espacio de nombres: Company.Product.Feature.Name. El atributo solo puede contener caracteres alfanuméricos y está limitado a 100 caracteres.
 - **Version** - Este atributo especifica un intervalo de versiones con las versiones mínimas y máximas admitidas de esta SKU. Un paquete puede detallar las versiones de las STU que admite. La notación del rango de versiones es [12.0, 13.0], donde:
 - [- versión mínima incluida.
 -] - versión máxima incluida.
 - (- versión mínima exclusiva.
 -) - versión máxima exclusiva.
 - Versión única: solo la versión especificada.
 - **DisplayName** - Este atributo es el nombre para mostrar del paquete dependiente, que se utiliza en

elementos de interfaz de usuario como cuadros de diálogo y mensajes de error. El atributo es opcional a menos que MSI instale el paquete dependiente.

- **Location** - Este atributo opcional especifica la ruta de acceso relativa dentro de este VSIX a un paquete VSIX anidado o una dirección URL a la ubicación de descarga de la dependencia. Este atributo se utiliza para ayudar al usuario a localizar el paquete de requisitos previos.
- **AnyAttribute*** - **Dependency** El elemento acepta un conjunto abierto de atributos que se expondrán en tiempo de ejecución como un diccionario de pares nombre-valor.

Elemento Activos

Este elemento contiene **<Asset>** una lista de etiquetas para cada extensión o elemento de contenido que expone este paquete.

- **<Asset>** - Este elemento contiene los siguientes atributos y elementos:
 - **Type** - Tipo de extensión o contenido representado por este elemento. Cada **<Asset>** elemento debe **Type** tener un **<Asset>** único, **Type** pero varios elementos pueden tener el mismo. Este atributo debe representarse como un nombre completo, según las convenciones de espacio de nombres. Los tipos conocidos son:
 1. Microsoft.VisualStudio.VsPackage
 2. Microsoft.VisualStudio.MefComponent
 3. Microsoft.VisualStudio.ToolboxControl
 4. Microsoft.VisualStudio.Samples
 5. Microsoft.VisualStudio.ProjectTemplate
 6. Microsoft.VisualStudio.ItemTemplate
 7. Microsoft.VisualStudio.Assembly

Puede crear sus propios tipos y darles nombres únicos. En tiempo de ejecución dentro de Visual Studio, el código puede enumerar y tener acceso a estos tipos personalizados a través de la API de Extension Manager.

- **Path** - la ruta relativa al archivo o carpeta dentro del paquete que contiene el recurso.
- **TargetVersion** - el rango de versiones al que se aplica el activo dado. Se utiliza para el envío de varias versiones de activos a diferentes versiones de Visual Studio. Requiere Visual Studio 2017.3 o posterior para tener efecto.
- **AnyAttribute*** - Un conjunto abierto de atributos que se expone en tiempo de ejecución como un diccionario de pares nombre-valor.

<AnyElement>* - Se permite cualquier **<Asset>** contenido estructurado entre una etiqueta de inicio y fin. Todos los elementos se exponen como una lista de XmlElement objetos. Las extensiones VSIX pueden definir metadatos específicos de tipo estructurado en el archivo de manifiesto y enumerarlos en tiempo de ejecución.

Ejemplo de manifiesto

```

<?xml version="1.0" encoding="utf-8"?>
<PackageManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011"
xmlns:d="http://schemas.microsoft.com/developer/vsx-schema-design/2011">
  <Metadata>
    <Identity Id="00000000-0000-0000-0000-000000000000" Version="1.0" Language="en-US" Publisher="Company" />
    <DisplayName>Test Package</DisplayName>
    <Description>Information about my package</Description>
    <MoreInfo>http://www.fabrikam.com/Extension1/</MoreInfo>
    <License>eula.rtf</License>
    <ReleaseNotes>notes.txt</ReleaseNotes>
    <Icon>Images\icon.png</Icon>
    <PreviewImage>Images\preview.png</PreviewImage>
  </Metadata>
  <Installation InstalledByMsi="false" AllUsers="false" SystemComponent="false" Scope="ProductExtension">
    <InstallationTarget Id="Microsoft.VisualStudio.Pro" Version="[11.0, 12.0]" />
  </Installation>
  <Dependencies>
    <Dependency Id="Microsoft.Framework.NDP" DisplayName="Microsoft .NET Framework" d:Source="Manual"
Version="[4.5,)" />
    <Dependency Id="Microsoft.VisualStudio.MPF.12.0" DisplayName="Visual Studio MPF 12.0"
d:Source="Installed" Version="[12.0]" />
  </Dependencies>
  <Assets>
    <Asset Type="Microsoft.VisualStudio.VsPackage" d:Source="Project" d:ProjectName="%CurrentProject%"
Path="|%CurrentProject%;PkgdefProjectOutputGroup|" />
  </Assets>
</PackageManifest>

```

Vea también

- [Enviar extensiones de Visual Studio](#)

Estructura del archivo [Content_types].xml

06/04/2020 • 3 minutes to read • [Edit Online](#)

Contiene información sobre los tipos de contenido de un paquete VSIX. Visual Studio usa el archivo [Content_Types].xml para instalar el paquete, pero no instala el archivo en sí.

NOTE

Aunque este tema solo se aplica a los archivos [Content_Type].xml que se usan en paquetes VSIX, el tipo de archivo [Content_Types].xml forma parte del estándar *Open Packaging Conventions (OPC)*. Para obtener más información, consulte [OPC: un nuevo estándar para empaquetar](#) los datos en el sitio Web de MSDN.

Atributos y elementos

En las secciones siguientes se describe el elemento raíz y sus atributos y elementos secundarios.

Elemento raíz

ELEMENTO	DESCRIPCIÓN
<code>Types</code>	Contiene elementos secundarios que enumeran los tipos de archivo en el paquete VSIX.

Atributos

ATRIBUTO	DESCRIPCIÓN
<code>Xmlns</code>	(Requerido.) La ubicación del esquema utilizado para este archivo [Content_Types].xml.

•Nombre del atributo ? Atributo

VALUE	DESCRIPCIÓN
<code>http://schemas.openformats.org/package/2006/content-types</code>	La ubicación del esquema de tipos de contenido.

Elementos secundarios

El elemento `Types` puede contener cualquier número de elementos `Default`.

ELEMENTO	DESCRIPCIÓN
<code>Default</code>	Describe un tipo de contenido en el paquete VSIX. Cada tipo de archivo del <code>Default</code> paquete debe tener su propio elemento.

Atributos

ATRIBUTO	DESCRIPCIÓN

ATRIBUTO	DESCRIPCIÓN
<code>Extension</code>	La extensión de nombre de archivo de un archivo en el paquete VSIX.
<code>ContentType</code>	Describe el tipo de contenido asociado a la extensión de nombre de archivo.

•Nombre del atributo ? Atributo

Visual Studio reconoce `ContentType` los siguientes `Extension` valores para los tipos asociados.

EXTENSIÓN	CONTENTTYPE
txt	text/plain
pkgdef	text/plain
Xml	text/xml
vsixmanifest	text/xml
htm o html	text/html
Rtf	aplicación/rtf
pdf	aplicación/pdf
GIF	image/gif
jpg o jpeg	imagen/jpg
tiff	image/tiff
vsix	aplicación/zip
zip	aplicación/zip
dll	application/octet-stream
todos los demás tipos de archivos	application/octet-stream

Ejemplo

Descripción

El siguiente archivo [Content_Types].xml describe un paquete VSIX típico.

Código

```
<?xml version="1.0" encoding="utf-8" ?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="vsixmanifest" ContentType="text/xml" />
  <Default Extension="dll" ContentType="application/octet-stream" />
  <Default Extension="png" ContentType="application/octet-stream" />
  <Default Extension="txt" ContentType="text/plain" />
  <Default Extension="pkgdef" ContentType="text/plain" />
</Types>
```

Vea también

- [Anatomía de un paquete VSIX](#)
- [Referencia del esquema de extensión 1.0 de VSIX](#)
- [OPC: Un nuevo estándar para empaquetar sus datos](#)

Diseñador de manifiestos de VSIX

06/04/2020 • 10 minutes to read • [Edit Online](#)

Modifica un archivo de manifiesto de paquete VSIX, que establece el comportamiento de instalación de una extensión de Visual Studio.

El Diseñador de **manifiestos VSIX** se asigna al esquema VSIX subyacente. Cada elemento del esquema se puede establecer mediante un control correspondiente en el diseñador. Para obtener más información sobre el esquema, vea Referencia del esquema de [extensión 2.0](#) de VSIX.

Para abrir el Diseñador de **manifiestos VSIX**, busque un archivo *source.extension.vsixmanifest* en el **Explorador** de soluciones y abra el archivo. Si el archivo no contiene XML válido, el diseñador de manifiestos no se abrirá.

NOTE

El archivo *source.extension.vsixmanifest* se envía a *extension.vsixmanifest* cuando se compila el paquete.

Lista de UIElement

El Diseñador de **manifiestos VSIX** contiene cuatro secciones que corresponden a estos elementos de nivel superior del esquema:

- Metadatos
- Objetivos de instalación
- Recursos
- Dependencias

El área de encabezado contiene los siguientes controles.

Nombre del producto Describe el nombre de la extensión.

ID de producto Especifica la información de identificación única para este paquete.

Autor Especifica el nombre del autor de la extensión.

Versión Especifica el número de versión de la extensión.

La pestaña **Metadatos** contiene los siguientes controles.

Descripción Proporciona una descripción de texto de la extensión, que se mostrará en el **Administrador de extensiones**.

Idioma Especifica el idioma predeterminado del paquete, que corresponde a los datos textuales del manifiesto. El `Language` atributo sigue la convención de código de configuración regional de Common Language Runtime (CLR) para ensamblados de recursos, por ejemplo, en-us, en, fr-fr. De forma predeterminada, el valor es neutro, lo que significa que el paquete se ejecutará en cualquier versión de idioma de Visual Studio.

Licencia Especifica el archivo de texto que contiene la licencia de usuario, si hay una.

Icono Especifica el archivo de gráficos (*.png*, *.bmp*, *.jpeg*, *.ico*) que contiene el icono que se mostrará en el

Administrador de **extensiones**, si hay un icono presente. La imagen del icono debe ser de 32 x 32 píxeles o se cambia el tamaño a esas dimensiones. Si no se especifica ningún icono, **Extension Manager** utiliza un icono predeterminado.

Vista previa de la imagen Especifica el archivo de gráficos (*.png, .bmp, .jpeg, .ico*) que contiene la imagen de vista previa que se mostrará en el Administrador de **extensiones**, si hay una imagen de vista previa presente. La imagen de vista previa debe tener 200 x 200 píxeles. Si no se especifica ninguna imagen de vista previa, **Extension Manager** utiliza una imagen predeterminada.

Etiquetas Añade etiquetas de texto que se utilizarán para las sugerencias de búsqueda.

Notas de la versión Especifica un archivo (*.txt, .rtf*) que contiene notas de la versión. También utiliza la dirección URL de un sitio web que muestra las notas de la versión.

Guía de introducción Especifica un archivo (*.txt, .rtf*) que contiene información sobre cómo utilizar la extensión o el contenido del paquete VSIX. Esta guía aparece cuando se completa la instalación de la extensión. También toma la dirección URL de un sitio web que muestra la guía.

Más URL de información Especifica la dirección URL de un sitio web que contiene información adicional sobre el producto.

La pestaña **Destinos de instalación** contiene los siguientes controles.

Tipo de instalación Enumera el SDK de extensión y **extensión de Visual Studio** como tipos de instalación de destino. Las opciones varían en función del tipo que elija.

Extensión de Visual Studio Enumera los elementos **InstallationTarget** que describen cómo se puede instalar el paquete y en qué productos de Visual Studio se puede instalar esta extensión. Cada producto se identifica por separado por nombre y una versión o rango de versiones. Los productos se pueden agregar a la lista, modificar y eliminar. El nombre y la versión de un producto corresponden a los atributos **Id** y **Version** del elemento **InstallationTarget** asociado.

El rango de **versiones** es [12.0, 14.0] y utiliza la siguiente notación:

- [- versión mínima incluida
-] - versión máxima incluida
- (- versión mínima exclusiva
-) - versión máxima exclusiva
- Versión única - sólo la versión especificada

SDK de extensión Especifica una instalación global que no tiene el ámbito de un producto y una versión específicos. **Identificador** de plataforma de destino es la plataforma, como "Windows", a la que se dirige. La **versión** de plataforma de destino es la versión, como 8.0, de la plataforma de destino. **SDK Name** y **SDK Version** son el nombre y el número de versión del SDK, respectivamente.

Este VSIX se instala para todos los usuarios (requiere elevación en la instalación) Si selecciona esta casilla de verificación, la extensión se instala para todos los usuarios; de lo contrario, se instala sólo para el usuario actual.

Este VSIX es instalado por Windows Installer Si selecciona esta casilla de verificación, la extensión la instala Windows Installer (archivo *.msi*); de lo contrario, se instala como un paquete VSIX típico (archivo *.vsix*).

La pestaña **Activos** contiene los siguientes controles.

Lista de activos Enumera los elementos **Asset** que describen los elementos de extensión o contenido que

expone este paquete. Cada extensión o elemento de contenido se enumera por separado por origen, tipo y ruta de acceso. Las extensiones y los elementos de contenido se pueden agregar a la lista, modificar y eliminar. El tipo y la ruta de acceso `Type` de `Path` una extensión `Asset` o elemento de contenido corresponde a los atributos y del elemento asociado. Se conocen los siguientes tipos:

- Microsoft.VisualStudio.Package
- Microsoft.VisualStudio.MefComponent
- Microsoft.VisualStudio.ToolboxControl
- Microsoft.VisualStudio.Samples
- Microsoft.VisualStudio.ProjectTemplate
- Microsoft.VisualStudio.ItemTemplate
- Microsoft.VisualStudio.Assembly
- Microsoft.ExtensionSDK

Para agregar o editar un activo, debe especificar el tipo de activo, si el activo es un proyecto de la solución actual o un archivo en el sistema de archivos y el nombre del proyecto. También puede especificar el nombre de la carpeta en la que se va a incrustar.

También puede crear sus propios tipos y darles nombres únicos.

La pestaña **Dependencias** contiene los siguientes controles.

Nombre, Origen y Rango de versiones Enumera los elementos Dependency de este paquete, que son otros paquetes de los que depende este paquete. Si se especifica un paquete de dependencia, debe instalarse antes de instalar este paquete; de lo contrario, este paquete debe instalarlo.

Los paquetes de dependencia se especifican mediante identificador, nombre, intervalo de versiones, origen y cómo se va a resolver la dependencia. Cada paquete de dependencia se enumera por separado por nombre, versión y origen. Los paquetes de dependencia se pueden agregar a la lista, modificar y eliminar.

El identificador debe `ID` coincidir con el atributo de los metadatos del paquete de dependencia. El origen puede ser un proyecto en la solución actual, una extensión instalada actualmente o un archivo. La configuración **Cómo se resuelve por dependencia** puede ser la ruta de acceso relativa de un paquete anidado o la dirección URL de la ubicación de descarga de la dependencia. El identificador, la versión y la resolución `Id` `Version` del `Location` paquete de `Dependency` dependencia scorresponden a los atributos `,` `,` y del elemento asociado.

Vea también

- [Referencia del esquema de extensión VSIX 2.0](#)
- [Anatomía de un paquete VSIX](#)

Adaptación de paquetes VSIX

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede localizar un paquete VSIX creando un archivo *Extension.vsixlangpack* para cada idioma de destino y, a continuación, colgándolos en la carpeta correcta. Cuando se instala un paquete localizado, el nombre localizado de la extensión se muestra junto con una descripción localizada. Si proporciona un archivo de licencia localizado o una dirección URL que apunta a información localizada, también se muestran.

Si el contenido del paquete VSIX incluye un VSPackage que agrega comandos de menú u otra interfaz de usuario, vea [localizar comandos](#) de menú para obtener información sobre cómo localizar los nuevos elementos de interfaz de usuario.

Estructura de directorios

Cuando un usuario instala una extensión, **Extensiones y actualizaciones** comprueba el nivel superior del paquete VSIX para una carpeta cuyo nombre coincide con la configuración regional de Visual Studio del equipo de destino. Si **Extensiones y actualizaciones** encuentra un archivo *.vsixlangpack* en la carpeta, sustituye los valores localizados de ese archivo por los valores correspondientes en el archivo *.vsixmanifest*. Estos valores se muestran cuando se instala la extensión. En el ejemplo siguiente se muestra la estructura de directorios de un paquete VSIX que está localizado en español (es-ES) y francés (fr-FR).

```
.
├── MyExtension.dll
├── Extension.vsixmanifest
├── [Content_Types].xml
├── es-ES
│   └── Extension.vsixlangpack
└── fr-FR
    └── Extension.vsixlangpack
```

NOTE

Las plantillas de proyecto compatibles Visual Studio SDK con VSIX en el manifiesto generar VSIX y así saber, así se llama *source.extension.vsixmanifest*. Cuando Visual Studio compila el proyecto, copia el contenido de ese archivo en *Extension.VsixManifest* en el paquete VSIX.

El archivo Extension.vsixlangpack

El archivo *Extension.vsixlangpack* sigue el esquema 2.0 del paquete de [idioma VSIX](#). Este esquema

`PackageLanguagePackManifest` tiene un , que `Metadata` va seguido inmediatamente de un elemento secundario. El elemento `Metadata` puede contener hasta `DisplayName` `Description` 6 `MoreInfo` `License` elementos secundarios, , , , `ReleaseNotes` , y `Icon` . Estos elementos secundarios `DisplayName` `Description` corresponden `MoreInfo` `License` a `ReleaseNotes` los `Icon` elementos `Metadata` , , , , , y secundarios del elemento del archivo *Extension.vsixmanifest*.

Al crear un archivo *vsixlangpack*, `Include in Vsix` debe `true` establecer la propiedad en . De lo contrario, se omitirá el texto de instalación localizado.

Para establecer la propiedad Incluir en Vsix

1. En el Explorador de soluciones , haga clic con el botón secundario en el archivo *Extension.vsixlangpack* y,

a continuación, haga clic en **Propiedades**.

2. En la **cuadrícula de propiedades**, haga clic en **Incluir en Vsix** establezca su valor en `true`.

Ejemplo

Descripción

En el ejemplo siguiente se muestran partes relevantes de un archivo *Extension.vsixmanifest*. El archivo también incluye el archivo *Extension.vsixlangpack* correspondiente para español. Los valores del paquete de idioma reemplazan los valores del manifiesto si la configuración regional de Visual Studio del equipo de destino está establecida en español.

Código

- [*Extension.vsixmanifest*]

```
<?xml version="1.0" encoding="utf-8"?>
<PackageManifest ...>
  <Metadata ...>
    <DisplayName>Family Tree</DisplayName>
    <Description>This extension places a custom treeview control in the toolbox that is optimized for
handling family tree information.</Description>
    <MoreInfo>http://www.contoso.com/products/FamilyTree.htm</MoreInfo>
    <License>Eula.rtf</License>
    <ReleaseNotes>ReleaseNotes.rtf</ReleaseNotes>
    <Icon>Icon.png</Icon>
  </Metadata>
  <Installation .../>
  <Dependencies .../>
  <Prerequisites .../>
  <Assets .../>
</PackageManifest>
```

- [*Extension.vsixlangpack*]

```
<?xml version="1.0" encoding="utf-8"?>
<PackageLanguagePackManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011">
  <Metadata>
    <DisplayName>Arbol de Familia</DisplayName>
    <Description> Esta extensión pone control personalizado en la caja de herramientas por manejar
información de familia.</Description>
    <MoreInfo> http://www.contoso.com/products/es/ArbolDeFamilia.htm</MoreInfo>
    <License>Eula.rtf</License>
    <ReleaseNotes>ReleaseNotes.rtf</ReleaseNotes>
    <Icon>Icon.png</Icon>
  </Metadata>
</PackageLanguagePackManifest>
```

Vea también

TITLE	DESCRIPCIÓN
Referencia del esquema 2.0 del paquete de idioma VSIX	Un paquete de idioma VSIX describe la información de localización de un archivo de implementación .vsix.
Anatomía de un paquete VSIX	Describe la estructura y el contenido de un paquete vsix.

TITLE	DESCRIPCIÓN
Localizar comandos de menú	Muestra cómo localizar otros recursos de texto en una extensión.

Referencia del esquema del paquete de idioma VSIX 2,0

29/02/2020 • 2 minutes to read • [Edit Online](#)

El esquema del paquete de idioma VSIX proporciona información de instalación localizada para los paquetes VSIX. La versión 2,0 de este esquema admite elementos de localización adicionales.

Esquema del paquete de idioma

El elemento raíz del archivo de paquete de idioma es `<PackageLanguagePackManifest>`, con un atributo de `Version`, que es la versión del formato del paquete de idioma. En este artículo se describe la versión 2,0 del formato de paquete de idioma, que se especifica en el manifiesto estableciendo el atributo `Version` en el valor `Version="2.0.0"`. El elemento raíz contiene exactamente un elemento `<Metadata>` secundario.

Elemento `PackageLanguagePackManifest`

En el elemento `<PackageLanguagePackManifest>` debe existir el siguiente elemento:

TÍTULO	DESCRIPCIÓN
<code><Metadata></code>	El elemento contenedor para todos los metadatos de paquete localizados

Elemento metadata

En el elemento `<Metadata>` puede tener los elementos siguientes:

TÍTULO	DESCRIPCIÓN
<code><DisplayName></code>	Nombre localizado de la extensión que se va a instalar.
<code><Description></code>	La descripción localizada de la extensión que se va a instalar.
<code><License></code>	Una ruta de acceso a una versión localizada de la licencia de la extensión
<code><MoreInfo></code>	Un vínculo a información localizada sobre la extensión
<code><ReleaseNotes></code>	Una ruta de acceso o un vínculo a una versión localizada de las notas de la versión
<code><Icon></code>	Una ruta de acceso a una versión localizada del icono de extensiones

Manifiesto de ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<PackageLanguagePackManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011">
  <Metadata>
    <DisplayName>Arbol de Familia</DisplayName>
    <Description> Esta extensión pone control personalizado en la caja de herramientas por manejar información
de familia.</Description>
    <MoreInfo> http://www.contoso.com/products/es/ArbolDeFamilia.htm</MoreInfo>
    <License>Eula.rtf</License>
    <ReleaseNotes>ReleaseNotes.rtf</ReleaseNotes>
    <Icon>Icon.png</Icon>
  </Metadata>
</PackageLanguagePackManifest>
```

Consulte también

TÍTULO	DESCRIPCIÓN
Localizar paquetes VSIX	Muestra cómo proporcionar compatibilidad con la instalación localizada para un paquete VSIX.
Referencia del esquema de extensión VSIX 2,0	Un manifiesto VSIX describe el contenido de un archivo de implementación . <i>vsix</i> . El archivo de implementación permite instalar una extensión de Visual Studio mediante el cuadro de diálogo extensiones y actualizaciones .
Búsqueda y uso de extensiones de Visual Studio	Muestra cómo usar el cuadro de diálogo extensiones y actualizaciones para instalar, quitar, activar y desactivar extensiones.

Cómo: Actualizar una extensión de Visual Studio

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede actualizar una extensión de Visual Studio en el sistema mediante **Extensiones y actualizaciones** para instalar la versión actualizada. Si crea una versión actualizada de una extensión, puede significarla como actualizada incrementando el número de versión en el manifiesto VSIX.

Las actualizaciones se instalan cuando el manifiesto **ID** VSIX de la **Version** extensión entrante tiene el mismo que el instalado y un número mayor. Si **Version** el número es el mismo o menor, el paquete no se puede instalar. Si **ID** los valores no coinciden, el paquete que aún no está instalado se reconoce como una extensión independiente.

Para ayudar a evitar conflictos durante el desarrollo, se recomienda desinstalar versiones anteriores de extensiones en curso y también desinstalar o deshabilitar cualquier otra extensión potencialmente conflictiva.

Para actualizar una extensión en el sistema

1. En el menú **Herramientas**, haga clic en **Extensiones y actualizaciones**.
2. En el panel izquierdo, haga clic en **Actualizaciones**.
3. En el panel central, haga clic en la actualización que desea instalar.

El número de versión de la extensión actualizada se muestra en el panel derecho, junto con otra información.

4. En la parte inferior del panel derecho, haga clic en **Actualizar**.

Para publicar una actualización de una extensión

1. En Visual Studio, abra la solución para la extensión que desea actualizar. Realice los cambios.

IMPORTANT

Sin firmar, todas las extensiones de usuario no se actualizan automáticamente. Siempre debe firmar sus extensiones.

2. En el **Explorador de soluciones**, abra *source.extension.manifest*.
3. En el diseñador de manifiestos, aumente el valor del número en el campo **Versión**.
4. Guarde la solución y compílela.
5. Cargue el nuevo archivo *.vsix* (en* la carpeta *-bin-Debug del proyecto) en el sitio Web de [Visual Studio Marketplace](#).

Cuando un usuario que tiene una versión anterior de la extensión abre **Extensiones y actualizaciones**, la nueva versión aparecerá en la lista **Actualizaciones**, siempre que la herramienta esté configurada para buscar actualizaciones automáticamente.

Puede habilitar o deshabilitar la comprobación automática de actualizaciones en **Environment** > la parte inferior del panel **Actualizaciones (Habilitar/deshabilitar la detección automática de actualizaciones disponibles)**, que cambia la configuración **Buscar actualizaciones en** > **Extensiones y actualizaciones** de entorno de opciones > de herramientas.

NOTE

A partir de Visual Studio 2015 Update 2, puede especificar **Environment** > (**en** > Extensiones de entorno y **actualizaciones** de entorno de **opciones** > de herramientas) si desea actualizaciones automáticas para extensiones por usuario, todas las extensiones de usuario o ambas (la configuración predeterminada).

Vea también

- [Anatomía de un paquete VSIX](#)
- [Buscar y usar extensiones de Visual Studio](#)

Preparar extensiones para la implementación de Windows Installer

06/04/2020 • 2 minutes to read • [Edit Online](#)

No puede usar un paquete de Windows Installer (MSI) para implementar un paquete VSIX. Sin embargo, puede extraer el contenido de un paquete VSIX para la implementación msi. Este documento muestra cómo preparar un proyecto cuya salida predeterminada es un paquete VSIX para su inclusión en un proyecto de instalación.

Preparar un proyecto de extensión para la implementación de Windows Installer

Realice estos pasos en nuevos proyectos de extensión antes de agregarlos a un proyecto de instalación.

Para preparar un proyecto de extensión para la implementación de Windows Installer

1. Cree un VSPackage, componente MEF, Editor Adornment u otro tipo de proyecto de extensibilidad que incluya un manifiesto VSIX.
2. Abra el manifiesto VSIX en el editor de código.
3. Establezca `InstalledByMsi` el elemento del manifiesto `true` VSIX en `.vsix`. Para obtener más información sobre el manifiesto VSIX, vea Referencia 2.0 del esquema de [extensión VSIX](#).

Esto impide que el instalador de VSIX intente instalar el componente.

4. Haga clic con el botón derecho en el proyecto en el **Explorador** de soluciones y haga clic en **Propiedades**.
5. Seleccione la pestaña **VSIX**.
6. Active la casilla **Copiar contenido VSIX en la siguiente ubicación** y escriba la ruta de acceso a donde el proyecto de instalación recogerá los archivos.

Extraer archivos de un paquete VSIX existente

Realice estos pasos para agregar el contenido de un paquete VSIX existente a un proyecto de instalación cuando no tenga los archivos de origen.

Para extraer archivos de un paquete VSIX existente

1. Cambie el nombre del *archivo .VSIX* que contiene la extensión de *filename.vsix* a *filename.zip*.
2. Copie el contenido del archivo *.zip* en un directorio.
3. Elimine el archivo *[Content_types].xml* del directorio.
4. Edite el manifiesto VSIX, tal y como se muestra en el procedimiento anterior.
5. Agregue los archivos restantes al proyecto de instalación.

Vea también

- [Implementación del instalador de Visual Studio](#)
- [Tutorial: Crear una acción personalizada](#)

Firma de paquetes VSIX

06/04/2020 • 2 minutes to read • [Edit Online](#)

No es necesario firmar ensamblados de extensión para poder ejecutarse en Visual Studio, pero es recomendable hacerlo.

Si desea proteger la extensión y asegurarse de que no se ha manipulado, puede agregar una firma digital a un paquete VSIX. Cuando se firma un VSIX, el instalador de VSIX mostrará un mensaje que indica que está firmado, además de más información sobre la propia firma. Si se ha modificado el contenido de VSIX y VSIX no se ha firmado de nuevo, el instalador de VSIX mostrará que la firma no es válida. La instalación no se detiene, pero se advierte al usuario.

IMPORTANT

A partir de Visual Studio 2015, los paquetes VSIX firmados con cualquier cosa que no sea el cifrado SHA256 se identificarán como que tienen una firma no válida. La instalación de VSIX no está bloqueada, pero se advertirá al usuario.

Firma de un VSIX con VSIXSignTool

Hay una herramienta de firma de cifrado SHA256 disponible en [VisualStudioExtensibility](#) en nuget.org en [VsixSignTool](#).

Para usar VSIXSignTool

1. Agregue su VSIX a un proyecto.
2. Haga clic con el botón derecho en el nodo del proyecto en el Explorador de soluciones, **seleccionando Agregar | Administrar paquetes NuGet**. Para obtener más información sobre NuGet y agregar paquetes NuGet, consulte la documentación de [NuGet](#) y los temas de [interfaz](#) de usuario del Administrador de paquetes.
3. Busque VSIXSignTool en VisualStudioExtensibility e instale el paquete NuGet.
4. Ahora puede ejecutar VSIXSignTool desde la ubicación de paquetes locales del proyecto. Consulte la ayuda de la línea de comandos de la herramienta para su escenario de firma (VSIXSignTool.exe /?).

Por ejemplo, para firmar con un archivo de certificado protegido por contraseña:

```
VSIXSignTool.exe sign </f certfile <> </p password> VSIXfile>
```

Vea también

- [Suministro de extensiones de Visual Studio](#)

Galerías privadas

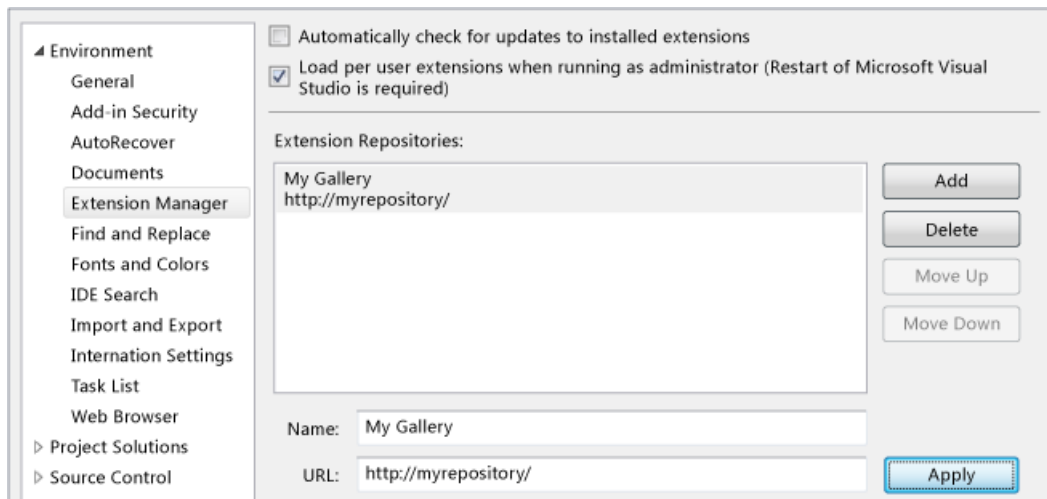
06/04/2020 • 4 minutes to read • [Edit Online](#)

Puede compartir los controles, plantillas y herramientas que desarrolla publicándolos en una *galería privada* en la intranet de su organización, como se indica a continuación:

- Cree una fuente Atom (RSS) en una ubicación central (repositorio) configurada adecuadamente en la intranet. Para obtener más información, consulte [Cómo: crear una fuente Atom para una galería privada](#).
- Distribuya un archivo *.pkgdef* que describa la galería privada. Se recomienda esta configuración para los administradores que desean conectar una galería privada a muchos equipos al mismo tiempo.

Agregar una galería privada a extensiones y actualizaciones en Visual Studio

Cuando hay una galería privada disponible, puede agregarla a **Extensiones y actualizaciones** en Visual Studio.



Para agregar una galería privada a Extensiones y actualizaciones

1. En la barra de menús, seleccione **Herramientas > Opciones**.
2. En el nodo **Entorno**, seleccione **Extensiones y actualizaciones**.
3. Elija el botón **Agregar**.
4. En el campo **Nombre**, escriba un nombre para `My Gallery` la galería privada, por ejemplo, .
5. En el campo **Dirección URL**, escriba la dirección URL de la fuente Atom o el sitio de SharePoint que hospeda la galería privada.
 - a. Si el host es una fuente Atom que se conecta a <http://www.mywebsite/mygallery/atom.xml> la galería privada, la dirección URL se parecería a esta: . Esta dirección URL puede hacer referencia a un archivo o a una ruta de acceso de red.
 - b. Si el host es un sitio de SharePoint, la dirección URL se parecería a este:
`http://mysharepoint/sites/mygallery/forms/AllItems.aspx`.

Gestionar galerías privadas

Un administrador puede poner una galería privada a disposición de varios equipos al mismo tiempo modificando el registro del sistema en cada equipo. Para ello, cree un archivo *.pkgdef* que describa las nuevas claves del

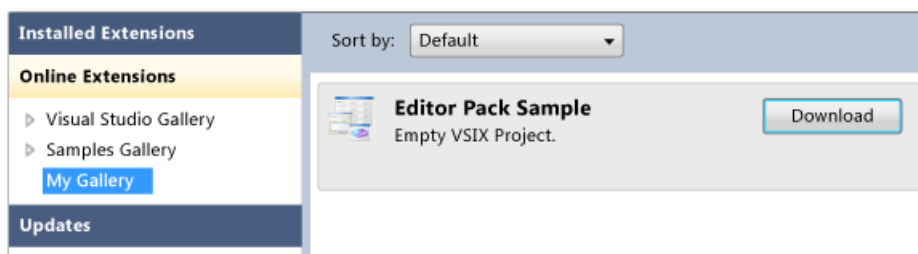
Registro y sus valores. El formato de este archivo es el siguiente.

```
[${RootKey}\ExtensionManager\Repositories\{UniqueGUID}]
@={URI} (REG_SZ)
Disabled=0 | 1 (DWORD)
Priority=0 (highest priority) ... MaxInt (lowest priority) (DWORD) (uint)
Protocol=Atom|Sharepoint (REG_SZ)
DisplayName={DisplayName} (REG_SZ)
DisplayNameResourceID={ID} (REG_SZ)
DisplayNamePackageGuid={GUID} (REG_SZ)
```

Para obtener más información, consulte [Cómo: administrar una galería privada mediante la configuración del Registro](#).

Instalar extensiones desde una galería privada

Puede buscar e instalar extensiones de Visual Studio desde una galería privada en **Extensiones y actualizaciones**. Los pasos siguientes utilizan `My Gallery` una galería privada denominada.

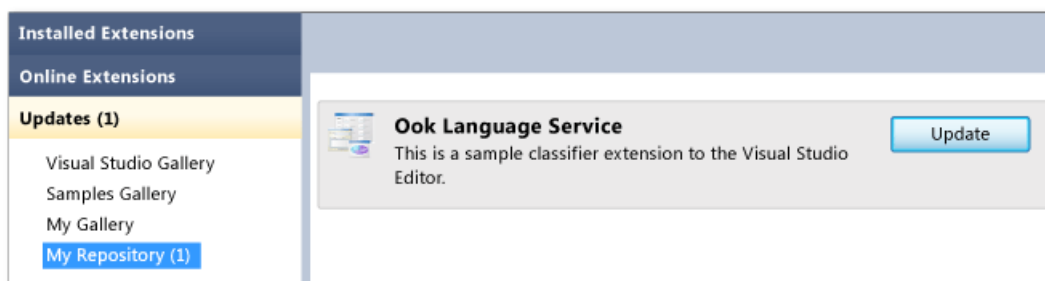


Para buscar e instalar extensiones desde una galería privada

1. En la barra de menús, elija **Extensiones > y actualizaciones** de herramientas.
2. En el panel izquierdo, seleccione **Extensiones en línea**, a continuación, seleccione **Mi galería**.
3. En el panel derecho, seleccione una extensión y, a continuación, elija el botón **Descargar**.

Actualizar extensiones de una galería privada

A medida que se publican nuevas versiones de las extensiones de Visual Studio en la galería privada, puede actualizar las extensiones que ha instalado. Los pasos siguientes utilizan `My Repository` una galería privada denominada.



Para actualizar una extensión instalada desde una galería privada

1. En la barra de menús, elija **Extensiones > y actualizaciones** de herramientas.
2. En el panel izquierdo, seleccione **Actualizaciones**, a continuación, seleccione **Mi repositorio**.
3. En el panel derecho, seleccione una extensión y, a continuación, elija el botón **Actualizar**.

Vea también

- [Búsqueda y uso de extensiones de Visual Studio](#)
- [Enviar extensiones de Visual Studio](#)

Cómo: Crear un feed Atom para una galería privada

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede crear una fuente Atom (RSS) en una ubicación de intranet que contenga extensiones y agregar la fuente a **Extensiones y actualizaciones** como una galería privada. Para más información, consulte [Private galleries](#) (Galerías privadas).

Crear una alimentación atom

Para crear una fuente Atom como una galería privada, primero debe recopilar las extensiones (archivos `.vsix`) en una carpeta. Puede organizarlos en subcarpetas si lo desea. También necesitará los siguientes recursos:

- Un archivo `atom.xml` que hace que las extensiones estén disponibles como una galería privada. Para obtener información sobre cómo conectar el archivo `atom.xml` a **Extensiones y actualizaciones**, consulte [Galerías privadas](#).
- Carpeta que contiene los archivos de imagen extraídos de las extensiones (por ejemplo, capturas de pantalla). El archivo `atom.xml` contiene vínculos relativos a estas imágenes para que estén disponibles en **Extensiones y actualizaciones**.

Por ejemplo, supongamos que ha recopilado las dos extensiones siguientes en una carpeta:

- `Template_Wizard_239.vsix`, que es una plantilla de proyecto VSIX vacía.
- `SelectionHighlight.vsix`, que es una herramienta para resaltar todas las instancias de una palabra seleccionada.

El contenido del archivo `atom.xml` se parecería al siguiente ejemplo:

```

<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text" />
  <id>uuid:bcecded5-97c8-4d24-96f1-7d9e16652433;id=1</id>
  <updated>2011-04-14T21:25:48Z</updated>
  <entry>
    <id>SelectionHighlight..a14874d2-8199-4a60-af8a-11d6447813aa</id>
    <title type="text">Highlight all occurrences of selected word</title>
    <summary type="text">This extends the editor to highlight ....</summary>
    <published>2011-04-14T14:24:51-07:00</published>
    <updated>2011-04-14T14:24:22-07:00</updated>
    <author>
      <name>Microsoft</name>
    </author>
    <link rel="icon" href="VSIXImages/SelectionHighlight..a14874d2-8199-4a60-af8a-11d6447813aa_Icon_SelectionHighlightIcon.jpg" />
    <link rel="previewimage" href="VSIXImages/SelectionHighlight..a14874d2-8199-4a60-af8a-11d6447813aa_PreviewImage_SelectionHighlight.jpg" />
    <content type="application/octet-stream" src="SelectionHighlight.vsix" />
    <Vsix xmlns="http://schemas.microsoft.com/developer/vsx-syndication-schema/2010"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Id>SelectionHighlight..a14874d2-8199-4a60-af8a-11d6447813aa</Id>
      <Version>1.31</Version>
      <References />
      <Rating xsi:nil="true" />
      <RatingCount xsi:nil="true" />
      <DownloadCount xsi:nil="true" />
    </Vsix>
  </entry>
  <entry>
    <id>Template_Wizard_239.Microsoft.3b38a7e3-5cbc-4389-a92a-d82tyc2ed592</id>
    ...
  </entry>
</feed>

```

Observe que las dos etiquetas de vínculo hacen referencia a capturas de pantalla en la carpeta generada de imágenes.

Vea también

- [Galerías privadas](#)

Cómo: Administrar una galería privada mediante la configuración del Registro

06/04/2020 • 2 minutes to read • [Edit Online](#)

Si es administrador o desarrollador de una extensión de Shell aislado, puede controlar el acceso a los controles, plantillas y herramientas de la Galería de Visual Studio, la Galería de ejemplos o galerías privadas. Para que una galería esté disponible o no esté disponible, cree un archivo *.pkgdef* que describa las claves del Registro modificadas y sus valores.

Gestionar galerías privadas

Puede crear un archivo *.pkgdef* para controlar el acceso a las galerías en varios equipos. Este archivo debe tener el siguiente formato.

```
[${RootKey}\ExtensionManager\Repositories\{UniqueGUID}]
@={URI} (REG_SZ)
Disabled=0 | 1 (DWORD)
Priority=0 (highest priority) ... MaxInt (lowest priority) (DWORD) (uint)
Protocol=Atom Feed|Sharepoint (REG_SZ)
DisplayName={DisplayName} (REG_SZ)
DisplayNameResourceID={ID} (REG_SZ)
DisplayNamePackageGuid={GUID} (REG_SZ)
```

La `Repositories` clave hace referencia a la galería que se va a habilitar o deshabilitar. La Galería de Visual Studio y la Galería de ejemplos usan los siguientes GUID de repositorio:

- Galería de Visual Studio : 0F45E408-7995-4375-9485-86B8DB553DC9
- Galería de muestras : AEB9CB40-D8E6-4615-B52C-27E307F8506C

El `Disabled` valor es opcional. De forma predeterminada, una galería está habilitada.

El `Priority` valor determina el orden en el que se enumeran las galerías en el cuadro de diálogo

Opciones. La Galería de Visual Studio tiene la prioridad 10 y la Galería de ejemplos tiene la prioridad 20.

Las galerías privadas comienzan en la prioridad 100. Si varias galerías tienen el mismo valor de prioridad, el orden `DisplayName` en el que aparecen viene determinado por los valores de sus atributos localizados.

El `Protocol` valor es necesario para las galerías basadas en Atom o SharePoint.

Debe `DisplayName` especificarse `DisplayNameResourceID` `DisplayNamePackageGuid`, o ambos y, . Si se especifican todos, `DisplayNameResourceID` `DisplayNamePackageGuid` se utiliza el par y.

Deshabilite la Galería de Visual Studio mediante un archivo .pkgdef

Puede deshabilitar una galería en un archivo *.pkgdef*. La entrada siguiente deshabilita la Galería de Visual Studio:

```
[${RootKey}\ExtensionManager\Repositories\{0F45E408-7995-4375-9485-86B8DB553DC9}]
"Disabled"=dword:00000001
```

La entrada siguiente deshabilita la Galería de ejemplos:

```
[$RootKey$\ExtensionManager\Repositores\{AEB9CB40-D8E6-4615-B52C-27E307F8506C}]  
"Disabled"=dword:00000001
```

Vea también

- [Galerías privadas](#)

Compatibilidad con varias versiones de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

El término *en paralelo* significa que puede instalar y mantener varias versiones de un producto en el mismo equipo. Para VSPackages, eso significa que un usuario puede tener varias versiones de Visual Studio instaladas en el mismo equipo. Sin embargo, no puede tener versiones en paralelo de los VSPackages cargados en una sola versión de Visual Studio.

Antes de que el VSPackage se pueda cargar en versiones en paralelo de Visual Studio, tenga en cuenta lo siguiente:

- Debe determinar qué estrategia de implementación en paralelo desea seguir.

Para obtener más información, vea [Elegir entre VSPackages compartidos y versionados](#).

- Los formatos de archivo de solución y proyecto deben ajustarse a su estrategia de implementación.

Para obtener más información, consulte [Actualización de proyectos personalizados](#) y Registro de extensiones de nombre de archivo [para implementaciones en paralelo](#).

- El instalador debe controlar la estrategia de implementación para que los componentes versionados, así como los componentes compartidos entre todas las versiones, se instalen y registren correctamente.

Para obtener más información, vea [Instalar VSPackages con Windows Installer](#) y también Administración de [componentes](#).

NOTE

La instalación de una versión de Visual Studio también instala una versión correspondiente de .NET Framework. Por ejemplo, la instalación de Visual Studio 2010 y Visual Studio 2012 en el mismo equipo también instala las versiones 4.0 y 4.5 de .NET Framework, respectivamente.

En esta sección

- [Elegir entre VSPackages compartidos y versionados](#) Explica cómo resolver problemas en paralelo en el VSPackage.
- [Registro de extensiones](#) de nombre de archivo para implementaciones en paralelo Describe cómo el VSPackage puede registrar asociaciones de archivos en un escenario en paralelo.

Secciones relacionadas

- [Instalación de VSPackages con Windows Installer](#)

Elija entre VSPackages compartidos y versionados

06/04/2020 • 7 minutes to read • [Edit Online](#)

Diferentes versiones de Visual Studio pueden coexistir en el mismo equipo. VSPackages puede admitir Visual Studio cualquier combinación de versiones.

Puede habilitar las instalaciones en paralelo de VSPackages a través de cualquiera de las dos estrategias, la estrategia compartida o la estrategia versionada. Ambos admiten la presencia Visual Studio de varias versiones y versiones asociadas de .NET Framework.

En la estrategia compartida, un VSPackage se Visual Studio registra para su uso en varias versiones de . En la estrategia versionada, se instalan varios archivos DLL Visual Studio de VSPackage, uno para cada versión que admite.

VSPackages compartidos

El uso de un VSPackage compartido es adecuado cuando Visual Studio se usa el mismo VSPackage en varias versiones de . Para implementar un VSPackage compartido, debe seguir los pasos siguientes:

- Haga que el VSPackage Visual Studio sea compatible con varias versiones de . Hay dos formas de hacerlo disponibles:
 - Limite el VSPackage a usar solo las Visual Studio características de la versión más antigua de la que admite.
 - Programe el VSPackage para Visual Studio adaptarlo a la versión en la que se está ejecutando. A continuación, si se produce un error en las consultas de Visual Studio servicios más recientes, el VSPackage puede ofrecer otros servicios que se admiten en versiones anteriores de .
- Registre el VSPackage correctamente. Para obtener más información, vea Registro de [VSPackage](#) y [Registro de VSPackage administrado](#).
- Registre las extensiones de archivo correctamente. Para obtener más información, consulte Registro de extensiones de nombre de [archivo para implementaciones en paralelo](#).
- Cree un instalador que implemente el VSPackage Visual Studio para las versiones adecuadas de . Para obtener más información, vea [Instalar VSPackages con Windows Installer](#) y Administración de [componentes](#).
- Abordar el problema de las colisiones de registro. Para obtener más información, vea Registro de [VSPackage](#).
- Asegúrese de que los archivos compartidos y versionados respeten el recuento de referencias para permitir la instalación y eliminación seguras de varias versiones. Para obtener más información, consulte [Gestión de componentes](#).

VSPackages versionados

En la estrategia de VSPackage versionada, se Visual Studio crea un VSPackage para cada versión que admite. Esto es adecuado cuando se espera aprovechar los servicios Visual Studio proporcionados por versiones posteriores de , porque cada VSPackage puede evolucionar sin afectar a los demás. Sin embargo, la estrategia versionada de crear varios binarios, ya sea desde una única base de código o desde varias bases de código independientes, podría implicar más desarrollo inicial que la estrategia compartida. Además, es posible que sea necesario realizar

un trabajo de instalación adicional porque Visual Studio debe crear una instalación independiente para cada versión o una única configuración que detecte las versiones de las que están instaladas y que admite el VSPackage.

Compatibilidad binaria

Por lo general, la compatibilidad binaria permite que los VSPackages de código nativo desarrollados con versiones anteriores de Visual Studio se ejecuten en versiones posteriores de Visual Studio. Sin embargo, hay tres excepciones importantes:

- Si el VSPackage se basa en una versión determinada de Common Visual Studio Language Runtime, debe determinar en qué versión se está ejecutando.
- Un VSPackage puede tener una dependencia en una característica específica de otro VSPackage u otro producto. Por lo tanto, el VSPackage solo puede ejecutarse donde se satisface la dependencia.
- Un VSPackage podría verse afectado por Visual Studio una corrección de Visual Studio seguridad en un Service Pack o una versión posterior de . En esos casos, un VSPackage desarrollado con Visual Studio SDK una versión Visual Studio anterior de la podría no ejecutarse en versiones de después de aplicar la corrección de seguridad. Sin embargo, puede volver a generar el paquete con la versión posterior y hacer que también se ejecute en versiones anteriores.

VSPackages administrados deben compilarse Visual Studio mediante Visual Studio SDK una versión Visual Studio que coincidan con la versión de destino de .

Además de planear la compatibilidad binaria para los archivos binarios de VSPackage, también debe tener en cuenta los formatos de archivo de solución y proyecto. Si el VSPackage crea un nuevo tipo de proyecto, debe decidir si Visual Studio puede ejecutar en una sola versión o en varias versiones de . Para obtener más información, consulte [Actualización de proyectos personalizados](#).

Vea también

- [Instalación de VSPackages con Windows Installer](#)
- [Gestión de componentes](#)

Registrar extensiones de nombre de archivo para implementaciones en paralelo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para VSPackages implementados en un entorno en paralelo, debe registrar las extensiones de Visual Studio nombre de archivo para asociar los archivos con la versión correcta de . A menos que utilice una extensión de nombre de archivo específica de la versión, el registro permite a los usuarios abrir los archivos de elementos de proyecto y proyecto en la versión adecuada de Visual Studio.

En esta sección

- [Acerca de las extensiones de nombre de archivo](#) Describe cómo se registran las extensiones de nombre de archivo.
- [Especificar controladores](#) de archivos para extensiones de nombre de archivo Proporciona información sobre cómo registrar las aplicaciones que pueden abrir, editar, etc., una extensión de nombre de archivo determinada.
- [Registrar verbos para extensiones](#) de nombre de archivo Describe cómo registrar verbos.
- [Administrar asociaciones de archivos en paralelo](#) Describe cómo controlar las instalaciones en paralelo en Visual Studio las que se debe invocar una versión determinada de para abrir un archivo.

Secciones relacionadas

- [Admite varias versiones de Visual Studio](#) Describe problemas relacionados con Visual Studio varias versiones de VSPackage durante el desarrollo y la implementación para los usuarios finales.

Acerca de las extensiones de nombre de archivo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando se registra una extensión de archivo de un Visual StudioVSPackage, se asocia a una versión de . Esto es importante si hay Visual Studio más de una versión instalada en un equipo.

Las extensiones de archivo para VSPackages se registran en **HKEY_CLASSES_ROOT** clave con un valor predeterminado que apunta al identificador de programación asociado (ProgID).

En el ejemplo siguiente se muestra información de registro para la extensión de archivo *.vcproj*:

```
HKEY_CLASSES_ROOT\  
  .vcproj\  
    (default)=" VisualStudio.vcproj.8.0"
```

Los archivos Visual Studio asociados deben tener un ProgID versionado, como `VisualStudio.vcproj.8.0`. Un ProgID versionado permite que las instalaciones en paralelo del producto mantengan las asociaciones de extensión de archivo entre las versiones del producto. Un ProgID específico de la versión también le permite utilizar verbos estándar, como abrir, editar, etc., sin la preocupación de sobrescribir o ser sobrescrito por otras aplicaciones o versiones de Visual Studio.

En algunos casos, el ProgID asociado con una extensión de archivo no debe cambiarse. Por ejemplo, el ProgID para la extensión de archivo *.htm* (progid - htmlfile) está codificado de forma rígida en varios lugares del sistema operativo y es ampliamente conocido y utilizado en asociación con archivos *.htm* y *.html*.

Vea también

- [Registrar extensiones de nombre de archivo para implementaciones en paralelo](#)
- [Especificar controladores de archivos para extensiones de nombre de archivo](#)

Especificación de identificadores de archivo para extensiones de nombre de archivo

06/04/2020 • 5 minutes to read • [Edit Online](#)

Hay varias maneras de determinar la aplicación que maneja un archivo que tiene una extensión de archivo determinada. Los verbos `OpenWithList` y `OpenWithProgids` son dos formas de especificar controladores de archivos en la entrada del Registro para la extensión de archivo.

OpenWithList Verbo

Al hacer clic con el botón derecho en un archivo en el Explorador de Windows, verá el comando **Abrir**. Si hay más de un producto asociado a una extensión, verá un submenú **Abrir con**.

Puede registrar diferentes aplicaciones para abrir una extensión estableciendo la clave `OpenWithList` para la extensión de archivo en `HKEY_CLASSES_ROOT`. Las aplicaciones enumeradas bajo esta clave para una extensión de archivo aparecen bajo el encabezado **Programas recomendados** en el cuadro de diálogo **Abrir con**. En el ejemplo siguiente se muestran las aplicaciones registradas para abrir la extensión de archivo `.vcproj`.

```
HKEY_CLASSES_ROOT\
  .vcproj\
    (default)="VisualStudio.vcproj.14.0"
    OpenWithList\
      devenv.exe
```

NOTE

Las claves que especifican las aplicaciones proceden de la lista de `HKEY_CLASSES_ROOT.Aplicaciones`.

Al agregar una clave `OpenWithList`, declara que la aplicación admite una extensión de archivo incluso si otra aplicación toma posesión de la extensión. Podría ser una versión futura de la aplicación u otra aplicación.

OpenWithProgIDs

Los identificadores de programación (`ProgID`) son versiones descriptuigénitas de `ClassID` que identifican una versión de una aplicación u objeto COM. Cada objeto co-creable debe tener su propio `ProgID`. Por ejemplo, `VisualStudio.DTE.7.1` inicia Visual Studio .NET 2003 mientras se Visual Studio inicia `VisualStudio.DTE.10.0`. Como propietario de un tipo de proyecto o de elemento de proyecto, debe crear un `ProgID` específico de la versión para la extensión de archivo. Estos `ProgID` pueden ser redundantes en el que más de un `ProgID` puede iniciar la misma aplicación. Para obtener más información, consulte [Registro de verbos para extensiones de nombre de archivo](#).

Utilice la siguiente convención de nomenclatura para los `ProgID` de archivos versionados para evitar la duplicación con el registro de otros proveedores:

EXTENSIÓN DE ARCHIVO	PROGID VERSIONADO
.extensión	Productname. extension.versionMajor:versionMinor

Puede registrar diferentes aplicaciones que son capaces de abrir una extensión de archivo en particular mediante la adición de `ProgIDs` versionados como valores a la extensión de `HKEY_CLASSES_ROOT`*<>* clave

OpenWithProgids. Esta clave del Registro contiene una lista de ProgID alternativos asociados con la extensión de archivo. Las aplicaciones asociadas con los ProgID enumerados aparecen en el submenú **Abrir con** nombre_de_producto_. Si se especifica la misma `OpenWithList` aplicación `OpenWithProgids` en las claves y, el sistema operativo combina los duplicados.

NOTE

La `OpenWithProgids` clave solo se admite en Windows XP. Dado que otros sistemas operativos ignoran esta clave, no la use como el único registro para los controladores de archivos. Utilice esta clave para proporcionar una mejor experiencia de usuario en Windows XP.

Agregue los ProgIDs deseados como valores del tipo REG_NONE. El código siguiente proporciona un ejemplo de registro de ProgIDs para una extensión de archivo (*.ext*).

```
HKEY_CLASSES_ROOT\
    .ext\
        (default)="MyProduct.ext.14.0"
        OpenWithProgids
            progid          REG_NONE (zero-length binary value)
            otherprogid     REG_NONE (zero-length binary value)
```

El ProgID especificado como valor predeterminado para la extensión de archivo es el controlador de archivos predeterminado. Si modifica el ProgID para una extensión de Visual Studio archivo que se incluye con una versión `OpenWithProgids` anterior de o que puede ser asumido por otras aplicaciones, debe registrar la clave para la extensión de archivo y especificar el nuevo ProgID en la lista junto con los ProgID antiguos que admite. Por ejemplo:

```
HKEY_CLASSES_ROOT\
    .vcproj\
        (default)="VisualStudio.vcproj.14.0"
        OpenWithProgids
            vcprojfile          //old progid
            VisualStudio.vcproj.12.0 //old progid
            VisualStudio.vcproj.14.0 //new progid
```

Si el antiguo ProgID tiene verbos asociados, estos verbos también aparecerán en **Abrir con** nombre *de producto* en el menú contextual.

Vea también

- [Acerca de las extensiones de nombre de archivo](#)
- [Registro de verbos para extensiones de nombre de archivo](#)

Registrar verbos para extensiones de nombre de archivo

06/04/2020 • 3 minutes to read • [Edit Online](#)

La asociación de una extensión de nombre de archivo con una aplicación generalmente tiene una acción preferida que se produce cuando un usuario hace doble clic en un archivo. Esta acción preferida está vinculada a un verbo, por ejemplo abierto, que corresponde a la acción.

Puede registrar verbos asociados a un identificador de programación (ProgID) para una extensión mediante la clave Shell ubicada en `HKEY_CLASSES_ROOT{progid}`. Para obtener más información, consulte [Tipos de archivo](#).

Registrar verbos estándar

El sistema operativo reconoce los siguientes verbos estándar:

- Abrir
- Editar
- Reproducir
- Print
- Vista previa

Siempre que sea posible, registre un verbo estándar. La opción más común es el verbo Abrir. Utilice el verbo Editar solo si hay una clara diferencia entre abrir el archivo y editar el archivo. Por ejemplo, al abrir un archivo `.htm` se muestra en el explorador, mientras que al editar un archivo `.htm` se inicia un editor HTML. Los verbos estándar se localizan con la configuración regional del sistema operativo.

NOTE

Al registrar verbos estándar, no establezca el valor predeterminado para la tecla Abrir. El valor predeterminado contiene la cadena de visualización en el menú. El sistema operativo proporciona esta cadena para verbos estándar.

Los archivos de proyecto deben registrarse para iniciar una nueva instancia de Visual Studio cuando un usuario abre el archivo. En el ejemplo siguiente se muestra Visual C# un registro de verbo estándar para un proyecto.


```
[HKEY_CLASSES_ROOT\.csproj]
@="VisualStudio.csproj.8.0"

[HKEY_CLASSES_ROOT\.csproj\OpenWithList]
[HKEY_CLASSES_ROOT\.csproj\OpenWithList\VSLauncher.exe]
@=""

[HKEY_CLASSES_ROOT\.csproj\OpenWithProgids]
"VisualStudio.csproj.8.0"=""

[HKEY_CLASSES_ROOT\Applications\VSLauncher.exe]
[HKEY_CLASSES_ROOT\Applications\VSLauncher.exe\Shell]
[HKEY_CLASSES_ROOT\Applications\VSLauncher.exe\Shell\Open]
[HKEY_CLASSES_ROOT\Applications\VSLauncher.exe\Shell\Open\Command]
@"C:\Program Files\Common Files\Microsoft Shared\MSEnv\VSLauncher.exe %1"

[HKEY_CLASSES_ROOT\VisualStudio.csproj.8.0]
@="C# Project file"

[HKEY_CLASSES_ROOT\VisualStudio.csproj.8.0\DefaultIcon]
@"C:\VisualStudioPath\VC#\VCSPackages\csproj.dll,0"

[HKEY_CLASSES_ROOT\VisualStudio.csproj.8.0\shell]
[HKEY_CLASSES_ROOT\VisualStudio.csproj.8.0\shell\Open]
[HKEY_CLASSES_ROOT\VisualStudio.csproj.8.0\shell\Open\Command]
@"C:\Program Files\Common Files\Microsoft Shared\MSEnv\VSLauncher.exe" %1
```

Para abrir un archivo en Visual Studio una instancia existente de , registre una clave DDEEXEC. En el ejemplo siguiente se muestra Visual C# un registro de verbo estándar para un archivo .cs.

```
[HKEY_CLASSES_ROOT\.cs]
@="VisualStudio.cs.8.0"

[HKEY_CLASSES_ROOT\.cs\OpenWithList]
[HKEY_CLASSES_ROOT\.cs\OpenWithList\devenv.exe]
@=""

[HKEY_CLASSES_ROOT\.cs\OpenWithProgids]
"VisualStudio.cs.8.0"=""

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0]
@="C# Source file"

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\DefaultIcon]
@"C:\VisualStudioPath\VC#\VCSPackages\csproj.dll,1"

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell]
[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell\Open]
[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell\Open\Command]
@"C:\VisualStudioPath\Common7\IDE\devenv.exe" /dde %1"

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell\Open\ddeexec]
@="Open(%1)"

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell\Open\ddeexec\Application]
@"VisualStudio.8.0"

[HKEY_CLASSES_ROOT\VisualStudio.cs.8.0\shell\Open\ddeexec\Topic]
@"system"
```

Establezca el verbo predeterminado

El verbo predeterminado es la acción que se ejecuta cuando un usuario hace doble clic en un archivo en el

Explorador de Windows. El verbo predeterminado es el verbo especificado como el valor predeterminado para la **clave de** *HKEY_CLASSES_ROOT progid*. Si no se especifica ningún valor, el verbo predeterminado es el primer verbo especificado en la lista de **claves** *HKEY_CLASSES_ROOT*progid.**

NOTE

Si tiene previsto cambiar el verbo predeterminado de una extensión en una implementación en paralelo, tenga en cuenta el impacto en la instalación y eliminación. Durante la instalación, se sobrescribe el valor predeterminado original.

Vea también

- [Administrar asociaciones de archivos en paralelo](#)

Administrar asociaciones de archivos en paralelo

06/04/2020 • 10 minutes to read • [Edit Online](#)

Si el VSPackage proporciona asociaciones de archivos, debe decidir cómo controlar las Visual Studio instalaciones en paralelo en las que se debe invocar una versión determinada de debe invocarse para abrir un archivo. Los formatos de archivo incompatibles agravan el problema.

Los usuarios esperan que una nueva versión de un producto sea compatible con versiones anteriores, de modo que los archivos existentes se puedan cargar en una nueva versión sin perder datos. Idealmente, el VSPackage puede cargar y guardar los formatos de archivo de versiones anteriores. Si eso no es cierto, debe ofrecer actualizar el formato de archivo a la nueva versión de su VSPackage. La desventaja de este enfoque es que el archivo actualizado no se puede abrir en la versión anterior.

Para evitar este problema, puede cambiar las extensiones cuando los formatos de archivo se vuelvan incompatibles. Por ejemplo, la versión 1 de su VSPackage podría usar la extensión *.mypkg10* y la versión 2 podría usar la extensión *.mypkg20*. Esta diferencia identifica el VSPackage que abre un archivo determinado. Si agrega VSPackages más recientes a la lista de programas que están asociados con una extensión antigua, los usuarios pueden hacer clic con el botón derecho en el archivo y elegir abrirlo en un VSPackage más reciente. En ese momento, el VSPackage puede ofrecer actualizar el archivo al nuevo formato o abrir el archivo y mantener la compatibilidad con versiones anteriores del VSPackage.

NOTE

Puede combinar estos enfoques. Por ejemplo, puede ofrecer compatibilidad con versiones anteriores cargando un archivo anterior y ofrecer actualizar el formato de archivo cuando el usuario lo guarde.

Enfréntate al problema

Si desea que varios VSPackages en paralelo para usar la misma Visual Studio extensión, debe elegir la versión de que está asociada con la extensión. Aquí hay dos alternativas:

- Abra el archivo en Visual Studio la versión más reciente de instalado en el equipo del usuario.

En este enfoque, el instalador es responsable Visual Studio de determinar la versión más reciente de la entrada del registro escrita para la asociación de archivos e incluirla en ella. En un paquete de Windows Installer, puede incluir acciones personalizadas Visual Studio para establecer una propiedad que indique la versión más reciente de .

NOTE

En este contexto, "último" significa "última versión compatible." Estas entradas del instalador no detectarán Visual Studio automáticamente una versión posterior de . Las entradas en [Detectar requisitos](#) del sistema y en [Comandos que deben ejecutarse después](#) Visual Studio de la instalación son similares a las que se presentan aquí y son necesarias para admitir versiones adicionales de .

Las siguientes filas de la tabla CustomAction establecen la propiedad DEVENV_EXE_LATEST para que sea una propiedad establecida por las tablas AppSearch y RegLocator que se describen en [Comandos que se deben ejecutar después](#) de la instalación. Las filas de la tabla InstallExecuteSequence programan las acciones personalizadas al principio de la secuencia de ejecución. Los valores de la columna Condición hacen que la lógica funcione:

- Visual Studio .NET 2002 es la versión más reciente si es la única versión actual.
- Visual Studio .NET 2003 es la versión Visual Studio más reciente solo si está presente y no está presente.
- Visual Studio es la última versión si es la única versión actual.

El resultado neto es que DEVENV_EXE_LATEST contiene la ruta de acceso de la última versión de devenv.exe.

Filas de tabla CustomAction que determinan la versión más reciente de Visual Studio

ACCIÓN	TIPO	SOURCE	DESTINO
CA_SetDevenvLatest_2002	51	DEVENV_EXE_LATEST	[DEVENV_EXE_2002]
CA_SetDevenvLatest_2003	51	DEVENV_EXE_LATEST	[DEVENV_EXE_2003]
CA_SetDevenvLatest_2005	51	DEVENV_EXE_LATEST	[DEVENV_EXE_2005]

InstallExecuteSequence filas de tabla que determinan la versión más reciente de Visual Studio

ACCIÓN	CONDICIÓN	SECUENCIA
CA_SetDevenvLatest_2002	DEVENV_EXE_2002 Y NO (DEVENV_EXE_2003 O DEVENV_EXE_2005)	410
CA_SetDevenvLatest_2003	DEVENV_EXE_2003 Y NO DEVENV_EXE_2005	420
CA_SetDevenvLatest_2005	DEVENV_EXE_2005	430

Puede usar la propiedad DEVENV_EXE_LATEST de la tabla Registry del paquete de Windows Installer para escribir el valor predeterminado de la clave ***HKEY_CLASSES_ROOT ProgId*ShellOpenCommand**, [DEVENV_EXE_LATEST] "%1"

- Ejecute un programa de iniciador compartido que puede hacer la mejor elección de las versiones de VSPackage disponibles.

Los desarrolladores eligieron Visual Studio este enfoque para manejar los complejos requisitos de Visual Studio los múltiples formatos de soluciones y proyectos que resultan de muchas versiones de . En este enfoque, se registra un programa de iniciador como el controlador de extensión. El iniciador examina el archivo y Visual Studio decide qué versión de y el VSPackage puede controlar ese archivo en particular. Por ejemplo, si un usuario abre un archivo que se guardó por última vez por una Visual Studio versión determinada de su VSPackage, el iniciador puede iniciar ese VSPackage en la versión coincidente de . Además, un usuario podría configurar el lanzador para iniciar siempre la última versión. Un lanzador también podría pedir a un usuario que actualice el formato del archivo. Si el formato del archivo incluye un número de versión, el iniciador podría informar a un usuario si el formato de archivo procede de una versión posterior a uno o varios de los VSPackages instalados.

El iniciador debe estar en un componente de Windows Installer que se comparte con todas las versiones de su VSPackage. Este proceso garantiza que la versión más reciente siempre está instalada y no se quita hasta que se desinstalan todas las versiones del VSPackage. De este modo, las asociaciones de archivos y otras entradas del registro del componente de iniciador se conservan incluso si se desinstala una versión del VSPackage.

Desinstalación y asociaciones de archivos

Desinstalación de un VSPackage que escribe entradas del Registro para asociaciones de archivos quita las asociaciones de archivos. Por lo tanto, la extensión no tiene programas asociados. Windows Installer no "recupera" las entradas del Registro que se agregaron cuando se instaló el VSPackage. Aquí hay algunas maneras de arreglar las asociaciones de archivos de un usuario:

- Utilice un componente de iniciador compartido como se ha descrito anteriormente.
- Indique al usuario que ejecute una reparación de la versión del VSPackage que el usuario desea poseer la asociación de archivos.
- Proporcione un programa ejecutable independiente que vuelva a escribir las entradas de registro adecuadas.
- Proporcione una página de opciones de configuración o un cuadro de diálogo que permita a los usuarios elegir asociaciones de archivos y reclamar asociaciones perdidas. Indique a los usuarios que lo ejecuten después de la desinstalación.

Vea también

- [Registrar extensiones de nombre de archivo para implementaciones en paralelo](#)
- [Registrar verbos para extensiones de nombre de archivo](#)

Busque Visual Studio

02/12/2019 • 3 minutes to read • [Edit Online](#)

A partir de Visual Studio 2017, puede instalar varias instancias de la misma versión o edición incluso. Esto resulta útil cuando desea obtener una vista previa de la nueva funcionalidad en el equipo de desarrollo principal manteniendo la instalación anterior. Debido a estos cambios, no hay ningún valor de variable o del registro de entorno solo que se puede utilizar para buscar una instancia. En su lugar, puede usar un [consulta COM API](#) para buscar instancias en función de criterios relacionados con la extensión.

Esto es una API rápida y de solo lectura con paquetes de NuGet disponibles para código nativo y administrado.

CÓDIGO	PAQUETE
Nativo	https://nuget.org/packages/Microsoft.VisualStudio.Setup.Configuration.Native
Administrado	https://nuget.org/packages/Microsoft.VisualStudio.Setup.Configuration.Interop

Puede buscar una sola instancia dada una ruta de acceso o el proceso actual, o enumerar todas las instancias. Consulte [nuestros ejemplos](#) para obtener ejemplos completos de cómo encontrar Visual Studio.

Herramientas

Para encontrar Visual Studio y otras herramientas en entornos de compilación, las secuencias de comandos de PowerShell, instaladores y más escenarios, hay una serie de herramientas de código abierto que puede usar directamente o redistribuir junto con sus propios scripts.

PROYECTO	DESCRIPCIÓN
vswhere	Solo archivo ejecutable nativo para buscar instancias que cumplen criterios de lanzamiento o versión preliminar, qué producto está instalado y qué cargas de trabajo están instalados. También admite la búsqueda de Visual Studio 2010 y versiones más recientes, aunque menos información se devuelve para Visual Studio 2017 y versiones más recientes. Consulte la wiki para obtener ejemplos.
Cmdlets de VSSetup	Admitidas los cmdlets de PowerShell 2.0 y versiones más recientes que devuelven información completa como objetos que puede usar para buscar instancias en función de los mismos criterios que <i>vswhere</i> y para detectar las propiedades incluso más acerca de las instancias. Consulte la wiki para obtener ejemplos.
VSIXBootstrapper	Busca automáticamente <i>VSIXInstaller</i> y pasa a través de la línea de comandos para instalar un *.vsix archivo. Esta característica puede ser útil en los instaladores que no tienen compatibilidad directa para la API de consulta. Consulte la wiki para obtener ejemplos.

Vea también

- Cambios realizados en el programa de instalación de Visual Studio 2017
- Inicie Visual Studio con el DTE

Inicio de Visual Studio con DTE

06/04/2020 • 3 minutes to read • [Edit Online](#)

A partir de Visual Studio 2017, el mecanismo para iniciar Visual Studio mediante DTE es diferente al lanzamiento de versiones anteriores de Visual Studio. Este cambio es necesario porque Visual Studio 2017 y versiones posteriores admiten instalaciones en paralelo de versiones principales (por ejemplo, puede tener una versión preliminar y una versión de lanzamiento instaladas en paralelo).

En el resto de este artículo se muestra el código que puede usar para iniciar Visual Studio 2019 mediante DTE.

Configuración del proyecto

Para ver el código de inicio en acción, cree un proyecto siguiendo estos pasos.

1. Cree un nuevo proyecto de aplicación de **consola** para .NET Framework.
2. Instale el paquete NuGet [Microsoft.VisualStudio.Setup.Configuration.Interop](#) y agregue una referencia al ensamblado.
3. Agregue una referencia a EnvDTE.
4. Pegue el código de [ejemplo](#) que sigue en el archivo *Program.cs*.
5. Presione F5 para ejecutar el programa. Debería ver Visual Studio 2019 abierto antes de que se cierre el programa.

Ejemplo de código

```
using Microsoft.VisualStudio.Setup.Configuration;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Runtime.InteropServices.ComTypes;
using System.Threading;

namespace ConsoleLauncherApp
{
    class Program
    {
        static void Main(string[] args)
        {
            EnvDTE.DTE dte = LaunchVsDte(isPreRelease: false);

            dte.MainWindow.WindowState = EnvDTE.vsWindowState.vsWindowStateMaximize;
            dte.MainWindow.WindowState = EnvDTE.vsWindowState.vsWindowStateMinimize;
            dte.MainWindow.WindowState = EnvDTE.vsWindowState.vsWindowStateNormal;
            dte.Quit();
        }

        private static EnvDTE.DTE LaunchVsDte(bool isPreRelease)
        {
            ISetupInstance setupInstance = GetSetupInstance(isPreRelease);
            string installationPath = setupInstance.GetInstallationPath();
            string executablePath = Path.Combine(installationPath, @"Common7\IDE\devenv.exe");
            Process vsProcess = Process.Start(executablePath);
        }
    }
}
```



```

        string runningObjectDisplayName = $"VisualStudio.DTE.16.0:{vsProcess.Id}";

        IEnumerable<string> runningObjectDisplayNames = null;
        object runningObject;
        for (int i = 0; i < 60; i++)
        {
            try
            {
                runningObject = GetRunningObject(runningObjectDisplayName, out runningObjectDisplayNames);
            }
            catch
            {
                runningObject = null;
            }

            if (runningObject != null)
            {
                return (EnvDTE.DTE)runningObject;
            }

            Thread.Sleep(millisecondsTimeout: 1000);
        }

        throw new TimeoutException($"Failed to retrieve DTE object. Current running objects:
{string.Join(";", runningObjectDisplayNames)}");
    }

    private static object GetRunningObject(string displayName, out IEnumerable<string>
runningObjectDisplayNames)
    {
        IBindCtx bindContext = null;
        NativeMethods.CreateBindCtx(0, out bindContext);

        IRunningObjectTable runningObjectTable = null;
        bindContext.GetRunningObjectTable(out runningObjectTable);

        IEnumMoniker monikerEnumerator = null;
        runningObjectTable.EnumRunning(out monikerEnumerator);

        object runningObject = null;
        List<string> runningObjectDisplayNameList = new List<string>();
        IMoniker[] monikers = new IMoniker[1];
        IntPtr numberFetched = IntPtr.Zero;
        while (monikerEnumerator.Next(1, monikers, numberFetched) == 0)
        {
            IMoniker moniker = monikers[0];

            string objectDisplayName = null;
            try
            {
                moniker.GetDisplayName(bindContext, null, out objectDisplayName);
            }
            catch (UnauthorizedAccessException)
            {
                // Some ROT objects require elevated permissions.
            }

            if (!string.IsNullOrEmpty(objectDisplayName))
            {
                runningObjectDisplayNameList.Add(objectDisplayName);
                if (objectDisplayName.EndsWith(displayName, StringComparison.Ordinal))
                {
                    runningObjectTable.GetObject(moniker, out runningObject);
                    if (runningObject == null)
                    {
                        throw new InvalidOperationException($"Failed to get running object with display
name {displayName}");
                    }
                }
            }
        }
    }

```

```

        }
    }

    runningObjectDisplayNames = runningObjectDisplayNameList;
    return runningObject;
}

private static ISetupInstance GetSetupInstance(bool isPreRelease)
{
    return GetSetupInstances().First(i => IsPreRelease(i) == isPreRelease);
}

private static IEnumerable<ISetupInstance> GetSetupInstances()
{
    ISetupConfiguration setupConfiguration = new SetupConfiguration();
    IEnumSetupInstances enumerator = setupConfiguration.EnumInstances();

    int count;
    do
    {
        ISetupInstance[] setupInstances = new ISetupInstance[1];
        enumerator.Next(1, setupInstances, out count);
        if (count == 1 &&
            setupInstances != null &&
            setupInstances.Length == 1 &&
            setupInstances[0] != null)
        {
            yield return setupInstances[0];
        }
    }
    while (count == 1);
}

private static bool IsPreRelease(ISetupInstance setupInstance)
{
    ISetupInstanceCatalog setupInstanceCatalog = (ISetupInstanceCatalog)setupInstance;
    return setupInstanceCatalog.IsPrerelease();
}

private static class NativeMethods
{
    [DllImport("ole32.dll")]
    public static extern int CreateBindCtx(uint reserved, out IBindCtx ppbc);

    [DllImport("ole32.dll")]
    public static extern void GetRunningObjectTable(int reserved, out IRunningObjectTable prot);
}
}

```

Vea también

- [Localización de Visual Studio](#)
- [Tutorial: Acceda al objeto DTE desde una extensión de editor](#)

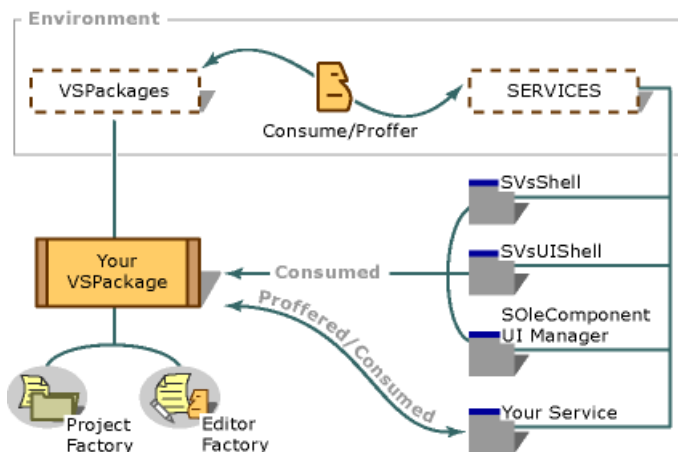
Dentro de Visual Studio SDK

06/04/2020 • 21 minutes to read • [Edit Online](#)

En esta sección se proporciona información detallada acerca de las extensiones de Visual Studio, incluida la arquitectura de Visual Studio, componentes, servicios, esquemas, utilidades y similares.

Arquitectura de extensibilidad

En la siguiente ilustración se muestra la arquitectura de extensibilidad de Visual Studio. VSPackages proporcionan funcionalidad de aplicación, que se comparte entre el IDE como servicios. El IDE estándar también ofrece una amplia [SVsUIShell](#) gama de servicios, como [SVsUIShell](#), que proporcionan acceso a la funcionalidad de ventanas IDE.



Vista generalizada de la arquitectura de Visual Studio

VSPackages

Los VSPackages son módulos de software que conforman y amplían Visual Studio con elementos de interfaz de usuario, servicios, proyectos, editores y diseñadores. VSPackages son la unidad arquitectónica central de Visual Studio. Para obtener más información, consulte [VSPackages](#).

Visual Studio Shell

El shell de Visual Studio proporciona funcionalidad básica y admite la comunicación cruzada entre sus componentes VSPackages y extensiones MEF. Para obtener más información, vea Shell de [Visual Studio](#).

Directrices de la experiencia de usuario

Si planea diseñar nuevas características para Visual Studio, debe echar un vistazo a estas directrices para sugerencias de diseño y facilidad de uso: [Directrices de experiencia del usuario de Visual Studio](#).

Comandos

Los comandos son funciones que realizan tareas, como la impresión de un documento, la actualización de una vista o la creación de un archivo nuevo.

Al ampliar Visual Studio, puede crear comandos y registrarlos con el shell de Visual Studio. Puede especificar cómo aparecerán estos comandos en el IDE, por ejemplo, en un menú o barra de herramientas. Normalmente aparece un comando personalizado en el menú **Herramientas** y un comando para mostrar una ventana de herramientas aparecería en el submenú **Otras ventanas** del menú **Ver**.

Al crear un comando, también debe crear un controlador de eventos para él. El controlador de eventos determina cuándo el comando está visible o habilitado, le permite modificar su texto y garantiza que el comando responde correctamente cuando se activa. En la mayoría de los casos, [IOleCommandTarget](#) el IDE controla los comandos mediante la interfaz. Los comandos de Visual Studio se controlan a partir del contexto de comandos más interno, basado en la selección local, y procediendo al contexto más externo, en función de la selección global. Los comandos agregados al menú principal están disponibles inmediatamente para los scripts.

Para obtener más información, consulte [Comandos, menús y barras de herramientas](#).

Menús y barras de herramientas

Los menús y las barras de herramientas proporcionan una manera para que los usuarios invoquen comandos. Los menús son filas o columnas de comandos que normalmente se muestran como elementos de texto individuales en la parte superior de una ventana de herramientas. Los submenús son menús secundarios que aparecen cuando un usuario hace clic en comandos que incluyen una flecha pequeña. Los menús contextuales aparecen cuando un usuario hace clic con el botón derecho en determinados elementos de la interfaz de usuario. Algunos nombres de menú comunes son **Archivo**, **Editar**, **Ver** y **Ventana**. Para obtener más información, consulte [Ampliación de menús y comandos](#).

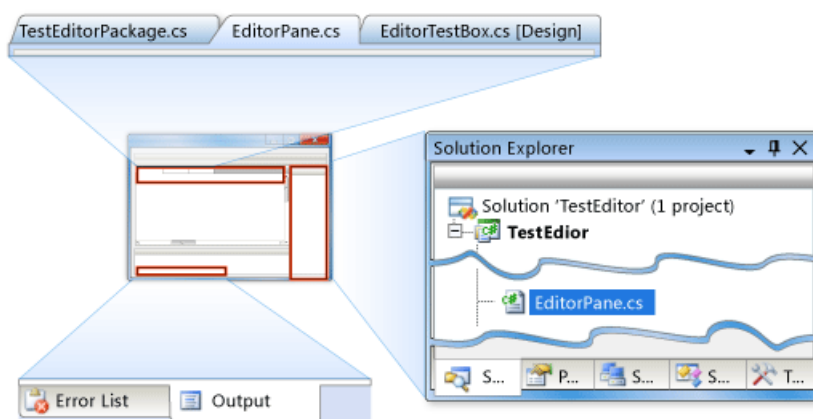
Las barras de herramientas son filas o columnas de botones y otros controles, como cuadros combinados, cuadros de lista y cuadros de texto. Los botones de la barra de herramientas suelen tener imágenes de icono, como un icono de carpeta para un comando **Abrir archivo** o una impresora para un comando **Imprimir**. Todos los elementos de la barra de herramientas están asociados con comandos. Al hacer clic en un botón de la barra de herramientas, se ejecuta su comando asociado. En el caso de un control desplegable, cada elemento de la lista desplegable está asociado a un comando diferente. Algunos controles de barra de herramientas, como un control divisor, son híbridos. Un lado del control es un botón de barra de herramientas y el otro lado es una flecha hacia abajo que muestra varios comandos cuando se hace clic en él.

Ventanas de herramientas

Las ventanas de herramientas se utilizan en el IDE para mostrar información. **Toolbox**, **Explorador de soluciones**, ventana **Propiedades** y **Explorador web** son ejemplos de ventanas de herramientas.

Las ventanas de herramientas suelen ofrecer varios controles con los que el usuario puede interactuar. Por ejemplo, la ventana **Propiedades** permite al usuario establecer propiedades de objetos que sirven para un propósito determinado. La ventana **Propiedades** está especializada en este sentido, pero también general porque se puede utilizar en muchas situaciones diferentes. De forma similar, la ventana **Salida** está especializada porque proporciona salida basada en texto, pero general porque muchos subsistemas en Visual Studio pueden usarlo para proporcionar salida al usuario de Visual Studio.

Considere la siguiente imagen de Visual Studio, que contiene varias ventanas de herramientas:



Algunas de las ventanas de herramientas se acoplan juntas en un único panel que muestra la ventana de

herramientas del Explorador de soluciones y oculta las otras ventanas de herramientas, pero las pone a disposición haciendo clic en las pestañas. La imagen muestra otras dos ventanas de herramientas, la **ventana Lista** de errores y **Salida**, acopladas en un solo panel.

También se muestra el panel principal del documento, que muestra varias ventanas del editor. Aunque las ventanas de herramientas suelen tener una sola instancia (por ejemplo, solo puede abrir un **Explorador** de soluciones), las ventanas del editor pueden tener varias instancias, cada una de las cuales se utiliza para editar un documento independiente, pero todas ellas acopladas en el mismo panel. La imagen muestra un panel de documento que tiene dos ventanas del editor, una ventana del diseñador de formularios. Todas las ventanas del panel de documentos están disponibles haciendo clic en las pestañas, pero la ventana del editor que contiene EditorPane.cs archivo está visible y activa.

Al ampliar Visual Studio, puede crear ventanas de herramientas que permiten a los usuarios de Visual Studio interactuar con la extensión. También puede crear sus propios editores que permiten a los usuarios de Visual Studio editar documentos. Dado que las ventanas de herramientas y los editores se integrarán en Visual Studio, no es necesario programarlos para acoplarlos o aparecer en una pestaña correctamente. Cuando se registran correctamente en Visual Studio, tendrán automáticamente las características típicas de las ventanas de herramientas y ventanas de documentos en Visual Studio. Para obtener más información, consulte [Ampliación y personalización](#) de las ventanas de herramientas .

Ventanas de documento

Una ventana de documento es una ventana secundaria enmarcada de una ventana de interfaz de varios documentos (MDI). Las ventanas de documento normalmente se usan para hospedar editores de texto, editores de formularios (también conocidos como diseñadores) o controles de edición, pero también pueden hospedar otros tipos funcionales. El cuadro de diálogo **Nuevo archivo** incluye ejemplos de ventanas de documento que proporciona Visual Studio.

La mayoría de los editores son específicos de un lenguaje de programación o de un tipo de archivo, como páginas HTML, conjuntos de marcos, archivos C++ o archivos de encabezado. Al seleccionar una plantilla en el cuadro de diálogo **Nuevo archivo**, un usuario crea dinámicamente una ventana de documento para el editor para el tipo de archivo asociado a la plantilla. También se crea una ventana de documento cuando un usuario abre un archivo existente.

Las ventanas de documento están restringidas al área de cliente MDI. Cada ventana del documento tiene una pestaña en la parte superior, y el orden de tabulación está vinculado a otras ventanas que pueden estar abiertas en el área MDI. Al hacer clic con el botón derecho en la pestaña de una ventana de documento, se muestra un menú contextual que incluye opciones para dividir el área MDI en varios grupos de fichas horizontales o verticales. La división del área MDI permite ver varios archivos al mismo tiempo. Para obtener más información, consulte [Document Windows](#).

Editores

El editor de Visual Studio le permite personalizarlo y usarlo para su propio tipo de contenido mediante Managed Extensibility Framework (MEF). En muchos casos no necesitará crear un VSPackage para ampliar el editor, aunque si desea incluir características del shell (por ejemplo, un comando de menú o una tecla de método abreviado), puede combinar una extensión MEF con un VSPackage.

También puede crear un editor personalizado, por ejemplo, si desea leer y escribir en una base de datos, o si desea utilizar un diseñador. También puede utilizar un editor externo como el Bloc de notas o Microsoft WordPad. Para obtener más información, consulte [Extensiones](#) de servicio de editor y lenguaje .

Servicios de idiomas

Si desea que el editor de Visual Studio admita nuevas palabras clave de programación o incluso un nuevo lenguaje

de programación, cree un servicio de lenguaje. Cada servicio de lenguaje puede implementar ciertas características del editor total, parcialmente o no en absoluto. Dependiendo de cómo se configure, el servicio de lenguaje puede proporcionar resaltado de sintaxis, coincidencia de llaves, compatibilidad con IntelliSense y otras características en el editor.

En el corazón de un servicio de idiomas hay un analizador y un escáner. Un analizador (o lexer) divide un archivo de origen en elementos que se conocen como tokens y un analizador establece las relaciones entre esos tokens. Al crear un servicio de lenguaje, debe implementar el analizador y el analizador para que Visual Studio pueda comprender los tokens y la gramática del lenguaje. Puede crear servicios de lenguaje administrados o no administrados. Para obtener más información, consulte [Extensibilidad](#) del servicio de lenguaje heredado .

Proyectos

En Visual Studio, los proyectos son los contenedores que los desarrolladores usan para organizar y compilar el código fuente y otros recursos. Los proyectos le permiten organizar, compilar, depurar e implementar código fuente, referencias a servicios web y bases de datos y otros recursos. VSPackages puede extender el sistema de proyectos de Visual Studio proporcionando tipos de proyecto, subtipos de proyecto y herramientas personalizadas.

Los proyectos también se pueden reunir en una *solución*, que es una agrupación de uno o más proyectos que trabajan juntos para crear una aplicación. La información de proyecto y estado que pertenece a la solución se almacena en dos archivos de solución, el archivo de solución basada en texto ([.sln](#)) y el archivo de opción de [usuario \(.suo\)](#) de la solución binaria. Estos archivos son similares a los archivos de grupo (.vbg) que se usaban en versiones anteriores de Visual Basic y los archivos de área de trabajo (.dsw) y opciones de usuario (.opt) que se usaban en versiones anteriores de C++.

Para obtener más información, consulte [Proyectos](#) y [soluciones](#).

Plantillas de proyecto y elemento

Visual Studio incluye plantillas de proyecto predefinidas y plantillas de elementos de proyecto. También puede crear sus propias plantillas o adquirir plantillas de la comunidad y, a continuación, integrarlas en Visual Studio. [MSDN Code Gallery](#) es el lugar a donde ir para plantillas y extensiones.

Las plantillas contienen la estructura del proyecto y los archivos básicos necesarios para crear un tipo determinado de aplicación, control, biblioteca o clase. Cuando desee desarrollar software similar a una de las plantillas, cree un proyecto basado en la plantilla y, a continuación, modifique los archivos de ese proyecto.

NOTE

Esta arquitectura de plantilla Visual C++ no se admite para proyectos.

Para obtener más información, vea Agregar plantillas de [proyecto y elemento](#) de proyecto .

Propiedades y opciones

La ventana **Propiedades** muestra las propiedades de uno o varios elementos seleccionados: [extender](#) las páginas Opciones de propiedades contiene conjuntos de opciones que pertenecen a un componente determinado, como un lenguaje de programación o un VSPackage: [páginas de opciones y opciones](#). La configuración suele ser características relacionadas con la interfaz de usuario que se pueden importar y exportar: [Compatibilidad con la configuración de usuario](#).

Servicios de Visual Studio

Un servicio proporciona un conjunto específico de interfaces para que los componentes se consuman. Visual Studio proporciona un conjunto de servicios que pueden usar cualquier componente, incluidas las extensiones. Por ejemplo, los servicios de Visual Studio permiten que las ventanas de herramientas se muestren u oculten dinámicamente, habilitan el acceso a la Ayuda, la barra de estado o los eventos de la interfaz de usuario. El editor de Visual Studio también proporciona servicios que se pueden importar mediante extensiones de editor. Para obtener más información, consulte [Uso y suministro de servicios](#).

Depurador

El depurador es la interfaz de usuario de los componentes de depuración específicos del idioma. Si ha creado un nuevo servicio de lenguaje, deberá crear un motor de depuración específico para enlazarlo al depurador. Para obtener más información, vea [Extensibilidad del depurador](#) de Visual Studio .

Control de código fuente

Para obtener información acerca de cómo implementar un complemento de control de código fuente o VSPackage, vea [Control de código fuente](#).

Asistentes

Puede crear un asistente junto con un nuevo tipo de proyecto, de modo que el asistente pueda ayudar a los usuarios a tomar las decisiones correctas cuando creen un nuevo proyecto de ese tipo. Para obtener más información, consulte [Asistentes](#).

Herramientas personalizadas

Las herramientas personalizadas le permiten asociar una herramienta con un elemento de un proyecto y ejecutarla siempre que se guarde el archivo. Para obtener más información, consulte [Herramientas personalizadas](#).

Utilidades VSSDK

El VSSDK incluye un conjunto de utilidades que puede necesitar para trabajar con diferentes aspectos de VSPackages. Para obtener más información, consulte [Utilidades VSSDK](#).

Uso de Windows Installer

En algunos casos, es posible que deba utilizar Windows Installer en lugar del instalador de VSIX: por ejemplo, es posible que deba escribir en el registro. Para obtener información sobre el uso de Windows Installer con las extensiones, vea [Instalar VSPackages con Windows Installer](#).

Visor de Ayuda

Puede integrar sus propias páginas de ayuda y F1 en el Visor de ayuda. Para obtener más información, consulte SDK del Visor de Ayuda de [Microsoft](#).

Visual Studio Shell

06/04/2020 • 4 minutes to read • [Edit Online](#)

El Visual Studio shell es el agente Visual Studio principal de integración en . El shell proporciona la funcionalidad necesaria para permitir vsPackages para compartir servicios comunes. Dado que el Visual Studio objetivo arquitectónico es establecer la funcionalidad principal en los VSPackages, el shell es un marco para proporcionar funcionalidad básica y admitir la comunicación cruzada entre sus componentes VSPackages.

Responsabilidades de Shell

El shell tiene las siguientes responsabilidades clave:

- Compatibilidad con elementos básicos (a través de interfaces COM) de la interfaz de usuario (UI). Estos incluyen menús y barras de herramientas predeterminados, marcos de ventanas de documento o ventanas secundarias de interfaz multidocumento (MDI), marcos de ventanas de herramientas y compatibilidad con el acoplamiento.
- Mantener una lista en ejecución de todos los documentos abiertos actualmente en una tabla de documentos en ejecución (RDT) con el fin de coordinar la persistencia de los documentos y garantizar que un documento no se puede abrir de más de una manera, o de maneras incompatibles.
- Compatible con la interfaz de enrutamiento `IOleCommandTarget` de comandos y gestión de comandos, .
- Carga de VSPackages en los momentos adecuados. Retrasar la carga de un VSPackage es necesario para mejorar el rendimiento del shell.
- Administración de determinados [SVShell](#) servicios compartidos, como , [SVUIShell](#) que proporciona funcionalidad básica de shell y , que proporciona funcionalidad básica de ventanas.
- Administración de los archivos de solución (.sln). Las soluciones contienen grupos de proyectos relacionados, similares a los archivos de área de trabajo (.dsw) en Visual C++ 6.0.
- Seguimiento de la selección, el contexto y la moneda de todo el shell. El shell realiza un seguimiento de los siguientes tipos de elementos:
 - El proyecto actual
 - El elemento de proyecto actual o ItemID el actual [IVsHierarchy](#)
 - La selección actual para la ventana **Propiedades** o `SelectionContainer`
 - Los IDs de contexto de interfaz de usuario o CmdUIGuids que controlan la visibilidad de comandos, menús y barras de herramientas
 - Los elementos activos actualmente, como la ventana activa, el documento y el administrador de deshacer
 - Los atributos de contexto de usuario que impulsan la Ayuda dinámica

El shell también media la comunicación entre VSPackages instalados y los servicios actuales. Es compatible con las características principales del shell y Visual Studio las pone a disposición de todos los VSPackages integrados en . Estas características principales incluyen los siguientes elementos:

- **Acerca del** cuadro de diálogo y la pantalla de presentación
- **Agregar nuevo y Agregar elemento existente** cuadros de diálogo

- **Ventana Vista de clases y Navegador de objetos**
- **Cuadro de diálogo Referencias**
- **Ventana Esquema del documento**
- **Ventana de Ayuda dinámica**
- **Buscar y reemplazar**
- **Abra** los cuadros de diálogo Proyecto y **Abrir archivo** en el menú **Nuevo**
- **Cuadro de diálogo Opciones del menú Herramientas**
- **Ventana Propiedades**
- **Explorador de soluciones**
- **Ventana Lista de tareas**
- **Cuadro de herramientas**

Vea también

- [IOleCommandTarget](#)
- [IVsHierarchy](#)
- [SVsShell](#)
- [SVsUIShell](#)
- [VSPackages](#)

Comandos, menús y barras de herramientas

06/04/2020 • 6 minutes to read • [Edit Online](#)

Los menús y las barras de herramientas son la forma en que los usuarios tienen acceso a los comandos en el VSPackage. Los comandos son funciones que realizan tareas, como la impresión de un documento, la actualización de una vista o la creación de un archivo nuevo. Los menús y las barras de herramientas son maneras gráficas y cómodas de presentar los comandos a los usuarios. Normalmente, los comandos relacionados se agrupan en clústeres en el mismo menú o en la misma barra de herramientas.

- Los menús suelen mostrarse como cadenas de una palabra agrupadas en clúster en una fila de la parte superior del entorno de desarrollo integrado (IDE) o una ventana de herramientas. Los menús también se pueden mostrar como resultado de un evento de botón derecho y se conocen como menús contextuales en ese contexto. Al hacer clic en ellos, los menús se expanden para mostrar uno o varios comandos. Cuando se hace clic en los comandos, pueden llevar a cabo tareas o iniciar submenús que contengan comandos adicionales. Algunos nombres de menú conocidos son **Archivo**, **Editar**, **Ver Ventana**. Para obtener más información, consulte [Extender menús y comandos](#).
- Las barras de herramientas normalmente son filas de botones y otros controles, como cuadros combinados, cuadros de lista, cuadros de texto y controladores de menús. Todos los controles de barra de herramientas están asociados a comandos. Cuando se hace clic en un botón de barra de herramientas, se activa su comando asociado. Los botones de barra de herramientas suelen tener iconos que sugieren los comandos subyacentes, como una impresora para un comando Imprimir. En un control de lista desplegable, cada elemento de la lista está asociado a un comando distinto. Un controlador de menú es un híbrido en el que un lado del control es un botón de barra de herramientas y el otro lado es una flecha hacia abajo que muestra comandos adicionales cuando se hace clic en ella. Para obtener más información, consulte [Agregar un controlador de menú a una barra de herramientas](#).
- Cuando se crea un comando, también debe crear un controlador de eventos para él. El controlador de eventos determina si el comando es visible o está activado, permite modificar su texto y garantiza que el comando responde correctamente ("enruta") cuando se activa. En la mayoría de los casos, el IDE controla los comandos mediante la interfaz [IOleCommandTarget](#). Los comandos de Visual Studio enrutan de forma jerárquica, empezando por el contexto de comandos más interno, según la selección local, y continuando hasta el contexto más externo, según la selección global. Los comandos agregados al menú principal están disponibles inmediatamente para los scripts. Para obtener más información, vea [MenuCommands frente a OleMenuCommands](#) y [Selection objetos de contexto](#).

Para definir nuevos menús y barras de herramientas, debe describirlos en un archivo de tabla de comandos de Visual Studio (.vsct). La plantilla de paquete de Visual Studio crea este archivo automáticamente, junto con los elementos necesarios para admitir los comandos, barras de herramientas y editores que seleccionó en la plantilla. Como alternativa, puede escribir su propio archivo .vsct mediante el esquema XML descrito aquí: referencia de [esquema XML DE VSCT](#).

Para obtener más información sobre cómo trabajar con archivos .vsct, vea [Visual Studio command table \(.vsct\) files](#).

Los temas de esta sección explican cómo funcionan los comandos, menús y barras de herramientas en VSPackages.

En esta sección

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Una descripción detallada de la especificación de formato de tabla de comandos.

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Describe una sintaxis basada en XML y un compilador para tablas de comandos.

- [Colocación predeterminada de comandos, grupos y barras de herramientas](#)

Describe comandos, grupos, menús y barras de herramientas predefinidos.

- [Comandos, menús y grupos definidos por IDE](#)

Especifica los menús predefinidos, comandos y grupos Visual Studio de comandos disponibles para su uso por el IDE.

- [Diseño de comandos](#)

Explica cómo diseñar comandos.

- [Optimizar los comandos del menú y la barra de herramientas](#)

Proporciona directrices para los comandos.

- [Hacer que los comandos estén disponibles](#)

Explica cómo hacer que los comandos estén disponibles para Visual Studio.

- [Comandos y menús que utilizan ensamblados de interoperabilidad](#)

Explica cómo implementar comandos que usan ensamblados de interoperabilidad.

Secciones relacionadas

- [Enrutamiento de comandos en VSPackages](#)

Explica el enrutamiento de comandos en VSPackages.

Cómo VSPackages agregan elementos de interfaz de usuario

06/04/2020 • 26 minutes to read • [Edit Online](#)

Un VSPackage puede agregar elementos de interfaz de usuario (UI), por ejemplo, menús, barras de herramientas y ventanas de herramientas, a Visual Studio mediante el archivo `.vsct`.

Puede encontrar directrices de diseño para elementos de interfaz de usuario en las directrices de experiencia de usuario de [Visual Studio](#).

La arquitectura de la tabla de comandos de Visual Studio

Como se ha indicado, la arquitectura de la tabla de comandos admite los principios arquitectónicos anteriores. Los principios detrás de las abstracciones, estructuras de datos y herramientas de la arquitectura de la tabla de comandos son los siguientes:

- Hay tres tipos básicos de elementos: menús, comandos y grupos. Los menús se pueden exponer en la interfaz de usuario como menús, submenús, barras de herramientas o ventanas de herramientas. Los comandos son procedimientos que el usuario puede ejecutar en el IDE y se pueden exponer como elementos de menú, botones, cuadros de lista u otros controles. Los grupos son contenedores para menús y comandos.
- Cada elemento se especifica mediante una definición que describe el elemento, su prioridad con respecto a otros elementos y las marcas que modifican su comportamiento.
- Cada elemento tiene una ubicación que describe el elemento primario del elemento. Un elemento puede tener varios elementos primarios, por lo que puede aparecer en varias ubicaciones de la interfaz de usuario.

Cada comando debe tener un grupo como su elemento primario, incluso si es el único elemento secundario de ese grupo. Cada menú estándar también debe tener un grupo primario. Las barras de herramientas y las ventanas de herramientas actúan como sus propios padres. Un grupo puede tener como elemento primario la barra de menús principal de Visual Studio o cualquier menú, barra de herramientas o ventana de herramientas.

Cómo se definen los elementos

Un archivo `.vsct` tiene formato XML. Define los elementos de la interfaz de usuario para un paquete y determina dónde aparecen esos elementos en el IDE. A cada menú, grupo o comando del paquete se `Symbol` le asigna primero un GUID y un identificador en la sección. En el resto del archivo `.vsct`, cada menú, comando y grupo se identifica mediante su combinación GUID e ID. En el ejemplo `Symbol` siguiente se muestra una sección típica generada por la plantilla de paquete de Visual Studio cuando se selecciona un comando de **menú** en la plantilla.

```

<Symbols>
  <!-- This is the package guid. -->
  <GuidSymbol name="guidMenuTextPkg" value="{b1253bc6-d266-402b-89e7-5e3d3b22c746}" />

  <!-- This is the guid used to group the menu commands together -->
  <GuidSymbol name="guidMenuTextCmdSet" value="{a633d4e4-6c65-4436-a138-1abe7c9a69}">

    <IDSymbol name="MyMenuGroup" value="0x1020" />
    <IDSymbol name="cmdidMyCommand" value="0x0100" />
  </GuidSymbol>

  <GuidSymbol name="guidImages" value="{53323d9a-972d-4671-bb5b-9e418480922f}" >
    <IDSymbol name="bmpPic1" value="1" />
    <IDSymbol name="bmpPic2" value="2" />
    <IDSymbol name="bmpPicSearch" value="3" />
    <IDSymbol name="bmpPicX" value="4" />
    <IDSymbol name="bmpPicArrows" value="5" />
  </GuidSymbol>
</Symbols>

```

El elemento de nivel `Symbols` superior de la sección es el [elemento GuidSymbol](#). `GuidSymbol` los elementos asignan nombres a GUID que utiliza el IDE para identificar paquetes y sus componentes.

NOTE

Los GUID se generan automáticamente mediante la plantilla de paquete de Visual Studio. También puede crear un GUID único haciendo clic en **Crear GUID** en el menú **Herramientas**.

El `GuidSymbol` primer `guid<PackageName>Pkg` elemento, es el GUID del propio paquete. Este es el GUID que usa Visual Studio para cargar el paquete. Normalmente, no tiene elementos secundarios.

Por convención, los menús `GuidSymbol` y `guid<PackageName>CmdSet` los comandos se agrupan `GuidSymbol` en `guidImages` un segundo elemento, y los mapas de bits están bajo un tercer elemento, . No es necesario seguir esta convención, pero cada menú, grupo, comando `GuidSymbol` y mapa de bits debe ser un elemento secundario de un elemento.

En el `GuidSymbol` segundo elemento, que representa el `IDSymbol` conjunto de comandos de paquete, hay varios elementos. Cada [elemento IDSymbol](#) asigna un nombre a un valor numérico y puede representar un menú, grupo o comando que forma parte del conjunto de comandos. Los `IDSymbol` elementos `GuidSymbol` del tercer elemento representan mapas de bits que se pueden utilizar como iconos para los comandos. Dado que los pares GUID/ID deben ser únicos `GuidSymbol` en una aplicación, no hay dos elementos secundarios del mismo elemento puede tener el mismo valor.

Menús, grupos y comandos

Cuando un menú, grupo o comando tiene un GUID y un identificador, se puede agregar al IDE. Cada elemento de interfaz de usuario debe tener las siguientes cosas:

- Atributo `guid` que coincide con `GuidSymbol` el nombre del elemento en el que se define el elemento de interfaz de usuario.
- Atributo `id` que coincide con el `IDSymbol` nombre del elemento asociado.

Juntos, `guid` `id` los atributos y componen la *firma* del elemento de interfaz de usuario.

- Atributo `priority` que determina la ubicación del elemento de interfaz de usuario en su menú o grupo primario.
- Un [elemento](#) Primario `guid` que tiene y `id` atributos que especifican la firma del menú o grupo

primario.

Menús

Cada menú se define como `Menus` un elemento `Menu` en la sección. Los menús `id` deben `priority` tener `guid`, `Parent`, y atributos, y un elemento, así como los siguientes atributos adicionales y elementos secundarios:

- Atributo `type` que especifica si el menú debe aparecer en el IDE como un tipo de menú o como una barra de herramientas.
- Un elemento `Strings` que contiene un elemento `ButtonText`, que especifica el título del menú en el IDE, y un elemento `CommandName`, que especifica el nombre que se utiliza en la ventana **Comando** para tener acceso al menú.
- Marcas opcionales. Un `CommandFlag` elemento puede aparecer en una definición de menú para cambiar su apariencia o comportamiento en el IDE.

Cada `Menu` elemento debe tener un grupo como su elemento primario, a menos que sea un elemento acoplable como una barra de herramientas. Un menú acoplable es su propio elemento primario. Para obtener más información acerca `type` de los menús y valores para el atributo, consulte la [menu elemento documentación](#).

En el ejemplo siguiente se muestra un menú que aparece en la barra de menús de Visual Studio, junto al menú **Herramientas**.

```
<Menu guid="guidTopLevelMenuCmdSet"
id="TopLevelMenu" priority="0x700" type="Menu">
  <Parent guid="guidSHLMainMenu"
    id="IDG_VS_MM_TOOLSADDINS" />
  <Strings>
    <ButtonText>TestMenu</ButtonText>
    <CommandName>TestMenu</CommandName>
  </Strings>
</Menu>
```

Grupos

Un grupo es un elemento `Groups` que se define en la sección del archivo `.vsct`. Los grupos son sólo contenedores. No aparecen en el IDE excepto como una línea divisoria en un menú. Por lo tanto, un `Group` elemento se define sólo por su firma, prioridad y elemento primario.

Un grupo puede tener un menú, otro grupo o a sí mismo como elemento primario. Sin embargo, el elemento primario suele ser un menú o una barra de herramientas. El menú del ejemplo anterior es `IDG_VS_MM_TOOLSADDINS` un elemento secundario del grupo y ese grupo es un elemento secundario de la barra de menús de Visual Studio. El grupo del ejemplo siguiente es un elemento secundario del menú en el ejemplo anterior.

```
<Group guid="guidTopLevelMenuCmdSet" id="MyMenuGroup"
priority="0x0600">
  <Parent guid="guidTopLevelMenuCmdSet" id="TopLevelMenu"/>
</Group>
```

Dado que forma parte de un menú, este grupo normalmente contendría comandos. Sin embargo, también podría contener otros menús. Así es como se definen los submenús, como se muestra en el ejemplo siguiente.

```
<Menu guid="guidTopLevelMenuCmdSet" id="SubMenu"
priority="0x0100" type="Menu">
  <Parent guid="guidTopLevelMenuCmdSet" id="MyMenuGroup"/>
  <Strings>
    <ButtonText>Sub Menu</ButtonText>
    <CommandName>Sub Menu</CommandName>
  </Strings>
</Menu>
```

Comandos

Un comando que se proporciona al IDE se define como un [Button elemento](#) o un [Combo elemento](#). Para aparecer en un menú o una barra de herramientas, el comando debe tener un grupo como su elemento primario.

Botones

Los botones `Buttons` se definen en la sección. Cualquier elemento de menú, botón u otro elemento en el que un usuario haga clic para ejecutar un único comando se considera un botón. Algunos tipos de botones también pueden incluir la funcionalidad de lista. Los botones tienen los mismos atributos obligatorios y opcionales que los menús y también pueden tener un [elemento Icon](#) que especifica el GUID y el identificador del mapa de bits que representa el botón en el IDE. Para obtener más información acerca de los botones y sus atributos, consulte la documentación [del elemento Buttons](#).

El botón del ejemplo siguiente es un elemento secundario del grupo en el ejemplo anterior y aparecería en el IDE como un elemento de menú en el menú primario de ese grupo.

```
<Button guid="guidTopLevelMenuCmdSet" id="cmdidTestCommand" priority="0x0100" type="Button">
  <Parent guid="guidTopLevelMenuCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <CommandName>cmdidTestCommand</CommandName>
    <ButtonText>Test Command</ButtonText>
  </Strings>
</Button>
```

Combos

Los combos se `Combos` definen en la sección. Cada `Combo` elemento representa un cuadro de lista desplegable en el IDE. El cuadro de lista puede o no ser grabable por `type` los usuarios, dependiendo del valor del atributo del combo. Los combos tienen los mismos elementos y comportamiento que los botones y también pueden tener los siguientes atributos adicionales:

- Atributo `defaultWidth` que especifica el ancho de píxel.
- Atributo `idCommandList` que especifica una lista que contiene los elementos que se muestran en el cuadro de lista. La lista de comandos `GuidSymbol` debe declararse en el mismo nodo que contiene el combo.

En el ejemplo siguiente se define un elemento combinado.

```

<Combos>
  <Combo guid="guidFirstToolWinCmdSet"
    id="cmdidWindowsMediaFilename"
    priority="0x0100" type="DynamicCombo"
    idCommandList="cmdidWindowsMediaFilenameGetList"
    defaultWidth="130">
    <Parent guid="guidFirstToolWinCmdSet"
      id="ToolbarGroupID" />
    <CommandFlag>IconAndText</CommandFlag>
    <CommandFlag>CommandWellOnly</CommandFlag>
    <CommandFlag>StretchHorizontally</CommandFlag>
    <Strings>
      <CommandName>Filename</CommandName>
      <ButtonText>Enter a Filename</ButtonText>
    </Strings>
  </Combo>
</Combos>

```

Mapas de bits

Los comandos que se mostrarán junto `Icon` con un icono deben incluir un elemento que haga referencia a un mapa de bits mediante su GUID e identificador. Cada mapa de bits se `Bitmaps` define como un elemento `Bitmap` en la sección. Los únicos atributos necesarios para una `Bitmap` definición son `guid` y `href`, que apunta al archivo de origen. Si el archivo de origen es una tira de recursos, también se requiere un atributo `usedList` para enumerar las imágenes disponibles en la tira. Para obtener más información, consulte la documentación del [elemento Bitmap](#).

Crianza

Las siguientes reglas rigen cómo un elemento puede llamar a otro elemento como su elemento primario.

ELEMENTO	DEFINIDO EN ESTA SECCIÓN DE LA TABLA DE COMANDOS	PUUEDE ESTAR CONTENIDO (COMO PADRE, O <code>COMMANDPLACEMENTS</code> POR COLOCACIÓN EN LA SECCIÓN, O AMBOS)	PUUEDE CONTENER (DENOMINADO PADRE)
Grupo	Grupos , el IDE, otros VSPackages	Un menú, un grupo, el elemento en sí	Menús, grupos y comandos
Menú	Menus , el IDE, otros VSPackages	1 a n grupos	0 a n grupos
Barra de herramientas	Menus , el IDE, otros VSPackages	El artículo en sí	0 a n grupos
Elemento de menú	Buttons , el IDE, otros VSPackages	1 a n grupos, el elemento en sí	De -0 a n grupos
Botón	Buttons , el IDE, otros VSPackages	1 a n grupos, el elemento en sí	
Combinado	Combos , el IDE, otros VSPackages	1 a n grupos, el elemento en sí	

Menú, comando y colocación de grupos

Un menú, grupo o comando puede aparecer en más de una ubicación en el IDE. Para que un elemento aparezca en varias ubicaciones, debe agregarse a la `CommandPlacements` sección como un elemento `CommandPlacement`. Cualquier menú, grupo o comando se puede agregar como una ubicación de comandos. Sin embargo, las barras de herramientas no se pueden colocar de esta manera porque no pueden

aparecer en varias ubicaciones sensibles al contexto.

Las ubicaciones `guid` `id` de `priority` comando tienen , , y atributos. El GUID y el identificador deben coincidir con los del elemento que se coloca. El `priority` atributo rige la colocación del elemento con respecto a otros elementos. Cuando el IDE combina dos o más elementos que tienen la misma prioridad, sus ubicaciones son indefinidas porque el IDE no garantiza que los recursos del paquete se lean en el mismo orden cada vez que se compila el paquete.

Si un menú o grupo aparece en varias ubicaciones, todos los elementos secundarios de ese menú o grupo aparecerán en cada instancia.

Visibilidad y contexto de comandos

Cuando se instalan varios VSPackages, una profusión de menús, elementos de menú y barras de herramientas puede saturar el IDE. Para evitar este problema, puede controlar la visibilidad de elementos de interfaz de usuario individuales mediante restricciones de *visibilidad* indicadores de comando.

Restricciones de visibilidad

Una restricción de visibilidad se establece `VisibilityConstraints` como un `VisibilityItem` elemento en la sección. Una restricción de visibilidad define contextos de interfaz de usuario específicos en los que el elemento de destino está visible. Un menú o comando que se incluye en esta sección solo es visible cuando uno de los contextos definidos está activo. Si no se hace referencia a un menú o comando en esta sección, siempre está visible de forma predeterminada. Esta sección no se aplica a los grupos.

`VisibilityItem` los elementos deben tener tres `guid` atributos, como se indica a continuación: el elemento de interfaz de usuario de `id` destino y `context`. El `context` atributo especifica cuándo estará visible el elemento de destino y toma cualquier contexto de interfaz de usuario válido como su valor. Las constantes de contexto de interfaz de usuario para Visual Studio son miembros de la `VSConstants` clase. Cada `VisibilityItem` elemento puede tomar solo un valor de contexto. Para aplicar un segundo contexto, cree un segundo `VisibilityItem` elemento que apunte al mismo elemento, como se muestra en el ejemplo siguiente.

```
<VisibilityConstraints>
  <VisibilityItem guid="guidSolutionToolbarCmdSet"
    id="cmdidTestCmd"
    context="UICONTEXT_SolutionHasSingleProject" />
  <VisibilityItem guid="guidSolutionToolbarCmdSet"
    id="cmdidTestCmd"
    context="UICONTEXT_SolutionHasMultipleProjects" />
</VisibilityConstraints>
```

Banderas de mando

Los siguientes indicadores de comando pueden afectar a la visibilidad de los menús y comandos a los que se aplican.

`AlwaysCreate` El menú se crea incluso si no tiene grupos ni botones.

Válido para: `Menu`

`CommandWellOnly` Aplique esta marca si el comando no aparece en el menú de nivel superior y desea que esté disponible para la personalización de shell adicional, por ejemplo, enlazarlo a una clave. Después de instalar el VSPackage, un usuario puede personalizar estos comandos abriendo el cuadro de diálogo **Opciones** y, a continuación, editando la ubicación del comando en la categoría Entorno de **teclado**. No afecta a la colocación en menús contextuales, barras de herramientas, controladores de menú o submenús.

Válido para: `Button` , `Combo`

`DefaultDisabled` De forma predeterminada, el comando está deshabilitado si el `VSPackage` que implementa el comando no está cargado o el `QueryStatus` no se ha llamado al método.

Válido para: `Button`, `Combo`

`DefaultInvisible` De forma predeterminada, el comando es invisible si el `VSPackage` que implementa el comando no está cargado o el `QueryStatus` no se ha llamado al método.

Debe combinarse `DynamicVisibility` con la bandera.

Válido `Button` para: `Combo`, `Menu`

`DynamicVisibility` La visibilidad del comando se puede `QueryStatus` cambiar mediante el método o `VisibilityConstraints` un GUID de contexto que se incluye en la sección.

Se aplica a los comandos que aparecen en los menús, no en las barras de herramientas. Los elementos de la barra de herramientas `OLECMDF_INVISIBLE` de nivel superior `QueryStatus` se pueden deshabilitar, pero no ocultar, cuando se devuelve la marca desde el método.

En un menú, esta marca también indica que debe ocultarse automáticamente cuando sus miembros están ocultos. Esta marca se asigna normalmente a los submenús porque los menús de nivel superior ya tienen este comportamiento.

Debe combinarse `DefaultInvisible` con la bandera.

Válido `Button` para: `Combo`, `Menu`

`NoShowOnMenuController` Si un comando que tiene esta marca se coloca en un controlador de menú, el comando no aparece en la lista desplegable.

Válido para: `Button`

Para obtener más información acerca de los indicadores de comando, consulte la documentación del [elemento CommandFlag](#).

Requisitos generales

El comando debe pasar la siguiente serie de pruebas antes de que se pueda mostrar y habilitar:

- El comando se coloca correctamente.
- La `DefaultInvisible` marca no está establecida.
- El menú principal o la barra de herramientas está visible.
- El comando no es invisible debido a una entrada de contexto en la sección del [elemento VisibilityConstraints](#).
- Código `vsPackage` que [IOleCommandTarget](#) implementa la interfaz muestra y habilita el comando. Ningún código de interfaz lo interceptó y actuó en él.
- Cuando un usuario hace clic en el comando, está sujeto al procedimiento que se describe en Algoritmo de [enrutamiento](#).

Llamar a comandos predefinidos

El [elemento UsedCommands](#) permite a `VSPackages` tener acceso a comandos proporcionados por otros `VSPackages` o por el IDE. Para ello, cree un [elemento UsedCommand](#) que tenga el GUID y el identificador del comando que se va a usar. Esto garantiza que Visual Studio cargará el comando, incluso si no forma parte de la configuración actual de Visual Studio. Para obtener más información, vea [Elemento UsedCommand](#).

Apariencia del elemento de interfaz

Las consideraciones para seleccionar y colocar elementos de comando son las siguientes:

- Visual Studio ofrece muchos elementos de interfaz de usuario que aparecen de forma diferente dependiendo de la ubicación.
- Un elemento de interfaz `DefaultInvisible` de usuario que se define mediante el uso de la marca no [QueryStatus](#) se mostrará en el IDE `VisibilityConstraints` a menos que se muestre por su implementación de `VSPackage` del método o asociado a un contexto de interfaz de usuario determinado en la sección.
- Es posible que no se muestre incluso un comando colocado correctamente. Esto se debe a que el IDE oculta o muestra automáticamente algunos comandos, dependiendo de las interfaces que el `VSPackage` tiene (o no) implementado. Por ejemplo, la implementación de un `VSPackage` de algunas interfaces de compilación hace que los elementos de menú relacionados con la compilación se muestren automáticamente.
- Aplicar la `CommandWellOnly` marca en la definición de elemento de interfaz de usuario significa que el comando solo se puede agregar mediante personalización.
- Los comandos pueden estar disponibles solo en determinados contextos de interfaz de usuario, por ejemplo, solo cuando se muestra un cuadro de diálogo cuando el IDE está en la vista de diseño.
- Para hacer que ciertos elementos de la interfaz de usuario se muestren en el IDE, debe implementar una o varias interfaces o escribir código.

Vea también

- [Extender menús y comandos](#)

Archivos de tabla de comandos de Visual Studio (.Vsct)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un archivo de configuración de tabla de comandos es un archivo de texto que describe el conjunto de comandos que contiene un VSPackage. El Visual Studio compilador de tabla de comandos (VSCT) compila archivos de configuración basados en XML (archivos .vsct) en archivos de salida de tabla de comandos binarios (.cto). Los archivos .cto resultantes son los mismos que los que se crean mediante el compilador de tabla de comandos (CTC) para compilar archivos de configuración .ctc. Sin embargo, los archivos .vsct basados en XML tienen algunas ventajas, como un editor XML e Xml IntelliSense.

Para obtener más información sobre la sintaxis y la semántica de los archivos .vsct, consulte [Diseño de la tabla de comandos XML \(. Vsct\) Archivos](#)

En esta sección

[Diseño de archivos de tabla de comandos XML \(.Vsct\)](#)

Describe cómo diseñar archivos .vsct.

[Creación de un archivo .Vsct](#)

Compara los métodos para crear un archivo .vsct. Describe el proceso para crear manualmente un nuevo archivo .vsct.

Secciones relacionadas

[Referencia del esquema XML de VSCT](#)

Proporciona detalles sobre cada sección del archivo de configuración XML de la tabla de comandos.

Configuración de la [tabla de comandos \(. Ctc\) Archivos](#) Presenta una visión general del formato de archivo .ctc en desuso.

[Cómo VSPackages agrega elementos de la interfaz de usuario](#)

Describe la especificación de formato de tabla de comandos.

[Recursos de VSPackages](#)

Describe cómo usar recursos administrados y no administrados en VSPackages administrados.

[Comandos, menús y barras de herramientas](#)

Explica cómo crear una interfaz de usuario que incluya menús, barras de herramientas y cuadros combinados de comandos.

Crear archivos .vsct

06/04/2020 • 23 minutes to read • [Edit Online](#)

En este documento se muestra cómo crear un archivo .vsct para agregar elementos de menú, barras de herramientas y otros elementos de interfaz de usuario (UI) al entorno de desarrollo integrado (IDE) de Visual Studio. Siga estos pasos al agregar elementos de interfaz de usuario a un paquete de Visual Studio (VSPackage) que aún no tiene un archivo .vsct.

Para los nuevos proyectos, se recomienda usar la plantilla de paquete de Visual Studio porque genera un archivo .vsct que, dependiendo de las selecciones, ya tiene los elementos necesarios para un comando de menú, una ventana de herramientas o un editor personalizado. Puede modificar este archivo .vsct para cumplir los requisitos de su VSPackage. Para obtener más información sobre cómo modificar un archivo .vsct, consulte los ejemplos de [Extender menús y comandos](#).

Crear el archivo

Crear un archivo .vsct en estas fases: cree la estructura de archivos y recursos, declare los elementos de la interfaz de usuario, coloque los elementos de la interfaz de usuario en el IDE y agregue comportamientos especializados.

Estructura de archivos

La estructura básica de un archivo .vsct es un elemento raíz `CommandTable` que contiene un elemento `Commands` y un elemento `Symbols`.

Para crear la estructura de archivos

1. Agregue un archivo .vsct al proyecto siguiendo los pasos descritos en [Cómo: Crear un archivo .vsct](#).
2. Agregue los espacios de `CommandTable` nombres necesarios al elemento, como se muestra en el ejemplo siguiente:

```
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

3. En `CommandTable` el elemento, `Commands` agregue un elemento para hospedar todos los menús personalizados, barras de herramientas, grupos de comandos y comandos. Para que los elementos de `Commands` interfaz `Package` de usuario personalizados se puedan cargar, el elemento debe tener su atributo establecido en el nombre del paquete.

Después `Commands` del elemento, agregue un `Symbols` elemento para definir los GUID para el paquete y los nombres y los id de comando para los elementos de la interfaz de usuario.

Incluir recursos de Visual Studio

Use el elemento `Extern` para tener acceso a los archivos que definen los comandos de Visual Studio y los menús necesarios para colocar los elementos de interfaz de usuario en el IDE. Si va a usar comandos definidos fuera del paquete, use el elemento `UsedCommands` para informar a Visual Studio.

Para incluir recursos de Visual Studio

1. En la parte `CommandTable` superior del `Extern` elemento, agregue un elemento para `href` cada archivo externo al que se haga referencia y establezca el atributo en el nombre del archivo. Puede hacer referencia a los siguientes archivos de encabezado para tener acceso a los recursos de Visual Studio:

- `Stdidcmd.h`. Define los IDs para todos los comandos expuestos por Visual Studio.

- *Vsshlibs.h*: Contiene los elementos de comando de los menús de Visual Studio.

- Si el paquete llama a los comandos definidos por `UsedCommands` Visual `Commands` Studio o por otros paquetes, agregue un elemento después del elemento. Rellene este elemento con un [elemento `UsedCommand`](#) para cada comando al que llame que no forme parte del paquete. Establezca `guid` los `id` atributos `UsedCommand` y de los elementos en los valores GUID e ID de los comandos que se llamarán.

Para obtener más información acerca de cómo buscar los GUID y los GUID de los comandos de Visual Studio, vea [GUID e ID de comandos de Visual Studio](#). Para llamar a comandos de otros paquetes, use el GUID y el identificador del comando tal como se define en el archivo `.vsct` para esos paquetes.

Declarar elementos de interfaz de usuario

Declare todos los nuevos `Symbols` elementos de la interfaz de usuario en la sección del archivo `.vsct`.

Para declarar elementos de interfaz de usuario

- En `Symbols` el elemento, agregue tres [GuidSymbol](#) elementos. Cada `GuidSymbol` elemento `name` tiene un `value` atributo y un atributo. Establezca `name` el atributo para que refleje el propósito del elemento. El `value` atributo toma un GUID. (Para generar un GUID, en el menú **Herramientas**, seleccione **Crear GUID**, a continuación, seleccione **Formato del Registro**.)

El `GuidSymbol` primer elemento representa el paquete y normalmente no tiene elementos secundarios. El `GuidSymbol` segundo elemento representa el conjunto de comandos y contendrá todos los símbolos que definen los menús, grupos y comandos. El `GuidSymbol` tercer elemento representa el almacén de imágenes y contiene símbolos para todos los iconos de los comandos. Si no tiene comandos que utilicen `GuidSymbol` iconos, puede omitir el tercer elemento.

- En `GuidSymbol` el elemento que representa el conjunto de comandos, agregue uno o varios [elementos `IDSymbol`](#). Cada uno de ellos representa un menú, barra de herramientas, grupo o comando que se agrega a la interfaz de usuario.

Para `IDSymbol` cada elemento, `name` establezca el atributo en el nombre que utilizará para hacer referencia `value` al menú, grupo o comando correspondiente y, a continuación, establezca el elemento en un número hexadecimal que representará su identificador de comando. No `IDSymbol` hay dos elementos que tengan el mismo elemento primario puede tener el mismo valor.

- Si alguno de los elementos `IDSymbol` de la interfaz `GuidSymbol` de usuario requiere iconos, agregue un elemento para cada icono al elemento que representa el almacén de imágenes.

Colocar elementos de interfaz de usuario en el IDE

Los [elementos `Menus`](#), [Groups](#) y [Buttons](#) contienen las definiciones de todos los menús, grupos y comandos definidos en el paquete. Coloque estos menús, grupos y comandos en el IDE mediante un [elemento `Parent`](#), que forma parte de la definición del elemento UI, o mediante un elemento [CommandPlacement](#) que se define en otro lugar.

Cada `Menu`, `Group`, `Button`, y `guid` elemento `id` tiene un atributo y un atributo. Establezca siempre `guid` el atributo para `GuidSymbol` que coincida con el nombre del `id` elemento que representa `IDSymbol` el conjunto de comandos y establezca `Symbols` el atributo en el nombre del elemento que representa el menú, grupo o comando de la sección.

Para definir elementos de interfaz de usuario

- Si está definiendo nuevos menús, submenús, menús `Menus` contextuales `Commands` o barras de herramientas, agregue un elemento al elemento. A continuación, para cada menú que se `Menus` va a crear, agregue un elemento [Menu](#) al elemento.

Establezca `guid` los `id` atributos `Menu` y del elemento `type` y, a continuación, establezca el atributo en el tipo de menú que desee. También puede establecer `priority` el atributo para establecer la posición relativa

del menú en el grupo primario.

NOTE

El `priority` atributo no se aplica a las barras de herramientas y los menús contextuales.

2. Todos los comandos del IDE de Visual Studio deben hospedarse en grupos de comandos, que son los elementos secundarios directos de los menús y las barras de herramientas. Si va a agregar nuevos menús o barras de herramientas al IDE, estos deben contener nuevos grupos de comandos. También puede agregar grupos de comandos a los menús y barras de herramientas existentes para que pueda agrupar visualmente los comandos.

Al agregar nuevos grupos de comandos, primero debe crear un `Groups` elemento y, a continuación, agregarle un elemento `Group` para cada grupo de comandos.

Establezca `guid` los `id` atributos `Group` y de cada `priority` elemento y, a continuación, establezca el atributo para establecer la posición relativa del grupo en el menú primario. Para obtener más información, consulte [Crear grupos de botones reutilizables](#).

3. Si va a agregar nuevos comandos `Buttons` al IDE, agregue un elemento al `Commands` elemento. A continuación, para cada comando, `Buttons` agregue un elemento `Button` al elemento.
 - a. Establezca `guid` los `id` atributos `Button` y de cada `type` elemento y, a continuación, establezca el atributo en el tipo de botón que desee. También puede establecer `priority` el atributo para establecer la posición relativa del comando en el grupo primario.

NOTE

Se `type="button"` utiliza para los comandos de menú estándar y los botones de las barras de herramientas.

- b. En `Button` el elemento, agregue un [elemento Strings](#) que contenga un elemento `ButtonText` y un elemento `CommandName`. El `ButtonText` elemento proporciona la etiqueta de texto para un elemento de menú o la información sobre herramientas para un botón de barra de herramientas. El `CommandName` elemento proporciona el nombre del comando que se va a utilizar en el pozo de comandos.
- c. Si el comando tendrá un icono, cree un `Button` elemento `Icon` `id` en el `Bitmap` elemento y establezca sus `guid` atributos y sus atributos en el elemento del icono.

NOTE

Los botones de la barra de herramientas deben tener iconos.

Para obtener más información, vea [MenuCommands vs OleMenuCommands](#).

4. Si alguno de los comandos requiere iconos, `Commands` agregue un elemento `Bitmaps` al elemento. A continuación, para cada `Bitmap` icono, agregue `Bitmaps` un elemento `Bitmap` al elemento. Aquí es donde se especifica la ubicación del recurso de mapa de bits. Para obtener más información, consulte [Agregar iconos a los comandos](#) de menú.

Puede confiar en la estructura de crianza para colocar correctamente la mayoría de los menús, grupos y comandos. Para conjuntos de comandos muy grandes, o cuando un menú, grupo o comando debe aparecer en varios lugares, se recomienda especificar la colocación del comando.

Confiar en la crianza para colocar elementos de interfaz de usuario en el IDE

1. Para la crianza típica, `Parent` cree `Menu` `Group` un `Command` elemento en cada , , y elemento que se define en el paquete.

El destino `Parent` del elemento es el menú o grupo que contendrá el menú, grupo o comando.

- a. Establezca `guid` el atributo en `GuidSymbol` el nombre del elemento que define el conjunto de comandos. Si el elemento de destino no forma parte del paquete, utilice el guid para ese conjunto de comandos, tal como se define en el archivo `.vsct` correspondiente.
- b. Establezca `id` el atributo `id` para que coincida con el atributo del menú o grupo de destino. Para obtener una lista de los menús y grupos expuestos por Visual Studio, vea [GUID e ID de menús de Visual Studio](#) o [GUID e IDs de barras de herramientas de Visual Studio](#).

Si tiene un gran número de elementos de interfaz de usuario para colocar en el IDE, o si tiene elementos que deben aparecer en varios lugares, defina sus ubicaciones en el [Elemento CommandPlacements](#), como se muestra en los pasos siguientes.

Para usar la colocación de comandos para colocar elementos de interfaz de usuario en el IDE

1. Después del elemento `Commands` , agregue un elemento `CommandPlacements` .
2. En `CommandPlacements` el elemento, `CommandPlacement` agregue un elemento para cada menú, grupo o comando que se va a colocar.

Cada `CommandPlacement` elemento `Parent` o elemento coloca un menú, grupo o comando en una ubicación IDE. Un elemento de interfaz de usuario solo puede tener un elemento primario, pero puede tener varias ubicaciones de comandos. Para colocar un elemento de interfaz de usuario en varias ubicaciones, agregue un `CommandPlacement` elemento para cada ubicación.

3. Establezca `guid` los `id` atributos `CommandPlacement` y de cada elemento en el menú `Parent` o grupo de hospedaje, tal como lo haría para un elemento. También puede establecer `priority` el atributo para establecer la posición relativa del elemento de interfaz de usuario.

Puede mezclar la ubicación mediante la colocación de la crianza y la colocación de comandos. Sin embargo, para conjuntos de comandos muy grandes, se recomienda utilizar solo la colocación de comandos.

Añadir comportamientos especializados

Puede utilizar el elemento [CommandFlag](#) para modificar el comportamiento de los menús y comandos, por ejemplo, para cambiar su apariencia y visibilidad. También puede afectar cuando un comando está visible mediante el [elemento VisibilityConstraints](#) o agregar métodos abreviados de teclado mediante el elemento [KeyBindings](#). Ciertos tipos de menús y comandos ya tienen comportamientos especializados integrados.

Para agregar comportamientos especializados

1. Para que un elemento de interfaz de usuario sea visible solo en determinados contextos de interfaz de usuario, por ejemplo, cuando se carga una solución, use restricciones de visibilidad.
 - a. Después del elemento `Commands` , agregue un elemento `VisibilityConstraints` .
 - b. Para que cada elemento de interfaz de usuario se restrinja, agregue un elemento [VisibilityItem](#).
 - c. Para `VisibilityItem` cada elemento, `guid` `id` establezca los atributos y en el menú, grupo o comando y, a continuación, establezca el `context` atributo en el contexto de interfaz de usuario que desee, tal como se define en la [UIContextGuids80](#) clase.
2. Para establecer la visibilidad o la disponibilidad de un elemento de interfaz de usuario en el código, utilice uno o varios de los siguientes indicadores de comando:

- `DefaultDisabled`
- `DefaultInvisible`

- `DynamicItemStart`
- `DynamicVisibility`
- `NoShowOnMenuController`
- `NotInTBLList`

Para obtener más información, vea el [elemento CommandFlag](#).

3. Para cambiar la forma en que aparece un elemento o cambiar su apariencia dinámicamente, utilice uno o varios de los siguientes indicadores de comando:

- `AlwaysCreate`
- `CommandWellOnly`
- `DefaultDocked`
- `DontCache`
- `DynamicItemStart`
- `FixMenuController`
- `IconAndText`
- `Pict`
- `StretchHorizontally`
- `TextMenuUseButton`
- `TextChanges`
- `TextOnly`

Para obtener más información, vea el [elemento CommandFlag](#).

4. Para cambiar cómo reacciona un elemento cuando recibe comandos, utilice uno o varios de los siguientes indicadores de comando:

- `AllowParams`
- `CaseSensitive`
- `CommandWellOnly`
- `FilterKeys`
- `NoAutoComplete`
- `NoButtonCustomize`
- `NoKeyCustomize`
- `NoToolbarClose`
- `PostExec`
- `RouteToDocs`
- `TextIsAnchorCommand`

Para obtener más información, vea el [elemento CommandFlag](#).

5. Para adjuntar un método abreviado de teclado dependiente del menú a un menú o un `ButtonText` elemento de un menú, agregue un carácter de y comercial (&) en el elemento del menú o elemento de menú. El carácter que sigue a la y comosigue es el método abreviado de teclado activo cuando el menú principal está abierto.
6. Para asociar un método abreviado de teclado independiente del menú a un comando, utilice el [KeyBindings](#) elemento. Para obtener más información, vea el [KeyBinding](#) elemento.
7. Para localizar el texto `LocCanonicalName` del menú, utilice el elemento. Para obtener más información, consulte el [strings](#) elemento.

Algunos tipos de menús y botones incluyen comportamientos especializados. En la lista siguiente se describen algunos tipos de menús y botones especializados. Para otros tipos, `types` consulte las descripciones de atributos en los elementos [Menu](#), [Button](#) y [Combo](#).

- Cuadro combinado: un cuadro combinado es una lista desplegable que se puede utilizar en una barra de herramientas. Para agregar cuadros combinados a la interfaz de usuario, cree un [combos](#) elemento en el `Commands` elemento. A continuación, `Combos` agregue `Combo` al elemento un elemento para cada cuadro combinado que desee agregar. `Combo` los elementos tienen los `Button` mismos `DefaultWidth` atributos y elementos secundarios que los elementos y también tienen y `idCommandList` atributos. El `DefaultWidth` atributo establece el ancho `idCommandList` en píxeles y el atributo apunta a un identificador de comando que se utiliza para rellenar el cuadro combinado.
- Controlador de menú: Un controlador de menú es un botón que tiene una flecha junto a él. Al hacer clic en la flecha se abre una lista. Para agregar un controlador de menú `Menu` a la `type` interfaz de usuario, cree un elemento y establezca su atributo en `MenuController` o `MenuControllerLatched`, dependiendo del comportamiento que desee. Para rellenar un controlador de menú, `Group` establézcalo como el elemento primario de un elemento. El controlador de menú mostrará todos los elementos secundarios de ese grupo en su lista desplegable.

Vea también

- [Extender menús y comandos](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)
- [Referencia de esquema XML de VSCT](#)

Diseñar archivos de tabla de comandos XML (.vsct)

06/04/2020 • 14 minutes to read • [Edit Online](#)

Un archivo de tabla de comandos XML (.vsct) describe el diseño y la apariencia de los elementos de comando para un VSPackage. Los elementos de comando incluyen botones, cuadros combinados, menús, barras de herramientas y grupos de elementos de comando. En este artículo se describen los archivos de tabla de comandos XML, cómo afectan a los elementos de comando y los menús y cómo crearlos.

Comandos, menús, grupos y el archivo .vsct

Los archivos .vsct se organizan en torno a comandos, menús y grupos de comandos. Las etiquetas XML del archivo .vsct representan cada uno de estos elementos, junto con otros elementos asociados, como botones de comando, ubicación de comandos y mapas de bits.

Al crear un nuevo VSPackage Visual Studio ejecutando la plantilla de paquete, la plantilla genera un archivo .vsct con los elementos necesarios para un comando de menú, ventana de herramientas o editor personalizado, dependiendo de las selecciones. Este archivo .vsct, a continuación, se puede modificar para cumplir los requisitos de un VSPackage específico. Para obtener ejemplos de cómo modificar un archivo .vsct, consulte [Extender menús y comandos](#).

Para crear un nuevo archivo .vsct en blanco, consulte [Cómo: Crear un archivo .vsct](#). Una vez creado, agregue elementos XML, atributos y valores al archivo para describir el diseño del elemento de comando. Para obtener un esquema XML detallado, vea la referencia de [esquema XML de VSCT](#).

Diferencias entre archivos .ctc y .vsct

Mientras que el significado detrás de las etiquetas XML en un archivo .vsct es el mismo que el de las etiquetas en el formato de archivo .ctc ahora en desuso, su implementación es un poco diferente:

- La `** <**` nueva etiqueta de `>externo` es donde se hace referencia a otros Visual Studio archivos .h que se van a compilar, como los archivos de la barra de herramientas.
- Mientras que los archivos .vsct admiten la instrucción `/include`, como lo hacen los archivos .ctc, también cuenta con un nuevo `** <**` elemento de `>de importación`. La diferencia es, `/include` trae *toda* la información, mientras que `** <la importación> **` trae solo los nombres.
- Aunque los archivos .ctc requieren un archivo de encabezado en el que se definen las directivas de preprocesador, no es necesario uno para los archivos .vsct. En su lugar, coloque las directivas `** <**` en la tabla de símbolos, ubicada en los elementos `Symbol>`, situados en la parte inferior del archivo .vsct.
- Los archivos .vsct cuentan con una `** <etiqueta de>` de anotación, `**` que le permite incrustar cualquier información que desee, como notas o incluso imágenes.
- Los valores se almacenan como atributos en el elemento.
- Los indicadores de comando se pueden almacenar individualmente o apilar. IntelliSense, sin embargo, no funciona en indicadores de comandos apilados. Para obtener más información acerca de los indicadores de comando, vea el [elemento CommandFlag](#).
- Puede especificar varios tipos, como listas desplegables divididas, combos, etc.
- Los GUID no se validan.
- Cada elemento de interfaz de usuario tiene una cadena que representa el texto que se muestra con él.

- El elemento primario es opcional. Si se omite, se utiliza el valor *Group Unknown*.
- El *argumento Icon* es opcional.
- Sección de mapa de bits: esta sección es la misma que en un archivo *.ctc*, excepto que ahora puede especificar un nombre de archivo a través de Href que el compilador *vsct.exe* extraerá en tiempo de compilación.
- ResID: el id de recurso de mapa de bits antiguo se puede utilizar y sigue funcionando igual que en los archivos *.ctc*.
- HRef: un nuevo método que permite especificar un nombre de archivo para el recurso de mapa de bits. Se supone que se utilizan todos, por lo que puede omitir la sección Usado. El compilador buscará primero recursos locales para el archivo, luego en los recursos compartidos de red y los recursos definidos por el modificador /I.
- Enlace de teclas: ya no es que tenga que especificar un emulador. Si especifica uno, el compilador asumirá que el editor y el emulador son los mismos.
- Keychord: Keychord se ha eliminado. El nuevo formato es *Key1,Mod1,Key2,Mod2*. Puede especificar una constante de carácter, hexadecimal o VK.

El nuevo compilador, *vsct.exe*, compila archivos *.ctc* y *.vsct*. Sin embargo, el antiguo compilador *ctc.exe* no reconocerá ni compilará archivos *.vsct*.

Puede usar el compilador *vsct.exe* para convertir un archivo *.cto* existente en un archivo *.vsct*. Para obtener más información, consulte [Cómo: crear un archivo .vsct a partir de un archivo .cto existente](#).

Los elementos de archivo .vsct

La tabla de comandos tiene la siguiente jerarquía y elementos:

- **CommandTable elemento:** representa todos los comandos, grupos de menús y menús asociados con el VSPackage.
- **Elemento Externo:** Hace referencia a los archivos *.h* externos que desea combinar con el archivo *.vsct*.
- **Incluir elemento:** Hace referencia a cualquier archivo de encabezado adicional (*.h*) que desee compilar junto con el archivo *.vsct*. Un archivo *.vsct* puede incluir archivos *.h* que contienen constantes que definen comandos, grupos de menús y menús que proporciona el IDE u otro VSPackage.
- **Elemento Commands:** Representa todos los comandos individuales que se pueden ejecutar. Cada comando tiene los cuatro elementos secundarios siguientes:
 - **Elemento Menus:** Representa todos los menús y barras de herramientas en el VSPackage. Los menús son contenedores para grupos de comandos.
 - **Elemento Groups:** Representa todos los grupos del VSPackage. Los grupos son colecciones de comandos individuales.
 - **Elemento Buttons:** Representa todos los botones de comando y elementos de menú en el VSPackage. Los botones son controles visuales que se pueden asociar con comandos.
 - **Elemento Bitmaps:** Representa todos los mapas de bits para todos los botones en el VSPackage. Los mapas de bits son imágenes que se muestran junto o en los botones de comando, dependiendo del contexto.
- **Elemento CommandPlacements:** indica ubicaciones adicionales donde los comandos individuales deben colocarse en los menús de su VSPackage.
- **VisibilityConstraints elemento:** Especifica si un comando se muestra en todo momento, o sólo en ciertos

contextos, como cuando se muestra un cuadro de diálogo o ventana determinado. Los menús y comandos que tienen un valor para este elemento solo se mostrarán cuando el contexto especificado esté activo. El comportamiento predeterminado es mostrar el comando en todo momento.

- **Elemento KeyBindings:** especifica los enlaces de clave para los comandos. Es decir, una o más combinaciones de teclas que se deben presionar para ejecutar el comando, como **Ctrl+S**.
- **Elemento UsedCommands:** Visual Studio informa al entorno de que aunque el comando especificado se implementa mediante otro código, cuando el VSPackage actual está activo, proporciona la implementación del comando.
- **Elemento Símbolos:** Contiene los nombres de símbolo y los ID GUID de todos los comandos del paquete.

Directrices de diseño de archivos .vsct

Para diseñar correctamente un archivo `.vsct`, siga estas directrices.

- Los comandos solo se pueden colocar en grupos, los grupos solo se pueden colocar en los menús y los menús solo se pueden colocar en grupos. Solo los menús se muestran realmente en el IDE, los grupos y los comandos no lo son.
- Los submenús no se pueden asignar directamente a un menú, sino que deben asignarse a un grupo, que a su vez se asigna a un menú.
- Los comandos, submenús y grupos se pueden asignar a un grupo o menú de crianza mediante el campo primario de su directiva de definición.
- Organizar una tabla de comandos únicamente a través de los campos primarios de las directivas tiene una limitación significativa. Las directivas que definen objetos solo pueden tomar un argumento primario.
- La reutilización de comandos, grupos o submenús requiere el uso de una `GUID:ID` nueva directiva para crear una nueva instancia del objeto con su propio par.
- Cada `GUID:ID` par debe ser único. La `ICommandTarget` interfaz controla la reutilización de un comando que, por ejemplo, se ha colocado en un menú, una barra de herramientas o en un menú contextual.
- Los comandos y submenús también se pueden asignar a varios grupos, y los grupos se pueden asignar a varios menús mediante el **elemento Comandos**.

Notas de archivo .vsct

Si realiza cambios en un archivo `.vsct` después de compilarlo y colocarlo en un archivo DLL satélite nativo, debe ejecutar **devenv.exe /setup /nosetupvstemplates**. Al hacerlo, se vuelven a leer los recursos de VSPackage especificados en el registro experimental y se vuelve Visual Studio a generar la base de datos interna que describe que se va a volver a generar.

Durante el desarrollo, es posible que varios proyectos de VSPackage se creen y registren en el subárbol del registro experimental que puede provocar un desorden confuso en el IDE. Para solucionar este problema, puede restablecer el subárbol experimental a la configuración predeterminada para quitar todos los VSPackages registrados y los cambios que puedan haber realizado en el IDE. Para restablecer el subárbol experimental, use la herramienta `CreateExpInstance.exe` que viene con el SDK de Visual Studio. Puedes encontrarlo en:

`%PROGRAMFILES(x86)%-Visual\<Studio versión> SDK-VisualStudioIntegration-Tools-Bin-CreateExpInstance.exe`

Ejecute la herramienta mediante el comando **CreateExpInstance /Reset**. Recuerde que esta herramienta quita del subárbol experimental todos Visual Studio los VSPackages registrados que normalmente no se instalan con .

Vea también

- [Extender menús y comandos](#)

Cómo: Crear un archivo .vsct

06/04/2020 • 7 minutes to read • [Edit Online](#)

Hay varias maneras de crear un archivo de configuración de tabla de comandos de Visual Studio basado en XML (.vsct).

- Puede crear un nuevo VSPackage en la Visual Studio plantilla de paquete.
- Puede utilizar el compilador de configuración de tabla de comandos basado en XML, *Vsct.exe*, para generar un archivo a partir de un archivo .ctc existente.
- Puede usar *Vsct.exe* para generar un archivo .vsct a partir de un archivo .cto existente.
- Puede crear manualmente un nuevo archivo .vsct.

En este artículo se explica cómo crear manualmente un nuevo archivo .vsct.

Para crear manualmente un nuevo archivo .vsct

1. Inicie Visual Studio.
2. En el menú **Archivo**, elija **Nuevos** haga clic en **Archivo**.
3. En el plantillas panel, haga clic en **archivo XML** y, a continuación, haga clic en **abrir**.
4. En el menú **Ver**, haga clic en **Propiedades** para mostrar las propiedades del archivo XML.
5. En el **propiedades** ventana, haga clic en el **examinar** botón en el **esquemas** propiedad.
6. En la lista de esquemas XSD, seleccione el esquema *vsct.xsd*. Si no está en la lista, haga clic en **Agregar** y, a continuación, busque el archivo en una unidad local. Haga clic en **Aceptar** cuando haya terminado.
7. En el archivo XML, escriba `<CommandTable>`, a continuación, presione **Tab**. Cierre la etiqueta `>` escribiendo `</CommandTable>`.

Esta acción crea un archivo .vsct básico.

8. Rellene los elementos del archivo XML que desea agregar, según la referencia de esquema XML de [VSCT](#). Para obtener más información, consulte [Crear archivos .vsct](#).

Cómo: Crear un archivo .vsct a partir de un archivo .ctc existente

Puede crear un archivo .vsct basado en XML a partir de un archivo de origen .ctc de tabla de comandos existente. Al hacerlo, puede aprovechar las ventajas del nuevo basado formato del compilador de la tabla de comandos (VSCT) Visual Studio.

Para crear un archivo .vsct a partir de un archivo .ctc existente

1. Obtenga una copia del lenguaje Perl.
2. Obtenga una copia de la *ConvertCTCToVSCT.pl* de secuencia de comandos de Perl, que normalmente se encuentra en la * <ruta* de instalación del SDK de Visual Studio> carpeta .VisualStudioIntegration.
3. Obtenga una copia del archivo de origen .ctc que desea convertir.
4. Coloque los archivos en el mismo directorio.
5. En Visual Studio la ventana del símbolo del sistema, vaya al directorio.

6. Tipo

```
perl.exe ConvertCTCtoVSCT.pl PkgCmd.ctc PkgCmd.vsct
```

donde *PkgCmd.ctc* es el nombre del archivo *.ctc* y *PkgCmd.vsct* es el nombre del archivo *.vsct* que desea crear.

Esta acción crea un nuevo archivo de origen de tabla de comandos *XML .vsct*. Puede compilar el archivo mediante *Vsct.exe*, el compilador VSCT, como lo haría con cualquier otro archivo *.vsct*.

NOTE

Puede mejorar la legibilidad del archivo *.vsct* reformateando los comentarios XML.

Cómo: Crear un archivo *.vsct* a partir de un archivo *.cto* existente

Puede crear un archivo *.vsct* basado en XML a partir de un archivo *.cto* binario existente. Si lo hace, podrá aprovechar el nuevo formato de compilador de tabla de comandos. Este proceso funciona incluso si el archivo *.cto* se compiló a partir de un archivo *.ctc*. Puede editar y compilar el archivo *.vsct* en otro archivo *.cto*.

Para crear un archivo *.vsct* a partir de un archivo *.cto*

1. Obtenga copias del archivo *.cto* y su archivo *.ctsym* correspondiente.
2. Coloque los archivos en el mismo directorio que el compilador *vsct.exe*.
3. En el símbolo del sistema de Visual Studio, vaya al directorio que contiene los archivos *.cto* y *.ctsym*.
4. Tipo

```
vsct.exe <ctofilename>.cto <vsctfilename>.vsct -S<symfilename>.ctsym
```

donde *<>* *ctofilename* es el nombre del <archivo> *.cto*, *vsctfilename* es el nombre del <archivo> *.vsct* que desea crear y *symfilename* es el nombre del archivo *.ctsym*.

Este proceso crea un nuevo archivo del compilador de tabla de comandos *XML .vsct*. Puede editar y compilar el archivo con *vsct.exe*, el compilador vsct, como lo haría con cualquier otro archivo *.vsct*.

Compilar el código

Simplemente agregar un archivo *.vsct* a un proyecto no hace que se compile. Debe incorporarlo en el proceso de compilación.

Para agregar un archivo *.vsct* a la compilación del proyecto

1. Abra el archivo de proyecto en el editor. Si se carga el proyecto, primero debe descargarlo.
2. Agregue un [elemento ItemGroup](#) que contenga un `VSCTCompile` elemento, como se muestra en el ejemplo siguiente.

```
<ItemGroup>
  <VSCTCompile Include="TopLevelMenu.vsct">
    <ResourceName>Menus.ctmenu</ResourceName>
  </VSCTCompile>
</ItemGroup>
```


El `ResourceName` elemento siempre debe `Menus.ctmenu` establecerse en .

3. Si el proyecto contiene un archivo `.resx`, agregue un `EmbeddedResource` elemento que contenga un `MergeWithCTO` elemento, como se muestra en el ejemplo siguiente:

```
<EmbeddedResource Include="VSPackage.resx">
  <MergeWithCTO>true</MergeWithCTO>
  <ManifestResourceName>VSPackage</ManifestResourceName>
</EmbeddedResource>
```

Este marcado debe `ItemGroup` ir dentro del elemento que contiene recursos incrustados.

4. Abra el archivo de paquete, normalmente denominado * `<ProjectName>Package.cs` o * `<ProjectName>Package.vb`, en el editor.
5. Agregue `ProvideMenuResource` un atributo a la clase de paquete, como se muestra en el ejemplo siguiente.

```
[ProvideMenuResource("Menus.ctmenu", 1)]
```

El primer valor del parámetro `ResourceName` debe coincidir con el valor del atributo definido en el archivo de proyecto.

Vea también

- [Crear archivos .vsct](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)
- [Referencia de esquema XML de VSCT](#)

Marcas de la línea de comandos del compilador de VSCT

06/04/2020 • 6 minutes to read • [Edit Online](#)

El compilador de la tabla de comandos de Visual Studio (VSCT) proporciona modificadores de línea de comandos para garantizar la compilación correcta de archivos .vsct.

Parámetros de línea de comandos

Para ver la ayuda Visual Studio básica de VSCT desde una ventana **de comandos**, desplácese hasta la carpeta y el tipo de la ruta de instalación del SDK de *Visual Studio*:

```
vsct /?
```

Esto devuelve:

```
Microsoft (R) Visual Studio (R) Command Table Compiler Version 3.00.2000

Syntax: vsct <infile> [<outfile>] [-S[symbols file]] [-D<preprocessor-define>]*
[-I<include-path>]* [-L<language>] [-E[C|H|N]:<name>]

-D    Specify any additional preprocessor defines
-I    Indicate what additional include paths to send to the preprocessor
-L    Specify the language to use when selecting strings
-E    Emit C# objects in the specified namespace for command items,
      followed by [L|F|H|N]:<value>
      F = Name of the file to emit (used if -EL is provided)
      L = Name of a language providing a CodeDOM provider
      N = namespace (required if -EL is provided)
      H = C++ header
-c    Clean build skipping dependency checks
-v    Verbose output
```

NOTE

Los caracteres - (dash) y / (barra diagonal) son notación aceptada para indicar parámetros de línea de comandos.

Las banderas aceptables y lo que quieren decir son las siguientes.

SWITCH	DESCRIPCIÓN
-d	Especifique los símbolos definidos adicionales.
-I	Indique las rutas de acceso de inclusión adicionales que se deben utilizar al resolver referencias de archivo.
-l	Especifique CultureInfo el nombre de la referencia cultural, por ejemplo "en-US".

SWITCH	DESCRIPCIÓN
-E	Emita objetos de C- en el espacio de nombres especificado para los elementos de comando, seguido de [C H N]: nombre <i>dearchivo</i> en el que se encuentra el encabezado de C- C, H, C++, N . El espacio de nombres es necesario para C.
-v	Salida detallada.

El modificador -L indica al compilador que seleccione un grupo de cadenas para generar [CultureInfo](#) el archivo binario .cto que corresponde al nombre de referencia cultural especificado. El nombre de referencia cultural especificado debe coincidir con el atributo Language de uno o varios [elementos Strings](#) del archivo .vsct. Si un elemento Strings no tiene ningún atributo Language, se hereda del [elemento CommandTable](#) que lo contiene.

Un archivo .vsct puede tener varios elementos Strings y cada uno puede tener un atributo Language diferente. La globalización se logra ejecutando el compilador VSCT varias veces y cambiando el modificador -L para cada nombre de referencia cultural.

Si el nombre de referencia cultural especificado por el modificador -L no coincide con el atributo Language de ningún elemento Strings, el compilador intentará que coincida con el idioma y no con la región. Por ejemplo, si no se puede encontrar "en-US", el compilador intentará "en" en su lugar. En caso contrario, intentará la referencia cultural actual del sistema operativo. En caso contrario, compilará el primer elemento Strings que encuentre.

El modificador -E se puede utilizar para emitir un archivo de encabezado de estilo C que contiene los símbolos que utiliza la tabla de comandos, o para emitir un archivo de C- que contiene objetos para los símbolos de comando.

Los modificadores -D y -I tienen la sintaxis de los indicadores de preprocesador Cl.exe C que tienen el mismo nombre. -D las definiciones que tienen el formato X-Y <se utilizan `Condition` para la expansión de las pruebas definidas> basadas en XML en atributos. -Incluir rutas de acceso <se <utilizan para <resolver las referencias de archivo Include>, Extern> y Bitmap>. Para obtener más información, vea referencia de [esquema XML de VSCT](#).

El compilador VSCT también puede descompilar un archivo binario compilado anteriormente. Para ello, proporcione un archivo <binario para el> infile. Si el archivo binario fue producido por el compilador VSCT, tendrá sus símbolos ya <incrustados y producirá la salida con los nombres simbólicos en un símbolos> sección de la salida. Si el binario fue producido por el compilador CTC, la salida contendrá los GUID y los GUID reales. Si el archivo *.ctsym generado por las versiones actuales de Ctc.exe está en la misma carpeta que el archivo de entrada binario, los símbolos se cargarán desde ese archivo y se utilizarán para la salida.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)
- [Referencia del esquema XML de VSCT](#)
- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)

Colocación predeterminada de comandos, grupos y barras de herramientas

06/04/2020 • 3 minutes to read • [Edit Online](#)

Para la uniformidad y estabilidad del producto, la interfaz de usuario muestra determinados grupos de comandos de forma predeterminada y Visual Studio proporciona definiciones para comandos y grupos de comandos. VSPackages también puede usar los comandos estándar y grupos de comandos.

Los grupos de comandos predeterminados se dividen en tres categorías: comandos IDE, comandos de producto y comandos de editor.

Comandos IDE predeterminados

La barra de herramientas IDE predeterminada incluye Visual Studio comandos compartidos por todos los productos contenidos en . Estos incluyen comandos relacionados con operaciones de proyecto genéricas, como el comando **Guardar** y el comando **Agregar elemento**. VSPackages no debe agregar o restar de esta barra de herramientas, con una excepción: si el producto o VSPackage agrega una nueva ventana de herramientas, a continuación, la ventana debe agregarse a la lista de ventanas de herramientas disponibles en el **vista** menú. Los nuevos productos o VSPackages pueden agregar su propia barra de herramientas.

Comandos de producto predeterminados

Cada producto puede proporcionar al IDE su propia barra de herramientas predeterminada que contiene comandos importantes y utilizados con frecuencia. Sin embargo, es mejor utilizar menús y barras de herramientas existentes siempre que sea posible y complementarlos con otras barras de herramientas específicas de la tarea según sea necesario.

El campo de prioridad de una barra de herramientas determina su ubicación de fila. Prioridad cero coloca la barra de herramientas en la tercera fila (fila 3), debajo de la barra de menús (fila 1) y la barra de herramientas **Estándar** (fila 2). Por lo tanto, otras barras de herramientas aparecen en la fila (prioridad + 3). Las barras de herramientas posteriores se colocan en la misma fila, si hay espacio; de lo contrario, se mueven automáticamente a la siguiente fila.

Comandos de editor predeterminados

Un VSPackage que proporciona un editor personalizado debe proporcionar una barra de herramientas predeterminada que contiene los comandos más importantes y utilizados con frecuencia en ese editor. La barra de herramientas del editor debe aparecer cuando el editor está activo y debe estar oculto cuando el editor no está activo. Esta visibilidad se controla en el `VisibilityConstraints` elemento del archivo `.vsct`.

Las barras de herramientas del editor deben colocarse debajo del IDE y las barras de herramientas del producto.

Vea también

- [Comandos, menús y grupos definidos por IDE](#)
- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Grupos, menús y comandos definidos por el IDE

06/04/2020 • 2 minutes to read • [Edit Online](#)

Muchos menús, comandos y grupos de Visual Studio comandos ya están definidos para su uso por el IDE. Estos comandos también están disponibles Visual Studio para su uso cuando se extiende .

Búsqueda de comandos definidos por el entorno

Los comandos de entorno se definen en un conjunto de cuatro archivos .vsct:

- SharedCmdDef.vsct
- SharedCmdPlace.vsct
- ShellCmdDef.vsct
- ShellCmdPlace.vsct

Estos archivos se * < *encuentran en la ruta de instalación del\SDK de Visual Studio> . Estos archivos proporcionan las definiciones y GUID de los menús y grupos que puede usar en el archivo de configuración de tabla de comandos (.vsct) de su VSPackage como contenedores para sus propios menús, grupos y comandos.

En esta sección

- [GUID e identificadores de menús de Visual Studio](#)

Proporciona los valores GUID e ID de los menús de la barra de menús de Visual Studio y de los grupos que contienen.

- [GUID e identificadores de barras de herramientas de Visual Studio](#)

Proporciona los valores GUID e ID de las barras de herramientas en el IDE de Visual Studio y de los grupos que contienen.

- [GUID e identificadores de comandos de Visual Studio](#)

Proporciona los valores GUID e ID de los comandos definidos por el IDE de Visual Studio.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)
- [Comandos definidos por el IDE para ampliar sistemas del proyecto](#)
- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)

GUID e IDs de los menús de Visual Studio

06/04/2020 • 6 minutes to read • [Edit Online](#)

En este artículo se enumeran los valores GUID e ID de los menús y grupos de la barra de menús de Visual Studio. Estos valores se definen en archivos `.vsct` que se instalan como parte del SDK de Visual Studio. Para obtener más información, consulte [Comandos, menús y grupos definidos por IDE](#).

Para obtener más información sobre cómo trabajar con objetos de entorno de desarrollo integrado (IDE) definidos en archivos `.vsct`, vea [Extender menús y comandos](#).

Los menús y grupos de la `guidSHLMainMenu` barra de menús de Visual Studio usan el GUID `IDM_VS_TOOL_MAINMENU` en sí tiene un ID de `IDM_VS_TOOL_MAINMENU`.

Grupos en la barra de menús de Visual Studio

Para agregar un menú a la barra de menús, establezca uno de estos grupos como su elemento primario.

GRUPO	ID
Archivo/Editar/Ver	IDG_VS_MM_FILEEDITVIEW
Refactorización	IDG_VS_MM_REFACTORING:
proyecto	IDG_VS_MM_PROJECT
Compilar	IDG_VS_MM_BUILDDEBUGRUN
Formato/Herramientas	IDG_VS_MM_TOOLSADDINS
Ventana/Ayuda/Comunidad	IDG_VS_MM_WINDOWHELP
Complementos	IDG_VS_MM_MACROS
FullScreenBar	IDG_VS_MM_FULLSCREENBAR

Menús en la barra de menús de Visual Studio

Para agregar un grupo a un menú existente de Visual Studio, establezca uno de los siguientes menús como su elemento primario. Los submenús no se incluyen en esta lista.

MENÚ	ID
Archivo	IDM_VS_MENU_FILE
Editar	IDM_VS_MENU_EDIT
Ver	IDM_VS_MENU_VIEW
Refactorización	IDM_VS_MENU_REFACTORING

MENÚ	ID
proyecto	IDM_VS_MENU_PROJECT
Compilar	IDM_VS_MENU_BUILD
Formato	IDM_VS_MENU_FORMAT
Herramientas	IDM_VS_MENU_TOOLS
Extensiones	IDM_VS_MENU_EXTENSIONS
Periodo	IDM_VS_MENU_WINDOW
Complementos	IDM_VS_MENU_ADDINS
Comunidad	IDM_VS_MENU_COMMUNITY
Ayuda	IDM_VS_MENU_HELP

Grupos en menús de Visual Studio

En las listas siguientes se muestran los grupos que descienden directamente de los menús de la barra de menús de Visual Studio. La forma más rápida de agregar un comando a un menú de Visual Studio es establecer uno de estos grupos como el elemento primario. Los grupos que descienden de los submenús no aparecen en esta sección.

Grupos de menú sin archivos

GRUPO	ID
Nuevo/Abierto	IDG_VS_FILE_FILE
Sumar	IDG_VS_FILE_ADD
Solución	IDG_VS_FILE_SOLUTION
Varios	IDG_VS_FILE_MISC
Save	IDG_VS_FILE_SAVE
Cambiar nombre	IDG_VS_FILE_RENAME
Browser	IDG_VS_FILE_BROWSER
Print	IDG_VS_FILE_PRINT
Utilizado más recientemente	IDG_VS_FILE_MRU
Move	IDG_VS_FILE_MOVE
Salir	IDG_VS_FILE_EXIT

Editar grupos de menús

GRUPO	ID
Deshacer/rehacer	IDG_VS_EDIT_UNDOREDO
Cortar/Copiar/Pegar	IDG_VS_EDIT_CUTCOPY
Seleccionar	IDG_VS_EDIT_SELECT
Goto	IDG_VS_EDIT_GOTO
Buscar	IDG_VS_EDIT_FIND
Objetos	IDG_VS_EDIT_OBJECTS
Verbos OLE	IDG_VS_EDIT_OLEVERBS
Comando Bien	IDG_VS_EDIT_COMMANDWELL

Refactorizar grupos de menús

GRUPO	ID
Comunes	IDG_REFACTORING_COMMON
Avanzado	IDG_REFACTORING_ADVANCED

Ver grupos de menús

GRUPO	ID
Código de formulario	IDG_VS_VIEW_FORMCODE
Browser	IDG_VS_VIEW_BROWSER
Definir vistas	IDG_VS_VIEW_DEFINEVIEWS
Windows	IDG_VS_VIEW_WINDOWS
Arquitecto de Windows	IDG_VS_VIEW_ARCH_WINDOWS
Organización Windows	IDG_VS_VIEW_ORG_WINDOWS
Navegador de código	IDG_VS_VIEW_CODEBROWSENAV_WINDOWS
Dev Windows	IDG_VS_VIEW_DEV_WINDOWS
Barras de herramientas	IDG_VS_VIEW_TOOLBARS
Symbols	IDG_VS_VIEW_SYMBOLNAVIGATE
Navegar	IDG_VS_VIEW_NAVIGATE

GRUPO	ID
Navegación pequeña	IDG_VS_VIEW_SMALLNAVIGATE
Examinador de objetos	IDG_VS_VIEW_OBJBRWSR
Comando Bien	IDG_VS_VIEW_COMMANDWELL
Páginas de propiedades	IDG_VS_VIEW_PROPPAGES
Actualizar	IDG_VS_VIEW_REFRESH

Grupos de menús del proyecto

GRUPO	ID
Añadir varios	IDG_VS_PROJ_MISCADD
Sumar	IDG_VS_PROJ_ADD
Carpeta	IDG_VS_PROJ_FOLDER
Descarga/Recarga	IDG_VS_PROJ_UNLOADRELOAD
Referencia	IDG_VS_PROJ_REFERENCE
Opciones	IDG_VS_PROJ_OPTIONS
Configuración	IDG_VS_PROJ_SETTINGS

Crear grupos de menús

GRUPO	ID
Solución	IDG_VS_BUILD_SOLUTION
Número de selección	IDG_VS_BUILD_SELECTION
Optimización guiada por perfiles	IDG_VS_PGO_SELECTION
Varios	IDG_VS_BUILD_MISC
Cancelar	IDG_VS_BUILD_CANCEL

Grupos de menú stools

GRUPO	ID
Línea de comandos	IDG_VS_TOOLS_CMDLINE
Fragmentos de código	IDG_VS_TOOLS_SNIPPETS
Subconjunto de objetos	IDG_VS_TOOLS_OBJSUBSET

GRUPO	ID
Opciones	IDG_VS_TOOLS_OPTIONS
Otros 2	IDG_VS_TOOLS_OTHER2
Herramientas externas	IDG_VS_TOOLS_EXT_TOOLS
Personalizaciones externas	IDG_VS_TOOLS_EXT_CUST

Grupos de menús de ventanas

GRUPO	ID
Nuevo	IDG_VS_WINDOW_NEW
Dock/Cerrar	IDG_VS_DOCKCLOSE
Dock/Ocultar	IDG_VS_DOCKHIDE
Organizar	IDG_VS_WINDOW_ARRANGE
Navegación	IDG_VS_WINDOW_NAVIGATION
List	IDG_VS_WINDOW_LIST

Grupos de menú de ayuda

GRUPO	ID
Ejemplos	IDG_VS_HELP_SAMPLES
Soporte técnico	IDG_VS_HELP_SUPPORT
Información	IDG_VS_HELP_ABOUT

Submenús de los menús de Visual Studio

La jerarquía siguiente muestra los submenús asociados a los menús de la barra de menús de Visual Studio. Dado que solo un grupo puede tener un menú como elemento primario, cada submenú debe descender de un grupo de un menú, en lugar de directamente desde el menú. Para obtener más información sobre la relación entre menús, grupos y submenús, consulte [Agregar un submenú a un menú](#).

NOTE

Los nombres de los menús de la barra de menús de Visual Studio no se muestran por separado en esta jerarquía porque se pueden deducir de la convención de nomenclatura para los grupos del IDE, como se indica a continuación:
IDG_VS_ <Nombre> <> del menú _ Nombre de grupo .

GRUPO PRIMARIO	SUBMENÚ	GRUPOS DE NIÑOS
IDG_VS_FILE_FILE	IDM_VS_CSCD_NEW	IDG_VS_FILE_NEW_CASCADE

GRUPO PRIMARIO	SUBMENÚ	GRUPOS DE NIÑOS
	IDM_VS_CSCD_OPEN	IDG_VS_FILE_OPENP_CASCADE
		IDG_VS_FILE_OPENF_CASCADE
IDG_VS_FILE_ADD	IDM_VS_CSCD_ADD	IDG_VS_FILE_ADD_PROJECT_NEW
		IDG_VS_FILE_ADD_PROJECT_EXI
IDG_VS_FILE_MRU	IDM_VS_CSCD_FILEMRU	IDG_VS_FILE_FMRU_CASCADE
	IDM_VS_CSCD_PROJMRU	IDG_VS_FILE_PMRU_CASCADE
IDG_VS_FILE_MOVE	IDM_VS_CSCD_MOVETOPRJ	IDG_VS_FILE_MOVE_CASCADE
		IDG_VS_FILE_MOVE_PICKER
IDG_VS_VIEW_DEV_WINDOWS	IDM_VS_CSCD_FINDRESULTS	IDG_VS_WNDO_FINDRESULTS
	IDM_VS_CSCD_WINDOWS	IDG_VS_VIEW_CALLBROWSER
		IDG_VS_WNDO_OTRWNDWS1... 6
		IDG_VS_WNDO_WINDOWS2
IDG_VS_VIEW_TOOLBARS	IDM_VS_CSCD_COMMANDBARS	
IDG_VS_EDIT_GOTO	IDM_VS_EDITOR_FIND_MENU	
IDG_VS_EDIT_OBJECTS	IDM_VS_CSCD_MNUDES	IDG_VS_MNUDES_INSERT
		IDG_VS_MNUDES_EDITNAMES
	IDM_VS_CSCD_OLEVERBS	IDG_VS_EDIT_OLEVERBS
IDG_VS_BUILD_SELECTION	IDM_VS_CSCD_BUILD	IDG_VS_BUILD_CASCADE
		IDG_VS_BUILD_PROJPICKER
	IDM_VS_CSCD_REBUILD	IDG_VS_REBUILD_CASCADE
		IDG_VS_REBUILD_PROJPICKER
	IDM_VS_CSCD_PROJONLY	IDG_VS_PROJONLY_CASCADE
	IDM_VS_CSCD_CLEAN	IDG_VS_CLEAN_CASCADE
		IDG_VS_CLEAN_PROJPICKER
	IDM_VS_CSCD_DEPLOY	IDG_VS_DEPLOY_CASCADE

GRUPO PRIMARIO	SUBMENÚ	GRUPOS DE NIÑOS
		IDG_VS_DEPLOY_PROJPICKER
IDG_VS_PGO_SELECTION	IDM_VS_CSCD_PGO_BUILD	IDG_VS_PGO_BUILD_CASCADE_BUILD
		IDG_VS_PGO_BUILD_CASCADE_RUN

Vea también

- [GUID e iDs de las barras de herramientas de Visual Studio](#)
- [GUID e iD de comandos de Visual Studio](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

GUID e IDs de las barras de herramientas de Visual Studio

06/04/2020 • 6 minutes to read • [Edit Online](#)

En este tema se enumeran los valores GUID e ID de las barras de herramientas que se incluyen en el entorno de desarrollo integrado (IDE) de Visual Studio y de los grupos que contienen. Estos valores se definen en archivos `.vsct` que se instalan como parte del SDK de Visual Studio. Para obtener más información, consulte [Comandos, menús y grupos definidos por IDE](#).

NOTE

Visual Studio no define muchas de las barras de herramientas disponibles para Visual Studio y sus valores GUID e ID no son públicos. En este tema se enumeran solo las barras de herramientas que se definen en los archivos `.vsct` del SDK de Visual Studio.

Para obtener más información sobre cómo trabajar con objetos IDE definidos en archivos `.vsct`, vea [Extender menús y comandos](#).

Las barras de herramientas predeterminadas proporcionadas `guidSHLMainMenu` por el IDE de `GUID:ID` Visual Studio usan el GUID, excepto cuando se especifique lo contrario mediante la sintaxis.

Barras de herramientas del IDE

Las siguientes barras de herramientas las proporciona el IDE de Visual Studio. Las barras de herramientas se pueden mostrar **seleccionándolas** en el submenú Barras de herramientas del menú **Herramientas**. Las barras de herramientas de las ventanas de herramientas no se incluyen en esta sección.

Solo los grupos pueden descender directamente de las barras de herramientas. Para agregar un grupo, establezca su elemento primario en el GUID y el identificador de la barra de herramientas. Para agregar un botón a una barra de herramientas, establezca su elemento primario en un grupo de la barra de herramientas.

BARRA DE HERRAMIENTAS	ID
Estándar	IDM_VS_TOOL_STANDARD
Compilar	IDM_VS_TOOL_BUILD
Editor de texto	IDM_VS_TOOL_TEXTEDITOR
Depuración	guidVSDebugGroup:IDM_DEBUG_TOOLBAR
Ubicación de depuración	guidVSDebugGroup:IDM_DEBUG_CONTEXT_TOOLBAR

Barras de herramientas especiales

Estas barras de herramientas se definen mediante el IDE de Visual Studio, pero sirven funciones especializadas y no hospedan grupos de comandos.

BARRA DE HERRAMIENTAS	ID
Add (Comando)	IDM_VS_TOOL_ADDCOMMAND
No definido	IDM_VS_TOOL_UNDEFINED
esquema XML	IDM_VS_TOOL_SCHEMA
datos XML	IDM_VS_TOOL_DATA

Grupos en las barras de herramientas IDE

Para agregar un botón a una barra de herramientas estándar, establezca uno de los siguientes grupos como su elemento primario. Los grupos se ordenan por barra de herramientas principal.

Grupos de barras de herramientas estándar

NOMBRE	ID
Guardar/Abrir	IDG_VS_TOOLS_SAVEOPEN
Cortar/Copiar	IDG_VS_TOOLS_CUTCOPY
Deshacer/rehacer	IDG_VS_TOOLS_UNDOREDO
Ejecutar/Construir	IDG_VS_TOOLS_RUNBUILD
Search	IDG_VS_TOOLS_SEARCH
Windows	IDG_VS_TOOLS_WINDOWS
Ventanas nuevas	IDG_VS_TOOLS_NEWWINDOWS
Cargar/Guardar	IDG_VS_WINDOWUI_LOADSAVE
Indicador	IDG_VS_TOOLS_GAUGE

Crear grupos de barras de herramientas

NOMBRE	ID
Construir barra	IDG_VS_BUILDBAR
Cancelar	IDG_VS_BUILD_CANCEL

Grupos de barras de herramientas del editor de texto

NOMBRE	ID
Completion	IDM_VS_TOOL_TEXTEDITOR
Sangría	IDG_VS_EDITTOOLBAR_INDENT

NOMBRE	ID
Comentario	IDG_VS_EDITTOOLBAR_COMMENT
Marcadores	IDG_VS_EDITTOOLBAR_TEMPBOOKMARKS

Depurar grupos de barras de herramientas

NOMBRE	ID
Ejecución	IDM_DEBUG_TOOLBAR
Recorrido paso a paso	IDG_DEBUG_TOOLBAR_STEPPING
Watch	IDG_DEBUG_TOOLBAR_WATCH
Windows	IDG_DEBUG_TOOLBAR_WINDOWS

Depurar grupos de barras de herramientas de ubicación

NOMBRE	ID
Ubicación de depuración	IDG_DEBUG_CONTEXT_TOOLBAR

Ventana de herramientas (barras de herramientas)

Las barras de herramientas pueden aparecer directamente en el IDE o en ventanas de herramientas como el **Explorador de soluciones**. Dado que las ventanas de herramientas no están definidas en archivos *.vsct*, las barras de herramientas de la ventana de herramientas no tienen elementos primarios definidos. En su lugar, se colocan en el código. En la tabla siguiente se muestran las barras de herramientas que aparecen en las ventanas de herramientas del IDE y los grupos de comandos que contienen.

NOTE

Las barras de herramientas `guidSHLMainMenu` y los grupos usan el GUID, excepto cuando se especifique lo contrario mediante la sintaxis GUID:ID. Cuando se especifica un GUID para una barra de herramientas, también se aplica a los grupos que descienden de esa barra de herramientas.

VENTANA DE HERRAMIENTAS	BARRA DE HERRAMIENTAS	GRUPOS
Explorador de soluciones	IDM_VS_TOOL_PROJWIN	IDG_VS_PROJ_TOOLBAR1.. 5
Explorador de servidores	guid_SE_MenuGroup:IDM_SE_TOOLBAR_SERVEREXPLORER	IDG_SE_TOOLBAR_REFRESH
Propiedades	IDM_VS_TOOL_PROPERTIES	IDG_VS_PROPERTIES_SORT IDG_VS_PROPERTIES_PAGES

VENTANA DE HERRAMIENTAS	BARRA DE HERRAMIENTAS	GRUPOS
Vista de clases	IDM_VS_TOOL_CLASSVIEW	IDG_VS_CLASSVIEW_FOLDERS IDG_VS_CLASSVIEW_SEARCH IDG_VS_CLASSVIEW_SETTINGS
Vista de clases	IDM_VS_TOOL_CLASSVIEW_GO	IDG_VS_CLASSVIEW_SEARCH2
Examinador de objetos	IDM_VS_TOOL_OBJBROWSER	IDG_VS_OBJBROWSER_SUBSETS IDG_VS_OBJBROWSER_SEARCH IDG_VS_OBJBROWSER_ADDREFERENCE IDG_VS_OBJBROWSER_BROWSERSETTINGS
Examinador de objetos	IDM_VS_TOOL_OBJECT_BROWSER_GO	IDG_VS_OBJBROWSER_SEARCH2
Salida	IDM_VS_TOOL_OUTPUTWINDOW	IDG_VS_OUTPUTWINDOW_SELECT IDG_VS_OUTPUTWINDOW_GOTO IDG_VS_OUTPUTWINDOW_NEXTPREV IDG_VS_OUTPUTWINDOW_CLEAR IDG_VS_OUTPUTWINDOW_WORDWRAP
Buscar y reemplazar	IDM_VS_TOOL_UNIFIEDFIND	IDG_VS_FINDTAB IDG_VS_REPLACETAB
Buscar resultados 1	IDM_VS_TOOL_FINDRESULTS1	IDG_VS_FINDRESULTS1_GOTO IDG_VS_FINDRESULTS1_NEXTPREV IDG_VS_FINDRESULTS1_CLEAR IDG_VS_FINDRESULTS1_STOPFIND
Resultados de búsqueda 2	IDM_VS_TOOL_FINDRESULTS2	IDG_VS_FINDRESULTS2_GOTO IDG_VS_FINDRESULTS2_NEXTPREV IDG_VS_FINDRESULTS2_CLEAR IDG_VS_FINDRESULTS2_STOPFIND
Fragmento de código	IDM_VS_TOOL_SNIPPETMENUS	IDG_VS_SNIPPET_REPL IDG_VS_SNIPPET_REF IDG_VS_SNIPPET_PROP

VENTANA DE HERRAMIENTAS	BARRA DE HERRAMIENTAS	GRUPOS
Marcadores	IDM_VS_TOOL_BOOKMARKWIND	IDG_VS_BWNEWFOLDER IDG_VS_BWNEXTBM IDG_VS_BWNEXTBMF IDG_VS_BWENABLE IDG_VS_BWDELETE
Lista de tareas	IDM_VS_TOOL_TASKLIST	IDG_VS_TASKLIST_PROVIDERLIST
Tareas de usuario	IDM_VS_TOOL_USERTASKS	IDG_VS_TASKLIST_PROVIDERLIST IDG_VS_USERTASKS_EDIT
Lista de errores	IDM_VS_TOOL_ERRORLIST	IDG_VS_ERRORLIST_ERRORGROUP IDG_VS_ERRORLIST_WARNINGGROUP IDG_VS_ERRORLIST_MESSAGEGROUP
Explorador de llamadas	IDM_VS_TOOL_CALLBROWSER1.. 16	IDG_VS_TOOLBAR_CALLBROWSER1_ACTIONS IDG_VS_TOOLBAR_CALLBROWSER1_TYPE IDG_VS_TOOLBAR_CALLBROWSER1_CBSSETTINGS
Puntos de interrupción	guidVSDebugGroup:IDM_BREAKPOINTS_WINDOW_TOOLBAR	IDG_BREAKPOINTS_WINDOW_NEW IDG_BREAKPOINTS_WINDOW_DELETE IDG_BREAKPOINTS_WINDOW_ALL IDG_BREAKPOINTS_WINDOW_VIEW IDG_BREAKPOINTS_WINDOW_EDIT IDG_BREAKPOINTS_WINDOW_COLUMNS
Desensamblado	guidVSDebugGroup:IDM_DISASM_WINDOW_TOOLBAR	IDG_DISASM_WINDOW_TOOLBAR
Memoria 1-4	guidVSDebugGroup:IDM_MEMORY_WINDOW_TOOLBAR1... 4	IDG_MEMORY_EXPRESSION1.. 4 IDG_MEMORY_COLUMNS1.. 4

VENTANA DE HERRAMIENTAS	BARRA DE HERRAMIENTAS	GRUPOS
Procesos	guidVSDebugGroup:IDM_ATTACHED_PROCS_TOOLBAR	IDG_ATTACHED_PROCS_EXECCNTRL IDG_ATTACHED_PROCS_STEPPING IDG_ATTACHED_PROCS_EXECCNTRL2 IDG_ATTACHED_PROCS_ATTACH IDG_ATTACHED_PROCS_COLUMNS

Vea también

- [Agregar un controlador de menú a una barra de herramientas](#)
- [Añadir una barra de herramientas a una ventana de herramientas](#)
- [GUID e IDs de los menús de Visual Studio](#)

GUID e iD de comandos de Visual Studio

06/04/2020 • 5 minutes to read • [Edit Online](#)

Los valores GUID e ID de los comandos incluidos en el entorno de desarrollo integrado (IDE) de Visual Studio se definen en archivos .vsct que se instalan como parte del SDK de Visual Studio. Para obtener más información, consulte [Comandos, menús y grupos definidos por IDE](#).

Para obtener más información sobre cómo trabajar con objetos IDE definidos en archivos .vsct, vea [Extender menús y comandos](#).

Buscar una definición de comando

Dado que Visual Studio define más de 1000 comandos, no es práctico enumerarlos todos aquí. En su lugar, siga estos pasos para buscar la definición de un comando.

Para localizar una definición de comando

1. En Visual Studio, abra los siguientes archivos en la ruta de instalación del *SharedCmdDef.vsct* SDK de <Visual ShellCmdDef.vsct VsDbgCmdUsed.vsct> Studio.\ * Venusmenu.vsct

La mayoría de los comandos de Visual Studio se definen en *SharedCmdDef.vsct* y *ShellCmdDef.vsct*. *VsDbgCmdUsed.vsct* define comandos que pertenecen al depurador y *Venusmenu.vsct* define comandos que son específicos del desarrollo web.

2. Si el comando es un elemento de menú, anote el texto exacto del elemento de menú. Si el comando es un botón de una barra de herramientas, tenga en cuenta el texto de información sobre herramientas que aparece al hacer una pausa en él.
3. Presione **Ctrl+F** para abrir el cuadro de diálogo **Buscar**.
4. En el cuadro **Buscar**, escriba el texto que anotó en el paso 2.
5. Compruebe que **Todos los documentos abiertos** se muestra en el cuadro **Buscar en**.
6. Haga clic en el botón **Buscar** Button.

El

Cuando haya encontrado la definición de comando, puede colocar una copia del comando en otro elemento CommandPlacement que tenga el mismo y los valores que el comando. Para obtener más información, consulte [Crear grupos de botones reutilizables](#).

Casos especiales

En los siguientes casos, es posible que el texto del menú o el texto de información sobre herramientas no coincidan exactamente con lo que hay en la definición del comando.

- Elementos de menú que incluyen un carácter subrayado, como el comando **Imprimir** del menú **Archivo**, en el que se subraya la *P*.

Los caracteres precedidos por el carácter de y comercial (&) en los nombres de elementos de menú se muestran como subrayados. Sin embargo, los archivos .vsct se escriben en XML, que utiliza el carácter ampersand (&) para indicar caracteres especiales y requiere que una y como muestra se demuestre debe deletrear como * &*. Por lo tanto, en un archivo .vsct, el comando **Imprimir** aparece como * & Imprimir*.

- Comandos que tienen texto **Save** <dinámico,> como Guardar nombre de archivo actual , y elementos de menú generados dinámicamente, como los elementos de la lista **Archivos recientes**.

No hay una forma confiable de buscar texto dinámico. En su lugar, busque un grupo que hospede el comando deseado consultando [GUID e identificadores de menús](#) o GUID de Visual Studio [e identificadores de las](#) barras de herramientas de Visual Studio y busque en el identificador de ese grupo. Si la definición de comando no tiene el grupo como su [elemento primario](#), busque *SharedCmdPlace.vsct* y *ShellCmdPlace.vsct* (o *VsDbgCmdPlace.vsct* para los comandos del depurador) para un `<CommandPlacement>` elemento que establezca el elemento primario del comando. *SharedCmdPlace.vsct*, *ShellCmdPlace.vsct* y *VsDbgCmdPlace.vsct* se encuentran en la * <ruta> \ * de instalación del SDK de Visual Studio .

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)
- [Referencia de esquema XML de VSCT](#)

Diseño de comandos

06/04/2020 • 8 minutes to read • [Edit Online](#)

Al agregar un comando a un VSPackage, debe especificar dónde va a aparecer, cuándo está disponible y cómo se debe controlar.

Definir comandos

Para definir nuevos comandos, incluya un archivo de tabla de comandos de Visual Studio (.vsct) en el proyecto de VSPackage. Si ha creado un VSPackage mediante la plantilla de paquete de Visual Studio, el proyecto incluye uno de estos archivos. Para obtener más información, vea Archivos de tabla de [comandos de Visual Studio \(.vsct\)](#).

Visual Studio combina todos los archivos .vsct que encuentra para que pueda mostrar los comandos. Dado que estos archivos son distintos del binario de VSPackage, Visual Studio no tiene que cargar el paquete para buscar los comandos. Para obtener más información, vea [Cómo VSPackages agregan elementos](#) de interfaz de usuario .

Visual Studio [ProvideMenuResourceAttribute](#) usa el atributo registration para definir comandos y recursos de menú. Para obtener más información, consulte [Implementación de comandos](#).

Los comandos se pueden cambiar en tiempo de ejecución de varias maneras diferentes. Se pueden mostrar u ocultar, habilitar o deshabilitar. Pueden mostrar texto o iconos diferentes, o contener valores diferentes. Se puede realizar una gran cantidad de personalización antes de que Visual Studio cargue el VSPackage. Para obtener más información, vea [Cómo VSPackages agregan elementos](#) de interfaz de usuario .

Controladores de comandos

Al crear un comando, debe proporcionar un controlador de eventos para ejecutar el comando. Si el usuario selecciona el comando, debe enrutarse adecuadamente. Enrutar un comando significa enviarlo al VSPackage correcto para habilitarlo o deshabilitarlo, ocultarlo o mostrarlo y ejecutarlo si el usuario decide hacerlo. Para obtener más información, consulte [Algoritmo de enrutamiento de comandos](#) .

Entorno de comandos de Visual Studio

Visual Studio puede hospedar cualquier número de VSPackages y cada uno puede contribuir con su propio conjunto de comandos. El entorno muestra solo los comandos que son adecuados para la tarea actual. Para obtener más información, consulte [Objetos](#) de contexto Disponibilidad de [comandos](#) y Selección .

Un VSPackage que define nuevos comandos, menús, barras de herramientas o menús contextuales proporciona su información de comandos a Visual Studio en tiempo de instalación a través de entradas del Registro que hacen referencia a recursos en ensamblados nativos o administrados. A continuación, cada recurso hace referencia a un archivo de recursos de datos binarios (.cto), que se genera al compilar un archivo de tabla de comandos de Visual Studio (.vsct). Esto permite a Visual Studio proporcionar conjuntos de comandos combinados, menús y barras de herramientas sin tener que cargar todos los VSPackage instalados.

Organización de comandos

El entorno posiciona los comandos por grupo, prioridad y menú.

- Los grupos son colecciones lógicas de comandos relacionados, por ejemplo, el grupo de comandos **Cortar**, **Copiar** **Pegar**. Los grupos son los comandos que aparecen en los menús.
- La prioridad determina el orden en el que aparecen los comandos individuales de un grupo en el menú.

- Los menús actúan como contenedores para grupos.

El entorno predefine algunos comandos, grupos y menús. Para obtener más información, consulte Colocación predeterminada de [comandos, grupos y barras](#) de herramientas .

Se puede asignar un comando a un grupo principal. El grupo principal controla la posición del comando en la estructura del menú principal y en el cuadro de diálogo **Personalizar**. Un comando puede aparecer en varios grupos; por ejemplo, un comando puede estar en el menú principal, en un menú contextual y en una barra de herramientas. Para obtener más información, vea [Cómo VSPackages agregan elementos](#) de interfaz de usuario .

Enrutamiento de comandos

El proceso de invocar y enrutar comandos para VSPackages difiere del proceso de llamar a métodos en instancias de objeto.

El entorno enruta los comandos secuencialmente desde el contexto de comando más interno (local), que se basa en la selección actual, al contexto más externo (global). El primer contexto que es capaz de ejecutar el comando es el que lo controla. Para obtener más información, consulte [Algoritmo de enrutamiento de comandos](#) .

En la mayoría de los casos, [IOleCommandTarget](#) el entorno controla los comandos mediante la interfaz. Dado que el esquema de enrutamiento de [IOleCommandTarget](#) comandos permite que muchos objetos diferentes controlen comandos, se pueden implementar mediante cualquier número de objetos; estos incluyen controles ActiveX de Microsoft, implementaciones de vista de ventana, objetos de documento, jerarquías de proyecto y objetos VSPackage (para comandos globales). En algunos casos especializados, por ejemplo, los [IVsHierarchy](#) comandos de ruteo en una jerarquía, la interfaz se debe implementar.

Temas relacionados

TITLE	DESCRIPCIÓN
Implementación del comando	Describe cómo implementar comandos en un VSPackage.
Disponibilidad de comandos	Describe cómo el contexto de Visual Studio determina qué comandos están disponibles.
Algoritmo de enrutamiento de comandos	Describe cómo la arquitectura de enrutamiento de comandos de Visual Studio permite que los comandos se controlen mediante diferentes VSPackages.
Directrices de colocación de comandos	Sugiere cómo colocar comandos en el entorno de Visual Studio.
Cómo VSPackages agregan elementos de interfaz de usuario	Describe cómo VSPackages puede utilizar mejor la arquitectura de comandos de Visual Studio.
Colocación predeterminada de comandos, grupos y barras de herramientas	Describe cómo VSPackages puede usar mejor los comandos que se incluyen en Visual Studio.
Administración de VSPackages	Describe cómo Visual Studio carga VSPackages.
Archivos de tabla de comandos de Visual Studio (.vsct)	Proporciona información acerca de los archivos .vsct basados en XML, que se usan para describir el diseño y la apariencia de los comandos en VSPackages.

Implementación del comando

06/04/2020 • 8 minutes to read • [Edit Online](#)

Para implementar un comando en un VSPackage, debe realizar las siguientes tareas:

1. En el archivo `.vsct`, configure un grupo de comandos y, a continuación, agréguele el comando. Para obtener más información, vea Archivos de tabla de [comandos de Visual Studio \(.vsct\)](#).
2. Registre el comando con Visual Studio.
3. Implemente el comando.

En las secciones siguientes se explica cómo registrar e implementar comandos.

Registrar comandos con Visual Studio

Si el comando va a aparecer en [ProvideMenuResourceAttribute](#) un menú, debe agregar el a su VSPackage y usar como un valor el nombre del menú o su identificador de recurso.

```
[ProvideMenuResource("Menus.ctmenu", 1)]
...
public sealed class MyPackage : Package
{.. ..}
```

Además, debe registrar el [OleMenuCommandService](#) comando con el archivo `.Puede obtener este servicio GetService mediante el método si el PackageVSPackage se deriva de .`

```
OleMenuCommandService mcs = GetService(typeof(IMenuCommandService)) as OleMenuCommandService;
if ( null != mcs )
{
    // Create the command for the menu item.
    CommandID menuCommandID = new CommandID(guidCommandGroup, myCommandID);
    MenuCommand menuItem = new MenuCommand(MenuItemCallback, menuCommandID );
    mcs.AddCommand( menuItem );
}
```

Implementar comandos

Hay varias maneras de implementar comandos. Si desea un comando de menú estático, que es un comando que siempre aparece [MenuCommand](#) de la misma manera y en el mismo menú, cree el comando utilizando como se muestra en los ejemplos de la sección anterior. Para crear un comando estático, debe proporcionar un controlador de eventos que sea responsable de ejecutar el comando. Dado que el comando siempre está habilitado y visible, no es necesario proporcionar su estado a Visual Studio. Si desea cambiar el estado de un comando en función de determinadas [OleMenuCommand](#) condiciones, puede crear el comando como una instancia `QueryStatus` de la clase y, en su constructor, proporcionar un controlador de eventos para ejecutar el comando y un controlador para notificar a Visual Studio cuando cambie el estado del comando. También puede [IOleCommandTarget](#) implementar como parte de una clase [IVsHierarchy](#) de comando o, puede implementar si está proporcionando un comando como parte de un proyecto. Las dos interfaces [OleMenuCommand](#) y la clase tienen métodos que notifican a Visual Studio de un cambio en el estado de un comando y otros métodos que proporcionan la ejecución del comando.

Cuando se agrega un comando al servicio de comandos, se convierte en uno de una cadena de comandos. Cuando implemente los métodos de notificación de estado y ejecución para el comando, tenga cuidado de proporcionar solo para ese comando en particular y pasar todos los demás casos a los demás comandos de la cadena. Si no pasa el comando (normalmente `OLECMDERR_E_NOTSUPPORTED` devolviendo), Visual Studio puede dejar de funcionar correctamente.

Métodos QueryStatus

Si está implementando `QueryStatus` el `QueryStatusCommand` método o el método, busque el GUID del conjunto de comandos al que pertenece el comando y el identificador del comando. Siga estas instrucciones:

- Si no se reconoce el GUID, la `OLECMDERR_E_UNKNOWNGROUP` implementación de cualquiera de los métodos debe devolver .
- Si la implementación de cualquiera de los métodos reconoce el GUID `OLECMDERR_E_NOTSUPPORTED` pero no ha implementado el comando, el método debe devolver .
- Si la implementación de cualquiera de los métodos reconoce el GUID y el comando, `prgCmds` el método debe `OLECMDF` establecer el campo command-flags de cada comando (en el parámetro) mediante las siguientes marcas:
 - `OLECMDF_SUPPORTED` : Se admite el comando.
 - `OLECMDF_INVISIBLE` : El comando no debe estar visible.
 - `OLECMDF_LATCHED` : El comando está activado y parece haber sido activado.
 - `OLECMDF_ENABLED` : El comando está habilitado.
 - `OLECMDF_DEFHIDEONCTXTMENU` : El comando debe estar oculto si aparece en un menú contextual.
 - `OLECMDF_NINCHED` : El comando es un controlador de menú y no está habilitado, pero su lista de menú desplegable no está vacía y todavía está disponible. (Esta bandera rara vez se utiliza.)
- Si el comando se definió en `TextChanges` el archivo `.vsct` con la marca, establezca los siguientes parámetros:
 - Establezca `rgwz` el elemento `pCmdText` del parámetro en el nuevo texto del comando.
 - Establezca `cwActual` el elemento `pCmdText` del parámetro en el tamaño de la cadena de comandos.

Además, asegúrese de que el contexto actual no es una función de automatización, a menos que el comando esté diseñado específicamente para controlar las funciones de automatización.

Para indicar que admite un `S_OK` comando determinado, devuelva . Para todos los `OLECMDERR_E_NOTSUPPORTED` demás comandos, devuelva .

En el ejemplo `QueryStatus` siguiente, el método primero se asegura de que el contexto no es una función de automatización y, a continuación, busca el GUID del conjunto de comandos y el identificador de comando correctos. El comando en sí se establece para ser habilitado y soportado. No se admiten otros comandos.


```

public int QueryStatus(ref Guid pguidCmdGroup, uint cCmds, OLECMD[] prgCmds, IntPtr pCmdText)
{
    if (!VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider))
    {
        if (pguidCmdGroup == VSConstants.VSStd2K && cCmds > 0)
        {
            // make the Right command visible
            if ((uint)prgCmds[0].cmdID == (uint)VSConstants.VSStd2KCmdID.RIGHT)
            {
                prgCmds[0].cmdf = (int)Microsoft.VisualStudio.OLE.Interop.Constants.MSOCMDF_ENABLED |
(int)Microsoft.VisualStudio.OLE.Interop.Constants.MSOCMDF_SUPPORTED;
                return VSConstants.S_OK;
            }
        }
        return Constants.OLECMDERR_E_NOTSUPPORTED;
    }
}

```

Métodos de ejecución

La implementación del método es similar a la `Exec` implementación del `QueryStatus` método. En primer lugar, asegúrese de que el contexto no es una función de automatización. A continuación, pruebe el GUID y el identificador de comando. Si no se reconoce el GUID `OLECMDERR_E_NOTSUPPORTED` o el identificador de comando, devuelva .

Para controlar el comando, `S_OK` ejecute y devuelva si la ejecución se realiza correctamente. El comando es responsable de la detección y notificación de errores; por lo tanto, devuelva un código de error si se produce un error en la ejecución. En el ejemplo siguiente se muestra cómo se debe implementar el método de ejecución.

```

public int Exec(ref Guid pguidCmdGroup, uint nCmdID, uint nCmdexopt, IntPtr pvaIn, IntPtr pvaOut)
{
    if (!VsShellUtilities.IsInAutomationFunction(m_provider.ServiceProvider))
    {
        if (pguidCmdGroup == VSConstants.GUID_VSStandardCommandSet97)
        {
            if (nCmdID == (uint) VSConstants.VSStd2KCmdID.RIGHT)
            {
                //execute the command
                return VSConstants.S_OK;
            }
        }
    }
    return Constants.OLECMDERR_E_NOTSUPPORTED;
}

```

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Disponibilidad de comandos

06/04/2020 • 3 minutes to read • [Edit Online](#)

El contexto de Visual Studio determina qué comandos están disponibles. El contexto puede cambiar en función del proyecto actual, el editor actual, los VSPackages que se cargan y otros aspectos del entorno de desarrollo integrado (IDE).

Contextos de comando

Los siguientes contextos de comando son los más comunes:

- IDE: los comandos proporcionados por el IDE siempre están disponibles.
- VSPackage: VSPackages puede definir cuándo se van a mostrar u ocultar comandos.
- Proyecto: los comandos de proyecto solo aparecen para el proyecto seleccionado actualmente.
- Editor: Solo un editor puede estar activo a la vez. Los comandos del editor activo están disponibles. Un editor trabaja en estrecha colaboración con un servicio de lenguaje. El servicio de lenguaje debe procesar sus comandos en el contexto del editor asociado.
- Tipo de archivo: un editor puede cargar más de un tipo de archivo. Los comandos disponibles pueden cambiar en función del tipo de archivo.
- Ventana activa: la última ventana de documento activo establece el contexto de la interfaz de usuario (UI) para los enlaces de clave. Sin embargo, una ventana de herramientas que tiene una tabla de enlace de claves similar al explorador web interno también podría establecer el contexto de la interfaz de usuario. Para las ventanas de documentos con varias pestañas, como el editor HTML, cada pestaña tiene un GUID de contexto de comando diferente. Después de registrar una ventana de herramientas, siempre está disponible en el menú **Ver**.
- Contexto de interfaz de usuario: los `VSConstants.UICONTEXT` contextos de `SolutionBuilding_guid` interfaz de usuario `Debugging_guid` se identifican mediante los valores de la clase, por ejemplo, cuando se compila la solución o cuando el depurador está activo. Varios contextos de interfaz de usuario pueden estar activos al mismo tiempo.

Definir GUID de contexto personalizados

Si aún no se ha definido un GUID de contexto de comando adecuado, puede definir uno en el VSPackage y, a continuación, programarlo para que esté activo o inactivo según sea necesario para controlar la visibilidad de los comandos:

1. Registrar GUID de contexto `GetCmdUIContextCookie` mediante una llamada al método.
2. Obtener el estado de un `IsCmdUIContextActive` GUID de contexto mediante una llamada al método.
3. Active y desactive los GUID de `SetCmdUIContext` contexto llamando al método.

Caution

Asegúrese de que el VSPackage no afecta a los GUID de contexto existentes porque otros VSPackages pueden depender de ellos.

Vea también

- [Objetos de contexto de selección](#)
- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Algoritmo de enrutamiento de comandos

06/04/2020 • 5 minutes to read • [Edit Online](#)

En Visual Studio los comandos se controlan mediante varios componentes diferentes. Los comandos se enrutan desde el contexto más interno, que se basa en la selección actual, al contexto más externo (también conocido como global). Para obtener más información, consulte [Disponibilidad de comandos](#).

Orden de resolución de mando

Los comandos se pasan a través de los siguientes niveles de contexto de comando:

1. Complementos: el entorno ofrece primero el comando a los complementos que están presentes.
2. Comandos de prioridad: estos [IVsRegisterPriorityCommandTarget](#) comandos se registran mediante `IVsRegisterPriorityCommandTarget`. Se llaman para cada comando en Visual Studio y se llaman en el orden en que se registraron.
3. Comandos de menú contextual: un comando ubicado en un menú contextual se ofrece primero al destino del comando que se proporciona al menú contextual, y después de eso al enrutamiento típico.
4. Destinos de comandos del conjunto de [SetupToolbar2](#) barras de herramientas: estos destinos de comando se registran cuando se llama a `IVsSetupToolbar2`. El `pCmdTarget` parámetro `null` puede ser `IVsSetupToolbar2`. Si no `IVsSetupToolbar2` es así, este destino de comando se utiliza para actualizar los comandos ubicados en la barra de herramientas que está configurando. Si el shell está configurando la barra de herramientas, pasa el marco de la ventana como `pCmdTarget` que todas las actualizaciones de los comandos de la barra de herramientas fluyan a través del marco de la ventana, incluso cuando no está enfocada.
5. Ventana de herramientas: las ventanas [IVsWindowPane](#) de herramientas, [IOleCommandTarget](#) que normalmente implementan la interfaz, también deben implementar la interfaz para que Visual Studio pueda obtener el destino del comando cuando la ventana de herramientas es la ventana activa. Sin embargo, si la ventana de herramientas que tiene el [IVsUIHierarchy](#) foco es la ventana **Proyecto**, el comando se enruta a la interfaz que es el elemento primario común de los elementos seleccionados. Si esta selección abarca varios proyectos, el [IVsSolution](#) comando se enruta a la jerarquía. La [IVsUIHierarchy](#) interfaz [QueryStatusCommand](#) [ExecCommand](#) contiene los métodos y que [IOleCommandTarget](#) son análogos a los comandos correspondientes en la interfaz.
6. Ventana de documento: si `RouteToDocs` el comando tiene el indicador establecido en su archivo `.vsct`, Visual Studio busca [IVsWindowPane](#) un destino de comando en el [IVsTextLines](#) objeto [IVsTextBuffer](#) de vista de documento, que es una instancia de una interfaz o una instancia de un objeto de documento (normalmente una interfaz o una interfaz). Si el objeto de vista de documento no admite el [IOleCommandTarget](#) comando, Visual Studio enruta el comando a la interfaz que se devuelve. (Esta es una interfaz opcional para objetos de datos de documento.)
7. Jerarquía actual: la jerarquía actual puede ser el proyecto que posee la ventana de documento activa o la jerarquía seleccionada en el **Explorador** de soluciones. Visual Studio busca [IOleCommandTarget](#) la interfaz que se implementa en la jerarquía actual o activa. La jerarquía debe admitir comandos que sean válidos siempre que la jerarquía esté activa, incluso si una ventana de documento de un elemento de proyecto tiene el foco. Sin embargo, los comandos que se aplican [IVsUIHierarchy](#) solo cuando [QueryStatusCommand](#) [ExecCommand](#) el **Explorador** de soluciones tiene el foco deben admitirse mediante la interfaz y sus métodos.

Los comandos **Cortar**, **Copiar**, **Pegar**, **Eliminar**, **Cambiar nombre**, **Introy DoubleClick** requieren un control especial. Para obtener información sobre cómo controlar los comandos [IVsHierarchyDeleteHandler](#)

Delete y **Remove** en jerarquías, consulte la interfaz.

8. Global: si los contextos mencionados anteriormente no han controlado un comando, Visual Studio intenta enrutarlo [IOleCommandTarget](#) al VSPackage que posee un comando que implementa la interfaz. Si el VSPackage no se ha cargado ya, [QueryStatus](#) no se carga cuando Visual Studio llama al método. El VSPackage se carga [Exec](#) solo cuando se llama al método.

Vea también

- [Diseño de comandos](#)

Directrices de colocación de comandos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Los procedimientos recomendados para colocar comandos en el entorno de desarrollo integrado (IDE) de Visual Studio varían en función del tamaño del conjunto de comandos. Los comandos se definen y colocan de acuerdo con la información de los archivos `.vsct`.

Prácticas recomendadas para todos los conjuntos de comandos

Para cada conjunto de comandos, siga estas pautas:

- Prepare un gráfico de la estructura de comandos de antemano. Identifique los comandos, cuadros combinados, grupos de comandos y menús contextuales que se utilizarán en más de una ubicación.
- Los comandos que aparecen en el mismo grupo deben estar relacionados.
- Los grupos que contienen un solo comando son aceptables.
- Los paquetes no deben agregar muchos comandos a los menús existentes de Visual Studio. En su lugar, deben crear menús o submenús para hospedar los nuevos comandos.
- Cuando coloque un comando en un menú existente, asigne un nombre al comando para que su propósito sea claro y no se confunda con los comandos existentes.

Prácticas recomendadas para conjuntos de comandos pequeños

Si está desarrollando un `VSPackage` que tiene solo unos pocos comandos, también siga estas directrices:

- Cuando sea posible, utilice el elemento `Parent` de un comando, cuadro combinado, grupo o menú secundario para colocarlo en el grupo adecuado.
- Asigne estos grupos a los menús mostrados por el `VSPackage`.
- El elemento primario de un menú secundario o un comando debe ser un `Group` elemento. Asigne comandos y menús secundarios a grupos y, a continuación, asigne los grupos a los menús primarios.
- Puede colocar un comando en grupos adicionales agregando una sección de elemento `CommandPlacements` después de la definición del comando y, a continuación, agregando al `CommandPlacements` elemento un elemento `CommandPlacement` para cada grupo adicional.

Prácticas recomendadas para grandes conjuntos de comandos

Si el `VSPackage` tendrá muchos comandos que aparecerán en varios contextos, también siga estas directrices:

- Haga menús, grupos y comandos de auto-paternidad. Es decir, no `Parent` asigne un elemento en la definición del elemento.
- Utilice `CommandPlacement` entradas de `CommandPlacements` elementos en la sección de elementos para colocar menús, grupos y comandos en sus menús y grupos primarios.
- En `CommandPlacements` la sección de elementos, las entradas que rellenan un menú o grupo determinado deben estar adyacentes entre sí. Esto ayuda a la `Priority` legibilidad y hace que las clasificaciones sean más fáciles de determinar.

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Optimización de comandos de menú y barra de herramientas

06/04/2020 • 2 minutes to read • [Edit Online](#)

La adición de VSPackages Visual Studio y sus comandos correspondientes a puede provocar una interfaz de usuario concurrido. Visual Studio proporciona maneras de ayudar a minimizar la confusión de comandos de la interfaz de usuario.

En esta sección

- [Puesta a disposición de comandos](#)

Proporciona directrices generales para minimizar Visual Studio el hacinamiento de la interfaz de usuario al agregar VSPackages.

- [Instrucciones sobre la selección de ubicación](#)

Proporciona instrucciones específicas para implementar un VSPackage según el tamaño del conjunto de comandos.

Secciones relacionadas

- [Comandos, menús y barras de herramientas](#)

Explica cómo crear una interfaz de usuario que incluya menús, barras de herramientas y cuadros combinados de comandos.

Puesta a disposición de los comandos

06/04/2020 • 9 minutes to read • [Edit Online](#)

Cuando se agregan varios VSPackages a Visual Studio, la interfaz de usuario (UI) puede estar superpoblada con comandos. Puede programar su paquete para ayudar a reducir este problema, de la siguiente manera:

- Programe el paquete para que se cargue solo cuando un usuario lo requiera.
- Programe el paquete para que sus comandos se muestren solo cuando puedan ser necesarios en el contexto del estado actual del entorno de desarrollo integrado (IDE).

Carga retrasada

La forma típica de habilitar la carga retrasada es diseñar el VSPackage para que sus comandos se muestran en la interfaz de usuario, pero el paquete en sí no se carga hasta que un usuario hace clic en uno de los comandos. Para ello, en el archivo .vsct, cree comandos que no tengan marcas de comando.

En el ejemplo siguiente se muestra la definición de un comando de menú de un archivo .vsct. Este es el comando generado por la plantilla de paquete de Visual Studio cuando se selecciona la opción **Comando** de menú de la plantilla.

```
<Button guid="guidTopLevelMenuCmdSet" id="cmdidTestCommand" priority="0x0100" type="Button">
  <Parent guid="guidTopLevelMenuCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <Strings>
    <CommandName>cmdidTestCommand</CommandName>
    <ButtonText>Test Command</ButtonText>
  </Strings>
</Button>
```

En el ejemplo, si `MyMenuGroup` el grupo primario, es un elemento secundario de un menú de nivel superior como el menú **Herramientas**, el comando estará visible en ese menú, pero el paquete que ejecuta el comando no se cargará hasta que un usuario haga clic en el comando. Sin embargo, al programar [IOleCommandTarget](#) el comando para implementar la interfaz, puede habilitar el paquete para que se cargue cuando se expanda por primera vez el menú que contiene el comando.

Tenga en cuenta que la carga retrasada también puede mejorar el rendimiento de inicio.

Contexto actual y visibilidad de los comandos

Puede programar comandos de VSPackage para que sean visibles u ocultos, dependiendo del estado actual de los datos de VSPackage o las acciones que son relevantes actualmente. Puede habilitar el VSPackage para establecer el estado de sus comandos, normalmente mediante una implementación del [QueryStatus](#) método desde la [IOleCommandTarget](#) interfaz, pero esto requiere que el VSPackage se cargue antes de que pueda ejecutar el código. En su lugar, se recomienda habilitar el IDE para administrar la visibilidad de los comandos sin cargar el paquete. Para ello, en el archivo .vsct, asocie comandos a uno o varios contextos de interfaz de usuario especiales. Estos contextos de interfaz de usuario se identifican mediante un GUID conocido como GUID de contexto de *comando*.

Visual Studio supervisa los cambios resultantes de las acciones del usuario, como cargar un proyecto o pasar de la edición a la creación. A medida que se producen cambios, la apariencia del IDE se modifica automáticamente. En la tabla siguiente se muestran Visual Studio cuatro contextos principales de cambio IDE que supervisa.

TIPO DE CONTEXTO	DESCRIPCIÓN
Tipo de proyecto activo	Para la mayoría <code>GUID</code> de los tipos de proyecto, este valor es el mismo que el GUID del VSPackage que implementa el proyecto. Sin Visual C++ embargo, los <code>GUID</code> proyectos usan el tipo de proyecto como valor.
Ventana activa	Normalmente, esta es la última ventana de documento activa que establece el contexto de interfaz de usuario actual para los enlaces de clave. Sin embargo, también podría ser una ventana de herramientas que tiene una tabla de enlace de claves que se asemeja al explorador web interno. Para ventanas de documentos con varias pestañas, como <code>GUID</code> el editor HTML, cada pestaña tiene un contexto de comando diferente.
Servicio de idiomas activos	El servicio de lenguaje asociado al archivo que se muestra actualmente en un editor de texto.
Ventana de herramientas activas	Una ventana de herramientas que está abierta y tiene el foco.

Una quinta área de contexto principal es el estado de la interfaz de usuario del IDE. Los contextos de interfaz `GUID` de usuario se identifican mediante contextos de comando activos, como se indica a continuación:

- [SolutionBuilding_guid](#)
- [Debugging_guid](#)
- [Dragging_guid](#)
- [FullScreenMode_guid](#)
- [DesignMode_guid](#)
- [NoSolution_guid](#)
- [SolutionExists_guid](#)
- [EmptySolution_guid](#)
- [SolutionHasSingleProject_guid](#)
- [SolutionHasMultipleProjects_guid](#)
- [CodeWindow_guid](#)

Estos GUID se marcan como activos o inactivos, dependiendo del estado actual del IDE. Varios contextos de interfaz de usuario pueden estar activos al mismo tiempo.

Ocultar y mostrar comandos basados en el contexto

Puede mostrar u ocultar un comando de paquete en el IDE sin cargar el propio paquete. Para ello, defina el comando en el archivo .vsct del paquete mediante `DefaultDisabled` los indicadores de comando , `DefaultInvisible` , y `DynamicVisibility` agregando uno o varios elementos [VisibilityItem](#) a la sección [VisibilityConstraints](#). Cuando un contexto `GUID` de comando especificado se activa, el comando se muestra sin cargar el paquete.

GUID de contexto personalizado

Si aún no se ha definido un GUID de contexto de comando adecuado, puede definir uno en el VSPackage y, a continuación, programarlo para que esté activo o inactivo según sea necesario para controlar la visibilidad de los

comandos. Utilice [SVsShellMonitorSelection](#) el servicio para:

- Registrar GUID de contexto (mediante una llamada al [GetCmdUIContextCookie](#) método).
- Obtener el estado `GUID` de un [IsCmdUIContextActive](#) contexto (mediante una llamada al método).
- Active `GUID` y desactive context s [SetCmdUIContext](#) (llamando al método).

Caution

Asegúrese de que el VSPackage no afecta al estado de cualquier GUID de contexto existente porque otros VSPackages pueden depender de ellos.

Ejemplo

En el ejemplo siguiente de un comando VSPackage se muestra la visibilidad dinámica de un comando administrado por contextos de comando sin cargar el VSPackage.

El comando se establece para habilitarse y mostrarse siempre que exista una solución; es decir, siempre que uno de los siguientes GUID de contexto de comando esté activo:

- [EmptySolution_guid](#)
- [SolutionHasMultipleProjects_guid](#)
- [SolutionHasSingleProject_guid](#)

En el ejemplo, observe que cada indicador de comando es un elemento [de indicador](#) de comando independiente.

```
<Button guid="guidDynamicVisibilityCmdSet" id="cmdidMyCommand"
    priority="0x0100" type="Button">
  <Parent guid="guidDynamicVisibilityCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <CommandFlag>DefaultDisabled</CommandFlag>
  <CommandFlag>DefaultInvisible</CommandFlag>
  <CommandFlag>DynamicVisibility</CommandFlag>
  <Strings>
    <CommandName>cmdidMyCommand</CommandName>
    <ButtonText>My Command name</ButtonText>
  </Strings>
</Button>
```

Observe también que cada contexto de `VisibilityItem` interfaz de usuario debe darse en un elemento independiente, como se indica a continuación.

```
<VisibilityConstraints>
  <VisibilityItem guid="guidDynamicVisibilityCmdSet"
    id="cmdidMyCommand" context="UICONTEXT_EmptySolution" />
  <VisibilityItem guid="guidDynamicVisibilityCmdSet"
    id="cmdidMyCommand" context="UICONTEXT_SolutionHasSingleProject" />
  <VisibilityItem guid="guidDynamicVisibilityCmdSet"
    id="cmdidMyCommand" context="UICONTEXT_SolutionHasMultipleProjects" />
</VisibilityConstraints>
```

Vea también

- [Agregue un comando a la barra de herramientas del Explorador de soluciones](#)
- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)
- [Enrutamiento de comandos en VSPackages](#)
- [Adición dinámica de elementos de menú](#)

Comandos y menús que utilizan ensamblados de interoperabilidad

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un VSPackage que implementa comandos de menú y barra de herramientas mediante el uso de ensamblados de interoperabilidad debe:

- Informar Visual Studio al entorno de desarrollo integrado (IDE) sobre los comandos que admite y si están habilitados actualmente.
- Adherirse a las reglas (contrato) para el manejo de comandos.
- Implemente explícitamente el control `IOleCommandTarget` de `IVsUIHierarchy` comandos mediante la interfaz o.

En la siguiente sección se describe cómo realizar estas tareas.

En esta sección

- [Determinar el estado del comando mediante ensamblados de interoperabilidad](#)

Describe cómo un VSPackage notifica al IDE sobre qué comandos admite y si están habilitados actualmente.

- [Contratos de comandos en ensamblados de interoperabilidad](#)

Proporciona una definición del contrato de comandos básico utilizado por todos los VSPackages que implementan comandos mediante ensamblados de interoperabilidad.

- [Implementación del comando](#)

Proporciona información general sobre cómo un VSPackage implementa un comando.

- [Registrar controladores de comandos de ensamblado de interoperabilidad](#)

Describe las entradas del Registro necesarias para notificar al IDE que un VSPackage proporciona un controlador de comandos.

Secciones relacionadas

- [Disponibilidad de comandos](#)

Describe los criterios que usa el IDE para determinar qué comandos de VSPackage están disponibles y qué objeto los controla.

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)

Proporciona detalles sobre cómo crear Visual Studio una interfaz de usuario que usa compatibilidad con comandos.

- [Enrutamiento de comandos en VSPackages](#)

Visión general del proceso utilizado para relacionar un objeto con la solicitud de comando correcta.

Determinar el estado del comando mediante ensamblados de interoperabilidad

06/04/2020 • 3 minutes to read • [Edit Online](#)

Un VSPackage debe realizar un seguimiento del estado de los comandos que puede controlar. El entorno no puede determinar cuándo un comando controlado dentro del VSPackage se habilita o deshabilita. Es responsabilidad del VSPackage informar al entorno sobre los estados de comando, por ejemplo, el estado de los comandos generales como **Cortar**, **Copiar** y **Pegar**.

Fuentes de notificación de estado

El entorno recibe información acerca de [QueryStatus](#) los comandos a través del método [IOleCommandTarget](#) de VSPackages, que forma parte de la implementación de la interfaz de VSPackage. El entorno [QueryStatus](#) llama al método del VSPackage en dos condiciones:

- Cuando un usuario abre un menú principal o un menú contextual (haciendo clic con el botón derecho), el entorno ejecuta el [QueryStatus](#) método en todos los comandos de ese menú para determinar su estado.
- Cuando el VSPackage solicita que el entorno actualizar la interfaz de usuario actual (UI). Esta actualización se produce cuando los comandos que actualmente son visibles para el usuario, como la agrupación **Cortar**, **Copiar** y **Pegar** en la barra de herramientas estándar, se habilitan y deshabilitan en respuesta a las acciones de contexto y usuario.

Puesto que el shell hospeda varios VSPackages, el rendimiento del shell se degradaría inaceptablemente si fuera necesario sondear cada VSPackage para determinar el estado del comando. En su lugar, el VSPackage debe notificar activamente el entorno cuando su interfaz de usuario cambia en el momento del cambio. Para obtener más información sobre la notificación de actualización, consulte [Actualizar la interfaz de usuario](#).

Error de notificación de estado

Error de VSPackage para notificar al entorno de un cambio de estado de comando puede colocar la interfaz de usuario en un estado incoherente. Recuerde que cualquiera de los comandos de menú o menú contextual puede ser colocado en una barra de herramientas por el usuario. Por lo tanto, no es suficiente actualizar la interfaz de usuario solo cuando se abre un menú o un menú contextual.

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Implementación](#)

Contratos de comando en ensamblados de interoperabilidad

06/04/2020 • 2 minutes to read • [Edit Online](#)

El contrato básico para [IOleCommandTarget](#) controlar comandos a [QueryStatus](#) través de la interfaz es que el entorno llama al método para determinar si se admite el comando y, si se admite, para determinar su estado y texto. A continuación, el [Exec](#) entorno llama al método para ejecutar el comando.

El [QueryStatus](#) método se controla de forma idéntica para todos los comandos. La comunicación adicional, si es necesario (por ejemplo, [Exec](#) con listas desplegables), se administra llamando al método con los parámetros adecuados. La interpretación de estos parámetros depende del comando especificado.

Si el destino del comando devuelve valores en el parámetro de salida, el autor de la llamada siempre es responsable de liberar los recursos que se han asignado. Dado que este parámetro es una variante, la compensación de la variante libera los recursos.

En los casos en que los [IVsUIHierarchy](#) comandos deben funcionar dentro de una ventana de jerarquía, se debe utilizar la interfaz. La [IVsUIHierarchy](#) interfaz tiene un contrato [QueryStatusCommand](#) [ExecCommands](#) similar con métodos similares: y .

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Enrutamiento de comandos en VSPackages](#)
- [Implementación del comando](#)

Registro de controladores de comandos de ensamblado de interoperabilidad

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un VSPackage debe registrarse en Visual Studio para que el entorno de desarrollo integrado (IDE) enrute sus comandos correctamente.

El registro se puede actualizar mediante la edición manual o mediante un archivo Registrar (.rgs). Para obtener más información, consulta [Creating Registrar Scripts](#).

Managed Package Framework (MPF) proporciona esta [ProvideMenuResourceAttribute](#) funcionalidad a través de la clase.

- Los recursos de referencia de [formato](#) de tabla de comandos se encuentran en dll de interfaz de usuario satélite no administradas.

Registro del controlador de comandos de un VSPackage

Un VSPackage que actúa como controlador para comandos basados en la interfaz de usuario (UI) requiere una entrada del Registro con el nombre del VSPackage `GUID`. Esta entrada del Registro especifica la ubicación del archivo de recursos de interfaz de usuario de VSPackage y el recurso de menú dentro de ese archivo. La propia entrada del Registro se encuentra en `HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\Visual Studio <Versión> menús`, donde * <version> * es la versión de , por ejemplo, 9.0.

NOTE

La ruta de acceso de la raíz de `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Visual Studio\VisualStudio* <Versión> *` se puede invalidar con una raíz alternativa cuando se inicializa el shell. Para obtener más información acerca de la ruta de acceso raíz, vea [Instalar VSPackages con Windows Installer](#).

La entrada del Registro de Recursos CTMENU

La estructura de la entrada del registro es:

```
HKEY_LOCAL_MACHINE\Software\VisualStudio\<Version>\
Menus\
  <GUID> = <Resource Information>
```

<GUID> `GUID` es el de vspackage en el formulario .XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX.

Resource> consta de tres elementos separados por comas. * < * Estos elementos son, en orden:

< Ruta de acceso <a> DLL de recursos, ID de recurso de menú>, < Versión de menú>

En la tabla siguiente <se describen los campos de información de recursos>.

ELEMENTO	DESCRIPCIÓN

ELEMENTO	DESCRIPCIÓN
< Ruta de acceso a DLL de recursos>	Esta es la ruta de acceso completa al archivo DLL de recursos que contiene el recurso de menú o se deja en blanco, lo que indica que se va a usar el archivo DLL de recursos de VSPackage (como se especifica en la subclave Packages donde se registra el propio VSPackage). Es habitual dejar este campo en blanco.
< ID de recurso de menú>	Este es el identificador <code>CTMENU</code> de recurso del recurso que contiene todos los elementos de interfaz de usuario para el VSPackage compilado a partir de un archivo <code>.vsct</code>.
< Versión del menú>	Este es un número utilizado <code>CTMENU</code> como versión para el recurso. Visual Studio utiliza este valor para determinar si necesita recuperar <code>CTMENU</code> el contenido del <code>CTMENU</code> recurso con su caché de todos los recursos. Una refusión se desencadena ejecutando el comando <code>devenv setup</code>. Este valor debe establecerse inicialmente en 1 e <code>CTMENU</code> incrementarse después de cada cambio en el recurso y antes de que se produzca la recombinación.

Ejemplo

A continuación se muestra un ejemplo de un par de entradas de recursos:

```
HKEY_LOCAL_MACHINE\Software\VisualStudio\9.0Exp\
Menus\
{019971D6-4685-11D2-B48A-0000F87572EB} = ,1, 10
{1b027a40-8f43-11d0-8d11-00a0c91bc942} = , 10211, 3
```

Vea también

- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)
- [Comandos y menús que usan ensamblados de interoperabilidad](#)

Uso de ensamblados de interoperabilidad de Visual Studio

06/04/2020 • 13 minutes to read • [Edit Online](#)

Los ensamblados de interoperabilidad de Visual Studio permiten que las aplicaciones administradas tienen acceso a las interfaces COM que proporcionan extensibilidad de Visual Studio. Hay algunas diferencias entre las interfaces COM rectas y sus versiones de interoperabilidad. Por ejemplo, los HRESULT se representan generalmente como valores int y deben controlarse de la misma manera que las excepciones, y los parámetros (especialmente los parámetros out) se tratan de forma diferente.

Control de valores HRESULT devueltos al código administrado desde COM

Cuando se llama a una interfaz COM desde código administrado, se examina el valor HRESULT y se genera una excepción si es necesario. La clase [ErrorHandler](#) contiene el método [ThrowOnFailure](#), que produce una excepción de COM, dependiendo del valor HRESULT que se pasa.

De forma predeterminada, [ThrowOnFailure](#) generará una excepción siempre que se pase un valor HRESULT menor que cero. En los casos en los que los valores HRESULT sean aceptables y no se generen excepciones, los valores HRESULT adicionales deben pasarse a [ThrowOnFailure](#) después de probarlos. Si el valor HRESULT que se está probando coincide con los valores HRESULT pasados explícitamente a [ThrowOnFailure](#), no se generará ninguna excepción.

NOTE

La [VSConstants](#) clase contiene constantes para HRESULTS [S_OK](#) [E_NOTIMPL](#) comunes, Visual Studio por ejemplo, [VS_E_INCOMPATIBLEDOCDATA](#) y [VS_E_UNSUPPORTEDFORMAT](#), y HRESULTS, por ejemplo, [E_FAIL](#). [VSConstants](#) también proporciona los métodos [Succeeded](#) y [Failed](#), que corresponden a las macros SUCCEEDED y FAILED en COM.

Por ejemplo, considere la siguiente llamada de función, en el que [E_NOTIMPL](#) es un valor devuelto aceptable, pero cualquier otro valor HRESULT menor que cero representa un error.

```
Dim hr As Integer = MyInterface.MyFunction(cmdID)
ErrorHandler.ThrowOnFailure(hr, VSConstants.E_NOTIMPL)
```

```
int hr = MyInterface.MyFunction(cmdID);
ErrorHandler.ThrowOnFailure(hr, VSConstants.E_NOTIMPL);
```

Si hay más de un valor devuelto aceptable, los valores HRESULT adicionales solo se pueden anexar a la lista en la llamada a [ThrowOnFailure](#).

```
Dim hr As Integer = MyInterface.MyFunction(cmdID)
ErrorHandler.ThrowOnFailure(hr, VSConstants.E_NOINTERFACE, VSConstants.E_NOTIMPL)
```

```
int hr = MyInterface.MyFunction(cmdID);
ErrorHandler.ThrowOnFailure(hr, VSConstants.E_NOINTERFACE, VSConstants.E_NOTIMPL);
```

Devolución de valores HRESULT a COM desde el código administrado

Si no se produce ninguna excepción, el código administrado devuelve `S_OK` a la función COM que lo llamó. La interoperabilidad COM admite excepciones comunes que están fuertemente tipadas en el código administrado. Por ejemplo, un método que recibe un argumento `null` inaceptable produce `ArgumentNullException`.

Si no está seguro de la excepción que se debe generar pero conoce el valor HRESULT que desea devolver a COM, puede usar el método `ThrowExceptionForHR` para generar la excepción adecuada. Esto funciona incluso con errores no estándar como, por ejemplo, `VS_E_INCOMPATIBLEDOCDATA`. `ThrowExceptionForHR` intenta asignar el valor HRESULT pasado a una excepción fuertemente tipada. Si no es posible, genera una excepción COM genérica. El resultado final es que el valor HRESULT que se pasa a `ThrowExceptionForHR` desde el código administrado se devuelve a la función COM que lo llamó.

NOTE

Las excepciones afectan al rendimiento y están destinadas para indicar condiciones del programa anormales. Las condiciones que se producen con frecuencia deben controlarse en línea, en lugar de generar una excepción.

Parámetros IUnknown pasados como Type void**

Busque parámetros [out] que `void **` se definen como tipo en `[`iid_is`]` la Visual Studio interfaz COM, pero que se definen como en el prototipo del método de ensamblado de interoperabilidad.

A veces, una `IUnknown` interfaz COM genera un objeto `void **` y, a continuación, la interfaz COM lo pasa como tipo . Estas interfaces son especialmente importantes porque si la variable se define `IUnknown` como [out] en `AddRef` el IDL, el objeto se cuenta como referencia con el método. Se produce una pérdida de memoria si el objeto no se controla correctamente.

NOTE

Un `IUnknown` objeto creado por la interfaz COM y devuelto en una variable [out] provoca una pérdida de memoria si no se libera explícitamente.

Los métodos administrados `IntPtr` que controlan `IUnknown` estos objetos `GetObjectForIUnknown` deben tratar como un puntero a un objeto y llamar al método para obtener el objeto. A continuación, el autor de la llamada debe convertir el valor devuelto al tipo que sea adecuado. Cuando el objeto ya `Release` no sea necesario, llame para liberarlo.

A continuación se muestra `QueryViewInterface` un ejemplo `IUnknown` de llamada al método y controlar el objeto correctamente:

```

MyClass myclass;
Object object;
IntPtr pObj;
Guid iid = Typeof(MyClass).Guid;
int hr = windowFrame.QueryViewInterface(ref iid, out pObj);
if (NativeMethods.Succeeded(hr))
{
    try
    {
        object = Marshal.GetObjectForIUnknown(pObj);
        myclass = object;
    }
    finally
    {
        Marshal.Release(pObj);
    }
}
else
{
    // error calling QueryViewInterface
}

```

NOTE

Se sabe que los `IUnknown` métodos siguientes [IntPtr](#) pasan punteros de objeto como tipo . Manéjalos como se describe en esta sección.

- [CreateProject](#)
- [InitializeForOwner](#)
- [GetNestedHierarchy](#)
- [CreateProject](#)
- [QueryViewInterface](#)
- [get_CfgType](#)

Parámetros [out] opcionales

Busque parámetros que se definen como `int` un `object` tipo de datos [out] (, , etc.) en la Visual Studio interfaz COM, pero que se definen como matrices del mismo tipo de datos en el prototipo del método de ensamblado de interoperabilidad.

Algunas interfaces COM, como [GetCfgs](#), tratan los parámetros [out] como opcionales. Si no se requiere un objeto, `null` estas interfaces COM devuelven un puntero como el valor de ese parámetro en lugar de crear el objeto [out]. es así por diseño. Para estas interfaces, `null` los punteros se asumen como parte del comportamiento correcto del VSPackage y no se devuelve ningún error.

Dado que CLR no permite que el valor `null` de un parámetro [out] sea , parte del comportamiento diseñado de estas interfaces no está disponible directamente en el código administrado. Los Visual Studio métodos de ensamblado de interoperabilidad para las interfaces afectadas solucionan el `null` problema definiendo los parámetros relevantes como matrices porque CLR permite el paso de matrices.

Las implementaciones administradas de `null` estos métodos deben colocar una matriz en el parámetro cuando no hay nada que devolver. De lo contrario, cree una matriz de un elemento del tipo correcto y coloque el valor devuelto en la matriz.

Los métodos administrados que reciben información de interfaces con parámetros [out] opcionales reciben el parámetro como una matriz. Simplemente examine el valor del primer elemento de la matriz. Si no `null` es así, trate el primer elemento como si fuera el parámetro original.

Pasar constantes en parámetros de puntero

Busque parámetros que se definen como punteros [in] en `IntPtr` la interfaz COM, pero que se definen como un tipo en el Visual Studio prototipo del método de ensamblado de interoperabilidad.

Un problema similar se produce cuando una interfaz COM pasa un valor especial, como 0, -1 o -2, en lugar de un puntero de objeto. A Visual C++ diferencia de , CLR no permite que las constantes se conviertan como objetos. En su Visual Studio lugar, el ensamblado `IntPtr` de interoperabilidad define el parámetro como un tipo.

Las implementaciones administradas de estos métodos `IntPtr` deben `int` aprovechar `void *` el hecho `IntPtr` de que la clase tiene ambos y constructores para crear un a partir de un objeto o una constante entera, según corresponda.

Los métodos `IntPtr` administrados que reciben parámetros de este tipo deben usar los `IntPtr` operadores de conversión de tipos para controlar los resultados. Primero convierta `IntPtr` `int` el a y pruébelo con constantes enteras relevantes. Si no coincide ningún valor, conviértalo en un objeto del tipo requerido y continúe.

Para ver ejemplos [OpenStandardEditor](#) de [OpenSpecificEditor](#) esto, vea y .

Valores de valor devuelto OLE pasados como parámetros [out]

Busque métodos que `retval` tengan un valor devuelto en `int` la interfaz COM, pero que Visual Studio tengan un valor devuelto y un parámetro de matriz [out] adicional en el prototipo del método de ensamblado de interoperabilidad. Debe quedar claro que estos métodos Visual Studio requieren un control especial porque los prototipos del método de ensamblado de interoperabilidad tienen un parámetro más que los métodos de interfaz COM.

Muchas interfaces COM que tratan con la actividad OLE envían información `retval` sobre el estado OLE al programa de llamada almacenado en el valor devuelto de la interfaz. En lugar de usar un Visual Studio valor devuelto, los métodos de ensamblado de interoperabilidad correspondientes devuelven la información al programa que realiza la llamada almacenado en un parámetro de matriz [out].

Las implementaciones administradas de estos métodos deben crear una matriz de un solo elemento del mismo tipo que el parámetro [out] y colocarla en el parámetro. El valor del elemento de matriz debe `retval` ser el mismo que el COM adecuado.

Los métodos administrados que llaman a interfaces de este tipo deben extraer el primer elemento de la matriz [out]. Este elemento se puede tratar `retval` como si fuera un valor devuelto de la interfaz COM correspondiente.

Vea también

- [Interoperar con código no administrado](#)

Enrutamiento de comandos en VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un comando se Visual Studio enruta en función del contexto en el que se ejecuta. Se enruta desde el contexto inicial hacia el contexto global.

En esta sección

- [Algoritmo de enrutamiento de comandos](#)

Describe el orden de resolución de enrutamiento de comandos.

- [Disponibilidad de comandos](#)

Describe el enrutamiento de comandos.

- [Comandos y menús que utilizan ensamblados de interoperabilidad](#)

Describe las consideraciones en los comandos de enrutamiento entre código administrado y COM.

Secciones relacionadas

- [Objetos de contexto de selección](#)

Describe el modelo de cómo puede determinar el foco de contexto de selección del usuario en una ventana.

- [Comandos, menús y barras de herramientas](#)

Explica cómo crear una interfaz de usuario que incluya menús, barras de herramientas y cuadros combinados de comandos.

VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

VSPackages son módulos Visual Studio de software que amplían el entorno de desarrollo integrado (IDE) proporcionando elementos de interfaz de usuario, servicios, proyectos, editores y diseñadores.

En esta sección

- [Especificación de la ubicación del archivo de VSPackage en el Shell de VS](#)

Explica cómo especificar la ubicación de VSPackage para el shell de Visual Studio.

- [Recursos de VSPackages](#)

Explica cómo administrar recursos en VSPackages.

- [Procedimientos recomendados para seguridad en VSPackages](#)

Le ayuda a crear productos más seguros mediante la comprensión de las vulnerabilidades de seguridad.

Secciones relacionadas

Especificación de la ubicación del archivo de VSPackage en el Shell de VS

06/04/2020 • 3 minutes to read • [Edit Online](#)

Visual Studio debe ser capaz de localizar el archivo DLL de ensamblado para cargar el VSPackage. Puede localizarlo de varias maneras, como se describe en la tabla siguiente.

MÉTODO	DESCRIPCIÓN
Utilice la clave del Registro CodeBase.	La clave CodeBase se Visual Studio puede usar para dirigir para cargar el ensamblado de VSPackage desde cualquier ruta de acceso de archivo completa. El valor de la clave debe ser la ruta de acceso del archivo al archivo DLL. Esta es la mejor Visual Studio manera de cargar el ensamblado del paquete. Esta técnica se conoce a veces como la "técnica de directorio de instalación privada/CodeBase." Durante el registro, el valor del código base se pasa RegistrationAttribute.RegistrationContext a las clases de atributo de registro a través de una instancia del tipo.
Coloque el archivo DLL en el directorio PrivateAssemblies .	Coloque el ensamblado en el subdirectorio Visual Studio PrivateAssemblies del directorio. Los ensamblados ubicados en PrivateAssemblies se detectan automáticamente, pero no están visibles en el cuadro de diálogo Agregar referencias . La diferencia entre PrivateAssemblies y PublicAssemblies es que los ensamblados de PublicAssemblies se enumeran en el cuadro de diálogo Agregar referencias . Si decide no utilizar la técnica "CodeBase/private installation directory", debe instalar en el directorio PrivateAssemblies .
Use un ensamblado con nombre seguro y la clave del Registro de ensamblados.	La clave de ensamblado se Visual Studio puede usar para dirigir explícitamente para cargar un ensamblado VSPackage con nombre seguro. El valor de la clave debe ser el nombre seguro del ensamblado.
Coloque el archivo DLL en el directorio PublicAssemblies .	Por último, el ensamblado también se puede colocar en el PublicAssemblies subdirectorio. Los ensamblados ubicados en PublicAssemblies se detectan automáticamente y Visual Studio también aparecerán en el cuadro de diálogo Agregar referencias en . Los ensamblados de VSPackage solo deben colocarse en el directorio PublicAssemblies si contienen componentes administrados destinados a ser reutilizados por otros desarrolladores de VSPackage. La mayoría de las asambleas no cumplen este criterio.

NOTE

Use ensamblados firmados con nombre seguro para todos los ensamblados dependientes. Estos ensamblados también deben instalarse en su propio directorio o en la caché global de ensamblados (GAC). Esto protege contra conflictos con ensamblados que tienen el mismo nombre de archivo base, conocido como enlace de nombre débil.

Recursos de VSPackages

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede incrustar recursos localizados en archivos DLL de interfaz de usuario satélite nativos, archivos DLL de satélite administrados o en un VSPackage administrado.

Algunos recursos no se pueden incrustar en VSPackages. Se pueden incrustar los siguientes tipos administrados:

- Cadenas
- Claves de carga de paquete (que también son cadenas)
- Iconos de la ventana de herramientas
- Archivos de salida de tabla de comandos compilados (CTO)
- Mapas de bits CTO
- Ayuda de la línea de comandos
- Acerca de los datos del cuadro de diálogo

Los recursos de un paquete administrado se seleccionan por identificador de recurso. Una excepción es el archivo CTO, que debe llamarse CTMENU. El archivo CTO debe aparecer en `byte[]` la tabla de recursos como un archivo. Todos los demás elementos de recursos se identifican por tipo.

Puede usar [PackageRegistrationAttribute](#) el atributo Visual Studio para indicar que los recursos administrados están disponibles.

```
[PackageRegistration(UseManagedResourcesOnly = true)]  
public sealed class MyPackage : Package
```

```
<PackageRegistration(UseManagedResourcesOnly:=True)>  
Public NotInheritable Class MyPackage  
    Inherits Package
```

Establecer [PackageRegistrationAttribute](#) de esta manera Visual Studio indica que debe omitir los archivos DLL satélite no administrados cuando busca recursos, por ejemplo, mediante [LoadPackageString](#). Si Visual Studio encuentra dos o más recursos que tienen el mismo identificador de recurso, usa el primer recurso que encuentra.

Ejemplo

El ejemplo siguiente es una representación administrada de un icono de ventana de herramientas.

```
<data name="1001"
type="System.Resources.ResXFileRef, System.Windows.Forms">
  <value>
    MyToolWinIcon.bmp;
    System.Drawing.Bitmap,
    System.Drawing,
    Version=1.0.0.0,
    Culture=neutral,
    PublicKeyToken=b03f5f7f11d50a3a
  </value>
</data>
```

En el ejemplo siguiente se muestra cómo incrustar la matriz de bytes CTO, que debe llamarse CTMENU.

```
<data name="CTMENU"
type="System.Resources.ResXFileRef, System.Windows.Forms">
  <value>
    MyPackage.cto;
    System.Byte[],
    mscorlib,
    Version=1.0.0.0,
    Culture=neutral,
    PublicKeyToken=b03f5f7f11d50a3a
  </value>
</data>
```

Notas de implementación

Visual Studio retrasa la carga de VSPackages siempre que sea posible. Una consecuencia de incrustar un archivo CTO Visual Studio en un VSPackage es que debe cargar todos estos VSPackages en memoria durante la instalación, que es cuando se compila una tabla de comandos combinada. Los recursos se pueden extraer de un VSPackage examinando los metadatos sin ejecutar código en el VSPackage. El VSPackage no se inicializa en este momento, por lo que la pérdida de rendimiento es mínima.

Cuando Visual Studio solicita un recurso de un VSPackage después de la instalación, es probable que ese paquete ya esté cargado e inicializado, por lo que la pérdida de rendimiento es mínima.

Vea también

- [Administración de VSPackages](#)
- [Recursos localizados en aplicaciones MFC: archivos DLL satélite](#)

Prácticas recomendadas para la seguridad en VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para instalar Visual Studio SDK el equipo, debe estar ejecutándose en un contexto con credenciales administrativas. La unidad básica de seguridad Visual Studio e implementación de una aplicación es [el VSPackage](#). Un VSPackage debe registrarse mediante Visual Studio, que también requiere credenciales administrativas.

Los administradores tienen permisos completos para escribir en el registro y el sistema de archivos, y para ejecutar cualquier código. Debe tener estos permisos para desarrollar, implementar o instalar un VSPackage.

Tan pronto como se instala, un VSPackage es de plena confianza. Debido a este alto nivel de permiso asociado a un VSPackage, es posible instalar inadvertidamente un VSPackage que tiene intención malintencionada.

Los usuarios deben asegurarse de que instalan VSPackages solo desde orígenes de confianza. Las empresas que desarrollan VSPackages deben nombrarlos y firmarlos fuertemente, para asegurar al usuario que se impide la manipulación. Las empresas que desarrollan VSPackages deben examinar sus dependencias externas, como servicios web e instalación remota, para evaluar y corregir cualquier problema de seguridad.

Para obtener más información, vea Directrices de [codificación segura para .NET Framework](#).

Vea también

- [Seguridad de complementos](#)
- [Seguridad DDEX](#)

Registro de VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

Visual Studiose basa en archivos .pkgdef para describir y localizar un VSPackage. Un archivo .pkgdef contiene toda la información de registro que de otro modo se agregaría al registro del sistema. VSPackages administrados se registran agregando atributos al código fuente y, a continuación, ejecutando la [utilidad CreatePkgDef](#) en el ensamblado resultante para generar un archivo .pkgdef.

En esta sección

- [Especificación de la ubicación del archivo de VSPackage en el Shell de VS](#)

Describe la ruta de acceso de carga para VSPackages.

- [Registrar y anulación deel registro de VSPackages](#)

Explica cómo registrar un VSPackage.

Ventanas de documento

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio, una ventana de *documento* es una ventana secundaria con marco que está asociada a una ventana de interfaz de varios documentos (MDI). Las ventanas de documento se usan normalmente para la visualización y modificación del código fuente o el texto, pero también pueden hospedar otros tipos funcionales. Ventanas de documentos:

- Se puede organizar en grupos de pestañas horizontales o verticales independientes en el MDI primario para que se puedan ver varios archivos al mismo tiempo.
- Se puede acoplar en cualquier orden en el MDI principal.
- Se puede flotar libremente.
- Se vinculan en orden de tabulación a otras ventanas MDI.

Los comandos para agrupar, acoplar y flotante se pueden encontrar en el menú contextual de una pestaña de ventana de documento.

Para obtener más información sobre el comportamiento de la ventana en Visual Studio, vea [Personalizar diseños](#) de ventana .

Implementación de la ventana de documentos

Las ventanas de documento se crean mediante la implementación de un editor. La [IVsEditorFactory](#) interfaz crea ventanas de documentos como parte de la creación de instancias de un editor. Para obtener más información, consulte [Interfaces heredadas en el editor](#).

NOTE

Para proporcionar puntos de navegación hacia [IVsBackForwardNavigation](#) atrás y hacia delante en una ventana, implemente la interfaz. El editor de texto utiliza marcadores de texto para identificar los puntos de navegación en el documento.

La tabla Documento en ejecución

El IDE utiliza la tabla de documentos en ejecución (RDT) para realizar un seguimiento del estado de cada ventana de documento. El RDT es el mecanismo a través del cual se notifican eventos a las ventanas de documentos, como cuando se cierra una solución o cuando se ha editado un archivo. Para obtener más información, consulte Ejecución de la tabla de [documentos](#).

Vea también

- [Carga de documentos retrasada](#)

Tabla de documentos en ejecución

06/04/2020 • 13 minutes to read • [Edit Online](#)

El IDE mantiene la lista de todos los documentos abiertos actualmente en una estructura interna denominada tabla de documentos en ejecución (RDT). Esta lista incluye todos los documentos abiertos en la memoria, independientemente de si estos documentos se están editando actualmente. Un documento es cualquier elemento que se conserva, incluidos los archivos de un proyecto o el archivo de proyecto principal (por ejemplo, un archivo .vcxproj).

Elementos de la tabla de documentos en ejecución

La tabla de documentos en ejecución contiene las siguientes entradas.

ELEMENTO	DESCRIPCIÓN
Apodo de documento	Cadena que identifica de forma única el objeto de datos del documento. Esta sería la ruta de acceso de archivo absoluta para un sistema de proyectos que administra archivos (por ejemplo, C:-MiProyecto-MiArchivo). Esta cadena también se utiliza para proyectos guardados en almacenes distintos de los sistemas de archivos, como los procedimientos almacenados en una base de datos. En este caso, el sistema de proyecto puede inventar una cadena única que puede reconocer y posiblemente analizar para determinar cómo almacenar el documento.
Propietario de la jerarquía	El objeto de jerarquía que posee el IVsHierarchy documento, representado por una interfaz.
ID de artículo	Identificador de elemento para un elemento específico dentro de la jerarquía. Este valor es único entre todos los documentos de la jerarquía que posee este documento, pero no se garantiza que este valor sea único en distintas jerarquías.
Objeto de datos de documento	Como mínimo, se trata de un <code>IUnknown</code> objeto. El IDE no requiere ninguna <code>IUnknown</code> interfaz en particular más allá de la interfaz para el objeto de datos de documento de un editor personalizado. Sin embargo, para un editor estándar, IVsPersistDocData2 la implementación de la interfaz del editor es necesaria para controlar las llamadas de persistencia de archivos desde el proyecto. Para obtener más información, consulte Guardar un documento estándar .
Marcas	Los indicadores que controlan si se guarda el documento, si se aplica un bloqueo de lectura o edición, etc., se pueden especificar cuando se agregan entradas al RDT. Para obtener más información, vea la enumeración <code>_VSRDTFLAGS</code> .

ELEMENTO	DESCRIPCIÓN
Editar recuento de bloqueos	Recuento de bloqueos de edición. Un bloqueo de edición indica que algún editor tiene el documento abierto para su edición. Cuando el recuento de bloqueos de edición pasa a cero, se solicita al usuario que guarde el documento, si se ha modificado. Por ejemplo, cada vez que se abre un documento en un editor mediante el comando Nueva ventana , se agrega un bloqueo de edición para ese documento en el RDT. Para establecer un bloqueo de edición, el documento debe tener una jerarquía o un identificador de elemento.
Leer recuento de bloqueos	Recuento de cerraduras de lectura. Un bloqueo de lectura indica que el documento se está leyendo a través de algún mecanismo, como un asistente. Un bloqueo de lectura mantiene un documento activo en el RDT mientras indica que el documento no se puede editar. Puede establecer un bloqueo de lectura incluso si el documento no tiene una jerarquía o un identificador de elemento. Esta característica le permite abrir un documento en la memoria e introducirlo en el RDT sin que el documento sea propiedad de ninguna jerarquía. Esta característica rara vez se utiliza.
Soporte de bloqueo	Instancia de IVsDocumentLockHolder una interfaz. El soporte de bloqueo se implementa mediante características como asistentes que abren y editan documentos fuera de un editor. Un soporte de bloqueo permite que la función agregue un bloqueo de edición al documento para evitar que el documento se cierre mientras se está editando. Normalmente, los bloqueos de edición solo se agregan mediante ventanas de documento (es decir, editores).

Cada entrada en el RDT tiene una jerarquía única o identificador de elemento asociado, que generalmente corresponde a un nodo del proyecto. Todos los documentos disponibles para su edición suelen ser propiedad de una jerarquía. Las entradas realizadas en el control RDT que proyecto, o, más exactamente, qué jerarquía, posee actualmente el objeto de datos de documento que se está editando. Con la información del RDT, el IDE puede impedir que más de un proyecto abra un documento a la vez.

La jerarquía también controla la persistencia de los datos y utiliza la información del RDT para actualizar los cuadros de diálogo **Guardar** y **Guardar como**. Cuando los usuarios modifican un documento y, a continuación, eligen el comando **Salir** del menú **Archivo**, el IDE les solicita con el cuadro de diálogo **Guardar cambios** que les muestre todos los proyectos y elementos de proyecto que se modifican actualmente. Esto permite a los usuarios elegir cuál de los documentos guardar. La lista de documentos que se va a guardar (es decir, los documentos que tienen cambios) se genera desde el RDT. Todos los elementos que espera ver en el cuadro de diálogo **Guardar cambios** al salir de la aplicación deben tener registros en el RDT. El RDT coordina qué documentos se guardan y si se solicita al usuario acerca de una operación de salvar utilizando los valores especificados en la entrada Indicadores para cada documento. Para obtener más información sobre los [_VSRDTFLAGS](#) indicadores RDT, vea la enumeración.

Editar bloqueos y leer bloqueos

Editar bloqueos y bloqueos de lectura residen en el RDT. La ventana del documento incrementa y disminuye el bloqueo de edición. Por lo tanto, cuando un usuario abre una nueva ventana de documento, el recuento de bloqueos de edición aumenta en uno. Cuando el número de bloqueos de edición llega a cero, se indica a la jerarquía que persista o guarde los datos del documento asociado. A continuación, la jerarquía puede conservar los datos de cualquier manera, incluida la persistencia como un archivo o como un elemento en un repositorio. Puede usar [LockDocument](#) el método [IVsRunningDocumentTable](#) en la interfaz para agregar [UnlockDocument](#)

bloqueos de edición y bloqueos de lectura, y el método para quitar estos bloqueos.

Normalmente, cuando se crea una instancia de la ventana de documento de un editor, el marco de ventana agrega automáticamente un bloqueo de edición para el documento en el RDT. Sin embargo, si crea una vista personalizada de un documento que no utiliza [IVsWindowFrame](#) una ventana de documento estándar (es decir, no implementa la interfaz), debe establecer su propio bloqueo de edición. Por ejemplo, en un asistente, un documento se edita sin abrirse en un editor. Para que los bloqueos de documentos se abran los asistentes y entidades similares, estas entidades deben implementar la [IVsDocumentLockHolder](#) interfaz. Para registrar el titular del [RegisterDocumentLockHolder](#) bloqueo de documentos, llame al método y pase la implementación. [IVsDocumentLockHolder](#) Al hacerlo, se añade el soporte de bloqueo de documentos al RDT. Otro escenario para implementar un soporte de bloqueo de documento es si abre un documento a través de una ventana de herramientas especial. En este caso, no puede hacer que la ventana de herramientas cierre el documento. Sin embargo, al registrarse como titular de bloqueo de documentos [CloseDocumentHolder](#) en el RDT, el IDE puede llamar a la implementación del método para solicitar un cierre del documento.

Otros usos de la tabla de documentos en ejecución

Otras entidades del IDE usan el RDT para obtener información sobre los documentos. Por ejemplo, el administrador de control de código fuente utiliza el RDT para indicar al sistema que vuelva a cargar un documento en el editor, después de obtener la versión más reciente del archivo. Para ello, el administrador de control de código fuente busca los archivos en el RDT para ver si alguno de ellos está abierto. Si lo son, el administrador de control de código fuente [ReloadItem](#) comprueba primero para ver que la jerarquía implementa el método. Si el proyecto no [ReloadItem](#) implementa el método, el administrador de [ReloadDocData](#) control de código fuente comprueba si hay una implementación del método en el objeto de datos de documento directamente.

El IDE también usa el RDT para resurgir (llevar al frente) un documento abierto, si un usuario solicita ese documento. Para obtener más información, consulte [Visualización de archivos mediante el comando Abrir archivo](#). Para determinar si un archivo está abierto en el RDT, haga una de las siguientes acciones.

- Consulte el moniker del documento (es decir, la ruta de acceso completa del documento) para averiguar si el elemento está abierto.
- Utilice la jerarquía o el identificador de elemento para solicitar al sistema de proyectos la ruta de acceso completa del documento y, a continuación, busque el elemento en el RDT.

Vea también

- [Uso de RDT_ReadLock](#)
- [Persistencia y tabla de documentos en ejecución](#)

Uso de RDT_ReadLock

06/04/2020 • 5 minutes to read • [Edit Online](#)

[_VSRDTFLAGS.RDT_ReadLock](#) es una marca que proporciona lógica para bloquear un documento en la tabla de documentos en ejecución (RDT), que es la lista de todos los documentos que están abiertos actualmente en el IDE de Visual Studio. Esta marca determina cuándo se abren los documentos y si un documento es visible en la interfaz de usuario o se mantiene invisiblemente en la memoria.

Por lo general, [usas _VSRDTFLAGS.RDT_ReadLock](#) cuando se cumple una de las siguientes condiciones:

- Desea abrir un documento de forma invisible y de solo lectura, [IVsHierarchy](#) pero aún no se ha establecido el que debe poseerse.
- Desea que se pida al usuario que guarde un documento que se abrió de forma invisible antes de que el usuario lo mostrara en la interfaz de usuario y, a continuación, intentara cerrarlo.

Cómo administrar documentos visibles e invisibles

Cuando un usuario abre un documento [IVsHierarchy](#) en la interfaz de usuario, se debe establecer un propietario para el documento y un [_VSRDTFLAGS.RDT_EditLock](#) marca debe establecerse. Si [IVsHierarchy](#) no se puede establecer ningún propietario, el documento no se guardará cuando el usuario haga clic en **Guardar todo** o cierre el IDE. Esto significa que si un documento está abierto de forma invisible donde se modifica en la memoria y se pide al `RDT_ReadLock` usuario que guarde el documento al apagarlo o guardarlo si se elige **Guardar todo**, no se puede utilizar un archivo. En su lugar, `RDT_EditLock` debe [IVsDocumentLockHolder](#) usar un y registrar un [_VSREGDOCLOCKHOLDER.RDLH_WeakLockHolder](#) bandera.

modificación de RDT_EditLock y documentos

La marca anterior mencionada indica que la apertura `RDT_EditLock` invisible del documento dará su cuando el documento sea abierto por el usuario en un **DocumentWindow** visible. Cuando esto ocurre, se presenta al usuario un símbolo del sistema **Save** cuando se cierra la **DocumentWindow** visible.

`Microsoft.VisualStudio.Package.Automation.OAProject.CodeModel` las implementaciones [IVsInvisibleEditorManager](#) que utilizan el servicio `RDT_ReadLock` funcionan inicialmente cuando solo se toma un (es decir, cuando el documento se abre de forma invisible para analizar información). Más adelante, si el documento se debe modificar, el bloqueo se actualiza a un **RDT_EditLock** débil. Si el usuario abre el documento en `CodeModel` una **DocumentWindow** visible, `RDT_EditLock` se libera el débil 's.

Si el usuario cierra el **DocumentWindow** y elige **No** cuando se le `CodeModel` pide que guarde el documento abierto, la implementación elimina toda la información en el documento y vuelve a abrir el documento desde el disco de forma invisible la próxima vez que se requiere más información para el documento. La sutileza de este comportamiento es una instancia donde el usuario abre el **DocumentWindow** del documento abierto invisible, lo modifica, lo cierra y, a continuación, elige **No** cuando se le pide que guarde el documento. En este caso, si `RDT_ReadLock` el documento tiene un, entonces el documento no se cerrará realmente y el documento modificado permanecerá abierto invisiblemente en la memoria, aunque el usuario haya elegido no guardar el documento.

Si la apertura invisible del `RDT_EditLock` documento utiliza un débil, entonces produce su bloqueo cuando el usuario abre el documento visiblemente y no se mantienen otros bloqueos. Cuando el usuario cierra **DocumentWindow** y elige **No** cuando se le pide que guarde el documento, el documento debe cerrarse de memoria. Esto significa que el cliente invisible debe escuchar los eventos RDT para realizar un seguimiento de esta ocurrencia. La próxima vez que se requiera el documento, el documento debe volver a abrirse.

Persistencia y tabla de documentos en ejecución

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio el IDE, los proyectos son completamente responsables de administrar la persistencia de sus elementos de proyecto, que realizan mediante el servicio, [SVsRunningDocumentTable](#). Los documentos son la unidad básica de persistencia en el entorno de Visual Studio. Los proyectos coordinan la apertura, el guardado y el cambio de nombre de los documentos con la tabla de documentos en ejecución (RDT), un recurso que realiza un seguimiento del estado de todos los documentos abiertos.

Gestión de la persistencia

Los proyectos controlan el servicio de [IVsPersistHierarchyItem](#) persistencia del entorno mediante la implementación de la interfaz. Aunque el entorno nunca pide directamente que un documento persista, pide al proyecto propietario (o jerarquía) que guarde el documento. Esto permite que el proyecto guarde los datos de sus elementos de proyecto en archivos locales, archivos remotos, una base de datos, un repositorio u otro medio.

El entorno global mantiene el RDT. El entorno mantiene entradas para todas las ventanas y documentos abiertos en el RDT, lo que les permite recibir notificaciones especiales, como cuando se cierra una solución. Además, el RDT permite que el entorno realice un seguimiento de sus nodos correspondientes en el **Explorador** de soluciones. El RDT mantiene un registro por objeto abierto y persistente, incluidos los archivos de proyecto y los documentos de elemento de proyecto.

Vea también

- [Tabla de documentos en ejecución](#)
- [Selección y moneda en el IDE](#)

Carga de documentos retrasada

06/04/2020 • 5 minutes to read • [Edit Online](#)

Cuando un usuario vuelve a abrir una solución de Visual Studio, la mayoría de los documentos asociados no se cargan inmediatamente. El marco de la ventana del documento se crea en un estado de inicialización pendiente y un documento de marcador de posición (denominado marco de código auxiliar) se coloca en la tabla Documento en ejecución (RDT).

La extensión puede hacer que los documentos del proyecto se carguen innecesariamente consultando elementos de los documentos antes de que se carguen, lo que puede aumentar el espacio de memoria general de Visual Studio.

Carga de documentos

El marco de código auxiliar y el documento se inicializan completamente cuando el usuario accede al documento, por ejemplo, seleccionando la pestaña del marco de ventana. El documento también se puede inicializar mediante una extensión que solicita los datos del documento, ya sea accediendo al RDT directamente para adquirir los datos del documento o accediendo al RDT indirectamente realizando una de las siguientes llamadas:

- El método [Show](#) de marco de ventana.
- El método [GetProperty](#) de marco de ventana en cualquiera de las siguientes propiedades:
 - [__VSFPROPID.VSFPROPID_DocView](#)
 - [__VSFPROPID.VSFPROPID_ViewHelper](#)
 - [__VSFPROPID.VSFPROPID_DocData](#)
 - [__VSFPROPID.VSFPROPID_AltDocData](#)
 - [__VSFPROPID.VSFPROPID_RDTDocData](#)
 - [__VSFPROPID.VSFPROPID_SPProjContext](#)
- Si la extensión utiliza código administrado, no debe llamar [GetDocumentInfo](#) a menos que esté seguro de que el documento no está en el estado de inicialización pendiente o desea que el documento se inicialice completamente. La razón es porque el método siempre devuelve el objeto de datos doc, creándolo si es necesario. En su lugar, debe llamar `IVsRunningDocumentTable4` a uno de los métodos en la interfaz.
- Si la extensión utiliza C++, puede pasar `null` los parámetros que no desea.
- Puede evitar la carga innecesaria de documentos llamando a uno de los métodos siguientes antes de solicitar las propiedades relevantes antes de solicitar otras propiedades:
 - [GetProperty](#) usando [__VSFPROPID6.VSFPROPID_PendingInitialization](#).
 - [GetDocumentFlags](#). Este método [_VSRDTFLAGS4](#) devuelve un objeto que incluye un valor para [_VSRDTFLAGS4.RDT_PendingInitialization](#) si el documento aún no se ha inicializado.

Puede averiguar cuándo se ha cargado un documento suscribiéndose al evento RDT que se genera cuando un documento se inicializa por completo. Hay dos posibilidades:

- Si el receptor [IVsRunningDocTableEvents2](#) de eventos se [OnAfterAttributeChangeEx](#)implementa, puede suscribirse a ,

- De lo contrario, [OnAfterAttributeChange](#) puede suscribirse a .

El ejemplo siguiente es un escenario de acceso a documentos hipotético: una extensión de Visual Studio desea mostrar información sobre documentos abiertos, por ejemplo, el recuento de bloqueos de edición y algo sobre los datos del documento. Enumera los documentos en el [IEnumRunningDocuments](#) RDT [GetDocumentInfo](#) mediante , a continuación, llama a cada documento con el fin de recuperar el recuento de bloqueo de edición y los datos del documento. Si el documento está en estado de inicialización pendiente, solicitar los datos del documento hace que se inicialice innecesariamente.

Una forma más eficaz de obtener [GetDocumentEditLockCount](#) acceso a un documento es [GetDocumentFlags](#) usar para obtener el recuento de bloqueos de edición y, a continuación, usar para determinar si el documento se ha inicializado. Si las marcas no incluyen [_VSRDTFLAGS4. RDT_PendingInitialization](#), el documento ya se ha inicializado [GetDocumentData](#) y solicitar los datos del documento con no provoca ninguna inicialización innecesaria. Si las marcas incluyen [_VSRDTFLAGS4. RDT_PendingInitialization](#), la extensión debe evitar solicitar los datos del documento hasta que se inicialice el documento. Esta inicialización se [OnAfterAttributeChange\(Ex\)](#) puede detectar en el controlador de eventos.

Probar extensiones para ver si fuerzan la inicialización

No hay ninguna indicación visible para indicar si se ha inicializado un documento, por lo que puede ser difícil averiguar si la extensión está forzando la inicialización. Puede establecer una clave del Registro que facilite la verificación, ya que hace que el título de cada documento que no está completamente inicializado tenga el texto *[Stub]* en el título.

En `HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 14.0, BackgroundSolutionLoad`, establezca `StubTabTitleFormatString` en `* {0} [Stub]*`.

Extensibilidad de servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un servicio de lenguaje proporciona compatibilidad específica del idioma para editar código fuente en el IDE.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, vea [Extensiones de servicio de editor y lenguaje](#).

En esta sección se describe la estructura y la implementación de un servicio de lenguaje heredado.

En esta sección

- [Migración de un servicio de lenguaje heredado](#)

Explica cómo actualizar un servicio de lenguaje de Visual Studio 2008 a la versión más reciente.

- [Fundamentos de servicio de lenguaje heredado](#)

Proporciona información importante sobre cómo desarrollar servicios de lenguaje para integrar un lenguaje de programación en Visual Studio.

- [Desarrollo de un servicio de lenguaje heredado](#)

Proporciona vínculos a temas que pueden ayudarle a crear un servicio de lenguaje.

- [Colores de la sintaxis en un servicio de lenguaje heredado](#)

Proporciona información sobre cómo admitir el resaltado de sintaxis en un servicio de lenguaje.

- [Implementación de un servicio de lenguaje heredado](#)

Proporciona información sobre cómo usar el marco de paquete administrado (MPF) para implementar un servicio de lenguaje con todas las funciones en código administrado.

- [Compatibilidad con herramientas de exploración de símbolos](#)

Describe bibliotecas y herramientas que le permiten examinar vistas de árbol de símbolos en el IDE.

Secciones relacionadas

- [Editor y extensiones de servicio de lenguaje](#)

Proporciona información general de los editores de Visual Studio.

- [Compatibilidad del servicio de lenguaje con la depuración](#)

Proporciona información y un vínculo al SDK de depuración de Visual Studio, que contiene la información necesaria para crear y personalizar los componentes del depurador que se usan para depurar programas.

Migración de un servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

Puede migrar un servicio de lenguaje heredado a una versión posterior de Visual Studio actualizando el proyecto y agregando un archivo `source.extension.vsixmanifest` al proyecto. El propio servicio de lenguaje seguirá funcionando como antes, porque el editor de Visual Studio lo adapta.

Los servicios de lenguaje heredados se implementan como parte de un `VSPackage`, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, vea [Extensiones de servicio de editor y lenguaje](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Migración de una solución de servicio de lenguaje de Visual Studio 2008 a una versión posterior

En los pasos siguientes se muestra cómo adaptar un ejemplo de Visual Studio 2008 denominado `RegExLanguageService`. Puede encontrar este ejemplo en una instalación del SDK de Visual Studio 2008, en la ruta de instalación del SDK de *Visual Studio*.

IMPORTANT

Si el servicio de lenguaje no define `RequestStockColors` colores, debe establecer explícitamente `true` en el `VSPackage`:

```
[Microsoft.VisualStudio.Shell.ProvideLanguageService(typeof(YourLanguageService), YourLanguageServiceName, 0, RequestStockColors = true)]
```

Para migrar un servicio de lenguaje de Visual Studio 2008 a una versión posterior

1. Instale las versiones más recientes de Visual Studio y el SDK de Visual Studio. Para obtener más información acerca de las formas de instalar el SDK, vea [Instalar el SDK de Visual Studio](#).
2. Edite el archivo `RegExLangServ.csproj` (sin cargarlo en Visual Studio).

En `Import` el nodo que hace referencia al archivo `Microsoft.VsSDK.targets`, reemplace el valor por el texto siguiente.

```
$(MSBuildExtensionsPath)\Microsoft\VisualStudio\v14.0\VSSDK\Microsoft.VsSDK.targets
```

3. Guarde el archivo y, a continuación, ciérrelo.
4. Abra la solución `RegExLangServ.sln`.
5. Aparece la ventana **Actualización unidireccional**. Haga clic en **OK**.
6. Actualice las propiedades del proyecto. Abra la ventana **Propiedades del proyecto** seleccionando el nodo del proyecto en el **Explorador de soluciones**, haciendo clic con el botón derecho y seleccionando

Propiedades.

- En la pestaña **Aplicación**, cambie Marco de destino a **4.6.1**.
- En la pestaña **Depurar**, en el cuadro **Iniciar programa externo**, escriba **** <Ruta*>de instalación de Visual Studio**.

En el cuadro **Argumentos** de línea de comandos, escriba **/rootsuffix Exp**.

7. Actualice las siguientes referencias:

- Quite la referencia a **Microsoft.VisualStudio.Shell.9.0.dll** y, a continuación, agregue referencias a **Microsoft.VisualStudio.Shell.14.0.dll** y **Microsoft.VisualStudio.Shell.Immutable.11.0.dll**.
- Quite la referencia a **Microsoft.VisualStudio.Package.LanguageService.9.0.dll** y, a continuación, agregue una referencia a **Microsoft.VisualStudio.Package.LanguageService.14.0.dll**.
- Agregue una referencia a **Microsoft.VisualStudio.Shell.Interop.10.0.dll**.

8. Abra el archivo **VsPkg.cs** y **DefaultRegistryRoot** cambie el valor del atributo a

```
"Software\\Microsoft\\VisualStudio\\14.0Exp"
```

9. El ejemplo original no registra su servicio de lenguaje, por lo que debe agregar el siguiente atributo a **VsPkg.cs**.

```
[ProvideLanguageService(typeof(RegularExpressionLanguageService), "RegularExpressionLanguage", 0, RequestStockColors=true)]
```

10. Debe agregar un archivo **source.extension.vsixmanifest**.

- Copie este archivo de una extensión existente en el directorio del proyecto. (Una forma de obtener este archivo es crear un proyecto VSIX (en **Archivo**, haga clic en **Nuevo**, a continuación, haga clic en **Proyecto**. En Visual Basic o en C, haga clic en **Extensibilidad**, a continuación, seleccione **Proyecto VSIX**.)
- Agregue el archivo al proyecto.
- En **Propiedades del archivo**, establezca **Acción de compilación** en **Ninguno**.
- Abra el archivo con el Editor de **manifiestos VSIX**.
- Cambie los siguientes campos:
 - **ID**: **RegExLangServ**
 - **Nombre del producto**: **RegExLangServ**
 - **Descripción**: **Un servicio de lenguaje de expresiones regulares.**
- En **Assets**, haga clic en **New**, seleccione **Type** to **Microsoft.VisualStudio.VsPackage**, establezca **Source** en **A project in the solution** **actualy**, a continuación, establezca **Project** en **RegExLangServ**.
- Guarde y cierre el archivo.

11. Compile la solución. Los archivos compilados se implementan en **%USERPROFILE%-AppData-Local-Microsoft-VisualStudio-14.0Exp-Extensions-MSIT-RegExLangServ**.

12. Inicie la depuración. Se ha abierto una segunda instancia de Visual Studio.

Vea también

- [Extensibilidad de servicio de lenguaje heredado](#)

Fundamentos de servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

Debe proporcionar un servicio de lenguaje para integrar un lenguaje de programación en Visual Studio. En este tema se explican las características disponibles en los servicios de lenguaje heredado.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, vea [Extensiones de servicio de editor y lenguaje](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Los servicios de lenguaje heredados proporcionan las siguientes características:

CARACTERÍSTICA	DESCRIPCIÓN
Seleccionar el color de la sintaxis	Hace que la vista del editor muestre diferentes colores y estilos de fuente para los diferentes elementos de un idioma. Esta diferenciación puede facilitar la lectura y edición de archivos. Para obtener información general, consulte Coloración de sintaxis en un servicio de lenguaje heredado. Para obtener información sobre esta característica en el marco de trabajo de paquete administrado (MPF), vea Colorear sintaxis en un servicio de lenguaje heredado.
Instrucciones completadas	Completa una instrucción o palabra clave que el usuario ha comenzado a escribir. La finalización de instrucciones ayuda a los usuarios a introducir instrucciones difíciles más fácilmente, con menos escritura y menos posibilidades de error. Para obtener información general, vea Finalización de instrucciones en un servicio de lenguaje heredado. Para obtener información acerca de esta característica en el MPF, vea Finalización de Word en un servicio de lenguaje heredado.
Coincidencia de llaves	Resalta los caracteres emparejados, como llaves. Cuando el usuario escribe un carácter de cierre, como por ejemplo, "-", la coincidencia de llaves resalta el carácter de apertura correspondiente, como, por ejemplo, ". Cuando hay varios niveles de caracteres envolventes, esta característica ayuda a los usuarios a confirmar que los caracteres envolventes están emparejados correctamente. Para obtener información sobre esta característica en el MPF, vea Coincidencia de llaves en un servicio de lenguaje heredado.

CARACTERÍSTICA	DESCRIPCIÓN
Información sobre parámetros sobre herramientas	Muestra una lista de posibles firmas para el método sobrecargado que el usuario está escribiendo actualmente. Para obtener información general, consulte Información de parámetros en un servicio de lenguaje heredado . Para obtener información sobre esta característica en el MPF, consulte Información de parámetros en un servicio de lenguaje heredado .
Marcadores de error	Muestra un subrayado rojo ondulado, también conocido como ondulado, debajo del texto que es sintácticamente incorrecto. Los marcadores de error suelen utilizarse para dar a conocer a los usuarios palabras clave mal escritas, paréntesis no cerrados, caracteres no válidos y errores similares. En las clases MPF, los marcadores de error se controlan automáticamente en el AddError método de la AuthoringSink clase.

Muchas de estas características requieren que el servicio de lenguaje analice el código fuente. A menudo puede reutilizar el código de tokenización y análisis para el compilador o intérprete.

Las siguientes características están relacionadas con la compatibilidad con lenguajes de programación, pero no forman parte de los servicios de lenguaje:

CARACTERÍSTICA	DESCRIPCIÓN
Evaluadores de expresiones	Admite Visual Studio el depurador validando puntos de interrupción y proporcionando una lista de expresiones que se mostrarán en la ventana de depuración Autos. Para obtener más información, vea Compatibilidad del servicio de lenguaje para la depuración.
Herramientas de exploración de símbolos	Soporta Examinador de objetos, Vista de clases, Navegador de llamadas y Buscar resultados de símbolos.

Desarrollar un servicio de lenguaje heredado

06/04/2020 • 3 minutes to read • [Edit Online](#)

En esta sección se vinculan a temas que le ayudan a crear un servicio de lenguaje heredado.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, consulte [Editor y extensiones](#) de servicio de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

En esta sección

- [Modelo de un servicio de lenguaje heredado](#)

Proporciona un modelo de un Visual Studio servicio de lenguaje mínimo para el editor principal. Puede utilizar este modelo como guía para crear su propio servicio de lenguaje.

- [Interfaces de servicio de lenguaje heredadas](#)

Describe los objetos necesarios para implementar un servicio de lenguaje y proporciona una lista de objetos adicionales que puede usar para proporcionar resaltado de sintaxis, datos de método y otras características.

- [Interceptar comandos de servicio de lenguaje heredado](#)

Describe cómo insertar un filtro de comandos en el servicio de lenguaje para interceptar comandos que la vista de texto controlaría de otro modo.

- [Registrar un servicio de lenguaje heredado](#)

Proporciona información sobre cómo registrar el Visual Studio servicio de lenguaje mediante .

- [Compatibilidad con el servicio de lenguaje para la depuración](#)

Describe cómo un servicio de lenguaje puede proporcionar características para admitir un depurador.

- [Lista de verificación: Crear un servicio de lenguaje heredado](#)

Proporciona instrucciones paso a paso para crear e integrar un servicio de lenguaje para el editor principal.

Secciones relacionadas

- [Coloreación de sintaxis en un servicio de lenguaje heredado](#)

Describe cómo implementar el resaltado de sintaxis en el servicio de lenguaje.

- [Finalización de instrucciones en un servicio de lenguaje heredado](#)

Describe la finalización de instrucciones, el proceso mediante el cual un servicio de lenguaje ayuda a los usuarios a finalizar una palabra clave de idioma o elemento que han comenzado a escribir.

- [Información de parámetros en un servicio de lenguaje heredado](#)

Describe cómo proporcionar sugerencias de método para funciones y métodos sobrecargados.

- [Cómo: Proporcionar soporte de texto oculto en un servicio de lenguaje heredado](#)

Explica el propósito de una región de texto oculto y proporciona instrucciones sobre cómo implementar una región de texto oculta.

- [Cómo: Proporcionar soporte de esquematización ampliada en un servicio de lenguaje heredado](#)

Explica las dos opciones que amplían la compatibilidad de esquematización para el lenguaje más allá de admitir el comando *Contraer a definiciones*.

Modelo de un servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un servicio de lenguaje define los elementos y características de un idioma específico y se usa para proporcionar al editor información específica de ese idioma. Por ejemplo, el editor necesita conocer los elementos y palabras clave del lenguaje para admitir el color de sintaxis.

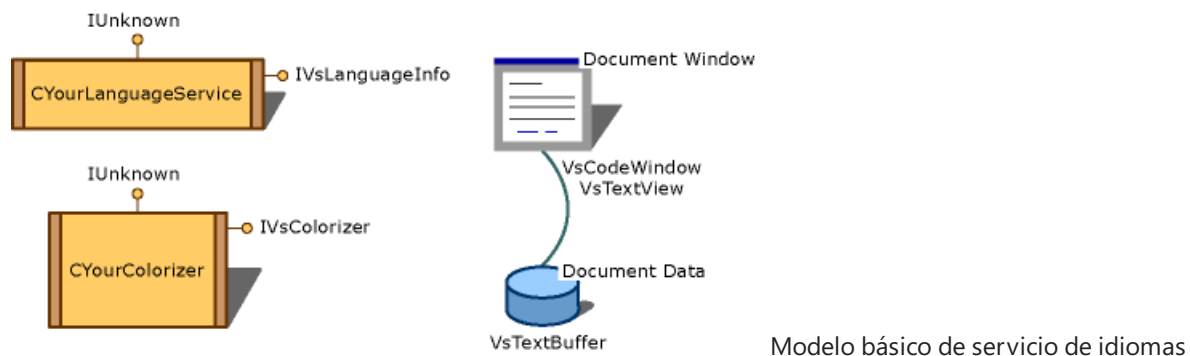
El servicio de lenguaje funciona estrechamente con el búfer de texto administrado por el editor y la vista que contiene el editor. La opción **Información rápida** de Microsoft IntelliSense es un ejemplo de una característica proporcionada por un servicio de lenguaje.

Un servicio de lenguaje mínimo

El servicio de lenguaje más básico contiene los dos objetos siguientes:

- El *servicio de* [IVsLanguageInfo](#) lenguaje implementa la interfaz. Un servicio de lenguaje tiene información sobre el idioma, incluido su nombre, extensiones de nombre de archivo, administrador de ventanas de código y colorante.
- El *colorante* implementa [IVsColorizer](#) la interfaz.

El siguiente dibujo conceptual muestra un modelo de un servicio de lenguaje básico.



La ventana del documento aloja la *vista* de Visual Studio documento del editor, en este caso el editor principal. La vista de documento y el búfer de texto son propiedad del editor. Estos objetos Visual Studio funcionan a través de una ventana de documento especializada denominada ventana de *código*. La ventana de código [IVsWindowFrame](#) está contenida en un objeto creado y controlado por el IDE.

Cuando se carga un archivo con una extensión determinada, el editor busca el servicio de lenguaje [GetCodeWindowManager](#) asociado a esa extensión y le pasa la ventana de código llamando al método. El servicio de lenguaje devuelve un administrador [IVsCodeWindowManager](#) de ventanas de *código*, que implementa la interfaz.

En la tabla siguiente se proporciona una visión general de los objetos del modelo.

COMPONENTE	OBJECT	FUNCIÓN
Búfer de texto	VsTextBuffer	Una secuencia de texto de lectura/escritura Unicode. Es posible que el texto utilice otras codificaciones.

COMPONENTE	OBJECT	FUNCIÓN
Ventana de código	VsCodeWindow	Ventana de documento que contiene una o varias vistas de texto. Cuando Visual Studio está en modo de interfaz de varios documentos (MDI), la ventana de código es un elemento secundario MDI.
Vista de texto	VsTextView	Una ventana que permite al usuario navegar y ver texto con el teclado y el ratón. Una vista de texto aparece al usuario como editor. Puede utilizar vistas de texto en las ventanas normales del editor, la ventana Salida y la ventana Inmediato. Además, puede configurar una o varias vistas de texto dentro de una ventana de código.
Administrador de texto	Gestionado por SVsTextManager el servicio, del IVsTextManager que se obtiene un puntero	Componente que mantiene información común compartida por todos los componentes descritos anteriormente.
Servicio de idiomas	Depende de la implementación; Implementa IVsLanguageInfo	Objeto que proporciona al editor información específica del lenguaje, como resaltado de sintaxis, finalización de instrucciones y coincidencia de llaves.

Vea también

- [Datos de documento y vista de documento en editores personalizados](#)

Interfaces de servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

Para cualquier lenguaje de programación en particular, solo puede haber una instancia de un servicio de lenguaje a la vez. Sin embargo, un servicio de idioma único puede servir a más de un editor.

Visual Studio asocia un servicio de lenguaje con ningún editor en particular. Por lo tanto, cuando solicite una operación de servicio de lenguaje, debe identificar el editor adecuado como parámetro.

Interfaces comunes asociadas con los servicios de idiomas

El editor obtiene el servicio [QueryService](#) de lenguaje llamando al `VSPackage` adecuado. El identificador de servicio (SID) pasado en esta llamada identifica el servicio de lenguaje que se solicita.

Puede implementar las interfaces de servicio de lenguaje principal en cualquier número de clases independientes. Sin embargo, un enfoque común es implementar las siguientes interfaces en una sola clase:

- [IVsLanguageInfo](#)
- [IVsProvideColorableItems](#)
- [IVsLanguageDebugInfo](#)
- [IVsLanguageBlock](#) (opcional)

La [IVsLanguageInfo](#) interfaz debe implementarse en todos los servicios de lenguaje. Proporciona información sobre el servicio de lenguaje, como el nombre localizado del idioma, las extensiones de nombre de archivo asociadas con el servicio de lenguaje y cómo recuperar un colorante.

Interfaces de servicio de lenguaje adicionales

Se pueden proporcionar otras interfaces con su servicio de lenguaje. Visual Studio solicita una instancia independiente de estas interfaces para cada instancia del búfer de texto. Por lo tanto, debe implementar cada una de estas interfaces en su propio objeto. En la tabla siguiente se muestran las interfaces que requieren una instancia por instancia de búfer de texto.

INTERFAZ	DESCRIPCIÓN
IVsCodeWindowManager	Administra los adornos de ventanas de código, como la barra desplegable. Puede obtener esta interfaz GetCodeWindowManager mediante el método. Hay uno IVsCodeWindowManager por ventana de código.
IVsColorizer	Coloriza las palabras clave y delimitadores de idioma. Puede obtener esta interfaz GetColorizer mediante el método. IVsColorizer se llama en el momento de la pintura. Evite el trabajo IVsColorizer intensivo en computación dentro o el rendimiento podría sufrir.

INTERFAZ	DESCRIPCIÓN
IVsMethodData	Proporciona información sobre herramientas de parámetros de IntelliSense. Cuando el servicio de lenguaje reconoce un carácter que indica que se deben mostrar los SetMethodData datos del método, como un paréntesis abierto, llama al método para notificar a la vista de texto que el servicio de lenguaje está listo para mostrar una información de parámetros. A continuación, la vista de texto vuelve a IVsMethodData llamar al servicio de lenguaje mediante los métodos de la interfaz para obtener la información necesaria para mostrar la información sobre herramientas.
IVsCompletionSet	Proporciona la finalización de instrucciones IntelliSense. Cuando el servicio de lenguaje está listo para UpdateCompletionStatus mostrar una lista de finalización, llama al método en la vista de texto. A continuación, la vista de texto vuelve IVsCompletionSet a llamar al servicio de lenguaje mediante métodos en el objeto.
IVsTextViewFilter	Permite modificar la vista de texto mediante el controlador de comandos. La clase en la IVsTextViewFilter que implementa IOleCommandTarget la interfaz también debe implementar la interfaz. La vista de IVsTextViewFilter texto recupera el IOleCommandTarget objeto consultando AddCommandFilter el objeto que se pasa al método. Debe haber IVsTextViewFilter un objeto para cada vista.
IOleCommandTarget	Intercepta comandos que el usuario escribe en la ventana de código. Supervise la IOleCommandTarget salida de su implementación para proporcionar información de finalización personalizada y ver la modificación Para pasar IOleCommandTarget el objeto a AddCommandFilter la vista de texto, llame a .

Vea también

- [Desarrollo de un servicio de lenguaje heredado](#)
- [Lista de comprobación: creación de un servicio de lenguaje heredado](#)

Intercepción de comandos del servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Con Visual Studio, puede tener los comandos de interceptación del servicio de lenguaje que la vista de texto manejaría de otro modo. Esto es útil para el comportamiento específico del idioma que la vista de texto no administra. Puede interceptar estos comandos agregando uno o varios filtros de comandos a la vista de texto desde el servicio de lenguaje.

Obtención y enrutamiento del comando

Un filtro de [IOleCommandTarget](#) comandos es un objeto que supervisa determinadas secuencias de caracteres o comandos de teclado. Puede asociar más de un filtro de comandos a una sola vista de texto. Cada vista de texto mantiene una cadena de filtros de comandos. Después de crear un nuevo filtro de comandos, agregue el filtro a la cadena para la vista de texto adecuada.

Llame [AddCommandFilter](#) al método [IVsTextView](#) en el para agregar el filtro de comandos a la cadena. Cuando se [AddCommandFilter](#) Visual Studio llama a , devuelve otro filtro de comandos al que puede pasar los comandos que el filtro de comandos no controla.

Dispone de las siguientes opciones para el control de comandos:

- Controle el comando y, a continuación, pase el comando al siguiente filtro de comandos de la cadena.
- Controle el comando y no pase el comando al siguiente filtro de comandos.
- No controle el comando, pero pase el comando al siguiente filtro de comandos.
- Ignore el comando. No lo maneje en el filtro actual y no lo pase al siguiente filtro.

Para obtener información sobre qué comandos debe controlar el servicio de lenguaje, consulte [Comandos importantes para filtros](#) de servicio de lenguaje .

Comandos importantes para los filtros del servicio de lenguaje

06/04/2020 • 4 minutes to read • [Edit Online](#)

Si desea crear un filtro de servicio de lenguaje con todas las funciones, considere la posibilidad de controlar los siguientes comandos. La lista completa de identificadores [VSConstants.VSStd2KCmdID](#) de comando se define en la enumeración para Visual C++ el código administrado y el archivo de encabezado Stdidcmd.h para código no administrado. Puede encontrar el archivo Stdidcmd.h en la ruta de *instalación del SDK* de Visual Studio .

Comandos para manejar

NOTE

No es obligatorio filtrar por cada comando de la tabla siguiente.

GET-HELP	DESCRIPCIÓN
VSConstants.VSStd2KCmdID	Se envía cuando el usuario hace clic con el botón derecho. Este comando indica que es el momento de proporcionar un menú contextual. Si no controla este comando, el editor de texto proporciona un menú contextual predeterminado sin ningún comando específico del idioma. Para incluir sus propios comandos en este menú, controle el comando y muestre un menú contextual usted mismo.
VSConstants.VSStd2KCmdID	Normalmente se envía cuando el usuario escribe CTRL+J. Llame UpdateCompletionStatus al método IVsTextView en el para mostrar el cuadro de finalización de la instrucción.
VSConstants.VSStd2KCmdID	Se envía cuando el usuario escribe un carácter. Supervise este comando para determinar cuándo se escribe un carácter de desencadenador y para proporcionar la finalización de instrucciones, sugerencias de método y marcadores de texto, como el color de sintaxis, la coincidencia de llaves y los marcadores de error. Llame UpdateCompletionStatus al método IVsTextView en la SetMethodData finalización IVsMethodTipWindow de la instrucción for y el método en las sugerencias del método for. Para admitir marcadores de texto, supervise este comando para determinar si el carácter que se está escribiendo requiere que actualice los marcadores.
VSConstants.VSStd2KCmdID	Se envía cuando el usuario escribe la tecla Intro. Supervise este comando para determinar cuándo se debe OnDismiss descartar IVsMethodData una ventana de sugerencia de método llamando al método en el archivo . De forma predeterminada, la vista de texto controla este comando.
VSConstants.VSStd2KCmdID	Se envía cuando el usuario escribe la tecla Retroceso. Supervisar para determinar cuándo se debe descartar OnDismiss una IVsMethodData ventana de sugerencia de método llamando al método en el archivo . De forma predeterminada, la vista de texto controla este comando.

GET-HELP	DESCRIPCIÓN
VSConstants.VSStd2KCmdID	Se envía desde un menú o una tecla de método abreviado. Llame UpdateTipWindow al método IVsTextView en el para actualizar la ventana de sugerencia con la información del parámetro.
VSConstants.VSStd2KCmdID	Se envía cuando el usuario pasa el cursor sobre una variable o coloca el cursor en una variable y selecciona Información rápida de IntelliSense en el menú Editar . Devolver el tipo de la variable UpdateTipWindow en una IVsTextView sugerencia llamando al método en el archivo . Si la depuración está activa, la sugerencia también debe mostrar el valor de la variable.
VSConstants.VSStd2KCmdID	Normalmente se envía cuando el usuario escribe CTRL+BARRA ESPACIADORA. Este comando indica al servicio UpdateCompletionStatus de lenguaje IVsTextView que llame al método en el archivo .
VSConstants.VSStd2KCmdID VSConstants.VSStd2KCmdID	Se envía desde un menú, normalmente Selección de comentarios o Selección de descomisade de Avanzado en el menú Edición . VSConstants.VSStd2KCmdID indica que el usuario desea comentar el texto seleccionado; VSConstants.VSStd2KCmdID indica que el usuario desea quitar el comentario del texto seleccionado. Estos comandos solo se pueden implementar mediante el servicio de lenguaje.

Vea también

- [Desarrollo de un servicio de lenguaje heredado](#)

Registro de un servicio de lenguaje heredado

06/04/2020 • 14 minutes to read • [Edit Online](#)

En las secciones siguientes se proporcionan listas de Visual Studio entradas de registro para las distintas opciones de servicio de lenguaje disponibles en .

En la siguiente lista de entradas del Registro, raíz de *vs*, *VS Reg* es igual a Visual Studio HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio\X.Y, donde X.Y es el número de versión.

Entradas del Registro para Opciones de Servicio de Lenguaje

La clave de nombre de *idioma* de VS Reg Root (Raíz de registro de *VS*) puede contener los siguientes valores.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	<GUID>	GUID del servicio de lenguaje.
LangResID	REG_DWORD	0x0-0xffff	Identificador de recursos de cadena (ResID) para el nombre de texto localizado del idioma.
Paquete	REG_SZ	<GUID>	GUID del VSPackage.
ShowCompletion	REG_DWORD	0-1	Especifica si las opciones de finalización de instrucción en el cuadro de diálogo Opciones están habilitadas.
ShowSmartIndent	REG_DWORD	0-1	Especifica si la opción para seleccionar Sangría inteligente en el cuadro de diálogo Opciones está habilitada.
RequestStockColors	REG_DWORD	0-1	Especifica si los colores personalizados o predeterminados se utilizan para colorear palabras clave.
ShowHotURLs	REG_DWORD	0-1	Especifica si el usuario puede hacer clic en direcciones URL.
Predeterminado en URL no calientes	REG_DWORD	0-1	Especifica la configuración inicial de la opción Habilitar navegación URL con un solo clic en el cuadro de diálogo Opciones .

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
DefaultToInsertSpaces	REG_DWORD	0-1	Especifica si el servicio de lenguaje tiene "insertar espacios" como su opción de ficha predeterminada.
ShowDropDownBarOption	REG_DWORD	0-1	Habilita o deshabilita la opción Barra de navegación en el cuadro de diálogo Opciones que muestra u oculta la barra de navegación .
Solo una ventana de código único	REG_DWORD	0-1	Habilita o deshabilita la opción Nueva ventana del menú Ventana de un servicio de idioma.
EnableAdvancedMembersOption	REG_DWORD	0-1	Habilita o deshabilita una configuración de cuadro de diálogo Opciones para Ocultar miembros avanzados .
Soporte CF_HTML	REG_DWORD	0-1	Especifica si el editor permite copiar y pegar datos HTML.
EnableLineNumbersOption	REG_DWORD	0-1	Especifica si las opciones Números de línea del cuadro de diálogo Opciones están habilitadas para un servicio de lenguaje.
HideAdvancedMembersByDefault	REG_DWORD	0-1	Especifica si los miembros avanzados, como los campos privados, están ocultos en las listas de finalización.
ShowBraceCompletion	REG_DWORD	0-1	Especifica si la opción Finalización de llaves en el cuadro de diálogo Opciones está habilitada.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  Languages\
    Language Services\
      C/C++\
        (Default)           = reg_sz:{B2F072B0-ABC1-11D0-9D62-00C04FD9D9D9}
        LangResID            = reg_dword:0x00000000
        Package              = reg_sz:{8C2EA640-ABC1-11D0-9D62-00C04FD9D9D9}
        ShowCompletion        = reg_dword:0x00000001
        ShowSmartIndent       = reg_dword:0x00000001
        ShowDropDownBarOption = reg_dword:0x00000001
```

Entradas del Registro para Opciones de Lenguajes del Depurador

La clave DE LA *raíz de registro* de VS , "Idiomas" y Nombre de *idioma* de los servicios de lenguaje de los servicios de lenguaje , el *GUID* de los idiomas del depurador, puede incluir los siguientes valores.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	text	El valor predeterminado se puede utilizar para documentar el nombre del idioma. El nombre de esta clave es un GUID de un * < *evaluador de expresiones que tiene una entrada correspondiente en la raíz de vs reg>.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  Languages\
    Language Services\
      C/C++\
        Debugger Languages\
          {3A12D0B7-C26C-11D0-B442-00A0244A1DD2}\
            (Default) = reg_sz:C++
```

Entradas del Registro para opciones de herramientas del editor

Puede agregar claves del Registro en la clave EditorToolsOptions para páginas de propiedades y nodos de propiedades. Estas claves y sus valores identifican las páginas de propiedades en el cuadro de diálogo **Opciones** (en el menú **Herramientas**) que se usan para configurar el servicio de lenguaje. En el ejemplo siguiente, *Nombre* de página es el nombre de una página de propiedades y Nombre de *nodo* es el nombre de un nodo del árbol en el cuadro de diálogo **Opciones**. La entrada de página y la entrada de nodo deben especificarse por separado.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	Resnum	El nombre para mostrar localizado de esta página de opciones. El nombre puede ser texto <code>nnn</code> literal, o , donde <code>nnn</code> es un identificador de recurso de cadena en el archivo DLL satélite del VSPackage especificado.
Paquete	REG_SZ	<i>GUID</i>	GUID del VSPackage que implementa esta página de opciones.
Página	REG_SZ	<i>GUID</i>	El GUID de la página de propiedades para GetPropertyPage solicitar desde el VSPackage mediante una llamada al método. Si esta entrada del Registro no está presente, la clave del Registro describe un nodo, no una página.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  Languages\
    Language Services\
      CSharp\
        EditorToolsOptions\
          Formatting\
            (Default) = reg_sz:#242
            Package   = reg_sz:{A066E284-DCAB-11D2-B551-00C04F68D4DB}
          General\
            (Default) = reg_sz:#255
            Package   = reg_sz:{A066E284-DCAB-11D2-B551-00C04F68D4DB}
            Page      = reg_sz:{3EB2CC0B-033E-4D75-B26A-B2362C25227E}
          Indentation\
            (Default) = reg_sz:#250
            Package   = reg_sz:{A066E284-DCAB-11D2-B551-00C04F68D4DB}
            Page      = reg_sz:{5E21D017-6D2A-4114-A1F1-C923F001CBBB}
          Newlines\
            (Default) = reg_sz:#253
            Package   = reg_sz:{A066E284-DCAB-11D2-B551-00C04F68D4DB}
            Page      = reg_sz:{607D8062-68D1-41E4-9A35-B5E7F14D0481}
```

Entradas de registro para opciones de extensión de nombre de archivo

La entrada de la extensión de archivo debe incluir el período inicial, por ejemplo ".myext".

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	<i>GUID</i>	GUID de servicio para el servicio de idioma predeterminado para este tipo de extensión de nombre de archivo.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  Languages\
    File Extensions\
      .cpp\
        (Default) = {B2F072B0-ABC1-11D0-9D62-00C04FD9DFD9}
```

Entradas del Registro para Opciones del Editor

La clave *raíz de VS Reg*, editores, puede contener los siguientes valores:

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	""	Sin usar; puede poner su nombre aquí para la documentación.
DefaultToolboxTab	REG_SZ	""	Nombre de la pestaña de la caja de herramientas para que se haga por defecto cuando el editor está activo.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
DisplayName	REG_SZ	Resnum	Nombre para mostrar en el cuadro de diálogo Abrir con . El nombre es el identificador de recurso de cadena o un nombre en formato estándar.
ExcludeDefTextEditor	REG_DWORD	0-1	Se utiliza para el comando de menú Abrir con . Si no desea enumerar el editor de texto predeterminado en la lista de editores disponibles para un tipo de archivo específico, establezca este valor en 1.
LinkedEditorGUID	REG_SZ	<GUID>	Se utiliza para cualquier servicio de lenguaje que pueda abrir un archivo con compatibilidad con la página de códigos. Por ejemplo, al abrir un archivo .txt mediante el comando Abrir con, se proporcionan opciones para usar el editor de código fuente con y sin codificación. El GUID especificado en el nombre de la subclave es para el generador del editor de páginas de códigos; el GUID vinculado especificado en esta entrada de registro específica es para el generador de editornormal. El propósito de esta entrada es que si el IDE no abre un archivo mediante el editor predeterminado, el IDE intentará usar el siguiente editor de la lista. Este siguiente editor no debe ser la fábrica del editor de páginas de códigos porque esta fábrica de editores es básicamente la misma que la fábrica del editor que falló.
Paquete	REG_SZ	<GUID>	GUID de VSPackage para el nombre para mostrar ResID.

Ejemplo


```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  \Editors\
    {8281C572-2171-45AA-A642-7D8BC1662F1C}\
      (Default)           = reg_sz:Html Editor with Encoding
      DefaultToolboxTab    = reg_sz:HTML
      DisplayName          = reg_sz:#20101
      LinkedEditorGUID     = reg_sz:{C76D83F8-A489-11D0-8195-00A0C91BBEE3}
      Package              = reg_sz:{1B437D20-F8FE-11D2-A6AE-00104BCC7269}
```

Entradas del Registro para Opciones de Vista Lógica

La *interfaz de usuario* de LA GUI de VS Reg Root (Raíz de registro de VS) > la clave .LogicalViews puede contener los siguientes valores.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ		Sin usar.
<GUID>	REG_SZ	""	Clave de las vistas lógicas admitidas. Puedes tener tantos de estos como necesites. El nombre de la entrada del registro es lo que es importante, no el valor, que siempre es una cadena vacía.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  \Editors\
    {8281C572-2171-45AA-A642-7D8BC1662F1C}\
      LogicalViews\
        (Default) = reg_sz:
        {7651a700-06e5-11d1-8ebd-00a0c90f26ea} = reg_sz:
        {7651a701-06e5-11d1-8ebd-00a0c90f26ea} = reg_sz:
        {7651a702-06e5-11d1-8ebd-00a0c90f26ea} = reg_sz:
        {7651a703-06e5-11d1-8ebd-00a0c90f26ea} = reg_sz:
```

Entradas del Registro para Opciones de Extensión Editor

La clave GUID de *la raíz de vsregistro* de VS Reg \, *editor* de editores de editores, puede contener los siguientes valores. La extensión de nombre de archivo no incluye el período inicial.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ		Sin usar.
<ext>	REG_DWORD	0-0xffffffff	Prioridad relativa de las extensiones. Si dos o más idiomas comparten la misma extensión, se elige el idioma de mayor prioridad.

Además, la selección predeterminada del usuario actual para un editor se almacena en HKEY_Current_User, Software, Microsoft, VisualStudioX.Y, Editores\predeterminados. El GUID del servicio de idioma seleccionado se encuentra en la entrada personalizada. Esto tiene prioridad para el usuario actual.

Ejemplo

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\9.0\
\Editors\
{8281C572-2171-45AA-A642-7D8BC1662F1C}\
Extensions\
(Default) = reg_sz:
*          = reg_dword:0x00000018
html       = reg_dword:0x00000027
shtm       = reg_dword:0x00000027
shtml      = reg_dword:0x00000027
```

Entradas del Registro para opciones de servicio de lenguaje de Managed Package Framework

Las siguientes entradas del Registro son específicas de las clases de servicio de lenguaje del marco de paquete administrado (MPF). Estas entradas del Registro indican compatibilidad en el servicio de lenguaje para varias características de IntelliSense y para otras características de edición avanzadas.

Se accede a estas [LanguagePreferences](#) entradas del registro a través de la clase.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
CodeSense	REG_DWORD	0-1	Compatibilidad con operaciones de IntelliSense.
MatchBraces	REG_DWORD	0-1	Compatibilidad con pares de idiomas coincidentes, como llaves, paréntesis y corchetes.
InformaciónRápida	REG_DWORD	0-1	Compatibilidad con la operación de información rápida de IntelliSense.
ShowMatchingBrace	REG_DWORD	0-1	Compatibilidad para mostrar el par de idioma coincidente en la barra de estado.
MatchBracesAtCaret	REG_DWORD	0-1	Compatibilidad con la visualización de pares de idiomas coincidentes, normalmente resaltando los dos elementos.
MaxErrorMessage	REG_DWORD	0-n	El número máximo de errores que se pueden mostrar en la ventana Lista de errores .
CodeSenseDelay	REG_DWORD	0-n	El número de milisegundos que se debe retrasar antes de iniciar cualquier análisis en segundo plano para una operación de IntelliSense.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
EnableAsyncCompletion	REG_DWORD	0-1	Soporte para el análisis en segundo plano.
Habilitarcomentarios	REG_DWORD	0-1	Compatibilidad con comentarios de bloques de texto seleccionados, y también implica compatibilidad para quitar los comentarios del texto seleccionado.
EnableFormatSelection	REG_DWORD	0-1	Compatibilidad con el formato de texto, como la sangría automática o el ajuste de la posición de las llaves.
AutoOutlining	REG_DWORD	0-1	Compatibilidad con esquematización (regiones que se pueden contraer).
MaxRegions	REG_DWORD	0-n	El número máximo de regiones ocultas por archivo.

```

Example\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0Exp\
  Languages\
    Language Services\
      XML\
        (Default)           = reg_sz:{f6819a78-a205-47b5-be1c-675b3c7f0b8e}
        MatchBraces         = reg_dword:0x00000001
        QuickInfo           = reg_dword:0x00000001
        ShowMatchingBrace   = reg_dword:0x00000001
        MatchBracesAtCaret  = reg_dword:0x00000000
        MaxErrorMessages    = reg_dword:0x00000064
        CodeSenseDelay      = reg_dword:0x000001f4
        MaxRegions          = reg_dword:0x0000000a

```

Vea también

- [Desarrollo de un servicio de lenguaje heredado](#)

Lista de verificación: Crear un servicio de lenguaje heredado

06/04/2020 • 7 minutes to read • [Edit Online](#)

La siguiente lista de comprobación resume los pasos básicos Visual Studio que debe seguir para crear un servicio de lenguaje para el editor principal. Para integrar el Visual Studio servicio de lenguaje en , debe crear un evaluador de expresiones de depuración. Para obtener más información, vea [Escribir un evaluador](#) de expresiones CLR en la [extensibilidad](#) del depurador de Visual Studio .

Pasos para crear un servicio de idiomas

1. Implemente la interfaz [IVsPackage](#).

- En el VSPackage, [IServiceProvider](#) implemente la interfaz para proporcionar el servicio de lenguaje.
- Haga que su servicio de lenguaje esté disponible [SetSite](#) para el entorno de desarrollo integrado (IDE) en su implementación.

2. Implemente [IVsLanguageInfo](#) la interfaz en la clase de servicio de lenguaje principal.

La [IVsLanguageInfo](#) interfaz es el punto de partida de la interacción entre el editor principal y el servicio de lenguaje.

Características opcionales

Las siguientes características son opcionales y se pueden implementar en cualquier orden. Estas características aumentan la funcionalidad de su servicio de lenguaje.

- Seleccionar el color de la sintaxis

Implemente la interfaz [IVsColorizer](#). La implementación de esta interfaz debe la información del analizador para devolver la información de color adecuada.

El [GetColorizer](#) método [IVsColorizer](#) devuelve la interfaz. Se crea una instancia de colorante independiente para [IVsColorizer](#) cada búfer de texto, por lo que debe implementar la interfaz por separado. Para obtener más información, consulte [Coloración](#) de sintaxis en un servicio de lenguaje heredado.

- Ventana de código

Implemente [IVsCodeWindowManager](#) la interfaz para permitir que el servicio de lenguaje reciba una notificación de cuándo se crea una nueva ventana de código.

El [GetCodeWindowManager](#) método [IVsCodeWindowManager](#) devuelve la interfaz. A continuación, el servicio de lenguaje [AddAdornments](#) puede agregar una interfaz de usuario especial a la ventana de código en . El servicio de lenguaje también puede realizar cualquier procesamiento [OnNewView](#) especial, como agregar un filtro de vista de texto en .

- Filtro de vista de texto

Para proporcionar la finalización de instrucciones de IntelliSense en un servicio de lenguaje, debe interceptar algunos de los comandos que la vista de texto controlaría de otro modo. Para interceptar estos comandos, siga estos pasos:

- Implemente [IOleCommandTarget](#) para participar en la cadena de comandos y controlar los comandos del editor.

- Llame [AddCommandFilter](#) al método y [IOleCommandTarget](#) pase la implementación.
- Llame [RemoveCommandFilter](#) al método cuando se desasocie de la vista para que estos comandos ya no se le pasen.

Los comandos que se deben controlar dependen de los servicios que se proporcionan. Para obtener más información, consulte [Comandos importantes para filtros](#) de servicio de lenguaje.

NOTE

La [IVsTextViewFilter](#) interfaz debe implementarse en [IOleCommandTarget](#) el mismo objeto que la interfaz.

- Instrucciones completadas

Implemente la interfaz [IVsCompletionSet](#).

Admite el comando statement completion [VSConstants.VSStd2KCmdID](#) (es [UpdateCompletionStatus](#) decir, [IVsTextView](#)) y [IVsCompletionSet](#) llama al método en la interfaz, pasando la interfaz. Para obtener más información, consulte Finalización de instrucciones en un servicio de [lenguaje heredado](#).

- Consejos de método

Implemente [IVsMethodData](#) la interfaz para proporcionar datos para la ventana de sugerencia de método.

Instale el filtro de vista de texto para controlar los comandos de forma adecuada, de modo que sepa cuándo mostrar una ventana de sugerencia de datos de método. Para obtener más información, consulte [Información de parámetros en un servicio](#) de lenguaje heredado.

- Marcadores de error

Implemente la interfaz [IVsTextMarkerClient](#).

Cree los objetos de [IVsTextMarkerClient](#) marcador de [CreateLineMarker](#) error que [IVsTextMarkerClient](#) implementan la interfaz y llaman al método, pasando la interfaz del objeto de marcador de error.

Normalmente, cada marcador de error administra un elemento en la ventana de lista de tareas.

- Elementos de la lista de tareas

Implemente una clase de [IVsTaskItem](#) elemento de tarea que proporcione la interfaz.

Implemente una clase de [IVsTaskProvider](#) proveedor [IVsTaskProvider2](#) de tareas que proporcione la interfaz y la interfaz.

Implemente una clase de [IVsEnumTaskItems](#) enumerador de tareas que proporcione la interfaz.

Registre el proveedor de tareas [RegisterTaskProvider](#) con el método de la lista de tareas.

Obtenga [IVsTaskList](#) la interfaz llamando al proveedor de servicios [SVsTaskList](#) del servicio de lenguaje con el GUID del servicio.

Cree objetos de [RefreshTasks](#) elemento de [IVsTaskList](#) tarea y llame al método en la interfaz cuando haya tareas nuevas o actualizadas.

- Comentar elementos de tarea

Utilice [IVsCommentTaskInfo](#) la interfaz y la [IVsEnumCommentTaskTokens](#) interfaz para obtener los tokens de tarea de comentario.

Obtenga [IVsCommentTaskInfo](#) una interfaz del [SVsTaskList](#) servicio.

Obtenga [IVsEnumCommentTaskTokens](#) la interfaz del [EnumTokens](#) método.

Implemente [IVsTaskListEvents](#) la interfaz para escuchar los cambios en la lista de tokens.

- **esquematar**

Hay varias opciones para apoyar la esquematización. Por ejemplo, puede admitir el comando **Contraer a definiciones**, proporcionar regiones de esquema controladas por el editor o admitir regiones controladas por el cliente. Para obtener más información, consulte [Cómo: proporcionar compatibilidad de esquematización ampliada en un servicio](#) de lenguaje heredado.

- **Registro del servicio de idiomas**

Para obtener más información acerca de cómo registrar un servicio de lenguaje, vea [Registrar un servicio](#) de lenguaje heredado y Administrar [VSPackages](#).

- **Ayuda contextual**

Proporcione contexto al editor de una de las siguientes maneras:

- Proporcione contexto para los [IVsTextMarkerContextProvider](#) marcadores de texto mediante la implementación de la interfaz.
- Proporcione todo el contexto [IVsLanguageContextProvider](#) de usuario mediante la implementación de la interfaz.

Vea también

- [Desarrollar un servicio de lenguaje heredado](#)
- [Escribir un evaluador de expresiones CLR](#)

Características del servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

En los temas siguientes se enumeran algunas de las características del servicio de lenguaje heredado que puede proporcionar.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, vea [Extensiones de servicio de editor y lenguaje](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

En esta sección

- [Colores de la sintaxis en un servicio de lenguaje heredado](#)

Explica cómo implementar el color de sintaxis.

- [Formato automático en un servicio de lenguaje heredado](#)

Explica cómo implementar el formato automático.

- [Información de parámetros en un servicio de lenguaje heredado](#)

Explica cómo implementar la información sobre herramientas de parámetros de IntelliSense.

- [Finalización de instrucciones en un servicio de lenguaje heredado](#)

Explica cómo implementar la lista de instrucciones de IntelliSense y la lista de finalización de miembros.

- [Esquematización y texto oculto en un servicio de lenguaje heredado](#)

Explica cómo implementar la esquematización u texto oculto.

- [Provisión de compatibilidad con esquematización ampliada en un servicio de lenguaje heredado](#)

Explica algunos de los pasos para implementar la compatibilidad con el depurador.

Secciones relacionadas

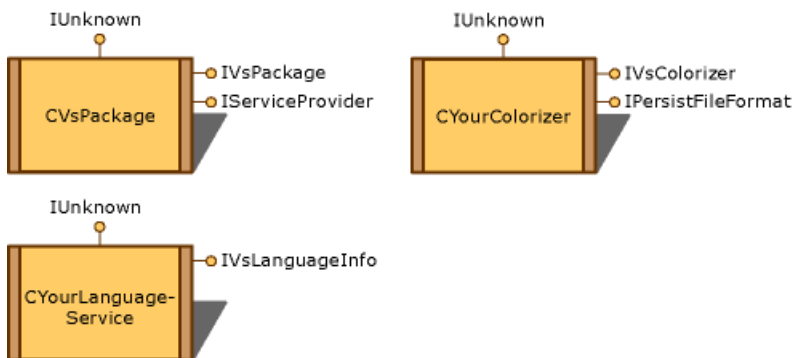
Colores de la sintaxis en un servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

Visual Studio usa un servicio de colorear para identificar elementos del lenguaje y mostrarlos con los colores especificados en un editor.

Modelo de colorante

El servicio de [IVsColorizer](#) lenguaje implementa la interfaz, que luego usa los editores. Esta implementación es un objeto independiente del servicio de lenguaje, como se muestra en la siguiente ilustración:



NOTE

El servicio de coloración de sintaxis es independiente del mecanismo general de Visual Studio para colorear texto. Para obtener más Visual Studio SDK información sobre el mecanismo general que admite la coloración, consulte [Uso de fuentes y colores](#).

Además del colorante, el servicio de lenguaje puede proporcionar elementos coloreables personalizados que son utilizados por el editor, mediante la publicidad de que proporciona artículos coloreables personalizados. Puede hacerlo implementando [IVsProvideColorableItems](#) la interfaz en el [IVsLanguageInfo](#) mismo objeto que implementa la interfaz. Devuelve el número de elementos coloreables personalizados cuando el editor llama al [GetItemCount](#) método [GetColorableItem](#) y devuelve un elemento coloreable personalizado individual cuando el editor llama al método.

El [GetColorableItem](#) método devuelve un objeto [IVsColorableItem](#) que implementa la interfaz. Si el servicio de lenguaje admite valores de color [IVsHiColorItem](#) de 24 bits [IVsColorableItem](#) o altos, debe implementar la interfaz en el mismo objeto que la interfaz.

Cómo un VSPackage utiliza un colorante de servicio de lenguaje

1. El VSPackage debe obtener el servicio de lenguaje adecuado, que requiere el servicio de lenguaje VSPackage para hacer lo siguiente:
 - a. Utilice un objeto [IVsTextBuffer](#) que implemente la interfaz para obtener el texto que se va a colorear.

Normalmente, el texto se muestra mediante [IVsTextView](#) un objeto que implementa la interfaz.
 - b. Obtenga el servicio de lenguaje consultando el proveedor de servicios de VSPackage para el GUID del servicio de lenguaje. Los servicios de idioma se identifican en el registro por extensión de

archivo.

- c. Asocie el servicio [IVsTextBuffer](#) de lenguaje [SetLanguageServiceID](#) con el mediante una llamada a su método.

2. El VSPackage ahora puede obtener y usar el objeto colorizer de la siguiente manera:

NOTE

VSPackages que usan el editor principal no tienen que obtener explícitamente los objetos de colorizador de un servicio de lenguaje. Tan pronto como una instancia del editor principal obtiene un servicio de lenguaje adecuado, realiza todas las tareas de coloración que se muestran aquí.

- a. Obtener el objeto colorizador del servicio de [IVsColorizer](#) lenguaje, que implementa el , y [IVsColorizer2](#) las interfaces, mediante una llamada al [GetColorizer](#) método en el objeto del [IVsLanguageInfo](#) servicio de lenguaje.
- b. Llame [ColorizeLine](#) al método para obtener la información del colorante para un intervalo determinado de texto.

[ColorizeLine](#) devuelve una matriz de valores, uno para cada carácter del intervalo de texto que se está coloreando. Los valores son índices en una lista de elementos coloreable que es la lista de elementos coloreables predeterminada que mantiene el editor principal o una lista de elementos coloreable sin formato personalizada mantenida por el propio servicio de lenguaje.

- c. Utilice la información de [ColorizeLine](#) coloración devuelta por el método para mostrar el texto seleccionado.

NOTE

Además de usar un colorizador de servicio de lenguaje, un VSPackage también puede usar el mecanismo de color de texto de Visual Studio de uso general. Para obtener más información acerca de este mecanismo, consulte [Uso de fuentes y colores](#).

En esta sección

- [Implementación de colores de la sintaxis](#)

Describe cómo un editor tiene acceso a la coloración de sintaxis de un servicio de lenguaje y lo que el servicio de lenguaje debe implementar para admitir el color de sintaxis.

- [Uso de elementos coloreables integrados](#)

Muestra cómo usar elementos coloreables integrados del servicio de lenguaje.

- [Elementos coloreables personalizados](#)

Describe cómo implementar elementos coloreables personalizados.

Implementación de colores de la sintaxis

06/04/2020 • 5 minutes to read • [Edit Online](#)

Cuando el servicio de lenguaje proporciona coloración de sintaxis, el analizador convierte una línea de texto en una matriz de elementos coloreables y devuelve tipos de token correspondientes a estos elementos coloreables. El analizador debe devolver tipos de token que pertenecen a una lista de elementos coloreables. Visual Studio muestra cada elemento coloreable en la ventana de código según los atributos asignados por el objeto `Colorizer` al tipo de token adecuado.

Visual Studio especifica una interfaz de analizador y la implementación del analizador depende completamente de usted. Sin embargo, se proporciona una implementación de analizador predeterminada en el proyecto paquete de lenguaje de Visual Studio. Para el código administrado, el marco de paquete administrado (MPF) proporciona compatibilidad completa para colorear texto.

Los servicios de lenguaje heredados se implementan como parte de un `VSPackage`, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar el color de sintaxis, consulte [Tutorial: Resaltar texto](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Pasos seguidos por un editor para colorear texto

1. El editor obtiene el colorante `GetColorizer` llamando `IVsLanguageInfo` al método en el objeto.
2. El editor `GetStateMaintenanceFlag` llama al método para determinar si el colorante necesita que el estado de cada línea se mantenga fuera del colorante.
3. Si el colorizador requiere que el estado se mantenga fuera `GetStartState` del colorizador, el editor llama al método para obtener el estado de la primera línea.
4. Para cada línea del búfer, `ColorizeLine` el editor llama al método, que realiza los pasos siguientes:
 - a. La línea de texto se pasa a un analizador para convertir el texto en tokens. Cada token especifica el texto del token y el tipo de token.
 - b. El tipo de token se convierte en un índice en una lista de elementos coloreables.
 - c. La información del token se utiliza para rellenar una matriz de modo que cada elemento de la matriz corresponda a un carácter de la línea. Los valores almacenados en la matriz son los índices de la lista de elementos coloreables.
 - d. El estado al final de la línea se devuelve para cada línea.
5. Si el colorizador requiere que se mantenga el estado, el editor almacena en caché el estado de esa línea.
6. El editor representa la línea de texto `ColorizeLine` utilizando la información devuelta por el método. Para ello, siga estos pasos:
 - a. Para cada carácter de la línea, obtenga el índice de elementos coloreables.
 - b. Si utiliza los elementos coloreables predeterminados, acceda a la lista de elementos coloreables del

editor.

- c. De lo contrario, llame [GetColorableItem](#) al método del servicio de lenguaje para obtener un elemento coloreable.
- d. Utilice la información del elemento coloreable para representar el texto en la pantalla.

Colorizador de marco de paquetes administrados

El marco de paquete administrado (MPF) proporciona todas las clases necesarias para implementar un colorante. La clase de servicio [LanguageService](#) de lenguaje debe heredar la clase e implementar los métodos necesarios. Debe proporcionar un analizador y un [IScanner](#) analizador implementando la interfaz [GetScanner](#) y devolver una instancia de esa [LanguageService](#) interfaz desde el método (uno de los métodos que se deben implementar en la clase). Para obtener más información, vea [Colorear sintaxis en un servicio de lenguaje heredado](#).

Vea también

- [Uso de elementos coloreables integrados](#)
- [Elementos coloreables personalizados](#)
- [Desarrollo de un servicio de lenguaje heredado](#)
- [Coloreado de sintaxis en un servicio de lenguaje heredado](#)

Cómo: Usar elementos coloreables incorporados

06/04/2020 • 2 minutes to read • [Edit Online](#)

Antes de utilizar los elementos coloreables integrados, primero debe indicar al entorno de desarrollo integrado (IDE) que [IVsProvideColorableItems](#) no está proporcionando sus propios elementos coloreables personalizados, que en este caso serían objetos. Para ello, establezca una entrada del Registro para el servicio de lenguaje.

Para usar elementos coloreables integrados

1. En `HKEY_LOCAL_MACHINE\VisualStudio\<X.Y>\Languages,>Language Services<Language Name >`, donde `<X.Y>` es una versión de Visual Studio y `Language Name` es el nombre de su idioma, cree un valor de entrada de registro DWORD denominado **RequestStockColors**.
2. Establezca el valor de entrada del Registro **RequestStockColors** en `1`.

Después de crear la entrada del [ColorizeLine](#) Registro, el método [DEFAULTITEMS](#) del colorante puede usar los miembros de la enumeración para rellenar la matriz de atributos de color para su uso por el editor.

NOTE

No establezca esta entrada del Registro si proporciona elementos coloreables personalizados. Para obtener más información, consulte [Elementos coloreables personalizados](#).

Vea también

- [Coloración de sintaxis en editores personalizados](#)
- [Coloreación de sintaxis en un servicio de lenguaje heredado](#)
- [Implementación de coloración de sintaxis](#)
- [Artículos coloreables personalizados](#)
- [Registrar un servicio de lenguaje heredado](#)

Artículos coloreables personalizados

06/04/2020 • 6 minutes to read • [Edit Online](#)

Puede invalidar la lista de tipos para colorear, como palabras clave y comentarios, implementando elementos coloreables personalizados como parte del servicio de lenguaje.

Configuración de usuario de elementos coloreables

Puede mostrar el cuadro de diálogo **Fuentes y colores** seleccionando **Opciones** en el menú **Herramientas** y, a continuación, seleccionando **Fuentes y colores** en **Entorno**. Al seleccionar una visualización, como **Editor** de texto o **Ventana de comandos**, el cuadro de lista **Mostrar elementos** muestra todos los elementos coloreables para esa visualización. Puede ver y cambiar la fuente, el tamaño, el color de primer plano y el color de fondo de cada elemento coloreable. Las opciones se almacenan en una memoria caché en el registro y se accede a ellas mediante el nombre del elemento coloreable.

Presentación de artículos coloreables

Dado que el IDE controla las invalidaciones de usuario de elementos coloreables en el **fuentes y colores** cuadro de diálogo, solo necesita proporcionar cada elemento coloreable personalizado con un nombre. Este nombre es el que aparece en la lista **Mostrar elementos**. Los elementos coloreables aparecen en orden alfabético. Para agrupar los elementos coloreables personalizados de su servicio de lenguaje, puede comenzar cada nombre con su nombre de idioma, por ejemplo **NewLanguage - Comment** y **NewLanguage - Keyword**.

Caution

Debe incluir el nombre del idioma en el nombre del elemento coloreable para evitar colisiones con los nombres de elementos coloreables existentes. Si cambia el nombre de uno de los elementos coloreables durante el desarrollo, debe restablecer la memoria caché que se creó la primera vez que se tuvo acceso a los elementos coloreables. Puede restablecer la memoria caché experimental con la herramienta **CreateExpInstance**, que se instala con el SDK de Visual Studio, normalmente en el directorio:

C:\Archivos de programa (x86)-Microsoft Visual Studio 14.0-VSSDK-VisualStudioIntegration-Tools-Bin

Para restablecer la memoria caché, escriba **CreateExpInstance /Reset**. Para obtener más información acerca de **CreateExpInstance**, vea [CreateExpInstance utility](#).

Nunca se hace referencia al primer elemento de la lista de elementos coloreables. El primer elemento corresponde a un índice de Visual Studio elemento coloreable de 0 y siempre proporciona los colores y atributos de texto predeterminados para ese elemento. La forma más fácil de tratar con este elemento sin referencia es proporcionar un elemento coloreable marcador de posición en su lista como el primer elemento.

Implementar elementos coloreables personalizados

1. Defina lo que se debe colorear en su idioma, por ejemplo Palabra clave, Operador e Identificador.
2. Cree una enumeración de estos elementos coloreables.
3. Asocie los tipos de token devueltos desde un analizador o analizador con los valores enumerados.

Por ejemplo, los valores que representan los tipos de token podrían ser los mismos valores en la enumeración de elementos coloreables personalizados.

4. En la implementación del método en el [ColorizeLine IVsColorizer](#) objeto, rellene la lista de atributos con los valores de la enumeración de elementos coloreables personalizados correspondientes a los tipos de token

devueltos desde el analizador o el analizador.

5. En la misma clase [IVsLanguageInfo](#) que implementa [IVsProvideColorableItems](#) la interfaz, [GetItemCount](#) implemente [GetColorableItem](#) la interfaz y sus dos métodos, y .
6. Implemente la interfaz [IVsColorableItem](#).
7. Si desea admitir valores de color de 24 [IVsHiColorItem](#) bits o altos, implemente también la interfaz.
8. En el objeto de servicio de [IVsColorableItem](#) lenguaje, cree una lista que contenga los objetos, una para cada elemento coloreable que el analizador o el analizador pueda identificar.

Puede tener acceso a cada elemento de la lista mediante el valor correspondiente de la enumeración de elementos coloreables personalizados. Utilice los valores de enumeración como un índice en la lista. Nunca se tiene acceso al primer elemento de la lista, porque corresponde al estilo de texto predeterminado que Visual Studio siempre se controla a sí mismo. Puede compensar esto insertando un elemento coloreable de marcador de posición al principio de la lista.

9. En la implementación del [GetItemCount](#) método, devuelva el número de elementos de la lista de elementos coloreables personalizados.
10. En la implementación del [GetColorableItem](#) método, devuelva el elemento coloreable solicitado de la lista.

Para obtener un ejemplo [IVsColorableItem](#) de [IVsHiColorItem](#) cómo implementar [IVsHiColorItem](#) las interfaces y, consulte .

Vea también

- [Modelo de un servicio de lenguaje heredado](#)
- [Coloración de sintaxis en editores personalizados](#)
- [Coloreación de sintaxis en un servicio de lenguaje heredado](#)
- [Implementar coloración de sintaxis](#)
- [Cómo: Usar elementos coloreables incorporados](#)

Formato automático en un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Con el formato automático, un servicio de lenguaje inserta automáticamente un fragmento de código cuando un usuario comienza a escribir una construcción de código conocida.

Comportamiento de formato automático

Por ejemplo, al escribir *if*, el servicio de lenguaje inserta automáticamente llaves coincidentes, o si presiona la tecla ENTRAR, el servicio de lenguaje fuerza el punto de inserción de la nueva línea al nivel de sangría adecuado, dependiendo de si la línea anterior abre un nuevo ámbito.

El filtro de comandos utilizado para el resto del servicio de lenguaje también se puede utilizar para el formato automático. También puede resaltar llaves coincidentes llamando a [HighlightMatchingBrace](#).

Vea también

- [Desarrollar un servicio de lenguaje heredado](#)

Información de parámetros en un servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

La información sobre parámetros de IntelliSense proporciona a los usuarios sugerencias sobre dónde se encuentran en una construcción de lenguaje.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Ampliación del editor](#) y los servicios de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Cómo funcionan las descripciones emergentes de información de parámetros

Al escribir una instrucción en el editor, el VSPackage muestra una pequeña ventana de información sobre herramientas que contiene la definición de la instrucción que se escribe. Por ejemplo, si escribe una instrucción Microsoft Foundation `pMainFrame ->UpdateWindow` Classes (MFC) (como) y presiona la tecla de paréntesis `UpdateWindow` de apertura para comenzar a enumerar los parámetros, aparece una sugerencia de método que muestra la definición del método.

Las descripciones emergentes de información de parámetros se suelen utilizar junto con la finalización de instrucciones. Son más útiles para los idiomas que tienen parámetros u otra información formateada después del nombre del método o la palabra clave.

La información de parámetros se inicia mediante el servicio de lenguaje a través de la interceptación de comandos. Para interceptar caracteres de usuario, [IOleCommandTarget](#) el objeto de servicio de [IOleCommandTarget](#) lenguaje debe [AddCommandFilter](#) implementar la [IVsTextView](#) interfaz y pasar la vista de texto un puntero a la implementación, llamando al método en la interfaz. El filtro de comandos intercepta los comandos que escriba en la ventana de código. Supervise la información del comando para saber cuándo mostrar información de parámetros al usuario. Puede utilizar el mismo filtro de comandos para la finalización de instrucciones, marcadores de error, etc.

Al escribir una palabra clave para la que el servicio [IVsMethodTipWindow](#) de lenguaje [UpdateTipWindow](#) puede proporcionar [IVsTextView](#) sugerencias, el servicio de lenguaje crea un objeto y llama al método en la interfaz para notificar al IDE para mostrar una sugerencia. Cree [IVsMethodTipWindow](#) el `VSLocalCreateInstance` objeto utilizando y `CLSID_VsMethodTipWindow` especificando la coclase . `VSLocalCreateInstance` es una función definida en el archivo `QueryService` de encabezado vsdoc.h que llama al registro local y llama `CreateInstance` a este objeto para el `CLSID_VsMethodTipWindow` archivo .

Proporcionar una sugerencia de método

Para proporcionar una sugerencia [SetMethodData](#) de método, llame al método en la [IVsMethodTipWindow](#) interfaz, pasándole la implementación de la [IVsMethodData](#) interfaz.

Cuando [IVsMethodData](#) se invoca la clase, se llama a sus métodos en el siguiente orden:

- [GetContextStream](#)

Devuelve la posición y la longitud de los datos relevantes en el búfer de texto actual. Esto indica al IDE que no oscurezca esos datos con la ventana de información sobre herramientas.

- [GetCurMethod](#)

Devuelve el número de método (índice de base cero) que desea que se muestre inicialmente. Por ejemplo, si devuelve cero, se presenta inicialmente el primer método sobrecargado.

- [GetOverloadCount](#)

Devuelve el número de métodos sobrecargados que son aplicables en el contexto actual. Si devuelve un valor mayor que 1 para este método, la vista de texto muestra las flechas arriba y abajo. Si hace clic en la flecha [NextMethod](#) hacia abajo, el IDE llama al método. Si hace clic en la flecha [PrevMethod](#) hacia arriba, el IDE llama al método.

- [GetMethodText](#)

El texto de la información de parámetros [GetMethodText](#) información [GetParameterText](#) sobre herramientas se construye durante varias llamadas a los métodos y.

- [GetParameterCount](#)

Devuelve el número de parámetros que se mostrarán en el método.

- [GetParameterText](#)

Si devuelve un número de método correspondiente a la sobrecarga que desea mostrar, [UpdateView](#) se llama a este método, seguido de una llamada al método.

- [UpdateView](#)

Informa al servicio de lenguaje para actualizar el editor cuando se muestra una sugerencia de método. En [UpdateView](#) el método, llame a lo siguiente:

```
<pTxWin> ->UpdateTipWindow(<pTip>, UTW_CONTENTCHANGED | UTW_CONTEXTCHANGED).
```

- [OnDismiss](#)

Recibirá una llamada [OnDismiss](#) al método al cerrar la ventana de sugerencia del método.

Finalización de instrucciones en un servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

La finalización de instrucciones es el proceso mediante el cual el servicio de lenguaje ayuda a los usuarios a finalizar una palabra clave de idioma o elemento que han comenzado a escribir en el editor principal. En este tema se describe cómo funciona la finalización de instrucciones y cómo implementarla en el servicio de lenguaje.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar la finalización de instrucciones, vea [Tutorial: Mostrar finalización](#) de instrucciones .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Implementación de la finalización de la declaración

En el editor principal, la finalización de instrucciones activa una interfaz de usuario especial que le ayuda de forma interactiva a escribir código de forma más fácil y rápida. La finalización de instrucciones ayuda al mostrar objetos o clases pertinentes cuando son necesarios, lo que evita tener que recordar elementos específicos o tener que buscarlos en un tema de referencia de ayuda.

Para implementar la finalización de instrucciones, el idioma debe tener un desencadenador de finalización de instrucciones, que se puede analizar. Por ejemplo, Visual Basic utiliza un operador de Visual C++ punto (.), mientras que utiliza un operador de flecha (->). Un servicio de lenguaje puede usar más de un desencadenador para iniciar la finalización de instrucciones. Estos desencadenadores se programan en el filtro de comandos.

Filtros y disparadores de comandos

Los filtros de comandos interceptan las apariciones del desencadenador o los desencadenadores. Para agregar el filtro de comandos a la vista, implemente la [IOleCommandTarget](#) interfaz y adjúntela a la vista llamando al [AddCommandFilter](#) método. Puede usar el mismo [IOleCommandTarget](#) filtro de comandos () para todos los aspectos del servicio de lenguaje, como la finalización de instrucciones, marcadores de error y sugerencias de método. Para obtener más información, consulte [Interceptación de comandos](#) de servicio de lenguaje heredado .

Cuando se introduce el desencadenador en el editor, específicamente el [UpdateCompletionStatus](#) búfer de texto, el servicio de lenguaje llama al método. Esto hace que el editor para que aparece la interfaz de usuario para que el usuario puede elegir entre los candidatos de finalización de la instrucción. Este método requiere que [IVsCompletionSet](#) implemente [UpdateCompletionFlags](#) y las marcas como parámetros. La lista de elementos de finalización aparece en un cuadro de lista de desplazamiento. A medida que el usuario continúa escribiendo, la selección dentro del cuadro de lista se actualiza para reflejar la coincidencia más cercana a los caracteres más recientes escritos. El editor principal implementa la interfaz de usuario para [IVsCompletionSet](#) la finalización de instrucciones, pero el servicio de lenguaje debe implementar la interfaz para definir un conjunto de elementos de finalización de candidatos para la instrucción.

Vea también

- [Intercepción de comandos del servicio de lenguaje heredado](#)

Esquematización y texto oculto en un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

La esquematización permite contraer una secuencia de líneas de texto en una sola línea. Por ejemplo, en C++ todos los métodos se pueden contraer en una sola línea, mostrando solo la firma del método. El texto oculto es la secuencia de líneas de texto que se pueden mostrar u ocultar.

En esta sección

- [Compatibilidad de la esquematización en un servicio de lenguaje heredado](#)

Explica cómo implementar

- [Provisión de compatibilidad con texto oculto en un servicio de lenguaje heredado](#)

Explica el propósito de una región de texto oculto y proporciona instrucciones sobre cómo implementar una región de texto oculta.

- [Provisión de compatibilidad con esquematización ampliada en un servicio de lenguaje heredado](#)

Explica las dos opciones que amplían la compatibilidad de esquematización para el lenguaje más allá de admitir el comando *Contraer a definiciones*.

Cómo: Apoyar la esbozación en un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

La esquematización se utiliza para expandir o contraer diferentes regiones de texto. La forma en que se utiliza la esquematización se puede definir de manera diferente por diferentes idiomas. Para obtener más información, vea [Esquematización](#).

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar la esquematización, consulte [Tutorial: esquematización](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

A continuación se muestra cómo admitir este comando para el servicio de lenguaje.

Para apoyar la esbozación

1. Implementar [IVsOutliningCapableLanguage](#) en el objeto de servicio de lenguaje.
2. Llame [AddOutlineRegions](#) al objeto de sesión de esquematización actual para agregar nuevas regiones de esquema.

Programación sólida

Cuando un usuario selecciona Contraer a **definiciones** en el [CollapseToDefinitions](#) menú **Esquema**, el IDE llama al servicio de lenguaje.

Cuando se llama a este método, [IVsTextLines](#) el IDE pasa un puntero [IVsOutliningSession](#) (un puntero a un búfer de texto) y un (un puntero a la sesión de esquematización actual).

Puede llamar [AddOutlineRegions](#) al método para varias regiones de `rgOutLnReg` esquema especificando estas regiones en el parámetro. El `rgOutLnReg` parámetro [NewOutlineRegion](#) es una estructura. Este proceso le permite especificar diferentes características de la región oculta, como si una región determinada está expandida o contraída.

NOTE

Tenga cuidado de ocultar caracteres de nueva línea. El texto oculto debe extenderse desde el inicio de la primera línea hasta el último carácter de la última línea de una sección, dejando el carácter final de la nueva línea visible.

Vea también

- [Cómo: Proporcionar soporte de texto oculto en un servicio de lenguaje heredado](#)
- [Cómo: Proporcionar soporte de esquematización ampliada en un servicio de lenguaje heredado](#)

Cómo: Proporcionar soporte de texto oculto en un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede crear regiones de texto ocultas además de regiones de esquema. Las regiones de texto ocultas pueden estar controladas por el cliente o por el editor y se utilizan para ocultar una región de texto por completo. El editor muestra una región oculta como líneas horizontales. Un ejemplo de esto es la vista **Solo script** en el editor HTML.

Para implementar una región de texto oculto

1. Llame `QueryService` `SVsTextManager` para .

Esto devuelve un `IVsHiddenTextManager` puntero a .

2. Llamar `GetHiddenTextSession` , pasando un puntero para un búfer de texto determinado. Esto determina si ya existe una sesión de texto oculto para el búfer.
3. Si ya existe uno, no es necesario crear uno `IVsHiddenTextSession` y se devuelve un puntero al objeto existente. Utilice este puntero para enumerar y crear regiones de texto ocultas. De lo contrario, llame para crear una sesión de texto oculto para el búfer.

Se devuelve `IVsHiddenTextSession` un puntero al objeto.

NOTE

Al llamar `CreateHiddenTextSession` a , puede especificar un cliente `IVsHiddenTextClient` de texto oculto (es decir,). El cliente de texto oculto le notifica cuando el usuario expande o contrae texto oculto o esquematización.

4. Llame `AddHiddenRegions` para agregar una o más regiones de esquema nuevas a la vez, especificando la siguiente información en el `reHidReg` parámetro (`NewHiddenRegion`):
 - a. Especifique un `hrtConcealed` valor `iType` de en `NewHiddenRegion` el miembro de la estructura para indicar que está creando una región oculta, en lugar de una región de esquema.

NOTE

Cuando las regiones ocultas están ocultas, el editor muestra automáticamente las líneas alrededor de las regiones ocultas para indicar su presencia.

- b. Especifique si la región está controlada por el `dwBehavior` cliente o `NewHiddenRegion` por el editor en los miembros de la estructura. La implementación de esquematización inteligente puede contener una combinación de esquemas controlados por el editor y el cliente y regiones de texto ocultas.

Cómo: Proporcionar soporte de esquematización ampliada en un servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

Hay dos opciones para ampliar la compatibilidad de esquematización para su idioma más allá de admitir el comando **Contraer a definiciones**. Puede agregar regiones de esquema controladas por el editor y regiones de esquema controladas por el cliente.

Adición de regiones de esquema controladas por el editor

Use este enfoque para crear una región de esquema y, a continuación, permita que el editor controle si la región está expandida, contraída, etc. De las dos opciones para proporcionar soporte de esquematización, esta opción es la menos robusta. Para esta opción, se crea una nueva región de [AddOutlineRegions](#)esquema en un intervalo de texto especificado mediante `IVsHiddenTextSession`. Después de crear esta región, el editor controla su comportamiento. Use el siguiente procedimiento para implementar regiones de esquema controladas por el editor.

Implementar una región de esquema controlada por el editor

1. Llamar `QueryService` [SVsTextManager](#)

Esto devuelve un [IVsHiddenTextManager](#) puntero a `IVsHiddenTextManager`.

2. Llamar [GetHiddenTextSession](#), pasando un puntero para un búfer de texto determinado. Esto devuelve un [IVsHiddenTextSession](#) puntero al objeto para el búfer.
3. Llame [QueryInterface](#) [IVsHiddenTextSession](#) para un [IVsOutliningSession](#) puntero a `IVsOutliningSession`.
4. Llame [AddOutlineRegions](#) para agregar una o más regiones de esquema nuevas a la vez.

Este método permite especificar el intervalo de texto que se va a esbozar, si se quitan o conservan las regiones de esquema existentes y si la región de esquema se expande o contrae de forma predeterminada.

Agregar regiones de esquema controladas por el cliente

Utilice este enfoque para implementar la esquematización controlada por Visual C# el Visual Basic cliente (o inteligente) como la utilizada por los servicios de lenguaje y. Un servicio de lenguaje que administra su propia esquematización supervisa el contenido del búfer de texto con el fin de destruir las regiones de esquema antiguas cuando dejan de ser válidas y para crear nuevas regiones de esquema según sea necesario.

Implementar una región de esquema controlada por el cliente

1. Llame `QueryService` [SVsTextManager](#) para `SVsTextManager`. Esto devuelve un [IVsHiddenTextManager](#) puntero a `IVsHiddenTextManager`.
2. Llamar [GetHiddenTextSession](#), pasando un puntero para un búfer de texto determinado. Esto determina si ya existe una sesión de texto oculto para el búfer.
3. Si ya existe una sesión de texto, no es necesario [IVsHiddenTextSession](#) crear una y se devuelve un puntero al objeto existente. Utilice este puntero para enumerar y crear regiones de esquema. De lo contrario, llame [CreateHiddenTextSession](#) para crear una sesión de texto oculto para el búfer. Se devuelve [IVsHiddenTextSession](#) un puntero al objeto.

NOTE

Cuando se [CreateHiddenTextSession](#) llama a , puede especificar un [IVsHiddenTextClient](#) cliente de texto oculto (es decir, un objeto). Este cliente le notifica cuando el usuario expande o contrae un texto oculto o una región de esquema.

4. Parámetro [AddHiddenRegions](#) de estructura de llamada): `iType` especifique un [NewHiddenRegion](#) valor de [HIDDEN_REGION_TYPE](#) en el miembro de la estructura para indicar que está creando una región de esquema, en lugar de una región oculta. Especifique si la región está controlada por el `dwBehavior` cliente o [NewHiddenRegion](#) por el editor en el miembro de la estructura. La implementación de esquematización inteligente puede contener una combinación de regiones de esquema controladas por el editor y el cliente. Especifique el texto del banner que se muestra cuando se contrae `pszBanner` la región [NewHiddenRegion](#) de esquema, como "...", en el miembro de la estructura. El texto de banner predeterminado del editor para una región oculta es "...".

Compatibilidad del servicio de lenguaje con la depuración

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un servicio de lenguaje puede proporcionar [IVsLanguageDebugInfo](#) características que admiten un depurador a través de la interfaz. Estas características incluyen la validación de puntos de interrupción y el suministro de una lista de expresiones a la ventana **Automático**.

Sin embargo, debe tener un evaluador de expresiones para depurar el idioma. El evaluador de expresiones es responsable de evaluar expresiones para generar valores durante la depuración. Para obtener información sobre la implementación de evaluadores de expresiones CLR, consulte:

- [Evaluadores de expresiones CLR](#)
- [Ejemplo de evaluador de expresiones administradas](#)

Salida del compilador

El tipo de compilador determina lo que debe hacer para implementar la depuración para el lenguaje. Si el compilador tiene como destino el sistema operativo Windows y escribe un archivo .pdb, puede depurar programas con el motor de depuración de código nativo que está integrado en Visual Studio. Si el compilador genera el lenguaje intermedio de Microsoft (MSIL), puede depurar programas con el motor de depuración de código administrado, que también está integrado en Visual Studio. Si el compilador tiene como destino un sistema operativo propietario o un entorno de tiempo de ejecución diferente, debe escribir su propio motor de depuración.

Para obtener más información sobre cómo implementar la depuración para el lenguaje, vea [Introducción](#) en el SDK de depuración de Visual Studio.

Implementación de un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede usar clases en el marco de paquete administrado (MPF) para implementar un servicio de lenguaje heredado que admita una amplia variedad de características, como resaltado de sintaxis, coincidencia de llaves y finalización de IntelliSense.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar un servicio de lenguaje, vea [Extensiones de servicio de editor y lenguaje](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

En esta sección

- [Información general del servicio de lenguaje heredado](#)

Una visión general de las características del servicio de lenguaje que se admiten en MPF.

- [Implementación de un servicio de lenguaje heredado](#)

Describe lo que se requiere para implementar un servicio de lenguaje mediante MPF.

- [Registro de un servicio de lenguaje heredado](#)

Describe los pasos necesarios para registrar un servicio Visual Studio de lenguaje basado en MPF con .

- [Escáner y analizador del servicio de lenguaje heredado](#)

Describe los dos analizadores necesarios para implementar todas las características de un servicio de lenguaje mediante el MPF.

- [Tutorial: Creación de un servicio de lenguaje heredado](#)

Proporciona los pasos básicos necesarios para implementar un servicio de lenguaje MPF en un VSPackage.

- [Tutorial: Obtención de una lista de fragmentos de código instalados \(implementación heredada\)](#)

Muestra las técnicas de recuperación de una lista de fragmentos de código instalados.

- [Características del servicio de lenguaje heredado](#)

Proporciona vínculos a temas que detallan lo que se debe hacer para implementar todas las características de un servicio de lenguaje mediante MPF.

Información general del servicio de lenguaje heredado

06/04/2020 • 3 minutes to read • [Edit Online](#)

Un servicio de lenguaje proporciona compatibilidad Visual Studio con el editor que le permite implementar ciertas características. Las clases de servicio de lenguaje de Managed Package Framework (MPF) proporcionan compatibilidad completa con las características de uso frecuente y compatibilidad parcial con otras características.

Características totalmente compatibles en el MPF

Las clases de servicio de lenguaje MPF admiten las siguientes características:

- Resaltado de sintaxis
- esquematizar
- Comentar bloques de código
- Coincidencia de llaves
- Fragmentos de código
- Propiedades personalizadas del documento
- Información de parámetros de IntelliSense
- Información rápida de IntelliSense
- Finalización de miembros de IntelliSense
- Finalización de palabras de IntelliSense

Características parcialmente admitidas en el MPF

El MPF solo proporciona compatibilidad parcial con las siguientes características. Esto significa que debe implementar los métodos a los que llama el MPF.

- Reformatear el código. Proporcione el código que implementa el reformato.
- Validación de puntos de interrupción mediante la identificación de intervalos de código válidos. Proporcione el código que identifica los intervalos de código.
- Compatibilidad con la ventana **Autos** del depurador para mostrar variables. Proporcione el código que determina qué mostrar en la ventana.
- Compatibilidad con la barra de **navegación** para una navegación rápida entre tipos y miembros. Implemente y devuelva una clase auxiliar que rellene las listas en los cuadros combinados Barra de **navegación**.

Implementación

Debe completar varios pasos para implementar el propio servicio de lenguaje y las características del servicio de lenguaje que desea admitir para su idioma. Estos pasos se describen en los temas siguientes:

- [Implementación de un servicio de lenguaje heredado](#)

- [Registro de un servicio de lenguaje heredado](#)
- [Coloreado de sintaxis en un servicio de lenguaje heredado](#)
- [Coincidencia de llaves en un servicio de lenguaje heredado](#)
- [Esquematización en un servicio de lenguaje heredado](#)
- [Comentario de código en un servicio de lenguaje heredado](#)
- [Cambio de formato de código en un servicio de lenguaje heredado](#)
- [Propiedades de documento personalizadas en un servicio de lenguaje heredado](#)
- [Compatibilidad con fragmentos de código en un servicio de lenguaje heredado](#)
- [Compatibilidad con la barra de navegación en un servicio de lenguaje heredado](#)
- [Finalización de palabras en un servicio de lenguaje heredado](#)
- [Finalización de miembros en un servicio de lenguaje heredado](#)
- [Información de parámetros en un servicio de lenguaje heredado](#)
- [Información rápida en un servicio de lenguaje heredado](#)
- [Compatibilidad con la ventana Automático en un servicio de lenguaje heredado](#)
- [Validación de puntos de interrupción en un servicio de lenguaje heredado](#)

Vea también

- [Implementación de un servicio de lenguaje heredado](#)
- [Extensibilidad de servicio de lenguaje heredado](#)

Implementación de un servicio de lenguaje heredado

06/04/2020 • 19 minutes to read • [Edit Online](#)

Para implementar un servicio de lenguaje mediante el marco de paquete [LanguageService](#) administrado (MPF), debe derivar una clase de la clase e implementar los siguientes métodos y propiedades abstractas:

- El método [GetLanguagePreferences](#)
- El método [GetScanner](#)
- El método [ParseSource](#)
- La propiedad [Name](#)

Consulte las secciones correspondientes a continuación para obtener más información sobre la implementación de estos métodos y propiedades.

Para admitir características adicionales, el servicio de lenguaje puede tener que derivar una clase de una de las clases de servicio de lenguaje MPF; por ejemplo, para admitir comandos de menú [ViewFilter](#) adicionales, debe derivar una clase [ViewFilter](#) de la clase e invalidar varios de los métodos de control de comandos (consulte para obtener más información). La [LanguageService](#) clase proporciona una serie de métodos que se llaman para crear nuevas instancias de varias clases y se reemplaza el método de creación adecuado para proporcionar una instancia de la clase. Por ejemplo, debe invalidar [CreateViewFilter](#) el [LanguageService](#) método de la clase [ViewFilter](#) para devolver una instancia de su propia clase. Consulte la sección "Instantáneas de clases personalizadas" para obtener más detalles.

Su servicio de idiomas también puede proporcionar sus propios iconos, que se utilizan en muchos lugares. Por ejemplo, cuando se muestra una lista de finalización de IntelliSense, cada elemento de la lista puede tener un icono asociado, marcando el elemento como un método, clase, espacio de nombres, propiedad o lo que sea necesario para su idioma. Estos iconos se utilizan en todas las listas de IntelliSense, la barra de **navegación** y en la ventana de tareas **Lista** de errores. Consulte la sección "Imágenes del servicio de lenguaje" a continuación para obtener más información.

GetLanguagePreferences Método

El [GetLanguagePreferences](#) método siempre devuelve la [LanguagePreferences](#) misma instancia de una clase. Puede utilizar la [LanguagePreferences](#) clase base si no necesita ninguna preferencia adicional para su servicio de idiomas. Las clases de servicio de lenguaje MPF asumen la presencia de al menos la clase base.

[LanguagePreferences](#)

Ejemplo

En este ejemplo se [GetLanguagePreferences](#) muestra una implementación típica del método. En este ejemplo [LanguagePreferences](#) se utiliza la clase base.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        private LanguagePreferences m_preferences;

        public override LanguagePreferences GetLanguagePreferences()
        {
            if (m_preferences == null)
            {
                m_preferences = new LanguagePreferences(this.Site,
                                                         typeof(TestLanguageService).GUID,
                                                         this.Name );

                m_preferences.Init();
            }
            return m_preferences;
        }
    }
}

```

GetScanner Método

Este método devuelve una [IScanner](#) instancia de un objeto que implementa un analizador o analizador orientado a líneas que se utiliza para obtener tokens y sus tipos y desencadenadores. Este escáner se [Colorizer](#) utiliza en la clase para la coloración, aunque el escáner también se puede utilizar para obtener tipos de token y desencadenadores como prelude de una operación de análisis más compleja. Debe proporcionar la clase que [IScanner](#) implementa la interfaz y [IScanner](#) debe implementar todos los métodos en la interfaz.

Ejemplo

En este ejemplo se [GetScanner](#) muestra una implementación típica del método. La `TestScanner` clase implementa [IScanner](#) la interfaz (no se muestra).

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        private TestScanner m_scanner;

        public override IScanner GetScanner(IVsTextLines buffer)
        {
            if (m_scanner == null)
            {
                m_scanner = new TestScanner(buffer);
            }
            return m_scanner;
        }
    }

    internal class TestScanner : IScanner
    {
        private IVsTextBuffer m_buffer;
        string m_source;

        public TestScanner(IVsTextBuffer buffer)
        {
            m_buffer = buffer;
        }

        bool IScanner.ScanTokenAndProvideInfoAboutIt(TokenInfo tokenInfo, ref int state)
        {
            tokenInfo.Type = TokenType.Unknown;
            tokenInfo.Color = TokenColor.Text;
            return true;
        }

        void IScanner.SetSource(string source, int offset)
        {
            m_source = source.Substring(offset);
        }
    }
}

```

ParseSource Método

Analiza el archivo de origen en función de varias razones diferentes. A este método [ParseRequest](#) se le da un objeto que describe lo que se espera de una operación de análisis determinada. El [ParseSource](#) método invoca un analizador más complejo que determina la funcionalidad y el ámbito del token. El [ParseSource](#) método se utiliza en soporte para las operaciones de IntelliSense, así como la coincidencia de llaves. Incluso si no admite estas operaciones avanzadas, [AuthoringScope](#) debe devolver un objeto válido y eso [AuthoringScope](#) requiere crear una clase que implemente la interfaz e implemente todos los métodos en esa interfaz. Puede devolver valores null de [AuthoringScope](#) todos los métodos, pero el propio objeto no debe ser un valor nulo.

Ejemplo

En este ejemplo se [ParseSource](#) muestra una [AuthoringScope](#) implementación mínima del método y la clase, suficiente para permitir que el servicio de lenguaje compile y funcione sin admitir realmente ninguna de las características más avanzadas.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        public override AuthoringScope ParseSource(ParseRequest req)
        {
            return new TestAuthoringScope();
        }
    }

    internal class TestAuthoringScope : AuthoringScope
    {
        public override string GetDataTipText(int line, int col, out TextSpan span)
        {
            span = new TextSpan();
            return null;
        }

        public override Declarations GetDeclarations(IVsTextView view,
                                                    int line,
                                                    int col,
                                                    TokenInfo info,
                                                    ParseReason reason)
        {
            return null;
        }

        public override string Goto(VSConstants.VSStd97CmdID cmd, IVsTextView textView, int line, int col, out
TextSpan span)
        {
            span = new TextSpan();
            return null;
        }

        public override Methods GetMethods(int line, int col, string name)
        {
            return null;
        }
    }
}

```

Propiedad Name

Esta propiedad devuelve el nombre del servicio de lenguaje. Debe tener el mismo nombre que se indica cuando se registró el servicio de lenguaje. Este nombre se utiliza en varios lugares, el [LanguagePreferences](#) más destacado de los cuales es la clase donde se utiliza el nombre para acceder al registro. El nombre devuelto por esta propiedad no debe estar localizado, ya que se utiliza en el registro para la entrada del Registro y los nombres de clave.

Ejemplo

En este ejemplo se [Name](#) muestra una posible implementación de la propiedad. Tenga en cuenta que el nombre aquí está codificado de forma rígida: el nombre real debe obtenerse de un archivo de recursos para que se pueda utilizar en el registro de un servicio de lenguaje (consulte Registro de un servicio de [lenguaje heredado](#)).


```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        public override string Name
        {
            get { return "Test Language"; }
        }
    }
}
```

Creación de instancias de clases personalizadas

Los siguientes métodos en las clases especificadas se pueden invalidar para proporcionar instancias de sus propias versiones de cada clase.

En la clase **LanguageService**

MÉTODO	CLASE DEVUELTA	DESCRIPCIÓN
CreateCodeWindowManager	CodeWindowManager	Para admitir adiciones personalizadas a la vista de texto.
CreateDocumentProperties	DocumentProperties	Para admitir propiedades de documento personalizadas.
CreateDropDownHelper	TypeAndMemberDropdownBars	Para admitir la barra de navegación .
CreateExpansionFunction	ExpansionFunction	Para admitir funciones en plantillas de fragmento de código.
CreateExpansionProvider	ExpansionProvider	Para admitir fragmentos de código (normalmente no se invalida este método).
CreateParseRequest	ParseRequest	Para admitir la ParseRequest personalización de la estructura (este método normalmente no se invalida).
CreateSource	Source	Para admitir el formato de código fuente, especificar caracteres de comentario y personalizar las firmas de método.
CreateViewFilter	ViewFilter	Para admitir comandos de menú adicionales.
GetColorizer	Colorizer	Para admitir el resaltado de sintaxis (este método normalmente no se invalida).

MÉTODO	CLASE DEVUELTA	DESCRIPCIÓN
GetLanguagePreferences	LanguagePreferences	Para admitir el acceso a las preferencias de idioma. Este método debe implementarse, pero puede devolver una instancia de la clase base.
GetScanner	IScanner	Proporcionar un analizador utilizado para identificar tipos de tokens en una línea. Este método debe implementarse y IScanner debe derivarse de.
ParseSource	AuthoringScope	Proporcionar un analizador utilizado para identificar la funcionalidad y el ámbito en todo un archivo de origen. Este método debe implementarse y debe devolver una AuthoringScope instancia de la versión de la clase. Si todo lo que desea admitir es resaltado IScanner de sintaxis GetScanner (que requiere el analizador devuelto desde el AuthoringScope método), no puede hacer nada en este método que no sea devolver una versión de la clase cuyos métodos devuelven valores null.

En la clase Source

MÉTODO	CLASE DEVUELTA	DESCRIPCIÓN
CreateCompletionSet	CompletionSet	Para personalizar la visualización de las listas de finalización de IntelliSense (este método normalmente no se invalida).
CreateErrorTaskItem	DocumentTask	Para admitir marcadores en la lista de tareas Lista de errores; específicamente, soporte para características más allá de abrir el archivo y saltar a la línea que causó el error.
CreateMethodData	MethodData	Para personalizar la visualización de Información de parámetros de IntelliSense Info ToolTips.
GetCommentFormat	CommentInfo	Para admitir el código de comentarios.
CreateAuthoringSink	AuthoringSink	Para recopilar información durante la operación de análisis.

En la clase AuthoringScope

MÉTODO	CLASE DEVUELTA	DESCRIPCIÓN

MÉTODO	CLASE DEVUELTA	DESCRIPCIÓN
GetDeclarations	Declarations	Proporciona una lista de declaraciones como miembros o tipos. Este método debe implementarse, pero puede devolver un valor nulo. Si este método devuelve un objeto válido, el objeto Declarations debe ser una instancia de la versión de la clase.
GetMethods	Methods	Proporciona una lista de firmas de método para un contexto determinado. Este método debe implementarse, pero puede devolver un valor nulo. Si este método devuelve un objeto válido, el objeto Methods debe ser una instancia de la versión de la clase.

Imágenes del servicio de lenguaje

Para proporcionar una lista de iconos que se usarán en todo el servicio de lenguaje, invalide el [GetImageList](#) método de la [LanguageService](#) clase y devuelva un [ImageList](#) que contenga los iconos. La [LanguageService](#) clase base carga un conjunto predeterminado de iconos. Puesto que especifica el índice de imagen exacto en aquellos lugares que necesitan iconos, la forma de organizar su propia lista de imágenes depende totalmente de usted.

Imágenes utilizadas en las listas de finalización de IntelliSense

Para las listas de finalización de IntelliSense, [GetGlyph](#) se especifica [Declarations](#) el índice de imagen para cada elemento del método de la clase, que debe invalidar si desea proporcionar un índice de imagen. El valor devuelto [GetGlyph](#) por el método es un [CompletionSet](#) índice en la lista de imágenes proporcionada [LanguageService](#) al constructor de clase y que [CompletionSet](#) es la [CreateCompletionSet](#) misma [Source](#) lista de imágenes devuelta por el [GetImageList](#) método de la clase (puede cambiar la lista de imágenes que se usará para el if invalidar el método en la clase para proporcionar una lista de imágenes diferente).

Imágenes utilizadas en la barra de navegación

La barra de **navegación** muestra listas de tipos y miembros y se utiliza para la navegación rápida puede mostrar iconos. Estos iconos se obtienen [GetImageList](#) del [LanguageService](#) método de la clase y no se pueden invalidar específicamente para la barra de **navegación**. Los índices utilizados para cada elemento de los cuadros combinados se especifican [OnSynchronizeDropdowns](#) cuando las [TypeAndMemberDropDownBars](#) listas que representan los cuadros combinados se rellenan en el método de la clase (consulte Compatibilidad con la barra de navegación en un servicio de [lenguaje heredado](#)). Estos índices de imagen se obtienen de alguna manera [Declarations](#) del analizador, normalmente a través de la versión de la clase. La forma en que se obtienen los índices depende totalmente de usted.

Imágenes utilizadas en la ventana de tareas Lista de errores

Siempre [ParseSource](#) que el analizador de métodos (consulte [Analizador y analizador](#) de legacy [AddError](#) Language Service [AuthoringSink](#)) encuentra un error y pasa ese error al método de la clase, el error se notifica en la ventana de tareas **Lista de errores**. Un icono se puede asociar a cada elemento que aparece en la ventana de [GetImageList](#) tareas y [LanguageService](#) ese icono proviene de la misma lista de imágenes devuelta desde el método de la clase. El comportamiento predeterminado de las clases MPF es no mostrar una imagen con el mensaje de error. Sin embargo, puede invalidar este comportamiento [Source](#) derivando una [CreateErrorTaskItem](#) clase de la clase y reemplazando el método. En ese método, se [DocumentTask](#) crea un nuevo objeto. Antes de devolver ese objeto, [ImageIndex](#) puede [DocumentTask](#) utilizar la propiedad del objeto para establecer el índice de imagen. Esto se vería algo parecido al siguiente ejemplo. Tenga `TestIconImageIndex` en cuenta que es una enumeración que enumera todos los iconos y es específica de este ejemplo. Es posible que tenga una forma diferente de identificar iconos en su servicio de idiomas.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    class TestSource : Source
    {
        public override DocumentTask CreateErrorTaskItem(
            TextSpan span,
            string filename,
            string message,
            TaskPriority priority,
            TaskCategory category,
            MARKERTYPE markerType,
            TaskErrorCategory errorCategory)
        {
            DocumentTask taskItem = base.CreateErrorTaskItem(
                span,
                filename,
                message,
                priority,
                category,
                markerType,
                errorCategory);

            if (errorCategory == TaskErrorCategory.Error)
            {
                taskItem.ImageIndex = TestIconImageIndex.Error;
            }

            return taskItem;
        }
    }
}

```

La lista de imágenes predeterminadas para un servicio de lenguaje

La lista de imágenes predeterminada proporcionada con las clases de servicio de lenguaje MPF base contiene una serie de iconos asociados con los elementos de lenguaje más comunes. La mayor parte de estos iconos están dispuestos en conjuntos de seis variaciones, correspondientes a los conceptos de acceso de público, interno, amigo, protegido, privado y acceso directo. Por ejemplo, puede tener diferentes iconos para un método dependiendo de si es público, protegido o privado.

La enumeración siguiente especifica los nombres típicos para cada conjunto de elementos y especifica el índice asociado. Por ejemplo, en función de la enumeración, puede

`(int)IconImageIndex.Method + (int)IconImageIndex.AccessProtected` especificar el índice de imagen para un método protegido como `.` Puede cambiar los nombres de esta enumeración como desee.

```

public enum IconImageIndex
{
    // access types
    AccessPublic = 0,
    AccessInternal = 1,
    AccessFriend = 2,
    AccessProtected = 3,
    AccessPrivate = 4,
    AccessShortcut = 5,

    Base = 6,
    // Each of the following icon type has 6 versions,
    //corresponding to the access types
    Class = Base + 0,
    Constant = Base + 1,

```

```

Delegate = Base + 2,
Enumeration = Base + 3,
EnumMember = Base + 4,
Event = Base + 5,
Exception = Base + 6,
Field = Base + 7,
Interface = Base + 8,
Macro = Base + 9,
Map = Base + 10,
MapItem = Base + 11,
Method = Base + 12,
OverloadedMethod = Base + 13,
Module = Base + 14,
Namespace = Base + 15,
Operator = Base + 16,
Property = Base + 17,
Struct = Base + 18,
Template = Base + 19,
Typedef = Base + 20,
Type = Base + 21,
Union = Base + 22,
Variable = Base + 23,
ValueType = Base + 24,
Intrinsic = Base + 25,
JavaMethod = Base + 26,
JavaField = Base + 27,
JavaClass = Base + 28,
JavaNamespace = Base + 29,
JavaInterface = Base + 30,
// Miscellaneous icons with one icon for each type.
Error = 187,
GreyedClass = 188,
GreyedPrivateMethod = 189,
GreyedProtectedMethod = 190,
GreyedPublicMethod = 191,
BrowseResourceFile = 192,
Reference = 193,
Library = 194,
VBProject = 195,
VBWebProject = 196,
CSProject = 197,
CSWebProject = 198,
VB6Project = 199,
CPlusProject = 200,
Form = 201,
OpenFolder = 202,
ClosedFolder = 203,
Arrow = 204,
CSClass = 205,
Snippet = 206,
Keyword = 207,
Info = 208,
CallBrowserCall = 209,
CallBrowserCallRecursive = 210,
XMLEditor = 211,
VJProject = 212,
VJClass = 213,
ForwardedType = 214,
CallsTo = 215,
CallsFrom = 216,
Warning = 217,
}

```

Vea también

- [Implementación de un servicio de lenguaje heredado](#)

- [Información general del servicio de lenguaje heredado](#)
- [Registro de un servicio de lenguaje heredado](#)
- [Escáner y analizador del servicio de lenguaje heredado](#)

Registro de un servicio de lenguaje heredado

06/04/2020 • 7 minutes to read • [Edit Online](#)

En el marco de paquete administrado (MPF), el servicio de lenguaje es ofrecido Visual Studio por un VSPackage (consulte [VSPackages](#)) y se registra con agregando claves y entradas del Registro. Este proceso de registro se realiza en parte durante la instalación y en parte en tiempo de ejecución.

Registrar el servicio de lenguaje mediante atributos

Los siguientes atributos se utilizan para registrar un servicio de lenguaje.

- [ProvideServiceAttribute](#)
- [ProvideLanguageServiceAttribute](#)
- [ProvideLanguageExtensionAttribute](#)
- [ProvideLanguageCodeExpansionAttribute](#)
- [ProvideLanguageEditorOptionPageAttribute](#)

Estos atributos se explican a continuación

ProvideServiceAttribute

Este atributo registra el servicio de lenguaje como un servicio.

Ejemplo

```
using Microsoft.VisualStudio.Shell;

namespace TestLanguagePackage
{
    [ProvideServiceAttribute(typeof(TestLanguageService),
                           ServiceName = "Test Language Service")]

    public class TestLanguagePackage : Package
    {
    }
}
```

ProvideLanguageServiceAttribute

Este atributo registra el servicio de lenguaje específicamente como un servicio de lenguaje. Le permite establecer opciones que especifican las características que ofrece su servicio de lenguaje. En el ejemplo se muestra un subconjunto de las opciones que puede proporcionar un servicio de lenguaje. Para ver el conjunto completo [ProvideLanguageServiceAttribute](#) de opciones de servicio de lenguaje, consulte .

Ejemplo

```

using Microsoft.VisualStudio.Shell;

namespace TestLanguagePackage
{
    [ProvideLanguageServiceAttribute(typeof(TestLanguageService),
        "Test Language",
        106,           // resource ID of localized language name
        CodeSense = true,           // Supports IntelliSense
        RequestStockColors = false, // Supplies custom colors
        EnableCommenting = true,    // Supports commenting out code
        EnableAsyncCompletion = true // Supports background parsing
    )]

    public class TestLanguagePackage : Package
    {
    }
}

```

ProvideLanguageExtensionAttribute

Este atributo asocia el servicio de lenguaje con una extensión de archivo. Cada vez que se carga un archivo con esa extensión, en cualquier proyecto, el servicio de lenguaje se inicia y se utiliza para mostrar el contenido del archivo.

Ejemplo

```

using Microsoft.VisualStudio.Shell;

namespace TestLanguagePackage
{
    [ProvideLanguageExtensionAttribute(typeof(TestLanguageService),
        ".Testtext")]

    public class TestLanguagePackage : Package
    {
    }
}

```

ProvideLanguageCodeExpansionAttribute

Este atributo registra una ubicación desde la que se obtienen plantillas de expansión de código o de fragmento de código. Esta información la utiliza el Explorador de **fragmentos** de código y el editor cuando se inserta un fragmento de código en el archivo de origen.

Ejemplo


```

using Microsoft.VisualStudio.Shell;

namespace TestLanguagePackage
{
    [ProvideLanguageCodeExpansionAttribute(
        typeof(TestLanguageService),
        "Test Language", // Name of language used as registry key.
        106,             // Resource ID of localized name of language service.
        "testlanguage", // language key used in snippet templates.
        @"%InstallRoot%\Test Language\SnippetsIndex.xml", // Path to snippets index
        SearchPaths = @"%InstallRoot%\Test Language\Snippets\%LCID%\Snippets;" +
                      @"%TestDocs%\Code Snippets\Test Language\Test Code Snippets"
    )]

    public class TestLanguagePackage : Package
    {
    }
}

```

ProvideLanguageEditorOptionPageAttribute

Este atributo registra una página de propiedades que se mostrará en el cuadro de diálogo **Opciones** en la categoría Editor de **texto**. Utilice uno de estos atributos para cada página que se mostrará para el servicio de lenguaje. Si necesita organizar las páginas en una estructura de árbol, utilice atributos adicionales para definir cada nodo del árbol.

Ejemplo

En este ejemplo se muestran dos páginas de propiedades, **Opciones** y **Sangría**, y un nodo que contiene la segunda página de propiedades.

```

using Microsoft.VisualStudio.Shell;

namespace TestLanguagePackage
{
    [ProvideLanguageEditorOptionPageAttribute(
        "Test Language", // Registry key name for language
        "Options",       // Registry key name for property page
        "#242",          // Localized name of property page
        OptionPageGuid = "{A2FE74E1-FFFF-3311-4342-123052450768}" // GUID of property page
    )]

    [ProvideLanguageEditorOptionPageAttribute(
        "Test Language", // Registry key name for language
        "Advanced",       // Registry key name for node
        "#243",          // Localized name of node
    )]

    [ProvideLanguageEditorOptionPageAttribute(
        "Test Language", // Registry key name for language
        @"Advanced\Indenting", // Registry key name for property page
        "#244",          // Localized name of property page
        OptionPageGuid = "{A2FE74E2-FFFF-3311-4342-123052450768}" // GUID of property page
    )]

    public class TestLanguagePackage : Package
    {
    }
}

```

Proffer the Language Service en tiempo de ejecución

Cuando se carga el paquete Visual Studio de idioma, debe indicar que el servicio de idioma está listo. Para ello, profunde el servicio. Esto se hace [Initialize](#) en el método. Además, debe iniciar un temporizador que llame al

servicio de lenguaje durante los períodos de inactividad para que se pueda realizar el análisis en segundo plano. Este temporizador de inactividad también se utiliza para actualizar [DocumentProperties](#) las propiedades del documento si ha implementado alguna a través de la clase. Para admitir un temporizador, el paquete [IOleComponent](#) debe implementar [FDIdle](#) la interfaz (solo es necesario implementar completamente el método; los métodos restantes pueden devolver valores predeterminados).

Ejemplo

En este ejemplo se muestra un enfoque típico para ofrecer un servicio y proporcionar un temporizador inactivo.

```
using System;
using System.Runtime.InteropServices;
using System.ComponentModel.Design;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.OLE.Interop;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    [Microsoft.VisualStudio.Shell.ProvideService(typeof(TestLanguageService))]
    [Microsoft.VisualStudio.Shell.ProvideLanguageExtension(typeof(TestLanguageService), ".testtext")]
    [Microsoft.VisualStudio.Shell.ProvideLanguageService(typeof(TestLanguageService), "Test Language", 0,
        AutoOutlining = true,
        EnableCommenting = true,
        MatchBraces = true,
        ShowMatchingBrace = true)]
    [Guid("00000000-0000-0000-0000-000000000000")] //provide a unique GUID for the package

    public class TestLanguagePackage : Package, IOleComponent
    {
        private uint m_componentID;

        protected override void Initialize()
        {
            base.Initialize(); // required

            // Proffer the service.
            IServiceContainer serviceContainer = this as IServiceContainer;
            TestLanguageService langService = new TestLanguageService();
            langService.SetSite(this);
            serviceContainer.AddService(typeof(TestLanguageService),
                langService,
                true);

            // Register a timer to call our language service during
            // idle periods.
            IOleComponentManager mgr = GetService(typeof(SOleComponentManager))
                as IOleComponentManager;
            if (m_componentID == 0 && mgr != null)
            {
                OLECRINFO[] crinfo = new OLECRINFO[1];
                crinfo[0].cbSize = (uint)Marshal.SizeOf(typeof(OLECRINFO));
                crinfo[0].grfcrcf = (uint)_OLECRF.olecrfNeedIdleTime |
                    (uint)_OLECRF.olecrfNeedPeriodicIdleTime;
                crinfo[0].grfcadvf = (uint)_OLECADVF.olecadvfModal |
                    (uint)_OLECADVF.olecadvfRedrawOff |
                    (uint)_OLECADVF.olecadvfWarningsOff;
                crinfo[0].uIdleTimeInterval = 1000;
                int hr = mgr.FRegisterComponent(this, crinfo, out m_componentID);
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (m_componentID != 0)
            {
                mgr.FUnregisterComponent(m_componentID);
            }
        }
    }
}
```

```

    {
        IOleComponentManager mgr = GetService(typeof(SOleComponentManager))
                                   as IOleComponentManager;

        if (mgr != null)
        {
            int hr = mgr.FRevokeComponent(m_componentID);
        }
        m_componentID = 0;
    }

    base.Dispose(disposing);
}

#region IOleComponent Members

public int FDoIdle(uint grfIdle)
{
    bool bPeriodic = (grfIdle & (uint)_OLEIDLEF.oleidlefPeriodic) != 0;
    // Use typeof(TestLanguageService) because we need to
    // reference the GUID for our language service.
    LanguageService service = GetService(typeof(TestLanguageService))
                               as LanguageService;

    if (service != null)
    {
        service.OnIdle(bPeriodic);
    }
    return 0;
}

public int FContinueMessageLoop(uint uReason,
                                IntPtr pvLoopData,
                                MSG[] pMsgPeeked)
{
    return 1;
}

public int FPreTranslateMessage(MSG[] pMsg)
{
    return 0;
}

public int FQueryTerminate(int fPromptUser)
{
    return 1;
}

public int FReserved1(uint dwReserved,
                      uint message,
                      IntPtr wParam,
                      IntPtr lParam)
{
    return 1;
}

public IntPtr HwndGetWindow(uint dwWhich, uint dwReserved)
{
    return IntPtr.Zero;
}

public void OnActivationChange(IOleComponent pic,
                              int fSameComponent,
                              OLECRINFO[] pcrinfo,
                              int fHostIsActivating,
                              OLECHOSTINFO[] pchostinfo,
                              uint dwReserved)
{
}

public void OnAppActivate(int fActive, uint dwOtherThreadID)

```

```
{  
}  
  
public void OnEnterState(uint uStateID, int fEnter)  
{  
}  
  
public void OnLoseActivation()  
{  
}  
  
public void Terminate()  
{  
}  
  
#endregion  
}  
}
```

Escáner y analizador del servicio de lenguaje heredado

06/04/2020 • 19 minutes to read • [Edit Online](#)

El analizador es el corazón del servicio de idiomas. Las clases de idioma de Managed Package Framework (MPF) requieren un analizador de idioma para seleccionar información sobre el código que se muestra. Un analizador separa el texto en tokens léxicos y, a continuación, identifica esos tokens por tipo y funcionalidad.

Discusión

A continuación se muestra un método de C.

```
namespace MyNamespace
{
    class MyClass
    {
        public void MyFunction(int arg1)
        {
            int var1 = arg1;
        }
    }
}
```

En este ejemplo, los tokens son las palabras y los marcadores de puntuación. Los tipos de tokens son los siguientes.

NOMBRE DEL TOKEN	TIPO DE TOKEN
espacio de nombres, clase, público, vacío, int	palabra clave
=	operator
{ } () ;	delimiter
MyNamespace, MyClass, MyFunction, arg1, var1	identificador
MyNamespace	espacio de nombres
MyClass	clase
MyFunction	method
arg1	parámetro
var1	variable local

El rol del analizador es identificar los tokens. Algunos tokens pueden tener más de un tipo. Una vez que el analizador ha identificado los tokens, el servicio de lenguaje puede usar la información para proporcionar características útiles, como resaltado de sintaxis, coincidencia de llaves y las operaciones de IntelliSense.

Tipos de analizadores

Un analizador de servicio de lenguaje no es lo mismo que un analizador utilizado como parte de un compilador. Sin embargo, este tipo de analizador debe usar un analizador y un analizador, de la misma manera que un analizador del compilador.

- Un analizador se utiliza para identificar tipos de tokens. Esta información se usa para resaltar la sintaxis y para identificar rápidamente los tipos de token que pueden desencadenar otras operaciones, por ejemplo, la concordancia de llaves. Este escáner está representado [IScanner](#) por la interfaz.
- Un analizador se utiliza para describir las funciones y el ámbito de los tokens. Esta información se utiliza en las operaciones de IntelliSense para identificar elementos de lenguaje, como métodos, variables, parámetros y declaraciones, y para proporcionar listas de miembros y firmas de método en función del contexto. Este analizador también se utiliza para localizar pares de elementos de lenguaje coincidentes, como llaves y paréntesis. Se accede a este [ParseSource](#) analizador [LanguageService](#) a través del método de la clase.

La forma de implementar un analizador y un analizador para su servicio de lenguaje depende de usted. Hay varios recursos disponibles que describen cómo funcionan los analizadores y cómo escribir su propio analizador. Además, hay varios productos gratuitos y comerciales disponibles que ayudan a crear un analizador.

El analizador ParseSource

A diferencia de un analizador que se usa como parte de un compilador (donde los tokens se convierten en alguna forma de código ejecutable), se puede llamar a un analizador de servicio de lenguaje por muchas razones diferentes y en muchos contextos diferentes. La forma de implementar [ParseSource](#) este [LanguageService](#) enfoque en el método de la clase depende de usted. Es importante tener en cuenta [ParseSource](#) que se puede llamar al método en un subproceso en segundo plano.

Caution

La [ParseRequest](#) estructura contiene una [IVsTextView](#) referencia al objeto. Este [IVsTextView](#) objeto no se puede utilizar en el subproceso en segundo plano. De hecho, muchas de las clases MPF base no se pueden utilizar en el subproceso en segundo plano. Estas incluyen [Source ViewFilter](#) las [CodeWindowManager](#) clases , , y cualquier otra clase que se comunique directa o indirectamente con la vista.

Este analizador normalmente analiza todo el archivo de origen la primera vez [ParseReason](#) que se llama o cuando se proporciona el valor de motivo de análisis. Las llamadas [ParseSource](#) posteriores al método controlan una pequeña parte del código analizado y se pueden ejecutar mucho más rápidamente mediante los resultados de la operación de análisis completo anterior. El [ParseSource](#) método comunica los resultados de la [AuthoringSink](#) operación de análisis a través de los objetos y. [AuthoringScope](#) El [AuthoringSink](#) objeto se utiliza para recopilar información para un motivo de análisis específico, por ejemplo, información sobre los intervalos de llaves coincidentes o firmas de método que tienen listas de parámetros. Proporciona [AuthoringScope](#) colecciones de declaraciones y firmas de método y también compatibilidad con la opción de edición avanzada [Ira](#) ([Ir a definición](#), [Ir a declaración](#), [Ir a referencia](#)).

El escáner IScanner

También debe implementar un analizador [IScanner](#) que implemente . Sin embargo, dado que este analizador funciona línea [Colorizer](#) por línea a través de la clase, normalmente es más fácil de implementar. Al principio de cada línea, el [Colorizer](#) MPF proporciona a la clase un valor para utilizarlo como una variable de estado que se pasa al analizador. Al final de cada línea, el analizador devuelve la variable de estado actualizada. El MPF almacena en caché esta información de estado para cada línea para que el analizador pueda comenzar a analizar desde cualquier línea sin tener que iniciarse al principio del archivo de origen. Este escaneo rápido de una sola línea permite al editor proporcionar retroalimentación rápida al usuario.

Análisis de llaves a juego

En este ejemplo se muestra el flujo de control para hacer coincidir una llave de cierre que el usuario ha escrito. En este proceso, el analizador que se utiliza para la coloración también se utiliza para determinar el tipo de token y si el token puede desencadenar una operación de coincidencia. Si se encuentra el [ParseSource](#) desencadenador, se llama al método para buscar la llave coincidente. Por último, se resaltan las dos llaves.

Aunque las llaves se utilizan en los nombres de desencadenadores y razones de análisis, este proceso no se limita a llaves reales. Se admite cualquier par de caracteres que se especifique que sea un par coincidente. Los ejemplos incluyen `< ( y )`, `y >`, y `y [ y ]`.

Supongamos que el servicio de lenguaje admite llaves coincidentes.

1. El usuario escribe una llave de cierre (`.`).
2. La llave se inserta en el cursor en el archivo de origen y el cursor se avanza por uno.
3. Se [OnCommand](#) llama [Source](#) al método de la clase con la llave de cierre con tipo.
4. El [OnCommand](#) método [GetTokenInfo](#) llama al [Source](#) método de la clase para obtener el token en la posición justo antes de la posición actual del cursor. Este token corresponde a la llave de cierre con tipo).
 - a. El [GetTokenInfo](#) método [GetLineInfo](#) llama al [Colorizer](#) método en el objeto para obtener todos los tokens de la línea actual.
 - b. El [GetLineInfo](#) método [SetSource](#) llama al [IScanner](#) método en el objeto con el texto de la línea actual.
 - c. El [GetLineInfo](#) método llama [ScanTokenAndProvideInfoAboutIt](#) repetidamente [IScanner](#) al método en el objeto para recopilar todos los tokens de la línea actual.
 - d. El [GetTokenInfo](#) método llama a [Source](#) un método privado en la clase para obtener el token que contiene [GetLineInfo](#) la posición deseada y pasa en la lista de tokens obtenidos del método.
5. El [OnCommand](#) método busca una marca [TokenTriggers](#) de desencadenador de token [GetTokenInfo](#) en el token que se devuelve desde el método; es decir, el token que representa la llave de cierre).
6. Si se encuentra [TokenTriggers](#) la marca [MatchBraces](#) de [Source](#) desencadenador de, se llama al método de la clase.
7. El [MatchBraces](#) método inicia una operación de análisis con el valor de motivo de análisis de [ParseReason](#). En última instancia, esta operación llama al [ParseSource](#) método de la [LanguageService](#) clase. Si el análisis asíncrono está [ParseSource](#) habilitado, esta llamada al método se produce en un subproceso en segundo plano.
8. Cuando finaliza la operación de análisis, se llama a un `HandleMatchBracesResponse` controlador de [Source](#) finalización interno (también conocido como método de devolución de llamada) denominado en la clase. Esta llamada se realiza [LanguageService](#) automáticamente por la clase base, no por el analizador.
9. El `HandleMatchBracesResponse` método obtiene una lista de [AuthoringSink](#) intervalos del [ParseRequest](#) objeto que se almacena en el objeto. (Un intervalo [TextSpan](#) es una estructura que especifica un intervalo de líneas y caracteres en el archivo de origen.) Esta lista de intervalos normalmente contiene dos intervalos, uno para las llaves de apertura y cierre.
10. El `HandleBracesResponse` método [HighlightMatchingBrace](#) llama al [IVsTextView](#) método en el [ParseRequest](#) objeto que se almacena en el objeto. Esto resalta los intervalos dados.
11. Si [LanguagePreferences](#) la [EnableShowMatchingBrace](#) propiedad está `HandleBracesResponse` habilitada, el método obtiene el texto que abarca el intervalo coincidente y muestra los primeros 80 caracteres de ese intervalo en la barra de estado. Esto funciona mejor [ParseSource](#) si el método incluye el elemento de lenguaje que acompaña al par coincidente. Para obtener más información, vea la propiedad

12. Listo.

Resumen

La operación de llaves coincidentes normalmente se limita a pares simples de elementos de lenguaje. Los elementos más complejos, `if(...)` como `{` los triples coincidentes (`" "`, `" }`" y `" "`, o `" else "` `" { y " }`"), se pueden resaltar como parte de una operación de finalización de palabras. Por ejemplo, cuando finaliza la palabra "else", se puede resaltar la instrucción `" if "` coincidente. Si hubiera una `if / else if` serie de instrucciones, todas ellas podrían resaltarse utilizando el mismo mecanismo que las llaves coincidentes. La [Source](#) clase base ya admite esto, como se indica [TokenTriggers](#) a continuación: [TokenTriggers](#) el analizador debe devolver el valor del desencadenador de token combinado con el valor de desencadenador para el token que está antes de la posición del cursor.

Para obtener más información, vea [Coincidencia de llaves en un servicio](#) de lenguaje heredado .

Análisis para la coloración

Colorear el código fuente es sencillo, solo identifica el tipo de token y devuelve información de color sobre ese tipo. La [Colorizer](#) clase actúa como intermediario entre el editor y el analizador para proporcionar información de color sobre cada token. La [Colorizer](#) clase [IScanner](#) utiliza el objeto para ayudar a colorear una línea y también para recopilar información de estado para todas las líneas del archivo de origen. En las clases de [Colorizer](#) servicio de lenguaje MPF, la clase no tiene [IScanner](#) que ser reemplazada porque se comunica con el analizador sólo a través de la interfaz. Proporcione el objeto que [IScanner](#) implementa la interfaz reemplazando el [GetScanner](#) método en la [LanguageService](#) clase.

Al [IScanner](#) analizador se le da una [SetSource](#) línea de código fuente a través del método. Las llamadas al [ScanTokenAndProvideInfoAboutIt](#) método se repiten para obtener el siguiente token en la línea hasta que se agote la línea de tokens. Para la coloración, el MPF trata todo el código fuente como una secuencia de líneas. Por lo tanto, el escáner debe ser capaz de hacer frente a la fuente que viene en él como líneas. Además, cualquier línea se puede pasar al escáner en cualquier momento, y la única garantía es que el escáner recibe la variable de estado de la línea antes de que la línea esté a punto de ser escaneada.

La [Colorizer](#) clase también se utiliza para identificar desencadenadores de token. Estos desencadenadores indican al MPF que un token determinado puede iniciar una operación más compleja, como la finalización de palabras o las llaves coincidentes. Dado que la identificación de estos desencadenadores debe ser rápida y debe producirse en cualquier ubicación, el escáner es el más adecuado para esta tarea.

Para obtener más información, vea [Colorear sintaxis en un servicio](#) de lenguaje heredado .

Análisis de funcionalidad y alcance

El análisis de la funcionalidad y el ámbito requiere más esfuerzo que simplemente identificar los tipos de tokens que se encuentran. El analizador tiene que identificar no solo el tipo de token, sino también la funcionalidad para la que se utiliza el token. Por ejemplo, un identificador es solo un nombre, pero en su idioma, un identificador podría ser el nombre de una clase, espacio de nombres, método o variable, dependiendo del contexto. El tipo general del token puede ser un identificador, pero el identificador también puede tener otros significados, dependiendo de lo que es y dónde se define. Esta identificación requiere que el analizador tenga un conocimiento más amplio sobre el lenguaje que se está analizando. Aquí es [AuthoringSink](#) donde entra en acción la clase. La [AuthoringSink](#) clase recopila información sobre identificadores, métodos, pares de idiomas coincidentes (como llaves y paréntesis) y triples `foreach()` de `{` lenguaje (similar a los pares de idiomas, excepto que hay tres partes, por ejemplo, `" " " "` y `" }`"). Además, puede [AuthoringSink](#) invalidar la clase para admitir la identificación de código, que se utiliza en la validación temprana de puntos de interrupción para que no tenga que cargarse el depurador, y la ventana de depuración **Autos**, que muestra variables y parámetros locales automáticamente cuando se está depurando un programa y requiere que el analizador identifique las variables y

parámetros locales adecuados además de los que presenta el depurador.

El [AuthoringSink](#) objeto se pasa al analizador [ParseRequest](#) como parte del [AuthoringSink](#) objeto y se crea [ParseRequest](#) un nuevo objeto cada vez que se crea un nuevo objeto. Además, [ParseSource](#) el método [AuthoringScope](#) debe devolver un objeto, que se utiliza para controlar varias operaciones de IntelliSense. El [AuthoringScope](#) objeto mantiene una lista para las declaraciones y una lista para los métodos, cualquiera de los cuales se rellena, dependiendo del motivo del análisis. La [AuthoringScope](#) clase debe implementarse.

Vea también

- [Implementación de un servicio de lenguaje heredado](#)
- [Información general del servicio de lenguaje heredado](#)
- [Coloreado de sintaxis en un servicio de lenguaje heredado](#)
- [Coincidencia de llaves en un servicio de lenguaje heredado](#)

Tutorial: Creación de un servicio de lenguaje heredado

06/04/2020 • 8 minutes to read • [Edit Online](#)

El uso de las clases de lenguaje del Visual C# marco de paquetes administrados (MPF) para implementar un servicio de lenguaje es sencillo. Necesita un VSPackage para hospedar el servicio de lenguaje, el propio servicio de lenguaje y un analizador para el idioma.

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Ubicaciones de la plantilla de proyecto de paquete de Visual Studio

La plantilla de proyecto de paquete de Visual Studio se puede encontrar en tres ubicaciones de plantilla diferentes en el cuadro de diálogo **Nuevo proyecto**:

1. En Extensibilidad de Visual Basic. El lenguaje predeterminado del proyecto es Visual Basic.
2. En Extensibilidad de C#. El lenguaje predeterminado del proyecto es C#.
3. En otro tipos de extensibilidad del proyecto. El lenguaje predeterminado del proyecto es C++.

Crear un VSPackage

1. Crear un nuevo VSPackage con el paquete de Visual Studio plantilla de proyecto.

Si va a agregar un servicio de lenguaje a un VSPackage existente, omita los pasos siguientes y vaya directamente al procedimiento "Crear la clase de servicio de lenguaje".

2. Escriba MyLanguagePackage para el nombre del proyecto y haga clic en **Aceptar**.

Puedes usar el nombre que quieras. Estos procedimientos detallados aquí asumen MyLanguagePackage como el nombre.

3. Seleccione Visual C# como idioma y la opción para generar un nuevo archivo de claves. Haga clic en **Next**.
4. Introduzca la información de la empresa y del paquete correspondiente. Haga clic en **Next**.
5. Seleccione **Comando de menú**. Haga clic en **Next**.

Si no tiene la intención de admitir fragmentos de código, puede hacer clic en Finalizar e ignorar el paso siguiente.

6. Ingrese el fragmento de `cmdidInsertSnippet` la **inserción** como el nombre del **comando** y para el ID de **comando**. Haga clic en **Finalizar**

El nombre de **comando** y el **identificador** de comando pueden ser lo que desee, estos son solo ejemplos.

Crear la clase de servicio de lenguaje

1. En el **Explorador de soluciones**, haga clic con el botón secundario en el proyecto MyLanguagePackage, elija **Agregar**, **Referenciay**, a continuación, elija el botón **Agregar nueva referencia**.
2. En el cuadro de diálogo **Agregar referencia**, seleccione

Microsoft.VisualStudio.Package.LanguageService en la pestaña .NET y haga clic en **Aceptar**.

Esto debe hacerse solo una vez para el proyecto de paquete de idioma.

3. En el **Explorador de soluciones**, haga clic con el botón secundario en el proyecto VSPackage y seleccione **Agregar, Clase**.
4. Asegúrese de que **Clase** está seleccionada en la lista de plantillas.
5. Escriba **MyLanguageService.cs** para el nombre del archivo de clase y haga clic en **Agregar**.

Puedes usar el nombre que quieras. Estos procedimientos `MyLanguageService` detallados aquí asumen como el nombre.

6. En el archivo MyLanguageService.cs, `using` agregue las siguientes directivas.

```
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;
using Microsoft.VisualStudio.OLE.Interop;
```

```
Imports Microsoft.VisualStudio
Imports Microsoft.VisualStudio.Package
Imports Microsoft.VisualStudio.TextManager.Interop
Imports Microsoft.VisualStudio.OLE.Interop
```

7. Modifique `MyLanguageService` la clase para [LanguageService](#) derivar de la clase:

```
class MyLanguageService : LanguageService
```

```
Friend Class MyLanguageService
    Inherits LanguageService
```

8. Coloque el cursor en "LanguageService" y, en el menú **Editar, IntelliSense**, seleccione **Implementar clase abstracta**. Esto agrega los métodos mínimos necesarios para implementar una clase de servicio de lenguaje.
9. Implemente los métodos abstractos como se describe en [Implementación de un servicio de lenguaje heredado](#).

Registrar el servicio de idiomas

1. Abra el archivo MyLanguagePackagePackage.cs `using` y agregue las siguientes directivas:

```
Imports Microsoft.VisualStudio
Imports Microsoft.VisualStudio.Package
Imports Microsoft.VisualStudio.TextManager.Interop
```

```
using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;
```

2. Registre la clase de servicio de lenguaje como se describe en [Registro de un servicio de lenguaje heredado](#). Esto incluye las secciones ProvideXX attributes y "Proffering the Language Service". Use MyLanguageService donde este tema usa TestLanguageService.

El analizador y el escáner

1. Implemente un analizador y un analizador para su idioma como se describe en Analizador y analizador de servicios de [lenguaje heredado](#).

La forma de implementar el analizador y el analizador depende totalmente de usted y está fuera del alcance de este tema.

Características del servicio de idiomas

Para implementar cada característica en el servicio de lenguaje, normalmente deriva una clase de la clase de servicio de lenguaje MPF adecuada, implementa los métodos abstractos según sea necesario e invalida los métodos adecuados. Las clases que cree y/o de las que derive dependerán de las características que desee admitir. Estas características se describen en detalle en Características del servicio de [lenguaje heredado](#). El siguiente procedimiento es el enfoque general para derivar una clase de las clases MPF.

Derivado de una clase MPF

1. En el **Explorador de soluciones**, haga clic con el botón secundario en el proyecto VSPackage y seleccione **Agregar, Clase**.
2. Asegúrese de que **Clase** está seleccionada en la lista de plantillas.

Escriba un nombre adecuado para el archivo de clase y haga clic en **Agregar**.

3. En el nuevo archivo de `using` clase, agregue las siguientes directivas.

```
using Microsoft.VisualStudio;  
using Microsoft.VisualStudio.Package;  
using Microsoft.VisualStudio.TextManager.Interop;  
using Microsoft.VisualStudio.OLE.Interop;
```

```
Imports Microsoft.VisualStudio  
Imports Microsoft.VisualStudio.Package  
Imports Microsoft.VisualStudio.TextManager.Interop  
Imports Microsoft.VisualStudio.OLE.Interop
```

4. Modifique la clase para derivar de la clase MPF deseada.
5. Agregue un constructor de clase que tome al menos los mismos parámetros que el constructor de la clase base y pase los parámetros del constructor al constructor de la clase base.

Por ejemplo, el constructor de una [Source](#) clase derivada de la clase podría tener el siguiente aspecto:

```
namespace MyLanguagePackage  
{  
    class MySource : Source  
    {  
        public MySource(LanguageService service,  
                        IVsTextLines textLines,  
                        Colorizer colorizer)  
            : base(service, textLines, colorizer)  
        {  
        }  
    }  
}
```

```
Namespace MyLanguagePackage
    Friend Class MySource
        Inherits Source
        Public Sub New(ByVal service As LanguageService, ByVal textLines As IVsTextLines,
                        ByVal colorizer As Colorizer)
            MyBase.New(service, textLines, colorizer)
        End Sub
    End Class
End Namespace
```

6. En el menú **Editar, IntelliSense**, seleccione **Implementar clase abstracta** si la clase base tiene métodos abstractos que se deben implementar.
7. De lo contrario, coloque el intercalador dentro de la clase e introduzca el método que se va a invalidar.

Por ejemplo, `public override` escriba para ver una lista de todos los métodos que se pueden invalidar en esa clase.

Vea también

- [Implementación de un servicio de lenguaje heredado](#)

Tutorial: Obtención de una lista de fragmentos de código instalados (implementación heredada)

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un fragmento de código es un fragmento de código que se puede insertar en el búfer de origen con un comando de menú (que permite elegir entre una lista de fragmentos de código instalados) o seleccionando un acceso directo de fragmento de código de una lista de finalización de IntelliSense.

El [EnumerateExpansions](#) método obtiene todos los fragmentos de código para un GUID de idioma específico. Los accesos directos para esos fragmentos de código se pueden insertar en una lista de finalización de IntelliSense.

Consulte [Compatibilidad con fragmentos](#) de código en un servicio de lenguaje heredado para obtener más información sobre cómo implementar fragmentos de código en un servicio de lenguaje de marco de paquetes administrados (MPF).

Para recuperar una lista de fragmentos de código

1. El código siguiente muestra cómo obtener una lista de fragmentos de código para un idioma determinado. Los resultados se almacenan [VsExpansion](#) en una matriz de estructuras. Este método utiliza [GetGlobalService](#) el método [IVsTextManager](#) estático [SVsTextManager](#) para obtener la interfaz del servicio. Sin embargo, también puede usar el proveedor de [QueryService](#) servicios dado a su VSPackage y llamar al método.

```
using System;
using System.Collections;
using System.Runtime.InteropServices;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.Shell;
using Microsoft.VisualStudio.TextManager.Interop;

[Guid("00000000-0000-0000-0000-000000000000")] //create a new GUID for the language service
namespace TestLanguage
{
    class TestLanguageService : LanguageService
    {
        private void GetSnippets(Guid languageGuid,
                                ref ArrayList expansionsList)
        {
            IVsTextManager textManager =
            (IVsTextManager)Package.GetGlobalService(typeof(SVsTextManager));
            if (textManager != null)
            {
                IVsTextManager2 textManager2 = (IVsTextManager2)textManager;
                if (textManager2 != null)
                {
                    IVsExpansionManager expansionManager = null;
                    textManager2.GetExpansionManager(out expansionManager);
                    if (expansionManager != null)
                    {
                        // Tell the environment to fetch all of our snippets.
                        IVsExpansionEnumeration expansionEnumerator = null;
                        expansionManager.EnumerateExpansions(languageGuid,
                            0, // return all info
                            null, // return all types
                            0, // return all types
                            1, // include snippets without types
                            0, // do not include duplicates
                            out expansionEnumerator);
                    }
                }
            }
        }
    }
}
```

```

        if (expansionEnumerator != null)
        {
            // Cache our expansions in a VsExpansion array
            uint count = 0;
            uint fetched = 0;
            VsExpansion expansionInfo = new VsExpansion();
            IntPtr[] pExpansionInfo = new IntPtr[1];

            // Allocate enough memory for one VSExpansion structure. This memory is
            // filled in by the Next method.
            pExpansionInfo[0] = Marshal.AllocCoTaskMem(Marshal.SizeOf(expansionInfo));

            expansionEnumerator.GetCount(out count);
            for (uint i = 0; i < count; i++)
            {
                expansionEnumerator.Next(1, pExpansionInfo, out fetched);
                if (fetched > 0)
                {
                    // Convert the returned blob of data into a structure that can be
                    // read in managed code.
                    expansionInfo =
                        (VsExpansion)Marshal.PtrToStructure(pExpansionInfo[0], typeof(VsExpansion));

                    if (!String.IsNullOrEmpty(expansionInfo.shortcut))
                    {
                        expansionsList.Add(expansionInfo);
                    }
                }
            }
            Marshal.FreeCoTaskMem(pExpansionInfo[0]);
        }
    }
}

```

Para llamar al método GetSnippets

1. El método siguiente muestra `GetSnippets` cómo llamar al método al finalizar una operación de análisis. Se [OnParseComplete](#) llama al método después de una operación de análisis que se inició con el motivo [ParseReason](#).

NOTE

La `expansionsList` lista de matrices se almacena en caché por motivos de rendimiento. Los cambios en los fragmentos de código no se reflejan en la lista hasta que Visual Studio el servicio de lenguaje se detiene y se vuelve a cargar (por ejemplo, deteniendo y reiniciando).

```
class TestLanguageService : LanguageService
{
    private ArrayList expansionsList;

    public override void OnParseComplete(ParseRequest req)
    {
        if (this.expansionsList == null)
        {
            this.expansionsList = new ArrayList();
            GetSnippets(this.GetLanguageServiceGuid(),
                ref this.expansionsList);
        }
    }
}
```

Para usar la información del fragmento de código

1. El código siguiente muestra cómo utilizar la `GetSnippets` información de fragmento de código devuelta por el método. Se `AddSnippets` llama al método desde el analizador en respuesta a cualquier motivo de análisis que se use para rellenar una lista de fragmentos de código. Esto debe tener lugar después de que el análisis completo se ha hecho por primera vez.

El `AddDeclaration` método crea una lista de declaraciones que se muestra más adelante en una lista de finalización.

La `TestDeclaration` clase contiene toda la información que se puede mostrar en una lista de finalización, así como el tipo de declaración.


```

class TestAuthoringScope : AuthoringScope
{
    public void AddDeclarations(TestDeclaration declaration)
    {
        if (m_declarations == null)
            m_declarations = new List<TestDeclaration>();
        m_declarations.Add(declaration);
    }
}
class TestDeclaration
{
    private string m_name;
    private string m_description;
    private string m_type;

    public TestDeclaration(string name, string desc, string type)
    {
        m_name = name;
        m_description = desc;
        m_type = type;
    }
}

class TestLanguageService : LanguageService
{
    internal void AddSnippets(ref TestAuthoringScope scope)
    {
        if (this.expansionsList != null && this.expansionsList.Count > 0)
        {
            int count = this.expansionsList.Count;
            for (int i = 0; i < count; i++)
            {
                VsExpansion expansionInfo = (VsExpansion)this.expansionsList[i];
                scope.AddDeclaration(new TestDeclaration(expansionInfo.title,
                    expansionInfo.description,
                    "snippet"));
            }
        }
    }
}

```

Vea también

- [Compatibilidad con fragmentos de código en un servicio de lenguaje heredado](#)

Características del servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un servicio de lenguaje de marco de paquetes administrados (MPF) puede admitir una o varias Visual Studio características, como resaltado de sintaxis, IntelliSense y validación de puntos de interrupción. Cada característica se puede implementar independientemente de las demás, pero todas requieren un analizador y un analizador, excepto para el resaltado de sintaxis, que requiere sólo un analizador.

En esta sección

- [Coincidencia de llaves en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la coincidencia de pares de idioma, también conocida como coincidencia de llaves.

- [Comentario de código en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir comentarios y quitar comentarios del código seleccionado.

- [Propiedades de documento personalizadas en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir las propiedades del documento que están incrustadas en un archivo de origen.

- [Esquematización en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la esquematización a través de la implementación de regiones ocultas.

- [Cambio de formato de código en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir el reformato del código.

- [Compatibilidad con fragmentos de código en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir fragmentos de código, que son segmentos de código que se insertan y se pueden editar.

- [Información de parámetros en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la operación información de parámetros de IntelliSense para mostrar una firma de método a medida que se escribe el método.

- [Información rápida en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la operación de información rápida de IntelliSense para mostrar información sobre un identificador.

- [Finalización de miembros en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la operación de finalización de miembro de IntelliSense para seleccionar un miembro de un espacio de nombres de una lista.

- [Finalización de palabras en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir la operación IntelliSense Complete Word para completar palabras con tipo parcial.

- [Compatibilidad con la ventana Automático en un servicio de lenguaje heredado](#)

Describe lo que un servicio de lenguaje puede hacer para admitir la ventana **Autos** mientras se está depurando.

- [Compatibilidad con la barra de navegación en un servicio de lenguaje heredado](#)

Describe cómo utilizar la barra de **navegación** en la parte superior de la vista del editor para proporcionar una navegación rápida a cualquier tipo o miembro del archivo que se muestra en esa vista.

- [Coloreado de sintaxis en un servicio de lenguaje heredado](#)

Describe lo que se requiere para admitir el resaltado de sintaxis del código fuente.

- [Validación de puntos de interrupción en un servicio de lenguaje heredado](#)

Describe lo que un servicio de lenguaje puede hacer para admitir la validación de puntos de interrupción fuera de un depurador.

Secciones relacionadas

- [Escáner y analizador del servicio de lenguaje heredado](#)

Describe el analizador y el analizador necesarios para implementar todas las características de un servicio de lenguaje que usa el marco de paquete administrado.

- [Implementación de un servicio de lenguaje heredado](#)

Describe lo que se requiere para implementar un servicio de lenguaje mediante el MPF.

- [Registro de un servicio de lenguaje heredado](#)

Describe los pasos necesarios para registrar un servicio Visual Studio de lenguaje basado en MPF con .

- [Usar IntelliSense](#)

Explica cómo IntelliSense facilita el acceso a las referencias de lenguaje.

- [Implementación de un servicio de lenguaje heredado](#)

Proporciona información sobre cómo usar el marco de paquete administrado (MPF) para implementar un servicio de lenguaje con todas las funciones en código administrado.

Coincidencia de llaves en un servicio de lenguaje heredado

06/04/2020 • 6 minutes to read • [Edit Online](#)

La coincidencia de llaves ayuda al desarrollador a realizar un seguimiento de los elementos del lenguaje que deben producirse juntos, como paréntesis y llaves. Cuando un desarrollador introduce una llave de cierre, se resalta la llave de apertura.

Puede hacer coincidir dos o tres elementos co-ocurrentes, llamados pares y triples. Los triples son conjuntos de tres elementos co-ocurrentes. Por ejemplo, en C-, `foreach` la `foreach()` instrucción forma un triple: `{ , y }`. Los tres elementos se resaltan cuando se escribe la llave de cierre.

Los servicios de lenguaje heredados se implementan como parte de un `VSPackage`, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información sobre la nueva forma de implementar la coincidencia de llaves, consulte [Tutorial: Mostrar llaves coincidentes](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

La [AuthoringSink](#) clase admite pares y [MatchPair](#) triples con los métodos y [MatchTriple](#)

Implementación

El servicio de lenguaje debe identificar todos los elementos coincidentes en el idioma y, a continuación, buscar todos los pares coincidentes. Esto se logra normalmente mediante la implementación [IScanner](#) para [ParseSource](#) detectar un idioma coincidente y, a continuación, usar el método para que coincida con los elementos.

El [OnCommand](#) método llama al analizador para tokenizar la línea y devolver el token justo antes del símbolo de intercalación. El analizador indica que se ha encontrado un par de [TokenTriggers](#) elementos de lenguaje estableciendo un valor de desencadenador de token en el token actual. El [OnCommand](#) método [MatchBraces](#) llama al método [BeginParse](#) que a su vez [ParseReason](#) llama al método con el valor de motivo de análisis de para buscar el elemento de lenguaje coincidente. Cuando se encuentra el elemento de idioma coincidente, ambos elementos se resaltan.

Para obtener una descripción completa de cómo escribir una llave desencadena el resaltado de llaves, consulte la sección Ejemplo de *operación* de análisis en el artículo Analizador y analizador de servicios de [lenguaje heredados](#).

Habilite la compatibilidad con la coincidencia de llaves

El [ProvideLanguageServiceAttribute](#) atributo puede establecer las entradas del Registro [MatchBraces](#), [MatchBracesAtCaret](#) [LanguagePreferences](#) [ShowMatchingBrace](#) que establecen las propiedades correspondientes de la clase. El usuario también puede establecer las propiedades de preferencia de idioma.

ENTRADA DEL REGISTRO	PROPIEDAD	DESCRIPCIÓN
MatchBraces	EnableMatchBraces	Habilita la coincidencia de llaves.

ENTRADA DEL REGISTRO	PROPIEDAD	DESCRIPCIÓN
MatchBracesAtCaret	EnableMatchBracesAtCaret	Permite la coincidencia de llaves a medida que se mueve el intercalador.
ShowMatchingBrace	EnableShowMatchingBrace	Resalta la llave correspondiente.

Coincidir con las declaraciones condicionales

Puede hacer coincidir instrucciones condicionales, `if` como `else if`, y `else` `#if`, o `#elif`, `#else`, `#endif`, de la misma manera que los delimitadores coincidentes. Puede subclase [AuthoringSink](#) de la clase y proporcionar un método que puede agregar intervalos de texto, así como delimitadores a la matriz interna de elementos coincidentes.

Ajuste el gatillo

En el ejemplo siguiente se muestra cómo detectar paréntesis coincidentes, llaves y llaves cuadradas, y establecer el desencadenador para él en el escáner. El [OnCommand](#) método [Source](#) de la clase detecta el desencadenador y llama al analizador para buscar el par coincidente (consulte la sección *Buscar la coincidencia* en este artículo). Este ejemplo es solo para fines ilustrativos. Se supone que el analizador `GetNextToken` contiene un método que identifica y devuelve tokens de una línea de texto.

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestScanner : IScanner
    {
        private const string braces = "()[]{}";
        private Lexer lex;

        public bool ScanTokenAndProvideInfoAboutIt(TokenInfo tokenInfo,
                                                    ref int state)
        {
            bool foundToken = false;
            string token = lex.GetNextToken();
            if (token != null)
            {
                foundToken = true;
                char firstChar = token[0];
                if (Char.IsPunctuation(firstChar) && token.Length > 0)
                {
                    if (braces.IndexOf(firstChar) != -1)
                    {
                        tokenInfo.Trigger = TokenTriggers.MatchBraces;
                    }
                }
            }
            return foundToken;
        }
    }
}
```

Coincide con los frenos

Este es un ejemplo simplificado `{ }` para `( )` hacer `[ ]` coincidir los elementos [AuthoringSink](#) de lenguaje `,` y `,` y agregar sus intervalos al objeto. Este enfoque no es un enfoque recomendado para analizar el código fuente; es sólo para fines ilustrativos.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class Parser
    {
        private IList<TextSpan[]> m_braces;
        public IList<TextSpan[]> Braces
        {
            get { return m_braces; }
        }
        private void AddMatchingBraces(TextSpan braceSpan1, TextSpan braceSpan2)
        {
            if IsMatch(braceSpan1, braceSpan2)
                m_braces.Add(new TextSpan[] { braceSpan1, braceSpan2 });
        }

        private bool IsMatch(TextSpan braceSpan1, TextSpan braceSpan2)
        {
            //definition for matching here
        }
    }

    public class TestLanguageService : LanguageService
    {
        Parser parser = new Parser();
        Source source = (Source) this.GetSource(req.FileName);

        private AuthoringScope ParseSource(ParseRequest req)
        {
            if (req.Sink.BraceMatching)
            {
                if (req.Reason==ParseReason.MatchBraces)
                {
                    foreach (TextSpan[] brace in parser.Braces)
                    {
                        req.Sink.MatchPair(brace[0], brace[1], 1);
                    }
                }
            }
            return new TestAuthoringScope();
        }
    }
}

```

Vea también

- [Características del servicio de lenguaje heredado](#)
- [Analizador y analizador de servicios de lenguaje heredados](#)

Código de comentarios en un servicio de lenguaje heredado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los lenguajes de programación suelen proporcionar un medio para anotar o comentar el código. Un comentario es una sección de texto que proporciona información adicional sobre el código, pero se omite durante la compilación o interpretación.

Las clases de marco de paquete administrado (MPF) proporcionan compatibilidad para comentar y quitar los comentarios del texto seleccionado.

Estilos de comentario

Hay dos estilos generales de comentario:

1. Comentarios de línea, donde el comentario está en una sola línea.
2. Bloquear comentarios, donde el comentario puede incluir varias líneas.

Los comentarios de línea suelen tener un carácter inicial (o caracteres), mientras que los comentarios de bloque tienen caracteres inicial y final. Por ejemplo, en C-, un `//` comentario de línea `/*` comienza con `*/`, y un comentario de bloque comienza con y termina con .

Cuando el usuario selecciona el comando Selección de **comentarios** en [CommentSpan](#) el menú [Source Editor > avanzada](#), el comando se enruta al método de la clase. Cuando el usuario selecciona el comando **Anular comentario**selección [UncommentSpan](#) , el comando se enruta al método.

Comentarios del código de soporte

Puede hacer que su servicio de lenguaje `EnableCommenting` admita [ProvideLanguageServiceAttribute](#) comentarios de código mediante el parámetro con nombre del archivo . Esto establece [EnableCommenting](#) la [LanguagePreferences](#) propiedad de la clase. Para obtener más información acerca de cómo establecer características de servicio de lenguaje, vea [Registrar un servicio](#) de lenguaje heredado .

También debe invalidar [GetCommentFormat](#) el método [CommentInfo](#) para devolver una estructura con los caracteres de comentario para el idioma. Los caracteres de comentario de línea de estilo de C-style son el valor predeterminado.

Ejemplo

Este es un ejemplo [GetCommentFormat](#) de implementación del método.

```
using Microsoft.VisualStudio.Package;

namespace MyLanguagePackage
{
    class MySource : Source
    {
        public override CommentInfo GetCommentFormat() {
            CommentInfo info = new CommentInfo();
            info.LineStart      = "//";
            info.BlockStart     = "/*";
            info.BlockEnd       = "*/";
            info.UseLineComments = true;
            return info;
        }
    }
}
```

Vea también

- [Características del servicio de lenguaje heredado](#)
- [Registrar un servicio de lenguaje heredado](#)

Propiedades de documento personalizadas en un servicio de lenguaje heredado

06/04/2020 • 9 minutes to read • [Edit Online](#)

Las propiedades del documento Visual Studio se pueden mostrar en la ventana **Propiedades**. Los lenguajes de programación generalmente no tienen propiedades asociadas con archivos de origen individuales. Sin embargo, XML admite propiedades de documento que afectan a la codificación, el esquema y la hoja de estilos.

Discusión

Si el lenguaje necesita propiedades de documento personalizadas, debe derivar una clase de la [DocumentProperties](#) clase e implementar las propiedades necesarias en la clase derivada.

Además, las propiedades del documento se almacenan normalmente en el propio archivo de origen. Esto requiere que el servicio de lenguaje analice la información de propiedad del archivo de origen para mostrarla en la ventana **Propiedades** y que actualice el archivo de origen cuando se realice un cambio en las propiedades del documento en la ventana **Propiedades**.

Personalizar la clase DocumentProperties

Para admitir propiedades de documento personalizadas, [DocumentProperties](#) debe derivar una clase de la clase y agregar tantas propiedades como necesite. También debe proporcionar atributos de usuario para organizarlos en la pantalla de la ventana **Propiedades**. Si una propiedad `get` solo tiene un descriptor de acceso, se muestra como de solo lectura en la ventana **Propiedades**. Si una propiedad `get` `set` tiene ambos y descriptors de acceso, la propiedad también se puede actualizar en el **propiedades** ventana.

Ejemplo

A continuación se muestra [DocumentProperties](#) una clase de `Filename` `Description` ejemplo derivada de , que muestra dos propiedades y . Cuando se actualiza una propiedad, [LanguageService](#) se llama a un método personalizado en la clase para escribir la propiedad en el archivo de origen.

```
using System.ComponentModel;
using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    class TestDocumentProperties : DocumentProperties
    {
        private string m_filename;
        private string m_description;

        public TestDocumentProperties(CodeWindowManager mgr)
            : base(mgr)
        {
        }

        // Helper function to initialize this property without
        // going through the FileName property (which does a lot
        // more than we need when the filename is set).
        public void SetFileName(string filename)
        {
            m_filename = filename;
        }
    }
}
```

```

// Helper function to initialize this property without
// going through the Description property (which does a lot
// more than we need when the description is set).
public void SetDescription(string description)
{
    m_description = description;
}

////////////////////////////////////

// The document properties

[CategoryAttribute("General")]
[DescriptionAttribute("Name of the file")]
[DisplayNameAttribute("Filename")]
public string FileName
{
    get { return m_filename; }
    set
    {
        if (value != m_filename)
        {
            m_filename = value;
            SetPropertyValue("Filename", m_filename);
        }
    }
}

[CategoryAttribute("General")]
[DescriptionAttribute("Description of the file")]
[DisplayNameAttribute("Description")]
public string Description
{
    get { return m_description; }
    set
    {
        if (value != m_description)
        {
            m_description = value;
            SetPropertyValue("Description", m_description);
        }
    }
}

////////////////////////////////////

// Private methods.

private void SetPropertyValue(string propertyName, string propertyValue)
{
    Source src = this.GetSource();
    if (src != null)
    {
        TestLanguageService service = src.LanguageService as TestLanguageService;
        if (service != null)
        {
            // Set the property in to the source file.
            service.SetPropertyValue(src, propertyName, propertyValue);
        }
    }
}
}

```

Crear una instancia de la clase DocumentProperties personalizada

Para crear instancias de la clase [CreateDocumentProperties](#) de propiedades de [LanguageService](#) documento personalizada, debe [DocumentProperties](#) invalidar el método en la versión de la clase para devolver una única

instancia de la clase.

Ejemplo

```
using System.ComponentModel;
using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    class TestLanguageService : LanguageService
    {
        private TestDocumentProperties m_documentProperties;

        public override DocumentProperties CreateDocumentProperties(CodeWindowManager mgr)
        {
            if (m_documentProperties == null)
            {
                m_documentProperties = new TestDocumentProperties(mgr);
            }
            return m_documentProperties;
        }
    }
}
```

Propiedades en el archivo de origen

Dado que las propiedades del documento suelen ser específicas del archivo de origen, los valores se almacenan en el propio archivo de origen. Esto requiere la compatibilidad del analizador de lenguaje o del analizador para definir estas propiedades. Por ejemplo, las propiedades de un documento XML se almacenan en el nodo raíz. Los valores del nodo raíz se modifican cuando se cambian los valores de la ventana **Propiedades** y el nodo raíz se actualiza en el editor.

Ejemplo

Este ejemplo almacena `Filename` `Description` las propiedades y en las dos primeras líneas del archivo de origen, incrustadas en un encabezado de comentario especial, como:

```
//!Filename = file.testtext
//!Description = A sample file
```

En este ejemplo se muestran los dos métodos necesarios para obtener y establecer las propiedades del documento de las dos primeras líneas del archivo de origen, así como cómo se actualizan las propiedades si el usuario modifica el archivo de origen directamente. El `SetPropertyValue` método del ejemplo que se muestra `TestDocumentProperties` aquí es el mismo al que se llama desde la clase, como se muestra en la sección *Personalizar la clase DocumentProperties*.

En este ejemplo se utiliza el analizador para determinar el tipo de tokens en las dos primeras líneas. Este ejemplo es solo para fines ilustrativos. Un enfoque más típico de esta situación es analizar el archivo de origen en lo que se denomina un árbol de análisis donde cada nodo del árbol contiene información sobre un token determinado. El nodo raíz contendría las propiedades del documento.

```
using System.ComponentModel;
using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    // TokenType.Comment is the last item in that enumeration.
    public enum TestTokenTypes
    {
        DocPropertyLine = TokenType.Comment + 1,
```

```

        DocPropertyName,
        DocPropertyAssign,
        DocPropertyValue
    }

class TestLanguageService : LanguageService
{
    // Search this many lines from the beginning for properties.
    private static int maxLinesToSearch = 2;

    private TestDocumentProperties m_documentProperties;

    // Called whenever a full parsing operation has completed.
    public override void OnParseComplete(ParseRequest req)
    {
        if (m_documentProperties != null)
        {
            Source src = GetSource(req.View);
            if (src != null)
            {
                string value = GetPropertyValue(src, "Filename");
                m_documentProperties.SetFileName(value);

                value = GetPropertyValue(src, "Description");
                m_documentProperties.SetDescription(value);

                // Update the Properties Window.
                m_documentProperties.Refresh();
            }
        }
    }

    // Retrieves the specified property value from the given source.
    public string GetPropertyValue(Source src, string propertyName)
    {
        string propertyValue = "";

        if (src != null)
        {
            IVsTextColorState colorState = src.ColorState;
            if (colorState != null)
            {
                string line;
                TokenInfo[] lineInfo = null;
                int i;

                for (i = 0; i < maxLinesToSearch; i++)
                {
                    line = src.GetLine(i);
                    lineInfo = src.GetColorizer().GetLineInfo(
                        src.GetTextLines(),
                        i,
                        colorState);

                    if (lineInfo == null)
                    {
                        continue;
                    }
                    if (lineInfo[0].Type != (TokenType)TestTokenTypes.DocPropertyLine)
                    {
                        // Not a property line.
                        continue;
                    }
                    TokenInfo valueInfo = new TokenInfo();
                    int tokenIndex = -1;
                    for (tokenIndex = 0;
                        tokenIndex < lineInfo.Length;
                        tokenIndex++)
                    {
                        if (lineInfo[tokenIndex].Type == (TokenType)TestTokenTypes.DocPropertyName)

```

```

                if (lineInfo[tokenIndex].Type == (TokenType)TestTokenTypes.DocPropertyName)
                {
                    break;
                }
            }
            if (tokenIndex >= lineInfo.Length)
            {
                // No property name on the line.
                continue;
            }
            string name = src.GetText(i,
                                    lineInfo[tokenIndex].StartIndex,
                                    i,
                                    lineInfo[tokenIndex].EndIndex + 1);

            if (name != null)
            {
                if (String.Compare(name, propertyName, true) == 0)
                {
                    for ( ;
                        tokenIndex < lineInfo.Length;
                        tokenIndex++)
                    {
                        if (lineInfo[tokenIndex].Type ==
                            (TokenType)TestTokenTypes.DocPropertyValue)
                        {
                            break;
                        }
                    }
                    if (tokenIndex < lineInfo.Length)
                    {
                        propertyValue = src.GetText(i,
                                                    lineInfo[tokenIndex].StartIndex,
                                                    i,
                                                    lineInfo[tokenIndex].EndIndex + 1);
                    }
                    break;
                }
            }
        }
    }
}

return propertyValue;
}

```

// Sets the specified property into the given source file.

// Called from the TestDocumentProperties class.

```

public void SetPropertyValue(Source src, string propertyName, string propertyValue)
{

```

```

    string newline;

```

```

    if (propertyName == null || propertyName == "")
    {

```

```

        // No property name, so nothing to do
        return;
    }

```

```

    if (propertyValue == null)
    {

```

```

        propertyValue = "";
    }

```

```

    // This is the line to be inserted.

```

```

    newline = String.Format("//!{0} = {1}", propertyName, propertyValue);

```

```

    // First, find the line on which the property belongs.

```

```

    // If line is found, replace it.

```

```

    // Otherwise, insert the line at the beginning of the file.

```

```

    if (src != null)
    {

```

```

        IVsTextColorState colorState = src.ColorState;

```

```

        if (colorState != null)
        {

```

```

1
int         lineNumber = -1;
string      line;
TokenInfo[] lineInfo   = null;
int         i;

for (i = 0; i < maxLinesToSearch; i++)
{
    line      = src.GetLine(i);
    lineInfo = src.GetColorizer().GetLineInfo(
                                                src.GetTextLines(),
                                                i,
                                                colorState);

    if (lineInfo == null)
    {
        continue;
    }
    if (lineInfo[0].Type != (TokenType)TestTokenTypes.DocPropertyLine)
    {
        // Not a property line
        continue;
    }
    TokenInfo valueInfo = new TokenInfo();
    int tokenIndex = -1;
    for (tokenIndex = 0;
         tokenIndex < lineInfo.Length;
         tokenIndex++)
    {
        if (lineInfo[tokenIndex].Type == (TokenType)TestTokenTypes.DocPropertyName)
        {
            break;
        }
    }
    if (tokenIndex >= lineInfo.Length)
    {
        // No property name on the line.
        continue;
    }
    string name = src.GetText(i,
                              lineInfo[tokenIndex].StartIndex,
                              i,
                              lineInfo[tokenIndex].EndIndex + 1);

    if (name != null)
    {
        if (String.Compare(name, propertyName, true) == 0)
        {
            lineNumber = i;
            break;
        }
    }
}

// Set up an undo context that also handles the insert/replace.
EditArray editArray = new EditArray(src,
                                     true,
                                     "Update Property");

if (editArray != null)
{
    TextSpan span = new TextSpan();
    if (lineNumber != -1)
    {
        // Replace line.
        int lineLength = 0;
        src.GetTextLines().GetLengthOfLine(lineNumber,
                                             out lineLength);

        span.iStartLine = lineNumber;
        span.iStartIndex = 0;
        span.iEndLine   = lineNumber;
        span.iEndIndex  = lineLength;
    }
}

```

```
    }
    else
    {
        // Insert new line.
        span.iStartLine = 0;
        span.iStartIndex = 0;
        span.iEndLine = 0;
        span.iEndIndex = 0;
        newLine += "\n";
    }
    editArray.Add(new EditSpan(span, newLine));
    editArray.ApplyEdits();
}
}
}
}
}
```

Vea también

- [Características del servicio de lenguaje heredado](#)

Esquematización en un servicio de lenguaje heredado

06/04/2020 • 6 minutes to read • [Edit Online](#)

La esquematización permite contraer un programa complejo en una visión general o un esquema. Por ejemplo, en C- todos los métodos se pueden contraer en una sola línea, mostrando solo la firma del método. Además, las estructuras y clases se pueden contraer para mostrar solo los nombres de las estructuras y clases. Dentro de un único método, la lógica compleja se puede contraer para mostrar `foreach` el `if` flujo `while` general mostrando solo la primera línea de instrucciones como `, , y`.

Los servicios de lenguaje heredados se implementan como parte de un `VSPackage`, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Tutorial: Esquema](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Habilitación del soporte para la esquematización

La `AutoOutlining` entrada del Registro se establece en 1 para habilitar la esquematización automática. La esquematización automática configura un análisis de todo el origen cuando se carga o cambia un archivo para identificar las regiones ocultas y mostrar los glifos de esquematización. La esquematización también puede ser controlada manualmente por el usuario.

El valor `AutoOutlining` de la entrada del `AutoOutlining` Registro se `LanguagePreferences` puede obtener a través de la propiedad de la clase. La `AutoOutlining` entrada del Registro se puede inicializar `ProvideLanguageServiceAttribute` con un parámetro con nombre para el atributo (consulte [Registro de un servicio de lenguaje heredado](#) para obtener más información).

La región oculta

Para proporcionar esquematización, el servicio de lenguaje debe admitir regiones ocultas. Se trata de intervalos de texto que se pueden expandir o contraer. Las regiones ocultas se pueden delimitar mediante símbolos de idioma estándar, como llaves o símbolos personalizados. Por ejemplo, C- `#region` / `#endregion` tiene un par que delimita una región oculta.

Las regiones ocultas son administradas por `IVsHiddenTextSession` un administrador de regiones ocultas, que se expone como la interfaz.

La esquematización `IVsHiddenRegion` utiliza regiones ocultas de la interfaz y contienen el intervalo de la región oculta, el estado visible actual y el banner que se mostrará cuando se contraiga el intervalo.

El analizador de servicios `AddHiddenRegion` de lenguaje utiliza el método para agregar una `AddHiddenRegion` nueva región oculta con el comportamiento predeterminado para las regiones ocultas, mientras que el método permite personalizar la apariencia y el comportamiento del esquema. Una vez que las regiones Visual Studio ocultas se asignan a la sesión de región oculta, administra las regiones ocultas para el servicio de lenguaje.

Si necesita determinar cuándo se destruye la sesión de región oculta, se cambia una región oculta o debe

asegurarse de que una región oculta determinada esté visible; debe derivar una clase [Source](#) de la clase [OnBeforeSessionEnd](#) de [OnHiddenRegionChange](#) invalidar [MakeBaseSpanVisible](#) los métodos adecuados, , , y , respectivamente.

Ejemplo

Este es un ejemplo simplificado de creación de regiones ocultas para todos los pares de llaves. Se supone que el lenguaje proporciona coincidencia de llaves y que las llaves que se van a coincidir incluyen al menos las llaves (. . .
..... Este enfoque es sólo para fines ilustrativos.

Una implementación completa tendría un [ParseSource](#) manejo completo de los casos en . Este ejemplo también muestra [AutoOutlining](#) cómo `true` establecer la preferencia temporalmente. Una alternativa es `AutoOutlining` especificar el `ProvideLanguageServiceAttribute` parámetro con nombre en el atributo del paquete de idioma.

En este ejemplo se supone que las reglas de C- para comentarios, cadenas y literales.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace MyLanguagePackage
{
    public class MyLanguageService : LanguageService
    {
        private LanguagePreferences m_preferences;

        public override LanguagePreferences GetLanguagePreferences()
        {
            if (m_preferences == null)
            {
                m_preferences = new LanguagePreferences(this.Site,
                                                         typeof(MyLanguageService).GUID,
                                                         Name);

                m_preferences.Init();
                // Temporarily enabled auto-outlining
                m_preferences.AutoOutlining = true;
            }
            return m_preferences;
        }

        public override AuthoringScope ParseSource(ParseRequest req)
        {
            Source source = (Source) this.GetSource(req.FileName);
            switch (req.Reason)
            {
                case ParseReason.HighlightBraces:
                case ParseReason.MatchBraces:
                case ParseReason.MemberSelectAndHighlightBraces:
                    if (source.Braces != null)
                    {
                        foreach (TextSpan[] brace in source.Braces)
                        {
                            if (brace.Length == 2)
                            {
                                if (req.Sink.HiddenRegions == true
                                    && source.GetText(brace[0]).Equals("{")
                                    && source.GetText(brace[1]).Equals("}"))
                                {
                                    //construct a TextSpan of everything between the braces
                                    TextSpan hideSpan = new TextSpan();
                                    hideSpan.iStartIndex = brace[0].iStartIndex;
                                    hideSpan.iStartLine = brace[0].iStartLine;
                                    hideSpan.iEndIndex = brace[1].iEndIndex;
                                    hideSpan.iEndLine = brace[1].iEndLine;
                                    req.Sink.ProcessHiddenRegions = true;
                                    req.Sink.AddHiddenRegion(hideSpan);
                                }
                                req.Sink.MatchPair(brace[0], brace[1], 1);
                            }
                            else if (brace.Length >= 3)
                                req.Sink.MatchTriple(brace[0], brace[1], brace[2], 1);
                        }
                    }
                    break;
                default:
                    break;
            }

            // Must always return a valid AuthoringScope object.
            return new MyAuthoringScope();
        }
    }
}

```

Vea también

- [Características del servicio de lenguaje heredado](#)
- [Registro de un servicio de lenguaje heredado](#)

Cambio de formato de código en un servicio de lenguaje heredado

06/04/2020 • 6 minutes to read • [Edit Online](#)

En Visual Studio el código fuente se puede volver a formatear normalizando el uso de sangrías y espacios en blanco. Esto puede incluir insertar o eliminar espacios o pestañas al principio de cada línea, agregar nuevas líneas entre líneas o reemplazar espacios por pestañas o pestañas por espacios.

NOTE

Insertar o eliminar caracteres de nueva línea puede afectar a marcadores como puntos de interrupción y marcadores, pero agregar o quitar espacios o tabulaciones no afecta a los marcadores.

Los usuarios pueden iniciar una operación de reformateo seleccionando **Formato de selección** o **Formato de documento** en el menú **Avanzado** del menú **Edición**. Una operación de reformateo también se puede desencadenar cuando se inserta un fragmento de código o un carácter determinado. Por ejemplo, cuando se escribe una llave de cierre en C-, todo entre la llave abierta coincidente y la llave de cierre se aplica automáticamente al nivel adecuado.

Cuando Visual Studio envía el comando **Formato de selección** o **Formato de documento** al servicio de lenguaje, la `ViewFilter` clase llama al `ReformatSpan` método de la `Source` clase. Para admitir el formato, debe invalidar el `ReformatSpan` método y proporcionar su propio código de formato.

Habilitación de la compatibilidad con el formateo

Para admitir el `EnableFormatSelection` formato, `ProvideLanguageServiceAttribute` el parámetro `true` de la debe establecerse en al registrar el `VSPackage`. Esto establece `EnableFormatSelection` la `true` propiedad en . El `CanReformat` método devuelve el valor de esta propiedad. Si devuelve true, `ViewFilter` la `ReformatSpan` clase llama al archivo .

Implementación del reformateo

Para implementar el reformateo, debe `Source` derivar una `ReformatSpan` clase de la clase e invalidar el método. El `TextSpan` objeto describe el intervalo `EditArray` para dar formato y el objeto contiene las ediciones realizadas en el intervalo. Tenga en cuenta que este intervalo puede ser el documento completo. Sin embargo, dado que es probable que se realicen varios cambios en el intervalo, todos los cambios deben ser reversibles en una sola acción. Para ello, ajuste todos `CompoundAction` los cambios en un objeto (consulte la sección "Uso de la clase `CompoundAction`" en este tema).

Ejemplo

En el ejemplo siguiente se garantiza que hay un único espacio después de cada coma de la selección, a menos que la coma vaya seguida de una tabulación o esté al final de la línea. Se eliminan los espacios finales después de la última coma de una línea. Vea la sección "Uso de la clase `CompoundAction`" en este `ReformatSpan` tema para ver cómo se llama a este método desde el método.

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace MyLanguagePackage
{
```

```

class MySource : Source
{
    private void DoFormatting(EditArray mgr, TextSpan span)
    {
        // Make sure there is one space after every comma unless followed
        // by a tab or comma is at end of line.
        IVsTextLines pBuffer = GetTextLines();
        if (pBuffer != null)
        {
            int startIndex = span.iStartIndex;
            int endIndex = 0;
            string line = "";
            // Loop over all lines in span
            for (int i = span.iStartLine; i <= span.iEndLine; i++)
            {
                if (i < span.iEndLine)
                {
                    pBuffer.GetLengthOfLine(i, out endIndex);
                }
                else
                {
                    endIndex = span.iEndIndex;
                }
                pBuffer.GetLineText(i, startIndex, i, endIndex, out line);

                if (line.Length > 0)
                {
                    int index = 0;
                    // Loop over all commas in line
                    while ((index = line.IndexOf(',', index)) != -1)
                    {
                        int spaceIndex = index + 1;
                        // Determine how many spaces after comma
                        while (spaceIndex < line.Length && line[spaceIndex] == ' ')
                        {
                            ++spaceIndex;
                        }

                        int spaceCount = spaceIndex - (index + 1);
                        string replacementText = " ";
                        bool fDoEdit = true;
                        TextSpan editTextSpan = new TextSpan();

                        editTextSpan.iStartLine = i;
                        editTextSpan.iEndLine = i;
                        editTextSpan.iStartIndex = index + 1;

                        if (spaceIndex == line.Length)
                        {
                            if (spaceCount > 0)
                            {
                                // Delete spaces after a comma at end of line
                                editTextSpan.iEndIndex = spaceIndex;
                                replacementText = "";
                            }
                        }
                        else
                        {
                            // Otherwise, do not add spaces if comma is
                            // at end of line.
                            fDoEdit = false;
                        }
                    }
                }
                else if (spaceCount == 0)
                {
                    if (spaceIndex < line.Length && line[spaceIndex] == '\t')
                    {
                        // Do not insert space if a tab follows
                        // a comma.
                        fDoEdit = false;
                    }
                }
            }
        }
    }
}

```

```

        fDoEdit = false;
    }
    else
    {
        // No space after comma so add a space.
        editTextSpan.iEndIndex = index + 1;
    }
}
else if (spaceCount > 1)
{
    // More than one space after comma, replace
    // with a single space.
    editTextSpan.iEndIndex = spaceIndex;
}
else
{
    // Single space after a comma and not at end
    // of the line, leave it alone.
    fDoEdit = false;
}
if (fDoEdit)
{
    // Add edit operation
    mgr.Add(new EditSpan(editTextSpan, replacementText));
}
index = spaceIndex;
    }
}
startIndex = 0; // All subsequent lines start at 0
}
// Apply all edits
mgr.ApplyEdits();
}
}
}
}
}

```

Uso de la clase CompoundAction

Todo el reformato realizado en una sección de código debe ser reversible en una sola acción. Esto se puede [CompoundAction](#) lograr mediante una clase. Esta clase ajusta un conjunto de operaciones de edición en el búfer de texto en una sola operación de edición.

Ejemplo

Este es un ejemplo de [CompoundAction](#) cómo utilizar la clase. Vea el ejemplo de la sección "Implementación de compatibilidad `DoFormatting` para el formato" de este tema para obtener un ejemplo del método.

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace MyLanguagePackage
{
    class MySource : Source
    {
        public override void ReformatSpan(EditArray mgr, TextSpan span)
        {
            string description = "Reformat code";
            CompoundAction ca = new CompoundAction(this, description);
            using (ca)
            {
                ca.FlushEditActions();    // Flush any pending edits
                DoFormatting(mgr, span);   // Format the span
            }
        }
    }
}
```

Vea también

- [Características del servicio de lenguaje heredado](#)

Compatibilidad con fragmentos de código en un servicio de lenguaje heredado

06/04/2020 • 21 minutes to read • [Edit Online](#)

Un fragmento de código es un fragmento de código que se inserta en el archivo de origen. El propio fragmento de código es una plantilla basada en XML con un conjunto de campos. Estos campos se resaltan después de insertar el fragmento de código y pueden tener valores diferentes en función del contexto en el que se inserte el fragmento de código. Inmediatamente después de insertar el fragmento de código, el servicio de lenguaje puede dar formato al fragmento de código.

El fragmento de código se inserta en un modo de edición especial que permite navegar por los campos del fragmento de código mediante la tecla TAB. Los campos pueden admitir menús desplegables de estilo IntelliSense. El usuario confirma el fragmento de código en el archivo de origen escribiendo la tecla ENTER o ESC. Para obtener más información sobre los fragmentos de código, consulte [Fragmentos de código](#).

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Tutorial: Implementación de fragmentos de código](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Compatibilidad con Managed Package Framework para fragmentos de código

El marco de paquete administrado (MPF) admite la mayoría de la funcionalidad de fragmentos de código, desde leer la plantilla hasta insertar el fragmento de código y habilitar el modo de edición especial. El soporte se [ExpansionProvider](#) administra a través de la clase.

Cuando [Source](#) se crea una [CreateExpansionProvider](#) instancia de [LanguageService](#) la clase, [ExpansionProvider](#) se llama al [LanguageService](#) método de la [ExpansionProvider](#) clase [Source](#) para obtener un objeto (tenga en cuenta que la clase base siempre devuelve un nuevo objeto para cada objeto).

El MPF no admite funciones de expansión. Una función de expansión es una función con nombre que está incrustada en una plantilla de fragmento de código y devuelve uno o varios valores que se colocarán en un campo. Los valores son devueltos por [ExpansionFunction](#) el propio servicio de lenguaje a través de un objeto. El [ExpansionFunction](#) servicio de lenguaje debe implementar el objeto para admitir funciones de expansión.

Proporcionar soporte para fragmentos de código

Para habilitar la compatibilidad con fragmentos de código, debe proporcionar o instalar los fragmentos de código y debe proporcionar los medios para que el usuario inserte esos fragmentos de código. Hay tres pasos para habilitar la compatibilidad con fragmentos de código:

1. Instalación de los archivos de fragmento de código.
2. Habilitar fragmentos de código para el servicio de lenguaje.

3. Invocar el [ExpansionProvider](#) objeto.

Instalación de los archivos de fragmento de código

Todos los fragmentos de código de un idioma se almacenan como plantillas en archivos XML, normalmente una plantilla de fragmento de código por archivo. Para obtener más información sobre el esquema XML utilizado para las plantillas de fragmento de código, vea Referencia de esquema de [fragmentos](#) de código . Cada plantilla de fragmento de código se identifica con un identificador de idioma. Este identificador de idioma se especifica en `Language` el registro <y se coloca en el atributo de la etiqueta Code> de la plantilla.

Normalmente hay dos ubicaciones donde se almacenan los archivos de plantilla de fragmento de código: 1) donde se instaló el idioma y 2) en la carpeta del usuario. Estas ubicaciones se agregan al registro para que el Administrador de **fragmentos** de código de Visual Studio pueda encontrar los fragmentos de código. La carpeta del usuario es donde se almacenan los fragmentos creados por el usuario.

El diseño típico de la carpeta para los archivos de plantilla de fragmento\de código instalados tiene este aspecto: `[InstallRoot]\ [TestLanguage]?Snippets [LCID]`.

`[InstallRoot]` es la carpeta en la que está instalado el idioma.

`[TestLanguage]` es el nombre de su idioma como nombre de carpeta.

`[LCID]` es el ID de configuración regional. Así es como se almacenan las versiones localizadas de los fragmentos de código. Por ejemplo, el ID de configuración regional para inglés es 1033, por lo que `[LCID]` se reemplaza por 1033.

Se debe proporcionar un archivo adicional y que es un archivo de índice, normalmente denominado SnippetsIndex.xml o ExpansionsIndex.xml (puede usar cualquier nombre de archivo válido que termine en .xml). Este archivo se almacena normalmente en la carpeta `[InstallRoot]\ [TestLanguage]` y especifica la ubicación exacta de la carpeta de fragmentos de código, así como el identificador de idioma y el GUID del servicio de lenguaje que usa los fragmentos de código. La ruta exacta del archivo de índice se coloca en el registro como se describe más adelante en "Instalación de las entradas del registro". A continuación se muestra un ejemplo de un archivo SnippetsIndex.xml:

```
<?xml version="1.0" encoding="utf-8" ?>
<SnippetCollection>
  <Language Lang="Testlanguage" Guid="{b614a40a-80d9-4fac-a6ad-fc2868fff7cd}">
    <SnippetDir>
      <OnOff>On</OnOff>
      <Installed>true</Installed>
      <Locale>1033</Locale>
      <DirPath>%InstallRoot%\TestLanguage\Snippets\%LCID%\Snippets</DirPath>
      <LocalizedName>Snippets</LocalizedName>
    </SnippetDir>
  </Language>
</SnippetCollection>
```

La <etiqueta Language> especifica el `Lang` identificador de idioma (el atributo) y el GUID del servicio de lenguaje.

En este ejemplo se supone que ha instalado el servicio de lenguaje en la carpeta de instalación de Visual Studio. %LCID% se sustituye por el id de configuración regional actual del usuario. Se <pueden agregar varias etiquetas de> SnippetDir, una para cada directorio y configuración regional diferentes. Además, una carpeta de fragmento de código puede contener subcarpetas, cada una de las cuales se identifica en el archivo de índice con la <etiqueta de> SnippetSubDir que está incrustada en una <etiqueta de> SnippetDir.

Los usuarios también pueden crear sus propios fragmentos de código para su idioma. Normalmente se almacenan en la carpeta de configuración del usuario, por ejemplo `[TestDocs]`, fragmentos de código `[TestLanguage]`, donde `[TestDocs]` es la ubicación de la carpeta de configuración del usuario para Visual Studio.

Los siguientes elementos de sustitución se <pueden colocar en la ruta de acceso almacenada en la etiqueta de> DirPath en el archivo de índice.

ELEMENTO	DESCRIPCIÓN
%LCID%	Id. de configuración regional.
%InstallRoot%	Carpeta de instalación raíz para Visual Studio, por ejemplo, C:- Archivos de programa-Microsoft Visual Studio 8.
%ProjDir%	Carpeta que contiene el proyecto actual.
%ProjItem%	Carpeta que contiene el elemento de proyecto actual.
%TestDocs%	Carpeta en la carpeta de configuración del usuario, por ejemplo, C:-Documents and Settings [<i>nombre de usuario</i>].

Habilitación de fragmentos de código para su servicio de lenguaje

Puede habilitar fragmentos de código para [ProvideLanguageCodeExpansionAttribute](#) el servicio de lenguaje agregando el atributo al VSPackage (consulte Registrar un servicio de [lenguaje heredado](#) para obtener más información). Los [ShowRoots SearchPaths](#) parámetros y son opcionales, pero debe incluir el `SearchPaths` parámetro con nombre para informar al Administrador de **fragmentos** de código de la ubicación de los fragmentos de código.

A continuación se muestra un ejemplo de cómo utilizar este atributo:

```
[ProvideLanguageCodeExpansion(  
    typeof(TestSnippetLanguageService),  
    "Test Snippet Language",          // Name of language used as registry key  
    0,                                // Resource ID of localized name of language service  
    "Test Snippet Language",          // Name of Language attribute in snippet template  
    @"%InstallRoot%\Test Snippet Language\Snippets\%LCID%\SnippetsIndex.xml", // Path to snippets index  
    SearchPaths = @"%InstallRoot%\Test Snippet Language\Snippets\%LCID%\"] // Path to snippets
```

Llamar al proveedor de expansión

El servicio de lenguaje controla la inserción de cualquier fragmento de código, así como la forma en que se invoca la inserción.

Llamar al proveedor de expansión para fragmentos de código

Hay dos maneras de invocar el proveedor de expansión: mediante un comando de menú o mediante un acceso directo de una lista de finalización.

Insertar un fragmento de código mediante un comando de menú

Para utilizar un comando de menú para mostrar el explorador [DisplayExpansionBrowser](#) de fragmentos de código, agregue un comando de menú y, a continuación, llame al método en la [ExpansionProvider](#) interfaz en respuesta a ese comando de menú.

1. Agregue un comando y un botón al archivo .vsct. Puede encontrar instrucciones para hacerlo en Creación de [una extensión con un comando](#) de menú .
2. Derive una clase [ViewFilter](#) de la [QueryCommandStatus](#) clase e invalide el método para indicar compatibilidad con el nuevo comando de menú. Este ejemplo siempre habilita el comando menu.

```

using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    class TestViewFilter : ViewFilter
    {
        public TestViewFilter(CodeWindowManager mgr, IVsTextView view)
            : base(mgr, view)
        {
        }

        protected override int QueryCommandStatus(ref Guid guidCmdGroup,
            uint nCmdId)
        {
            int hr = base.QueryCommandStatus(ref guidCmdGroup, nCmdId);
            // If the base class did not recognize the command then
            // see if we can handle the command.
            if (hr == (int)Microsoft.VisualStudio.OLE.Interop.Constants.OLECMDERR_E_UNKNOWNGROUP)
            {
                if (guidCmdGroup == GuidList.guidTestLanguagePackageCmdSet)
                {
                    if (nCmdId == PkgCmdIDList.InvokeCodeSnippetsBrowser)
                    {
                        hr = (int)(OLECMDF.OLECMDF_SUPPORTED | OLECMDF.OLECMDF_ENABLED);
                    }
                }
            }
            return hr;
        }
    }
}

```

3. Invalide [HandlePreExec](#) el [ViewFilter](#) método de [ExpansionProvider](#) la clase [DisplayExpansionBrowser](#) para obtener el objeto y llame al método en ese objeto.

```

using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    class TestViewFilter : ViewFilter
    {
        public override bool HandlePreExec(ref Guid guidCmdGroup,
                                            uint nCmdId,
                                            uint nCmdexect,
                                            IntPtr pvaIn,
                                            IntPtr pvaOut)
        {
            if (base.HandlePreExec(ref guidCmdGroup,
                                    nCmdId,
                                    nCmdexect,
                                    pvaIn,
                                    pvaOut))
            {
                // Base class handled the command. Do nothing more here.
                return true;
            }

            if (guidCmdGroup == GuidList.guidTestLanguagePackageCmdSet)
            {
                if (nCmdId == PkgCmdIDList.InvokeCodeSnippetsBrowser)
                {
                    ExpansionProvider ep = this.GetExpansionProvider();
                    if (this.TextView != null && ep != null)
                    {
                        bool bDisplayed = ep.DisplayExpansionBrowser(
                            this.TextView,
                            "TestLanguagePackage Snippet:",
                            null,
                            false,
                            null,
                            false);
                    }
                    return true; // Handled the command.
                }
            }
            return false; // Did not handle the command.
        }
    }
}

```

Visual Studio llama [ExpansionProvider](#) a los siguientes métodos de la clase en el orden especificado durante el proceso de inserción del fragmento de código:

4. [OnItemChosen](#)
5. [IsValidKind](#)
6. [OnBeforeInsertion](#)
7. [FormatSpan](#)
8. [OnAfterInsertion](#)

Después [OnAfterInsertion](#) de llamar al método, se [ExpansionProvider](#) ha insertado el fragmento de código y el objeto está en un modo de edición especial utilizado para modificar un fragmento de código que se acaba de insertar.

Insertar un fragmento de código mediante un acceso directo

La implementación de un acceso directo desde una lista de finalización es mucho más implicada que la

implementación de un comando de menú. Primero debe agregar accesos directos de fragmento de código a la lista de finalización de palabras de IntelliSense. A continuación, debe detectar cuándo se ha insertado un nombre de acceso directo de fragmento de código como resultado de la finalización. Por último, debe obtener el título y la ruta de [InsertNamedExpansion](#) acceso [ExpansionProvider](#) del fragmento de código mediante el nombre de acceso directo y pasar esa información al método en el método.

Para agregar accesos directos de fragmento de [Declarations](#) código [AuthoringScope](#) a la lista de finalización de palabras, agréguelos al objeto de la clase. Debe asegurarse de que puede identificar el acceso directo como un nombre de fragmento de código. Para obtener un ejemplo, vea [Tutorial: Obtener una lista de fragmentos de código instalados \(implementación heredada\)](#).

Puede detectar la inserción del [OnAutoComplete](#) acceso directo [Declarations](#) del fragmento de código en el método de la clase. Dado que el nombre del fragmento de código ya se ha insertado en el archivo de origen, debe quitarse cuando se inserta la expansión. El [InsertNamedExpansion](#) método toma un intervalo que describe el punto de inserción del fragmento de código; si el intervalo incluye el nombre completo del fragmento de código en el archivo de origen, ese nombre se reemplaza por el fragmento de código.

Esta es una [Declarations](#) versión de una clase que controla la inserción de fragmentos de código dado un nombre de acceso directo. Otros métodos [Declarations](#) de la clase se han omitido para mayor claridad. Tenga en cuenta que el [LanguageService](#) constructor de esta clase toma un objeto. Esto se puede pasar desde [AuthoringScope](#) la versión del objeto [AuthoringScope](#) (por ejemplo, la implementación de la clase podría tomar el [LanguageService](#) objeto en su constructor y pasar ese objeto al constructor de `TestDeclarations` clase).

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    internal class TestDeclarations : Declarations
    {
        private ArrayList      declarations;
        private LanguageService languageService;
        private TextSpan        commitSpan;

        public TestDeclarations(LanguageService langService)
            : base()
        {
            languageService = langService;
            declarations = new ArrayList();
        }

        // This method is used to add declarations to the internal list.
        public void AddDeclaration(TestDeclaration declaration)
        {
            declarations.Add(declaration);
        }

        // This method is called to get the string to commit to the source buffer.
        // Note that the initial extent is only what the user has typed so far.
        public override string OnCommit(IVsTextView textView,
                                         string textSoFar,
                                         char commitCharacter,
                                         int index,
                                         ref TextSpan initialExtent)
        {
            // We intercept this call only to get the initial extent
            // of what was committed to the source buffer.
            commitSpan = initialExtent;

            return base.OnCommit(textView,
                                textSoFar,
                                commitCharacter,
                                index.
```

```

        ref initialExtent);
    }

    // This method is called after the string has been committed to the source buffer.
    public override char OnAutoComplete(IVsTextView textView,
        string committedText,
        char commitCharacter,
        int index)
    {
        TestDeclaration item = declarations[index] as TestDeclaration;
        if (item != null)
        {
            // In this example, TestDeclaration identifies types with a string.
            // You can choose a different approach.
            if (item.Type == "snippet")
            {
                Source src = languageService.GetSource(textView);
                if (src != null)
                {
                    ExpansionProvider ep = src.GetExpansionProvider();
                    if (ep != null)
                    {
                        string title;
                        string path;
                        int commitLength = commitSpan.iEndIndex - commitSpan.iStartIndex;
                        if (commitLength < committedText.Length)
                        {
                            // Replace everything that was inserted
                            // so calculate the span of the full
                            // insertion, taking into account what
                            // was inserted when the commitSpan
                            // was obtained in the first place.
                            commitSpan.iEndIndex += (committedText.Length - commitLength);
                        }

                        if (ep.FindExpansionByShortcut(textView,
                            committedText,
                            commitSpan,
                            true,
                            out title,
                            out path))
                        {
                            ep.InsertNamedExpansion(textView,
                                title,
                                path,
                                commitSpan,
                                false);
                        }
                    }
                }
            }
        }
        return '\0';
    }
}

```

Cuando el servicio de lenguaje obtiene el [FindExpansionByShortcut](#) nombre de acceso directo, llama al método para obtener el nombre de archivo y el título del fragmento de código. A continuación, el [InsertNamedExpansion](#) servicio [ExpansionProvider](#) de lenguaje llama al método de la clase para insertar el fragmento de código. Visual Studio llama a los métodos siguientes [ExpansionProvider](#) en el orden especificado de la clase durante el proceso de inserción del fragmento de código:

1. [IsValidKind](#)
2. [OnBeforeInsertion](#)

3. [FormatSpan](#)

4. [OnAfterInsertion](#)

Para obtener más información sobre cómo obtener una lista de fragmentos de código instalados para el servicio de lenguaje, vea [Tutorial: Obtener una lista de fragmentos de código instalados \(implementación heredada\)](#).

Implementación de la clase `ExpansionFunction`

Una función de expansión es una función con nombre que está incrustada en una plantilla de fragmento de código y devuelve uno o varios valores que se colocarán en un campo. Para admitir funciones de expansión en el servicio `ExpansionFunction` de lenguaje, `GetCurrentValue` debe derivar una clase de la clase e implementar el método. A continuación, `CreateExpansionFunction` debe invalidar `LanguageService` el método de la clase para `ExpansionFunction` devolver una nueva instancia de la versión de la clase para cada función de expansión que admita. Si admite una lista de valores posibles de una `GetIntellisenseList` función `ExpansionFunction` de expansión, también debe invalidar el método de la clase para devolver una lista de esos valores.

Una función de expansión que toma argumentos o necesita tener acceso a otros campos no debe asociarse a un campo editable, ya que es posible que el proveedor de expansión no se inicialice completamente en el momento en que se llama a la función de expansión. Como resultado, la función de expansión no puede obtener el valor de sus argumentos ni de ningún otro campo.

Ejemplo

Este es un ejemplo de cómo `GetName` se puede implementar una función de expansión simple llamada. Esta función de expansión anexa un número a un nombre de clase base cada vez que se crea una instancia de la función de expansión (que corresponde a cada vez que se inserta el fragmento de código asociado).

```

using Microsoft.VisualStudio.Package;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        private int classNameCounter = 0;

        public override ExpansionFunction CreateExpansionFunction(
            ExpansionProvider provider,
            string functionName)
        {
            ExpansionFunction function = null;
            if (functionName == "GetName")
            {
                ++classNameCounter;
                function = new TestGetNameExpansionFunction(provider, classNameCounter);
            }
            return function;
        }
    }

    internal class TestGetNameExpansionFunction : ExpansionFunction
    {
        private int nameCount;

        TestGetNameExpansionFunction(ExpansionProvider provider, int counter)
            : base(provider)
        {
            nameCount = counter;
        }

        public override string GetCurrentValue()
        {
            string name = "TestClass";
            name += nameCount.ToString();
            return name;
        }
    }
}

```

Vea también

- [Características del servicio de lenguaje heredado](#)
- [Registro de un servicio de lenguaje heredado](#)
- [Fragmentos de código](#)
- [Tutorial: Obtención de una lista de fragmentos de código instalados \(implementación heredada\)](#)

Información de parámetros en un servicio de lenguaje heredado

06/04/2020 • 8 minutes to read • [Edit Online](#)

Información de parámetros de IntelliSense es una información sobre herramientas que muestra la firma de un método cuando el usuario escribe el carácter de inicio de la lista de parámetros (normalmente un paréntesis abierto) para la lista de parámetros de método. A medida que se introduce cada parámetro y se escribe el separador de parámetros (normalmente una coma), la información sobre herramientas se actualiza para mostrar el siguiente parámetro en negrita.

Las clases de marco de trabajo de paquete administrado (MPF) proporcionan compatibilidad para administrar la información sobre parámetros. El analizador debe detectar caracteres de inicio, siguiente y final de parámetro de parámetro, y debe proporcionar una lista de las firmas del método y sus parámetros asociados.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Ampliación del editor](#) y los servicios de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Implementación

El analizador debe establecer `TokenTriggers` el valor del desencadenador se establece cuando encuentra un carácter de inicio de lista de parámetros (a menudo un paréntesis abierto). Debe establecer `TokenTriggers` un desencadenador cuando encuentra un separador de parámetros (a menudo una coma). Esto hace que se actualice una información sobre parámetros y se muestre el siguiente parámetro en negrita. El analizador debe establecer `TokenTriggers` el valor del desencadenador cuando busque el carácter final de la lista de parámetros (a menudo un paréntesis cercano).

El `TokenTriggers` valor del desencadenador inicia `MethodTip` una llamada al `ParseSource` método, que a su vez llama al analizador de métodos con un motivo de análisis de . Si el analizador determina que el identificador antes del carácter de inicio de la lista de `AuthoringScope` parámetros es un nombre de método reconocido, devuelve una lista de firmas de método coincidentes en el objeto. Si se encontró alguna firma de método, la información de parámetros se muestra con la primera firma de la lista. Esta información sobre herramientas se actualiza a medida que se escribe más de la firma. Cuando se escribe el carácter final de la lista de parámetros, la información de parámetros se elimina de la vista.

NOTE

Para asegurarse de que la información de parámetros información `Methods` sobre herramientas tiene el formato correcto, debe invalidar las propiedades de la clase para proporcionar los caracteres adecuados. La `Methods` clase base asume una firma de método de estilo C. Consulte `Methods` la clase para obtener más información sobre cómo se puede hacer esto.

Habilitación de la compatibilidad con la información de parámetros

Para admitir la información sobre parámetros, debe establecer el `ShowCompletion` parámetro con nombre de `ProvideLanguageServiceAttribute` en `true`. El servicio de lenguaje lee el valor `EnableCodeSense` de esta entrada del Registro de la propiedad.

Además, `ParameterInformation` la propiedad `true` debe establecerse para que se muestre la información de parámetros.

Ejemplo

Este es un ejemplo simplificado de detección de los caracteres de la lista de parámetros y la configuración de los desencadenadores adecuados. Este ejemplo es solo para fines ilustrativos. Se supone que el analizador `GetNextToken` contiene un método que identifica y devuelve tokens de una línea de texto. El código de ejemplo simplemente establece los desencadenadores cada vez que ve el tipo correcto de carácter.

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestScanner : IScanner
    {
        private Lexer lex;
        private const char parameterListStartChar = '(';
        private const char parameterListEndChar = ')';
        private const char parameterNextChar = ',';

        public bool ScanTokenAndProvideInfoAboutIt(TokenInfo tokenInfo,
                                                    ref int state)
        {
            bool foundToken = false;
            string token = lex.GetNextToken();
            if (token != null)
            {
                foundToken = true;
                char c = token[0];
                if (Char.IsPunctuation(c))
                {
                    tokenInfo.Type = TokenType.Operator;
                    tokenInfo.Color = TokenColor.Keyword;
                    tokenInfo.EndIndex = index;

                    if (c == parameterListStartChar)
                    {
                        tokenInfo.Trigger |= TokenTriggers.ParameterStart;
                    }
                    else if (c == parameterListNextChar)
                    {
                        tokenInfo.Trigger |= TokenTriggers.ParameterNext;
                    }
                    else if (c == parameterListEndChar)
                    {
                        tokenInfo.Trigger |= TokenTriggers.ParameterEnd;
                    }
                }
            }
            return foundToken;
        }
    }
}
```

Soporte de la información de parámetros en el analizador

La `Source` clase hace algunas suposiciones `AuthoringScope` `AuthoringSink` sobre el contenido de las clases y cuando se muestra y actualiza la información de parámetros.

- El analizador se [ParseReason](#) proporciona cuando se escribe el carácter de inicio de la lista de parámetros.
- La ubicación indicada [ParseRequest](#) en el objeto es inmediatamente después del carácter de inicio de la lista de parámetros. El analizador debe recopilar las firmas de todas las declaraciones de método disponibles [AuthoringScope](#) en esa posición y almacenarlas en una lista en la versión del objeto. Esta lista incluye el nombre del método, el tipo de método (o el tipo de valor devuelto) y una lista de parámetros posibles. Más adelante, esta lista se busca la firma o las firmas del método que se mostrarán en la información de parámetros.

A continuación, el analizador debe analizar [ParseRequest](#) la línea especificada por el objeto para recopilar el nombre del método que se está introduciendo, así como la distancia a lo largo del usuario en los parámetros de escritura. Esto se logra pasando el nombre [StartName](#) del método [AuthoringSink](#) al método [StartParameters](#) en el objeto y, a continuación, [NextParameter](#) llamando al método cuando se analiza el [EndParameters](#) carácter de inicio de la lista de parámetros, llamando al método cuando se analiza el carácter siguiente de la lista de parámetros y, finalmente, llamando al método cuando se analiza el carácter final de la lista de parámetros. La [Source](#) clase utiliza los resultados de estas llamadas al método para actualizar la información de parámetros de forma adecuada.

Ejemplo

Aquí hay una línea de texto que el usuario puede introducir. Los números debajo de la línea indican qué paso toma el analizador en esa posición en la línea (suponiendo que el análisis se mueve de izquierda a derecha). La suposición aquí es que todo antes de la línea ya se ha analizado para las firmas de método, incluida la firma del método "testfunc".

```
testfunc("a string",3);
    ^^          ^ ^
    12          3 4
```

Los pasos que toma el analizador se describen a continuación:

1. El analizador [StartName](#) llama con el texto "testfunc".
2. El analizador [StartParameters](#) llama .
3. El analizador [NextParameter](#) llama .
4. El analizador [EndParameters](#) llama .

Información rápida en un servicio de lenguaje heredado

06/04/2020 • 5 minutes to read • [Edit Online](#)

Información rápida de IntelliSense muestra información sobre un identificador en el origen cuando el usuario coloca el intercalador en el identificador y selecciona **Información rápida** en el menú **IntelliSense** o mantiene el cursor del mouse sobre el identificador. Esto hace que aparezca una información sobre herramientas con información sobre el identificador. Esta información normalmente consta del tipo de identificador. Cuando el motor de depuración está activo, esta información puede incluir el valor actual. El motor de depuración proporciona valores de expresión, mientras que el servicio de lenguaje solo controla los identificadores.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Tutorial: Visualización de información sobre herramientas quickinfo](#).

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Las clases de servicio de lenguaje de marco de paquetes administrados (MPF) proporcionan compatibilidad completa para mostrar la información rápida de IntelliSense. Todo lo que tiene que hacer es proporcionar el texto que se mostrará y habilitar la función de información rápida.

El texto que se va a [ParseSource](#) mostrar se obtiene llamando al [ParseReason](#) analizador de métodos con un valor de motivo de análisis de `VSConstants.VSStd2KCmdID`. Este motivo indica al analizador que obtenga la información de tipo (o lo que sea apropiado que se muestre [ParseRequest](#) en la información rápida de la herramienta) para el identificador en la ubicación especificada en el objeto. El [ParseRequest](#) objeto es lo [ParseSource](#) que se pasó al método.

El analizador debe analizar todo hasta [ParseRequest](#) la posición en el objeto para determinar los tipos de todos los identificadores. A continuación, el analizador debe obtener el identificador en la ubicación de la solicitud de análisis. Por último, el analizador debe pasar los datos [AuthoringScope](#) de información sobre herramientas asociados [GetDataTipText](#) a ese identificador al objeto para que el objeto pueda devolver el texto del método.

Habilitación de la función de información rápida

Para habilitar la función Información `CodeSense` rápida, debe establecer los parámetros con nombre del `QuickInfo` [ProvideLanguageServiceAttribute](#) archivo. Estos atributos [EnableCodeSense](#) [EnableQuickInfo](#) establecen las propiedades y.

Implementación de la función de información rápida

La [ViewFilter](#) clase controla la operación de información rápida de IntelliSense. Cuando [ViewFilter](#) la clase [VSConstants.VSStd2KCmdID](#) recibe el comando, [ParseSource](#) la clase llama [ParseReason](#) al método con el motivo [VSConstants.VSStd2KCmdID](#) de análisis y la ubicación del símbolo de intercalación en el momento en que se envió el comando. A [ParseSource](#) continuación, el analizador de métodos debe analizar el origen hasta la ubicación dada y, a continuación, analizar el identificador en la ubicación dada para determinar qué mostrar en la información rápida de la información.

La mayoría de los analizadores realizan un análisis inicial de todo el archivo de origen y almacenan los resultados

en un árbol de análisis. El análisis completo [ParseReason](#) se lleva [ParseSource](#) a cabo cuando se pasa al método. Otros tipos de análisis pueden utilizar el árbol de análisis para obtener la información deseada.

Por ejemplo, el valor [ParseReason](#) de motivo de análisis de puede encontrar el identificador en la ubicación de origen y buscarlo en el árbol de análisis para obtener la información de tipo. A continuación, esta información [AuthoringScope](#) de tipo se [GetDataTipText](#) pasa a la clase y la devuelve el método.

Finalización de miembros en un servicio de lenguaje heredado

06/04/2020 • 12 minutes to read • [Edit Online](#)

La finalización de miembros de IntelliSense es una información sobre herramientas que muestra una lista de posibles miembros de un ámbito determinado, como una clase, estructura, enumeración o espacio de nombres. Por ejemplo, en C-, si el usuario escribe "this" seguido de un punto, se presenta una lista de todos los miembros de la clase o estructura en el ámbito actual en una lista desde la que el usuario puede seleccionar.

El marco de paquete administrado (MPF) proporciona compatibilidad con la información sobre herramientas y la administración de la lista en la información sobre herramientas; todo lo que se necesita es la cooperación del analizador para proporcionar los datos que aparecen en la lista.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Ampliación del editor](#) y los servicios de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Cómo funciona

Las siguientes son las dos formas en que se muestra una lista de miembros utilizando las clases MPF:

- Colocar el intercalador en un identificador o después de un carácter de finalización de miembro y seleccionar **Miembros** de lista en el menú **IntelliSense**.
- El **IScanner** analizador detecta un carácter de finalización de miembro y establece un desencadenador de token de **TokenTriggers.MemberSelect** para ese carácter.

Un carácter de finalización de miembro indica que un miembro de una clase, estructura o enumeración debe seguir. Por ejemplo, en C o Visual Basic, `.` el carácter de finalización `.` de `->` miembro es un `,` mientras que en C++ el carácter es `o` un archivo `.` El valor del desencadenador se establece cuando se escanea el carácter de selección de miembro.

El comando de lista de miembros de IntelliSense

El **VSConstants.VSStd2KCmdID** comando inicia una **Completion** llamada al **Source** método **Completion** en la clase y **ParseSource** el método, a su vez, llama al analizador de métodos con el motivo de análisis de **ParseReason.DisplayMemberList**.

El analizador determina el contexto de la posición actual, así como el token por debajo o inmediatamente antes de la posición actual. En función de este token, se presenta una lista de declaraciones. Por ejemplo, en C-, si coloca el intercalador en un miembro de clase y selecciona **Miembros de lista** , obtendrá una lista de todos los miembros de la clase. Si coloca el intercalador después de un período que sigue a una variable de objeto, obtendrá una lista de todos los miembros de la clase que representa el objeto. Tenga en cuenta que si el intercalador se coloca en un miembro cuando se muestra la lista de miembros, al seleccionar un miembro de la lista se reemplaza el miembro en el que se encuentra el intercalador por el de la lista.

El disparador de tokens

El desencadenador `TokenTriggers.MemberSelect` inicia una `Completion` llamada `Source` al método `Completion` de la clase y el método, a su vez, llama al analizador con el motivo de análisis de `ParseReason.MemberSelect`. Si el desencadenador de token también incluye la marca `TokenTriggers.MatchBraces`, el motivo de análisis es `ParseReason.MemberSelectAndHighlightBraces`, que combina la selección de miembros y el resaltado de llaves.

El analizador determina el contexto de la posición actual, así como lo que se ha escrito antes del carácter de selección de miembro. A partir de esta información, el analizador crea una lista de todos los miembros del ámbito solicitado. Esta lista de declaraciones `AuthoringScope` se almacena en `ParseSource` el objeto que se devuelve desde el método. Si se devuelve nifiquiera alguna declaración, se muestra la información sobre la herramienta de finalización de miembros. La información sobre herramientas se `CompletionSet` administra mediante una instancia de la clase.

Habilitar soporte para la finalización de miembros

Debe tener `CodeSense` la entrada del Registro establecida en 1 para admitir cualquier operación de IntelliSense. Esta entrada del Registro se puede establecer `ProvideLanguageServiceAttribute` con un parámetro con nombre pasado al atributo de usuario asociado al paquete de idioma. Las clases de servicio de lenguaje `EnableCodeSense` leen `LanguagePreferences` el valor de esta entrada del Registro de la propiedad de la clase.

Si el analizador devuelve el desencadenador de token de `TokenTriggers.MemberSelect` el analizador devuelve una lista de declaraciones, se muestra la lista de finalización de miembros.

Finalización de miembros de soporte en el escáner

El analizador debe ser capaz de detectar un carácter de finalización de miembro y establecer el desencadenador de token de `TokenTriggers.MemberSelect` cuando se analiza ese carácter.

Ejemplo de escáner

Este es un ejemplo simplificado de detectar el `TokenTriggers` carácter de finalización de miembro y establecer la marca adecuada. Este ejemplo es solo para fines ilustrativos. Se supone que el analizador `GetNextToken` contiene un método que identifica y devuelve tokens de una línea de texto. El código de ejemplo simplemente establece el desencadenador cada vez que ve el tipo correcto de carácter.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestScanner : IScanner
    {
        private Lexer lex;
        private const char memberSelectChar = '.';

        public bool ScanTokenAndProvideInfoAboutIt(TokenInfo tokenInfo,
                                                    ref int state)
        {
            bool foundToken = false;
            string token = lex.GetNextToken();
            if (token != null)
            {
                foundToken = true;
                char c = token[0];
                if (c == memberSelectChar)
                {
                    tokenInfo.Trigger |= TokenTriggers.MemberSelect;
                }
            }
            return foundToken;
        }
    }
}

```

Finalización de miembros de apoyo en el analizador

Para la finalización del miembro, la [Source](#) clase llama al [GetDeclarations](#) método. Debe implementar la lista en una clase [Declarations](#) que se deriva de la clase. Consulte [Declarations](#) la clase para obtener más información sobre los métodos que debe implementar.

Se llama al analizador con [ParseReason.MemberSelect](#) o [ParseReason.MemberSelectAndHighlightBraces](#) cuando se escribe un carácter de selección de miembro. La ubicación proporcionada [ParseRequest](#) en el objeto es inmediatamente después de que el miembro seleccione el carácter. El analizador debe recopilar los nombres de todos los miembros que pueden aparecer en una lista de miembros en ese punto concreto del código fuente. A continuación, el analizador debe analizar la línea actual para determinar el ámbito que el usuario desea asociar con el carácter de selección de miembro.

Este ámbito se basa en el tipo del identificador antes de que el miembro seleccione el carácter. Por ejemplo, en C-, `languageService` dada la variable `LanguageService` miembro que tiene un tipo de , escribir `languageService`. produce una lista de todos `LanguageService` los miembros de la clase. También en C, escribiendo `esto`. produce una lista de todos los miembros de la clase en el ámbito actual.

Ejemplo de analizador

En el ejemplo siguiente se [Declarations](#) muestra una forma de rellenar una lista. Este código supone que el analizador construye una declaración y la `AddDeclaration` agrega `TestAuthoringScope` a la lista llamando a un método en la clase.

```

using System.Collections;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    internal class TestDeclaration
    {

```



```

    public string Name;           // Name of declaration
    public int    TypeImageIndex; // Glyph index
    public string Description;     // Description of declaration

    public TestDeclaration(string name, int typeImageIndex, string description)
    {
        this.Name = name;
        this.TypeImageIndex = typeImageIndex;
        this.Description = description;
    }
}

//=====
internal class TestDeclarations : Declarations
{
    private ArrayList declarations;

    public TestDeclarations()
        : base()
    {
        declarations = new ArrayList();
    }

    public void AddDeclaration(TestDeclaration declaration)
    {
        declarations.Add(declaration);
    }

    ////////////////////////////////////////
    // Declarations of class methods that must be implemented.
    public override int GetCount()
    {
        // Return the number of declarations to show.
        return declarations.Count;
    }

    public override string GetDescription(int index)
    {
        // Return the description of the specified item.
        string description = "";
        if (index >= 0 && index < declarations.Count)
        {
            description = ((TestDeclaration)declarations[index]).Description;
        }
        return description;
    }

    public override string GetDisplayText(int index)
    {
        // Determine what is displayed in the tool tip list.
        string text = null;
        if (index >= 0 && index < declarations.Count)
        {
            text = ((TestDeclaration)declarations[index]).Name;
        }
        return text;
    }

    public override int GetGlyph(int index)
    {
        // Return index of image to display next to the display text.
        int imageIndex = -1;
        if (index >= 0 && index < declarations.Count)
        {
            imageIndex = ((TestDeclaration)declarations[index]).TypeImageIndex;
        }
        return imageIndex;
    }
}

```

```

public override string GetName(int index)
{
    string name = null;
    if (index >= 0 && index < declarations.Count)
    {
        name = ((TestDeclaration)declarations[index]).Name;
    }
    return name;
}
}

//=====
public class TestAuthoringScope : AuthoringScope
{
    private TestDeclarations declarationsList;

    public void AddDeclaration(TestDeclaration declaration)
    {
        if (declaration != null)
        {
            if (declarationsList == null)
            {
                declarationsList = new TestDeclarations();
            }
            declarationsList.AddDeclaration(declaration);
        }
    }

    public override Declarations GetDeclarations(IVsTextView view,
                                                int line,
                                                int col,
                                                TokenInfo info,
                                                ParseReason reason)
    {
        return declarationsList;
    }

    ///////////////////////////////////////////////////////////////////
    // Remainder of AuthoringScope methods not shown.
    ///////////////////////////////////////////////////////////////////
}

//=====
class TestLanguageService : LanguageService
{
    public override AuthoringScope ParseSource(ParseRequest req)
    {
        TestAuthoringScope scope = new TestAuthoringScope();
        if (req.Reason == ParseReason.MemberSelect ||
            req.Reason == ParseReason.MemberSelectAndHighlightBraces)
        {
            // Gather list of declarations based on what the user
            // has typed so far. In this example, this list is an array of
            // MemberDeclaration objects (a helper class you might implement
            // to hold member declarations).
            // How this list is gathered is dependent on the parser
            // and is not shown here.
            MemberDeclarations memberDeclarations;
            memberDeclarations = GetDeclarationsForScope();

            // Now populate the Declarations list in the authoring scope.
            // GetImageIndexBasedOnType() is a helper method you
            // might implement to convert a member type to an index into
            // the image list returned from the language service.
            foreach (MemberDeclaration dec in memberDeclarations)
            {
                scope.AddDeclaration(new TestDeclaration(
                    dec.Name,
                    GetImageIndexBasedOnType(dec.Type)
                ));
            }
        }
    }
}

```

```
        dec.Description));  
    }  
    }  
    return scope;  
}  
}
```

Finalización de palabras en un servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

La finalización de la palabra rellena los caracteres que faltan en una palabra con tipo parcial. Si solo hay una posible finalización, la palabra se completa cuando se introduce el carácter de finalización. Si la palabra parcial coincide con más de una posibilidad, se muestra una lista de posibles terminaciones. Un carácter de finalización puede ser cualquier carácter que no se utilice para los identificadores.

Los servicios de lenguaje heredados se implementan como parte de un `VSPackage`, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Ampliación del editor](#) y los servicios de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Pasos de implementación

1. Cuando el usuario selecciona **Palabra completa** en el menú **IntelliSense**, el `VSConstants.VSStd2KCmdID` comando se envía al servicio de lenguaje.
2. La `ViewFilter` clase detecta el comando `Completion` y llama al `ParseReason` método con el motivo de análisis de .
3. A `Source` continuación, `ParseSource` la clase llama al método para obtener la lista de posibles `CompletionSet` terminaciones de palabras y, a continuación, muestra las palabras en una lista de información sobre herramientas mediante la clase.

Si solo hay una palabra `Source` coincidente, la clase completa la palabra.

Como alternativa, si el analizador `TokenTriggers` devuelve el valor del desencadenador cuando `Source` se escribe el `Completion` primer carácter `ParseReason` de un identificador, la clase lo detecta y llama al método con el motivo de análisis de . En este caso, el analizador debe detectar la presencia de un carácter de selección de miembro y proporcionar una lista de miembros.

Habilitación de soporte para la palabra completa

Para habilitar la compatibilidad `CodeSense` con la finalización de palabras, establezca el parámetro con nombre pasado al `ProvideLanguageServiceAttribute` atributo de usuario asociado al paquete de idioma. Esto establece `EnableCodeSense` la `LanguagePreferences` propiedad en la clase.

El analizador debe devolver una lista de declaraciones `ParseReason` en respuesta al valor de motivo de análisis, para que funcione la finalización de palabras.

Implementación de Word Completa en el Método ParseSource

Para la finalización de palabras, la `Source` clase llama al `GetDeclarations` método de la `AuthoringScope` clase para obtener una lista de posibles coincidencias de palabras. Debe implementar la lista en una clase `Declarations` que se

deriva de la clase. Consulte [Declarations](#) la clase para obtener más información sobre los métodos que debe implementar.

Si la lista contiene una sola [Source](#) palabra, la clase inserta automáticamente esa palabra en lugar de la palabra parcial. Si la lista contiene más [Source](#) de una palabra, la clase presenta una lista de información sobre herramientas de la que el usuario puede seleccionar la opción adecuada.

Consulte también el ejemplo [Declarations](#) de una implementación de clase en [Finalización de miembros en un servicio](#) de lenguaje heredado .

Compatibilidad con la ventana Automático en un servicio de lenguaje heredado

06/04/2020 • 3 minutes to read • [Edit Online](#)

La ventana **Automático** muestra expresiones como variables y parámetros que están en el ámbito cuando el programa que se está depurando está en pausa (ya sea debido a un punto de interrupción o una excepción). Las expresiones pueden incluir variables, locales o globales, y parámetros que se han cambiado en el ámbito local. La ventana **Autos** también puede incluir instancias de una clase, estructura o algún otro tipo. Cualquier cosa que un evaluador de expresiones pueda evaluar puede mostrarse potencialmente en la ventana **Automático**.

El marco de paquete administrado (MPF) no proporciona compatibilidad directa con la ventana **Autos**. Sin embargo, [GetProximityExpressions](#) si invalida el método, puede devolver una lista de expresiones que se presentarán en la ventana **Automático**.

Implementación de soporte para la ventana Autos

Todo lo que necesita hacer para admitir la [GetProximityExpressions](#) ventana [LanguageService Autos](#) es implementar el método en la clase. La implementación debe decidir, dada una ubicación en el archivo de origen, qué expresiones deben aparecer en la ventana **Automático**. El método devuelve una lista de cadenas en las que cada cadena representa una sola expresión. Un valor [S_OK](#) devuelto de indica que la [S_FALSE](#) lista contiene expresiones, mientras que indica que no hay expresiones para mostrar.

Las expresiones reales devueltas son los nombres de las variables o parámetros que aparecen en esa ubicación en el código. Estos nombres se pasan al evaluador de expresiones para obtener valores y tipos que, a continuación, se muestran en la ventana **Automático**.

Ejemplo

En el ejemplo siguiente [GetProximityExpressions](#) se muestra una implementación del [ParseSource](#) método que [ParseReason](#) obtiene una lista de expresiones del método mediante el motivo de análisis. Cada una de las expresiones se ajusta como una `TestVsEnumBSTR` que implementa la [IVsEnumBSTR](#) interfaz.

Tenga en `GetAutoExpressionsCount` `GetAutoExpression` cuenta que los `TestAuthoringSink` métodos y son métodos personalizados en el objeto y se agregaron para admitir este ejemplo. Representan una forma en la `TestAuthoringSink` que se puede tener acceso [AutoExpression](#) a las expresiones agregadas al objeto por el analizador (mediante una llamada al método) fuera del analizador.

```

using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        public override int GetProximityExpressions(IVsTextBuffer buffer,
                                                    int line,
                                                    int col,
                                                    int cLines,
                                                    out IVsEnumBSTR ppEnum)

        {
            int retval = VSConstants.E_NOTIMPL;
            ppEnum = null;
            if (buffer != null)
            {
                IVsTextLines textLines = buffer as IVsTextLines;
                if (textLines != null)
                {
                    Source src = this.GetSource(textLines);
                    if (src != null)
                    {
                        TokenInfo tokenInfo = new TokenInfo();
                        string text = src.GetText();
                        ParseRequest req = CreateParseRequest(src,
                                                            line,
                                                            col,
                                                            tokenInfo,
                                                            text,
                                                            src.GetFilePath(),
                                                            ParseReason.Autos,
                                                            null);

                        req.Scope = this.ParseSource(req);
                        TestAuthoringSink sink = req.Sink as TestAuthoringSink;

                        retval = VSConstants.S_FALSE;
                        int spanCount = sink.GetAutoExpressionsCount();
                        if (spanCount > 0) {
                            TestVsEnumBSTR bstrList = new TestVsEnumBSTR();
                            for (int i = 0; i < spanCount; i++)
                            {
                                TextSpan span;
                                sink.GetAutoExpression(i, out span);
                                string expression = src.GetText(span.iStartLine,
                                                                    span.iStartIndex,
                                                                    span.iEndLine,
                                                                    span.iEndIndex);

                                bstrList.AddString(expression);
                            }
                            ppEnum = bstrList;
                            retval = VSConstants.S_OK;
                        }
                    }
                }
            }
            return retval;
        }
    }
}

```

Compatibilidad con la barra de navegación en un servicio de lenguaje heredado

06/04/2020 • 10 minutes to read • [Edit Online](#)

La barra de navegación en la parte superior de la vista del editor muestra los tipos y miembros del archivo. Los tipos se muestran en el menú desplegable izquierdo y los miembros se muestran en el menú desplegable derecho. Cuando el usuario selecciona un tipo, el intercalador se coloca en la primera línea del tipo. Cuando el usuario selecciona un miembro, el intercalador se coloca en la definición del miembro. Los cuadros desplegables se actualizan para reflejar la ubicación actual del intercalador.

Visualización y actualización de la barra de navegación

Para admitir la barra de navegación, `TypeAndMemberDropDownBars` debe derivar `OnSynchronizeDropdowns` una clase de la clase e implementar el método. Cuando el servicio de lenguaje recibe `LanguageService` una ventana `CodeWindowManager` de código, `IVsCodeWindow` la clase base crea una instancia de , que contiene el objeto que representa la ventana de código. A `CodeWindowManager` continuación, se `IVsTextView` le da un nuevo objeto al objeto. El `CreateDropDownHelper` método obtiene `TypeAndMemberDropDownBars` un objeto. Si devuelve una instancia `TypeAndMemberDropDownBars` de `CodeWindowManager` la `OnSynchronizeDropdowns` clase, el método llama al `TypeAndMemberDropDownBars` método Visual Studio para rellenar las listas internas y pasa el objeto al administrador de barras desplegables. El administrador de barras desplegables, `SetDropDownBar` a su `TypeAndMemberDropDownBars` vez, `IVsDropDownBar` llama al método del objeto para establecer el objeto que contiene las dos barras desplegables.

Cuando el intercalador `OnIdle` se `OnCaretMoved` mueve, el método llama al método. El `OnCaretMoved` método base `OnSynchronizeDropdowns` llama `TypeAndMemberDropDownBars` al método de la clase para actualizar el estado de la barra de navegación. Se pasa un `DropDownMember` conjunto de objetos a este método. Cada objeto representa una entrada en la lista desplegable.

El contenido de la barra de navegación

La barra de navegación normalmente contiene una lista de tipos y una lista de miembros. La lista de tipos incluye todos los tipos disponibles en el archivo de origen actual. Los nombres de tipo incluyen la información completa del espacio de nombres. A continuación se muestra un ejemplo de código de C- con dos tipos:

```
namespace TestLanguagePackage
{
    public class TestLanguageService
    {
        internal struct Token
        {
            int tokenID;
        }
        private Tokens[] tokens;
        private string serviceName;
    }
}
```

Se mostrará `TestLanguagePackage.TestLanguageService` la `TestLanguagePackage.TestLanguageService.Tokens` lista de tipos y .

La lista de miembros muestra los miembros disponibles del tipo seleccionado en la lista de tipos. Mediante el

ejemplo de `TestLanguagePackage.TestLanguageService` código anterior, si es el tipo seleccionado, la lista de miembros contendría los miembros privados `tokens` y `serviceName`. No se `Token` visualiza la estructura interna.

Puede implementar la lista de miembros para que el nombre de un miembro sea negrita cuando el intercalador se coloca dentro de él. Los miembros también se pueden mostrar en texto atenuado, lo que indica que no están dentro del ámbito donde el intercalador está colocado actualmente.

Habilitación de la compatibilidad con la barra de navegación

Para habilitar la compatibilidad con la `ShowDropDownBarOption` barra de `ProvideLanguageServiceAttribute` navegación, debe establecer el parámetro del atributo en `true`. Este parámetro establece la propiedad `ShowNavigationBar`. Para admitir la barra de `TypeAndMemberDropDownBars` navegación, `CreateDropDownHelper` debe implementar `LanguageService` el objeto en el método de la clase.

En la `CreateDropDownHelper` implementación del `ShowNavigationBar` método, si `true` la propiedad `TypeAndMemberDropDownBars` está establecida en , puede devolver un objeto. Si no devuelve el objeto, no se muestra la barra de navegación.

La opción para mostrar la barra de navegación puede ser establecida por el usuario, por lo que es posible que este control se restablezca mientras la vista del editor está abierta. El usuario debe cerrar y volver a abrir la ventana del editor antes de que se realice el cambio.

Implementación de soporte para la barra de navegación

El `OnSynchronizeDropdowns` método toma dos listas (una para cada lista desplegable) y dos valores que representan la selección actual en cada lista. Las listas y los valores de selección `OnSynchronizeDropdowns` se pueden `true` actualizar, en cuyo caso el método debe volver para indicar que las listas han cambiado.

A medida que la selección cambia en la lista desplegable de tipos, la lista de miembros debe actualizarse para reflejar el nuevo tipo. Lo que se muestra en la lista de miembros puede ser:

- La lista de miembros para el tipo actual.
- Todos los miembros disponibles en el archivo de origen, pero con todos los miembros que no están en el tipo actual mostrados en texto atenuado. El usuario todavía puede seleccionar los miembros atenuados, por lo que se pueden utilizar para una navegación rápida, pero el color indica que no forman parte del tipo seleccionado actualmente.

Una implementación `OnSynchronizeDropdowns` del método normalmente realiza los pasos siguientes:

1. Obtenga una lista de las declaraciones actuales para el archivo de origen.

Hay varias maneras de rellenar las listas. Un enfoque consiste en crear un método `LanguageService` personalizado en `ParseSource` la versión de la clase que llama al método con un motivo de análisis personalizado que devuelve una lista de todas las declaraciones. Otro enfoque podría ser `ParseSource` llamar al `OnSynchronizeDropdowns` método directamente desde el método con el motivo de análisis personalizado. Un tercer enfoque podría ser almacenar `AuthoringScope` en caché las declaraciones de la `LanguageService` clase devuelta `OnSynchronizeDropdowns` por la última operación de análisis completo de la clase y recuperarla del método.

2. Rellene o actualice la lista de tipos.

El contenido de la lista de tipos puede actualizarse cuando el origen ha cambiado o si ha elegido cambiar el estilo de texto de los tipos en función de la posición actual del intercalador. Tenga en cuenta que `OnSynchronizeDropdowns` esta posición se pasa al método.

3. Determine el tipo que desea seleccionar en la lista de tipos en función de la posición actual del intercalador.

Puede buscar las declaraciones que se obtuvieron en el paso 1 para buscar el tipo que encierra la posición del intercalador actual y, a continuación, buscar en la lista de tipos de ese tipo para determinar su índice en la lista de tipos.

4. Rellene o actualice la lista de miembros en función del tipo seleccionado.

La lista de miembros refleja lo que se muestra actualmente en el menú desplegable **Miembros**. Es posible que sea necesario actualizar el contenido de la lista de miembros si el origen ha cambiado o si solo se muestran los miembros del tipo seleccionado y el tipo seleccionado ha cambiado. Si decide mostrar todos los miembros en el archivo de origen, el estilo de texto de cada miembro de la lista debe actualizarse si el tipo seleccionado actualmente ha cambiado.

5. Determine el miembro que desea seleccionar en la lista de miembros en función de la posición actual del intercalador.

Busque las declaraciones que se obtuvieron en el paso 1 para el miembro que contiene la posición de intercalación actual y, a continuación, busque en la lista de miembros ese miembro para determinar su índice en la lista de miembros.

6. Devuelve `true` si se han realizado cambios en las listas o en las selecciones de cualquiera de las listas.

Coloreado de sintaxis en un servicio de lenguaje heredado

06/04/2020 • 12 minutes to read • [Edit Online](#)

La coloración de sintaxis es una característica que hace que diferentes elementos de un lenguaje de programación se muestren en un archivo de origen en diferentes colores y estilos. Para admitir esta característica, debe proporcionar un analizador o escáner que pueda identificar los tipos de elementos léxicos o tokens en el archivo. Muchos idiomas distinguen palabras clave, delimitadores (como paréntesis o llaves) y comentarios colorizándolos de diferentes maneras.

Los servicios de lenguaje heredados se implementan como parte de un VSPackage, pero la forma más reciente de implementar características de servicio de lenguaje es usar extensiones MEF. Para obtener más información, consulte [Ampliación del editor](#) y los servicios de lenguaje .

NOTE

Le recomendamos que comience a usar la nueva API del editor lo antes posible. Esto mejorará el rendimiento de su servicio de lenguaje y le permitirá aprovechar las nuevas características del editor.

Implementación

Para admitir la coloración, el marco [Colorizer](#) de paquete administrado [IVsColorizer](#) (MPF) incluye la clase, que implementa la interfaz. Esta clase interactúa [IScanner](#) con un para determinar el token y los colores. Para obtener más información sobre los analizadores, consulte [Analizador y analizador](#) de servicios de lenguaje heredado . A [Colorizer](#) continuación, la clase marca cada carácter del token con la información de color y devuelve esa información al editor que muestra el archivo de origen.

La información de color devuelta al editor es un índice en una lista de elementos coloreables. Cada elemento coloreable especifica un valor de color y un conjunto de atributos de fuente, como negrita o tachado. El editor proporciona un conjunto de elementos coloreables predeterminados que el servicio de lenguaje puede usar. Todo lo que necesita hacer es especificar el índice de color adecuado para cada tipo de token. Sin embargo, puede proporcionar un conjunto de elementos coloreables personalizados y los índices que proporcione para los tokens, y hacer referencia a su propia lista de elementos coloreables en lugar de la lista predeterminada. También debe establecer `RequestStockColors` la entrada del Registro en `RequestStockColors` 0 (o no especifique la entrada en absoluto) para admitir colores personalizados. Puede establecer esta entrada del Registro [ProvideLanguageServiceAttribute](#) con un parámetro con nombre en el atributo definido por el usuario. Para obtener más información sobre cómo registrar un servicio de lenguaje y establecer sus opciones, consulte [Registro de un servicio de lenguaje heredado](#) .

Elementos coloreables personalizados

Para proporcionar sus propios elementos coloreables personalizados, debe invalidar el [GetItemCount](#) método y [GetColorableItem](#) en la [LanguageService](#) clase. El primer método devuelve el número de elementos coloreables personalizados que admite el servicio de lenguaje y el segundo obtiene el elemento coloreable personalizado por índice. Cree la lista predeterminada de elementos coloreables personalizados. En el constructor de su servicio de lenguaje, todo lo que necesita hacer es proporcionar cada elemento coloreable con un nombre. Visual Studio controla automáticamente el caso en el que el usuario selecciona un conjunto diferente de elementos coloreables. Este nombre es el que aparece en la página de propiedades **Fuentes y colores** del

cuadro de diálogo **Opciones** (disponible en el menú **Herramientas** de Visual Studio) y este nombre determina qué color ha reemplazado un usuario. Las opciones del usuario se almacenan en una memoria caché en el registro y se accede por el nombre de color. La página de propiedades **Fuentes y colores** enumera todos los nombres de color en orden alfabético, por lo que puede agrupar los colores personalizados precediendo cada nombre de color con su nombre de idioma; por ejemplo, "**TestLanguage- Comment**" y "**TestLanguage- Keyword**". O puede agrupar los elementos coloreables por tipo, "**Comment (TestLanguage)**" y "**Keyword (TestLanguage)**". Se prefiere agrupar por nombre de idioma.

Caution

Se recomienda encarecidamente que incluya el nombre del idioma en el nombre del elemento coloreable para evitar colisiones con los nombres de elementos coloreables existentes.

NOTE

Si cambia el nombre de uno de los colores durante el desarrollo, debe restablecer la memoria caché que Visual Studio creó la primera vez que se tenía acceso a los colores. Puede hacerlo ejecutando el comando **Restablecer el subárbol experimental** desde el menú del programa SDK de Visual Studio.

Tenga en cuenta que nunca se hace referencia al primer elemento de la lista de elementos coloreables. Visual Studio siempre proporciona los colores y atributos de texto predeterminados para ese elemento. La forma más fácil de lidiar con esto es proporcionar un elemento coloreable marcador de posición como el primer elemento.

Artículos coloreables de alto color

Los elementos coloreables también pueden admitir [IVsHiColorItem](#) valores de color de 24 bits o altos a través de la interfaz. La clase [ColorableItem](#) MPF [IVsHiColorItem](#) admite la interfaz y los colores de 24 bits se especifican en el constructor junto con los colores normales. Vea la clase [ColorableItem](#) para obtener información más detallada. En el ejemplo siguiente se muestra cómo establecer los colores de 24 bits para palabras clave y comentarios. Los colores de 24 bits se utilizan cuando el color de 24 bits se admite en el escritorio del usuario; de lo contrario, se utilizan los colores de texto normales.

Recuerde, estos son los colores predeterminados para su idioma; el usuario puede cambiar estos colores a lo que quiera.

Ejemplo

En este ejemplo se muestra una forma de declarar [ColorableItem](#) y rellenar una matriz de elementos coloreables personalizados mediante la clase. En este ejemplo se establecen los colores de palabra clave y comentario utilizando colores de 24 bits.

```

using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    public class TestLanguageService : LanguageService
    {
        private ColorableItem[] m_colorableItems;

        TestLanguageService() : base()
        {
            m_colorableItems = new ColorableItem[] {
                new ColorableItem("TestLanguage - Text",
                                "Text",
                                COLORINDEX.CI_SYSPLAINTEXT_FG,
                                COLORINDEX.CI_SYSPLAINTEXT_BK,
                                System.Drawing.Color.Empty,
                                System.Drawing.Color.Empty,
                                FONTFLAGS.FF_DEFAULT),
                new ColorableItem("TestLanguage - Keyword",
                                "Keyword",
                                COLORINDEX.CI_MAROON,
                                COLORINDEX.CI_SYSPLAINTEXT_BK,
                                System.Drawing.Color.FromArgb(192,32,32),
                                System.Drawing.Color.Empty,
                                FONTFLAGS.FF_BOLD),
                new ColorableItem("TestLanguage - Comment",
                                "Comment",
                                COLORINDEX.CI_DARKGREEN,
                                COLORINDEX.CI_LIGHTGRAY,
                                System.Drawing.Color.FromArgb(32,128,32),
                                System.Drawing.Color.Empty,
                                FONTFLAGS.FF_DEFAULT)

                // ...
                // Add as many colorable items as you want to support.
            };
        }
    }
}

```

La clase Colorizer y el Escáner

La [LanguageService](#) clase base [GetColorizer](#) tiene un método [Colorizer](#) que instantánea la clase. El analizador que se [GetScanner](#) devuelve desde [Colorizer](#) el método se pasa al constructor de clase.

Debe implementar [GetScanner](#) el método en su [LanguageService](#) propia versión de la clase. La [Colorizer](#) clase utiliza el analizador para obtener toda la información de color del token.

El analizador debe [TokenInfo](#) rellenar una estructura para cada token que encuentre. Esta estructura contiene información como el intervalo que ocupa el token, el índice de color [TokenTriggers](#) que se va a usar, qué tipo es el token y los desencadenadores de token (consulte). Solo el intervalo y el índice de [Colorizer](#) color son necesarios para la coloración de la clase.

El índice de [TokenInfo](#) color almacenado en la [TokenColor](#) estructura suele ser un valor de la enumeración, que proporciona una serie de índices con nombre correspondientes a varios elementos de lenguaje, como palabras clave y operadores. Si la lista de elementos coloreables personalizados coincide con los elementos presentados en la [TokenColor](#) enumeración, puede usar la enumeración como el color para cada token. Sin embargo, si tiene elementos coloreables adicionales o no desea utilizar los valores existentes en ese orden, puede organizar la lista de elementos coloreables personalizados para que se adapte a sus necesidades y devolver el índice adecuado a esa lista. Solo asegúrese de convertir [TokenColor](#) el índice en [TokenInfo](#) un al almacenarlo en la estructura; Visual Studio sólo ve el índice.

Ejemplo

En el ejemplo siguiente se muestra cómo el analizador puede identificar tres tipos de token: números, puntuación e identificadores (cualquier cosa que no sea un número o puntuación). Este ejemplo es solo para fines ilustrativos y no representa una implementación completa del analizador y del analizador. Se supone que hay `Lexer` una `GetNextToken()` clase con un método que devuelve una cadena.

```
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    private Lexer lex;

    public class TestScanner : IScanner
    {
        public bool ScanTokenAndProvideInfoAboutIt(TokenInfo tokenInfo,
                                                    ref int state)
        {
            bool foundToken = false;
            string token = lex.GetNextToken();
            if (token != null)
            {
                char firstChar = token[0];
                if (Char.IsPunctuation(firstChar))
                {
                    tokenInfo.Type = TokenType.Operator;
                    tokenInfo.Color = TokenColor.Keyword;
                }
                else if (Char.IsNumber(firstChar))
                {
                    tokenInfo.Type = TokenType.Literal;
                    tokenInfo.Color = TokenColor.Number;
                }
                else
                {
                    tokenInfo.Type = TokenType.Identifier;
                    tokenInfo.Color = TokenColor.Identifier;
                }
            }
            return foundToken;
        }
    }
}
```

Vea también

- [Características del servicio de lenguaje heredado](#)
- [Escáner y analizador del servicio de lenguaje heredado](#)
- [Registro de un servicio de lenguaje heredado](#)

Validación de puntos de interrupción en un servicio de lenguaje heredado

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un punto de interrupción indica que la ejecución del programa debe detenerse en un punto determinado mientras se ejecuta en un depurador. Un usuario puede colocar un punto de interrupción en cualquier línea del archivo de origen, ya que el editor no tiene conocimiento de lo que constituye una ubicación válida para un punto de interrupción. Cuando se inicia el depurador, todos los puntos de interrupción marcados (denominados puntos de interrupción pendientes) se enlazan a la ubicación adecuada en el programa en ejecución. Al mismo tiempo, los puntos de interrupción se validan para asegurarse de que marcan ubicaciones de código válidas. Por ejemplo, un punto de interrupción en un comentario no es válido, porque no hay código en esa ubicación en el código fuente. El depurador deshabilita los puntos de interrupción no válidos.

Dado que el servicio de lenguaje conoce el código fuente que se muestra, puede validar los puntos de interrupción antes de iniciar el depurador. Puede invalidar `ValidateBreakpointLocation` el método para devolver un intervalo que especifique una ubicación válida para un punto de interrupción. La ubicación del punto de interrupción todavía se valida cuando se inicia el depurador, pero se notifica al usuario de puntos de interrupción no válidos sin esperar a que se cargue el depurador.

Implementación de soporte para validar puntos de interrupción

- Al `ValidateBreakpointLocation` método se le da la posición del punto de interrupción. La implementación debe decidir si la ubicación es válida e indicarlo devolviendo un intervalo de texto que identifique el código asociado con la posición de línea del punto de interrupción.
- Devuelve `S_OK` si la ubicación `S_FALSE` es válida o si no es válida.
- Si el punto de interrupción es válido, el intervalo de texto se resalta junto con el punto de interrupción.
- Si el punto de interrupción no es válido, aparece un mensaje de error en la barra de estado.

Ejemplo

En este ejemplo se `ValidateBreakpointLocation` muestra una implementación del método que llama al analizador para obtener el intervalo de código (si existe) en la ubicación especificada.

En este ejemplo se supone `GetCodeSpan` que `AuthoringSink` ha agregado un método a `true` la clase que valida el intervalo de texto y devuelve si es una ubicación de punto de interrupción válida.

```

using Microsoft.VisualStudio;
using Microsoft.VisualStudio.Package;
using Microsoft.VisualStudio.TextManager.Interop;

namespace TestLanguagePackage
{
    class TestLanguageService : LanguageService
    {
        public override int ValidateBreakpointLocation(IVsTextBuffer buffer,
                                                    int line,
                                                    int col,
                                                    TextSpan[] pCodeSpan)
        {
            int retval = VSConstants.S_FALSE;
            if (pCodeSpan != null)
            {
                // Initialize span to current line by default.
                pCodeSpan[0].iStartLine = line;
                pCodeSpan[0].iStartIndex = col;
                pCodeSpan[0].iEndLine = line;
                pCodeSpan[0].iEndIndex = col;
            }

            if (buffer != null)
            {
                IVsTextLines textLines = buffer as IVsTextLines;
                if (textLines != null)
                {
                    Source src = this.GetSource(textLines);
                    if (src != null)
                    {
                        TokenInfo tokenInfo = new TokenInfo();
                        string text = src.GetText();
                        ParseRequest req = CreateParseRequest(src,
                                                            line,
                                                            col,
                                                            tokenInfo,
                                                            text,
                                                            src.GetFilePath(),
                                                            ParseReason.CodeSpan,
                                                            null);

                        req.Scope = this.ParseSource(req);
                        TestAuthoringSink sink = req.Sink as TestAuthoringSink;

                        TextSpan span = new TextSpan();
                        if (sink.GetCodeSpan(out span))
                        {
                            pCodeSpan[0] = span;
                            retval = VSConstants.S_OK;
                        }
                    }
                }
            }

            return retval;
        }
    }
}

```

Vea también

- [Características del servicio de lenguaje heredado](#)

Compatibilidad con herramientas de exploración de símbolos

06/04/2020 • 6 minutes to read • [Edit Online](#)

Las herramientas Examinador de **objetos**, Vista de **clases**, Explorador de **llamadas** y Buscar resultados de **símbolos** proporcionan capacidades de exploración de símbolos en Visual Studio. Estas herramientas muestran vistas jerárquicas de árbol de símbolos y muestran las relaciones entre los símbolos del árbol. Los símbolos pueden representar espacios de nombres, objetos, clases, miembros de clase y otros elementos de lenguaje contenidos en varios componentes. Los componentes incluyen proyectos de Visual Studio, componentes externos de .NET Framework y bibliotecas de tipos (.tlb). Para obtener más información, consulte [Visualización de la estructura de código](#).

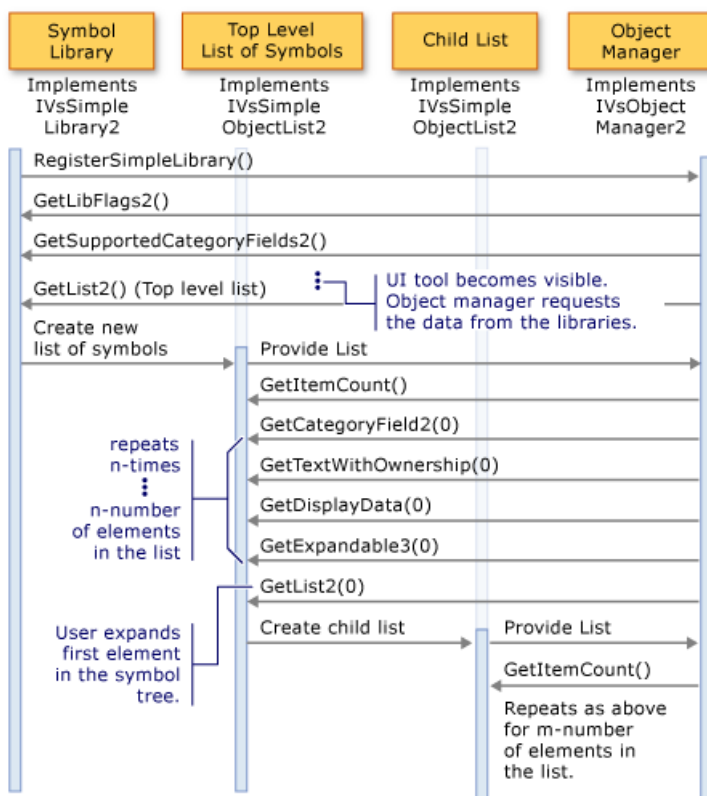
Bibliotecas de navegación de símbolos

Como implementador de lenguaje, puede ampliar las capacidades de exploración de símbolos de Visual Studio mediante la creación de bibliotecas que realizan un seguimiento de los símbolos de los componentes y proporcionan las listas de símbolos al Administrador de objetos de Visual Studio a través de un conjunto de interfaces. La interfaz describe [IVsSimpleLibrary2](#) una biblioteca. El Administrador de objetos de Visual Studio responde a las solicitudes de nuevos datos de las herramientas de exploración de símbolos obteniendo los datos de las bibliotecas y organizándolos. Posteriormente rellena o actualiza las herramientas con los datos solicitados. Para obtener una referencia al administrador [IVsObjectManager2](#) de [SVsObjectManager](#) objetos de `IService` Visual Studio, , pase el identificador de servicio al método.

Cada biblioteca debe registrarse con el Administrador de objetos de Visual Studio, que recopila la información de todas las bibliotecas. Para registrar una biblioteca, llame al [RegisterSimpleLibrary](#) método. Dependiendo de la herramienta que inicia la solicitud, el Administrador de objetos de Visual Studio busca la biblioteca adecuada y solicita datos. Los datos viajan entre las Visual Studio bibliotecas y el gestor [IVsSimpleObjectList2](#) de objetos en listas de símbolos descritos por la interfaz.

El Visual Studio administrador de objetos es responsable de actualizar periódicamente las herramientas de exploración de símbolos para reflejar los datos más actuales contenidos en las bibliotecas.

El diagrama siguiente contiene un ejemplo de elementos clave del proceso de intercambio de solicitudes/datos entre una biblioteca y el administrador de objetos de Visual Studio. Las interfaces del diagrama forman parte de una aplicación de código administrado.



Para proporcionar las listas de símbolos al Administrador de objetos de Visual Studio, [RegisterSimpleLibrary](#) primero debe registrar la biblioteca con el Administrador de objetos de Visual Studio llamando al método. Una vez registrada la biblioteca, el Administrador de objetos de Visual Studio solicita cierta información sobre las capacidades de la biblioteca. Por ejemplo, solicita las marcas de biblioteca y [GetLibFlags2](#) las categorías admitidas mediante una llamada a los métodos y. En algún momento, cuando una de las herramientas solicita datos de esta biblioteca, el [GetList2](#) administrador de objetos solicita la lista de símbolos de nivel superior llamando al método. En respuesta, la biblioteca fabrica una lista de símbolos y la [IVsSimpleObjectList2](#) expone al Administrador de objetos de Visual Studio a través de la interfaz. El Visual Studio administrador de objetos determina cuántos elementos hay en la lista llamando al [GetItemCount](#) método. Todas las solicitudes siguientes se relacionan con un elemento determinado de la lista y proporcionan el número de índice del elemento en cada solicitud. El Administrador de objetos de Visual Studio procede a recopilar la información sobre el [GetCategoryField2](#) tipo, la accesibilidad y otras propiedades del elemento mediante una llamada al método.

Determina el nombre del elemento llamando [GetTextWithOwnership](#) al método y solicita la [GetDisplayData](#) información del icono llamando al método. El icono se muestra a la izquierda del nombre del elemento y representa el tipo del elemento, la accesibilidad y otras propiedades.

El Visual Studio administrador de [GetExpandable3](#) objetos llama al método para determinar si un elemento de lista determinado se puede expandir y tiene elementos secundarios. Si la interfaz de usuario envía una solicitud para expandir un [GetList2](#) elemento, el administrador de objetos solicita la lista secundaria de símbolos llamando al método. El proceso continúa con diferentes partes del árbol que se construyen a petición.

NOTE

Para implementar un proveedor de [IVsLibrary2](#) símbolos de código nativo, use las interfaces y. [IVsObjectList2](#)

Vea también

- [Registro de una biblioteca con el Administrador de objetos](#)
- [Exposición de listas de símbolos proporcionadas por la biblioteca al Administrador de objetos](#)
- [Identificación de símbolos en una biblioteca](#)

Cómo: Registrar una biblioteca con el administrador de objetos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Las herramientas de exploración de símbolos, como Vista de **clases**, **Examinador de objetos**, Navegador de **llamadas** y Buscar resultados de **símbolos**, le permiten ver símbolos en el proyecto o en componentes externos. Los símbolos incluyen espacios de nombres, clases, interfaces, métodos y otros elementos de lenguaje. Las bibliotecas realizan un seguimiento Visual Studio de estos símbolos y los exponen al administrador de objetos que rellena las herramientas con los datos.

El administrador de objetos realiza un seguimiento de todas las bibliotecas disponibles. Cada biblioteca debe registrarse en el administrador de objetos antes de proporcionar símbolos para las herramientas de exploración de símbolos.

Normalmente, se registra una biblioteca cuando se carga un VSPackage. Sin embargo, se puede hacer en otro momento según sea necesario. Anule el registro de la biblioteca cuando el VSPackage se cierra.

Para registrar una biblioteca, utilice el [RegisterLibrary](#) método. Para una biblioteca de [RegisterSimpleLibrary](#) código administrado, utilice el método.

Para anular el registro [UnregisterLibrary](#) de una biblioteca, utilice el método.

Para obtener una referencia al [IVsObjectManager2](#) gestor [SVsObjectManager](#) de objetos, pase el identificador de servicio al `GetService` método.

Registrar y anular el registro de una biblioteca con el administrador de objetos

Para registrar una biblioteca con el administrador de objetos

1. Cree una biblioteca.

```
Private m_CallBrowserLibrary As CallBrowser.Library = Nothing
Private m_nLibraryCookie As UInteger = 0
' Create Library.
m_CallBrowserLibrary = New CallBrowser.Library()
```

```
private CallBrowser.Library m_CallBrowserLibrary = null;
private uint m_nLibraryCookie = 0;
// Create Library.
m_CallBrowserLibrary = new CallBrowser.Library();
```

2. Obtenga una referencia a [IVsObjectManager2](#) un objeto [RegisterSimpleLibrary](#) del tipo y llame al método.

```

Private Sub RegisterLibrary()
    If m_nLibraryCookie <> 0 Then
        Throw New Exception("Library already registered with Object Manager")
    End If

    ' Obtain a reference to IVsObjectManager2 type object.
    Dim objManager As IVsObjectManager2 = TryCast(GetService(GetType(SVsObjectManager)),
IVsObjectManager2)
    If objManager Is Nothing Then
        Throw New NullReferenceException("GetService failed for SVsObjectManager")
    End If

    Try
        Dim hr As Integer = objManager.RegisterSimpleLibrary(m_CallBrowserLibrary, m_nLibraryCookie)
        Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(hr)
    Catch e As Exception
        ' Code to handle any CLS-compliant exception.
        Trace.WriteLine(e.Message)
        Throw
    End Try
End Sub

```

```

private void RegisterLibrary()
{
    if (m_nLibraryCookie != 0)
        throw new Exception("Library already registered with Object Manager");

    // Obtain a reference to IVsObjectManager2 type object.
    IVsObjectManager2 objManager =
        GetService(typeof(SVsObjectManager)) as IVsObjectManager2;
    if (objManager == null)
        throw new NullReferenceException("GetService failed for SVsObjectManager");

    try
    {
        int hr =
            objManager.RegisterSimpleLibrary(m_CallBrowserLibrary,
                                                out m_nLibraryCookie);
        Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(hr);
    }
    catch (Exception e)
    {
        // Code to handle any CLS-compliant exception.
        Trace.WriteLine(e.Message);
        throw;
    }
}

```

Para anular el registro de una biblioteca con el administrador de objetos

1. Obtenga una referencia a [IVsObjectManager2](#) un objeto [UnregisterLibrary](#) del tipo y llame al método.

```

Private Sub UnregisterLibrary()
    If m_nLibraryCookie <> 0 Then
        ' Obtain a reference to IVsObjectManager2 type object.
        Dim objManager As IVsObjectManager2 = TryCast(GetService(GetType(SVsObjectManager)),
IVsObjectManager2)
        If objManager Is Nothing Then
            Throw New NullReferenceException("GetService failed for SVsObjectManager")
        End If

        Try
            objManager.UnregisterLibrary(m_nLibraryCookie)
        Catch e As Exception
            ' Code to handle any CLS-compliant exception.
            Trace.WriteLine(e.Message)
            Throw
        Finally
            m_nLibraryCookie = 0
        End Try
    End If
End Sub

```

```

private void UnregisterLibrary()
{
    if (m_nLibraryCookie != 0)
    {
        // Obtain a reference to IVsObjectManager2 type object.
        IVsObjectManager2 objManager = GetService(typeof(SVsObjectManager)) as IVsObjectManager2;
        if (objManager == null)
            throw new NullReferenceException("GetService failed for SVsObjectManager");

        try
        {
            objManager.UnregisterLibrary(m_nLibraryCookie);
        }
        catch (Exception e)
        {
            // Code to handle any CLS-compliant exception.
            Trace.WriteLine(e.Message);
            throw;
        }
        finally
        {
            m_nLibraryCookie = 0;
        }
    }
}

```

Vea también

- [Extensibilidad del servicio de idiomas heredado](#)
- [Soporta herramientas de navegación de símbolos](#)
- [Cómo: Exponer listas de símbolos proporcionados por la biblioteca al gestor de objetos](#)

Cómo: Exponer listas de símbolos proporcionados por la biblioteca al gestor de objetos

06/04/2020 • 10 minutes to read • [Edit Online](#)

Las herramientas de exploración de símbolos, Vista de **clases**, Examinador de Visual Studio **objetos**, Navegador de **llamadas** y Buscar resultados de **símbolos**, pasan solicitudes de nuevos datos al administrador de objetos. El administrador de objetos busca las bibliotecas adecuadas y solicita nuevas listas de símbolos. Las bibliotecas responden proporcionando Visual Studio los datos [IVsSimpleObjectList2](#) solicitados al administrador de objetos a través de la interfaz. El Visual Studio administrador de objetos llama a los métodos de la [IVsSimpleObjectList2](#) interfaz para obtener los datos y los utiliza para rellenar o actualizar las vistas de las herramientas de exploración de símbolos.

Una biblioteca puede obtener solicitudes de datos cuando se invoca la herramienta, se expande el nodo o se actualiza la vista. Cuando se invoca una herramienta de exploración de símbolos por primera vez, el administrador de objetos solicita a la biblioteca que proporcione la lista de nivel superior. Cuando el usuario expande un nodo de lista, la biblioteca proporciona una lista de elementos secundarios bajo ese nodo. Cada consulta del administrador de objetos contiene un índice del elemento de interés. Para mostrar una nueva lista, el administrador de objetos debe determinar cuántos elementos hay en la lista, el tipo de los elementos, sus nombres, accesibilidad y otras propiedades.

NOTE

Los siguientes ejemplos de código administrado muestran [IVsSimpleObjectList2](#) cómo proporcionar listas de símbolos mediante la implementación de la interfaz. El administrador de objetos llama a los métodos de esta interfaz y utiliza los datos obtenidos para rellenar o actualizar las herramientas de exploración de símbolos.

Para la implementación del [IVsObjectList2](#) proveedor de símbolos de código nativo, use la interfaz.

Para proporcionar listas de símbolos al gestor de objetos

1. Obtener el número de elementos de [GetItemCount](#) la lista de símbolos mediante la implementación del método. En el ejemplo siguiente se muestra cómo el administrador de objetos obtiene la información sobre el número de elementos de la lista.

```
Protected m_Methods As System.Collections.Generic.SortedList(Of String, Method) = New
System.Collections.Generic.SortedList(Of String, Method)()

Public Function GetItemCount(ByRef pCount As UInteger) As Integer
    pCount = CUInt(m_Methods.Count)
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function
```

```
protected System.Collections.Generic.SortedList<string, Method> m_Methods = new
System.Collections.Generic.SortedList<string, Method>();

public int GetItemCount(out uint pCount)
{
    pCount = (uint)m_Methods.Count;
    return Microsoft.VisualStudio.VSConstants.S_OK;
}
```

2. Obtenga información sobre las categorías y los atributos [GetCategoryField2](#) de un elemento de lista determinado implementando el método. Las categorías de elementos [LIB_CATEGORY](#) se especifican en la enumeración. En el ejemplo siguiente se muestra cómo el administrador de objetos obtiene atributos de elementos para una categoría determinada.

```
Public Function GetCategoryField2(ByVal index As UInteger, ByVal Category As Integer, ByRef pfCatField
As UInteger) As Integer
    pfCatField = 0

    Select Case CType(Category, LIB_CATEGORY)
        Case LIB_CATEGORY.LC_MEMBERTYPE
            pfCatField = CUInt(_LIBCAT_MEMBERTYPE.LCMT_METHOD)

        Case LIB_CATEGORY.LC_MEMBERACCESS
            Dim method As Method = m_Methods.Values(CInt(Fix(index)))

            If method.IsPublic Then
                pfCatField = CUInt(_LIBCAT_MEMBERACCESS.LCMA_PUBLIC)
            ElseIf method.IsPrivate Then
                pfCatField = CUInt(_LIBCAT_MEMBERACCESS.LCMA_PRIVATE)
            ElseIf method.IsFamily Then
                pfCatField = CUInt(_LIBCAT_MEMBERACCESS.LCMA_PROTECTED)
            ElseIf method.IsFamilyOrAssembly Then
                pfCatField = CUInt(_LIBCAT_MEMBERACCESS.LCMA_PROTECTED) Or
CUInt(_LIBCAT_MEMBERACCESS.LCMA_PACKAGE)
            Else
                ' Show everything else as internal.
                pfCatField = CUInt(_LIBCAT_MEMBERACCESS.LCMA_PACKAGE)
            End If

        Case LIB_CATEGORY.LC_VISIBILITY
            pfCatField = CUInt(_LIBCAT_VISIBILITY.LCV_VISIBLE)

        Case LIB_CATEGORY.LC_LISTTYPE
            pfCatField = CUInt(_LIB_LISTTYPE.LLT_MEMBERS)

        Case Else
            Return Microsoft.VisualStudio.VSConstants.S_FALSE
    End Select
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function
```



```

public int GetCategoryField2(uint index, int Category, out uint pfCatField)
{
    pfCatField = 0;

    switch ((LIB_CATEGORY)Category)
    {
        case LIB_CATEGORY.LC_MEMBERTYPE:
            pfCatField = (uint)_LIBCAT_MEMBERTYPE.LCMT_METHOD;
            break;

        case LIB_CATEGORY.LC_MEMBERACCESS:
            {
                Method method = m_Methods.Values[(int)index];

                if (method.IsPublic)
                {
                    pfCatField = (uint)_LIBCAT_MEMBERACCESS.LCMA_PUBLIC;
                }
                else if (method.IsPrivate)
                {
                    pfCatField = (uint)_LIBCAT_MEMBERACCESS.LCMA_PRIVATE;
                }
                else if (method.IsFamily)
                {
                    pfCatField = (uint)_LIBCAT_MEMBERACCESS.LCMA_PROTECTED;
                }
                else if (method.IsFamilyOrAssembly)
                {
                    pfCatField = (uint)_LIBCAT_MEMBERACCESS.LCMA_PROTECTED |
                        (uint)_LIBCAT_MEMBERACCESS.LCMA_PACKAGE;
                }
                else
                {
                    // Show everything else as internal.
                    pfCatField = (uint)_LIBCAT_MEMBERACCESS.LCMA_PACKAGE;
                }
            }
            break;

        case LIB_CATEGORY.LC_VISIBILITY:
            pfCatField = (uint)_LIBCAT_VISIBILITY.LCV_VISIBLE;
            break;

        case LIB_CATEGORY.LC_LISTTYPE:
            pfCatField = (uint)_LIB_LISTTYPE.LLT_MEMBERS;
            break;

        default:
            return Microsoft.VisualStudio.VSConstants.S_FALSE;
    }
    return Microsoft.VisualStudio.VSConstants.S_OK;
}

```

3. Obtener la representación de texto de [GetTextWithOwnership](#) un elemento de lista determinado mediante la implementación del método. En el ejemplo siguiente se muestra cómo obtener un nombre completo de un elemento determinado.

```

Public Function
GetTextWithOwnership(<System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.OLE.
Interop.ULONG")> ByVal index As UInteger,
<System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.Shell.Interop.VSTREETEXTO
PTIONS")> ByVal tto As Microsoft.VisualStudio.Shell.Interop.VSTREETEXTOPTIONS,
<System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.OLE.Interop.WCHAR")>
ByRef ppszText As String) As Integer
    ppszText = m_Methods.Values(CInt(Fix(index))).FullName
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function

```

```

public int
GetTextWithOwnership([System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.OLE.
Interop.ULONG")] uint index,
[System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.Shell.Interop.VSTREETEXTO
PTIONS")] Microsoft.VisualStudio.Shell.Interop.VSTREETEXTOPTIONS tto,
[System.Runtime.InteropServices.ComAliasNameAttribute("Microsoft.VisualStudio.OLE.Interop.WCHAR")] out
string ppszText)
{
    ppszText = m_Methods.Values[(int)index].FullName;
    return Microsoft.VisualStudio.VSConstants.S_OK;
}

```

4. Obtener la información de icono para un [GetDisplayData](#) elemento de lista determinado mediante la implementación del método. El icono representa el tipo (clase, método, etc.) y la accesibilidad (privado, público, etc.) de un elemento de lista. En el ejemplo siguiente se muestra cómo obtener la información de icono basada en atributos de elemento determinados.

```

Public Overridable Function GetDisplayData(ByVal index As UInteger, ByVal pData As
Microsoft.VisualStudio.Shell.Interop.VSTREEDISPLAYDATA()) As Integer
    If pData Is Nothing Then
        Return Microsoft.VisualStudio.VSConstants.E_INVALIDARG
    End If

    Dim method As Method = m_Methods.Values(CInt(Fix(index)))

    Dim iImage As Integer = 12 * 6 ' See env\inc\OMGlyphs.h.

    Const OM_GLYPH_ACC_PUBLIC As Integer = 0
    Const OM_GLYPH_ACC_INTERNAL As Integer = 1
    Const OM_GLYPH_ACC_PROTECTED As Integer = 3
    Const OM_GLYPH_ACC_PRIVATE As Integer = 4

    If method.IsPublic Then
        iImage += OM_GLYPH_ACC_PUBLIC
    ElseIf method.IsPrivate Then
        iImage += OM_GLYPH_ACC_PRIVATE
    ElseIf method.IsFamily Then
        iImage += OM_GLYPH_ACC_PROTECTED
    ElseIf method.IsFamilyOrAssembly Then
        iImage += OM_GLYPH_ACC_PROTECTED
    Else
        iImage += OM_GLYPH_ACC_INTERNAL
    End If

    pData(0).Image = CUShort(iImage)
    pData(0).SelectedImage = CUShort(iImage)

    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function

```

```

public virtual int GetDisplayData(uint index, Microsoft.VisualStudio.Shell.Interop.VSTREEDISPLAYDATA[]
pData)
{
    if (pData == null)
    {
        return Microsoft.VisualStudio.VSConstants.E_INVALIDARG;
    }

    Method method = m_Methods.Values[(int)index];

    int iImage = 12 * 6;    // See env\inc\OMGlyphs.h.

    const int OM_GLYPH_ACC_PUBLIC = 0;
    const int OM_GLYPH_ACC_INTERNAL = 1;
    const int OM_GLYPH_ACC_PROTECTED = 3;
    const int OM_GLYPH_ACC_PRIVATE = 4;

    if (method.IsPublic)
    {
        iImage += OM_GLYPH_ACC_PUBLIC;
    }
    else if (method.IsPrivate)
    {
        iImage += OM_GLYPH_ACC_PRIVATE;
    }
    else if (method.IsFamily)
    {
        iImage += OM_GLYPH_ACC_PROTECTED;
    }
    else if (method.IsFamilyOrAssembly)
    {
        iImage += OM_GLYPH_ACC_PROTECTED;
    }
    else
    {
        iImage += OM_GLYPH_ACC_INTERNAL;
    }

    pData[0].Image = (ushort)iImage;
    pData[0].SelectedImage = (ushort)iImage;

    return Microsoft.VisualStudio.VSConstants.S_OK;
}

```

5. Obtenga la información sobre si un elemento de [GetExpandable3](#) lista determinado se puede expandir implementando el método. En el ejemplo siguiente se muestra cómo obtener la información sobre si se puede expandir un elemento determinado.

```

Public Function GetExpandable(ByVal index As UInteger, ByRef pfExpandable As Integer) As Integer
    pfExpandable = Microsoft.VisualStudio.VSIP.Samples.CallBrowser.Constants.TRUE
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function

Public Function GetExpandable3(ByVal index As UInteger, ByVal ListTypeExcluded As UInteger, ByRef
pfExpandable As Integer) As Integer
    Return GetExpandable(index, pfExpandable)
End Function

```

```

public int GetExpandable(uint index, out int pfExpandable)
{
    pfExpandable = Microsoft.VisualStudio.VSIP.Samples.CallBrowser.Constants.TRUE;
    return Microsoft.VisualStudio.VSConstants.S_OK;
}

public int GetExpandable3(uint index, uint ListTypeExcluded, out int pfExpandable)
{
    return GetExpandable(index, out pfExpandable);
}

```

6. Obtener una lista secundaria de símbolos de [GetList2](#) un elemento de lista determinado mediante la implementación del método. En el ejemplo siguiente se muestra cómo obtener una lista secundaria de símbolos de un elemento determinado para los gráficos **Call** o **Callers**.

```

' Call graph list.
Public Class CallsList
    Inherits ResultsList
    Implements Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
    Public Sub New(ByVal library As Library)
        MyBase.New(library)
    End Sub

    Public Function GetList2(ByVal index As UInteger, ByVal ListType As UInteger, ByVal flags As
UInteger, ByVal pobSrch As Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2(), ByRef ppList As
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2) As Integer
        Return MyBase.GetCallsList(index, ppList)
    End Function
End Class

' Callers graph list.
Public Class CallersList
    Inherits ResultsList
    Implements Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
    Public Sub New(ByVal library As Library)
        MyBase.New(library)
    End Sub

    Public Function GetList2(ByVal index As UInteger, ByVal ListType As UInteger, ByVal flags As
UInteger, ByVal pobSrch As Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2(), ByRef ppList As
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2) As Integer
        Return MyBase.GetCallersList(index, ppList)
    End Function
End Class

' Call graph list.
Public Function GetCallsList(ByVal index As UInteger, ByRef ppList As
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2) As Integer
    ppList = Nothing
    Dim callsList As ResultsList = New CallsList(m_Library)

    Dim method As Method = m_Methods.Values(CInt(Fix(index)))
    Dim strMethod As String = method.m_strPrototype

    Dim Calls As System.Collections.Generic.List(Of CallInstance) =
m_Library.CallGraph.GetCallGraph(method)

    For i As Integer = 0 To Calls.Count - 1
        Dim caller As Method = Calls(i).m_Target
        callsList.AddMethod(caller)
    Next i

    ppList = CType(callsList, Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2)
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function

```

```

End Function

' Callers graph list.
Public Function GetCallersList(ByVal index As UInteger, ByRef ppList As
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2) As Integer
    ppList = Nothing
    Dim callersList As ResultsList = New CallersList(m_Library)

    Dim method As Method = m_Methods.Values(CInt(Fix(index)))
    Dim strMethod As String = method.m_strPrototype

    Dim Callers As System.Collections.Generic.List(Of CallInstance) =
m_Library.CallGraph.GetCallersGraph(method)

    For i As Integer = 0 To Callers.Count - 1
        Dim caller As Method = Callers(i).m_Source
        callersList.AddMethod(caller)
    Next i

    ppList = CType(callersList, Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2)
    Return Microsoft.VisualStudio.VSConstants.S_OK
End Function

' Get a child list of symbols for a given list item.
Public Function GetList2(ByVal index As UInteger, ByVal ListType As UInteger, ByVal flags As UInteger,
ByVal pobSrCh As Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2(), ByRef ppList As
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2) As Integer
    ppList = Nothing

    Dim method As Method = m_Methods.Values(CInt(Fix(index)))
    Dim strMethod As String = method.m_strPrototype

    ' Determine if the list belongs to Call or Callers graphs.
    If CUInt(m_nFlags And
CUInt(Microsoft.VisualStudio.Shell.Interop._VSOBSEARCHOPTIONS2.VSOBSO_CALLSFROM)) > 0 Then
        ' Build the list for the Call graph.
        Return MyBase.GetCallsList(index, ppList)
    ElseIf CUInt(m_nFlags And
CUInt(Microsoft.VisualStudio.Shell.Interop._VSOBSEARCHOPTIONS2.VSOBSO_CALLSTO)) > 0 Then
        ' Build the list for the Callers graph.
        Return MyBase.GetCallersList(index, ppList)
    End If

    Return Microsoft.VisualStudio.VSConstants.E_FAIL
End Function

```

```

// Call graph list.
public class CallsList :
    ResultsList,
    Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
{
    public CallsList(Library library) :
        base(library)
    {
    }

    public int GetList2(uint index, uint ListType, uint flags,
Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2[] pobSrCh, out
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2 ppList)
    {
        return base.GetCallsList(index, out ppList);
    }
}

// Callers graph list.
public class CallersList :
    ResultsList,
    Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
{
    public CallersList(Library library) :
        base(library)
    {
    }

    public int GetList2(uint index, uint ListType, uint flags,
Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2[] pobSrCh, out
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2 ppList)
    {
        return base.GetCallersList(index, out ppList);
    }
}

```

```

Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
{
    public CallersList(Library library) :
        base(library)
    {
    }

    public int GetList2(uint index, uint ListType, uint flags,
Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2[] pobSrch, out
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2 pplList)
    {
        return base.GetCallersList(index, out pplList);
    }
}

// Call graph list.
public int GetCallsList(uint index, out Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
pplList)
{
    pplList = null;
    ResultsList callsList = new CallsList(m_Library);

    Method method = m_Methods.Values[(int)index];
    string strMethod = method.m_strPrototype;

    System.Collections.Generic.List<CallInstance> Calls = m_Library.CallGraph.GetCallGraph(method);

    for (int i = 0; i < Calls.Count; i++)
    {
        Method caller = Calls[i].m_Target;
        callsList.AddMethod(caller);
    }

    pplList = (Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2)(callsList);
    return Microsoft.VisualStudio.VSConstants.S_OK;
}

// Callers graph list.
public int GetCallersList(uint index, out Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2
pplList)
{
    pplList = null;
    ResultsList callersList = new CallersList(m_Library);

    Method method = m_Methods.Values[(int)index];
    string strMethod = method.m_strPrototype;

    System.Collections.Generic.List<CallInstance> Callers =
m_Library.CallGraph.GetCallersGraph(method);

    for (int i = 0; i < Callers.Count; i++)
    {
        Method caller = Callers[i].m_Source;
        callersList.AddMethod(caller);
    }

    pplList = (Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2)(callersList);
    return Microsoft.VisualStudio.VSConstants.S_OK;
}

// Get a child list of symbols for a given list item.
public int GetList2(uint index, uint ListType, uint flags,
Microsoft.VisualStudio.Shell.Interop.VSOBSEARCHCRITERIA2[] pobSrch, out
Microsoft.VisualStudio.Shell.Interop.IVsSimpleObjectList2 pplList)
{
    pplList = null;

    Method method = m_Methods.Values[(int)index];
    string strMethod = method.m_strPrototype;

```

```

        // Determine if the list belongs to Call or Callers graphs.
        if ((uint)(m_nFlags &
(uint)Microsoft.VisualStudio.Shell.Interop._VSOBSEARCHOPTIONS2.VSOBSO_CALLSFROM) > 0)
        {
            // Build the list for the Call graph.
            return base.GetCallsList(index, out ppList);
        }
        else if ((uint)(m_nFlags &
(uint)Microsoft.VisualStudio.Shell.Interop._VSOBSEARCHOPTIONS2.VSOBSO_CALLSTO) > 0)
        {
            // Build the list for the Callers graph.
            return base.GetCallersList(index, out ppList);
        }

        return Microsoft.VisualStudio.VSConstants.E_FAIL;
    }

```

Vea también

- [Soporta herramientas de navegación de símbolos](#)
- [Cómo: Registrar una biblioteca con el administrador de objetos](#)
- [Cómo: Identificar símbolos en una biblioteca](#)
- [Extensibilidad del servicio de idiomas heredado](#)

Cómo: Identificar símbolos en una biblioteca

06/04/2020 • 5 minutes to read • [Edit Online](#)

Las herramientas de exploración de símbolos muestran vistas jerárquicas de símbolos. Los símbolos representan espacios de nombres, objetos, clases, miembros de clase y otros elementos de lenguaje.

Cada símbolo de la jerarquía se puede identificar mediante la Visual Studio información de navegación que pasa la biblioteca de símbolos al gestor de objetos a través de las interfaces siguientes:

- [IVsNavInfo](#)
- [IVsNavInfoNode](#)
- [IVsEnumNavInfoNodes](#).

La ubicación del símbolo en la jerarquía distingue un símbolo. Permite que las herramientas de exploración de símbolos naveguen a un símbolo específico. La ruta de acceso única y completa al símbolo determina la ubicación. Cada elemento de la ruta de acceso es un nodo. La ruta de acceso comienza con el nodo de nivel superior y termina con el símbolo específico. Por ejemplo, si el método M1 es miembro de la clase C1 y C1 está en el espacio de nombres N1, la ruta de acceso completa del método M1 es N1. C1. M1. Esta trayectoria contiene tres Nodos: N1, C1 y M1.

La información de Visual Studio navegación permite al administrador de objetos localizar, seleccionar y mantener seleccionados los símbolos de la jerarquía. Permite navegar de una herramienta de navegación a otra. Al utilizar el **Examinador** de Visual C++ objetos para examinar símbolos en un proyecto, puede hacer clic con el botón derecho en un método e iniciar la herramienta Explorador de **llamadas** para mostrar el método en un gráfico de llamadas.

Dos formularios describen la ubicación del símbolo. La forma canónica se basa en la ruta completa del símbolo. Representa una posición única del símbolo en la jerarquía. Es independiente de la herramienta de exploración de símbolos. Para obtener la información Visual Studio del formulario [EnumCanonicalNodes](#) canónico, el administrador de objetos llama al método. El formulario de presentación describe la ubicación del símbolo dentro de una herramienta de exploración de símbolos específica. La posición del símbolo es relativa a la posición de otros símbolos en la jerarquía. Un símbolo dado puede tener varias rutas de presentación, pero solo una ruta canónica. Por ejemplo, si la clase C1 se hereda de la clase C2 y ambas clases están en el espacio de nombres N1, el **Examinador de objetos** muestra el siguiente árbol jerárquico:

```
N1
  C1
    Bases and Interfaces
      C2
        Bases and Interfaces
        . . . . .
```

La ruta canónica de la clase C2, en este ejemplo, es N1 + C2. La ruta de presentación de C2 incluye nodos C1 y "Bases e Interfaces": N1 + C1 + "Bases e Interfaces" + C2.

Para obtener la información del formulario [EnumPresentationNodes](#) de presentación, el administrador de objetos llama al método.

Para obtener información canónica y de formularios de presentación

1. Implemente el método [EnumCanonicalNodes](#).

El administrador de objetos llama a este método para obtener la lista de nodos contenidos en la ruta canónica del símbolo.

```
Public Function EnumCanonicalNodes(ByRef ppEnum As
Microsoft.VisualStudio.Shell.Interop.IVsEnumNavInfoNodes) As Integer
    Dim EnumNavInfoNodes As CallBrowserEnumNavInfoNodes = _New CallBrowserEnumNavInfoNodes(m_strMethod)
    ppEnum = CType(EnumNavInfoNodes, IVsEnumNavInfoNodes)
    Return 0
End Function
```

```
public int EnumCanonicalNodes(out Microsoft.VisualStudio.Shell.Interop.IVsEnumNavInfoNodes ppEnum)
{
    CallBrowserEnumNavInfoNodes EnumNavInfoNodes =
        new CallBrowserEnumNavInfoNodes(m_strMethod);
    ppEnum = (IVsEnumNavInfoNodes)(EnumNavInfoNodes);
    return 0;
}
```

2. Implemente el método [EnumPresentationNodes](#).

El administrador de objetos llama a este método para obtener la lista de nodos contenidos en la ruta de presentación del símbolo.

Vea también

- [Soporta herramientas de navegación de símbolos](#)
- [Cómo: Registrar una biblioteca con el administrador de objetos](#)
- [Cómo: Exponer listas de símbolos proporcionados por la biblioteca al gestor de objetos](#)

Proyectos

06/04/2020 • 3 minutes to read • [Edit Online](#)

En Visual Studio, los proyectos son los contenedores que los desarrolladores usan para organizar los archivos de código fuente y otros recursos que aparecen en el Explorador de **soluciones**. Normalmente, los proyectos son archivos (por ejemplo, un archivo .csproj para un proyecto de C) que almacenan referencias a archivos de código fuente y recursos como archivos de mapa de bits. Los proyectos le permiten organizar, compilar, depurar e implementar código fuente, referencias a servicios web y bases de datos y otros recursos. VSPackages puede ampliar el sistema de proyectos de Visual Studio de tres maneras principales: tipos de *proyecto*, *subtipos* de proyecto y *herramientas personalizadas*.

En esta sección

- [Tipos de proyecto](#)

Los *tipos* de proyecto agregan compatibilidad con nuevos tipos de proyectos, como lenguajes de programación. Por ejemplo, cada lenguaje que admite Visual Studio tiene su propio tipo de proyecto y el ejemplo de integración de IronPython incluye un tipo de proyecto para el lenguaje IronPython. Debe crear un tipo de proyecto para los lenguajes que no sean de C o Visual Basic para personalizar cómo se compilan, depuran, implementan y muestran los elementos en el **Explorador** de soluciones. Para obtener más información, vea [Tipos de proyecto](#).

- [Subtipos de proyecto](#)

Los *subtipos* de proyecto se basan en tipos de proyecto y se pueden usar para personalizar la forma en que se compilan, depuran e implementan los proyectos. Visual Studio usa subtipos de proyecto con proyectos de dispositivo inteligente; personalizan la implementación copiando un programa de nueva construcción desde un equipo de desarrollo al dispositivo de destino. Los tipos de proyecto de C- y Visual Basic se pueden usar como base para los subtipos de proyecto; Los tipos de proyecto C++ no pueden. Sus propios tipos de proyecto también se pueden utilizar como base para subtipos de proyecto. Para obtener más información, vea [Subtipos de proyecto](#).

- [Proyectos web](#)

Explica el proyecto Web, que a su vez crea aplicaciones web.

- [Nueva generación de proyectos: Bajo la capucha, primera parte](#) y nueva generación de [proyectos: Bajo la capucha, segunda parte](#)

Explica lo que realmente ocurre cuando se crea un nuevo proyecto.

- [Muestras VSSDK](#) Contiene los ejemplos del VSSDK que tratan con proyectos y soluciones.

Secciones relacionadas

- [Dentro de Visual Studio SDK](#)

Explicar diferentes aspectos de la extensibilidad de Visual Studio.

Nueva generación de proyectos: aspectos técnicos, primera parte

06/04/2020 • 15 minutes to read • [Edit Online](#)

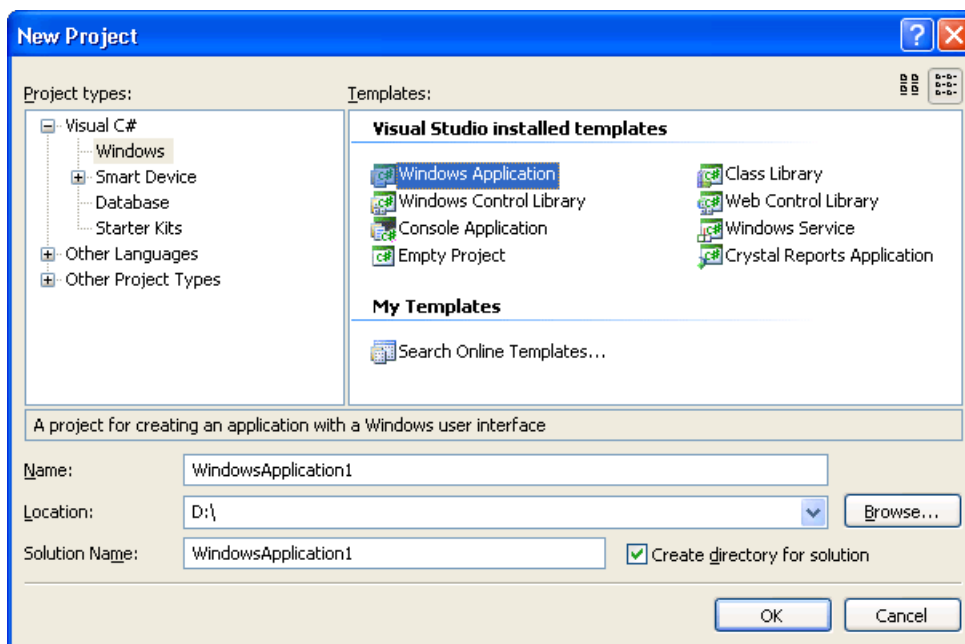
¿Alguna vez has pensado en cómo crear tu propio tipo de proyecto? ¿Te preguntas qué sucede realmente cuando creas un nuevo proyecto? Echemos un vistazo debajo del capó y veamos qué está pasando realmente.

Hay varias tareas que Visual Studio coordina automáticamente:

- Muestra un árbol de todos los tipos de proyecto disponibles.
- Muestra una lista de plantillas de aplicación para cada tipo de proyecto y le permite elegir una.
- Recopila información del proyecto para la aplicación, como el nombre del proyecto y la ruta de acceso.
- Pasa esta información a la fábrica del proyecto.
- Genera elementos y carpetas de proyecto en la solución actual.

El cuadro de diálogo Nuevo proyecto

Todo comienza cuando se selecciona un tipo de proyecto para un nuevo proyecto. Comencemos haciendo clic en **Nuevo proyecto** en el menú **Archivo**. Aparece el cuadro de diálogo **Nuevo proyecto**, con un aspecto similar al siguiente:



Veámoslo con más detalle. El árbol **Tipos** de proyecto enumera los distintos tipos de proyecto que puede crear. Cuando seleccione un tipo de proyecto como **Visual C- Windows**, verá una lista de plantillas de aplicación para empezar. **Visual Studio** instala las plantillas instaladas en Visual Studio y están disponibles para cualquier usuario del equipo. Las nuevas plantillas que cree o recopile se pueden agregar a **Mis plantillas** y solo están disponibles para usted.

Al seleccionar una plantilla como Aplicación de **Windows**, aparece una descripción del tipo de aplicación en el cuadro de diálogo; en este caso, **un proyecto para crear una aplicación con una interfaz de usuario de Windows**.

En la parte inferior del cuadro de diálogo **Nuevo proyecto**, verá varios controles que recopilan más información. Los controles que ve dependen del tipo de proyecto, pero generalmente incluyen un cuadro de texto **Nombre** del proyecto, un cuadro de texto **Ubicación** y el botón **Examinar** relacionado y un cuadro de texto Nombre de **solución** y una casilla de verificación **Crear directorio para solución** relacionada.

Rellenar el cuadro de diálogo Nuevo proyecto

¿De dónde obtiene el cuadro de diálogo **Nuevo proyecto** su información? Hay dos mecanismos en el trabajo aquí, uno de ellos en desuso. El cuadro de diálogo **Nuevo proyecto** combina y muestra la información obtenida de ambos mecanismos.

El método anterior (obsoleto) utiliza entradas del registro del sistema y archivos .vsdir. Este mecanismo se ejecuta cuando se abrió Visual Studio. El método más reciente utiliza archivos .vstemplate. Este mecanismo se ejecuta cuando Visual Studio se inicializa, por ejemplo, ejecutando

```
devenv /setup
```

or

```
devenv /installvstemplates
```

Tipos de proyecto

La posición y los nombres de los nodos raíz de los tipos de **proyecto**, como **Visual C-** y **otros lenguajes**, se determina mediante entradas del registro del sistema. La organización de los nodos secundarios, como **Base de datos** y **Dispositivo inteligente**, refleja la jerarquía de las carpetas que contienen los archivos .vstemplate correspondientes. Echemos un vistazo a los nodos raíz primero.

Nodos raíz de tipo de proyecto

Cuando Visual Studio se inicializa, recorre las subclaves de la clave del Registro del sistema HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 14,0, NewProjectTemplates, TemplateDirs para compilar y asignar un nombre a los nodos raíz del árbol de tipos de **proyecto**. Esta información se almacena en caché para su uso posterior. Observe la clave\TemplateDirs .FAE04EC1-301F-11D3-BF4B-00C04F79EFBC.\ Cada entrada es un GUID de VSPackage. El nombre de la subclave (/1) se omite, pero su presencia indica que se trata de un nodo raíz de tipos de **proyecto**. Un nodo raíz puede, a su vez, tener varias subclaves que controlan su apariencia en el árbol Tipos de **proyecto**. Echemos un vistazo a algunos de ellos.

(Es el valor predeterminado).

Este es el identificador de recurso de la cadena localizada que nombra el nodo raíz. El recurso de cadena se encuentra en el archivo DLL satélite seleccionado por el GUID de VSPackage.

En el ejemplo, el GUID de VSPackage es

•FAE04EC1-301F-11D3-BF4B-00C04F79EFBC

y el ID de recurso (valor predeterminado) del nodo raíz (/1) es #2345

Si busca el GUID en la clave Packages cercana y examina la subclave SatelliteDll, puede encontrar la ruta de acceso del ensamblado que contiene el recurso de cadena:

<Ruta de instalación de>, VC, VSPackages, 1033, csprojui.dll

Para comprobarlo, abra el Explorador de archivos y arrastre csprojui.dll al directorio de Visual Studio. La tabla de cadenas muestra que el #2345 de recursos tiene el título **visual**.

SortPriority

Esto determina la posición del nodo raíz en el árbol Tipos de **proyecto**.

SortPriority REG_DWORD 0x00000014 (20)

Cuanto menor sea el número de la prioridad, mayor será la posición en el árbol.

DeveloperActivity

Si esta subclave está presente, la posición del nodo raíz se controla mediante el cuadro de diálogo Configuración del desarrollador. Por ejemplo,

DeveloperActivity REG_SZ VC #

indica que Visual C. será un nodo raíz si Visual C++ Visual Studio está configurado para el desarrollo. De lo contrario, será un nodo secundario de **Otros idiomas**.

Carpeta

Si esta subclave está presente, el nodo raíz se convierte en un nodo secundario de la carpeta especificada. Aparece una lista de posibles carpetas bajo la clave

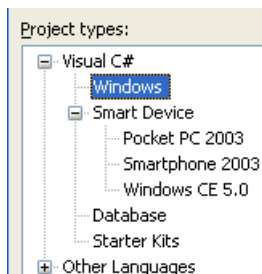
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\11.0\NewProjectTemplates\PseudoFolders

Por ejemplo, la entrada Proyectos de base de datos tiene una clave Carpeta que coincide con la entrada Otros tipos de proyecto en PseudoFolders. Por lo tanto, en el árbol **Tipos de proyecto**, Proyectos de **base de datos** será un nodo secundario de Otros tipos de **proyecto**.

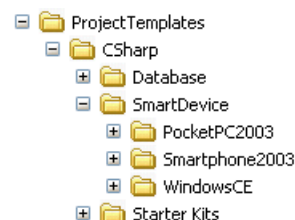
Nodos secundarios de tipo de proyecto y archivos .vstidir

La posición de los nodos secundarios en el árbol **Tipos de proyecto** sigue la jerarquía de las carpetas de las carpetas ProjectTemplates. Para las plantillas de máquina (**plantillas instaladas** de Visual Studio), la ubicación típica es "Archivos de programa" (Archivos de programa), Microsoft Visual Studio 14.0 (Common7)\y "Plantillas de usuario" (**Mis plantillas**), la ubicación típica es "Mis documentos", "Visual Studio 14.0", "Templates" y "ProjectTemplates". Las jerarquías de carpetas de estas dos ubicaciones se combinan para crear el árbol **Tipos de proyecto**.

Para la configuración de desarrollador de Visual Studio con C, el árbol **de tipos de proyecto** tiene un aspecto similar al siguiente:



La carpeta ProjectTemplates correspondiente tiene este aspecto:



Cuando se abre el Visual Studio cuadro de diálogo **Nuevo proyecto**, recorre la carpeta ProjectTemplates y vuelve a crear su estructura en el árbol **Tipos de proyecto** con algunos cambios:

- El nodo raíz del árbol **Tipos de proyecto** viene determinado por la plantilla de aplicación.
- El nombre del nodo se puede localizar y puede contener caracteres especiales.
- El criterio de ordenación se puede cambiar.

Encontrar el nodo raíz para un tipo de proyecto

Cuando Visual Studio recorre las carpetas ProjectTemplates, abre todos los archivos .zip y extrae los archivos .vstemplate. Un archivo .vstemplate utiliza XML para describir una plantilla de aplicación. Para obtener más información, consulte Nueva generación de [proyectos: Bajo la capucha, segunda parte](#).

La <etiqueta de> ProjectType determina el tipo de proyecto de la aplicación. Por ejemplo, el archivo .CSharp-SmartDevice-WindowsCE-1033-WindowsCE-EmptyProject.zip contiene un archivo EmptyProject.vstemplate que tiene esta etiqueta:

```
<ProjectType>CSharp</ProjectType>
```

El <ProjectType> etiqueta y no la subcarpeta de la carpeta ProjectTemplates, determina el nodo raíz de una aplicación en el árbol Tipos de **proyecto**. En el ejemplo, las aplicaciones de Windows CE aparecen bajo el nodo raíz de **Visual C**, e incluso si se va a mover la carpeta WindowsCE a la carpeta VisualBasic, las aplicaciones de Windows CE seguirían apareciendo en el nodo raíz de **Visual C**.

Localización del nombre del nodo

Cuando Visual Studio recorre las carpetas ProjectTemplates, examina los archivos .vstidir que encuentra. Un archivo .vstidir es un archivo XML que controla el aspecto del tipo de proyecto en el cuadro de diálogo **Nuevo proyecto**. En el archivo .vstidir, use la <etiqueta de> LocalizedName para asignar un nombre al nodo Tipos de **proyecto**.

Por ejemplo, el archivo .CSharp-Database-TemplateIndex.vstidir contiene esta etiqueta:

```
<LocalizedName Package="{462b036f-7349-4835-9e21-bec60e989b9c}" ID="4598"/>
```

Esto determina el archivo DLL satélite y el identificador de recurso de la cadena localizada que nombra el nodo raíz, en este caso, Base de **datos**. El nombre localizado puede contener caracteres especiales que no están disponibles para los nombres de carpeta, como **.NET**.

Si <no hay ninguna etiqueta de> LocalizedName, el tipo de proyecto se denomina mediante la propia carpeta, **SmartPhone2003**.

Búsqueda del criterio de ordenación para un tipo de proyecto

Para determinar el criterio de ordenación del tipo <de proyecto, los archivos .vstidir usan la etiqueta de> SortOrder.

Por ejemplo, el archivo .CSharp-Windows-Windows.vstidir contiene esta etiqueta:

```
<SortOrder>5</SortOrder>
```

El archivo .CSharp-Database-TemplateIndex.vstidir tiene una etiqueta con un valor mayor:

```
<SortOrder>5000</SortOrder>
```

Cuanto menor sea <el número de la etiqueta SortOrder>, mayor será la posición en el árbol, por lo que el nodo **Windows** aparece más alto que el nodo Base de **datos** en el árbol Tipos de **proyecto**.

Si <no se especifica ninguna etiqueta de> SortOrder para un tipo <de proyecto, aparece en orden alfabético siguiendo los tipos de proyecto que contienen Especificaciones de> SortOrder.

Tenga en cuenta que no hay archivos .vstidir en las carpetas Mis documentos (**Mis plantillas**). Los nombres de tipo de proyecto de aplicación de usuario no están localizados y aparecen en orden alfabético.

Una revisión rápida

Vamos a modificar el cuadro de diálogo **Nuevo proyecto** y crear una nueva plantilla de proyecto de usuario.

1. Agregue una subcarpeta MyProjectNode a la carpeta de archivos de programa de Microsoft Visual Studio 14.0, Common7, IDE, ProjectTemplates y CSharp.
2. Cree un archivo MyProject.vstidir en la carpeta MyProjectNode con cualquier editor de texto.
3. Agregue estas líneas al archivo .vstidir:

```
<TemplateDir Version="1.0.0">  
  <SortOrder>6</SortOrder>  
</TemplateDir>
```

4. Guarde y cierre el archivo .vstidir.
5. Cree un archivo MyProject.vstemplate en la carpeta MyProjectNode con cualquier editor de texto.
6. Agregue estas líneas al archivo .vstemplate:

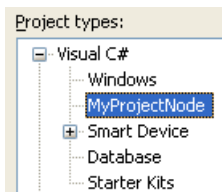
```
<VSTemplate Version="2.0.0" Type="Project"  
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">  
  <TemplateData>  
    <ProjectType>CSharp</ProjectType>  
  </TemplateData>  
</VSTemplate>
```

7. Guarde el archivo .vstemplate y cierre el editor.
8. Envíe el archivo .vstemplate a una nueva carpeta comprimida MyProjectNode-MyProject.zip.
9. En la ventana de comandos de Visual Studio, escriba:

```
devenv /installvstemplates
```

Abra Visual Studio.

10. Abra el cuadro de diálogo **Nuevo proyecto** y expanda el nodo del proyecto de **Visual C.**



MyProjectNode aparece como un nodo secundario de Visual C., justo debajo del nodo de Windows.

Vea también

- [Nueva generación de proyectos: aspectos técnicos, segunda parte](#)

Nueva generación de proyectos: aspectos técnicos, segunda parte

06/04/2020 • 6 minutes to read • [Edit Online](#)

En [Nueva generación de proyectos: Bajo la capucha, primera parte](#) vimos cómo se rellena el cuadro de diálogo Nuevo **proyecto**. Supongamos que ha seleccionado una aplicación de Windows de **Visual C**, ha rellenado los cuadros de texto **Nombre** y **ubicación** y ha hecho clic en Aceptar.

Generación de los archivos de solución

La elección de Visual Studio una plantilla de aplicación indica a descomprimir y abrir el archivo .vstemplate correspondiente e iniciar una plantilla para interpretar los comandos XML en este archivo. Estos comandos crean proyectos y elementos de proyecto en la solución nueva o existente.

La plantilla desempaqueta los archivos de origen, denominados plantillas de elementos, de la misma carpeta .zip que contiene el archivo .vstemplate. La plantilla copia estos archivos en el nuevo proyecto, personalizándolos en consecuencia.

Reemplazo de parámetros de plantilla

Cuando la plantilla copia una plantilla de elemento en un nuevo proyecto, reemplaza los parámetros de plantilla por cadenas para personalizar el archivo. Un parámetro de plantilla es un token especial que va precedido y seguido de un signo de dólar, por ejemplo, \$date\$.

Echemos un vistazo a una plantilla de elemento de proyecto típica. Extraiga y examine Program.cs en la carpeta Archivos de programa de Microsoft Visual Studio 8, Common7, IDE, ProjectTemplates, CSharp, Windows, Windows, WindowsApplication.zip.

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace $safeprojectname$
{
    static class Program
    {
        // source code deleted here for brevity
    }
}
```

Si crea un nuevo proyecto de aplicación de `$safeprojectname$` Windows denominado Simple, la plantilla reemplaza el parámetro por el nombre del proyecto.


```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace Simple
{
    static class Program
    {
        // source code deleted here for brevity
    }
}
```

Para obtener una lista completa de parámetros de plantilla, consulte [Parámetros de plantilla](#).

Una mirada dentro de un archivo .VSTemplate File

Un archivo .vstemplate básico tiene este formato

```
<VSTemplate Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vstemplate/2005"
  Type="Project">
  <TemplateData>
  </TemplateData>
  <TemplateContent>
  </TemplateContent>
</VSTemplate>
```

Hemos analizado <la sección templateData> en la nueva generación de [proyectos: Under the Hood, Parte Uno](#).

Las etiquetas de esta sección se utilizan para controlar el aspecto del cuadro de diálogo **Nuevo proyecto**.

Las etiquetas <de la sección de> TemplateContent controlan la generación de nuevos proyectos y elementos de proyecto. Aquí está <la sección de> TemplateContent del archivo cswindowsapplication.vstemplate en la carpeta

.

```

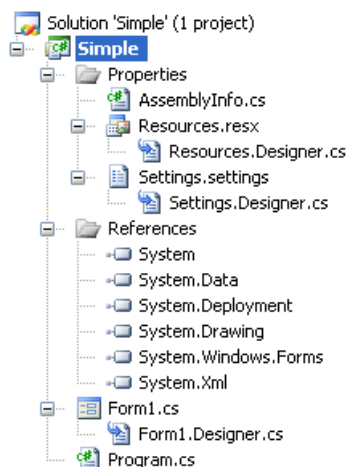
<TemplateContent>
  <Project File="WindowsApplication.csproj" ReplaceParameters="true">
    <ProjectItem ReplaceParameters="true"
      TargetFileName="Properties\AssemblyInfo.cs">
      AssemblyInfo.cs
    </ProjectItem>
    <ProjectItem TargetFileName="Properties\Resources.resx">
      Resources.resx
    </ProjectItem>
    <ProjectItem ReplaceParameters="true" TargetFileName="Properties\Resources.Designer.cs">
      Resources.Designer.cs
    </ProjectItem>
    <ProjectItem TargetFileName="Properties\Settings.settings">
      Settings.settings
    </ProjectItem>
    <ProjectItem ReplaceParameters="true" TargetFileName="Properties\Settings.Designer.cs">
      Settings.Designer.cs
    </ProjectItem>
    <ProjectItem ReplaceParameters="true" OpenInEditor="true">
      Form1.cs
    </ProjectItem>
    <ProjectItem ReplaceParameters="true">
      Form1.Designer.cs
    </ProjectItem>
    <ProjectItem ReplaceParameters="true">
      Program.cs
    </ProjectItem>
  </Project>
</TemplateContent>

```

La <etiqueta Project> controla la <generación de un proyecto y la etiqueta ProjectItem> controla la generación de un elemento de proyecto. Si el parámetro ReplaceParameters es true, la plantilla personalizará todos los parámetros de plantilla en el archivo o elemento del proyecto. En este caso, se personalizan todos los elementos del proyecto, excepto Settings.settings.

El parámetro TargetFileName especifica el nombre y la ruta de acceso relativa del archivo o elemento de proyecto resultante. Esto le permite crear una estructura de carpetas para el proyecto. Si no especifica este argumento, el elemento de proyecto tendrá el mismo nombre que la plantilla de elemento de proyecto.

La estructura de carpetas de aplicaciones de Windows resultante tiene este aspecto:



La primera <y única etiqueta de Project> de la plantilla dice:

```

<Project File="WindowsApplication.csproj" ReplaceParameters="true">

```

Esto indica a la plantilla Nuevo proyecto que cree el archivo de proyecto Simple.csproj copiando y

personalizando el elemento de plantilla windowsapplication.csproj.

Diseñadores y referencias

Puede ver en el Explorador de soluciones que la carpeta Propiedades está presente y contiene los archivos esperados. Pero, ¿qué pasa con las referencias de proyecto y las dependencias de archivos de diseñador, como Resources.Designer.cs a Resources.resx y Form1.Designer.cs a Form1.cs? Se configuran en el archivo Simple.csproj cuando se genera.

Aquí está <el> ItemGroup de Simple.csproj que crea las referencias del proyecto:

```
<ItemGroup>
  <Reference Include="System" />
  <Reference Include="System.Data" />
  <Reference Include="System.Deployment" />
  <Reference Include="System.Drawing" />
  <Reference Include="System.Windows.Forms" />
  <Reference Include="System.Xml" />
</ItemGroup>
```

Puede ver que estas son las seis referencias de proyecto que aparecen en el Explorador de soluciones. Aquí hay una sección <de otro> de ItemGroup. Muchas líneas de código se han eliminado para mayor claridad. Esta sección hace que Settings.Designer.cs dependa de Settings.settings:

```
<ItemGroup>
  <Compile Include="Properties\Settings.Designer.cs">
    <DependentUpon>Settings.settings</DependentUpon>
  </Compile>
</ItemGroup>
```

Vea también

- [Nueva generación de proyectos: aspectos técnicos, primera parte](#)
- [MSBuild](#)

Tipos de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Visual Studio incluye varios tipos de Visual C# proyectos Visual Basic para idiomas como y . Visual Studio también le permite crear sus propios tipos de proyecto.

En esta sección

- [Essentials](#)

Presenta la información importante que debe tener para empezar a trabajar con tipos de proyecto.

- [Creación de tipos de proyecto](#)

Describe el diseño de tipos de proyecto.

- [Adición de un comando a la barra de herramientas del Explorador de soluciones](#)

Detalla los pasos que debe Visual Studio seguir para agregar un botón a la barra de herramientas del Explorador de soluciones.

- [Adición de plantillas de proyecto y de elementos de proyecto](#)

Describe cómo puede agregar plantillas a los tipos de proyecto para que los usuarios puedan crear nuevos proyectos y elementos de proyecto según un patrón.

- [Apertura y guardado de elementos de proyecto](#)

Proporciona información sobre cómo administrar los elementos que admite el tipo de proyecto.

- [Administración de opciones de configuración](#)

Describe cómo los tipos de proyecto pueden admitir opciones de configuración como Depurar y Liberar que controlan cómo se compilan, depuran y demás los proyectos.

- [Compatibilidad con control de código fuente](#)

Proporciona información sobre cómo agregar compatibilidad con sistemas de control de código fuente a los tipos de proyecto.

- [Anidamiento de proyectos](#)

Explica cómo los tipos de proyecto pueden admitir el *anidamiento*, de modo que los proyectos se pueden agrupar en el **Explorador** de soluciones.

- [Actualización de proyectos](#)

Describe cómo el tipo de proyecto puede participar en el Asistente Visual Studio para actualización para actualizar los archivos de proyecto de una versión anterior de .

- [Architecture](#)

Proporciona información técnica detallada sobre los tipos de proyecto.

Secciones relacionadas

- [Jerarquías y selección](#)

Proporciona una visión Visual Studio general de cómo el entorno de desarrollo integrado (IDE) muestra un proyecto como una jerarquía.

- [Subtipos de proyecto](#)

Proporciona vínculos a temas de subtipo de proyecto. Los subtipos de proyecto permiten la extensión de la mayoría de los tipos de proyecto, incluido el suyo propio.

- [Proyectos](#)

Describe cómo ampliar Visual Studio el sistema del proyecto.

Conceptos básicos del tipo de proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Visual Studio incluye varios tipos de Visual C# proyectos Visual Basic para idiomas como o . Visual Studio también le permite crear sus propios tipos de proyecto.

Si solo desea agregar comandos personalizados, editores o ventanas de herramientas a Visual Studio, puede hacerlo sin crear un nuevo tipo de proyecto. Para obtener más información, vea los temas siguientes:

- [Comandos, menús y barras de herramientas](#)
- [Editor y extensiones de servicio de lenguaje](#)
- [Ampliación y personalización de ventanas de herramientas](#)

Del mismo modo, si desea personalizar Visual C# el Visual Basic comportamiento de los tipos proporcionados y de proyecto, puede hacerlo utilizando subtipos de proyecto. Para obtener más información, vea [Subtipos de proyecto](#).

Debe crear un nuevo tipo de proyecto para proyectos Visual C# Visual Basic que se basan en un idioma distinto de y si desea admitir uno o varios de los siguientes:

- Compilar
- Implementación
- Múltiples configuraciones
- Control de código fuente
- Depuración
- Elementos de proyecto en el Explorador de soluciones
- Los cuadros de diálogo **Abrir proyecto** o **Nuevo proyecto**
- Anidación de proyectos
- Para obtener más información acerca de las capacidades de los tipos de proyecto, consulte lo siguiente:
- Tipos de proyecto son objetos en un Visual Studio VSPackage que implementan el conjunto de interfaces espera. Si usa C- para desarrollar un tipo de proyecto, las clases de proyecto de Managed Package Framework implementan las interfaces necesarias para usted y le permiten heredar esa implementación. Para obtener más información, vea [Uso del marco de trabajo de paquete administrado para implementar un tipo de proyecto \(C-\)](#).
- Para los desarrolladores de C++, las clases de la biblioteca HierUtil funcionan de forma similar. Para obtener más información, vea [No en compilación: uso de clases de proyecto HierUtil7 para implementar un tipo de proyecto \(C++\)](#).
- Los tipos de proyecto pueden admitir datos distintos de los archivos de código fuente típicos que se compilan en un ensamblado .exe o .dll. Por ejemplo, Visual Studio los proyectos de base de datos contienen referencias a archivos de script y consulta almacenados en el disco y agregan comandos al **Explorador** de soluciones para ejecutar los scripts y consultas en una base de datos, pero los proyectos no admiten el comportamiento de compilación. Para obtener más información, consulte [Apertura y guardado de elementos de proyecto](#).

- Un tipo de proyecto no tiene que utilizar archivos en absoluto. Por ejemplo, un tipo de proyecto podría almacenar todos sus datos en una base de datos. Visual Studio proporciona a los tipos de proyecto un control completo sobre cómo conservan los datos de los proyectos y los elementos de proyecto. Para obtener más información, vea [Decisiones](#) de diseño de tipo de proyecto .
- Los tipos de proyecto deben proporcionar un generador de *proyectos*, Visual Studio que es un objeto que crea una instancia del tipo de proyecto siempre que se le indique que abra o cree un proyecto basado en ese tipo de proyecto. Para obtener más información, vea Creación de instancias de [proyecto mediante factorías](#) de proyecto .
- Los tipos de proyecto deben proporcionar plantillas para proyectos y elementos de proyecto. Visual Studio utiliza las plantillas cuando los usuarios crean nuevos proyectos y agregan nuevos elementos a los proyectos existentes. Para obtener más información, vea Agregar plantillas de [proyecto y elemento](#) de proyecto .
- Los tipos de proyecto pueden admitir varias configuraciones, como Depurar y Liberar. Los usuarios pueden cambiar las diferentes configuraciones de un proyecto mediante las páginas de propiedades que proporcione. Para obtener más información, consulte Administración de opciones de [configuración](#).

Vea también

- [Implementación de tipos de proyecto](#)

Crear tipos de proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Puede ampliar Visual Studio creando un nuevo tipo de proyecto. Para crear un nuevo tipo de proyecto, debe comprender varios conceptos y completar una serie de pasos. Los temas siguientes proporcionan información general sobre cómo crear tipos de proyecto.

En esta sección

- [Decisiones de diseño de tipo de proyecto](#)

Describe las decisiones de diseño de mecánica de elementos, persistencia de archivos de proyecto y compromiso que debe tomar antes de crear un nuevo tipo de proyecto.

- [Lista de verificación: Crear nuevos tipos de proyecto](#)

Proporciona información general sobre los pasos que debe seguir para crear un nuevo tipo de proyecto que admita tareas de programación como editar código y compilar, depurar e implementar aplicaciones en el proyecto.

- [Crear instancias de proyecto mediante fábricas de proyectos](#)

Proporciona información sobre cómo proporcionar y usar un generador de proyectos para crear instancias de un nuevo proyecto.

- [Registrar un tipo de proyecto](#)

Proporciona ejemplos de código de instrucciones del registro que proporcionan rutas de acceso y datos predeterminados, y una tabla que contiene entradas del script del Registro para cada instrucción.

- [Persistencia del proyecto](#)

Describe el uso `IPersistFileFormat` de conservar objetos de proyecto no basados en archivos y no basados en archivos.

- [Usar MSBuild](#)

Describe cómo el tipo de MSBuild proyecto puede usar Visual Studio el motor de compilación para permitir que los usuarios se compilen desde y en la línea de comandos.

Secciones relacionadas

- [Soporta herramientas de navegación de símbolos](#)

Explica la arquitectura de herramientas de visualización de código como el **Examinador de objetos** y la ventana Vista de **clases**. Describe las interfaces y métodos que se usan para implementar la exploración de objetos en un VSPackage.

- [Agregar plantillas de proyecto y elemento de proyecto](#)

Describe la importancia que juegan los proyectos para determinar qué editor se utiliza cuando se abre un elemento de proyecto y cómo se pueden manipular los recursos del proyecto.

- [Instalar VSPackages con Windows Installer](#)

Muestra cómo dar al VSPackage su propia identidad única y cómo ajustar los archivos DLL de VSPackage y otra información en un paquete de Windows Installer (. MSI) para su implementación a sus clientes.

- [Jerarquías en Visual Studio](#)

Describe cómo Visual Studio se describen las vistas y las jerarquías.

- [VSPackages](#)

Proporciona información general de un VSPackage, un objeto Visual Studio COM instalable que extiende el entorno y describe cómo implementar su propio VSPackage.

- [Tipos de proyecto](#)

Describe cómo usar proyectos para modificar código, compilar y compilar código y ejecutar y depurar código, y proporciona vínculos a temas detallados sobre cómo crear tipos de proyecto.

Decisiones de diseño del tipo de proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Antes de crear un nuevo tipo de proyecto, debe tomar varias decisiones de diseño con respecto al tipo de proyecto. Debe decidir qué tipos de elementos contendrán sus proyectos, cómo se conservarán los archivos de proyecto y qué modelo de compromiso usará.

Elementos de un proyecto

¿Utilizará el proyecto archivos u objetos abstractos? Si utiliza archivos, ¿serán archivos basados en referencias o en directorios? ¿Los archivos u objetos abstractos van a ser locales o remotos?

Los elementos de un proyecto pueden ser archivos o pueden ser objetos más abstractos, como objetos en un repositorio de base de datos o conexiones de datos a través de Internet. Si los elementos son archivos, el proyecto puede ser un proyecto basado en referencias o un proyecto basado en directorios.

En los proyectos basados en referencias, los elementos pueden aparecer en más de un proyecto. Sin embargo, el archivo real que representa un elemento se encuentra en un solo directorio. En los proyectos basados en directorios, todos los elementos de proyecto existen en la estructura de directorios.

Los elementos locales se almacenan en el mismo equipo donde está instalada la aplicación. Los elementos remotos se pueden almacenar en un servidor independiente en una red local o en otro lugar de Internet.

Persistencia del archivo de proyecto

¿Los datos se almacenarán en sistemas de archivos planos comunes o en almacenamiento estructurado? ¿Se abrirán los archivos mediante un editor estándar o un editor específico del proyecto?

Para conservar sus datos, la mayoría de las aplicaciones guardan sus datos en un archivo y, a continuación, los vuelven a leer cuando un usuario debe revisar o cambiar la información.

El almacenamiento estructurado, también denominado archivos compuestos, se utiliza normalmente cuando varios objetos del modelo de objetos componentes (COM) necesitan almacenar sus datos persistentes en un único archivo. Con el almacenamiento estructurado, varios componentes de software diferentes pueden compartir un único archivo de disco.

Tiene varias opciones a tener en cuenta con respecto a la persistencia de los elementos del proyecto. Puede realizar cualquiera de las siguientes opciones:

- Guarde cada archivo individualmente cuando se haya cambiado.
- Capture muchas transacciones en una sola operación **De guardar**.
- Guarde archivos localmente y, a continuación, publique en un servidor o use otro enfoque para guardar elementos de proyecto cuando el elemento representa una conexión de datos a un objeto remoto.

Para obtener más información acerca de la persistencia, vea [Persistencia](#) del proyecto y [Abrir y guardar elementos](#) de proyecto .

Modelo de compromiso del proyecto

¿Se abrirán los objetos de datos persistentes en modo directo o en modo de transacción?

Cuando los objetos de datos se abren en modo directo, los cambios realizados en los datos se incorporan

inmediatamente o cuando el usuario guarda manualmente el archivo.

Cuando los objetos de datos se abren mediante el modo de transacción, los cambios se guardan en una ubicación temporal en la memoria y no se confirman hasta que el usuario elige manualmente guardar el archivo. En ese momento, todos los cambios deben ocurrir juntos o no se realizarán cambios.

Vea también

- [Lista de comprobación: creación de nuevos tipos de proyecto](#)
- [Apertura y guardado de elementos de proyecto](#)
- [Persistencia de un proyecto](#)
- [Elementos de un modelo de proyecto](#)
- [Componentes principales del modelo de proyecto](#)
- [Creación de tipos de proyecto](#)

Lista de verificación: Crear nuevos tipos de proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Debe completar varias tareas para crear un nuevo tipo de proyecto. La siguiente lista de comprobación proporciona una guía para esas tareas:

1. Diseñe la funcionalidad para el nuevo tipo de proyecto. Para obtener más información, consulte [Decisiones](#) de diseño de tipo de proyecto .
2. Determine qué editores se utilizan para el código y otros elementos del proyecto. Puede utilizar los editores principales o estándar, o puede crear y utilizar editores específicos del proyecto. Para obtener más información, consulte [Crear editores y diseñadores personalizados](#) y [Cómo: Abrir editores específicos del proyecto](#).
3. Determine el nivel de participación que tendrán los elementos del proyecto en la **vista de clases** y el **Examinador de objetos**. Para obtener más información, consulte Compatibilidad con [herramientas de exploración de símbolos](#).
4. Derive nuevas clases basadas en las decisiones de diseño que tomó anteriormente para los elementos de proyecto y proyecto.
5. Escriba el código para los siguientes componentes de tipo de proyecto:
 - Fábrica de proyectos, para gestionar la creación de nuevos proyectos y la apertura de proyectos existentes. Para obtener más información, consulte Crear instancias de [proyecto mediante generadores](#) de proyectos.
 - Jerarquía de proyectos y control de comandos. Para obtener más información, vea Usar clases de [proyecto HierUtil7 para implementar un tipo de proyecto \(C++\)](#), [Elementos de un modelo](#) de proyecto , [Componentes principales](#) del modelo de proyecto y MenuCommands frente a [OleMenuCommands](#).
 - Administración de elementos de proyecto, incluida la adición del proyecto al cuadro de diálogo **Nuevo proyecto**. Para obtener más información, vea Agregar plantillas de [elementos](#) de proyecto y proyecto y Registrar plantillas de [proyecto y elemento](#).
 - Persistencia del estado del proyecto y de los elementos individuales. Para obtener más información, consulte [Abrir y guardar elementos](#) de proyecto . Para obtener información sobre la persistencia de la solución, consulte [Soluciones](#).
 - Propiedades independientes de la configuración que se mostrarán en la ventana Propiedades. Para obtener más información, consulte [Extender propiedades](#).
 - Propiedades de configuración del proyecto tal como se implementan en las páginas de propiedades para mostrar propiedades dependientes de la configuración. Para obtener más información, consulte [Administrar opciones](#) de configuración .
 - Enumeración de salidas para la implementación. Para obtener más información, consulte [Configuración del proyecto para la salida](#).
 - Servicios de inicio de proyectos. Para obtener más información, vea [Elementos de un modelo de proyecto](#) y [Componentes principales](#) del modelo de proyecto .
 - Objetos o clases `IDispatch` derivadas de , disponibles para la automatización.

- Archivos de tabla de comandos XML (.vsct). Para obtener más información, vea [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#).
6. Pruebe, depure e inicie el tipo de proyecto.
 7. Muestre el proyecto en la ficha **Proyecto** del `VARIANT_TRUE` cuadro de `VSHPROPID_ShowProjInSolutionPage` diálogo **Agregar referencia** estableciendo como valor para `.` Para más información, vea [__VSHPROPID](#) y [GetProperty](#).
 8. Cree el archivo microsoft Installer (.msi) para instalar los VSPackages. Para obtener más información, vea [Instalar VSPackages con Windows Installer](#), Registrar un tipo de [proyecto](#) [VSPackages](#).

Vea también

- [Jerarquías en Visual Studio](#)
- [Cuándo crear tipos de proyecto](#)
- [Crear tipos de proyecto](#)

Crear instancias de proyecto mediante fábricas de proyectos

06/04/2020 • 6 minutes to read • [Edit Online](#)

Tipos de Visual Studio proyecto en uso un generador de *proyectos* para crear instancias de objetos de proyecto. Un generador de proyectos es similar a un generador de clases estándar para objetos COM cocreables. Sin embargo, los objetos de proyecto no son cocreables; sólo se pueden crear mediante una fábrica de proyectos.

El Visual Studio IDE llama al generador de proyectos implementado en el VSPackage cuando Visual Studio un usuario carga un proyecto existente o crea un nuevo proyecto en . El nuevo objeto de proyecto proporciona al IDE suficiente información para rellenar el Explorador de **soluciones**. El nuevo objeto de proyecto también proporciona las interfaces necesarias para admitir todas las acciones de interfaz de usuario relevantes iniciadas por el IDE.

Puede implementar [IVsProjectFactory](#) la interfaz en una clase del proyecto. Normalmente, reside en su propio módulo.

Los proyectos que admiten la agregación de un propietario deben conservar una clave de propietario en el archivo de proyecto. Cuando [CreateProject](#) se llama al método en un proyecto con una clave de propietario, el `CreateProject` proyecto de propiedad convierte su clave de propietario en un GUID de generador de proyectos y, a continuación, llama al método en este generador de proyectos para realizar la creación real.

Crear un proyecto propio

Un propietario crea un proyecto propio en dos fases:

1. Llamando al [PreCreateForOwner](#) método. Esto da al proyecto propietario la oportunidad de crear un `IUnknown` objeto de proyecto agregado basado en el control de entrada. El proyecto de propiedad `IUnknown` pasa el objeto interno y agregado al proyecto propietario. Esto le da al proyecto propio `IUnknown` la oportunidad de almacenar el archivo .
2. Llamando al [InitializeForOwner](#) método. El proyecto de propiedad realiza toda su creación de `IVsProjectFactory::CreateProject` instancias cuando se llama a este método en lugar de llamar como sería el caso de los proyectos que no son propiedad. La `VSPROJECTOBJECT` enumeración de entrada suele ser el proyecto de propiedad agregado. El proyecto de propiedad puede usar esta variable para determinar si su objeto de proyecto ya se ha creado (cookie no es igual a NULL) o debe crearse (cookie es igual a NULL).

Los tipos de proyecto se identifican mediante un GUID de proyecto único, similar al CLSID de un objeto COM cocreable. Normalmente, un generador de proyectos controla la creación de instancias de un único tipo de proyecto, aunque es posible tener un generador de proyectos controlar más de un GUID de tipo de proyecto.

Los tipos de proyecto están asociados a una extensión de nombre de archivo determinada. Cuando un usuario intenta abrir un archivo de proyecto existente o intenta crear un nuevo proyecto clonando una plantilla, el IDE usa la extensión en el archivo para determinar el GUID del proyecto correspondiente.

Tan pronto como el IDE determina si debe crear un nuevo proyecto o abrir un proyecto existente de un tipo determinado, el IDE usa la información en el registro del sistema en `[HKEY_LOCAL_MACHINE, Software, Microsoft, VisualStudio, 8.0, Projects]` para buscar qué VSPackage implementa el generador de proyectos necesario. El IDE carga este VSPackage. En [SetSite](#) el método, el VSPackage debe

registrar su generador [RegisterProjectType](#) de proyectos con el IDE mediante una llamada al método.

El método principal `IVsProjectFactory` de [CreateProject](#) la interfaz es , que debe controlar dos escenarios: abrir un proyecto existente y crear un nuevo proyecto. La mayoría de los proyectos almacenan su estado de proyecto en un archivo de proyecto. Normalmente, los nuevos proyectos se crean haciendo una `CreateProject` copia del archivo de plantilla pasado al método y, a continuación, abriendo la copia. Se crean instancias de los proyectos `CreateProject` existentes abriendo directamente el archivo de proyecto pasado al método. El `CreateProject` método puede mostrar características de interfaz de usuario adicionales al usuario según sea necesario.

Un proyecto también puede usar ningún archivo y, en su lugar, almacenar su estado de proyecto en un mecanismo de almacenamiento distinto del sistema de archivos, como una base de datos o un servidor web. En este caso, el parámetro `CreateProject` de nombre de archivo pasado al método no es realmente una ruta de acceso del sistema de archivos, sino una cadena única (una dirección URL) para identificar los datos del proyecto. No es necesario copiar los archivos de `CreateProject` plantilla que se pasan para desencadenar la secuencia de construcción adecuada que se va a ejecutar.

Vea también

- [IVsOwnedProjectFactory](#)
- [IVsProjectFactory](#)
- [IVsRegisterProjectTypes](#)
- [Lista de verificación: Crear nuevos tipos de proyecto](#)

Uso de la plataforma de paquete administrado para implementar un tipo de proyecto (C#)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Managed Package Framework (MPF) proporciona clases de C# que puede usar o heredar para implementar sus propios tipos de proyecto. El MPF implementa muchas de las interfaces que Visual Studio espera que proporcione un tipo de proyecto, lo que le permite concentrarse en implementar los detalles del tipo de proyecto.

Uso del código fuente del proyecto MPF

Managed Package Framework for Projects (MPFProj) proporciona clases auxiliares para crear y administrar un nuevo sistema de proyectos. A diferencia de otras clases en el MPF, las clases de proyecto no se incluyen en los ensamblados incluidos con Visual Studio. En su lugar, las clases de proyecto se proporcionan como código fuente en [MPF para Proyectos 2013](#).

Para agregar este proyecto a la solución de VSPackage, haga lo siguiente:

1. Descargue los archivos MPFProj en *MPFProjectDir*.
2. En el archivo *MPFProjectDir*, cambie el siguiente bloque:

```
<!-- Provide a default value for $(ProjectBasePath) -->
<PropertyGroup>
  <ProjectBasePath>MPFProjDir\Dev10\Src\CSharp\</ProjectBasePath>
</PropertyGroup>
```

1. Crear un proyecto de VSPackage.
2. Descargue el proyecto de VSPackage.
3. Edite el archivo .csproj de VSPackage agregando el siguiente bloque antes que los otros `<Import>` bloques:

```
<Import Project="MPFProjectDir\Dev10\Src\CSharp\ProjectBase.files" />
<PropertyGroup>
  <!--To specify a different registry root to register your package, uncomment the TargetRegistryRoot tag
and specify a registry root in it.
  <TargetRegistryRoot></TargetRegistryRoot-->
  <RegisterOutputPackage>true</RegisterOutputPackage>
  <RegisterWithCodebase>true</RegisterWithCodebase>
</PropertyGroup>
```

1. Guarde el proyecto.
2. Cierre y vuelva a abrir la solución VSPackage.
3. Vuelva a abrir el proyecto VSPackage. Debería ver un nuevo directorio denominado ProjectBase.
4. Agregue la siguiente referencia al proyecto VSPackage:

Microsoft.Build.Tasks.4.0

5. Compile el proyecto.

Clases de jerarquía

En la tabla siguiente se resumen las clases de MPFProj que admiten jerarquías de proyectos. Para obtener más información, consulte [Jerarquías y selección](#).

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.HierarchyNode
Microsoft.VisualStudio.Package.ProjectNode
Microsoft.VisualStudio.Package.ProjectContainerNode
Microsoft.VisualStudio.Package.FileNode
Microsoft.VisualStudio.Package.FolderNode
Microsoft.VisualStudio.Package.ReferenceContainerNode
Microsoft.VisualStudio.Package.ReferenceNode
Microsoft.VisualStudio.Package.ProjectReferenceNode
Microsoft.VisualStudio.Package.ComReferenceNode
Microsoft.VisualStudio.Package.AssemblyReferenceNode
Microsoft.VisualStudio.Package.BuildDependency

Clases de gestión de documentos

En la tabla siguiente se enumeran las clases del MPF que admiten el control de documentos. Para obtener más información, consulte [Apertura y guardado de elementos de proyecto](#).

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.DocumentManager
Microsoft.VisualStudio.Package.FileDocumentManager

Clases de configuración y salida

En la tabla siguiente se enumeran las clases del MPF que permiten que los tipos de proyecto admitan varias configuraciones, como depuración y versión, y colecciones de salida del proyecto. Para obtener más información, consulte [Administración de opciones de configuración](#).

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.ConfigProvider
Microsoft.VisualStudio.Package.ProjectConfig

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.BuildableProjectConfig
Microsoft.VisualStudio.Package.OutputGroup
Microsoft.VisualStudio.Package.ProjectElement

Clases de automatización-soporte

En la tabla siguiente se enumeran las clases del MPF que admiten la automatización para que los usuarios del tipo de proyecto puedan escribir complementos.

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.Automation.OAProject
Microsoft.VisualStudio.Package.Automation.OANavigableProjectItems
Microsoft.VisualStudio.Package.Automation.OAProjectItems
Microsoft.VisualStudio.Package.Automation.OAProjectItem
Microsoft.VisualStudio.Package.Automation.OANestedProjectItem

Clases de propiedades

En la tabla siguiente se enumeran las clases del MPF que permiten a los tipos de proyecto agregar propiedades que los usuarios pueden examinar y modificar en un explorador de propiedades.

NOMBRE DE LA CLASE
Microsoft.VisualStudio.Package.LocalizableProperties
Microsoft.VisualStudio.Package.NodeProperties
Microsoft.VisualStudio.Package.FileNodeProperties
Microsoft.VisualStudio.Package.ProjectNodeProperties
Microsoft.VisualStudio.Package.FolderNodeProperties
Microsoft.VisualStudio.Package.ReferenceNodeProperties

Registro de un tipo de proyecto

06/04/2020 • 16 minutes to read • [Edit Online](#)

Al crear un nuevo tipo de proyecto, debe Visual Studio crear entradas del Registro que permitan reconocer y trabajar con el tipo de proyecto. Normalmente, estas entradas del Registro se crean mediante un archivo de script de registro (.rgs).

En el ejemplo siguiente, las instrucciones del registro proporcionan rutas de acceso y datos predeterminados cuando corresponda, seguidas de una tabla que contiene entradas del script del Registro para cada instrucción. Las tablas proporcionan las entradas de script e información adicional sobre las instrucciones.

NOTE

La siguiente información del Registro está pensada para ser un ejemplo del tipo y los propósitos de las entradas en los scripts del Registro que va a escribir para registrar el tipo de proyecto. Las entradas reales y sus usos pueden variar en función de los requisitos específicos del tipo de proyecto. Debe revisar los ejemplos disponibles para encontrar uno que se asemeje mucho al tipo de proyecto que está desarrollando y, a continuación, revise el script del Registro para ese ejemplo.

Los siguientes ejemplos son de HKEY_CLASSES_ROOT.

Ejemplo

```
\.figp
  @="FigPrjFile"
  "Content Type"="text/plain"
\.figp\ShellNew
  "NullFile"=""
\FigPrjFile
  @="Figure Project File"
\DefaultIcon
  @="<Visual Studio SDK installation
path>\9.0VSIIntegration\SomeFolder\FigPkgs\FigPrj\Debug\FigPrj.dll,-206"
\shell\open
  @="&Open in Visual Studio"
\shell\open\command
  @="devenv.exe \"%1\""
```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	FigPrjFile	Nombre y descripción de los archivos de tipo de proyecto que tienen la extensión .figp.
Content Type	REG_SZ	Text/plain	Tipo de contenido para los archivos de proyecto.
NullFile	REG_SZ	Null	

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	%MODULE%, - 206	Icono predeterminado utilizado para el proyecto de este tipo. La instrucción %MODULE% se completa en el registro en la ubicación predeterminada del archivo DLL de tipo de proyecto.
@	REG_SZ	&Open in Visual Studio	Aplicación predeterminada en la que se abrirá este tipo de proyecto.
@	REG_SZ	devenv.exe "%1"	Comando predeterminado que se ejecutará cuando se abra un proyecto de este tipo.

Los siguientes ejemplos son de HKEY_LOCAL_MACHINE y se encuentran en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 99.0Exp, Packages].

Ejemplo

```

\{ACEF4EB2-57CF-11D2-96F4-000000000000} (The CLSID for the VSPackage)
  @="FigPrj Project Package"
  "InprocServer32"="9.0<Visual Studio SDK installation path>\VSIntegration\Archive\FigPkgs\FigPrj\Debug\FigPrj.dll"
  "CompanyName"="Microsoft"
  "ProductName"="Figure Project Sample"
  "ProductVersion"="9.0"
  "MinEdition"="professional"
  "ID"=dword:00000001
\{ACEF4EB2-57CF-11D2-96F4-000000000000}\SatelliteDLL
  "DllName"="FigPrjUI.dll"
  "Path"="9.0<Visual Studio SDK installation path>\VSIntegration\Archive\FigPkgs\FigPrj\Debug\"
\{ACEF4EB2-57CF-11D2-96F4-000000000000}\Automation
  "FigProjects"=""
\{ACEF4EB2-57CF-11D2-96F4-000000000000}\AutomationEvents
  "FigProjectsEvents"="Returns the FigProjectsEvents Object"
  "FigProjectItemsEvents"="Returns the FigProjectItemsEvents Object"

```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@ (Valor predeterminado)	REG_SZ	FigPrj Project VSPackage	Nombre localizable de este VSPackage registrado (tipo de proyecto).
InprocServer32	REG_SZ	%MODULE%	Ruta de acceso del tipo de proyecto DLL. El IDE carga este archivo DLL y pasa <code>DllGetClassObject</code> el IClassFactory VSPackage IVsPackage CLSID para llegar a construir el objeto.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
CompanyName	REG_SZ	Microsoft	Nombre de la empresa que desarrolló el tipo de proyecto.
ProductName	REG_SZ	Figure Project Sample	Nombre del tipo de proyecto.
ProductVersion	REG_SZ	9.0	Número de versión de la versión de tipo de proyecto.
MinEdition	REG_SZ	professional	Edición del VSPackage que se está registrando.
ID	REG_DWORD	%IDS_PACKAGE_LOAD_KEY%	La clave de carga del paquete para el proyecto VSPackage. La clave se valida cuando se carga un proyecto después de que se ha iniciado el entorno.
DllName	REG_SZ	%RESOURCE_DLL%	Nombre de archivo del archivo DLL satélite que contiene recursos localizados para el tipo de proyecto.
Path	REG_SZ	%RESOURCE_PATH%	Ruta de acceso del archivo DLL de satélite.
FigProjectsEvents	REG_SZ	Consulte la instrucción para obtener valor.	Determina la cadena de texto devuelta para este evento de automatización.
FigProjectItemsEvents	REG_SZ	Consulte la instrucción para obtener valor.	Determina la cadena de texto devuelta para este evento de automatización.

Todos los ejemplos siguientes se encuentran en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9.0Exp, Projects].

Ejemplo

```

\{C061DB26-5833-11D2-96F5-000000000000} (The CLSID for projects of this type)
  @="FigPrj Project"
  "DisplayName"="#2"
  "Package"="{ACEF4EB2-57CF-11D2-96F4-000000000000}"
  "ProjectTemplatesDir"="C:\\Program Files\\VSIP 9.0\\EnvSDK\\FigPkgs\\
    FigPrj\\FigPrjProjects"
  "ItemTemplatesDir"="<Visual Studio SDK installation path>\\VSIntegration\\Archive9.0\\FigPkgs\\FigPrj\\
    FigPrjProjectItems"
  "DisplayProjectFileExtensions"="#3"
  "PossibleProjectExtensions"="figp"
  "DefaultProjectExtension"=".figp"
\{C061DB26-5833-11D2-96F5-000000000000}\\Filters\1      (Folder 1 contains settings for Open Files filters.)
  @="#4"
  "CommonOpenFilesFilter"=dword:00000000
  "CommonFindFilesFilter"=dword:00000000
  "NotAddExistingItemFilter"=dword:00000000
  "FindInFilesFilter"=dword:00000000
  "NotOpenFileFilter"=dword:00000000
  "SortPriority"=dword:000003e8
\{C061DB26-5833-11D2-96F5-000000000000}\\Filters\2
  (Folder 2 contains settings for Find in Files filters.)
  @="#5"
  "CommonOpenFilesFilter"=dword:00000000
  "CommonFindFilesFilter"=dword:00000000
  "NotAddExistingItemFilter"=dword:00000001
  "FindInFilesFilter"=dword:00000001
  "NotOpenFileFilter"=dword:00000000
  "SortPriority"=dword:000003e8
\{C061DB26-5833-11D2-96F5-000000000000}\\AddItemTemplates\\TemplateDirs\\ {ACEF4EB2-57CF-11D2-96F4-
000000000000}\\1 (Second GUID indicates the registered project type for the Add Items templates.)
  @="#6"
  "TemplatesDir"="<Visual Studio SDK installation path>\\VSIntegration\\Archive9.0\\FigPkgs\\FigPrj\\
    FigPrjProjectItems"
  "SortPriority"=dword:00000064

```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	FigPrj Project	Nombre predeterminado de proyectos de este tipo.
DisplayName	REG_SZ	##IDS_PROJECT_TYPE%	Identificador de recurso del nombre que se va a recuperar del archivo DLL satélite registrado en Paquetes.
Package	REG_SZ	%CLSID_Package%	Identificador de clase del VSPackage registrado en paquetes.
ProjectTemplatesDir	REG_SZ	%TEMPLATE_PATH%FigPrjProjectItems	Ruta predeterminada de los archivos de plantilla de proyecto. Estos son los archivos mostrados por la plantilla Nuevo proyecto.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
ItemTemplatesDir	REG_SZ	%TEMPLATE_PATH%\FigPrjProjectItems	Ruta de acceso predeterminada de los archivos de plantilla de elemento de proyecto. Estos son los archivos mostrados por la plantilla Agregar nuevo elemento.
DisplayProjectFileExtensions	REG_SZ	##IDS_DISPLAY_PROJ_FILE_EXT%	Habilita el IDE para implementar el cuadro de diálogo Abrir .
PossibleProjectExtensions	REG_SZ	figp	Utilizado por el IDE para determinar si el proyecto que se está abriendo se controla mediante este tipo de proyecto (fábrica de proyectos). El formato de más de una entrada es una lista delimitada por punto y coma. Por ejemplo"vdproj;vdp".
DefaultProjectExtension	REG_SZ	.figp	Utilizado por el IDE como la extensión de nombre de archivo predeterminada para la operación Guardar como.
Filter Settings	REG_DWORD	Varios, vea las instrucciones y comentarios que se indican a continuación.	Esta configuración se utiliza para establecer los distintos filtros para mostrar archivos en los cuadros de diálogo de la interfaz de usuario.
@	REG_SZ	##IDS_ADDITEM_TEMPLATES_ENTRY%	Identificador de recurso para agregar plantillas de elemento.
TemplatesDir	REG_SZ	%TEMPLATE_PATH%\FigPrjProjectItems	Ruta de acceso de los elementos del proyecto que se muestran en el cuadro de diálogo de la plantilla Agregar nuevo elemento .
SortPriority	REG_DWORD	100 (vcprx64)	Determina el criterio de ordenación en el nodo de árbol de los archivos que se muestran en el cuadro de diálogo Agregar nuevo elemento .

En la tabla siguiente se muestran las opciones Filtros disponibles en el segmento de código anterior.

OPCIÓN DE FILTRO	DESCRIPCIÓN

OPCIÓN DE FILTRO	DESCRIPCIÓN
<code>CommonFindFilesFilter</code>	Indica que el filtro es uno de los filtros comunes del cuadro de diálogo Buscar en archivos . Los filtros comunes se enumeran en la lista de filtros antes de que los filtros no estén marcados como comunes.
<code>CommonOpenFilesFilter</code>	Indica que el filtro es uno de los filtros comunes en el cuadro de diálogo Abrir archivo . Los filtros comunes se enumeran en la lista de filtros antes de que los filtros no estén marcados como comunes.
<code>FindInFilesFilter</code>	Indica que el filtro será uno de los filtros del cuadro de diálogo Buscar en archivos y aparecerá después de los filtros comunes.
<code>NotOpenFileFilter</code>	Indica que el filtro no se utilizará en el cuadro de diálogo Abrir archivo .
<code>NotAddExistingItemFilter</code>	Indica que el filtro no se usará en el cuadro de diálogo Agregar elemento existente .

De forma predeterminada, si un filtro no tiene uno o varios de estos indicadores establecidos, el filtro se utiliza en el cuadro de diálogo **Agregar elemento existente** y en el cuadro de diálogo **Abrir archivo** después de que aparezcan los filtros comunes. El filtro no se utiliza en el cuadro de diálogo **Buscar en archivos**.

Todos los ejemplos siguientes se encuentran en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9.0Exp, Projects].

Ejemplo

```
{FE3B8BB6-72D5-11d2-9ACE-00C04F79A2A4} (The CLSID for Enterprise Projects)
\{FE3B8BB6-72D5-11d2-9ACE-00C04F79A2A4}\AddItemTemplates\TemplateDirs\ {ACEF4EB2-57CF-11D2-96F4-000000000000}\1 (CLSID for projects of this type)
    @="#7"
    "TemplatesDir"="<Visual Studio SDK installation path>\VSIntegration\Archive9.0\FigPrj\FigPrjProjects"
    "SortPriority"=dword:00000029
    "NewProjectDialogOnly"=dword:00000000
```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	##IDS_NEWPROJ_TEMPLATES_ENTRY%	Id. de recurso para las plantillas de nuevo proyecto.
TemplatesDir	REG_SZ	%TEMPLATE_PATH%\FigPrjProjects	Ruta de acceso predeterminada para proyectos del tipo de proyecto registrado.
SortPriority	REG_DWORD	41 (x29)	Establece el criterio de ordenación de los proyectos que se muestran en el cuadro de diálogo del asistente Nuevos proyectos.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
NewProjectDialogOnly	REG_DWORD	0	0 indica que los proyectos de este tipo solo se muestran en el cuadro de diálogo Nuevo proyecto.

Todos los ejemplos siguientes se encuentran en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9.0Exp, Projects].

Ejemplo

```
\{A2FE74E1-B743-11d0-AE1A-00A0C90FFFC3} (CLSID for Miscellaneous Files projects)
  @="Miscellaneous Files Project"
\AddItemTemplates\TemplateDirs\{ACEF4EB2-57CF-11D2-96F4-000000000000}\1
                                     (CLSID for Figures Project projects)
  @="#6"
  "TemplatesDir"="<Visual Studio SDK installation path>\VSIntegration\Archive9.0\FigPkgs\FigPrj\FigPrjProjectItems"
  "SortPriority"=dword:00000064
```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	None	Valor predeterminado que indica que las entradas siguientes son para las entradas de proyectos de archivos varios.
@	REG_SZ	#{IDS_ADDITEM_TEMPLATES_ENTRY}	Valor de identificador de recurso para los archivos de plantilla Agregar nuevos elementos.
TemplatesDir	REG_SZ	%TEMPLATE_PATH%\FigPrjProjectItems	Ruta predeterminada de los elementos que se mostrarán en el cuadro de diálogo Agregar nuevo elemento.
SortPriority	REG_DWORD	100 (vcprx64)	Establece el criterio de ordenación para mostrar en el nodo de árbol del cuadro de diálogo Agregar nuevo elemento.

En el ejemplo siguiente se encuentra en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9.0Exp, Menus].

Ejemplo

```
"{ACEF4EB2-57CF-11D2-96F4-000000000000}"="",1000,1"
```

La entrada de menú señala el IDE al recurso utilizado para recuperar la información del menú. Cuando estos datos se han combinado en la base de datos de menús, se agregará la misma clave en la sección MenusMerged del registro. El VSPackage no debe modificar nada en el MenusMerged sección directamente. En el campo Datos de la

tabla siguiente, hay tres campos separados por comas. El primer campo identifica una ruta de acceso completa de un archivo de recursos de menú:

- Si se omite el primer campo, el recurso de menú se carga desde el archivo DLL satélite identificado por el GUID de VSPackage.

El segundo campo identifica un identificador de recurso de menú del tipo CTMENU:

- Si se especifica el identificador de recurso y el primer parámetro proporciona la ruta de acceso del archivo, se carga un recurso de menú desde la ruta de acceso completa del archivo.
- Si se proporciona el identificador de recurso, pero la ruta de acceso del archivo no, el recurso de menú se carga desde el archivo DLL satélite.
- Si se proporciona la ruta de acceso completa del archivo y se omite el identificador de recurso, se espera que el archivo que se va a cargar sea un archivo CTO.

El último campo identifica el número de versión del recurso CTMENU. Puede volver a combinar el menú cambiando el número de versión.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
%CLSID_Package%	REG_SZ	,1000,1	El recurso para recuperar la información del menú.

Todos los ejemplos siguientes se encuentran en el registro bajo la clave [HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9.0Exp, NewProjectTemplates].

\TemplateDirs\{ACEF4EB2-57CF-11D2-96F4-000000000000}\1		(CLSID for Figures Project projects)
@="#7"		
"TemplatesDir"="<Visual Studio SDK installation path>\VSIntegration\Archive9.0\FigPkgs\FigPrj\FigPrjProjects"		
"SortPriority"=dword:00000029		
"NewProjectDialogOnly"=dword:00000000		

NOMBRE	TIPO	DATA	DESCRIPCIÓN
@	REG_SZ	##IDS_NEWPROJ_TEMPLATES_ENTR	Valor de identificador de recurso para las plantillas de proyecto nuevo de proyectos de figuras.
TemplatesDir	REG_SZ	%TEMPLATE_PATH%\FigPrjProjects	Ruta de acceso predeterminada del directorio Nuevos proyectos. Los elementos de este directorio se mostrarán en el cuadro de diálogo asistente Nuevo proyecto .
SortPriority	REG_DWORD	41 (x29)	Establece el orden en el que se mostrarán los proyectos en el nodo de árbol del cuadro de diálogo Nuevo proyecto .

NOMBRE	TIPO	DATA	DESCRIPCIÓN
NewProjectDialogOnly	REG_DWORD	0	0 indica que los proyectos de este tipo solo se muestran en el cuadro de diálogo Nuevo proyecto .

En el ejemplo siguiente se encuentra en el registro bajo la clave [HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, VisualStudio, 9.0Exp, InstalledProducts].

```
\FiguresProductSample
"Package"="{ACEF4EB2-57CF-11D2-96F4-000000000000}"
"UseInterface"=dword:00000001
```

NOMBRE	TIPO	DATA	DESCRIPCIÓN
Package	REG_SZ	%CLSID_Package%	Identificador de clase del VSPackage registrado.
UseInterface	REG_DWORD	1	1 indica que la interfaz de usuario se usará para interactuar con este proyecto. 0 indica que no hay interfaz de usuario.

Los archivos .vsz que controlan nuevos tipos de proyecto con frecuencia contienen una entrada RELATIVE_PATH. Esta ruta de acceso es relativa a la ruta de acceso especificada en la entrada .ProductDir del tipo de proyecto en la siguiente clave de instalación:

HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, VisualStudio, 7.0Exp, Configuración de instalación y

Por ejemplo, las plantillas de proyecto de Enterprise Frameworks agregan las siguientes entradas del Registro:

HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, VisualStudio, 7.0Exp, Setup, EF, ProductDir, C:, Archivos de programa, Microsoft Visual Studio, EnterpriseFrameworks, etc.

Esto significa que si incluye una entrada PROJECT_TYPE-EF en el archivo .vsz, el entorno busca los archivos .vsz en el directorio ProductDir especificado anteriormente.

Vea también

- [Lista de comprobación: creación de nuevos tipos de proyecto](#)
- [Elementos de un modelo de proyecto](#)
- [Creación de instancias de proyecto mediante generadores de proyecto](#)

Persistencia de un proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

La persistencia es una consideración de diseño clave para su proyecto. La mayoría de los proyectos usan elementos de proyecto que representan archivos; Visual Studio también admite proyectos cuyos datos no están basados en archivos. Tanto los archivos propiedad del proyecto como el archivo de proyecto deben conservarse. El IDE indica al proyecto que se guarde a sí mismo o a un elemento de proyecto.

Las plantillas para proyectos se pasan a la fábrica de proyectos. Las plantillas deben admitir la inicialización de todos los elementos de proyecto según los requisitos del tipo de proyecto específico. Estas plantillas se pueden guardar más adelante como archivos de proyecto y administrar por el IDE a través de la solución. Para obtener más información, vea [Creación de instancias](#) de proyecto mediante el uso de factorías y [soluciones](#) de proyectos .

Los elementos del proyecto pueden estar basados en archivos o no en archivos:

- Los elementos basados en archivos pueden ser locales o remotos. En los proyectos web en C-, por ejemplo, las conexiones a archivos en un sistema remoto persisten localmente, mientras que los propios archivos persisten en el sistema remoto.
- Los elementos no basados en archivos pueden guardar elementos en una base de datos o repositorio.

Confirmar modelos

Después de decidir dónde se encuentran los elementos del proyecto, debe elegir el modelo de confirmación adecuado. Por ejemplo, en un modelo basado en archivos con archivos locales, cada proyecto se puede guardar de forma autónoma. En un modelo de repositorio, puede guardar varios elementos en una transacción. Para obtener más información, vea [Decisiones](#) de diseño de tipo de proyecto .

Para determinar las extensiones [IPersistFileFormat](#) de nombre de archivo, los proyectos implementan la interfaz, que proporciona información que permite al cliente de un objeto implementar el cuadro de diálogo **Guardar como**, es decir, rellenar la lista desplegable Guardar como **tipo** y administrar la extensión de nombre de archivo inicial.

El IDE `IPersistFileFormat` llama a la interfaz en el proyecto para indicar que el proyecto debe conservar sus elementos de proyecto según corresponda. Por lo tanto, el objeto posee todos los aspectos de su archivo y formato. Esto incluye el nombre del formato del objeto.

En el caso de que `IPersistFileFormat` los elementos no sean archivos, sigue siendo cómo se conservan los elementos no basados en archivos. Los archivos de proyecto, como Visual Basic los archivos .vbp para proyectos o los archivos .vcproj para Visual C++ proyectos, también deben conservarse.

Para las acciones de guardado, el IDE examina la tabla de documentos [IVsPersistHierarchyItem](#) en [IVsPersistHierarchyItem2](#) ejecución (RDT) y la jerarquía pasa los comandos a las interfaces y las. El `IsItemDirty` método se implementa para determinar si el elemento se ha modificado. Si el elemento [SavelItem](#) tiene, el método se implementa para guardar el elemento modificado.

Los métodos `IVsPersistHierarchyItem2` de la interfaz se utilizan para determinar si se puede volver a cargar un elemento y, si el elemento puede ser, para volver a cargarlo. Además, [IgnoreItemFileChanges](#) el método se puede implementar para hacer que los elementos modificados se descarten sin guardarse.

Vea también

- [Lista de comprobación: creación de nuevos tipos de proyecto](#)

- Creación de instancias de proyecto mediante generadores de proyecto

Uso de MSBuild

06/04/2020 • 3 minutes to read • [Edit Online](#)

MSBuild proporciona un formato XML bien definido y extensible para crear archivos de proyecto que describen completamente los elementos de proyecto que se van a compilar, compilar tareas y generar configuraciones.

Consideraciones generales de MSBuild

Los archivos de proyecto de Visual C# MSBuild, Visual Basic por ejemplo, los archivos .csproj y .vbproj, contienen datos que se usan en tiempo de compilación, pero también pueden contener datos que se usan en tiempo de diseño. Los datos en tiempo de compilación se almacenan mediante primitivas de MSBuild, incluidos [Item Element \(MSBuild\)](#) y [Property Element \(MSBuild\)](#). Los datos en tiempo de diseño, que son datos específicos del tipo de proyecto y de los subtipos de proyecto relacionados, se almacenan en XML de forma libre reservado para él.

MSBuild no tiene compatibilidad nativa con objetos de configuración, pero proporciona atributos condicionales para especificar datos específicos de la configuración. Por ejemplo:

```
<OutputDir Condition="'$(Configuration)'=='release'">Bin\MyReleaseConfig</OutputDir>
```

Para obtener más información sobre los atributos condicionales, vea [Construcciones condicionales](#).

Ampliación de MSBuild para el tipo de proyecto

Las interfaces y API de MSBuild están Visual Studiosujetas a cambios en versiones futuras de . Por lo tanto, es prudente usar las clases de marco de paquete administrado (MPF) porque proporcionan blindaje de los cambios.

Managed Package Framework for Projects (MPFProj) proporciona clases auxiliares para crear y administrar un nuevo sistema de proyectos. Puede encontrar el código fuente y las instrucciones de compilación en [MPF for Projects - Visual Studio 2013](#).

Las clases MPF específicas del proyecto son las siguientes:

CLASE	IMPLEMENTACIÓN
<code>Microsoft.VisualStudio.Package.ProjectNode</code>	IVsProject3 IVsCfgProvider2 IPersistFileFormat IVsSolutionEvents
<code>Microsoft.VisualStudio.Package.ProjectFactory</code>	IVsProjectFactory
<code>Microsoft.VisualStudio.Package.HierarchyNode</code>	IVsHierarchy
<code>Microsoft.VisualStudio.Package.ProjectConfig</code>	IVsCfg IVsProjectCfg IVsBuildableProjectCfg IVsDebuggableProjectCfg

CLASE	IMPLEMENTACIÓN
<code>Microsoft.VisualStudio.Package.SettingsPage</code>	IPropertyPageSite

`Microsoft.VisualStudio.Package.ProjectElement` clase es un contenedor para los elementos de MSBuild.

Generadores de archivos únicos frente a tareas de MSBuild

Los generadores de archivos individuales solo son accesibles en tiempo de diseño, pero las tareas de MSBuild se pueden usar en tiempo de diseño y en tiempo de compilación. Por lo tanto, para obtener la máxima flexibilidad, use tareas de MSBuild para transformar y generar código. Para obtener más información, consulte [Herramientas personalizadas](#).

Vea también

- [Referencia de MSBuild](#)
- [MSBuild](#)
- [Herramientas personalizadas](#)

Agregar plantillas de proyecto y elemento de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Al crear sus propios tipos de proyecto, debe proporcionar compatibilidad para Visual Studio agregar nuevos proyectos y elementos de proyecto mediante los cuadros de diálogo de entorno de desarrollo integrado estándar (IDE). En los temas siguientes se describen diferentes técnicas para agregar proyectos y elementos de proyecto.

En esta sección

- [Contexto del proyecto](#)

Explica que el proyecto proporciona la mayor parte de la información de contexto para lo que sucede en el entorno.

- [Prioridad del proyecto](#)

Explica que un elemento de proyecto suele ser miembro de un proyecto para ayudar a evitar la ambigüedad sobre qué proyecto se usa para abrir el elemento.

- [Proyecto de Archivos Varios](#)

Proporciona información sobre los dos tipos de editores que se pueden usar para abrir archivos en un proyecto y el papel que desempeña el proyecto para determinar qué editor usar cuando se abre un elemento de proyecto.

- [Registrar plantillas de proyectos y elementos](#)

Explica lo que Visual Studio ocurre cuando se crea un proyecto.

- [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#)

Explica el proceso para agregar elementos al cuadro de diálogo **Agregar nuevo elemento**.

- [Agregar directorios al cuadro de diálogo Nuevo proyecto](#)

Proporciona un ejemplo de registro de un nuevo directorio que contiene plantillas personalizadas disponibles por un VSPackage.

- [Agregar directorios al cuadro de diálogo Agregar nuevo elemento](#)

Proporciona un ejemplo de registro de un nuevo conjunto de directorios para el **Agregar nuevo elemento** cuadro de diálogo.

- [Comandos definidos por IDE para ampliar los sistemas de proyectos](#)

Enumera los diferentes tipos de elementos de comando utilizados para ampliar los sistemas de proyecto.

- [CATID para objetos que se utilizan normalmente para extender proyectos](#)

Enumera los CATID para los Visual C++ objetos que se utilizan para extender , Visual C#y Visual Basic los sistemas de proyecto.

Secciones relacionadas

- [Cómo: Abrir editores específicos de proyectos](#)

Proporciona instrucciones paso a paso para abrir un elemento enlazado intrínsecamente a un editor específico para un proyecto.

- [Cómo: Abrir editores estándar](#)

Proporciona instrucciones paso a paso para abrir un editor estándar.

- [Subtipos de proyecto](#)

Proporciona vínculos a temas conceptuales de subtipo de proyecto. Los subtipos Visual C# de Visual Basic proyecto amplían los proyectos y los existentes.

- [Tipos de proyecto](#)

Proporciona vínculos a temas adicionales que ofrecen información sobre cómo diseñar nuevos tipos de proyecto.

Contexto del proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando el usuario agrega o trabaja con proyectos y elementos de proyecto, el IDE usa la noción de contexto de proyecto para determinar cómo se deben realizar varias operaciones.

Normalmente, los archivos son los objetos de proyecto estándar que el usuario crea explícitamente seleccionando el comando **Nuevo proyecto** o pone a disposición seleccionando el comando **Abrir proyecto** en el menú **Archivo**. En estos casos, los archivos se crean y se abren en el contexto de un proyecto y el tipo de proyecto define el contexto para editar el documento.

Algunos proyectos proporcionan un contexto muy rico. Por ejemplo, el proyecto administra un espacio de nombres de ámbito de proyecto, un espacio de nombres mediante programación o una conexión de base de datos de ámbito de proyecto para el enlace de datos. El usuario puede abrir con frecuencia archivos o conexiones de base de datos directamente mediante un objeto de proyecto determinado, como un elemento de proyecto que se muestra en el Explorador de soluciones.

En otras ocasiones, el contexto del proyecto de un elemento no se especifica explícitamente. Por ejemplo, el contexto de un elemento no está disponible cuando el usuario abre un archivo seleccionando el comando Abrir archivo **existente** en el menú **Archivo**, cuando el depurador funciona en un archivo o cuando el usuario hace clic en el comando **Buscar en archivos** del cuadro de diálogo Buscar y **reemplazar**. Para controlar estas situaciones, [IVsUIShellOpenDocument](#) el IDE llama para administrar el proceso de búsqueda del mejor proyecto para abrir un documento.

Vea también

- [Prioridad del proyecto](#)
- [Adición de plantillas de proyecto y de elementos de proyecto](#)

Prioridad del proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Normalmente, un elemento de proyecto es miembro de un solo proyecto de la solución. Por lo tanto, el IDE puede determinar fácilmente qué proyecto se usa para abrir el elemento. Sin embargo, si un elemento es miembro de más de un proyecto, el IDE usa un esquema de prioridad para determinar el mejor proyecto para abrir el elemento.

La lista siguiente muestra el esquema de prioridad del proyecto:

- El IDE `IsDocumentInProject` llama al método para cada proyecto de la solución para determinar si el documento es miembro de ese proyecto.
- Si el documento es miembro del proyecto, el proyecto responde con una prioridad que el proyecto asigna según su manejo de ese documento. Por ejemplo, un proyecto de lenguaje responde con una prioridad alta para sus archivos de origen de idioma, pero responde con una prioridad más baja para un tipo de archivo no reconocido que no se utiliza como parte de su proceso de compilación.
- Los proyectos que proporcionan editores o diseñadores personalizados específicos del proyecto para un documento también reciben una alta prioridad.
- La `VSDOCUMENTPRIORITY` enumeración proporciona los valores de prioridad del documento.
- El proyecto que especifica la prioridad más alta tiene el contexto para abrir el documento. Si dos proyectos devuelven valores de prioridad iguales, se prefiere el proyecto activo. Si ningún proyecto de la solución responde que puede abrir el documento, el IDE coloca el documento en el proyecto Archivos varios. Para obtener más información, consulte [Proyecto de archivos varios](#).

Vea también

- [Proyecto de archivos varios](#)
- [Apertura de editores para documentos abiertos](#)
- [Adición de plantillas de proyecto y de elementos de proyecto](#)

Proyecto de archivos varios

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando un usuario abre elementos de proyecto, el IDE asigna al proyecto Varios archivos cualquier elemento que no sea miembro de ningún proyecto de una solución.

Los proyectos desempeñan un papel importante a la hora de determinar qué editor se utiliza cuando un usuario abre un elemento de proyecto. Un proyecto se puede diseñar para abrir determinados archivos mediante un editor específico del proyecto o un editor estándar.

Un editor específico del proyecto normalmente requiere que el usuario tenga conocimientos especiales o use interfaces especiales del proyecto. Para obtener más información, consulte [Cómo: Abrir editores específicos de proyecto](#).

Un editor estándar puede abrir cualquier archivo de una extensión específica en cualquier proyecto. El usuario puede personalizar algunos editores Visual Studio estándar, como el editor de texto, para proyectos, pero aún conservan su carácter público. Los editores estándar se [OpenStandardEditor](#) crean mediante el método.

Si ningún proyecto de la solución responde que puede abrir un elemento de proyecto, el IDE proporciona un proyecto especial denominado proyecto de archivos varios que abre cualquier archivo.

Este proyecto especial prevé la apertura de un archivo fuera del contexto de un proyecto. Durante el procesamiento [OpenDocumentViaProject](#) del método, el proyecto Archivos varios siempre responde con una prioridad muy baja. Por lo tanto, el proyecto Archivos varios siempre cede a cualquier proyecto de mayor prioridad que pueda abrir archivos.

El proyecto Archivos varios no requiere que el usuario lo cree explícitamente con el cuadro de diálogo **Nuevo proyecto**. Además, el proyecto Archivos varios no administra permanentemente una lista de miembros del proyecto. Utiliza una característica opcional para registrar una lista de los archivos usados más recientemente para cada usuario.

Vea también

- [IVsProject3](#)
- [IVsUIShellOpenDocument](#)
- [VSDOCUMENTPRIORITY](#)
- [Apertura de editores específicos de proyecto](#)
- [Apertura de editores estándar](#)
- [Adición de plantillas de proyecto y de elementos de proyecto](#)
- [Adición de plantillas de proyecto y de elementos de proyecto](#)

Registro de plantillas para proyectos y elementos

06/04/2020 • 6 minutes to read • [Edit Online](#)

Los tipos de proyecto deben registrar los directorios donde se encuentran sus plantillas de proyecto y elemento de proyecto. Visual Studio usa la información de registro asociada a los tipos de proyecto para determinar qué mostrar en los cuadros de diálogo **Agregar nuevo proyecto** y **Agregar nuevo elemento**.

Para obtener más información acerca de las plantillas, vea [Agregar plantillas de proyecto y elemento de proyecto](#).

Entradas del Registro para Proyectos

En los ejemplos siguientes se muestran las entradas del Registro en HKEY_LOCAL_MACHINE, Software, Microsoft, VisualStudio *Versión*. Las tablas adjuntas explican los elementos utilizados en los ejemplos.

```
[Projects\{ProjectGUID}]
@="MyProjectType"
"DisplayName"="#2"
"Package"="{VSPackageGUID}"
"ProjectTemplatesDir"="C:\\MyProduct\\MyProjectTemplates"
```

NOMBRE	TIPO	DESCRIPCIÓN
@	REG_SZ	Nombre predeterminado de proyectos de este tipo.
DisplayName	REG_SZ	Identificador de recurso del nombre que se va a recuperar del archivo DLL satélite registrado en Paquetes.
Paquete	REG_SZ	ID de clase del paquete registrado en Paquetes.
ProjectTemplatesDir	REG_SZ	Ruta de acceso predeterminada de los archivos de plantilla de proyecto. Los archivos de plantilla de proyecto se muestran mediante la plantilla Nuevo proyecto .

Registro de plantillas de elementos

Debe registrar el directorio donde almacena las plantillas de elementos.

```
[Projects\{ProjectGUID}\AddItemTemplates\TemplateDirs\{VSPackageGUID}\1]
@="#7"
"TemplatesDir"="C:\\MyProduct\\MyProjectItemTemplates "
"TemplatesLocalizedSubDir"="#10"
"SortPriority"=dword:00000064
```

NOMBRE	TIPO	DESCRIPCIÓN
@	REG_SZ	Identificador de recurso para agregar plantillas de elemento.

NOMBRE	TIPO	DESCRIPCIÓN
TemplatesDir	REG_SZ	Ruta de acceso de los elementos del proyecto que se muestran en el cuadro de diálogo para el Asistente para agregar nuevo elemento .
TemplatesLocalizedSubDir	REG_SZ	Identificador de recurso de una cadena que nombra el subdirectorio de TemplatesDir que contiene plantillas localizadas. Dado Visual Studio que carga el recurso de cadena desde archivos DLL satélite si los tiene, cada DLL satélite puede contener un nombre de subdirectorio localizado diferente.
SortPriority	REG_DWORD	Establezca SortPriority para controlar el orden en que se muestran las plantillas en el cuadro de diálogo Agregar nuevo elemento . Los valores SortPriority más grandes aparecen anteriormente en la lista de plantillas.

Registro de filtros de archivos

Opcionalmente, puede registrar Visual Studio filtros que se usan cuando solicita nombres de archivo. Por ejemplo, Visual C# el filtro para el cuadro de diálogo **Abrir archivo** es:

Archivos de Visual*CTM*(.cs,* .resx, .settings,* .xsd,* .wsdl); *.cs,* .resx,* .settings,.xsd, .wsdl)**

Para admitir el registro de varios filtros, cada filtro se registra en\<su propia\<subclave en HKEY_LOCAL_MACHINE, Software, Microsoft, VisualStudio *Versión*>, Proyectos, *ProjectGUID*>, Filtros,\<subclave>. El nombre de la subclave es arbitrario; Visual Studio ignora el nombre de la subclave y utiliza solo sus valores.

Puede controlar los contextos en los que se utiliza un filtro estableciendo indicadores, que se muestran en la tabla siguiente. Si un filtro no tiene ninguna marca establecida, aparecerá después de los filtros comunes en el cuadro de diálogo **Agregar elemento existente** y el cuadro de diálogo **Abrir archivo**, pero no se usará en el cuadro de diálogo **Buscar en archivos**.

```
[Projects\{ProjectGUID}\Filters\MyLanguageFilter]
@="#3"
"CommonOpenFilesFilter"=dword:00000000
"CommonFindFilesFilter"=dword:00000000
"FindInFilesFilter"=dword:00000000
"NotOpenFileFilter"=dword:00000000
"NotAddExistingItemFilter"=dword:00000000
"SortPriority"=dword:00000064
```

NOMBRE	TIPO	DESCRIPCIÓN
CommonFindFilesFilter	REG_DWORD	Convierte el filtro en uno de los filtros comunes del cuadro de diálogo Buscar en archivos . Los filtros comunes se enumeran en la lista de filtros antes de que los filtros no estén marcados como comunes.

NOMBRE	TIPO	DESCRIPCIÓN
CommonOpenFilesFilter	REG_DWORD	Convierte el filtro en uno de los filtros comunes del cuadro de diálogo Abrir archivo . Los filtros comunes se enumeran en la lista de filtros antes de que los filtros no estén marcados como comunes.
FindInFilesFilter	REG_DWORD	Enumera el filtro después de los filtros comunes en el cuadro de diálogo Buscar en archivos .
NotOpenFileFilter	REG_DWORD	Indica que el filtro no se utiliza en el cuadro de diálogo Abrir archivo .
NotAddExistingItemFilter	REG_DWORD	Indica que el filtro no se utiliza en el cuadro de diálogo Agregar elemento existente .
SortPriority	REG_DWORD	Establezca SortPriority para controlar el orden en que se muestran los filtros. Los valores SortPriority más grandes aparecen anteriormente en la lista de filtros.

Estructura de directorios

VSPackages puede colocar archivos de plantilla y carpetas en cualquier lugar de un disco local o remoto, siempre y cuando la ubicación se registre a través del entorno de desarrollo integrado (IDE). Sin embargo, para facilitar la organización, se recomienda la siguiente estructura de directorios en la ruta de instalación del producto.

Plantillas

- Proyectos (contiene las plantillas de proyecto)

Aplicaciones

- Componentes

\ ...

- ProjectItems (contiene los elementos del proyecto)

- Clase

- Formulario

Página web

- HelperFiles (contiene los archivos utilizados en elementos de proyecto de varios archivos)

- WizardFiles

Vea también

- [Adición de plantillas de proyecto y de elementos de proyecto](#)
- [Asistentes](#)
- [Localizar aplicaciones](#)

- CATID para los objetos que se usan normalmente para ampliar proyectos

Agregar elementos al cuadro de diálogo Agregar nuevo elemento

06/04/2020 • 10 minutes to read • [Edit Online](#)

El proceso para agregar elementos al cuadro de diálogo **Agregar nuevo elemento** comienza con las claves del Registro. Como se muestra en las siguientes entradas del Registro, la sección **AddItemTemplates** contiene la ruta de acceso y el nombre del directorio en el que se colocan los elementos disponibles en el cuadro de diálogo **Agregar nuevo elemento**.

NOTE

La tabla inmediatamente después del segmento de código contiene información adicional sobre la entrada del Registro.

Esta sección se encuentra en **HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 14.0Exp, Proyectos**.

El primer GUID es el CLSID para proyectos de este tipo; el segundo GUID indica el tipo de proyecto registrado para las plantillas Agregar elementos:

```
\C061DB26-5833-11D2-96F5-0000000000000000 \AddItemTemplates\TemplatesDir\áACEF4EB2-57CF-11D2-96F4-000000000000 1\
```

@ • #6

TemplatesDir = \>\<\>\<\>\<Visual Studio>\SDK ruta de instalación VSIntegration SomeFolder SomeProject SomeProject Items<>

SortPriority á dword:00000064

NOMBRE	TIPO	DATOS (DESDE EL ARCHIVO .RGS)	DESCRIPCIÓN
(Predeterminado)	REG_SZ	%IDS_ADDITEM_TEMPLATE S_ENTRY%	Identificador de recurso para agregar plantillas de elemento .
Val TemplatesDir	REG_SZ	%TEMPLATE_PATH%\<SomeProjectItems>	Ruta de acceso de los elementos del proyecto que se muestran en el cuadro de diálogo del asistente Agregar nuevo elemento .
Val SortPriority	REG_DWORD	100x64()	Determina el criterio de ordenación en el nodo de árbol de los archivos que se muestran en el cuadro de diálogo Agregar nuevo elemento .

NOTE

Los GUID para los tipos de proyecto de Visual C- y Visual Basic son los siguientes:

- Visual C#: FAE04EC0-301F-11D3-BF4B-00C04F79EFBC
- Visual Basic: F184B08F-C81C-45F6-A57F-5ABD9991F28F

El directorio que aparece para **TemplatesDir**, que es `%TEMPLATE_PATH%\<>SomeProjectItems`, es el nodo en el lado izquierdo del árbol del cuadro de diálogo **Agregar nuevo elemento**. Los elementos adicionales del árbol se basan en el subdirectorio dentro de ese directorio raíz. Los archivos disponibles para agregar al proyecto son los elementos del panel derecho del cuadro de diálogo **Agregar nuevo elemento**.

Normalmente, esta carpeta contendrá los archivos de plantilla para el proyecto, como un archivo HTML o `.cpp` de plantilla, y cualquier archivo `.vsz` para iniciar asistentes. Para controlar cómo se muestran los elementos, también puede incluir archivos `.vsdir` para localizar nombres de directorio e iconos. La cadena localizada es el título que aparece en el cuadro de diálogo que representa este nodo en el árbol del cuadro de diálogo **Agregar nuevo elemento**.

Sin embargo, no es necesario tener todo en un archivo `.vsdir`. Puede tener un archivo `.vsdir` para cada elemento del directorio. Para obtener más información, vea [Archivos del asistente \(.vsz\)](#) y Archivos de descripción del directorio de plantilla ([.vsdir](#)).

NOTE

Los archivos `.vsdir` de los directorios de plantilla son opcionales. Si solo desea colocar un elemento de proyecto en el directorio y mostrarlo en el cuadro de diálogo **Agregar nuevo elemento**, puede colocar ese archivo en el directorio templates especificado en la instrucción **TemplatesDir**. El archivo se mostrará en el panel derecho del cuadro de diálogo **Agregar nuevo elemento** para ese proyecto. Sin embargo, si desea mostrar un título localizado para el archivo o un icono, debe incluir al menos un archivo `.vsdir` en el directorio templates.

Elementos de proyecto de grupo

Si desea contener grupos de plantillas en carpetas en el árbol del cuadro de diálogo **Agregar nuevo elemento**, debe tener subdirectorios en el directorio de plantilla raíz con los elementos en ellos. Cuando se muestra el cuadro de diálogo **Agregar nuevo elemento** a los usuarios, también verán las subcarpetas y podrán seleccionar elementos del proyecto de ellos.

La prioridad de ordenación en el segmento de código determina dónde se creará este directorio de plantilla en el árbol en relación con otros elementos del nodo de árbol. Para el cuadro de diálogo **Agregar nuevo elemento**, la prioridad de ordenación es todo lo que debe incluir para que los elementos se muestren en la ubicación correcta en el cuadro de diálogo.

También puede implementar [IVsFilterAddProjectItemDlg2](#) la interfaz para filtrar lo que se muestra en el cuadro de diálogo **Agregar nuevo elemento**. Al implementar esta interfaz, puede configurar un directorio de plantilla en el disco que contenga, por ejemplo, 50 archivos de plantilla y asistente. De este modo, puede tener diferentes tipos de proyecto con 20 archivos que pertenecen a un tipo de proyecto, los otros 30 archivos que pertenecen a otro tipo de proyecto y todos los archivos disponibles en un tipo general de proyecto. De esta manera, dependiendo de la plantilla de proyecto que se cree, puede mostrar un conjunto diferente de archivos de plantilla.

Por ejemplo, en un proyecto de Visual Basic, es posible que tenga proyectos web y proyectos de cliente. Los formularios web no son elementos útiles para agregar a un proyecto de cliente y los formularios de Windows no son elementos útiles para agregar a un proyecto de servidor web. Por lo tanto, puede crear un directorio de plantilla que contenga todos los archivos para ambos tipos de proyecto. A continuación, [IVsFilterAddProjectItemDlg2](#) mediante la implementación de , puede ocultar elementos que no se deben mostrar

en función del tipo de proyecto o configuración del proyecto en el proyecto.

Filtrar elementos de proyecto

`IVsFilterAddProjectItemDlg2` proporciona filtrado de elementos en el árbol (panel izquierdo) y archivos de proyecto (panel derecho) de las siguientes maneras:

- Por los nombres localizados (los subtítulos que se muestran en el `IVsFilterAddProjectItemDlg` cuadro de diálogo que se encuentra en el archivo `.vsdir`) proporcionado por .
- Por los nombres reales de los archivos y carpetas en el disco `IVsFilterAddProjectItemDlg` (no localizado — sin archivo `.vsdir`) proporcionado por .
- Por categoría, `IVsFilterAddProjectItemDlg2` proporcionada por .

Para filtrar por categoría, proporcione una cadena de categoría a un elemento del archivo `.vsdir`, como *el formulario web* o el elemento de *cliente* en Visual Basic. A continuación, el código del cuadro de diálogo recupera la clasificación de categoría del archivo `.vsdir` y se la pasa. A continuación, puede pasar esa información a la implementación de filtrar el **Agregar nuevo elemento** cuadro de

`IVsFilterAddProjectItemDlg2` diálogo por categorías. También puede filtrar elementos para páginas Web o como casos de aplicación Win32 de cliente. Además, puede Visual C++ identificar elementos etiquetados como microsoft Foundation Classes (MFC) o elementos de biblioteca de plantillas activas (ATL). Al identificar estas posiciones, el sistema de proyectos puede definir sus propias clasificaciones para que el sistema se pueda filtrar en función de categorías y clasificaciones.

Si implementa esta funcionalidad de filtro, no tiene que asignar una tabla de cada elemento que se debe ocultar. Simplemente puede clasificar elementos en tipos y colocar las clasificaciones en el archivo o archivos `.vsdir`. A continuación, puede ocultar cualquiera de los elementos que tienen una clasificación específica mediante la implementación de la interfaz. De este modo, puede hacer que los elementos del cuadro de diálogo **Agregar nuevo elemento** sean dinámicos en función del estado del proyecto.

Vea también

- [IVsFilterAddProjectItemDlg2](#)
- [Registrar plantillas de proyectos y elementos](#)
- [CATID para objetos que se utilizan normalmente para extender proyectos](#)
- [Agregar plantillas de proyecto y elemento de proyecto](#)
- [Archivos de descripción del directorio de plantillas \(.vsdir\)](#)
- [Archivo Wizard \(.vsz\)](#)

Agregar directorios al cuadro de diálogo Nuevo proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Al crear nuevos tipos de proyecto, también puede registrar un nuevo directorio en el cuadro de diálogo **Nuevo proyecto** para mostrarlos para usarlos como plantillas. En el ejemplo de código siguiente se explica cómo registrar un nuevo directorio, también conocido como nodo. En el ejemplo, se registran las plantillas expuestas por el VSPackage, *CLSID_Package*. Como resultado, el lado izquierdo del cuadro de diálogo **Nuevo proyecto** ofrece el nodo agregado, con un nombre determinado por el *recurso Folder_Label_ResID*. Este recurso se carga desde el archivo DLL satélite de VSPackage.

El valor **Folder** representa un GUID de una carpeta en la que se muestra el nodo *Folder_Label_ResID*. En el ejemplo, el GUID representa la carpeta **Otros proyectos** en el panel **Tipos de proyecto** del cuadro de diálogo **Nuevo proyecto**. Si el valor **Otros proyectos** está ausente, la etiqueta se coloca en el nivel superior.

El `TemplatesDir` valor especifica la ruta de acceso completa del directorio que contiene las plantillas de proyecto. Estos archivos pueden ser archivos *.vsz* o archivos de plantilla típicos que se van a clonar.

Si especifica `TemplatesLocalizedSubDir`, debe ser el identificador de recurso de `TemplatesDir` una cadena que nombra el subdirectorio que contiene plantillas localizadas. Dado Visual Studio que carga el recurso de cadena desde un archivo DLL satélite si tiene uno, cada DLL satélite puede contener un nombre de subdirectorio diferente.

El `SortPriority` valor especifica una prioridad de ordenación.

```
NoRemove NewProjectTemplates
{
    NoRemove TemplateDirs
    {
        ForceRemove %CLSID_Package%
        {
            ForceRemove /1 = s '%Folder_Label_ResID%'
            {
                val Folder = s '{DCF2A94A-45B0-11D1-ADBF-00C04FB6BE4C}'
                val TemplatesDir = s '%Template_Path%'
                val TemplatesLocalizedSubDir = s '#100'
                val SortPriority = d 1000
            }
        }
    }
}
```

Vea también

- [Registrar plantillas de proyectos y elementos](#)
- [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#)
- [Agregar directorios al cuadro de diálogo Agregar nuevo elemento](#)

Agregar directorios al cuadro de diálogo Agregar nuevo elemento

06/04/2020 • 2 minutes to read • [Edit Online](#)

En el ejemplo de código siguiente se muestra cómo registrar un nuevo conjunto de directorios para el **Agregar nuevo elemento** cuadro de diálogo. Los directorios del cuadro de diálogo **Agregar nuevo elemento** son diferentes para cada proyecto. Por lo tanto, los directorios se registran en la subclave **Proyectos**, que se encuentra en HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 8.0Exp, Proyectos.

Script de registro

```
NoRemove Projects
{
    NoRemove %GUID_Project%
    {
        NoRemove AddItemTemplates
        {
            NoRemove TemplateDirs
            {
                ForceRemove %CLSID_Package%
                {
                    ForceRemove /1 = s '%Folder_Label_ResID%'
                    {
                        val TemplatesDir = s '%Template_Path%'
                        val SortPriority = d 2000
                    }
                }
            }
        }
    }
}
```

El `%Template_Path%` valor especifica la ruta de acceso completa del directorio que contiene las plantillas de proyecto. Estas plantillas pueden ser archivos .vsz o archivos de plantilla prototípicos que se van a clonar.

El `SortPriority` valor especifica una prioridad de ordenación.

Agregar elementos a un proyecto existente

También puede agregar elementos a un proyecto existente. Por ejemplo, Visual C# para un proyecto, puede agregar elementos a la * <carpeta raíz>, Archivos de programa, Microsoft Visual Studio, VC, CSharpProjectItems, LocalProjectItems.* En este `%GUID_Project%` caso, es el GUID para un proyecto de C- ('FAE04EC0-301F-11D3-BF4B-00C04F79EFBC').

También puede ampliar un proyecto existente programando un subtipo de proyecto. Con un subtipo de proyecto, puede extender un proyecto sin escribir un nuevo tipo de proyecto. Para obtener más información acerca de los subtipos de proyecto, vea [Subtipos de proyecto](#).

Vea también

- [Registrar plantillas de proyectos y elementos](#)
- [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#)

- [Agregar directorios al cuadro de diálogo Nuevo proyecto](#)

Comandos definidos por el IDE para ampliar sistemas del proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Cuando desee ampliar los sistemas de proyecto, puede Visual Studio utilizar comandos y grupos de comandos proporcionados por el IDE.

En las secciones siguientes se enumeran los elementos de comando que son especialmente útiles para ampliar los sistemas de proyecto.

Menús de comandos

En la tabla siguiente se muestran los menús de comandos que son ubicaciones útiles para colocar comandos de alto nivel que invocan un extensor de proyecto.

MENÚ DE COMANDOS	DESCRIPCIÓN
IDM_VS_MENU_PROJECT	El menú de nivel superior del proyecto .
IDM_VS_TOOL_PROJWIN	La barra de herramientas del Explorador de soluciones .

Menús contextuales

En la tabla siguiente se muestran los menús contextuales que se aplican cuando se selecciona un único nodo en el **Explorador** de soluciones o cuando hay varias selecciones homogéneas en el **Explorador** de soluciones, que es cuando todos los nodos seleccionados son del mismo tipo.

MENÚ CONTEXTUAL	DESCRIPCIÓN
IDM_VS_CTXT_PROJNODE	Se aplica cuando se selecciona el nodo del proyecto.
IDM_VS_CTXT_ITEMNODE	Se aplica cuando se selecciona un archivo.
IDM_VS_CTXT_FOLDERNODE	Se aplica cuando se selecciona una carpeta.
IDM_VS_CTXT_WEBREFFOLDER	Se aplica cuando se selecciona la carpeta Referencia web.
IDM_VS_CTXT_REFERENCEROOT	Se aplica cuando se selecciona el nodo raíz de referencias denominado "Referencias".
IDM_VS_CTXT_REFERENCE	Se aplica cuando se seleccionan nodos de referencia; estos incluyen solo referencias de ensamblado, COM y proyecto. No incluye referencias Web.

En la tabla siguiente se muestran los menús contextuales que se aplican cuando la selección en el **Explorador** de soluciones abarca varias jerarquías,

MENÚ CONTEXTUAL	DESCRIPCIÓN
IDM_VS_CTXT_XPROJ_SLNPROJ	Se aplica cuando la selección actual contiene el nodo de solución y los nodos raíz del proyecto.
IDM_VS_CTXT_XPROJ_SLNITEM	Se aplica cuando la selección actual contiene el nodo de solución y los elementos del proyecto.
IDM_VS_CTXT_XPROJ_MULTIPROJ	Se aplica cuando la selección actual consta únicamente de varios nodos de proyecto raíz.
IDM_VS_CTXT_XPROJ_PROJITEM	Se aplica cuando la selección actual contiene una combinación de nodos de proyecto raíz y elementos de proyecto. Además, la selección puede contener el nodo de solución.
IDM_VS_CTXT_XPROJ_MULTIITEM	Se aplica cuando la selección actual contiene elementos de proyecto de varios proyectos de la solución o cuando se seleccionan elementos de diferentes tipos en el mismo proyecto.

Grupos de comandos

En la tabla siguiente se muestran los grupos de comandos que [IDM_VS_CTXT_PROJNODE](#) puede utilizar al ampliar proyectos y a los que puede acceder a través del menú contextual.

GRUPO DE COMANDOS	DESCRIPCIÓN
IDG_VS_CTXT_PROJECT_BUILD	Comandos para crear, reconstruir e implementar el proyecto.
IDG_VS_CTXT_COMPILELINK	Comandos para compilar y vincular el proyecto.
IDG_VS_CTXT_PROJECT_CONFIG	Comandos que establecen la configuración del proyecto y el orden de compilación.
IDG_VS_CTXT_PROJECT_ADD	Comandos que agregan elementos al proyecto.
IDG_VS_CTXT_PROJECT_START	Comandos que establecen el proyecto de inicio asociado a la tecla F5.
IDG_VS_CTXT_PROJECT_SAVE	Comandos para guardar elementos de proyecto.
IDG_VS_CTXT_PROJECT_DEBUG	Comandos para la depuración.
IDG_VS_CTXT_PROJECT_SCC	Comandos para el control de código fuente.
IDG_VS_CTXT_PROJECT_TRANSFER	Comandos para cortar, copiar y pegar operaciones.
IDG_VS_CTXT_PROJECT_PROPERTIES	Comandos que proporcionan acceso al cuadro de diálogo Propiedades del proyecto .

Vea también

- [Cómo VSPackages agrega elementos de la interfaz de usuario](#)
- [Creación de grupos reutilizables de botones](#)

CATID para objetos que se utilizan normalmente para extender proyectos

06/04/2020 • 3 minutes to read • [Edit Online](#)

En la tabla siguiente se enumeran los `Project` `ProjectItem` CATID Visual Basic Visual C# que Visual C++ se utilizan para extender y automatizar objetos para , , y proyectos. Estos CATID se definen en *VSLangProj.olb*.

Listado de CATIDs

NOMBRE	GUID
prjCATIDProject	N.o 610D4614-D0D5-11D2-8599-006097C68E81
prjCATIDProjectItem	N.o 610D4615-D0D5-11D2-8599-006097C68E81

Visual Basic CATIDs

En la tabla siguiente se enumeran los Visual Basic CATID que se utilizan para extender objetos de exploración. Todos ellos se definen en *VSLangProj.olb*.

NOMBRE	GUID
prjCATIDVBProjectBrowseObject	• E0FDC879-C32A-4751-A3D3-0B3824BD575F?
prjCATIDVBProjectConfigBrowseObject	N.o 67F8DD11-14EB-489b-87F0-F01C52AF3870
prjCATIDVBFileBrowseObject	• EA5BD05D-3C72-40A5-95A0-28A2773311CA
prjCATIDVBFolderBrowseObject	N.o 932DC619-2EAA-4192-B7E6-3D15AD31DF49
prjCATIDVBReferenceBrowseObject	N.o 2289B812-8191-4e81-B7B3-174045AB0CB5

CatID de Visual C.

Los siguientes CATID se pueden Visual C# utilizar para extender objetos de exploración. Todos ellos se definen en *VSLangProj.olb*.

NOMBRE	GUID
prjCATIDCSharpProjectBrowseObject	N.o 4EF9F003-DE95-4d60-96B0-212979F2A857
prjCATIDCSharpProjectConfigBrowseObject	A12CE10A-227F-4963-ADB6-3A43388513CA
prjCATIDCSharpFileBrowseObject	N.o 8D58E6AF-ED4E-48B0-8C7B-C74EF0735451
prjCATIDCSharpFolderBrowseObject	N.o 914FE278-054A-45DB-BF9E-5F22484CC84C

NOMBRE	GUID
prjCATIDCSharpReferenceBrowseObject	N.o 2F0FA3B8-C855-4a4e-95A5-CB45C67D6C27

CatIDcCCC+C++

Los Visual C++ siguientes CATID del sistema de proyectos Visual Studio no se exponen en las bibliotecas de tipos de .NET 2003 y deben incluirse en el código siempre que desee ampliar estos objetos de proyecto. Estos CATIDs se incluirán en las bibliotecas de tipos en versiones posteriores de Visual Studio.

NOMBRE	GUID
CVCProjectNode	EE8299CB-19B6-4f20-ABEA-E1FD9A33B683?
CVCFolderNode	EE8299CA-19B6-4f20-ABEA-E1FD9A33B683?
CVCFileNode	EE8299C9-19B6-4f20-ABEA-E1FD9A33B683?

En el ejemplo de código siguiente se muestra cómo programar estos CATIDs en el código.

```
const LPOLESTR CVCProjectNode::s_wszCATID = L"{EE8299CB-19B6-4f20-ABEA-E1FD9A33B683}";
const LPOLESTR CVCFolderNode::s_wszCATID = L"{EE8299CA-19B6-4f20-ABEA-E1FD9A33B683}";
const LPOLESTR CVCFileNode::s_wszCATID = L"{EE8299C9-19B6-4f20-ABEA-E1FD9A33B683}";
```

Los Visual C++ siguientes CATID del sistema de proyectos tampoco Visual Studio se exponen en las bibliotecas de tipos de .NET 2003 y deben incluirse en el código siempre que desee extender estos objetos de proyecto. Estos CATID solo están Visual Studio disponibles en .NET 2003 y Visual Studio no estarán disponibles en las versiones posteriores de .

NOMBRE	GUID
CVCAssemblyReferenceNode	•FE8299C9-19B6-4f20-ABEA-E1FD9A33B683?
CVCProjectReferenceNode	N.o 593DCFCE-20A7-48e4-ACA1-49ADE9049887?
CVCActiveXReferenceNode	N.o 9E8182D3-C60A-44f4-A74B-14C90EF9CACE
CVCReferences	•FE8299CA-19B6-4f20-ABEA-E1FD9A33B683?

En el ejemplo de código siguiente se muestra cómo programar estos CATIDs en el código:

```
const LPOLESTR CVCAssemblyReferenceNode::s_wszCATID = L"{FE8299C9-19B6-4f20-ABEA-E1FD9A33B683}";
const LPOLESTR CVCProjectReferenceNode::s_wszCATID = L"{593DCFCE-20A7-48e4-ACA1-49ADE9049887}";
const LPOLESTR CVCActiveXReferenceNode::s_wszCATID = L"{9E8182D3-C60A-44f4-A74B-14C90EF9CACE}";
const LPOLESTR CVCReferences::s_wszCATID = L"{FE8299CA-19B6-4f20-ABEA-E1FD9A33B683}";
```

Los GUID para Visual C# los Visual Basic tipos de proyecto y y se muestran en la tabla siguiente.

TIPO DE PROYECTO	GUID
Visual C#	•FAE04EC0-301F-11D3-BF4B-00C04F79EFBC

TIPO DE PROYECTO	GUID
Visual Basic	F184B08F-C81C-45F6-A57F-5ABD9991F28F

Vea también

- [Agregar plantillas de proyecto y elemento de proyecto](#)
- [Registrar plantillas de proyectos y elementos](#)

Apertura y guardado de elementos de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Al agregar un nuevo tipo de proyecto, debe administrar la Visual Studio apertura y el guardado de los archivos de proyectos en el entorno de desarrollo integrado (IDE). En los temas siguientes se describen los diferentes enfoques para abrir y guardar archivos.

En esta sección

- [Visualización de archivos mediante el comando Abrir archivo](#)

Proporciona una explicación paso a paso de cómo el IDE controla el comando **Abrir archivo** y el rol de los proyectos para responder a este comando.

- [Visualización de archivos mediante el comando Abrir con](#)

Proporciona una explicación detallada paso a paso de cómo el IDE controla el comando **Abrir con**, lo que solicita la apertura de un archivo que tiene alguna opción de editores estándar.

- [Apertura de editores específicos de proyecto](#)

Proporciona instrucciones paso a paso para especificar que los archivos de un tipo determinado en el proyecto se deben abrir mediante un editor específico del proyecto.

- [Apertura de editores estándar](#)

Proporciona instrucciones paso a paso para especificar cómo habilitar el IDE para abrir un editor estándar para los archivos en el tipo de proyecto.

- [Apertura de editores para documentos abiertos](#)

Proporciona instrucciones paso a paso para abrir un editor específico del proyecto para un archivo abierto.

- [Guardado de un documento estándar](#)

Proporciona una explicación detallada de cómo el IDE controla los comandos **Guardar**, **Guardar como** y **Guardar todo** para un documento abierto en un editor estándar.

- [Guardado de un documento personalizado](#)

Proporciona un diagrama y una explicación detallada de cómo el IDE controla los comandos **Guardar**, **Guardar como** y **Guardar todo** para los documentos abiertos en un editor personalizado.

- [Determinación del editor que abre un archivo en un proyecto](#)

Describe el proceso que sigue el IDE para seleccionar el editor o diseñador adecuado para un archivo.

Secciones relacionadas

- [Creación de diseñadores y editores personalizados](#)

Enumera los cuatro tipos de editores que el IDE puede hospedar y proporciona descripciones de cada editor.

- [Tipos de proyecto](#)

Describe cómo los proyectos controlan la forma en que se compila y compila el código, cómo se abren los

editores y cómo se da formato a los elementos de proyecto.

Mostrar archivos mediante el comando Abrir archivo

06/04/2020 • 4 minutes to read • [Edit Online](#)

Los pasos siguientes describen cómo el IDE controla el Visual Studio comando **Abrir archivo**, que está disponible en el menú **Archivo** de . Los pasos también describen cómo los proyectos deben responder a las llamadas que se originan a partir de este comando.

Cuando un usuario hace clic en el comando **Abrir archivo** del menú **Archivo** y selecciona un archivo en el cuadro de diálogo **Abrir archivo**, se produce el siguiente proceso:

1. Mediante la tabla de documentos en ejecución, el IDE determina si el archivo ya está abierto en un proyecto.
 - Si el archivo está abierto, el IDE vuelve a aparecer en la ventana.
 - Si el archivo no está abierto, el IDE llama `IsDocumentInProject` para consultar cada proyecto para determinar qué proyecto puede abrir el archivo.

NOTE

En la implementación del proyecto de , proporcione un valor de `IsDocumentInProject` prioridad que indique el nivel en el que el proyecto abre el archivo. Los valores de `VSDOCUMENTPRIORITY` prioridad se proporcionan en la enumeración.

2. Cada proyecto responde con un nivel de prioridad que indica la importancia que le da a ser el proyecto para abrir el archivo.
3. El IDE utiliza los siguientes criterios para determinar qué proyecto abre el archivo:
 - El proyecto que responde con `DP_Intrinsic` la prioridad más alta () abre el archivo. Si más de un proyecto responde con esta prioridad, el primer proyecto que responda abre el archivo.
 - Si ningún proyecto responde con `DP_Intrinsic` la prioridad más alta (), pero todos los proyectos responden con la misma prioridad inferior, el proyecto activo abre el archivo. Si no hay ningún proyecto activo, el primer proyecto que responda abre el archivo.
 - Si ningún proyecto reclama la `DP_Unsupported` propiedad del archivo (), el proyecto Archivos varios abre el archivo.

Si se crea una instancia del proyecto Archivos varios, el `DP_CanAddAsExternal` proyecto siempre responde con el valor . Este valor indica que el proyecto puede abrir el archivo. Este proyecto se utiliza para alojar archivos abiertos que no están en ningún otro proyecto. La lista de elementos de este proyecto no se conserva; este proyecto solo está visible en el **Explorador** de soluciones cuando se usa para abrir un archivo.

Si el proyecto Archivos varios no indica que puede abrir el archivo, no se ha creado una instancia del proyecto. En este caso, el IDE crea una instancia del proyecto Archivos varios e indica al proyecto que abra el archivo.

4. Tan pronto como el IDE determina qué proyecto `OpenItem` abre el archivo, llama al método en ese proyecto.
5. A continuación, el proyecto tiene la opción de abrir el archivo mediante un editor específico del proyecto o

un editor estándar. Para obtener más información, consulte [Cómo: Abrir editores específicos](#) del proyecto y [Cómo: Abrir editores estándar](#), respectivamente.

Vea también

- [Mostrar archivos mediante el comando Abrir con](#)
- [Abrir y guardar elementos del proyecto](#)
- [Cómo: Abrir editores específicos de proyectos](#)
- [Cómo: Abrir editores estándar](#)

Mostrar archivos mediante el comando Abrir con

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un proyecto puede pedir al IDE que muestre el cuadro de diálogo **Abrir con**. Esta solicitud solicita al usuario que abra un archivo que tiene una selección de editores estándar. Los pasos siguientes describen este proceso:

1. El proyecto `OpenStandardEditor` llama , especificando un valor de `OSE_UseOpenWithDialog` para el `OSEOpenDocEditor` parámetro.
2. En función de la extensión de nombre de archivo del documento, el IDE determina qué editores enumerados en el registro pueden abrir el documento especificado y muestra esta información en el cuadro de diálogo **Abrir con**.

NOTE

Los proyectos que tienen un editor intrínseco que se debe incluir en el cuadro de diálogo **Abrir con** deben registrar un generador de editores para cada uno de estos editores. Los editores intrínsecos solo funcionan junto con un tipo `CreateEditorInstance` determinado de proyecto, que se aplica en la implementación del método. El IDE tiene un generador de editores integrado para el editor de texto principal y el editor binario. El IDE también crea una instancia de un generador de editores en nombre de cada asociación de archivos de Windows registrada. Un ejemplo de este tipo de archivo es Microsoft Word.

3. Tan pronto como el usuario selecciona un elemento desde el **abrir con** cuadro `OpenStandardEditor` de diálogo, el IDE, a continuación, abre el documento mediante una llamada a método. Para obtener más información, consulte [Cómo: Abrir editores estándar](#).

Vea también

- [Abrir y guardar elementos del proyecto](#)
- [Mostrar archivos mediante el comando Abrir archivo](#)
- [Cómo: Abrir editores estándar](#)

06/04/2020 • 5 minutes to read • [Edit Online](#)

```

sequenceDiagram
    participant UIAction as UI Action
    participant CH as Command Handler
    participant RDT as Running Document Table
    participant P as Project
    participant IPHI as IVsPersistHierarchyItem
    participant IPDD as IVsPersistDocData2
    participant DD as DocData
    participant IPFF as IPersistFileFormat
    participant SSVS as SVsUIShell::SaveDocDataToFile
    participant SAD as SaveAs Dialog

    UIAction->>CH
    CH->>RDT: SaveAll
    RDT->>P
    P-->>IPHI: 
    IPHI-->>IPDD: 
    IPDD-->>DD: 
    DD-->>IPFF: 
    DD->>SSVS: SaveAs
    SSVS->>SAD

```

control de comandos para un editor estándar

1. Cuando se seleccionan los comandos **Guardar SVsShellMonitorSelection** y **Guardar como**, el entorno utiliza el servicio para determinar la ventana de documento activa y, por lo tanto, qué elementos se deben guardar. Una vez que se conoce la ventana de documento activa, el entorno busca el puntero de jerarquía y el identificador de elemento (itemID) para el documento en la tabla de documentos en ejecución. Para obtener más información, consulte Ejecución de tabla de [documentos](#).

2. Cuando la solución **QueryStatus** recibe una llamada, recorre en iteración el conjunto de elementos seleccionados (es decir, las múltiples selecciones expuestas por el **SVsShellMonitorSelection** servicio).

3. En cada elemento de la selección, la solución utiliza el puntero de jerarquía para llamar al `IsItemDirty` método para determinar si se debe habilitar el comando de menú **Guardar**. Si uno o más elementos están sucios, el comando **Guardar** está habilitado. Si la jerarquía utiliza un editor estándar, la jerarquía delega la `IsDocDataDirty` consulta de estado sucio en el editor llamando al método.

4. En cada elemento seleccionado que está sucio, la **Saveltem** solución usa el puntero de jerarquía para llamar al método en las jerarquías adecuadas.

Es común que la jerarquía utilice un editor estándar para editar el documento. En este caso, el objeto de datos `IVsPersistDocData2` de documento para ese editor debe admitir la interfaz. Al recibir `SaveItem` la llamada al método, el proyecto debe informar `SaveDocData` al editor de que el documento se está guardando llamando al método en el objeto de datos del documento. El editor puede permitir que el entorno controle el `Query Service` el `SVsUIShell` cuadro de diálogo **Guardar** como, llamando a la interfaz. Esto devuelve un `IVsUIShell` puntero a la interfaz. A continuación, el `SaveDocDataToFile` editor debe llamar al método, pasando un puntero a la implementación del `IPersistFileFormat` editor mediante el `pPersistFile` parámetro. A continuación, el entorno realiza la operación Guardar y proporciona el cuadro de diálogo **Guardar** como para el editor. A continuación, el entorno `IPersistFileFormat` vuelve a llamar al editor con .

5. Si el usuario está intentando guardar un documento sin título (es decir, un documento no guardado anteriormente), se realiza realmente un comando Guardar como.
6. Para el comando Guardar como, el entorno muestra el cuadro de diálogo Guardar como, solicitando al usuario un nombre de archivo.

Si el nombre del archivo ha cambiado, la jerarquía es responsable de [SetProperty](#) actualizar la información almacenada en caché del marco de documento mediante una llamada (VSFPROPID_MkDocument).

Si el comando **Guardar como** mueve la ubicación del documento y la jerarquía es sensible a la ubicación del documento, la jerarquía es responsable de entregar la propiedad de la ventana de documento abierta a otra jerarquía. Por ejemplo, esto ocurre si el proyecto realiza un seguimiento de si el archivo es un archivo interno o externo (archivo varios) en relación con el proyecto. Use el siguiente procedimiento para cambiar la propiedad de un archivo al proyecto Archivos varios.

Cambio de la propiedad del archivo

Para cambiar la propiedad del archivo al proyecto Archivos varios

1. Servicio de [SVsExternalFilesManager](#) consulta para la interfaz.

Se devuelve [IVsExternalFilesManager2](#) un puntero a.

2. Llame [TransferDocument](#) al `pszMkDocumentNew` `punkWindowFrame` método (,) para transferir el documento a la nueva jerarquía. La jerarquía que realiza el comando Guardar como llama a este método.

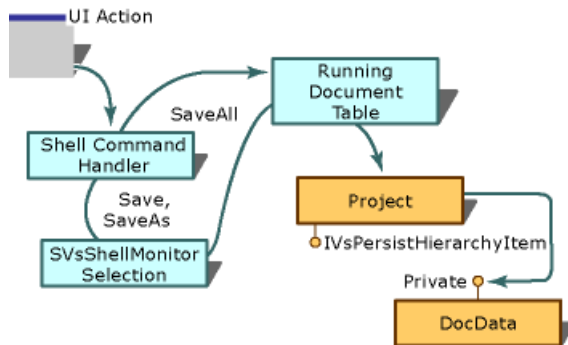
Vea también

- [IOleCommandTarget](#)
- [Apertura y guardado de elementos de proyecto](#)

Guardado de un documento personalizado

06/04/2020 • 3 minutes to read • [Edit Online](#)

El entorno controla los comandos **Guardar como**, **Guardar como** y **Guardar todo**. Cuando un usuario hace clic en **Guardar**, **Guardar como** o **Guardar todo** en el menú **Archivo** o cierra la solución, lo que da como resultado Guardar todo, se produce el siguiente proceso.



Guardar, Guardar como y Guardar todo el control de comandos para un editor personalizado

Este proceso se detalla en los pasos siguientes:

1. Para los comandos **Guardar** y Guardar **SVsShellMonitorSelection** como, el entorno utiliza el servicio para determinar la ventana de documento activa y, por lo tanto, qué elementos se deben guardar. Una vez que se conoce la ventana de documento activa, el entorno busca el puntero de jerarquía y el identificador de elemento (itemID) para el documento en la tabla de documentos en ejecución. Para obtener más información, consulte Ejecución de tabla de [documentos](#).

Para el comando Guardar todo, el entorno utiliza la información de la tabla de documentos en ejecución para compilar la lista de todos los elementos que se va a guardar.

2. Cuando la solución **QueryStatus** recibe una llamada, recorre en iteración el conjunto de elementos seleccionados (es decir, las múltiples selecciones expuestas por el **SVsShellMonitorSelection** servicio).
3. En cada elemento de la selección, la solución utiliza el puntero de jerarquía para llamar al **IsItemDirty** método para determinar si se debe habilitar el comando de menú Guardar. Si uno o más elementos están sucios, el comando Guardar está habilitado. Si la jerarquía utiliza un editor estándar, la jerarquía delega la **IsDocDataDirty** consulta de estado sucio en el editor llamando al método.
4. En cada elemento seleccionado que está sucio, la **Saveltem** solución usa el puntero de jerarquía para llamar al método en las jerarquías adecuadas.

En el caso de un editor personalizado, la comunicación entre el objeto de datos del documento y el proyecto es privada. Por lo tanto, cualquier preocupación de persistencia especial se controla entre estos dos objetos.

NOTE

Si implementa su propia persistencia, asegúrese de llamar al **QuerySaveFiles** método para ahorrar tiempo. Este método comprueba que es seguro guardar el archivo (por ejemplo, el archivo no es de solo lectura).

Vea también

- [IOleCommandTarget](#)
- [Apertura y guardado de elementos de proyecto](#)

Determinar qué editor abre un archivo en un proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Cuando un usuario abre un archivo en un proyecto, el entorno pasa por un proceso de sondeo, abriendo finalmente el editor o diseñador adecuado para ese archivo. El procedimiento inicial empleado por el entorno es el mismo para los editores estándar y personalizados. El entorno utiliza una variedad de criterios al sondear qué editor usar para abrir un archivo y el VSPackage debe coordinar con el entorno durante este proceso.

Por ejemplo, cuando un usuario selecciona el comando **Abrir** en el menú **Archivo** y, a continuación, elige *filename.rtf* (o cualquier otro archivo con una extensión *.rtf*), el entorno llama a la [IsDocumentInProject](#) implementación de cada proyecto, y finalmente recorre todas las instancias de proyecto de la solución. Los proyectos devuelven un conjunto de indicadores que identifican las notificaciones en un documento por prioridad. Con la prioridad más alta, [OpenItem](#) el entorno llama al método adecuado. Para obtener más información sobre el proceso de sondeo, vea [Agregar plantillas de elemento de proyecto y proyecto](#).

El proyecto Archivos varios reclama todos los archivos que no son reclamados por otros proyectos. De esta manera, los editores personalizados pueden abrir documentos antes de que los editores estándar los abran. Si un proyecto de archivos varios reclama [OpenStandardEditor](#) un archivo, el entorno llama al método para abrir el archivo con un editor estándar. El entorno comprueba su lista interna de editores registrados en busca de uno que controle los archivos *.rtf*. Esta lista se encuentra en el registro en la siguiente clave:

HKEY_LOCAL_MACHINE, Software, Microsoft, \<VisualStudio versión> editor \<editor de la fábrica guía> -Extensiones

El entorno también comprueba los identificadores de clase de la clave **HKEY_CLASSES_ROOT-CLSID** para los objetos que tienen una subclave **DocObject**. Si la extensión de archivo se encuentra allí, una versión incrustada de la aplicación, como Microsoft Word, se crea in situ en Visual Studio. Estos objetos de documento deben [IPersistStorage](#) ser archivos compuestos [IPersistFileFormat](#) que implementan la interfaz, o el objeto debe implementar la interfaz.

Si no hay ningún generador de editores para los archivos *.rtf* en el registro, el entorno busca en la **HKEY_CLASSES_ROOT** clave *.rtf* y abre el editor especificado allí. Si la extensión de archivo no se encuentra en **HKEY_CLASSES_ROOT**, el entorno usa el editor de texto principal de Visual Studio para abrir el archivo, si se trata de un archivo de texto.

Si se produce un error en el editor de texto principal, que se produce si el archivo no es un archivo de texto, el entorno utiliza su editor binario para el archivo.

Si el entorno encuentra un editor para la extensión *.rtf* en su registro, carga el VSPackage que implementa este generador de editores. El entorno [SetSite](#) llama al método en el nuevo VSPackage. El VSPackage `QueryService` `SID_SVsRegisterEditor` llama a [RegisterEditor](#), mediante el método para registrar el generador del editor con el entorno.

El entorno ahora vuelve a comprobar su lista interna de editores registrados para encontrar el generador de editores recién registrado para los archivos *.rtf*. El entorno llama a [CreateEditorInstance](#) la implementación del método, pasando el nombre de archivo y el tipo de vista que se desea crear.

Vea también

- [IPersistFileFormat](#)

- [IPersistStorage](#)
- [SetSite](#)
- [IsDocumentInProject](#)
- [OpenItem](#)
- [OpenStandardEditor](#)

Administración de opciones de configuración

06/04/2020 • 2 minutes to read • [Edit Online](#)

Al crear un nuevo tipo de proyecto, debe administrar la configuración del proyecto y la solución que determinan cómo se compilará, empaquetará, implementará y ejecutará el proyecto. En los temas siguientes se describe la configuración del proyecto y la solución.

En esta sección

- [Información general](#)

Describe cómo los Visual Studio proyectos en pueden admitir varias configuraciones.

- [Páginas de propiedades](#)

Explica que los usuarios pueden ver y cambiar las propiedades dependientes de la configuración del proyecto y las propiedades independientes mediante el uso de páginas de propiedades.

- [Configuración de soluciones](#)

Proporciona información sobre lo que se almacena en las configuraciones de la solución y cómo las configuraciones de la solución dirigen el comportamiento de los comandos **Inicio** y **Compilación**.

- [Objeto de configuración del proyecto](#)

Explica cómo el objeto de configuración del proyecto administra la visualización de la información de configuración a la interfaz de usuario.

- [Configuración del proyecto para la compilación](#)

Explica cómo se administra una lista de configuraciones de solución para una solución determinada mediante el cuadro de diálogo Configuraciones de la **solución**.

- [Configuración del proyecto para administrar la implementación](#)

Define el acto de implementación y las dos formas Visual Studio admiten proyectos que admiten la implementación.

- [Configuración del proyecto para la salida](#)

Explica los procesos de compilación que cada configuración puede admitir y las interfaces y métodos por los que los elementos de salida pueden estar disponibles.

Secciones relacionadas

- [Tipos de proyecto](#)

Proporciona una visión general de los Visual Studio proyectos como los bloques básicos del entorno de desarrollo integrado (IDE). Se proporcionan vínculos a temas adicionales que explican cómo los proyectos controlan la creación y compilación de código.

Visión general de las opciones de configuración

06/04/2020 • 9 minutes to read • [Edit Online](#)

Los Visual Studio proyectos en pueden admitir varias configuraciones que se pueden compilar, depurar, ejecutar o implementar. Una configuración es un tipo de compilación descrito con un conjunto con nombre de propiedades, normalmente modificadores del compilador y ubicaciones de archivos. De forma predeterminada, las nuevas soluciones contienen dos configuraciones, *Depurar* y *Liberar*. Estas configuraciones se pueden aplicar utilizando su configuración predeterminada o modificarse para satisfacer los requisitos específicos de la solución o del proyecto. Algunos paquetes se pueden crear de dos maneras: como un editor ActiveX o como un componente in situ. Sin embargo, los proyectos no necesitan admitir varias configuraciones. Si solo hay una configuración disponible, esa configuración se asigna a todas las configuraciones de la solución.

Las configuraciones suelen constar de dos partes: el nombre de configuración (como *Depurar* o *Liberar*) y la configuración de la plataforma. El nombre de plataforma de una configuración identifica el entorno al que se dirige la configuración, como un conjunto de API o una plataforma de sistema operativo. Los Visual Studio usuarios de no pueden crear una plataforma; deben elegir entre las selecciones que permite un proyecto vsPackage. Cuando un usuario instala un VSPackage, la plataforma de entrega creada durante el desarrollo del paquete puede mostrar cualquier nombre de plataforma deseado en función de los criterios establecidos por el creador del paquete. A continuación, el usuario puede seleccionar de la lista de plataformas disponibles a través de VSPackage cuando se crean instancias de las páginas de propiedades.

Los nombres de plataforma son opcionales, ya que no todos los proyectos admiten el concepto de plataformas. Cuando una configuración carece de un nombre de plataforma, la cadena **N/A** se muestra en la interfaz de usuario.

Cada solución tiene su propio conjunto de configuraciones, solo una de las cuales puede estar activa a la vez. Una configuración de solución es un conjunto de no más de una configuración de cada proyecto. La estipulación "no más que" se debe a la opción de excluir un proyecto de una configuración de solución. Los usuarios pueden crear sus propias configuraciones de solución personalizadas.

En la tabla siguiente se muestra la configuración típica de un proyecto. Las filas se etiquetan con nombres de configuración y las columnas con nombres de plataforma.

NOMBRE DE LA CONFIGURACIÓN	PLATAFORMA: WIN32	PLATAFORMA: WIN64
<i>Depurar</i>	<Configuración de depuración de Win32>	<Depurar configuración de Win64>
<i>Release</i>	<Sulte la configuración de Win32>	<Sulte la configuración de Win64>
<i>MyConfig</i>	N/D	<Configuración de MyConfig Win64>

NOTE

No puede crear una configuración de solución *MyConfig* que excluya una plataforma Win32 a menos que el proyecto al que se dirige no admita Win32.

Al cambiar la configuración activa de una solución, se selecciona el conjunto de configuraciones de proyecto que se compila, ejecuta, depura o implementa en esa solución. Por ejemplo, si cambia la configuración de la solución activa de *Release* a *Debug*, todos los proyectos dentro de esa solución se compilan automáticamente con la configuración de los proyectos indicada en la configuración de depuración de la solución. Las configuraciones de

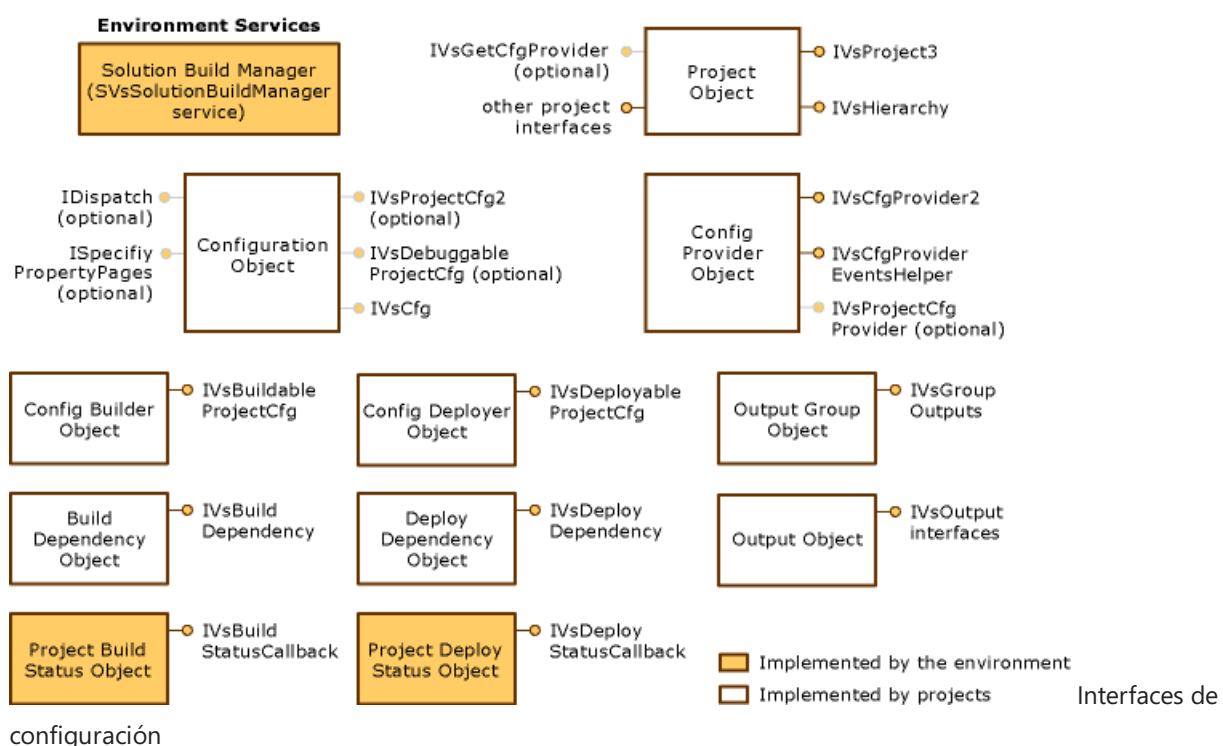
los proyectos también se denominan *Depurar* a menos que el usuario haya realizado cambios manuales en Configuration Manager del entorno.

Las propiedades de configuración de la solución almacenadas para cada proyecto incluyen el nombre del proyecto, el nombre de configuración del proyecto, las marcas para indicar si se va a compilar o implementar y el nombre de la plataforma. Para obtener más información, consulte Configuración de la [solución](#).

El usuario puede ver y establecer los parámetros de configuración de la solución seleccionando la solución en la jerarquía (Explorador de soluciones) y abriendo las páginas de propiedades. De forma similar, puede ver y establecer parámetros de configuración del proyecto seleccionando un proyecto en el Explorador de soluciones y abriendo las páginas de propiedades de ese proyecto.

El usuario también puede crear un proyecto mediante la configuración de versión y todo lo demás con los valores de configuración de depuración si es necesario. Para obtener más información, consulte [Configuración del proyecto para la creación](#).

En el diagrama siguiente se muestra cómo se implementan las interfaces que admiten las configuraciones de soluciones y proyectos:



Algunas notas relacionadas con el diagrama anterior:

- `IDispatch` se marca como opcional en el objeto de configuración. Específicamente, es opcional tener las interfaces de configuración en el objeto de exploración.
- `IVsDebuggableProjectCfg` se marca como opcional en el objeto de configuración, pero es necesario para la compatibilidad con la depuración.
- `IVsProjectCfg2` se marca como opcional en el objeto de configuración, pero es necesario para la compatibilidad con la agrupación de salida.
- El objeto Config Provider se marca como un objeto opcional, pero la opción es dónde implementarlo. Puede implementar el objeto en el objeto de proyecto o en un objeto independiente.
- `IVsCfgProvider2` es necesario para el soporte de la plataforma y la edición de la configuración. `IVsCfgProvider` es suficiente si no implementa esa funcionalidad.
- Algunos de estos objetos que se muestran en el diagrama como objetos independientes se pueden combinar en la misma clase cuando sea práctico en función de sus requisitos de diseño específicos. En otros

temas de esta sección, sin embargo, los objetos e interfaces asociados con esos objetos se tratarán según el escenario presentado en el diagrama.

- Algunos objetos se implementan por separado. Por ejemplo, la creación de proyectos y soluciones se producen en subprocesos independientes y el objeto para administrar la compilación vive por separado del objeto que describe la configuración de la compilación.

Para obtener más información sobre las interfaces de objeto de configuración y las interfaces de objeto de proveedor de configuración en el diagrama anterior, vea Objeto de [configuración de proyecto](#). Además, la configuración de [Project para la creación](#) proporciona más información sobre el generador de configuración y las interfaces de objetos de dependencia de compilación, y la configuración del proyecto para administrar la [implementación](#) describe más detalladamente las interfaces asociadas al implementador de configuración y a los objetos de dependencia de implementación. Por último, Configuración del [proyecto para la salida](#) describe el grupo de salida y las interfaces de objeto de salida, y el uso de páginas de propiedades para ver y establecer propiedades dependientes de la configuración.

Vea también

- [IVsCfgProvider2](#)
- [Configuración del proyecto para la construcción](#)
- [Configuración de la solución](#)

Páginas de propiedades

06/04/2020 • 10 minutes to read • [Edit Online](#)

Los usuarios pueden ver y cambiar las propiedades independientes y dependientes de la configuración del proyecto mediante páginas de propiedades. Un botón **Páginas** de propiedades está habilitado en la ventana **Propiedades** o en la barra de herramientas del Explorador de soluciones para los objetos que proporcionan una vista de página de propiedades del objeto seleccionado. Las páginas de propiedades son creadas por el entorno y están disponibles para soluciones y proyectos. Sin embargo, también pueden estar disponibles para los elementos de proyecto que hacen uso de propiedades dependientes de la configuración. Esta funcionalidad se puede usar cuando los archivos de un proyecto requieren una configuración de conmutador del compilador diferente para compilar correctamente.

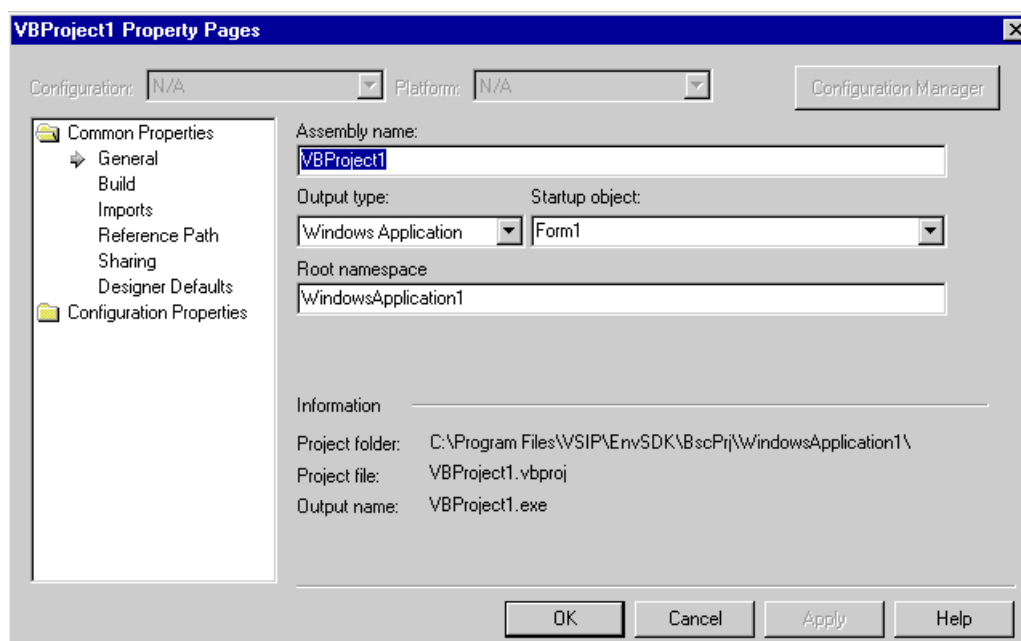
Uso de páginas de propiedades

Si ya se muestra una página de propiedades y la selección cambia (por ejemplo, de una solución a un proyecto), la información que se muestra en las páginas cambia para mostrar las propiedades de la nueva selección. Si no hay propiedades en el objeto que admiten páginas de propiedades, la página de propiedades está vacía.

Si se seleccionan varios objetos, la página de propiedades muestra la intersección de las propiedades de todos los elementos seleccionados. Si el elemento seleccionado no contiene propiedades dependientes de la configuración y se hace clic en el botón **Páginas** de propiedades de la barra de herramientas del Explorador de soluciones, el foco cambia a la ventana Propiedades. Para obtener más información sobre la ventana Propiedades y la selección, consulte [Ampliación de propiedades](#).

Si se muestran propiedades para varios objetos y se cambia un valor en una página de propiedades, todos los valores de los objetos se establecen en el nuevo valor, incluso si inicialmente eran diferentes y la página estaba en blanco cuando se mostraban las propiedades de un objeto individual.

Hay dos tipos generales de cuadros Visual Studio de diálogo **Páginas** de propiedades de proyecto disponibles en . En el primero, para proyectos de Visual Basic, por ejemplo, las páginas de propiedades se muestran con un formato de campo, como se muestra en la siguiente captura de pantalla. En la segunda, que se muestra más adelante en esta sección, la página de propiedades hospeda una cuadrícula de propiedades similar a la que se encuentra en la ventana Propiedades.



Cuadro de diálogo

La estructura de árbol del cuadro de [IVsHierarchy](#) diálogo Páginas de propiedades no se crea mediante . El entorno, basado en el nombre [ISpecifyPropertyPages](#) de [IVsPropertyPage](#) nivel que le pasan las interfaces y las, lo compila.

Solo hay dos categorías de Visual Studio nivel superior disponibles en las páginas de propiedades:

- Propiedades comunes, que muestra información independiente de la configuración para el objeto u objetos seleccionados. Como resultado, cuando se selecciona una de las subcategorías Propiedades comunes, las opciones Configuración, Plataforma y Administrador de configuración en la parte superior del cuadro de diálogo no están disponibles.
- Propiedades de configuración, que contiene información dependiente de la configuración relacionada con los parámetros de depuración, optimización y compilación para la solución o el proyecto.

No puede crear ninguna categoría de nivel superior adicional, pero puede optar por [IVsPropertyPage](#) no mostrar una u otra en la implementación de . Si, por ejemplo, no tiene ninguna propiedad independiente de la configuración para mostrar para un objeto, puede optar por no mostrar la categoría Propiedades comunes. Mostrar propiedades comunes [ISpecifyPropertyPages](#) si se implementa desde el objeto de exploración del elemento y propiedades de configuración cuando se implementa [ISpecifyPropertyPages](#) en el objeto de configuración (el objeto que implementa [IVsCfg](#) , [IVsProjectCfg](#) , e interfaces relacionadas).

Cada categoría que se muestra en una categoría de nivel superior representa una página de propiedades independiente. Las entradas de categoría y subcategoría disponibles en [ISpecifyPropertyPages](#) [IVsPropertyPage](#) el cuadro de diálogo se determinan mediante la implementación de y .

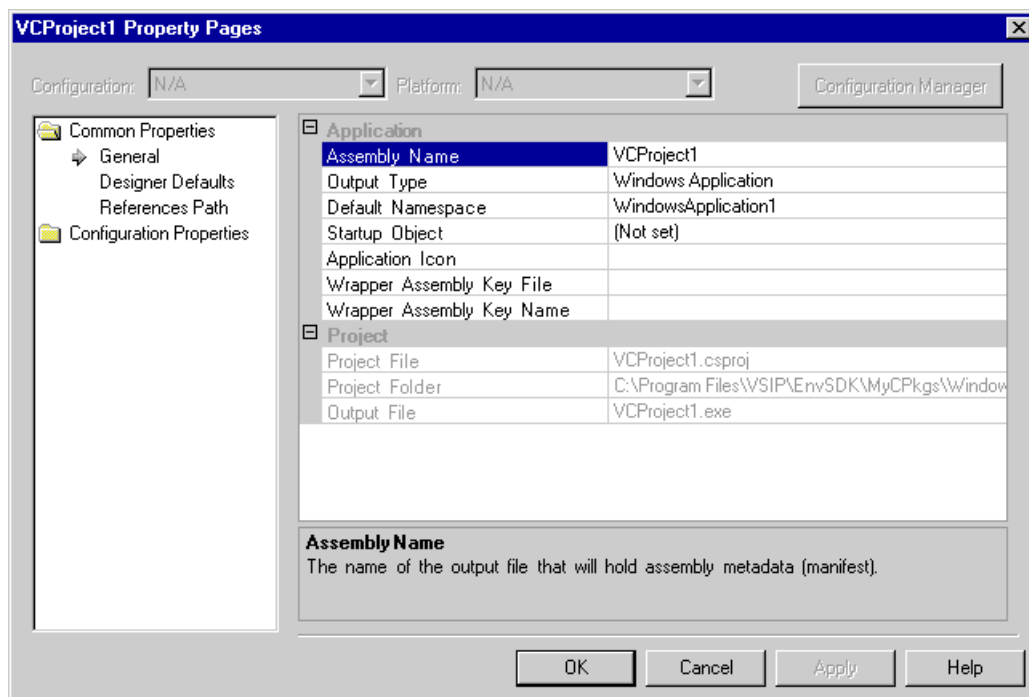
[IDispatch](#) Los objetos de los elementos del contenedor de [ISpecifyPropertyPages](#) selección que tienen propiedades que se mostrarán en las páginas de propiedades se implementan para enumerar una lista de ID de clase. Los ID de clase se pasan [ISpecifyPropertyPages](#) como variables y se usan para crear instancias de las páginas de propiedades. La lista de IDs de [IVsPropertyPage](#) clase también se pasa para crear la estructura de árbol a la izquierda del cuadro de diálogo. A continuación, las páginas [IDispatch](#) de propiedades [ISpecifyPropertyPages](#) devuelven información al objeto que implementa y rellena la información de cada página.

Las propiedades del objeto de [IDispatch](#) exploración se recuperan mediante para cada objeto del contenedor de selección.

Implementar [Help::DisplayTopicFromF1Keyword](#) en el [VSPackage](#) proporciona la funcionalidad para el botón Ayuda.

Para obtener más [IDispatch](#) [ISpecifyPropertyPages](#) información, consulte y en la biblioteca MSDN.

El segundo tipo de páginas de propiedades que se muestran en los ejemplos hospeda una forma de la cuadrícula de propiedades, como se muestra en la siguiente captura de pantalla.



Cuadro de

diálogo Páginas de propiedades con cuadrícula de propiedades

Las interfaces `IVSMDPropertyGrid` y (declaradas `IVSMDPropertyBrowser` en `vsmanaged.h`) se utilizan para crear y rellenar la cuadrícula de propiedades dentro de un cuadro de diálogo o ventana.

La arquitectura de los proyectos ha Visual Studiocambiado considerablemente con respecto a versiones anteriores de . En particular, la noción de qué proyecto está activo ha cambiado. En Visual Studio, no hay ningún concepto de un proyecto activo. En entornos de desarrollo anteriores, el proyecto activo era el proyecto que compilar e implementar comandos sería de forma predeterminada independientemente del contexto. Ahora, la solución controla y arbitra qué comandos de compilación e implementación se aplican a qué proyectos.

Lo que antes era un proyecto activo ahora se captura de una de tres maneras diferentes:

- El proyecto Startup

Puede especificar un proyecto o proyectos desde la página de propiedades de la solución que se iniciará cuando el usuario presione F5 o seleccione Ejecutar en el menú Generar. Esto funciona de una manera similar al proyecto activo anterior en el sentido de que su nombre se muestra en el Explorador de soluciones con fuente en negrita.

Puede recuperar el proyecto de inicio como una `DTE.Solution.SolutionBuild.StartupProjects` propiedad en el modelo de automatización llamando a . En un VSPackage, `get_StartupProject` se `get_StartupProject` llama a los métodos o. `IVsSolutionBuildManager` está disponible como `QueryService` un servicio por `SID_SVsSolutionBuildManager`. Para obtener más información, vea Objeto de configuración de [proyecto](#) y Configuración de la [solución](#).

- Configuración de compilación de soluciones activas

Visual Studiatiene una configuración de solución activa, `DTE.Solution.SolutionBuild.ActiveConfiguration` disponible en el modelo de automatización mediante la implementación de . Una configuración de solución es una colección que contiene una configuración de proyecto para cada proyecto de la solución (cada proyecto puede tener varias configuraciones, en varias plataformas, con nombres diferentes). Para obtener más información relacionada con las páginas de propiedades de la solución, vea Configuración de la [solución](#).

- Proyecto actualmente seleccionado

Implemente `GetCurrentSelection` el método para recuperar la jerarquía del proyecto y el elemento de

proyecto o los elementos seleccionados. Desde DTE, usaría `SelectedItems.SelectedItem.Project`
`SelectedItems.SelectedItem.ProjectItem` los métodos y. Hay código de ejemplo en esos Visual Studio encabezados en los documentos principales.

Vea también

- [IVsPropertyPage](#)
- [Administración de opciones de configuración](#)
- [Objeto de configuración del proyecto](#)
- [Configuración de soluciones](#)

Configuración de soluciones

06/04/2020 • 5 minutes to read • [Edit Online](#)

Las configuraciones de solución almacenan propiedades de nivel de solución. Dirigen el comportamiento de la tecla **Inicio** (F5) y los comandos **Decompilar**. De forma predeterminada, estos comandos compilan e inician la configuración de depuración. Ambos comandos se ejecutan en el contexto de una configuración de solución. Esto significa que el usuario puede esperar que F5 inicie y compile cualquier solución activa que se configure a través de la configuración. El entorno está diseñado para optimizar las soluciones en lugar de los proyectos cuando se trata de construir y ejecutar.

La barra de herramientas estándar de Visual Studio contiene un botón Inicio y una lista desplegable de configuración de la solución a la derecha del botón Inicio. Esta lista permite a los usuarios elegir la configuración que se iniciará cuando se presiona F5, crear sus propias configuraciones de solución o editar una configuración existente.

NOTE

No hay interfaces de extensibilidad para crear o editar las configuraciones de la solución. Se debe usar `DTE.SolutionBuild`. Sin embargo, hay API de extensibilidad para administrar la compilación de la solución. Para obtener más información, vea [IVsSolutionBuildManager2](#).

A continuación se muestra cómo implementar las configuraciones de solución admitidas por el tipo de proyecto:

- proyecto

Muestra los nombres de los proyectos que se encuentran en la solución actual.

- Configuración

Para proporcionar la lista de configuraciones admitidas por el tipo [IVsCfgProvider2](#) de proyecto y que se muestran en las páginas de propiedades, implemente .

La columna Configuración muestra el nombre de la configuración del proyecto que se va a compilar en esta configuración de la solución y enumera todas las configuraciones del proyecto al hacer clic en el botón de flecha. El entorno [GetCfgNames](#) llama al método para rellenar esta lista. Si [GetCfgProviderProperty](#) el método indica que el proyecto admite la edición de configuración, las selecciones Nuevo o Editar también se muestran en el encabezado Configuración. Cada una de estas selecciones inicia `IVsCfgProvider2` cuadros de diálogo que llaman a métodos de la interfaz para editar las configuraciones del proyecto.

Si un proyecto no admite configuraciones, la columna Configuración muestra Ninguno y está deshabilitada.

- Plataforma

Muestra la plataforma para la que se compila la configuración del proyecto seleccionado y enumera todas las plataformas disponibles para el proyecto al hacer clic en el botón de flecha. El entorno [GetPlatformNames](#) llama al método para rellenar esta lista. Si [GetCfgProviderProperty](#) el método indica que el proyecto admite la edición de plataformas, las selecciones Nuevo o Editar también se muestran en el encabezado Plataforma. Cada una de estas selecciones `IVsCfgProvider2` inicia cuadros de diálogo que llaman a métodos para editar las plataformas disponibles del proyecto.

Si un proyecto no admite plataformas, la columna de plataforma de ese proyecto muestra Ninguno y está deshabilitada.

- **Compilar**

Especifica si el proyecto se compila o no mediante la configuración de la solución actual. Los proyectos no seleccionados no se compilan cuando se invocan los comandos de compilación de nivel de solución a pesar de las dependencias de proyecto que contienen. Los proyectos no seleccionados para compilarse todavía se incluyen en la depuración, ejecución, empaquetado e implementación de la solución.

- **Implementación**

Especifica si el proyecto se implementará o no cuando se usen los comandos Inicio o Implementación con la configuración de compilación de la solución seleccionada. La casilla de verificación de este campo estará disponible [IVsDeployableProjectCfg](#) si [IVsProjectCfg2](#) el proyecto admite la implementación mediante la implementación de la interfaz en su objeto.

Una vez que se agrega una nueva configuración de solución, el usuario puede seleccionarla en el cuadro de lista desplegable Configuración de la solución de la barra de herramientas estándar para compilar o iniciar esa configuración.

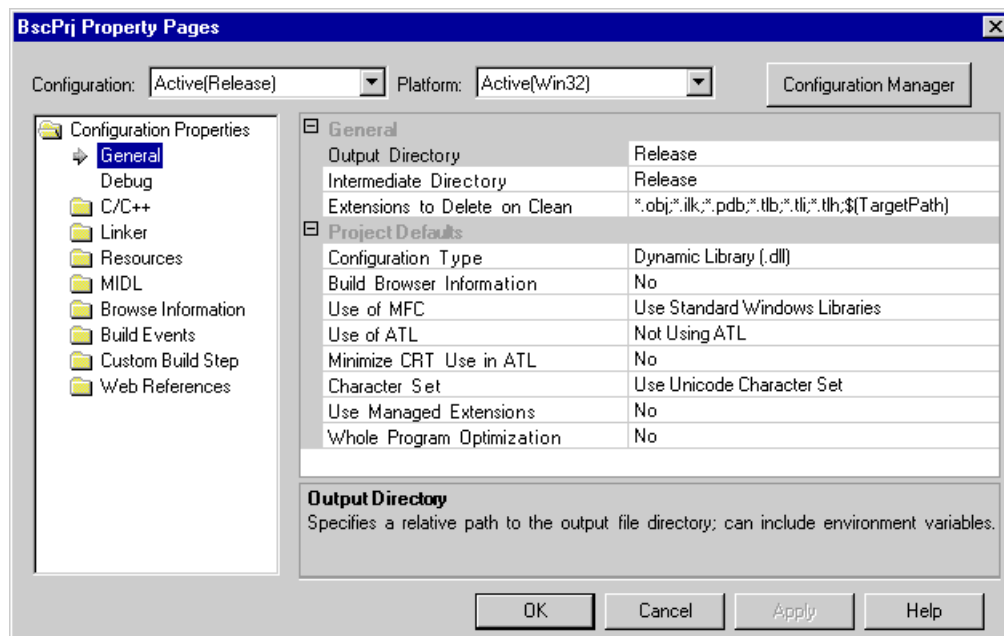
Vea también

- [Administración de opciones de configuración](#)
- [Configuración del proyecto para la compilación](#)
- [Objeto de configuración del proyecto](#)

Objeto de configuración del proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

El objeto de configuración del proyecto administra la visualización de la información de configuración en la interfaz de usuario.



de Visual Studio

Páginas de propiedades de configuración del proyecto

El proveedor de configuración del proyecto administra las configuraciones del proyecto. El entorno y otros paquetes, para obtener acceso y recuperar información sobre las configuraciones de un proyecto, llame a las interfaces adjuntas al objeto Proveedor de configuración del proyecto.

NOTE

No puede crear ni editar archivos de configuración de solución mediante programación. Se debe usar `DTE.SolutionBuilder`. Consulte Configuración de la [solución](#) para obtener más información.

Para publicar un nombre para mostrar que se usará `get_DisplayName` en la interfaz de usuario de configuración, el proyecto debe implementar `ISolutionCfg`. El entorno `GetCfgs` llama a `ISolutionCfg` que devuelve una lista de punteros que puede usar para obtener los nombres para mostrar de la información de configuración y plataforma que se mostrará en la interfaz de usuario del entorno. La configuración activa y la plataforma están determinadas por la configuración del proyecto almacenada en la configuración de la solución activa. El `FindActiveProjectCfg` método se puede utilizar para recuperar la configuración del proyecto activo.

El `IVsProjectCfgProvider` objeto se puede implementar `IVsCfgProvider2` opcionalmente `IVsCfgProviderEventsHelper` en el objeto con `IVsProjectCfg2` el objeto para permitirle recuperar un objeto basado en el nombre de configuración del proyecto canónico.

Otra forma de proporcionar el entorno y otros proyectos con acceso a `IVsCfgProvider2::GetCfgs` las configuraciones de proyecto es que los proyectos proporcionen una implementación del método para devolver uno o varios objetos de configuración. Los proyectos también `IVsProjectCfg2` pueden implementar, `IVsProjectCfg` que hereda de y, por lo no, de `IVsCfg`, para proporcionar información específica de la configuración. `IVsCfgProvider2` admite plataformas y funcionalidad para agregar, eliminar y cambiar el nombre de las configuraciones de proyecto.

NOTE

Dado que Visual Studio ya no está limitado a dos tipos de configuración, el código que procesa las configuraciones no debe escribirse con suposiciones sobre el número de configuraciones, ni debe escribirse con la suposición de que un proyecto que solo tiene una configuración es necesariamente Debug o Retail. Esto hace que [get_IsReleaseOnly](#) [get_IsDebugOnly](#) el uso y obsoleto.

Llamar `QueryInterface` al objeto `IVsGetCfgProvider::GetCfgProvider` devuelto `IVsCfgProvider2` desde `retrieve` . Si `IVsGetCfgProvider` no se `QueryInterface` encuentra `IVsProject3` llamando al objeto de proyecto, puede `QueryInterface` tener acceso al objeto de `IVsHierarchy::GetProperty(VSITEM_ROOT, VSHPROPID_BrowseObject)` proveedor de configuración llamando al objeto `IVsHierarchy::GetProperty(VSITEM_ROOT, VSHPROPID_ConfigurationProvider)` de explorador raíz de jerarquía para el objeto devuelto para , o a través de un puntero al proveedor de configuración devuelto para .

`IVsProjectCfg2` principalmente proporciona acceso a objetos de administración de compilación, depuración e implementación y permite a los proyectos la libertad de agrupar las salidas. Los `IVsProjectCfg` métodos `IVsProjectCfg2` y se [IVsBuildableProjectCfg](#) pueden usar para implementar [IVsOutputGroup](#) para administrar el proceso de compilación y punteros para los grupos de salida de una configuración.

El proyecto debe devolver el mismo número de grupos para cada configuración que admite aunque el número de salidas contenidas en un grupo puede variar de una configuración a otra. Los grupos también deben tener la misma información de identificador (nombre canónico, nombre para mostrar e información de grupo) desde la configuración hasta la configuración dentro de un proyecto. Para obtener más información, consulte [Configuración del proyecto para salida](#).

Para habilitar la depuración, [IVsDebuggableProjectCfg](#) las configuraciones deben implementar .

`IVsDebuggableProjectCfg` es una interfaz opcional implementada por los proyectos para permitir que el `IVsCfg` depurador `IVsProjectCfg` inicie una configuración y se implementa en el objeto de configuración con y . El entorno lo llama cuando el usuario elige iniciar el depurador presionando F5.

`ISpecifyPropertyPages` y `IDispatch` se utilizan junto con las páginas de propiedades para recuperar y mostrar información dependiente de la configuración al usuario. Para obtener más información, vea [Páginas de propiedades](#) .

Vea también

- [Administración de opciones de configuración](#)
- [Configuración del proyecto para la compilación](#)
- [Configuración del proyecto para la salida](#)
- [Páginas de propiedades](#)
- [Configuración de soluciones](#)

Configuración del proyecto para la compilación

06/04/2020 • 6 minutes to read • [Edit Online](#)

La lista de configuraciones de solución para una solución determinada se administra mediante el cuadro de diálogo Configuraciones de solución.

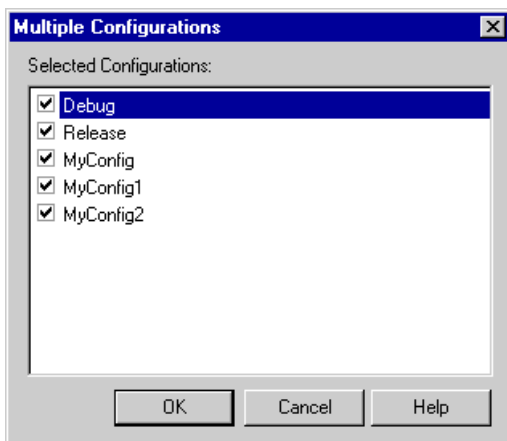
Un usuario puede crear configuraciones de solución adicionales, cada una con su propio nombre único. Cuando el usuario crea una nueva configuración de solución, el IDE tiene como valor predeterminado el nombre de configuración correspondiente en los proyectos o Depurar si no existe ningún nombre correspondiente. El usuario puede cambiar la selección para cumplir requisitos específicos si es necesario. La única excepción a este comportamiento es cuando el proyecto admite una configuración que coincide con el nombre de la nueva configuración de la solución. Por ejemplo, supongamos que una solución contiene Project1 y Project2. Project1 tiene configuraciones de proyecto Debug, Retail y MyConfig1. Project2 tiene configuraciones de proyecto Debug, Retail y MyConfig2.

Si el usuario crea una nueva configuración de solución denominada MyConfig2, Project1 enlaza su configuración de depuración a la configuración de la solución de forma predeterminada. Project2 también enlaza su configuración MyConfig2 a la configuración de la solución de forma predeterminada.

NOTE

El enlace no distingue mayúsculas de minúsculas.

Cuando el usuario selecciona el elemento **Selección múltiple** en la lista desplegable de configuración, el entorno muestra un cuadro de diálogo que proporciona la lista de configuraciones disponibles.

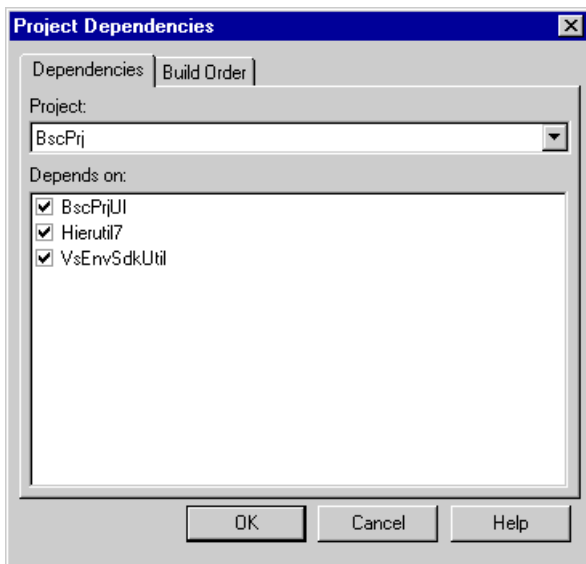


Múltiples configuraciones

Dentro de este cuadro de diálogo, el usuario puede seleccionar una o varias configuraciones. Una vez seleccionado, los valores de propiedad que se muestran en el cuadro de diálogo de páginas de propiedades reflejan la intersección de valores para las configuraciones seleccionadas.

Consulte Configuración de la [solución](#) para obtener información relacionada con la adición y el cambio de nombre de las configuraciones de soluciones y proyectos.

Las dependencias de proyecto y el orden de compilación son independientes de la configuración de la solución: es decir, solo puede configurar un árbol de dependencias para todos los proyectos de la solución. Al hacer clic con el botón derecho en la solución o proyecto y seleccionar la opción **Dependencias** del proyecto o Orden de **compilación** de proyecto, se abre el cuadro de diálogo Dependencias del **proyecto**. También se puede abrir desde el menú **Proyecto**.



Dependencias del proyecto

Las dependencias del proyecto determinan el orden en que se compilan los proyectos. Use la pestaña Orden de compilación del cuadro de diálogo para ver el orden exacto en el que se compilarán los proyectos de una solución y use la pestaña Dependencias para modificar el orden de compilación.

NOTE

Los proyectos de la lista que tienen sus casillas de verificación seleccionadas pero [IVsBuildDependency](#) aparecen [IVsDeployDependency](#) atenuadas se han agregado por el entorno debido a las dependencias explícitas especificadas por las interfaces o y no se pueden cambiar. Por ejemplo, al agregar Visual Basic una referencia de proyecto de un proyecto a otro proyecto, se agrega automáticamente una dependencia de compilación que solo se puede quitar eliminando la referencia. Los proyectos cuyas casillas de verificación están desactivadas y aparecen atenuadas no se pueden seleccionar porque al hacerlo se crearía un bucle de dependencia (por ejemplo, Project1 dependería de Project2 y Project2 dependería de Project1), lo que detendría la compilación.

Visual Studio los procesos de compilación incluyen las operaciones de compilación y vínculo típicas que se invocan con un único comando Build. También se pueden admitir otros dos procesos de compilación: una operación limpia para eliminar todos los elementos de salida de una compilación anterior y una comprobación actualizada para determinar si un elemento de salida de una configuración ha cambiado.

- [IVsProjectCfg2](#) objetos devuelven un correspondiente [IVsBuildableProjectCfg](#) (devuelta desde [get_CfgType](#)) para administrar sus procesos de compilación. Para informar del estado de una operación de compilación [IVsBuildStatusCallback](#) mientras se está produciendo, las configuraciones realizan llamadas a , una interfaz implementada por el entorno y cualquier otro objeto interesado en eventos de estado de compilación.

Una vez compiladas, los valores de configuración se pueden usar para determinar si se pueden ejecutar o no bajo el control del depurador. Las configuraciones se implementan [IVsDebuggableProjectCfg](#) para admitir la depuración.

Después de implementar las dependencias del proyecto, puede manipular mediante programación las dependencias a través del modelo de automatización. Llame [BuildDependencies](#) al modelo de automatización. No hay interfaces de nivel de API de VSIP disponibles que permitan la manipulación directa de las configuraciones del administrador de compilación de soluciones y sus propiedades.

Además, puede proporcionar una cuadrícula en la ventana de dependencias del proyecto. Para obtener más información, consulte Cuadrícula de [visualización de propiedades](#).

Vea también

- [Administración de opciones de configuración](#)

- Configuración del proyecto para administrar la implementación
- Configuración del proyecto para la salida

Configuración del proyecto para administrar la implementación

06/04/2020 • 3 minutes to read • [Edit Online](#)

La implementación es el acto de mover físicamente los elementos de salida de un proceso de compilación a la ubicación esperada para la depuración y la instalación. Por ejemplo, una aplicación web puede compilarse en un equipo local y, a continuación, colocarse en el servidor.

Visual Studio admite dos formas en que los proyectos pueden participar en la implementación:

- Como el tema del proceso de implementación.
- Como administrador del proceso de implementación.

Antes de implementar soluciones, primero debe agregar un proyecto de implementación para configurar las opciones de implementación. Si el proyecto de implementación aún no existe, se le preguntará si desea crear uno al seleccionar **Implementar solución** en el menú **Compilar** o hacer clic con el botón secundario en la solución. Al hacer clic en **Sí**, se abre el cuadro de diálogo **Agregar nuevo proyecto** con el proyecto Asistente para **implementación remota** seleccionado.

El Asistente para implementación remota le pide el tipo de aplicación (Windows o Web), los grupos de salida del proyecto que se van a incluir, los archivos adicionales que desea incluir y el equipo remoto en el que desea implementar. La última página del asistente muestra un resumen de las opciones seleccionadas.

Los proyectos que son objeto de un proceso de implementación producen elementos de salida que se deben mover a un entorno alternativo. Estos elementos de salida [IVsProjectCfg2](#) se describen como parámetros para la interfaz, cuyo propósito principal es permitir que los proyectos agrupen las salidas. Para obtener más información sobre `IVsProjectCfg2` la implementación de, vea Configuración del [proyecto para salida](#).

Los proyectos de implementación, que administran el proceso de implementación, habilitan el comando Deploy y responden cuando se selecciona este comando. Los proyectos [IVsDeployableProjectCfg](#) de implementación implementan la interfaz para realizar la implementación y realizar llamadas a la [IVsDeployStatusCallback](#) interfaz para informar de los eventos de estado de implementación.

Las configuraciones pueden especificar dependencias que afectan a sus operaciones de compilación o implementación. Las dependencias de compilación o implementación son proyectos que deben compilarse o implementarse antes o después de que se compilen o implementen las propias configuraciones. Las dependencias de compilación [IVsBuildDependency](#) entre proyectos se describen con la interfaz e implementan dependencias con la [IVsDeployDependency](#) interfaz. Para obtener más información, consulte [Configuración del proyecto para compilar](#).

Vea también

- [Administración de opciones de configuración](#)
- [Configuración del proyecto para la compilación](#)
- [Configuración del proyecto para la salida](#)

Configuración del proyecto para la salida

06/04/2020 • 5 minutes to read • [Edit Online](#)

Cada configuración puede admitir un conjunto de procesos de compilación que producen elementos de salida, como archivos ejecutables o de recursos. Estos elementos de salida son privados para el usuario y se pueden colocar en grupos que vinculan tipos de salida relacionados, como archivos ejecutables (.exe, .dll, .lib) y archivos de origen (archivos .idl, .h).

Los elementos de salida [IVsOutput2](#) pueden estar disponibles a través de los métodos y enumerarlos con los [IVsEnumOutputs](#) métodos. Cuando desee agrupar elementos de salida, [IVsOutputGroup](#) el proyecto también debe implementar la interfaz.

La construcción desarrollada `IVsOutputGroup` por la implementación permite a los proyectos agrupar las salidas según el uso. Por ejemplo, un archivo DLL podría agruparse con su base de datos de programas (PDB).

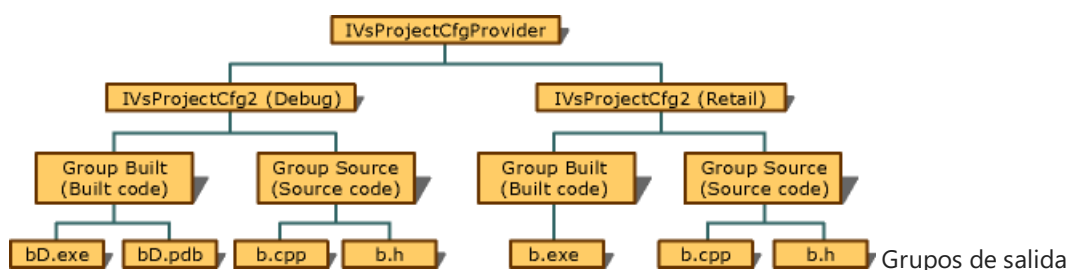
NOTE

Un archivo PDB contiene información de depuración y se crea cuando se especifica la opción 'Generar información de depuración' al compilar el archivo .dll o .exe. El archivo .pdb se genera normalmente solo para la configuración del proyecto de depuración.

El proyecto debe devolver el mismo número de grupos para cada configuración que admite, aunque el número de salidas contenidas en un grupo puede variar de una configuración a otra. Por ejemplo, el archivo DLL del proyecto Matt puede incluir matt.dll y matt.pdb en la configuración de depuración, pero solo incluye matt.dll en la configuración de Retail.

Los grupos también tienen la misma información de identificador, como el nombre canónico, el nombre para mostrar y la información de grupo, desde la configuración hasta la configuración dentro de un proyecto. Esta coherencia permite que la implementación y el empaquetado sigan funcionando incluso si cambian las configuraciones.

Los grupos también pueden tener una salida clave que permite que los accesos directos de empaquetado apunten a algo significativo. Cualquier grupo puede estar vacío en una configuración determinada, por lo que no se debe hacer ninguna suposición sobre el tamaño de un grupo. El tamaño (número de salidas) de cada grupo en cualquier configuración puede ser diferente del tamaño de otro grupo en la misma configuración. También puede ser diferente del tamaño del mismo grupo en otra configuración.



El uso principal [IVsProjectCfg](#) de la interfaz es proporcionar acceso para compilar, implementar y depurar objetos de administración y permitir a los proyectos la libertad de agrupar salidas. Para obtener más información sobre el uso de esta interfaz, vea Objeto de configuración de [proyecto](#).

En el diagrama anterior, Group Built tiene una salida clave entre configuraciones (bD.exe o b.exe) para que el usuario pueda crear un acceso directo a Built y saber que el acceso directo funcionará independientemente de la configuración implementada. Origen de grupo no tiene una salida de clave, por lo que el usuario no puede crear

un acceso directo a él. Si el grupo de depuración compilado tiene una salida clave, pero el grupo comercial compilado no, sería una implementación incorrecta. Se deduce, entonces, que si cualquier configuración tiene un grupo que no contiene salidas y, como resultado, ningún archivo de clave, otras configuraciones con ese grupo que contienen salidas no pueden tener archivos de clave. Los editores del instalador asumen que los nombres canónicos y los nombres para mostrar de los grupos, además de la existencia de un archivo de clave, no cambian en función de las configuraciones.

Tenga en cuenta que `IVsOutputGroup` si un proyecto tiene un que no desea empaquetar o implementar, es suficiente no colocar esa salida en un grupo. La salida todavía se puede enumerar `EnumOutputs` normalmente mediante la implementación del método que devuelve todas las salidas de una configuración independientemente de la agrupación.

Para obtener más información, `IVsOutputGroup` vea la implementación en el ejemplo De proyecto personalizado en [MPF para proyectos](#).

Vea también

- [Administración de opciones de configuración](#)
- [Configuración del proyecto para la compilación](#)
- [Objeto de configuración del proyecto](#)
- [Configuración de soluciones](#)

Compatibilidad con control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Visual Studio admite desprotecciones de archivos, protecciones y otras operaciones de control de código fuente para su proyecto o editor. Como cliente de Visual Studio control de código fuente, está diseñado Visual SourceSafe para interactuar con un paquete de control de código fuente, como , que proporciona instalaciones de archivado, control de versiones y control para un conjunto de archivos definido dinámicamente.

En esta sección

- [Modelo para paquetes de control de código fuente](#)

Describe las interfaces que un tipo de proyecto debe implementar para admitir el control de código fuente.

- [Decisiones de diseño](#)

Proporciona preguntas cuyas respuestas cambian la forma de implementar un tipo de proyecto.

- [Detalles de configuración](#)

Describe cómo la compatibilidad con el control de código fuente cambia la implementación de un tipo de proyecto.

- [Instrucciones adicionales para proyectos y editores](#)

Describe las prácticas recomendadas para los tipos de proyecto y los editores.

- [Detalles del entorno de ejecución](#)

Describe cómo registrar un proyecto cuando un usuario lo agrega a un sistema de control de código fuente.

Referencia

[IVsQueryEditQuerySave2](#) Indica al entorno o al paquete de control de código fuente que un archivo está a punto de cambiarse en la memoria o guardarse.

[IVsSccManager2](#) Permite que los proyectos y jerarquías se registren con el control de código fuente y obtengan información sobre el estado del control de código fuente.

[IVsSccProject2](#) Se implementa en un sistema de proyecto para proporcionar control de código fuente para los archivos de proyecto y los elementos de proyecto.

[IVsTrackProjectDocuments2](#) Utilizado por los proyectos para consultar el entorno para obtener permiso para agregar, quitar o cambiar el nombre de un archivo o directorio en una solución.

[IVsTrackProjectDocumentsEvents2](#) Notifica a los clientes de los cambios realizados en archivos de proyecto o directorios.

Secciones relacionadas

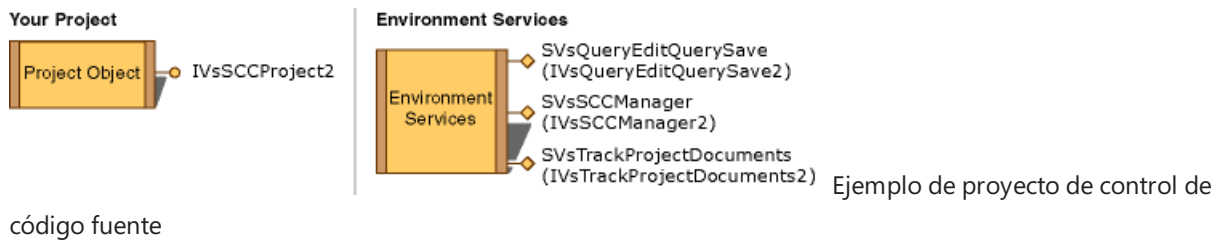
- [Tipos de proyecto](#)

Proporciona una visión general de los Visual Studio proyectos como los bloques básicos del entorno de desarrollo integrado (IDE). Se proporcionan vínculos a temas adicionales que explican cómo los proyectos controlan la creación y compilación de código.

Modelo para paquetes de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

El modelo siguiente representa un ejemplo de una implementación de control de código fuente. En el modelo, verá las interfaces que debe implementar y los servicios de entorno a los que debe llamar. Al igual que todos los servicios, en realidad se llaman los métodos de una interfaz determinada que se obtiene n.º del servicio. Los nombres de las clases se identifican para facilitar la ver cómo se lleva a cabo el control de código fuente.



Interfaces

Puede implementar el control de código fuente para los nuevos tipos de proyecto en Visual Studio mediante la lista de interfaces que se muestra en la tabla siguiente.

INTERFAZ	USO
IVsQueryEditQuerySave2	Llamado por proyectos y editores antes de guardar o cambiar archivos (sucios). Se accede a SVsQueryEditQuerySave esta interfaz mediante el servicio.
IVsTrackProjectDocuments2	Llamado por los proyectos para solicitar permiso para agregar, quitar o cambiar el nombre de un archivo o directorio. Los proyectos también llaman a esta interfaz para informar al entorno cuando se completa una acción de agregar, quitar o cambiar el nombre aprobada. Se accede mediante SVsTrackProjectDocuments el servicio.
IVsTrackProjectDocumentsEvents2	Implementado por cualquier entidad que se registre para recibir una notificación cuando los proyectos agreguen, renombre o quiten un archivo o directorio. Para registrarse para AdviseTrackProjectDocumentsEvents la notificación de eventos, llame a .
IVsScCManager2	Llamado por los proyectos para registrarse con el paquete de control de código fuente y para obtener información sobre el estado del control de código fuente. Se accede a SVsScCManager esta interfaz mediante el servicio.
IVsScCProject2	Implementado por el proyecto para responder a las solicitudes de control de código fuente para obtener información sobre los archivos y para obtener la configuración de control de código fuente necesaria para el archivo de proyecto.

Vea también

- [IVsQueryEditQuerySave2](#)

- [IVsSccManager2](#)
- [IVsSccProject2](#)
- [IVsTrackProjectDocuments2](#)
- [AdviseTrackProjectDocumentsEvents](#)
- [IVsTrackProjectDocumentsEvents2](#)
- [Compatibilidad con control de código fuente](#)

Decisiones de diseño del control de código fuente

06/04/2020 • 3 minutes to read • [Edit Online](#)

Las siguientes decisiones de diseño deben tenerse en cuenta para los proyectos al implementar el control de código fuente.

¿La información será compartida o privada?

La decisión de diseño más importante que puede tomar es qué información se puede retar y qué es privada. Por ejemplo, la lista de archivos para el proyecto se comparte, pero dentro de esta lista de archivos, es posible que algunos usuarios deseen tener archivos privados. La configuración del compilador se comparte, pero el proyecto de inicio es generalmente privado. La configuración es puramente compartida, compartida con una invalidación o puramente privada. Por diseño, los elementos privados, como los archivos de Visual SourceSafe opciones de usuario de solución (.suo), no se protegen en . Asegúrese de almacenar cualquier información privada en archivos privados, como el archivo .suo, o un archivo privado específico que cree, por ejemplo, un archivo .csproj.user para Visual C. o un archivo .vbproj.user para Visual Basic.

Esta decisión no es todo incluido y se puede tomar artículo por artículo.

¿El proyecto incluirá archivos especiales?

Otra decisión de diseño importante es si la estructura del proyecto utiliza archivos especiales. Los archivos especiales son archivos ocultos que subyacen a los archivos visibles en el Explorador de soluciones y en los cuadros de diálogo de protección y desprotección. Si utiliza archivos especiales, siga estas pautas:

1. No asocie archivos especiales con el nodo raíz del proyecto, es decir, con el propio archivo de proyecto. El archivo de proyecto debe ser un único archivo.
2. Cuando se agregan, quitan o se cambia el [IVsTrackProjectDocumentsEvents2](#) nombre de archivos especiales en un proyecto, los eventos adecuados deben desencadenarse con el conjunto de indicadores que indica que los archivos son archivos especiales. El entorno llama a estos eventos en respuesta [IVsTrackProjectDocuments2](#) al proyecto que llama a los métodos adecuados.
3. Cuando el proyecto [QueryEditFiles](#) o el editor llama a un archivo, los archivos especiales asociados a ese archivo no se desprotegen automáticamente. Pase los archivos especiales junto con el archivo principal. El entorno detectará la relación entre todos los archivos que se pasan y ocultará adecuadamente los archivos especiales en la interfaz de usuario de desprotección.

Vea también

- [QueryEditFiles](#)
- [IVsTrackProjectDocumentsEvents2](#)
- [IVsTrackProjectDocuments2](#)
- [Compatibilidad con control de código fuente](#)

Detalles de configuración del control de código fuente

06/04/2020 • 6 minutes to read • [Edit Online](#)

Para implementar el control de código fuente, debe configurar correctamente el sistema de proyectos o el editor para hacer lo siguiente:

- Solicitar permiso para la transición al estado cambiado
- Solicitar permiso para guardar un archivo
- Solicitar permiso para agregar, quitar o cambiar el nombre de los archivos en el proyecto

Solicitar permiso para la transición al estado cambiado

Un proyecto o editor debe solicitar permiso para pasar `IVsQueryEditQuerySave2` al estado modificado (sucio) llamando a `IsDocDataDirty`. Cada editor que implementa `QueryEditFiles` debe llamar y recibir aprobación para `True` `IsDocDataDirty` cambiar el documento del entorno antes de devolver para `True`. Un proyecto es esencialmente un editor para un archivo de proyecto y, como resultado, tiene la misma responsabilidad de implementar el seguimiento de estado modificado para el archivo de proyecto que un editor de texto para sus archivos. El entorno controla el estado modificado de la solución, pero debe controlar el estado modificado de cualquier objeto al que hace referencia la solución, pero no almacena, como un archivo de proyecto o sus elementos. En general, si el proyecto o editor es responsable de administrar la persistencia de un elemento, es responsable de implementar el seguimiento de estado modificado.

En respuesta `IVsQueryEditQuerySave2::QueryEditFiles` a la llamada, el entorno puede hacer lo siguiente:

- Rechazar la llamada a cambiar, en cuyo caso el editor o proyecto debe permanecer en el estado sin cambios (limpio).
- Indique que se deben volver a cargar los datos del documento. Para un proyecto, el entorno volverá a cargar los datos del proyecto. Un editor debe volver a `ReloadDocData` cargar los datos desde el disco a través de su implementación. En cualquier caso, el contexto del proyecto o del editor puede cambiar cuando se vuelven a cargar los datos.

Es una tarea compleja y difícil `IVsQueryEditQuerySave2::QueryEditFiles` reacondicionar las llamadas adecuadas a una base de código existente. Como resultado, estas llamadas deben integrarse durante la creación del proyecto o editor.

Solicitar permiso para guardar un archivo

Antes de que un proyecto o editor `QuerySaveFile` `QuerySaveFiles` guarde un archivo, debe llamar a `SaveDocDataToFile`. Para los archivos de proyecto, la solución completa automáticamente estas llamadas, que sabe cuándo guardar un archivo de proyecto. Los editores son responsables de realizar `IVsPersistDocData2` estas llamadas a menos que la implementación del editor de use la función `SaveDocDataToFile` auxiliar. Si el editor `IVsPersistDocData2` se implementa de esta `IVsQueryEditQuerySave2::QuerySaveFile` `IVsQueryEditQuerySave2::QuerySaveFiles` manera, la llamada o se realiza para usted.

NOTE

Realice siempre estas llamadas de forma preventiva, es decir, en un momento en que el editor puede recibir una cancelación.

Solicitar permiso para agregar, quitar o cambiar el nombre de los archivos en el proyecto

Para que un proyecto pueda agregar, cambiar el nombre o `IVsTrackProjectDocuments2::OnQuery*` quitar un archivo o directorio, debe llamar al método adecuado para solicitar permiso del entorno. Si se concede permiso, el proyecto debe completar la `IVsTrackProjectDocuments2::OnAfter*` operación y, a continuación, llamar al método adecuado para notificar al entorno que la operación se ha completado. El proyecto debe llamar [IVsTrackProjectDocuments2](#) a los métodos de la interfaz para todos los archivos (por ejemplo, archivos especiales) y no solo para los archivos primarios. Las llamadas a archivos son obligatorias, pero las llamadas de directorio son opcionales. Si el proyecto tiene información de directorio, debe llamar a los métodos adecuados, [IVsTrackProjectDocuments2](#) pero si no tiene esta información, el entorno deducirá la información del directorio.

El proyecto no debe `IVsTrackProjectDocuments2` llamar a los métodos de `at project open` o `close`. Los agentes de escucha que desean [OnAfterOpenSolution](#) esta información en el inicio pueden esperar el evento e iterar a través de la solución para encontrar la información que necesitan. Al apagar, esta información no es necesaria.

`IVsTrackProjectDocuments2` se proporciona [SVsTrackProjectDocuments](#) desde el .

Para cada acción de agregar, cambiar `OnQuery*` el nombre `OnAfter*` y quitar, hay un método y un método. Llame `OnQuery*` al método para solicitar permiso para agregar, cambiar el nombre o quitar el archivo o directorio. Llame `OnAfter*` al método después de que el archivo o directorio se haya agregado, cambiado de nombre o quitado y el estado del proyecto refleje el nuevo estado.

Vea también

- [IVsPersistDocData](#)
- [QuerySaveFile](#)
- [QuerySaveFiles](#)
- [IVsTrackProjectDocuments2](#)
- [SaveDocDataToFile](#)
- [SVsTrackProjectDocuments](#)
- [Compatibilidad con control de código fuente](#)

Directrices adicionales de control de código fuente para proyectos y editores

06/04/2020 • 3 minutes to read • [Edit Online](#)

Hay una serie de directrices que los proyectos y editores deben cumplir para admitir el control de código fuente.

Directrices

El proyecto o editor también debe hacer lo siguiente para admitir el control de código fuente:

ÁREA	PROYECTO	EDITOR	DETALLES
Copias privadas de archivos	X		El entorno admite copias privadas de archivos. Es decir, cada persona alistada en el proyecto tiene su propia copia privada de los archivos en ese proyecto.
Persistencia ANSI/Unicode	X	X	Si escribe el código de persistencia, conserve los archivos en el formulario ANSI porque la mayoría de los programas de control de código fuente no admiten Unicode actualmente.
Enumerar archivos	X		El proyecto debe contener una lista específica de todos los archivos que contiene IVsSccProject2.GetProperty y debe ser capaz de enumerar la lista de archivos mediante el o (VSH_PROPID_First_Child/Next_Sibling). El proyecto también debe exponer GetMkDocument los nombres de elemento a través de su implementación y búsqueda de nombres de soporte técnico (incluidos archivos especiales) a través de su IsDocumentInProject implementación.

ÁREA	PROYECTO	EDITOR	DETALLES
Formato de texto	X	X	Cuando sea posible, los archivos deben estar en formato de texto para admitir la fusión de diferentes versiones. Los archivos que no están en formato de texto no se pueden combinar con otras versiones del archivo más adelante. El formato de texto preferido es XML.
Basado en referencias	X		Los proyectos basados en referencias se admiten fácilmente en el control de código fuente. Sin embargo, los proyectos basados en directorios también son compatibles con el control de código fuente siempre que el proyecto pueda generar una lista de sus archivos a petición, independientemente de si esos archivos existen en el disco. Al abrir un proyecto desde el control de código fuente, el archivo de proyecto se derriba primero antes de cualquiera de sus archivos.
Conservar objetos y propiedades en orden predecible	X	X	Persiste tus archivos en un orden predecible, como el orden alfabético, para facilitar la fusión.

ÁREA	PROYECTO	EDITOR	DETALLES
Recargar	X	X	Cuando un archivo cambia en el disco, el editor debe poder volver a cargarlo. Cuando participe en el control de código fuente, ReloadDocData el entorno volverá a cargar los datos por usted llamando a la implementación. El caso de recarga más difícil es cuando se produce una QueryEditFiles desprotección cuando se ha llamado a <code>IVsQueryEditQuerySave::</code> y está procesando información. Sin embargo, el código de recarga debe poder ejecutarse en esta situación. El entorno vuelve a cargar automáticamente los archivos de proyecto. Sin embargo, IVsPersistHierarchyItem2 un proyecto debe implementarse si tiene jerarquías anidadas para admitir la recarga de archivos de proyecto anidados.

Vea también

- [Control de código fuente de soporte](#)

Detalles de tiempo de ejecución del control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un proyecto se agrega al control de código fuente cuando el usuario agrega un archivo en el proyecto al control de código fuente o a través de un controlador de automatización, como un asistente. Un proyecto no especifica por sí mismo que está bajo control de código fuente; admite el control de código fuente, pero debe agregarse a él manualmente.

Registrarse con un paquete de control de código fuente

Cuando se agrega un archivo del proyecto al [SetScclLocation](#) control de código fuente, el entorno llama para proporcionar cuatro cadenas opacas que el sistema de control de código fuente utiliza como cookies. Almacene estas cadenas en el archivo de proyecto. Estas cadenas se deben pasar al Código auxiliar de Control de código fuente (el componente [RegisterScclProject](#) de Visual Studio que administra los paquetes de control de código fuente) al iniciar el tipo de proyecto llamando a `IVsScclManager2::RegisterScclProject`. Esto, a su vez, carga el paquete de

`IVsScclManager2::RegisterScclProject` control de código fuente adecuado y reenvía la llamada a su implementación de `IVsScclManager2::RegisterScclProject`.

Vea también

- [RegisterScclProject](#)
- [SetScclLocation](#)
- [Compatibilidad con control de código fuente](#)

Anidamiento de proyectos

06/04/2020 • 2 minutes to read • [Edit Online](#)

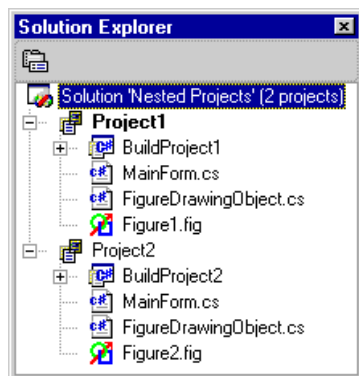
Los desarrolladores de aplicaciones empresariales que usan el paquete Visual Studio DE VS pueden agrupar convenientemente tipos similares de proyectos mediante el anidamiento de *proyectos*. Por ejemplo, el proyecto Plantilla de empresa utiliza proyectos anidados para agrupar proyectos en categorías. Los proyectos de fachada empresarial, los proyectos de interfaz de usuario web, etc., se agrupan en una categoría.

En este escenario, no hay límite en el número de proyectos que el desarrollador puede anidar en cada proyecto primario, aunque el desarrollador puede proporcionar límites mediante programación. Este tipo de agrupación también se puede hacer recursivo, en cuyo caso los proyectos del mismo tipo que un proyecto secundario se pueden anidar bajo el elemento secundario para convertirse en un subproyecto del elemento secundario, que es un subproyecto del elemento primario.

El anidamiento de proyectos Visual Studio no es una parte intrínseca de . Debe escribir el código para habilitar el anidamiento y el anidamiento de subproyectos dentro de proyectos secundarios. El proyecto primario es un VSPackage especial, o tipo de proyecto, creado y registrado con su propio GUID que incluye el código necesario para implementar el anidamiento del proyecto.

Puede encontrar un ejemplo sobre cómo anidar proyectos en [Cómo: Implementar proyectos anidados](#).

Ejemplo de proyectos anidados



Ejemplo de proyectos anidados

Vea también

- [Consideraciones para descargar y volver a cargar proyectos anidados](#)
- [Compatibilidad del asistente con los proyectos anidados](#)
- [Registro de plantillas para proyectos y elementos](#)
- [Implementación del control de comandos para proyectos anidados](#)
- [Filtrado del cuadro de diálogo AddItem para proyectos anidados](#)
- [Lista de comprobación: creación de nuevos tipos de proyecto](#)
- [Parámetros de contexto](#)
- [Archivo de asistente \(.Vsz\)](#)

Cómo: Implementar proyectos anidados

06/04/2020 • 11 minutes to read • [Edit Online](#)

Al crear un tipo de proyecto anidado, hay varios pasos adicionales que se deben implementar. Un proyecto primario asume algunas de las mismas responsabilidades que la solución tiene para sus proyectos anidados (secundarios). El proyecto primario es un contenedor de proyectos similar a una solución. En particular, hay varios eventos que debe generar la solución y los proyectos primarios para compilar la jerarquía de proyectos anidados. Estos eventos se describen en el siguiente proceso para crear proyectos anidados.

Crear proyectos anidados

1. El entorno de desarrollo integrado (IDE) carga el archivo de [IVsProjectFactory](#) proyecto principal y la información de inicio llamando a la interfaz. El proyecto primario se crea y se agrega a la solución.

NOTE

En este punto, es demasiado temprano en el proceso para que el proyecto primario cree el proyecto anidado porque el proyecto primario debe crearse antes de que se puedan crear los proyectos secundarios. Siguiendo esta secuencia, el proyecto primario puede aplicar la configuración a los proyectos secundarios y los proyectos secundarios pueden adquirir información de los proyectos primarios si es necesario. Esta secuencia es si es necesaria para clientes como el control de código fuente (SCC) y el **Explorador** de soluciones .

El proyecto primario debe [OpenChildren](#) esperar a que el IDE llame al método para poder crear su proyecto o proyectos anidados (secundarios).

2. El IDE `QueryInterface` llama al [IVsParentProject](#) proyecto primario para . Si esta llamada se realiza correctamente, el IDE llama al [OpenChildren](#) método del elemento primario para abrir todos los proyectos anidados para el proyecto primario.
3. El proyecto primario [FireOnBeforeOpeningChildren](#) llama al método para notificar a los agentes de escucha que los proyectos anidados están a punto de crearse. SCC, por ejemplo, está escuchando esos eventos para saber si los pasos en la solución y el proceso de creación de proyectos se están produciendo en orden. Si los pasos se producen fuera de servicio, es posible que la solución no se registre correctamente con el control de código fuente.
4. El proyecto [AddVirtualProject](#) primario llama [AddVirtualProjectEx](#) al método o al método en cada uno de sus proyectos secundarios.

Se `_VSADDVPFLAGS` pasa `AddVirtualProject` al método para indicar que el proyecto virtual (anidado) se debe agregar a la ventana del proyecto, se excluye de la compilación, se agrega al control de código fuente, etc. `VSADDVPFLAGS` le permite controlar la visibilidad del proyecto anidado e indicar qué funcionalidad está asociada a él.

Si vuelve a cargar un proyecto secundario existente anteriormente que tiene un GUID `AddVirtualProjectEx` de proyecto almacenado en el archivo de proyecto del proyecto primario, el proyecto primario llama . La única `AddVirtualProject` diferencia `AddVirtualProjectEX` entre `AddVirtualProjectEX` y es que tiene un parámetro `guidProjectID` para permitir que [GetProjectOfGuid](#) el [GetProjectOfProjref](#) proyecto primario especifique una instancia por instancia para que el proyecto secundario habilite y funcione correctamente.

Si no hay ningún GUID disponible, como cuando se agrega un nuevo proyecto anidado, la solución crea uno para el proyecto en el momento en que se agrega al elemento primario. Es responsabilidad del proyecto

primario conservar ese GUID del proyecto en su archivo de proyecto. Si elimina un proyecto anidado, también se puede eliminar el GUID de ese proyecto.

5. El IDE `OpenChildren()` llama al método en cada proyecto secundario del proyecto primario.

El proyecto primario `IVsParentProject` debe implementarse si desea anidar proyectos. Pero el proyecto `QueryInterface` principal `IVsParentProject` nunca requiere incluso si tiene proyectos primarios debajo de él. La solución controla la llamada y, `IVsParentProject` `OpenChildren` si se implementa, llama para crear los proyectos anidados. `AddVirtualProjectEX` siempre se `OpenChildren` llama desde . Nunca debe ser llamado por el proyecto primario para mantener los eventos de creación de jerarquía en orden.

6. El IDE `OnAfterOpenProject` llama al método en el proyecto secundario.
7. El proyecto primario `FireOnAfterOpeningChildren` llama al método para notificar a los agentes de escucha que se han creado los proyectos secundarios para el elemento primario.
8. El IDE `FireOnAfterOpenProject` llama al método en el proyecto primario después de que se han abierto todos los proyectos secundarios.

Si aún no existe, el proyecto primario crea un `CoCreateGuid` GUID para cada proyecto anidado llamando a .

NOTE

`CoCreateGuid` es una API COM a la que se llama cuando se va a crear un GUID. Para obtener más `CoCreateGuid` información, consulte GUID y GUID en MSDN Library.

El proyecto primario almacena este GUID en su archivo de proyecto que se recuperará la próxima vez que se abra en el IDE. Consulte el paso 4 para obtener `AddVirtualProjectEX` más información `guidProjectID` relacionada con la llamada de recuperar el proyecto secundario.

9. A `GetProperty` continuación, se llama al método para el elemento primario `ItemID` que por convención se delegue en el proyecto anidado. Recupera `GetProperty` las propiedades del nodo que anida un proyecto en el que desea delegar cuando se llama en el elemento primario.

Dado que los proyectos primarios y secundarios se crean instancias mediante programación, puede establecer propiedades para proyectos anidados en este momento.

NOTE

No solo recibe la información de contexto del proyecto anidado, sino que también puede preguntar si el proyecto primario tiene algún contexto para ese elemento comprobando `__VSHPROPID.VSHPROPID_UserContext`. De este modo, puede agregar atributos de ayuda dinámicos adicionales y opciones de menú específicas de proyectos anidados individuales.

10. La jerarquía se compila **Solution Explorer** para su presentación `GetNestedHierarchy` en el Explorador de soluciones con una llamada al método.

Pasar la jerarquía al `GetNestedHierarchy` entorno para compilar la jerarquía para su presentación en el Explorador de soluciones. De esta manera, la solución sabe que el proyecto existe y puede ser administrado para compilar por el administrador de compilación, o puede permitir que los archivos del proyecto se pongan bajo control de código fuente.

11. Cuando se han creado todos los proyectos anidados para `Project1`, el control se devuelve a la solución y el proceso se repite para `Project2`.

Este mismo proceso para crear proyectos anidados se produce para un proyecto secundario que tiene un elemento secundario. En este caso, si `BuildProject1`, que es un elemento secundario de `Project1`, tenía

proyectos secundarios, se crearían después de BuildProject1 y antes de Project2. El proceso es recursivo y la jerarquía se crea de arriba hacia abajo.

Cuando se cierra un proyecto anidado porque el usuario cerró la `IVsParentProject` solución o el propio proyecto específico, se llama al otro método en , [CloseChildren](#), . El proyecto primario ajusta [RemoveVirtualProject](#) las llamadas al método con el [OnBeforeClosingChildren](#) y los métodos para notificar a los [OnAfterClosingChildren](#) agentes de escucha a los eventos de solución que los proyectos anidados se están cerrando.

Los temas siguientes tratan con varios otros conceptos a tener en cuenta al implementar proyectos anidados:

- [Consideraciones para descargar y recargar proyectos anidados](#)
- [Soporte de asistente sin asistentes para proyectos anidados](#)
- [Implementar el control de comandos para proyectos anidados](#)
- [Filtrar el cuadro de diálogo AddItem para proyectos anidados](#)

Vea también

- [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#)
- [Registrar plantillas de proyectos y elementos](#)
- [Lista de verificación: Crear nuevos tipos de proyecto](#)
- [Parámetros de contexto](#)
- [Archivo Wizard \(.vsz\)](#)

Consideraciones para descargar y recargar proyectos anidados

06/04/2020 • 2 minutes to read • [Edit Online](#)

Al implementar tipos de proyecto anidados, debe realizar pasos adicionales al descargar y volver a cargar los proyectos. Para notificar correctamente a los agentes de `OnBeforeUnloadProject` `OnAfterLoadProject` escucha a los eventos de la solución, debe generar correctamente los eventos y.

Una razón para generar estos eventos es para el control de código fuente (SCC). No desea que SCC elimine los elementos del servidor y, a continuación, vuelva a agregarlos como *nuevos* si hay una operación `Get` de SCC. En ese caso, un nuevo archivo se cargaría fuera de SCC. Tendrías que descargar y volver a cargar todos los archivos en caso de que sean diferentes.

El control `ReloadItem` de código fuente llama . Implemente `IVsFireSolutionEvents` la interfaz para llamar `OnBeforeUnloadProject` y `OnAfterLoadProject` eliminar el proyecto y volver a crearlo. Al implementar la interfaz de esta manera, se informa a SCC de que el proyecto se eliminó temporalmente y se volvió a agregar. Por lo tanto, SCC no funciona en el proyecto como si el proyecto se eliminara y se hubiera reagregado realmente.

Recargar proyectos

Para admitir la recarga de proyectos `ReloadItem` anidados, implemente el método. En la `ReloadItem` implementación de , se cierran los proyectos anidados y, a continuación, se vuelven a crear.

Normalmente, cuando se vuelve a cargar un `OnBeforeUnloadProject OnAfterLoadProject` proyecto, el IDE genera los eventos y. Sin embargo, para los proyectos anidados que se eliminan y se vuelven a cargar, el proyecto primario inicia el proceso, no el IDE. Dado que el proyecto primario no genera eventos de solución y el IDE no tiene información sobre la inicialización del proceso, no se generan los eventos.

Para controlar este proceso, `QueryInterface` el `IVsFireSolutionEvents` proyecto primario llama a la interfaz. `IVsFireSolutionEvents` tiene funciones que indican al `OnBeforeUnloadProject` IDE que genere el evento `OnAfterLoadProject` para descargar el proyecto anidado y, a continuación, genere el evento para volver a cargar el mismo proyecto.

Vea también

- [IVsSolutionEvents3](#)
- [Proyectos Nest](#)

Compatibilidad del asistente con los proyectos anidados

06/04/2020 • 2 minutes to read • [Edit Online](#)

El IDE ejecuta dos asistentes que el proyecto primario para proyectos anidados puede implementar: el Asistente para **nuevo proyecto** y el Asistente para **agregar elementos**.

Si un usuario inicia el Asistente para **nuevo proyecto** seleccionando **Agregar proyecto** y haciendo clic en **Nuevo proyecto** en el menú Archivo o seleccionando **Agregar** y haciendo clic con el botón secundario en **Nuevo proyecto** en el Explorador de soluciones, el IDE ejecuta el comando **AddProject** y la implementación del proyecto primario del comando **AddProject** devuelve un archivo de proyecto de plantilla o un archivo de asistente (.vsz) que tiene un conjunto de parámetros de contexto.

De forma similar, la implementación de un proyecto primario de asistentes **AddItem** devuelve un archivo .vsz que tiene un conjunto diferente de parámetros de contexto.

Para obtener más información acerca de los asistentes, consulte [Asistente \(.Vsz\) Archivo](#), [Parámetros de contexto](#) y [Registro de plantillas de proyecto y elemento](#).

Vea también

- [IVsUIHierarchy](#)
- [Anidamiento de proyectos](#)

Implementación del control de comandos para proyectos anidados

06/04/2020 • 2 minutes to read • [Edit Online](#)

El IDE puede pasar comandos [IVsUIHierarchy](#) que [IOleCommandTarget](#) se pasan a través de las interfaces y a los proyectos anidados, o los proyectos primarios pueden filtrar o invalidar los comandos.

NOTE

Solo se pueden filtrar los comandos que normalmente controla el proyecto primario. Los comandos como **Build** e **Deploy** controlados por el IDE no se pueden filtrar.

En los pasos siguientes se describe el proceso para implementar el control de comandos.

Procedimientos

Para implementar el control de comandos

1. Cuando el usuario selecciona un proyecto anidado o un nodo en un proyecto anidado:
 - a. El IDE [QueryStatus](#) llama al método.

— o —
 - b. Si el comando se originó en una ventana de jerarquía, como un [QueryStatusCommand](#) comando de menú contextual en el Explorador de soluciones, el IDE llama al método en el elemento primario del proyecto.
2. El proyecto primario puede examinar `QueryStatus` los parámetros que se van a pasar a , como `pguidCmdGroup` y `prgCmds` , para determinar si el proyecto primario debe filtrar los comandos. Si el proyecto primario se implementa para filtrar comandos, debe establecer:

```
prgCmds[0].cmdf = OLECMDF_SUPPORTED;  
// make sure it is disabled  
prgCmds[0].cmdf &= ~MSOCMDF_ENABLED;
```

A continuación, el `S_OK` proyecto primario debe devolver .

Si el proyecto primario no filtra el `S_OK` comando, solo debe devolver . En este caso, el IDE enruta automáticamente el comando al proyecto secundario.

El proyecto primario no tiene que enrutar el comando al proyecto secundario. El IDE realiza esta tarea.

Vea también

- [IVsUIHierarchy](#)
- [Comandos, menús y barras de herramientas](#)
- [Anidamiento de proyectos](#)

Filtrar el cuadro de diálogo AddItem para proyectos anidados

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando se muestra un cuadro de diálogo **AddItem** para un proyecto anidado, el proyecto primario puede controlar qué elementos se muestran en el cuadro de diálogo.

La `IVsFilterAddProjectItemDlg2` interfaz le permite filtrar los nodos que estarán en un Cuadro de diálogo **AddItem**. Cuando el proyecto secundario muestra el Cuadro de `IVsFilterAddProjectItemDlg` diálogo **AddItem**, el elemento primario puede implementar la interfaz y filtrar los elementos que, de lo contrario, se mostrarían en el proyecto del elemento secundario.

Cuando los proyectos se agrupan por función `IVsFilterAddProjectItemDlg` en proyectos primarios específicos, puede implementar cuando el usuario selecciona **Agregar elemento** de proyecto en el menú contextual de un proyecto anidado. Implementar `IVsFilterAddProjectItemDlg displays` solo elementos de proyecto o archivos específicos de ese grupo. Los elementos de proyecto para otros grupos se filtran fuera del cuadro de diálogo, incluso si se almacenan en el mismo directorio.

Cuando un usuario abre el Cuadro de diálogo **AddItem** para `IVsFilterAddProjectItemDlg` el elemento secundario, se llama a la implementación de la interfaz del proyecto primario.

La `IVsFilterAddProjectItemDlg` interfaz también puede implementar el filtrado por categoría. Para obtener más información, vea [Agregar elementos al cuadro de diálogo Agregar nuevo elemento y Registrar plantillas de proyecto y elemento](#).

Vea también

- [IVsFilterAddProjectItemDlg2](#)
- [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#)
- [Registrar plantillas de proyectos y elementos](#)
- [Proyectos Nest](#)

Actualización de proyectos

06/04/2020 • 23 minutes to read • [Edit Online](#)

Los cambios en el modelo de proyecto de una versión de Visual Studio a la siguiente pueden requerir que los proyectos y soluciones se actualicen para que se puedan ejecutar en la versión más reciente. Proporciona Visual Studio SDK interfaces que se pueden usar para implementar la compatibilidad con la actualización en sus propios proyectos.

Estrategias de actualización

Para admitir una actualización, la implementación del sistema de proyectos debe definir e implementar una estrategia de actualización. Al determinar su estrategia, puede optar por admitir copias de seguridad en paralelo (SxS), copia de seguridad o ambas.

- Copia de seguridad SxS significa que un proyecto copia sólo los archivos que necesitan actualizar en su lugar, añadiendo un sufijo de nombre de archivo adecuado, por ejemplo, ".old".
- Copiar copia de seguridad significa que un proyecto copia todos los elementos del proyecto en una ubicación de copia de seguridad proporcionada por el usuario. A continuación, se actualizan los archivos relevantes en la ubicación del proyecto original.

Cómo funciona la actualización

Cuando se abre una solución creada en una versión anterior de Visual Studio en una versión más reciente, el IDE comprueba el archivo de solución para determinar si debe actualizarse. Si se requiere la actualización, el **Asistente para actualización** se inicia automáticamente para guiar al usuario a través del proceso de actualización.

Cuando una solución necesita actualizarse, consulta a cada generador de proyectos su estrategia de actualización. La estrategia determina si el generador de proyectos admite copia de seguridad de copia o copia de seguridad SxS. La información se envía al **Asistente para actualización**, que recopila la información necesaria para la copia de seguridad y presenta las opciones aplicables al usuario.

Soluciones multiproyecto

Si una solución contiene varios proyectos y las estrategias de actualización difieren, como cuando un proyecto de C++ que solo admite la copia de seguridad de SxS y un proyecto web que solo admite copia de seguridad de copia, los generadores de proyectos deben negociar la estrategia de actualización.

La solución consulta cada [IVsProjectUpgradeViaFactory](#) generador de proyectos para . A continuación, llama [UpgradeProject_CheckOnly](#) para ver si los archivos de proyecto globales necesitan actualización y para determinar las estrategias de actualización admitidas. A continuación, se invoca el **Asistente para actualización**.

Una vez que el [UpgradeProject](#) usuario completa el asistente, se llama en cada generador de proyectos para realizar la actualización real. Para facilitar la copia de seguridad, [SVsUpgradeLogger](#) [IVsProjectUpgradeViaFactory](#) métodos proporcionan el servicio para registrar los detalles del proceso de actualización. Este servicio no se puede almacenar en caché.

Después de actualizar todos los archivos globales relevantes, cada fábrica de proyectos puede elegir crear instancias de un proyecto. La implementación [IVsProjectUpgraded](#) del proyecto debe admitir . A [UpgradeProject](#) continuación, se llama al método para actualizar todos los elementos de proyecto relevantes.

NOTE

El [UpgradeProject](#) método no proporciona el servicio [SVsUpgradeLogger](#). Este servicio se puede [QueryService](#) obtener llamando a .

Prácticas recomendadas

Utilice [SVsQueryEditQuerySave](#) el servicio para comprobar si puede editar un archivo antes de editarlo y puede guardarlo antes de guardarlo. Esto ayudará a las implementaciones de copia de seguridad y actualización a controlar los archivos de proyecto bajo control de código fuente, archivos con permisos insuficientes, etc.

Utilice [SVsUpgradeLogger](#) el servicio durante todas las fases de copia de seguridad y actualización para proporcionar información sobre el éxito o el error del proceso de actualización.

Para obtener más información acerca de la copia de seguridad y actualización de proyectos, vea los comentarios de [IVsProjectUpgrade](#) en [vsshell2.idl](#).

Actualización de proyectos personalizados

Si cambia la información guardada en el archivo del proyecto entre diferentes versiones de Visual Studio de su producto, deberá admitir la actualización del archivo del proyecto de la versión anterior a la nueva. Para admitir la actualización que le permite participar en [IVsProjectUpgradeViaFactory](#) el Asistente para conversión de Visual Studio, implemente la interfaz. Esta interfaz contiene el único mecanismo disponible para la actualización de la copia. La actualización del proyecto se produce como parte de la apertura de la solución. El generador de proyectos implementa la interfaz [IVsProjectUpgradeViaFactory](#) o esta debe poderse obtener como mínimo desde el generador de proyectos.

El mecanismo anterior que usa la interfaz [IVsProjectUpgrade](#) sigue siendo compatible, pero conceptualmente actualiza el sistema del proyecto como parte de la apertura de la solución. Por [IVsProjectUpgrade](#) lo tanto, el entorno de [IVsProjectUpgradeViaFactory](#) Visual Studio llama a la interfaz incluso si se llama o implementa la interfaz. Este enfoque le permite usar [IVsProjectUpgradeViaFactory](#) para implementar la copia y solo proyectar partes de la actualización, y delegar el resto del trabajo para que lo realice la interfaz [IVsProjectUpgrade](#) localmente (posiblemente en la nueva ubicación).

Para obtener una [IVsProjectUpgrade](#) implementación de ejemplo de , vea [Ejemplos de VSSDK](#).

Con las actualizaciones del proyecto, se producen los escenarios siguientes:

- Si el archivo tiene un formato más reciente de los que puede admitir el proyecto, debe devolver un error en que se indique esto. Esto supone que la versión anterior del producto incluye código para comprobar la versión.
- Si se especifica la marca [PUVFF_SXSBACKUP](#) en el método [UpgradeProject](#), la actualización se implementará como una actualización local antes de la apertura del proyecto.
- Si se especifica la marca [PUVFF_COPYBACKUP](#) en el método [UpgradeProject](#), la actualización se implementa como una actualización de copia.
- Si se especifica la marca [UPF_SILENTMIGRATE](#) en la llamada [UpgradeProject](#), significa que el entorno ha solicitado al usuario que actualice el archivo del proyecto como una actualización local, una vez abierto el proyecto. Por ejemplo, el entorno pide al usuario que realice la actualización cuando este abra una versión anterior de la solución.
- Si no se especifica la marca [UPF_SILENTMIGRATE](#) en la llamada [UpgradeProject](#), significa que debe solicitar al usuario que actualice el archivo del proyecto.

A continuación, se muestra un ejemplo del mensaje de solicitud de actualización:

"El proyecto '%1' se creó con una versión anterior de Visual Studio. Si lo abre con esta versión de Visual Studio, es posible que no pueda abrirlo con versiones anteriores de Visual Studio. ¿Quiere continuar y abrir este proyecto?"

Para implementar IVsProjectUpgradeViaFactory

1. Implemente el método de la interfaz [IVsProjectUpgradeViaFactory](#), específicamente el método [UpgradeProject](#) de su implementación del generador de proyectos, o realice implementaciones que puedan llamarse desde su implementación del generador de proyectos.
2. Si quiere realizar una actualización local como parte de la apertura de la solución, proporcione la marca [PUVFF_SXSBACKUP](#) como el parámetro `VSPUVF_FLAGS` de su implementación [UpgradeProject](#).
3. Si quiere realizar una actualización local como parte de la apertura de la solución, proporcione la marca [PUVFF_COPYBACKUP](#) como el parámetro `VSPUVF_FLAGS` de su implementación [UpgradeProject](#).
4. En relación con los pasos 2 y 3, los pasos de actualización del archivo real mediante [IVsQueryEditQuerySave2](#) se pueden implementar según se describe en la sección "Implementación de `IVsProjectUpgrade`", o bien puede delegar la actualización del archivo real a [IVsProjectUpgrade](#).
5. Use los métodos de [IVsUpgradeLogger](#) para publicar mensajes relacionados con la actualización para el usuario que usa el Asistente para migración de Visual Studio.
6. La interfaz [IVsFileUpgrade](#) se utiliza para implementar cualquier tipo de actualización de archivo que se debe ejecutar como parte de la actualización del proyecto. Esta interfaz no se llama desde [IVsProjectUpgradeViaFactory](#), sino que se proporciona como un mecanismo para actualizar los archivos que forman parte del sistema del proyecto, pero que puede que el sistema del proyecto principal no tenga en cuenta directamente. Por ejemplo, esta situación podría producirse si el equipo de desarrollo que administra el resto del sistema del proyecto no administra las propiedades y los archivos relacionados con el compilador.

Implementación de IVsProjectUpgrade

Si [IVsProjectUpgrade](#) el sistema del proyecto solo implementa, no puede participar en el **Asistente para conversión** de Visual Studio. Sin embargo, aunque implemente la interfaz [IVsProjectUpgradeViaFactory](#), puede agregar de todos modos la actualización del archivo a la implementación [IVsProjectUpgrade](#).

Para implementar IVsProjectUpgrade

1. Cuando un usuario intenta abrir un proyecto, el entorno llama al método [UpgradeProject](#) después de abrir el proyecto y antes de que se pueda realizar ninguna otra acción de usuario en el proyecto. Si ya se había solicitado al usuario que actualizara la solución, la marca [UPF_SILENTMIGRATE](#) se pasa al parámetro `grfUpgradeFlags`. Si el usuario abre un proyecto directamente, por ejemplo, [UPF_SILENTMIGRATE](#) mediante el comando Agregar proyecto **existente**, no se pasa la marca y el proyecto debe solicitar al usuario que actualice.
2. En respuesta a la llamada [UpgradeProject](#), el proyecto debe evaluar si se actualiza el archivo del proyecto. Si el proyecto no necesita actualizar el tipo de proyecto a una nueva versión, simplemente puede devolver la marca [S_OK](#).
3. Si el proyecto necesita actualizar el tipo de proyecto a una nueva versión, debe determinar si el archivo del proyecto se puede modificar mediante una llamada al método [QueryEditFiles](#) y pasando un valor de [tagVSQueryEditFlags](#) para el parámetro `rgfQueryEdit`. A continuación, el proyecto debe hacer lo siguiente:
 - Si el valor `VSQueryEditResult` devuelto en el parámetro `pfEditCanceled` es [QER_EditOK](#), la actualización puede continuar porque se puede escribir en el archivo del proyecto.
 - Si el valor `VSQueryEditResult` devuelto en el parámetro `pfEditCanceled` es [QER_EditNotOK](#) y el valor

`VSQueryEditResult` tiene el bit `QER_ReadOnlyNotUnderSc` establecido, `UpgradeProject` debe devolver un error, ya que los usuarios deben resolver el problema de permisos. A continuación, el proyecto debe hacer lo siguiente:

Informe del error al `ReportErrorInfo` usuario llamando `VS_E_PROJECTMIGRATIONFAILED` y `IVsProjectUpgrade` devuelva el código de error a .

- Si el valor `VSQueryEditResult` es `QER_EditNotOK` y el valor `VSQueryEditResultFlags` tiene el bit `QER_ReadOnlyUnderSc` establecido, el archivo del proyecto se debería extraer del repositorio llamando a `QueryEditFiles` (`QEF_ForceEdit_NoPrompting`, `QEF_DisallowInMemoryEdits`,...).

4. Si la llamada `QueryEditFiles` del archivo del proyecto hace que el archivo se extraiga del repositorio y se recupere la última versión, el proyecto se descarga y se vuelve a cargar. El método `UpgradeProject` se llama de nuevo una vez que se crea otra instancia del proyecto. En esta segunda llamada, el archivo del proyecto puede escribirse en el disco; se recomienda que el proyecto guarde una copia del archivo del proyecto en el formato anterior con una extensión .OLD, realice los cambios de actualización necesarios y guarde el archivo del proyecto en el nuevo formato. Nuevamente, si se produce un error en cualquier parte del proceso de actualización, el método debe indicarlo mediante la devolución de `VS_E_PROJECTMIGRATIONFAILED`. Esto hace que el proyecto se descargue en el Explorador de soluciones.

Es importante comprender el proceso completo que tiene lugar en el entorno por si la llamada al método `QueryEditFiles` (que especifica un valor de `ReportOnly`) devuelve las marcas `QER_EditNotOK` y `QER_ReadOnlyUnderSc`.

5. El usuario intenta abrir el archivo del proyecto.
6. El entorno llama a su implementación `CanCreateProject`.
7. Si `CanCreateProject` devuelve `true`, entonces el entorno llama a su implementación `CanCreateProject`.
8. El entorno llama a su implementación `Load` para que abra el archivo e inicialice el objeto del proyecto, por ejemplo, `Proyecto1`.
9. El entorno llama a su implementación `IVsProjectUpgrade::UpgradeProject` para determinar si debe actualizarse el archivo del proyecto.
10. Debe llamar a `QueryEditFiles` y pasar un valor de `QEF_ReportOnly` para el parámetro `rgfQueryEdit`.
11. El entorno devuelve `QER_EditNotOK` para `VSQueryEditResult` y el bit `QER_ReadOnlyUnderSc` es establece en `VSQueryEditResultFlags`.
12. Su implementación `IVsProjectUpgrade` llama a `IVsQueryEditQuerySave::QueryEditFiles` (`QEF_ForceEdit_NoPrompting`, `QEF_DisallowInMemoryEdits`).

Esta llamada puede hacer que se extraiga del repositorio una nueva copia del archivo del proyecto y que se recupere la versión más reciente, así como que se deba volver a cargar el archivo del proyecto. En este punto, ocurren dos cosas:

- Si administra su propia recarga del proyecto, el entorno llama a su implementación `ReloadItem` (`VSITEMID_ROOT`). Cuando reciba esta llamada, vuelva a cargar la primera instancia del proyecto (`Proyecto1`) y continúe con la actualización del archivo del proyecto. El entorno sabe que administra su recarga del proyecto si devuelve `true` para `GetProperty` (`VSHPROPID_HandlesOwnReload`).
- Si no administra su recarga del proyecto, devuelve `false` para `GetProperty` (`VSHPROPID_HandlesOwnReload`). En este caso, antes `QueryEditFiles` de que (`QEF_ForceEdit_NoPrompting`, `QEF_DisallowInMemoryEdits`) se devuelva, el entorno crea otra nueva instancia del proyecto, por ejemplo, `Project2`, como se indica a continuación:

1. El entorno llama a [Close](#) en el primer objeto del proyecto, Proyecto1, y coloca este objeto en el estado inactivo.
2. El entorno llama a su implementación `IVsProjectFactory::CreateProject` para crear una segunda instancia de su proyecto, Proyecto2.
3. El entorno llama a su implementación `IPersistFileFormat::Load` para que abra el archivo e inicialice el segundo objeto del proyecto, Proyecto2.
4. El entorno llama a `IVsProjectUpgrade::UpgradeProject` por segunda vez para determinar si se debe actualizar el objeto del proyecto. Sin embargo, esta llamada se realiza en una segunda instancia nueva del proyecto, Proyecto2. Este es el proyecto que se abre en la solución.

NOTE

En caso de que su primer proyecto, Proyecto1, se coloque en el estado inactivo, debe devolver [S_OK](#) desde la primera llamada a su implementación `UpgradeProject`.

5. Debe llamar a [QueryEditFiles](#) y pasar un valor de [QEF_ReportOnly](#) para el parámetro `rgfQueryEdit`.
6. El entorno devuelve [QER_EditOK](#) y la actualización puede continuar porque se puede escribir en el archivo del proyecto.

Si no puede actualizar, devuelva [VS_E_PROJECTMIGRATIONFAILED](#) desde `IVsProjectUpgrade::UpgradeProject`. Si no es necesario realizar ninguna actualización o si opta por no realizar ninguna actualización, trate la llamada `IVsProjectUpgrade::UpgradeProject` como una operación inefectiva. Si devuelve [VS_E_PROJECTMIGRATIONFAILED](#), se agrega un nodo de marcador de posición a la solución del proyecto.

Actualización de elementos de proyecto

Si agrega o administra elementos dentro de sistemas de proyecto que no implementa, es posible que deba participar en el proceso de actualización del proyecto. Crystal Reports es un ejemplo de un elemento que se puede agregar al sistema de proyectos.

Normalmente, los implementadores de elementos de proyecto desean aprovechar un proyecto ya completamente creado y actualizado porque necesitan saber cuáles son las referencias de proyecto y qué otras propiedades del proyecto están allí para tomar una decisión de actualización.

Para obtener la notificación de actualización del proyecto

1. Establezca [SolutionOrProjectUpgrading](#) el indicador (definido en `vsshell80.idl`) en la implementación del elemento de proyecto. Esto hace que el elemento de proyecto `VSPackage` se cargue automáticamente cuando el shell de Visual Studio determina que un sistema de proyecto está en proceso de actualización.
2. Aconseje `IVsSolutionEventsProjectUpgrade` la [AdviseSolutionEvents](#) interfaz a través del método.
3. La `IVsSolutionEventsProjectUpgrade` interfaz se desencadena después de que la implementación del sistema de proyecto haya completado sus operaciones de actualización y se cree el nuevo proyecto actualizado. Dependiendo del escenario, `IVsSolutionEventsProjectUpgrade` la interfaz [OnAfterOpenSolution](#) se [OnAfterOpenProject](#) desencadena [OnAfterLoadProject](#) después de los métodos, `the` o los métodos.

Para actualizar los archivos de elementos del proyecto

1. Debe administrar cuidadosamente el proceso de copia de seguridad de archivos en la implementación del elemento de proyecto. Esto se aplica en particular para una copia `fUpgradeFlag` de [UpgradeProject](#) seguridad en [PUVFF_SXSBACKUP](#) paralelo, donde el parámetro del método se establece en , donde los archivos que se habían realizado una copia de seguridad se colocan a lo largo de los archivos laterales que se designan como ".old". Los archivos de copia de seguridad anteriores a la hora del sistema cuando se

actualizó el proyecto se pueden designar como obsoletos. Además, es posible que se sobrescriban a menos que tome medidas específicas para evitarlo.

2. En el momento en que el elemento de proyecto recibe una notificación de la actualización del proyecto, el **Asistente para conversión** de Visual Studio sigue mostrada. Por lo tanto, debe [IVsUpgradeLogger](#) usar los métodos de la interfaz para proporcionar mensajes de actualización a la interfaz de usuario del asistente.

Vea también

- [Proyectos](#)

Arquitectura de tipos de proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta sección contiene información detallada sobre Visual Studio la arquitectura de los tipos de proyecto en .

En esta sección

- [Elementos de un modelo de proyecto](#)

Enumera los servicios que puede consumir un tipo de proyecto y las interfaces que debe implementar.

- [Componentes principales del modelo de proyecto](#)

Describe los tipos de proyecto de interfaces deben implementarse y, opcionalmente, pueden implementarse para proporcionar funcionalidad adicional.

- [Momento para la creación de tipos de proyecto](#)

Le ayuda a decidir cuándo debe crear un Visual Studio tipo de proyecto y cuándo puede usar otra característica de extensibilidad como VSPackages y editores para lograr el mismo objetivo.

- [Jerarquías y selección](#)

Describe cómo Visual Studio se usan las jerarquías y el contexto de selección para proporcionar una experiencia de usuario coherente y simplificada.

Secciones relacionadas

- [Subtipos de proyecto](#)

Explica cómo los subtipos de proyecto permiten Visual Basic personalizar Visual C++ el comportamiento de los sistemas de proyecto de y .

- [Tipos de proyecto](#)

Proporciona una visión general de los Visual Studio proyectos como los bloques básicos del entorno de desarrollo integrado (IDE). Se proporcionan vínculos a temas adicionales que explican cómo los proyectos controlan la creación y compilación de código.

Elementos de un modelo de proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

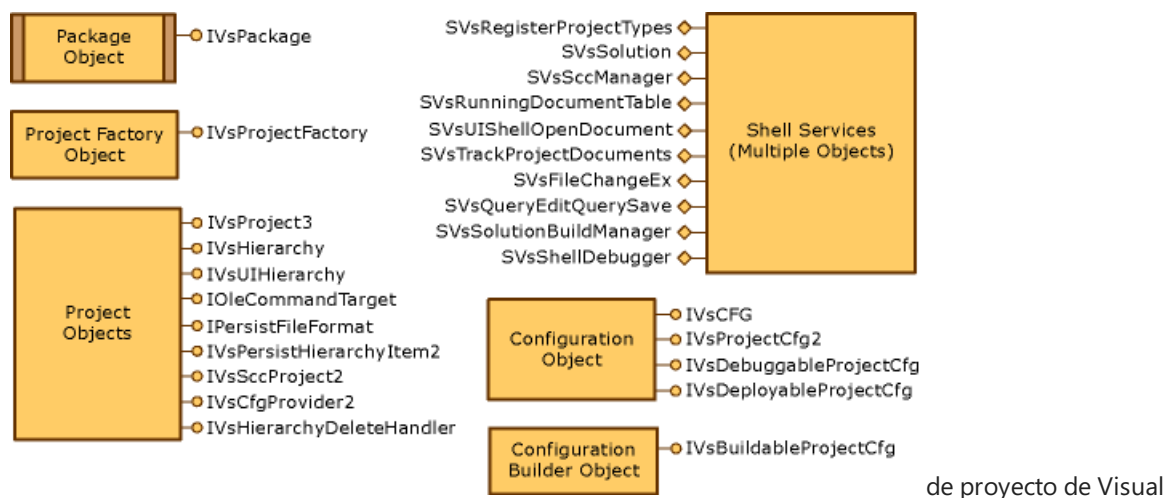
Las interfaces e implementaciones Visual Studio de todos los proyectos en comparten una estructura básica: el modelo de proyecto para el tipo de proyecto. En el modelo de proyecto, que es el VSPackage que está desarrollando, cree objetos que cumplan con las decisiones de diseño y trabajar junto con la funcionalidad global proporcionada por el IDE. Aunque controla cómo se conserva un elemento de proyecto, por ejemplo, no controla la notificación de que se debe conservar un archivo. Cuando un usuario coloca el foco en un elemento de proyecto Visual Studio abierto y elige **Guardar** en el menú **Archivo** de la barra de menús, el código de tipo de proyecto debe interceptar el comando desde el IDE, conservar el archivo y enviar una notificación al IDE de que el archivo ya no se cambia.

El VSPackage interactúa con el IDE a través de servicios que proporcionan acceso a las interfaces IDE. Por ejemplo, a través de determinados servicios, puede supervisar y enrutar comandos y proporcionar información de contexto para las selecciones realizadas en el proyecto. Toda la funcionalidad IDE global necesaria para el VSPackage es proporcionada por los servicios. Para obtener más información acerca de los servicios, vea [Cómo: Obtener un servicio](#).

Otras consideraciones de implementación:

- Un único modelo de proyecto puede contener más de un tipo de proyecto.
- Los tipos de proyecto y los generadores de proyectos asistentes se registran de forma independiente con GUID.
- Cada proyecto debe tener un archivo de plantilla o asistente para inicializar Visual Studio el nuevo archivo de proyecto cuando un usuario crea un nuevo proyecto a través de la interfaz de usuario. Por ejemplo, Visual C++ las plantillas inicializan lo que finalmente se convierten en archivos .vcproj.

En la ilustración siguiente se muestran las interfaces, los servicios y los objetos principales que componen una implementación de proyecto típica. Puede utilizar la aplicación `HierUtil7` auxiliar, para crear los objetos subyacentes y otros reutilizables de programación. Para obtener más `HierUtil7` información acerca de la aplicación auxiliar, vea [Usar clases de proyecto HierUtil7 para implementar un tipo de proyecto \(C++\)](#).



Studio Modelo de proyecto

Para obtener más información acerca de las interfaces y servicios enumerados en el diagrama anterior y otras interfaces opcionales no incluidas en el diagrama, vea [Componentes principales del modelo de proyecto](#).

Los proyectos pueden admitir comandos y, por lo tanto, deben implementar la [IOleCommandTarget](#) interfaz para participar en el enrutamiento de comandos a través de los GUID de contexto de comando.

Vea también

- [Lista de verificación: Crear nuevos tipos de proyecto](#)
- [Utilice las clases de proyecto HierUtil7 para implementar un tipo de proyecto \(C++\)](#)
- [Componentes principales del modelo de proyecto](#)
- [Crear instancias de proyecto mediante fábricas de proyectos](#)
- [Cómo: Obtener un servicio](#)
- [Crear tipos de proyecto](#)

Componentes principales del modelo de proyecto

06/04/2020 • 9 minutes to read • [Edit Online](#)

Las tablas siguientes se expanden en el modelo de proyecto. Las tablas presentan breves descripciones de las interfaces y servicios identificados en el modelo, y las interfaces y servicios asociados con objetos específicos. Además, las tablas detallan otras interfaces que son opcionales en la creación y el mantenimiento del proyecto en función de los requisitos de su tipo de proyecto específico.

Para obtener más información, consulte Compatibilidad con herramientas [de exploración de símbolos](#).

Objeto de paquete

INTERFAZ	COMENTARIOS
IVsPackage	Inicializa un VSPackage en el IDE y hace que sus servicios estén disponibles para el IDE.

Objeto Project Factory

INTERFAZ	COMENTARIOS
IVsProjectFactory	Gestiona la creación de nuevos proyectos y la apertura de proyectos existentes.

Objetos de proyecto

INTERFACES	COMENTARIOS
IVsProject3	Administra la adición y eliminación de elementos de proyecto, abre <code>VSITEMID</code> editores y mantiene la asignación entre cada moniker de documento y el archivo . Hereda <code>IVsProject</code> de <code>IVsProject2</code> y .
IVsHierarchy	Administra las propiedades de navegación y visualización y proporciona eventos.
IVsUIHierarchy	Habilita la ejecución de <code>IOleCommandTarget</code> comandos similar a la de los comandos como Cortar y Cambiar nombre que se aplican solo cuando el foco está en el Explorador de soluciones.
IOleCommandTarget	Sirve como interfaz de destino de comando principal para una jerarquía de proyectos. Es la interfaz estándar para consultar objetos para su estado de comando o estado y ejecutar comandos. Disponible cuando no está enfocado en la ventana Proyecto.
IPersistFileFormat	Coordina la persistencia del estado del proyecto. Normalmente, el estado del proyecto se almacena como un archivo de proyecto, pero se puede adaptar a sistemas de almacenamiento que no están basados en archivos.

INTERFACES	COMENTARIOS
IVsPersistHierarchyItem2	Permite al proyecto administrar todos los aspectos de la persistencia de los elementos de proyecto, ya sea como archivos en disco u objetos en otros sistemas de almacenamiento. La <code>IVsPersistHierarchyItem2</code> interfaz se utiliza para IVsPersistDocData2 los elementos que no implementan la interfaz.
IVsSccProject2	Coordina las interacciones con el control de código fuente.
IVsProjectFlavorCfgProvider	Permite que los proyectos administren la información de configuración.
IVsCfgProvider2	Administra objetos de configuración de proyecto, como configuraciones de depuración/liberación. Las operaciones de compilación, implementación y depuración se coordinan a través de objetos de configuración de proyecto.
IVsHierarchyDeleteHandler	Implementado por jerarquías para controlar las opciones de eliminación (destructiva) o eliminación (no destructivas) para los elementos de jerarquía. Interfaz de <code>IVsHierarchyDeleteHandler</code> consulta de <code>IVsHierarchy</code> llamada en la interfaz de la interfaz.
IVsGetCfgProvider	Proporciona la opción de implementación <code>IVsCfgProvider2</code> de tener el objeto que admite la <code>IVsHierarchy</code> interfaz en una identidad COM diferente que el objeto de proyecto que implementa la interfaz.
IVsProjectStartupServices	Interfaz opcional implementada para que el proyecto sea extensible por otros desarrolladores. La <code>IVsProjectStartupServices</code> interfaz permite que un VSPackage de terceros registre un GUID que se conserva en el archivo de proyecto para <code>QueryService</code> que cada vez que se carga el proyecto, cargue el GUID del servicio de terceros en el archivo de proyecto y llame a ese GUID.
IVsUIHierWinClipboardHelperEvents	Se implementa mediante jerarquías <code>UIHierarchy</code> de origen en una ventana para coordinar operaciones del portapapeles como cortar, copiar y pegar. Utilice <code>AdviseClipboardHelperEvents</code> la interfaz para registrar eventos del portapapeles.
IVsHierarchyDropDataSource2	Proporciona información sobre un elemento arrastrado en relación con su origen de datos durante una operación de arrastrar y colocar en una ventana de jerarquía de interfaz de usuario. Se llama <code>IVsHierarchy</code> desde la interfaz.
IVsHierarchyDropDataTarget	Proporciona información sobre un elemento arrastrado en relación con su destino de colocación durante una operación de arrastrar y colocar en una ventana de jerarquía de interfaz de usuario. Se llama <code>IVsHierarchy</code> desde la interfaz.

objeto de configuración

INTERFACES	COMENTARIOS
IVsCfg	Proporciona información sobre una configuración.
IVsProjectCfg2	Permite que los proyectos administren la información de configuración.
IVsDebuggableProjectCfg	Permite ejecutar un proyecto bajo el control del depurador.
IVsDeployableProjectCfg	Implementado por proyectos de implementación que realizan operaciones de implementación para otros proyectos.

Objeto Configuration Builder

INTERFACES	COMENTARIOS
IVsBuildableProjectCfg	Administra la operación de compilación de una configuración del proyecto.

Objetos adicionales del proyecto

INTERFACES	COMENTARIOS
IDispatch ISpecifyPropertyPages	Muestra las propiedades del elemento en la ventana Propiedades .
IVsOutput2 IVsEnumOutputs	Muestra las salidas para la implementación.

En la tabla siguiente se presentan breves descripciones de los servicios identificados en el modelo de proyecto.

Servicios

SERVICIO	COMENTARIOS
SVsRegisterProjectTypes	Utilizado por VSPackages que implementan tipos de proyecto para registrar que su generador de proyectos existe con el IDE. El VSPackage <code>QueryService</code> debe llamar a este servicio <code>IVsPackage::SetSite</code> y registrar su generador de proyectos cuando se llama al método. Si <code>SetSite</code> no se llama al método, no se crea una instancia del proyecto.
SVsSolution	Proporciona acceso a la noción interna integrada del IDE de la solución actual, como la capacidad de enumerar proyectos, crear nuevos proyectos, tomar nota de los cambios del proyecto, etc.
SVsSccManager	Llamado por proyectos que desean participar en el control de código fuente.
SVsRunningDocumentTable	Mantiene una tabla de documentos abiertos para determinar si uno o varios de los elementos del proyecto ya están abiertos.

SERVICIO	COMENTARIOS
SVsUIShellOpenDocument	Contiene las interfaces y métodos a los que se llama para abrir realmente un elemento de proyecto mediante el editor estándar o un editor específico.
SVsTrackProjectDocuments	Todos los proyectos deben llamarlos cuando agregan, quitan o cambian el nombre de sus elementos.
SVsFileChangeEx	Administra los cambios en un archivo o directorio y notifica a los clientes cuando se han cambiado los archivos seleccionados en el disco.
SVsQueryEditQuerySave	Se requiere que todos los proyectos y editores sean llamados antes de que ensucien los elementos o los guarden.
SVsSolutionBuildManager	Administra el orden de las operaciones de compilación e implementación para las configuraciones de proyecto.
SVsShellDebugger	Proporciona acceso a los servicios de depurador de bajo nivel utilizados para la mayoría de los controles de depuración.
SVsShellMonitorSelection	Habilita VSPackages acceso a información sobre las selecciones actuales y habilita la comunicación con el propiedades ventana.
SVsUIShell	Proporciona funcionalidad básica de IDE relacionada con la interfaz de usuario, como la capacidad de crear y enumerar ventanas de herramientas o ventanas de documentos o para notificar un error al usuario.
SVsStatusbar	Proporciona acceso a la barra de estado del IDE.
IVsExtensibility3	Se utiliza para implementar el modelo de automatización. En el modelo de proyecto, devolverá un objeto properties que le permite crear una instancia de ese objeto.
SVsUIHierWinClipboardHelper	Se utiliza para implementar eventos de portapapeles en el objeto de proyecto en la jerarquía. SVsUIHierWinClipboardHelper le permite manejar correctamente las operaciones de cortar, copiar y pegar.

Vea también

- [IOleCommandTarget](#)
- [Lista de comprobación: creación de nuevos tipos de proyecto](#)
- [No en compilación: uso de clases de proyecto HierUtil7 para implementar un tipo de proyecto \(C++\)](#)
- [Compatibilidad con herramientas de exploración de símbolos](#)
- [Elementos de un modelo de proyecto](#)

Momento para la creación de tipos de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

La creación de un nuevo Visual Studio tipo de proyecto proporciona una base para personalizar para los usuarios. Sin embargo, no es necesario Visual Studio crear un nuevo tipo de proyecto para todas las personalizaciones. Las siguientes directrices deben ayudarle a determinar si se requiere un nuevo tipo de proyecto para su escenario.

Crear un nuevo tipo de proyecto

Debe crear un tipo de proyecto Visual Studio si desea personalizar para que actúe de una o varias de las siguientes maneras:

- Participe en la compilación, la implementación, las configuraciones y el control de código fuente.
- Ofrezca compatibilidad con la depuración.
- Mostrar elementos de proyecto en el Explorador de **soluciones**.
- Utilice el cuadro de diálogo **Abrir proyecto** o **Nuevo proyecto**.
- Apoyar el anidamiento de proyectos.

Extender un tipo de proyecto existente

Es posible que desee crear un Visual Studio nuevo tipo de proyecto que se puede utilizar de las siguientes maneras para modificar o ampliar el comportamiento de un tipo de proyecto existente, por ejemplo, modificar el proceso de compilación de Visual C++ proyectos:

- Trabaje con varios archivos como una sola unidad.
- Mostrar un único archivo como una jerarquía de subelementos.
- Mostrar un contexto de comando alrededor de los editores.
- Mostrar un contexto de servicio para los editores.

Usar un tipo de proyecto existente

A veces no es necesario crear un nuevo proyecto. En la tabla siguiente se muestran las tareas para las que no es necesario crear un tipo de proyecto.

TAREA	DESCRIPCIÓN
Gestión de comandos	Cualquier VSPackage puede controlar comandos.
Construir un editor	Los editores personalizados se pueden registrar. Para obtener más información, consulte Documento de Windows y editores .
Poseer ventanas	Puede crear ventanas de herramientas y documentos sin agregar un nuevo tipo de proyecto.
Exposición de propiedades en la ventana Propiedades	Todos los objetos pueden exponer propiedades.

Crear un subtipo de proyecto

Puede usar subtipos de proyecto para extender un tipo de proyecto administrado sin tener que crear un nuevo tipo de proyecto. Los subtipos de proyecto utilizan la Visual Basic agregación COM para extender proyectos administrados escritos en Microsoft o Visual C#. Con la agregación COM, puede reutilizar gran parte de la implementación del sistema de proyectos administrados y personalizar para un escenario determinado mediante la agregación y el uso de interfaces auxiliares. Para obtener más información acerca de los subtipos de proyecto, vea [Subtipos de proyecto](#).

Vea también

- [Documentar Windows y Editores](#)
- [Lista de comprobación: creación de nuevos tipos de proyecto](#)
- [Jerarquías en Visual Studio](#)

Jerarquías y selección

06/04/2020 • 2 minutes to read • [Edit Online](#)

Al personalizar Visual Studio, debe Visual Studio comprender cómo controla las jerarquías, como los proyectos, y cómo utiliza el contexto de selección para determinar lo que se muestra al usuario. En esta sección se Visual Studio describen los conceptos de jerarquías y selección.

En esta sección

- [Jerarquías en Visual Studio](#)

Describe las jerarquías de proyectos y el concepto general de jerarquías.

- [Selección y moneda en el IDE](#)

Describe cómo Visual Studio el entorno de desarrollo integrado (IDE) mantiene información sobre los objetos activos del usuario actualmente y permite VSPackages realizar un seguimiento de la moneda.

- [Objetos de contexto de selección](#)

Describe el modelo de cómo puede determinar el foco de contexto de selección del usuario en una ventana.

- [Comentarios al usuario](#)

Describe cómo la funcionalidad Visual Studio disponible en se basa en el contexto de selección actual del usuario y el contexto IDE general.

Secciones relacionadas

- [Arquitectura de tipos de proyecto](#)

Proporciona información técnica detallada sobre los tipos de proyecto.

Jerarquías en Visual Studio

06/04/2020 • 3 minutes to read • [Edit Online](#)

El Visual Studio entorno de desarrollo integrado (IDE) muestra un proyecto como una *jerarquía*. En el IDE, una jerarquía es un árbol de nodos, donde cada nodo tiene un conjunto de propiedades asociadas. Una jerarquía de *proyecto* es un contenedor que contiene los elementos del proyecto, las relaciones de los elementos y las propiedades y comandos asociados de los elementos.

En Visual Studio, se administran las jerarquías [IVsHierarchy](#) de proyecto mediante la interfaz de jerarquía, . La [IVsUIHierarchy](#) interfaz redirige los comandos que se invocan desde los elementos del proyecto a la ventana de jerarquía adecuada en lugar del controlador de comandos estándar.

Jerarquías de proyectos

Cada jerarquía de proyectos contiene elementos que puede ver y editar. Estos elementos varían según el tipo de proyecto. Por ejemplo, un proyecto de base de datos puede contener procedimientos almacenados, vistas de base de datos y tablas de base de datos. Un proyecto de lenguaje de programación, por otro lado, probablemente incluirá archivos de origen y archivos de recursos para mapas de bits y cuadros de diálogo. Las jerarquías se pueden anidar, lo que proporciona cierta flexibilidad adicional al crear una jerarquía de proyecto.

Al crear un nuevo tipo de proyecto, el tipo de proyecto controla el conjunto completo de elementos que se pueden editar en él. Sin embargo, los proyectos pueden contener elementos para los que no tienen compatibilidad con la edición. Por ejemplo, los proyectos de Visual C++ pueden contener archivos HTML, aunque Visual C++ no proporciona ningún editor personalizado para el tipo de archivo HTML.

Las jerarquías administran la persistencia de los elementos que contienen. La implementación de la jerarquía debe controlar las propiedades especiales que afectan a la persistencia de los elementos dentro de la jerarquía. Por ejemplo, si los elementos representan objetos en un repositorio en lugar de archivos, la implementación de la jerarquía debe controlar la persistencia de esos objetos. El propio IDE dirige la jerarquía para guardar los elementos de acuerdo con la entrada del usuario, pero el IDE no controla las acciones necesarias para guardar esos elementos. En su lugar, el proyecto está en control.

Cuando un usuario abre un elemento en un editor, la jerarquía que controla ese elemento está seleccionada y se convierte en la jerarquía activa. La jerarquía seleccionada determina el conjunto de comandos disponibles para actuar sobre el elemento. El seguimiento del foco del usuario de esta manera permite que la jerarquía refleje el contexto actual del usuario.

Vea también

- [Tipos de proyecto](#)
- [Selección y moneda en el IDE](#)
- [Muestras VSSDK](#)

Selección y moneda en el IDE

06/04/2020 • 4 minutes to read • [Edit Online](#)

El Visual Studio entorno de desarrollo integrado (IDE) mantiene información sobre los objetos seleccionados actualmente por los usuarios mediante el *contexto de selección*. Con el contexto de selección, VSPackages puede participar en el seguimiento de divisas de dos maneras:

- Mediante la propagación de información de moneda sobre los VSPackages al IDE.
- Mediante la supervisión de las selecciones actualmente activas de los usuarios dentro del IDE.

Contexto de selección

El Visual Studio IDE realiza un seguimiento global de la moneda IDE en su propio objeto de contexto de selección global. En la tabla siguiente se muestran los elementos que componen el contexto de selección.

ELEMENTO	DESCRIPCIÓN
Jerarquía actual	Normalmente el proyecto actual; una jerarquía actual NULL indica que la solución en su conjunto es actual.
ItemID actual	El elemento seleccionado dentro de la jerarquía actual; cuando hay varias selecciones en una ventana de proyecto, puede haber varios elementos actuales.
Actual <code>SelectionContainer</code>	Contiene uno o varios objetos para los que la ventana Propiedades debe mostrar propiedades.

Además, el entorno mantiene dos listas globales:

- Una lista de identificadores de comandos de interfaz de usuario activos
- Una lista de tipos de elementos activos actualmente.

Tipos de ventanas y selección

El Visual Studio IDE organiza las ventanas en dos tipos generales:

- Ventanas de tipo jerarquía
- Ventanas de marco, como ventanas de herramientas y documentos

El IDE realiza un seguimiento de la moneda de forma diferente para cada uno de estos tipos de ventana.

La ventana de tipo de proyecto más común es el explorador de soluciones, que controla el IDE. Una ventana de tipo de proyecto realiza un seguimiento de la jerarquía global y ItemID del contexto de selección global y la ventana se basa en la selección del usuario para determinar la jerarquía actual. Para las ventanas de tipo de [SVsShellMonitorSelection](#) proyecto, el entorno proporciona el servicio global, a través del cual VSPackages puede supervisar los valores actuales de los elementos abiertos. La exploración de propiedades en el entorno está impulsada por este servicio global.

Las ventanas de marco, por otro lado, utilizan DocObject dentro de la ventana de marco para insertar el valor SelectionContext (el trío hierarchy/ItemID/SelectionContainer). Las ventanas [SVsShellMonitorSelection](#) de marco utilizan el servicio para este propósito. DocObject solo puede insertar valores para el contenedor de selección, dejando los valores locales para hierarchy y ItemID sin cambios,

como es típico de los documentos secundarios MDI.

Eventos y Moneda

Pueden producirse dos tipos de eventos que afectan a la noción de moneda del entorno:

- Eventos que se propagan al nivel global y cambian el contexto de selección de marco de ventana. Entre los ejemplos de este tipo de evento se incluye la apertura de una ventana secundaria MDI, una ventana de herramientas global o una ventana de herramientas de tipo proyecto que se está abriendo.
- Eventos que cambian los elementos rastreados dentro del contexto de selección de marco de ventana. Los ejemplos incluyen cambiar la selección dentro de un DocObject o cambiar la selección en una ventana de tipo de proyecto.

Vea también

- [Objetos de contexto de selección](#)
- [Comentarios para el usuario](#)

Objetos de contexto de selección

06/04/2020 • 3 minutes to read • [Edit Online](#)

El Visual Studio entorno de desarrollo integrado (IDE) utiliza un objeto de contexto de selección global para determinar lo que se debe mostrar en el IDE. Cada ventana del IDE puede tener su propio objeto de contexto de selección insertado en el contexto de selección global. El IDE actualiza el contexto de selección global con valores de una ventana cuando esa ventana tiene el foco. Para obtener más información, consulte [Comentarios al usuario](#).

Cada marco de ventana o sitio del [STrackSelection](#) IDE tiene un servicio llamado `QueryService`. El objeto creado por el VSPackage que se encuentra `QueryService` en el marco [ITrackSelection](#) de ventana debe llamar al método para obtener un puntero a la interfaz.

Las ventanas de marco pueden evitar que partes de su información de contexto de selección se propaguen al contexto de selección global cuando se inician. Esta capacidad es útil para las ventanas de herramientas que pueden tener que comenzar con una selección vacía.

Modificar el contexto de selección global desencadena eventos que VSPackages pueden supervisar. VSPackages puede realizar las `IVsTrackSelectionEx` siguientes [IVsMonitorSelection](#) tareas mediante la implementación e interfaces:

- Actualice el archivo activo actualmente en una jerarquía.
- Supervise los cambios en determinados tipos de elementos. Por ejemplo, si el VSPackage usa una ventana **propiedades** especiales, puede supervisar los cambios en la ventana **Propiedades** activa y reiniciar la suya cuando sea necesario.

La siguiente secuencia muestra el curso típico de seguimiento de selección.

1. El IDE recupera el contexto de selección de la ventana recién abierta y lo coloca en el contexto de selección global. Si el contexto de selección utiliza `HIERARCHY_DONTPROPAGATE` o `SELCONTAINER_DONTPROPAGATE`, esa información no se propaga al contexto global. Para obtener más información, consulte [Comentarios al usuario](#).
2. Los eventos de notificación se transmiten a cualquier VSPackage que los haya solicitado.
3. El VSPackage actúa en los eventos que recibe mediante la realización de actividades como actualizar una jerarquía, reactivar una herramienta u otras tareas similares.

Vea también

- [IVsTrackSelectionEx](#)
- [IVsMonitorSelection](#)
- [Jerarquías en Visual Studio](#)
- [Selección y moneda en el IDE](#)
- [Tipos de proyecto](#)

Comentarios al usuario

06/04/2020 • 4 minutes to read • [Edit Online](#)

En Visual Studio el entorno de desarrollo integrado (IDE), la retroalimentación visual con respecto a la funcionalidad disponible se basa en el contexto de selección actual y de selección global del usuario. En la tabla siguiente se muestra la funcionalidad que está disponible en diferentes contextos de selección.

CONTEXTO DE SELECCIÓN	FUNCIONALIDAD DISPONIBLE
IDE	Global
Conjunto de productos actual	Producto específico
Jerarquía activa	Tipo de jerarquía específico
Elemento jerárquico activo	Tipo de artículo de jerarquía específico
Documento activo	Tipo de documento específico
Ventana de interfaz de varios documentos (MDI) superior	Tipo de ventana específico
Contexto de selección actual	Contexto de selección específico

Si solo expone la funcionalidad que necesitan los usuarios y proporciona continuamente una selección coherente y comentarios de contexto de entorno, reducirá la complejidad en el IDE. Las siguientes reglas se aplican cada vez que se abre una ventana en el IDE:

- Si la ventana cambia su contexto de selección, los comentarios de selección se indican claramente en la ventana y la ventana **Ayuda dinámica**, si se muestra, se actualiza para reflejar el contexto actual.
- Si la ventana cambia el contexto de selección global, todos los menús específicos del contexto, la ventana de jerarquía activa y la barra de título de la aplicación se actualizan para reflejar el contexto actual.
- La ventana debe exponer las propiedades de la selección actual en la ventana **Propiedades** y, opcionalmente, si se muestra, el cuadro de diálogo **Páginas** de propiedades.
- Si la ventana no expone las propiedades o cambia el contexto de selección global, los comentarios de selección no deben permanecer en la ventana cuando ya no es la ventana activa en el IDE.
- Todas las ventanas de herramientas específicas del documento deben reflejar continuamente el documento activo.
- Los menús, las barras de herramientas y la barra de título de la aplicación deben reflejar la ventana de cliente de interfaz de varios documentos (MDI) superior.

Por ejemplo, cuando se abre la vista HTML de un **formulario web** dentro `<td>` de un proyecto de aplicación web de Visual Basic y el usuario selecciona una etiqueta, los comentarios se proporcionan de la siguiente manera:

- La selección se indica en la ventana activa y se refleja en la ventana **Propiedades**.
- El cuadro de **herramientas** específico del documento se actualiza para reflejar el documento activo.

- Se muestran la barra de herramientas **Editor** y el menú **Tabla** y la barra de título se actualiza para reflejar la ventana Formulario web.
- La ventana de jerarquía activa, que suele ser el Explorador de **soluciones**, y su barra de título se actualizan para reflejar el contexto actual y los comandos de menú **Proyecto** contextuales ahora se aplican al proyecto de aplicación web activo.

Vea también

- [Selección y moneda en el IDE](#)
- [Objetos de contexto de selección](#)
- [Jerarquías y selección](#)

Subtipos de proyecto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Los subtipos de proyecto permiten personalizar o Visual Studio dar sabor al comportamiento de los sistemas de proyecto de . Las personalizaciones incluyen guardar datos adicionales en el archivo de proyecto, agregar o filtrar elementos en el cuadro de diálogo **Agregar nuevo elemento**, controlar cómo se depuran e implementar los ensamblados y ampliar el cuadro de diálogo **Páginas** de propiedades del proyecto. VSPackages implementan subtipos de proyecto mediante la agregación COM.

NOTE

El sistema de proyectos Visual C++ no admite subtipos de proyecto. Visual Studio utiliza subtipos de proyecto para implementar proyectos de SQL Server y Smart Device.

En esta sección

- [Diseño de subtipos de proyecto](#)

Describe el concepto de subtipos de proyecto.

- [Secuencia de inicialización de subtipos de proyecto](#)

Describe la secuencia de inicialización Visual Studio de subtipo de proyecto mediante programación por entorno.

- [Propiedades y métodos ampliados por subtipos de proyecto](#)

Proporciona descripciones detalladas de las características y métodos que se extienden con más frecuencia mediante subtipos de proyecto.

- [Conservación de datos en el archivo de proyecto de MSBuild](#)

Describe cómo conservar datos en un archivo [IPersistXMLFragment](#) de proyecto y cómo usar para mantener los datos en el archivo de proyecto en los niveles de agregación de subtipos de proyecto.

- [Interfaz de usuario de propiedades de proyecto](#)

Describe cómo los subtipos de proyecto pueden modificar el cuadro de diálogo **Páginas** de propiedades del proyecto.

- [Ampliación del modelo de objetos del proyecto de base](#)

Proporciona información sobre cómo los subtipos de proyecto pueden utilizar los extensores de automatización para ampliar el modelo de objetos de automatización.

- [Contribución al cuadro de diálogo Agregar nuevo elemento](#)

Describe cómo agregar elementos al cuadro de diálogo **Agregar nuevo elemento**.

- [Guardado de datos en archivos de proyecto](#)

Explica cómo un subtipo de proyecto puede guardar y recuperar datos específicos del subtipo en el archivo de proyecto mediante Managed Package Framework (MPF).

- [Control de implementación especializada](#)

Explica cómo los subtipos de proyecto pueden [IVsDeployableProjectCfg](#) proporcionar un comportamiento de implementación especializado mediante la implementación de la interfaz.

- [Adición y eliminación de páginas de propiedades](#)

Describe cómo agregar y quitar páginas de propiedades en el Diseñador de proyectos.

Secciones relacionadas

- [Tipos de proyecto](#)

Proporciona vínculos a Visual Studio temas que detallan proyectos.

Diseño de subtipos de proyecto

06/04/2020 • 11 minutes to read • [Edit Online](#)

Los subtipos de proyecto permiten a VSPackages extender proyectos basados en el motor de compilación de Microsoft (MSBuild). El uso de la agregación le permite reutilizar la Visual Studio mayor parte del sistema de proyecto administrado principal implementado en, pero aún así personalizar el comportamiento para un escenario determinado.

Los temas siguientes detallan el diseño básico y la implementación de subtipos de proyecto:

- Diseño de subtipos de proyecto.
- Agregación de varios niveles.
- Interfaces de apoyo.

Diseño de subtipos de proyecto

La inicialización de un subtipo de proyecto se [IVsHierarchy IVsProject](#) logra agregando el principal y los objetos. Esta agregación permite que un subtipo de proyecto invalide o mejore la mayoría de las capacidades del proyecto base. Los subtipos de proyecto obtienen [IVsHierarchy](#) la [IOleCommandTarget](#) primera [IVsUIHierarchy](#) oportunidad de controlar [IVsProject3](#) las propiedades mediante , comandos mediante y , y la administración de elementos de proyecto mediante . Los subtipos de proyecto también pueden ampliar:

- Objetos de configuración del proyecto.
- Objetos dependientes de la configuración.
- Objetos de exploración independientes de la configuración.
- Objetos de automatización de proyectos.
- Colecciones de propiedades de automatización de proyectos.

Para obtener más información sobre la extensibilidad por subtipos de proyecto, vea [Propiedades y métodos extendidos por subtipos](#) de proyecto .

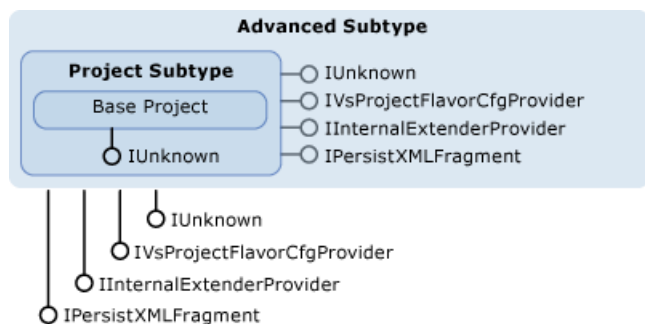
Archivos de directivas

El Visual Studio entorno proporciona un ejemplo de extensión del sistema de proyecto base con un subtipo de proyecto en su implementación de archivos de directivas. Un archivo de directiva Visual Studio permite dar forma al entorno mediante la administración de características que incluyen el Explorador de soluciones, **Agregar proyecto** cuadro de diálogo, **Agregar nuevo elemento** cuadro de diálogo y el **propiedades** cuadro de diálogo. El subtipo de directiva invalida [IVsFilterAddProjectItemDlg](#) [IOleCommandTarget](#) [IVsUIHierarchy](#) mejora estas características a través de , y las implementaciones.

Mecanismo de Agregación

El mecanismo de agregación de subtipos de proyecto del entorno admite varios niveles de agregación, lo que permite implementar un subtipo avanzado saborificando aún más un proyecto con sabor. Además, los objetos auxiliares de [IVsProjectFlavorCfg](#) un subtipo de proyecto, como , están diseñados para permitir varios niveles de capas. De acuerdo con las restricciones de las reglas de agregación COM y COM, los subtipos de proyecto y los proyectos base deben programarse de forma cooperativa para permitir que el subtipo interno o el proyecto base participen correctamente en la delegación de llamadas a métodos y la administración de recuentos de referencias. Es decir, el proyecto a punto de ser agregado tiene que ser programado para admitir la agregación.

En la ilustración siguiente se muestra una representación esquemática de una agregación de subtipos de proyecto de varios niveles.

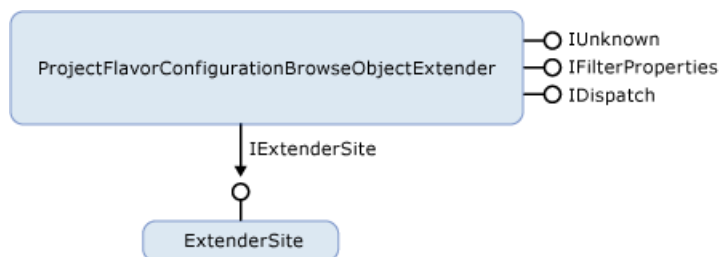


Una agregación de subtipos de proyecto de varios niveles consta de tres niveles, un proyecto base, que se agrega mediante un subtipo de proyecto y, a continuación, se agrega por un subtipo de proyecto avanzado. La figura se centra en algunas de las interfaces Visual Studio auxiliares que se proporcionan como parte de la arquitectura de subtipo según el proyecto.

Mecanismos de despliegue

Entre muchas de las funcionalidades del sistema de proyecto base mejoradas por un subtipo de proyecto se encuentran los mecanismos de implementación. Un subtipo de proyecto influye en los mecanismos [IVsBuildableProjectCfg](#) de implementación mediante la [IVsProjectCfgProvider](#) implementación de interfaces de configuración (como [IVsDeployableProjectCfg](#) y) que se recuperan mediante una llamada a `QueryInterface` en . En un escenario donde tanto el subtipo de proyecto como el subtipo `QueryInterface` de proyecto avanzado agregan `IUnknown` implementaciones de configuración diferentes, el proyecto base llama al subtipo de proyecto avanzado . Si el subtipo de proyecto interno contiene la implementación de configuración que el proyecto base está solicitando, el subtipo de proyecto avanzado delega en la implementación proporcionada por el subtipo de proyecto interno. Como mecanismo para conservar el estado de un nivel de agregación a otro, todos los niveles de subtipos de proyecto se implementan [IPersistXMLFragment](#) para conservar los datos XML no relacionados con la compilación en los archivos de proyecto. Para obtener más información, vea [Persisting Data in the MSBuild Project File](#). [IInternalExtenderProvider](#) se implementa como un mecanismo para recuperar extensores de automatización de los subtipos del proyecto.

La siguiente ilustración se centra en la implementación del extensor de automatización, el objeto de exploración de configuración del proyecto en particular, utilizado por los subtipos de proyecto para ampliar el sistema de proyecto base.



Los subtipos de proyecto pueden ampliar aún más el sistema de proyecto base ampliando el modelo de objetos de automatización. Se definen como una parte del objeto de automatización DTE `ProjectItem` y `Configuration` se utilizan para extender el objeto Project, el objeto y el objeto. Para obtener más información, consulte [Ampliación del modelo de objetos del proyecto base](#).

Agregación multinivel

Una implementación de subtipo de proyecto que envuelve un subtipo de proyecto de nivel inferior debe programarse de forma cooperativa para permitir que el subtipo de proyecto interno funcione correctamente. Una lista de responsabilidades de programación incluye:

- La [IPersistXMLFragment](#) implementación del subtipo de proyecto que está [IPersistXMLFragment](#) ajustando el subtipo interno [Load Save](#) debe delegar en la implementación del subtipo de proyecto interno para ambos y métodos.
- La [IInternalExtenderProvider](#) implementación del subtipo de proyecto contenedor debe delegar al de su subtipo de proyecto interno. En particular, la [GetExtenderNames](#) implementación de necesita obtener la cadena de nombres del subtipo de proyecto interno y, a continuación, concatenar las cadenas que desea agregar como extensores.
- La [IVsProjectCfgProvider](#) implementación de un subtipo [IVsProjectFlavorCfg](#) de proyecto contenedor debe crear una instancia del objeto de su subtipo de proyecto interno y mantenerlo como un delegado privado, ya que solo el objeto de configuración del proyecto base sabe directamente que existe el objeto de configuración de subtipo del proyecto contenedor. El subtipo de proyecto externo puede elegir inicialmente interfaces de configuración que desea controlar directamente [get_CfgType](#), a continuación, delegar el resto en la implementación del subtipo de proyecto interno de .

Interfaces de apoyo

El proyecto base delega las llamadas a las interfaces de soporte agregadas por un subtipo de proyecto, para ampliar varios aspectos de su implementación. Esto incluye extender los objetos de configuración del proyecto y varios objetos del explorador de propiedades. Estas interfaces se recuperan llamando a `QueryInterface` `IUnknown` `punkOuter` (un puntero al) del agregador de subtipo de proyecto más externo.

INTERFAZ	SUBTIPO DE PROYECTO
IVsProjectFlavorCfg	Permite que el subtipo del proyecto: - Proporcionar una IVsDeployableProjectCfg implementación de . - Controlar el inicio del depurador permitiendo que el subtipo de proyecto proporcione su propia implementación de IVsDebuggableProjectCfg. - Deshabilitar la evaluación de expresiones en tiempo de diseño manejando adecuadamente el <code>DBG_LAUNCH_DesignTimeExprEval</code> caso en su implementación de QueryDebugLaunch.
IInternalExtenderProvider	Permite que el subtipo del proyecto: - Ampliar VSHPROPID_BrowseObject el proyecto para añadir o eliminar propiedades independientes de configuración del proyecto. - Ampliar el objeto VSHPROPID_ExtObject de automatización del proyecto () del proyecto. Los valores de <code>__VSHPROPID2</code> propiedad anteriores se toman de la enumeración.
IVsCfgBrowseObject	Permite que el subtipo de IVsCfg proyecto se asigne de nuevo al objeto dado el objeto de exploración de configuración del proyecto.
IVsBrowseObject	Permite que el subtipo de IVsHierarchy proyecto <code>VSITEMID</code> se asigne de nuevo al objeto o al objeto, dado el objeto de exploración de configuración del proyecto.

INTERFAZ	SUBTIPO DE PROYECTO
IPersistXMLFragment	Permite que el subtipo del proyecto conserve datos estructurados XML arbitrarios en el archivo de proyecto (.vbproj o .csproj). Estos datos no son visibles para MSBuild.
IVsBuildPropertyStorage	Permite que el subtipo del proyecto: - Agregar nuevas propiedades de MSBuild que se conservarán. - Quitar propiedades innecesarias de MSBuild. - Consulta para un valor actual de una propiedad mSBuild. - Cambiar el valor actual de una propiedad MSBuild.

Vea también

- [__VSPROPID](#)
- [__VSPROPID2](#)

Secuencia de inicialización de subtipos de proyecto

06/04/2020 • 6 minutes to read • [Edit Online](#)

El entorno construye un proyecto llamando a [CreateProject](#) la implementación de la fábrica del proyecto base de . La construcción de un subtipo de proyecto se inicia cuando el entorno determina que la lista GUID de tipo de proyecto para la extensión de un archivo de proyecto no está vacía. La extensión de archivo de proyecto y Visual Basic Visual C# el GUID del proyecto especifican si el proyecto es un tipo o un tipo de proyecto. Por ejemplo, la extensión .vbproj y el valor de .F184B08F-C81C-45F6-A57F-5ABD9991F28F identifican un Visual Basic proyecto.

La inicialización del entorno de los subtipos de proyecto

El siguiente procedimiento detalla la secuencia de inicialización de un sistema de proyectos agregado por varios subtipos de proyecto.

1. El entorno llama al [CreateProject](#) proyecto base y, mientras el proyecto analiza su archivo de proyecto, `null` detecta que la lista de GUID de tipo de proyecto agregado no es . El proyecto deja de crear directamente su proyecto.
2. El proyecto `QueryService` [SVsCreateAggregateProject](#) llama a service para crear un subtipo [CreateAggregateProject](#) de proyecto mediante la implementación del entorno del método. Dentro de este método, el entorno realiza [PreCreateForOuter](#) llamadas de función recursivas a las implementaciones de , [SetInnerProject](#) y [InitializeForOuter](#) métodos mientras camina por la lista de GUID de tipo de proyecto, empezando por el subtipo de proyecto más externo.

A continuación se detallan los pasos de inicialización.

- a. La implementación del [CreateAggregateProject](#) método del `HrCreateInnerProj` entorno llama al método con la siguiente declaración de función:

```
<CodeContentPlaceholder>0
```

Cuando se llama a esta función por primera vez, es decir, `pOwner` para el `null` subtipo de proyecto más externo, los parámetros `pOuter` y se pasan como y la función establece el subtipo `IUnknown` de proyecto más externo en `pOuter` .

- b. A continuación, `HrCreateInnerProj` el entorno llama a la función con el segundo GUID de tipo de proyecto en la lista. Este GUID corresponde al segundo subtipo de proyecto interno que interpone paso a paso hacia el proyecto base en la secuencia de agregación.
- c. Ahora `pOuter` apunta al `IUnknown` subtipo de proyecto más `HrCreateInnerProj` externo y [PreCreateForOuter](#) llama a la implementación [SetInnerProject](#) de seguida de una llamada a la implementación de . En [PreCreateForOuter](#) el método se `IUnknown` pasa el control del `pOuter` subtipo de proyecto más externo, . El proyecto de propiedad (subtipo de proyecto interno) debe crear su objeto de proyecto agregado aquí. En [SetInnerProject](#) la implementación del método `IUnknown` se pasa un puntero al proyecto interno que se está agregando. Estos dos métodos crean el objeto de agregación y las implementaciones deben seguir las reglas de agregación COM para asegurarse de que un subtipo de proyecto no termina manteniendo un recuento de referencias para sí mismo.
- d. `HrCreateInnerProj` llama a [PreCreateForOuter](#) su implementación de . En este método, el subtipo de proyecto realiza su trabajo de inicialización. Por ejemplo, puede registrar eventos [InitializeForOuter](#) de solución en .

- e. `HrCreateInnerProj` se llama recursivamente hasta que se alcanza el último GUID (el proyecto base) de la lista. Para cada una de estas llamadas, se repiten los pasos, c a d. `pOuter` apunta al subtipo `IUnknown` de proyecto más externo para cada nivel de agregación.

Ejemplo

En el ejemplo siguiente se detalla [CreateAggregateProject](#) el proceso de programación en una representación aproximada del método a medida que lo implementa el entorno. El código es solo un ejemplo; no está destinado a ser compilado, y toda la comprobación de errores se eliminó para mayor claridad.

```
HRESULT CreateAggregateProject
(
    LPCOLESTR lpstrGuids,
    LPCOLESTR pszFilename,
    LPCOLESTR pszLocation,
    LPCOLESTR pszName,
    VSCREATEPROJFLAGS grfCreateFlags,
    REFIID iidProject,
    void **ppvProject)
{
    HRESULT hr = NOERROR;
    CComPtr<IUnknown> srpunkProj;
    CComPtr<IVsAggregatableProject> srpAggProject;
    CComBSTR bstrGuids = lpstrGuids;
    BOOL fCanceled = FALSE;
    *ppvProject = NULL;

    HrCreateInnerProj(
        bstrGuids, NULL, NULL, pszFilename, pszLocation,
        pszName, grfCreateFlags, &srpunkProj, &fCanceled);
    srpunkProj->QueryInterface(
        IID_IVsAggregatableProject, (void **)&srpAggProject));
    srpAggProject->OnAggregationComplete();
    srpunkProj->QueryInterface(iidProject, ppvProject);
}

HRESULT HrCreateInnerProj
(
    WCHAR *pwszGuids,
    IUnknown *pOuter,
    IVsAggregatableProject *pOwner,
    LPCOLESTR pszFilename,
    LPCOLESTR pszLocation,
    LPCOLESTR pszName,
    VSCREATEPROJFLAGS grfCreateFlags,
    IUnknown **ppInner,
    BOOL *pfCanceled
)
{
    HRESULT hr = NOERROR;
    CComPtr<IUnknown> srpInner;
    CComPtr<IVsAggregatableProject> srpAggInner;
    CComPtr<IVsProjectFactory> srpProjectFactory;
    CComPtr<IVsAggregatableProjectFactory> srpAggPF;
    GUID guid = GUID_NULL;
    WCHAR *pwszNextGuids = wcschr(pwszGuids, L';');
    WCHAR wszText[_MAX_PATH+150] = L"";

    if (pwszNextGuids)
    {
        *pwszNextGuids++ = 0;
    }

    CLSIDFromString(pwszGuids, &guid);
    GetProjectTypeMgr()->HrGetProjectFactoryOfGuid(
```

```

        guid, &srpProjectFactory);
srpProjectFactory->QueryInterface(
    IID_IVsAggregatableProjectFactory,
    (void **)&srpAggPF);
srpAggPF->PreCreateForOuter(pOuter, &srpInner);
srpInner->QueryInterface(
    IID_IVsAggregatableProject, (void **)&srpAggInner);

if (pOwner)
{
    IfFailGo(pOwner->SetInnerProject(srpInner));
}

if (pwszNextGuids)
{
    CComPtr<IUnknown> srpNextInner;
    HrCreateInnerProj(
        pwszNextGuids, pOuter ? pOuter : srpInner,
        srpAggInner, pszFilename, pszLocation, pszName,
        grfCreateFlags, &srpNextInner, pfCanceled);
}

return srpAggInner->InitializeForOuter(
    pszFilename, pszLocation, pszName, grfCreateFlags,
    IID_IUnknown, (void **)&ppInner, pfCanceled);
}

```

Vea también

- [Microsoft.VisualStudio.Shell.Flavor](#)
- [Subtipos de proyecto](#)

Propiedades y métodos ampliados por subtipos de proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un subtipo de proyecto tiene mucho poder para influir en el comportamiento del proyecto porque se construye como un agregador de un proyecto base. En esta sección se resumen algunas de las características que se pueden mejorar o modificar mediante subtipos de proyecto.

Características obtenidas por agregación

En la tabla siguiente se resumen muchos de los métodos que la agregación permite que los subtipos de proyecto se invaliden en proyectos base.

MÉTODOS ANULADOS POR LA AGREGACIÓN	SUBTIPO DE PROYECTO
Desde IVsHierarchy : GetProperty SetProperty GetGuidProperty SetGuidProperty	Permite que un subtipo de proyecto - Cambiar el título y el icono del nodo del proyecto. - Anular Browse completamente el objeto de proyecto. - Controlar si se puede cambiar el nombre del proyecto. - Ordenar el control. - Controlar el contexto del usuario para la ayuda dinámica.
Desde IVsProject : GetItemContext	Permite que un subtipo de proyecto controle qué servicios contextuales se proporcionan a diseñadores y editores.
Desde IOleCommandTarget : QueryStatus Exec QueryStatusCommand ExecCommand	Permite que un subtipo de proyecto - Participar en el enrutamiento de comandos para los comandos del proyecto. - Agregar, quitar o deshabilitar los comandos ambientales del proyecto y los comandos activos del Explorador de soluciones.
IVsFilterAddProjectItemDlg2	Permite que el subtipo de proyecto filtre lo que el usuario ve en el cuadro de diálogo Agregar nuevo elemento .
IVsSingleFileGeneratorFactory	Permite que un subtipo de proyecto - Determinar el generador predeterminado dada una extensión de archivo. - Asignar un nombre de generador legible humano a un objeto COM.

Propiedades utilizadas por los subtipos de proyecto

El entorno y el sistema de [__VSSPROPID](#) [__VSSPROPID2](#) proyecto base pueden utilizar las propiedades y enumeraciones detalladas en la tabla siguiente para permitir que un subtipo de proyecto controle varias

características del sistema de proyecto.

PROPIEDAD VSHPROPID	SUBTIPO DE PROYECTO
<code>AddItemTemplatesGuid</code>	Permite que un subtipo de proyecto controle el contenido del cuadro de diálogo Agregar elemento . El subtipo de proyecto puede proporcionar una nueva especificación de directorios de plantilla, agregar nuevos tipos de elementos, quitar elementos existentes y reorganizar un subconjunto de los elementos en el cuadro de diálogo Agregar elemento del proyecto base.
<code>PropertyPagesCLSIDList</code>	Permite que un subtipo de proyecto agregue o quite páginas de propiedades independientes de la configuración.
<code>CfgPropertyPagesCLSIDList</code>	Permite que un subtipo de proyecto agregue o quite páginas de propiedades dependientes de la configuración.
<code>ExtObjectCATID</code>	Permite que un subtipo de proyecto proporcione un extensor de automatización para los objetos de proyecto o elemento de proyecto conociendo el CATID de extensor. Por ejemplo, un subtipo de <code>Project.Extender("<subtype>")</code> proyecto puede proporcionar un objeto personalizado.
<code>BrowseObjectCATID</code>	Permite que un subtipo de proyecto <code>Browse</code> proporcione un extensor de automatización para el objeto conociendo el CATID de extensor. Por ejemplo, un subtipo de proyecto <code>Properties</code> puede agregar propiedades adicionales a la colección.
<code>CfgBrowseObjectCATID</code>	Permite que un subtipo de proyecto proporcione un extensor de automatización para el objeto de exploración de configuración del proyecto. Por ejemplo, un subtipo de proyecto <code>Properties</code> puede agregar propiedades adicionales a la colección.
<code>CfgExtObjectCATID</code>	Permite que un subtipo de proyecto proporcione un extensor de automatización para el objeto de configuración.
<code>DefaultPlatformName</code>	Permite que un subtipo de proyecto determine el nombre de la plataforma para los objetos de configuración del proyecto.

El proyecto base proporciona una implementación predeterminada de las propiedades anteriores. El proyecto base obtiene `QueryInterface` estos `IVsHierarchy` llamando al subtipo de proyecto más externo, lo que permite que el subtipo de proyecto invalide la implementación de las propiedades.

Vea también

- [Diseño de subtipos de proyecto](#)

Conservación de datos en el archivo de proyecto de MSBuild

06/04/2020 • 5 minutes to read • [Edit Online](#)

Es posible que un subtipo de proyecto deba conservar datos específicos del subtipo en el archivo de proyecto para su uso posterior. Un subtipo de proyecto utiliza la persistencia del archivo de proyecto para cumplir los siguientes requisitos:

1. Persista los datos utilizados como parte de la creación del proyecto. (Para obtener más información sobre el motor de compilación de Microsoft, vea [MSBuild](#).) La información relacionada con la compilación puede:
 - a. Datos independientes de la configuración. Es decir, datos almacenados en elementos de MSBuild con condiciones en blanco o que faltan.
 - b. Datos dependientes de la configuración. Es decir, los datos almacenados en elementos de MSBuild que están condicionados para una configuración de proyecto determinada. Por ejemplo:

```
<PropertyGroup Condition=" '$(Configuration)' == 'Debug' ">
```

2. Persista los datos que no son relevantes para compilar. Estos datos se pueden expresar en XML de forma libre que no se valida con un esquema XML.
 - a. Datos independientes de la configuración.
 - b. Datos dependientes de la configuración.

Persistir en la información relacionada con la compilación

La persistencia de datos útiles para crear un proyecto se controla a través de MSBuild. El sistema MSBuild mantiene una tabla maestra de información relacionada con la compilación. Los subtipos de proyecto son responsables de tener acceso a estos datos para obtener y establecer valores de propiedad. Los subtipos de proyecto también pueden aumentar la tabla de datos relacionada con la compilación agregando propiedades adicionales que se conservarán y quitando las propiedades para que no se conserven.

Para modificar los datos de MSBuild, un subtipo de proyecto es responsable de [IVsBuildPropertyStorage](#) recuperar el objeto de propiedad MSBuild desde el sistema del proyecto base a través de [IVsBuildPropertyStorage](#) es una interfaz implementada en el sistema de proyecto principal y la [QueryInterface](#) agregación de consultas de subtipo de proyecto para él mediante la ejecución de [IVsBuildPropertyStorage](#).

El siguiente procedimiento describe los pasos [IVsBuildPropertyStorage](#) para quitar una propiedad mediante [IVsBuildPropertyStorage](#).

Para quitar una propiedad de un archivo de proyecto de MSBuild

1. Llame [QueryInterface](#) [IVsBuildPropertyStorage](#) al subtipo de proyecto.
2. Llame [RemoveProperty](#) [pszPropName](#) con establecido en la propiedad que desea quitar.

Persistir en información relacionada con la construcción

La persistencia de datos en archivos de proyecto [IPersistXMLFragment](#) que no importa compilar se controla a través de [IPersistXMLFragment](#).

Puede implementar [IPersistXMLFragment](#) en [project subtype aggregator](#) el objeto

`project subtype project configuration` principal, el objeto o ambos.

Los siguientes puntos describen los conceptos principales relativos a la persistencia de la información no relacionada con la compilación.

- El proyecto base llama al objeto de agregador de subtipo de proyecto principal (es decir, el subtipo de proyecto más externo) para cargar y guardar datos independientes de la configuración, y llama a los objetos de configuración del proyecto de subtipo para cargar o guardar datos dependientes de la configuración.
- El proyecto base llama [IPersistXMLFragment](#) a los métodos de varias veces para cada nivel de agregación de subtipos de proyecto y pasa el GUID para cada nivel.
- El proyecto base pasa o recibe un fragmento XML dedicado a un subtipo de proyecto determinado y utiliza este mecanismo como una forma de conservar el estado entre los niveles de agregación.
- El proyecto base llama a la [IPersistXMLFragment](#) implementación del subtipo de proyecto más externo pasando un GUID. Si el GUID pertenece al subtipo de proyecto más externo, controla la propia llamada; de lo contrario, delega la llamada a un subtipo de proyecto interno, y así sucesivamente, hasta que se encuentra el subtipo de proyecto al que corresponde el GUID.
- Un subtipo de proyecto también puede modificar el fragmento XML antes o después de delegar la llamada a un subtipo de proyecto interno. En el ejemplo siguiente se muestra un extracto de un archivo de proyecto, donde un nombre de un archivo que contiene propiedades específicas de un subtipo de proyecto, se pasa a ese subtipo de proyecto.

```
<ProjectExtensions>
  <VisualStudio>
    <FlavorProperties GUID="{<FlavorGUID>}">
      <FlavorProject TestFileFolder="TestFile" />
    </FlavorProperties>
  </VisualStudio>
</ProjectExtensions>
```

Vea también

- [Subtipos de proyecto](#)

Interfaz de usuario de propiedades de proyecto

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un subtipo de proyecto puede usar los elementos del cuadro de diálogo **Páginas** de propiedades del proyecto tal como los proporciona el proyecto base, ocultar o crear controles de solo lectura y páginas enteras según lo proporcionado, o agregar páginas específicas del subtipo de proyecto al cuadro de diálogo **Páginas** de propiedades.

Ampliación del cuadro de diálogo Propiedad del proyecto

Un subtipo de proyecto implementa extensores de automatización y objetos de exploración de configuración de proyecto. Estos extensores [IFilterProperties](#) implementan la interfaz para que determinadas propiedades estén ocultas o de solo lectura. El cuadro de diálogo **Páginas** de propiedades del proyecto base, implementado por el proyecto base, respeta el filtrado realizado por los extensores de automatización.

El proceso de ampliación de un cuadro de diálogo Propiedad de **proyecto** se describe a continuación:

- El proyecto base recupera los extensores del [IInternalExtenderProvider](#) subtipo de proyecto mediante la implementación de la interfaz. Los objetos examinar, la automatización del proyecto y la configuración del proyecto implementan esta interfaz.
- La implementación del [IInternalExtenderProvider](#) objeto de exploración del [IInternalExtenderProvider](#) proyecto y el delegado de objeto de automatización del proyecto para la implementación del agregador de subtipo de proyecto (es decir, `QueryInterface` para [IInternalExtenderProvider](#) el [IVsHierarchy](#) objeto de proyecto).
- El objeto de exploración [IInternalExtenderProvider](#) de configuración del proyecto base también implementa para conectar directamente en el extensor de automatización desde el objeto de configuración de subtipo de proyecto. Su implementación se [IInternalExtenderProvider](#) delega en la interfaz implementada por el agregador de subtipos de proyecto.
- [GetProjectItem](#), implementado por el objeto de [IVsHierarchy](#) exploración de configuración del proyecto, devuelve el objeto.
- [GetCfg](#), también implementado por el objeto de [IVsCfg](#) exploración de configuración del proyecto, devuelve el objeto.
- Un subtipo de proyecto puede determinar los CATID adecuados para los distintos objetos extensibles del proyecto base en tiempo de ejecución recuperando los valores siguientes: [__VSHPROPID2](#)
 - [VSHPROPID_ExtObjectCATID](#)
 - [VSHPROPID_BrowseObjectCATID](#)
 - [VSHPROPID_CfgBrowseObjectCATID](#)

Para determinar los CATID para el ámbito del proyecto, el subtipo de proyecto recupera las propiedades anteriores para [VSITEMID](#). `Raíz` de `VSITEMID typedef` la . Un subtipo de proyecto también puede controlar qué páginas de cuadro de diálogo **Páginas** de propiedades se muestran para el proyecto, tanto dependiente de la configuración como independiente de la configuración. Es posible que algunos subtipos de proyecto necesiten quitar páginas integradas y agregar páginas específicas de subtipo de proyecto. Para habilitar esto, el proyecto de [GetProperty](#) cliente administrado llama al método para las siguientes propiedades:

- `VSHPROPID_PropertyPagesCLSIDList` — una lista delimitada por punto y coma de CLSID de páginas de

propiedades independientes de la configuración.

- `VSHPROPID_CfgPropertyPagesCLSIDList` – una lista delimitada por punto y coma de CLSID de páginas de propiedades dependientes de la configuración.

Dado que el subtipo [IVsHierarchy](#) de proyecto agrega el objeto, puede invalidar la definición de estas propiedades para controlar qué cuadros de diálogo **Páginas** de propiedades se muestran. El subtipo de proyecto puede recuperar estas propiedades del proyecto base interno y, a continuación, agregar o quitar CLSDI según sea necesario.

Las nuevas páginas de propiedades agregadas por un subtipo de proyecto se entregan un objeto de exploración de configuración de proyecto de la implementación del proyecto base. Este objeto de exploración de configuración de proyecto admite extensores de automatización. Para obtener más información sobre AutomationExtenders, consulte Implementación y uso de [extensores](#) de automatización . Las páginas de propiedades implementadas [Extender\[String\]](#) por la llamada al subtipo de proyecto para recuperar su propio objeto de exploración de configuración de subtipo de proyecto que extiende el objeto de exploración de configuración del proyecto base.

Vea también

- [IFilterProperties](#)
- [Cuadro de diálogo Páginas de propiedades](#)

Ampliar el modelo de objetos del proyecto base

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un subtipo de proyecto puede ampliar el modelo de objetos de automatización del proyecto base en los siguientes lugares:

- `Project.Extender("<ProjectSubtypeName>")`: esto permite que un subtipo de [Project](#) proyecto ofrezca un objeto con métodos personalizados del objeto. Un subtipo de proyecto puede utilizar `Project` extensores de automatización para exponer el objeto. La [IInternalExtenderProvider](#) interfaz implementada en el agregador de subtipo de `VSHPROPID_ExtObjectCATID` `__VSSPROPID2` proyecto principal `itemid` debe ofrecer su objeto para el from (correspondiente a un valor de [VSITEMID. Raíz](#)) CATID.
- `ProjectItem.Extender("<ProjectSubtypeName>")`: permite que un subtipo de proyecto [ProjectItem](#) ofrezca un objeto con métodos personalizados de un objeto determinado dentro del proyecto. Un subtipo de proyecto puede utilizar extensores de automatización para exponer este objeto. La [IInternalExtenderProvider](#) interfaz implementada en el agregador de subtipo de `VSHPROPID_ExtObjectCATID` proyecto `__VSHPROPID2` principal debe [VSConstants.VSITEMID](#) ofrecer su objeto para el CATID from (correspondiente a un deseado).
- `Project.Properties`: esta colección expone las propiedades `Project` independientes de la configuración del objeto. Para más información sobre propiedades `Project`, vea [Properties](#). Un subtipo de proyecto puede usar Extensores de automatización para agregar sus propiedades a esta colección. La [IInternalExtenderProvider](#) interfaz implementada en el agregador de subtipo de `VSHPROPID_BrowseObjectCATID` proyecto `__VSHPROPID2` principal debe `itemid` ofrecer su objeto para el from (correspondiente a un valor de [VSITEMID. Raíz](#)) CATID.
- `Configuration.Properties`: esta colección expone las propiedades dependientes de la configuración del proyecto para una configuración determinada (por ejemplo, Depurar). Para obtener más información, vea [Configuration](#). Un subtipo de proyecto puede usar Extensores de automatización para agregar sus propiedades a esta colección. La [IInternalExtenderProvider](#) interfaz implementada en el agregador de subtipos de `VSHPROPID_CfgBrowseObjectCATID` proyecto principal `itemid` ofrece su objeto para el CATID (correspondiente a un valor de [VSITEMID. Raíz](#)). La [IVsCfgBrowseObject](#) interfaz se utiliza para distinguir un objeto de exploración de configuración de otro.

Vea también

- [__VSFPROPID](#)

Contribuir al cuadro de diálogo Agregar nuevo elemento

06/04/2020 • 3 minutes to read • [Edit Online](#)

Un subtipo de proyecto puede proporcionar un nuevo directorio completo de elementos para el cuadro de diálogo **Agregar nuevo elemento** registrando Agregar plantillas de **elemento** en la subclave del Registro de **proyectos**.

Registrar Agregar nuevas plantillas de elemento

Esta sección se encuentra en **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0\Proyectos**, en el registro. Las entradas de registro Visual Studio siguientes asumen un proyecto agregado por un subtipo de proyecto hipotético. Las entradas para Visual Studio el proyecto se enumeran a continuación.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0\Projects\{F184B08F-C81C-45F6-A57F-5ABD9991F28F}]
@="#2143"
"DefaultProjectExtension"="vbproj"
"PossibleProjectExtensions"="vbproj;vbp"
"ProjectTemplatesDir"="visualStudioInstallPath\VB\.\VBProjects"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0\Projects\{F184B08F-C81C-45F6-A57F-5ABD9991F28F}\AddItemTemplates\TemplateDirs\{12345678-1234-1234-1122334455667788}\1]
@="#100"
"TemplatesDir"="projectSubTypeTemplatesDir\VBProjectItems"
```

La subclave **AddItemTemplates-TemplateDirs** contiene entradas del Registro con la ruta de acceso al directorio donde se colocan los elementos disponibles en el cuadro de diálogo **Agregar nuevo elemento**.

El entorno carga automáticamente todos los datos **AddItemTemplates** en la subclave del Registro **Proyectos**. Estos datos pueden incluir los datos para las implementaciones de proyectos base, así como los datos para tipos de subtipos de proyecto específicos. Cada subtipo de proyecto se identifica mediante un **GUID** de tipo de proyecto. El subtipo de proyecto puede especificar que se debe usar un conjunto alternativo de

`VSHPROPID_ AddItemTemplatesGuid` plantillas `__VSHPROPID2 GetProperty Add Item` para una instancia de proyecto con sabor determinada al admitir la enumeración de implementación para devolver el valor GUID del subtipo de proyecto. Si `VSHPROPID_ AddItemTemplatesGuid` no se especifica la propiedad, se usa el GUID del proyecto base.

Puede filtrar elementos en el cuadro de `IVsFilterAddProjectItemDlg` diálogo Agregar nuevo **elemento** implementando la interfaz en el objeto de agregador de subtipo de proyecto. Por ejemplo, un subtipo de proyecto que implementa un Visual Studio proyecto de Visual Studio base de datos agregando un proyecto, puede filtrar los elementos específicos del `VSHPROPID_ AddItemTemplatesGuid` `GetProperty` cuadro de diálogo Agregar nuevo **elemento** mediante la implementación de filtrado y, a su vez, puede agregar elementos específicos del proyecto de base de datos mediante la compatibilidad en . Para obtener más información sobre cómo filtrar y agregar elementos al cuadro de diálogo **Agregar nuevo elemento**, vea [Agregar elementos al cuadro de diálogo Agregar nuevo elemento](#) .

Vea también

- [IVsFilterAddProjectItemDlg2](#)
- [CATID](#) para objetos que se utilizan normalmente para extender proyectos

Proyectos web

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un sistema de proyectos web es un sistema de proyectos que crea proyectos web. Los proyectos web a su vez crean aplicaciones web. Puede programar una página Web utilizando Visual Basic lenguajes de programación tradicionales, como o Visual C#, para recopilar y procesar información de un usuario, almacenarla en una base de datos, etc. Las páginas web con código asociado se denominan páginas web inteligentes.

En esta sección

- [Essentials](#)

Presenta los Visual Studio elementos importantes Proyectos Web.

- [Compatibilidad del sitio web](#)

Presenta una Visual Studio visión general de los proyectos de sitio web. Un proyecto de sitio Web genera un archivo ejecutable para cada página web inteligente. Los archivos ejecutables adicionales se generan a partir de los archivos de código fuente de la carpeta /App_Code.

Secciones relacionadas

- [Proyectos](#)

Describe cómo ampliar Visual Studio el sistema del proyecto.

Conceptos básicos del proyecto web

06/04/2020 • 5 minutes to read • [Edit Online](#)

Los proyectos web crean aplicaciones web. Puede usar un proyecto Web para crear una aplicación web que tenga páginas web inteligentes. Una página Web inteligente tiene código del lado servidor que representa la página Web a petición.

Mediante el uso de Visual Basic Visual C#lenguajes de programación tradicionales, como o , puede crear páginas Web inteligentes para recopilar y procesar información de un usuario, almacenarla en una base de datos, etc.

- El modelo de código subyacente asocia archivos de código fuente dependientes con páginas Web que tienen la extensión de archivo .aspx o .asmx. Por ejemplo, hello.aspx podría tener el archivo de código fuente dependiente hello.aspx.cs.
- El código del lado servidor asociado a una página Web inteligente se compila en un archivo ejecutable que se encuentra en la carpeta /bin del sitio Web.
- Los archivos de código fuente adicionales, como las clases auxiliares que no están asociadas a una página Web específica, se encuentran en la carpeta /App_Code del sitio Web.
 - Un proyecto de sitio Web (WSP) genera un archivo ejecutable para cada página Web inteligente. Los archivos ejecutables adicionales se generan a partir de cualquier archivo de código fuente en la carpeta /App_Code.
 - Un proyecto de aplicación web (WAP) genera un único archivo ejecutable que combina el código para todas las páginas Web inteligentes, así como todos los archivos de origen de la carpeta /App_Code.
- El archivo de solución para un proyecto Web se encuentra por separado del propio sitio Web. De forma predeterminada, los archivos de la solución se encuentran en la sección "Documents and Settings YourAccount"(Documentos y configuración de su cuenta) (Mis documentos, \ <Visual Studio) y "> ", "ProjectsYourWebSite".

NOTE

Si desea mantener el archivo de solución con el sitio Web, simplemente muévelo allí y vuelva a abrirlo.

- Si abre un sitio Web que no tiene ningún archivo de solución en Visual Studio, se genera automáticamente un nuevo archivo de solución para él.
- Los proyectos web no tienen archivos de proyecto. La información del proyecto se almacena en el archivo de solución, el archivo web.config y en otro lugar.
- Agregar propiedades globales a un proyecto Web crea automáticamente un archivo de almacenamiento en la carpeta de solución de proyecto web.
- Una página Web inteligente se puede asociar a un lenguaje de programación del lado servidor mediante la directiva Page o el script runat="server"> etiqueta.
- Además, las páginas Web pueden tener cualquier número de bloques de scripting del lado cliente escritos en cualquier lenguaje de scripting.
- Un sistema de proyecto de sitio Web se implementa Visual Web Developer agregando plantillas de proyecto y elemento y registro al proyecto.

- Un sistema WAP se implementa como un subtipo de proyecto, también llamado sabor del proyecto. El Visual Web Developer proyecto está aromatizado por el subtipo WAP para crear el sistema WAP. Para obtener más información sobre los subtipos de proyecto, vea [Subtipos de proyecto](#).
- Una página Web inteligente combina HTML con un lenguaje de programación del lado servidor. El idioma del lado servidor se denomina idioma contenido. Para admitir un lenguaje contenido, el sistema [IVsContainedLanguage](#) de proyectos Web debe implementar la familia de interfaces.
 - Para admitir el lenguaje contenido en un editor, el servicio de lenguaje HTML debe aplazar la visualización de código de idioma contenido a un servicio de lenguaje contenido.
 - Los marcadores de error (rotulaciones rojas) siempre deben crearse en el búfer principal del editor de código.

Vea también

- [Proyectos web](#)

Compatibilidad del sitio web

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un sistema de proyectos de sitio Web es un sistema de proyectos que crea proyectos web. Los proyectos web a su vez crean aplicaciones web. Un proyecto de sitio Web genera un archivo ejecutable para cada página Web que tiene código asociado. Los archivos ejecutables adicionales se generan a partir de los archivos de código fuente de la carpeta /App_Code.

Los sistemas de proyectos de sitio web se crean agregando plantillas y atributos de registro a un sistema de proyectos existente. Uno de estos atributos selecciona el proveedor IntelliSense para el idioma. La implementación del proveedor IntelliSense controla las referencias y llama al compilador de lenguaje cuando se solicita una página Web inteligente que no está almacenada en caché.

El compilador de lenguaje utilizado para ASP.NETcompilar páginas Web debe estar registrado con . Puede utilizar < el compilador> Element en un archivo Web.config para registrar el compilador, como en el ejemplo siguiente:

```
<system.codedom> <compilers> <compiler language="py;IronPython" extension=".py"
  type="IronPython.CodeDom.PythonProvider, IronPython, Version=1.0.2391.18146, Culture=neutral,
  PublicKeyToken=b03f5f7f11d50a3a" /> </compilers></system.codedom>
```

En esta sección

- [Plantillas de compatibilidad del sitio web](#)

Enumera las plantillas que puede usar para crear nuevos proyectos de sitio Web y elementos asociados.

- [Atributos de compatibilidad del sitio web](#)

Presenta los atributos de registro Visual Studio que ASP.NETconectan un proyecto de sitio Web a y .

Secciones relacionadas

- [Proyectos web](#)

Presenta una visión general de los dos tipos de proyectos web, proyectos de sitio web y proyectos de aplicación web.

Plantillas de compatibilidad del sitio web

06/04/2020 • 12 minutes to read • [Edit Online](#)

Visual StudioLas plantillas de proyecto y elemento de sitio web proporcionan códigos auxiliares de elementos y proyectos de sitio web reutilizables y personalizables que aceleran el proceso de desarrollo eliminando la necesidad de crear nuevos proyectos y elementos de sitio web desde cero. Para obtener Visual Studio más información sobre las plantillas, vea Crear plantillas de [proyecto y elemento](#).

Carpeta de plantilla de proyecto

Las plantillas de proyecto web se instalan normalmente en [Ruta de\instalación de Visual *Studio*], Common7, IDE, ProjectTemplates, Web, cada una en una subcarpeta que recibe el nombre del lenguaje de programación web.

Archivo de proyecto

El Visual Studio entorno de desarrollo integrado (IDE) requiere una extensión de archivo de proyecto como una forma de asignar una plantilla al tipo de proyecto correcto. Dado que los proyectos web no tienen un archivo de proyecto, la extensión de archivo de proyecto ficticio .webproj se registra para asignar la plantilla al tipo de proyecto.

Opcionalmente, se puede agregar una cadena de nombre de idioma a la plantilla, para permitir que el sistema de proyectos web establezca el idioma predeterminado en el cuadro de diálogo **Agregar nuevo elemento** para los elementos basados en la plantilla. La cadena debe ser la primera línea del archivo. Debe coincidir con el nombre registrado en AddItemLanguageName en el registro del motor de IntelliSense y el nombre registrado en Subtipo de proyecto (VsTemplate). Para obtener más información, consulte [Atributos](#)de compatibilidad con sitios web .

Si la cadena no está presente, el sistema de proyectos web intenta determinar el idioma predeterminado en función del atributo Language y las extensiones de archivo de las páginas agregadas al proyecto Web por la plantilla de proyecto.

Plantillas de proyecto

Las plantillas de proyecto de sitio web se usan para crear nuevos sitios Web en respuesta al comando **Nuevo sitio web** del menú **Archivo**. Actualmente se admiten tres tipos de proyectos de sitio web:

- Proyectos de sitios web vacíos
- Proyectos de sitios web
- Proyectos de servicios web

Proyectos de sitios web vacíos

Estos archivos crean un nuevo sitio Web vacío en respuesta al comando **Sitio Web vacío**, que está disponible después de elegir **Archivo > nuevo sitio Web**:

- EmptyWeb.vstemplate

El archivo de plantilla que guía la creación del nuevo sitio Web vacío.

- EmptyWeb.webproj

Este archivo es un artefacto del sistema de plantillas de proyecto. Satisface la referencia del archivo de proyecto en el archivo EmptyWeb.vstemplate.

Proyectos de Sitios Web

Estos archivos crean un nuevo sitio Web en respuesta al comando **ASP.NET sitio Web**, que está disponible después de elegir **Archivo > nuevo sitio Web**:

- Default.aspx

La página de inicio predeterminada para el nuevo sitio Web. El language atributo especifica el lenguaje de código subyacente y el CodeFile atributo especifica el archivo dependiente que contiene el código subyacente código asociado a esta página.

- Default.aspx. *extensión*

El archivo dependiente que contiene el código subyacente para la página principal predeterminada. El lenguaje de código subyacente determina la *extensión* de este archivo.

- web.config

El archivo de configuración web.site raíz.

- WebApplication.vstemplate

El archivo de plantilla que determina el contenido de la solución de sitio Web y fuerza la creación de la carpeta App_Data.

- WebApplication.webproj

Este archivo es un artefacto del sistema de plantillas de proyecto. Satisface la referencia del archivo de proyecto en el archivo WebApplication.vstemplate.

Proyectos de Servicio Web

Estos archivos crean un nuevo sitio Web en respuesta al comando **ASP.NET servicio web**, que está disponible después de elegir **Archivo > nuevo sitio Web**:

- Service.aspx

La página HTML del nuevo servicio web. El language atributo especifica el lenguaje de código subyacente y el CodeBehind atributo especifica el archivo dependiente que contiene el código subyacente código asociado a este servicio.

- Servicio. *Extensión*

El archivo dependiente que implementa la clase de servicio. El lenguaje de código subyacente determina la *extensión* de este archivo.

- web.config

- El archivo de configuración web.site raíz.

- WebService.vstemplate

El archivo de plantilla que determina el contenido de la solución de sitio Web y fuerza la creación de las carpetas App_Data y App_Code. El servicio. *archivo* de extensión se copia en la carpeta App_Code.

- WebService.webproj

Este archivo es un artefacto del sistema de plantillas de proyecto. Satisface la referencia del archivo de proyecto en el archivo WebService.vstemplate.

Carpeta de plantilla de elemento de proyecto

Las plantillas de elemento sde proyecto web se instalan normalmente en [Ruta\de instalación de Visual *Studio*],

Common7, IDE, ItemTemplates, Web, cada una de ellas en una subcarpeta con el nombre de su lenguaje de programación web.

Plantillas de elementos de proyecto

Las plantillas de elemento de proyecto de sitio web se usan para agregar nuevas páginas Web a un sitio Web en respuesta al comando **Agregar elemento existente**. Actualmente se admiten estos tipos de páginas Web:

- Nueva clase
- Nueva página HTML
- Nuevo formulario web
- Nueva página maestra

Nueva clase

Esta plantilla crea un nuevo archivo de origen que define una clase vacía en respuesta al comando **Agregar nueva clase**.

- Clase. *Extensión*

El archivo de origen que implementa la clase vacía. El lenguaje de código subyacente determina la *extensión* de este archivo.

- Class.vstemplate

El archivo de plantilla que crea el archivo de origen y determina su contenido.

Nueva página HTML

Esta plantilla crea una nueva página Web en respuesta al comando **Agregar nueva página HTML**.

- HTMLPage.htm

El contenido inicial de la página Web. Esta página Web normalmente no tiene ningún archivo dependiente de código asociado. Para crear una página inteligente con un archivo de código subyacente asociado, utilice la plantilla de formulario web en su lugar.

- HTMLPage.vstemplate

El archivo de plantilla que crea la página Web y determina su contenido.

Nuevo WebForm

Esta plantilla crea una nueva página Web inteligente en respuesta al comando **Agregar nuevo formulario web**.

Para crear un archivo de código fuente dependiente detrás, seleccione **Colocar código en un archivo independiente**. De lo contrario, se crea una sola página <Web que tiene un bloque de scripting vacío y no % Page %> directivas para enlazar un archivo dependiente.

Para crear una página de contenido para una página maestra seleccionada, seleccione **Seleccionar página maestra**.

- WebForm.aspx

El contenido inicial de la página Web. Esta página Web no tiene ningún código asociado detrás del archivo dependiente.

- WebForm_cb.aspx

El contenido inicial de la página Web. Esta página Web tiene un archivo dependiente de código asociado.

- Codebehind. *Extensión*

El archivo dependiente que implementa la clase de formulario web. El lenguaje de código subyacente determina la *extensión* de este archivo.

- ContentPage.aspx

El contenido inicial de la página Web como una página de contenido. Esta página Web no tiene ningún código asociado detrás del archivo dependiente.

- ContentPage_cb.aspx

El contenido inicial de la página Web como una página de contenido. Esta página Web tiene un archivo dependiente de código asociado.

- WebForm.vstemplate

El archivo de plantilla que determina el contenido de la nueva página web y su archivo dependiente, si existe.

Nueva página maestra

Esta plantilla crea una nueva página maestra en respuesta al comando **Agregar nueva página maestra**.

Para crear un archivo de código fuente dependiente detrás, seleccione **Colocar código en un archivo independiente**. De lo contrario, se crea una sola <página Web que tiene un bloque de scripting vacío y no % Page %> directivas para enlazar un archivo dependiente.

- MasterPage.master

El contenido inicial de la página maestra. Esta página maestra no tiene ningún código asociado detrás del archivo dependiente.

- MasterPage_cb.maestro

El contenido inicial de la página maestra. Esta página maestra tiene un archivo dependiente de código asociado.

- Codebehind. *extensión*

El archivo dependiente que implementa la clase de página maestra. El lenguaje de código subyacente determina la *extensión* de este archivo.

- MasterPage.vstemplate

El archivo de plantilla que determina el contenido de la nueva página maestra y su archivo dependiente, si existe.

Vea también

- [Compatibilidad del sitio web](#)

Atributos de compatibilidad del sitio web

06/04/2020 • 3 minutes to read • [Edit Online](#)

Visual StudioEl proyecto de sitio web se puede ampliar para proporcionar compatibilidad con lenguajes de programación web. El idioma debe registrarse para que las plantillas de proyecto puedan aparecer en el cuadro de diálogo **Nuevo sitio Web** cuando se seleccione el idioma.

El ejemplo de IronPython Studio incluye compatibilidad con sitios web. El ejemplo contiene las siguientes clases de atributo para registrar IronPython como un lenguaje de código subyacente para nuevos proyectos Web.

WebSiteProjectAttribute

Este atributo se coloca en el proyecto de lenguaje. Agrega el idioma a la lista de lenguajes de programación Web en la lista **Idioma** del cuadro de diálogo **Nuevo sitio Web**. Por ejemplo, el código siguiente agrega IronPython a la lista:

```
[WebSiteProject("IronPython", "Iron Python")]
public class PythonProjectPackage : ProjectPackage
```

Este atributo también establece la ruta de acceso de las plantillas para que apunte a la carpeta templates. Para obtener más información sobre la ubicación de la carpeta de plantillas, vea Plantillas de [compatibilidad con sitios web](#).

WebSiteProjectRelatedFilesAttribute

Este atributo se coloca en el proyecto de lenguaje. Permite que el proyecto de sitio Web anide un tipo de archivo (relacionado) en otro tipo de archivo (primario) en el Explorador de **soluciones**.

Por ejemplo, el código siguiente especifica que un archivo de código subyacente IronPython está relacionado con un archivo .aspx. Cuando se crea una nueva página Web .aspx en una solución de sitio Web de IronPython, se genera un nuevo archivo de origen .py y aparece como un nodo secundario de la página .aspx.

```
[WebSiteProjectRelatedFiles("aspx", "py")]
public class PythonProjectPackage : ProjectPackage
```

ProvideIntellisenseProviderAttribute

Este atributo se coloca en el paquete de proyecto de idioma. Selecciona el proveedor IntelliSense para el idioma.

Por ejemplo, el código siguiente especifica que se debe crear [IVsIntellisenseProject](#) una instancia de `PythonIntellisenseProvider`, que implementa , a petición para proporcionar servicios de lenguaje.

```
[ProvideIntellisenseProvider(typeof(PythonIntellisenseProvider), "IronPythonCodeProvider", "Iron Python",
".py", "IronPython;Python", "IronPython")]
public class PythonPackage : Package, IOleComponent
```

El `IVsIntellisenseProject` implementación controla las referencias y llama al compilador de lenguaje cuando se solicita una página Web con código, pero no se almacena en caché.

Vea también

- [Compatibilidad del sitio web](#)

Manejar la implementación especializada

06/04/2020 • 6 minutes to read • [Edit Online](#)

Una implementación es una operación opcional para proyectos. Un proyecto Web, por ejemplo, admite una implementación para permitir que un proyecto actualice un servidor Web. Del mismo modo, un proyecto de **dispositivo inteligente** admite una implementación para copiar una aplicación compilada en el dispositivo de destino. Los subtipos de proyecto pueden proporcionar [IVsDeployableProjectCfg](#) un comportamiento de implementación especializado mediante la implementación de la interfaz. Esta interfaz define un conjunto completo de las operaciones de implementación:

- [AdviseDeployStatusCallback](#)
- [Commit](#)
- [QueryStartDeploy](#)
- [QueryStatusDeploy](#)
- [Rollback](#)
- [StartDeploy](#)
- [StopDeploy](#)
- [UnadviseDeployStatusCallback](#)

La operación de implementación real se Visual Studio debe realizar en el subprocesso independiente para que sea aún más sensible a la interacción del usuario. Los métodos [IVsDeployableProjectCfg](#) proporcionados por Visual Studio se llaman de forma asíncrona y funcionan en segundo plano, lo que permite al entorno consultar el estado de una operación de implementación en cualquier momento o detener la operación, si es necesario. El [IVsDeployableProjectCfg](#) entorno llama a las operaciones de implementación de la interfaz cuando el usuario selecciona el comando deploy.

Para notificar al entorno que una operación de implementación ha [OnStartDeploy](#) comenzado [OnEndDeploy](#) o finalizado, el subtipo de proyecto debe llamar a los métodos y.

Para gestionar una implementación especializada mediante un proyecto de subtipo

- Implemente [AdviseDeployStatusCallback](#) el método para registrar el entorno para recibir notificaciones de eventos de estado de implementación.

```
Private adviseSink As Microsoft.VisualStudio.Shell.EventSinkCollection = New
Microsoft.VisualStudio.Shell.EventSinkCollection()
Public Function AdviseDeployStatusCallback(ByVal pIVsDeployStatusCallback As IVsDeployStatusCallback, _
                                           ByRef pdwCookie As UInteger) As Integer

    If pIVsDeployStatusCallback Is Nothing Then
        Throw New ArgumentNullException("pIVsDeployStatusCallback")
    End If

    pdwCookie = adviseSink.Add(pIVsDeployStatusCallback)
    Return VSConstants.S_OK

End Function
```

```
private Microsoft.VisualStudio.Shell.EventSinkCollection adviseSink = new
Microsoft.VisualStudio.Shell.EventSinkCollection();
public int AdviseDeployStatusCallback(IVsDeployStatusCallback pIVsDeployStatusCallback,
out uint pdwCookie)
{
    if (pIVsDeployStatusCallback == null)
        throw new ArgumentNullException("pIVsDeployStatusCallback");

    pdwCookie = adviseSink.Add(pIVsDeployStatusCallback);
    return VSConstants.S_OK;
}
```

- Implemente [UnadviseDeployStatusCallback](#) el método para cancelar el registro del entorno para recibir notificaciones de eventos de estado de implementación.

```
Public Function UnadviseDeployStatusCallback(ByVal dwCookie As UInteger) As Integer
    adviseSink.RemoveAt(dwCookie)
    Return VSConstants.S_OK
End Function
```

```
public int UnadviseDeployStatusCallback(uint dwCookie)
{
    adviseSink.RemoveAt(dwCookie);
    return VSConstants.S_OK;
}
```

- Implemente [Commit](#) el método para realizar la operación de confirmación específica de la aplicación. Este método se utiliza principalmente para la implementación de bases de datos.

```
Public Function Commit(ByVal dwReserved As UInteger) As Integer
    'Implement commit operation here.
    Return VSConstants.S_OK
End Function
```

```
public int Commit(uint dwReserved)
{
    //Implement commit operation here.
    return VSConstants.S_OK;
}
```

- Implemente [Rollback](#) el método para realizar una operación de reversión. Cuando se llama a este método, el proyecto de implementación debe hacer lo que sea adecuado para revertir los cambios y restaurar el estado del proyecto. Este método se utiliza principalmente para la implementación de bases de datos.

```
Public Function Commit(ByVal dwReserved As UInteger) As Integer
    'Implement commit operation here.
    Return VSConstants.S_OK
End Function
```

```

public int Rollback(uint dwReserved)
{
    //Implement Rollback operation here.
    return VSConstants.S_OK;
}

```

- Implemente [QueryStartDeploy](#) el método para determinar si un proyecto puede iniciar o no una operación de implementación.

```

Public Function QueryStartDeploy(ByVal dwOptions As UInteger, ByVal pfSupported As Integer(), ByVal pfReady As Integer()) As Integer
    If Not pfSupported Is Nothing AndAlso pfSupported.Length > 0 Then
        pfSupported(0) = 1
    End If
    If Not pfReady Is Nothing AndAlso pfReady.Length > 0 Then
        pfReady(0) = 0
        If Not deploymentThread Is Nothing AndAlso (Not deploymentThread.IsAlive) Then
            pfReady(0) = 1
        End If
    End If
    Return VSConstants.S_OK
End Function

```

```

public int QueryStartDeploy(uint dwOptions, int[] pfSupported, int[] pfReady)
{
    if (pfSupported != null && pfSupported.Length > 0)
        pfSupported[0] = 1;
    if (pfReady != null && pfReady.Length > 0)
    {
        pfReady[0] = 0;
        if (deploymentThread != null && !deploymentThread.IsAlive)
            pfReady[0] = 1;
    }
    return VSConstants.S_OK;
}

```

- Implemente [QueryStatusDeploy](#) el método para determinar si una operación de implementación se ha completado correctamente o no.

```

Public Function QueryStatusDeploy(ByRef pfDeployDone As Integer) As Integer
    pfDeployDone = 1
    If Not deploymentThread Is Nothing AndAlso deploymentThread.IsAlive Then
        pfDeployDone = 0
    End If
    Return VSConstants.S_OK
End Function

```

```

public int QueryStatusDeploy(out int pfDeployDone)
{
    pfDeployDone = 1;
    if (deploymentThread != null && deploymentThread.IsAlive)
        pfDeployDone = 0;
    return VSConstants.S_OK;
}

```

- Implemente [StartDeploy](#) el método para iniciar una operación de implementación en un subproceso

independiente. Coloque el código específico de la `Deploy` implementación de la aplicación dentro del método.

```
Public Function StartDeploy(ByVal pIVsOutputWindowPane As IVsOutputWindowPane, ByVal dwOptions As
UInteger) As Integer
    If pIVsOutputWindowPane Is Nothing Then
        Throw New ArgumentNullException("pIVsOutputWindowPane")
    End If

    If Not deploymentThread Is Nothing AndAlso deploymentThread.IsAlive Then
        Throw New NotSupportedException("Cannot start deployment operation when it is already started;
Call QueryStartDeploy first")
    End If

    outputWindow = pIVsOutputWindowPane

    ' Notify that deployment is about to begin and see if any user wants to cancel.
    If (Not NotifyStart()) Then
        Return VSConstants.E_ABORT
    End If

    operationCanceled = False

    ' Create and start our thread
    deploymentThread = New Thread(AddressOf Me.Deploy)
    deploymentThread.Name = "Deployment Thread"
    deploymentThread.Start()

    Return VSConstants.S_OK
End Function
```

```
public int StartDeploy(IVsOutputWindowPane pIVsOutputWindowPane, uint dwOptions)
{
    if (pIVsOutputWindowPane == null)
        throw new ArgumentNullException("pIVsOutputWindowPane");

    if (deploymentThread != null && deploymentThread.IsAlive)
        throw new NotSupportedException("Cannot start deployment operation when it is already started;
Call QueryStartDeploy first");

    outputWindow = pIVsOutputWindowPane;

    // Notify that deployment is about to begin and see if any user wants to cancel.
    if (!NotifyStart())
        return VSConstants.E_ABORT;

    operationCanceled = false;

    // Create and start our thread
    deploymentThread = new Thread(new ThreadStart(this.Deploy));
    deploymentThread.Name = "Deployment Thread";
    deploymentThread.Start();

    return VSConstants.S_OK;
}
```

- Implemente [StopDeploy](#) el método para detener una operación de implementación. Se llama a este método cuando un usuario presiona el botón **Cancelar** durante el proceso de implementación.

```

Public Function StopDeploy(ByVal fSync As Integer) As Integer
    If Not deploymentThread Is Nothing AndAlso deploymentThread.IsAlive Then
        Return VSConstants.S_OK
    End If

    outputWindow.OutputStringThreadSafe("Canceling deployment")
    operationCanceled = True
    If fSync <> 0 Then
        ' Synchronous request, wait for the thread to terminate.
        If (Not deploymentThread.Join(10000)) Then
            Debug.Fail("Deployment thread did not terminate before the timeout, Aborting thread")
            deploymentThread.Abort()
        End If
    End If

    Return VSConstants.S_OK
End Function

```

```

public int StopDeploy(int fSync)
{
    if (deploymentThread != null && deploymentThread.IsAlive)
        return VSConstants.S_OK;

    outputWindow.OutputStringThreadSafe("Canceling deployment");
    operationCanceled = true;
    if (fSync != 0)
    {
        // Synchronous request, wait for the thread to terminate.
        if (!deploymentThread.Join(10000))
        {
            Debug.Fail("Deployment thread did not terminate before the timeout, Aborting thread");
            deploymentThread.Abort();
        }
    }

    return VSConstants.S_OK;
}

```

NOTE

Todos los ejemplos de código proporcionados en este tema son partes de un ejemplo más amplio en [ejemplos VSSDK](#).

Vea también

- [Subtipos de proyecto](#)

Contribuir al modelo de automatización

06/04/2020 • 4 minutes to read • [Edit Online](#)

Visual Studio proporciona un conjunto de interfaces de automatización para personalizar el entorno. El modelo de automatización es el modelo de objetos que permite a los usuarios finales crear complementos y extensiones de Visual Studio.

Además, es adecuado para usted, como desarrollador de VSPackage, para contribuir al modelo de automatización; al hacerlo, permite a los usuarios finales de SU VSPackage crear complementos y, por Visual Studio en general, proporcionar una experiencia de modelo de usuario coherente cuando usan el VSPackage en .

Para que la experiencia del usuario final sea coherente, puede seguir un conjunto de directrices a medida Visual Studio que diseña el VSPackage para que el modelo de automatización para el VSPackage siga las ideas de .

En esta sección

- [Visión general del modelo de automatización](#)

Define el modelo de automatización como un grupo relacionado de objetos que controlan las facetas principales del entorno común. Este conjunto de objetos se muestra en un diagrama del modelo de automatización.

- [Proporcionar automatización para VSPackages](#)

Describe las dos formas principales de proporcionar automatización para el VSPackage.

- [Exponer objetos de proyecto](#)

Proporciona instrucciones paso a paso para crear objetos específicos de VSPackage.

- [Modelado de proyectos](#)

Explica los objetos de proyecto estándar necesarios para crear la automatización para el nuevo tipo de proyecto e ilustra la ruta de acceso que sigue la automatización del proyecto. En este tema también se proporcionan listas de declaraciones e implementación para las clases.

- [Exponer eventos](#)

Proporciona instrucciones paso a paso para crear eventos para el modelo de automatización.

- [Soporte de automatización para páginas de opciones](#)

Describe cómo devolver un objeto de automatización para admitir las propiedades del cuadro de `DTE.Properties` diálogo **Opciones** personalizadas de un VSPackage en el menú **Herramienta** extendiendo el objeto.

- [Proporcionar automatización para el código](#)

Explica que no es necesario crear un modelo de automatización para el código. Sin embargo, se proporciona un vínculo en este tema que proporciona información detallada en los modelos de código.

- [Cómo: Proporcionar automatización para Windows](#)

Explica que proporcionar automatización es una buena idea siempre que desee que los objetos de automatización estén disponibles en una ventana y que el entorno no proporcione ya un objeto de automatización listo. Describe la automatización de ventanas de herramientas y ventanas de documentos.

- [Utilice el modelo de automatización](#)

Proporciona dos ejemplos de código que muestran cómo un consumidor de automatización obtiene los objetos de automatización de proyecto iniciales.

- [Automatización para objetos Configuration y SelectedItem](#)

Proporciona información sobre la automatización de los objetos Configuration y SelectedItems.

Referencia

- [GetAutomationObject](#) Proporciona un ejemplo de código que muestra cómo participa un VSPackage en el modelo de objetos de automatización de DTE. Enumera los parámetros, los valores devueltos y los comentarios seleccionados.

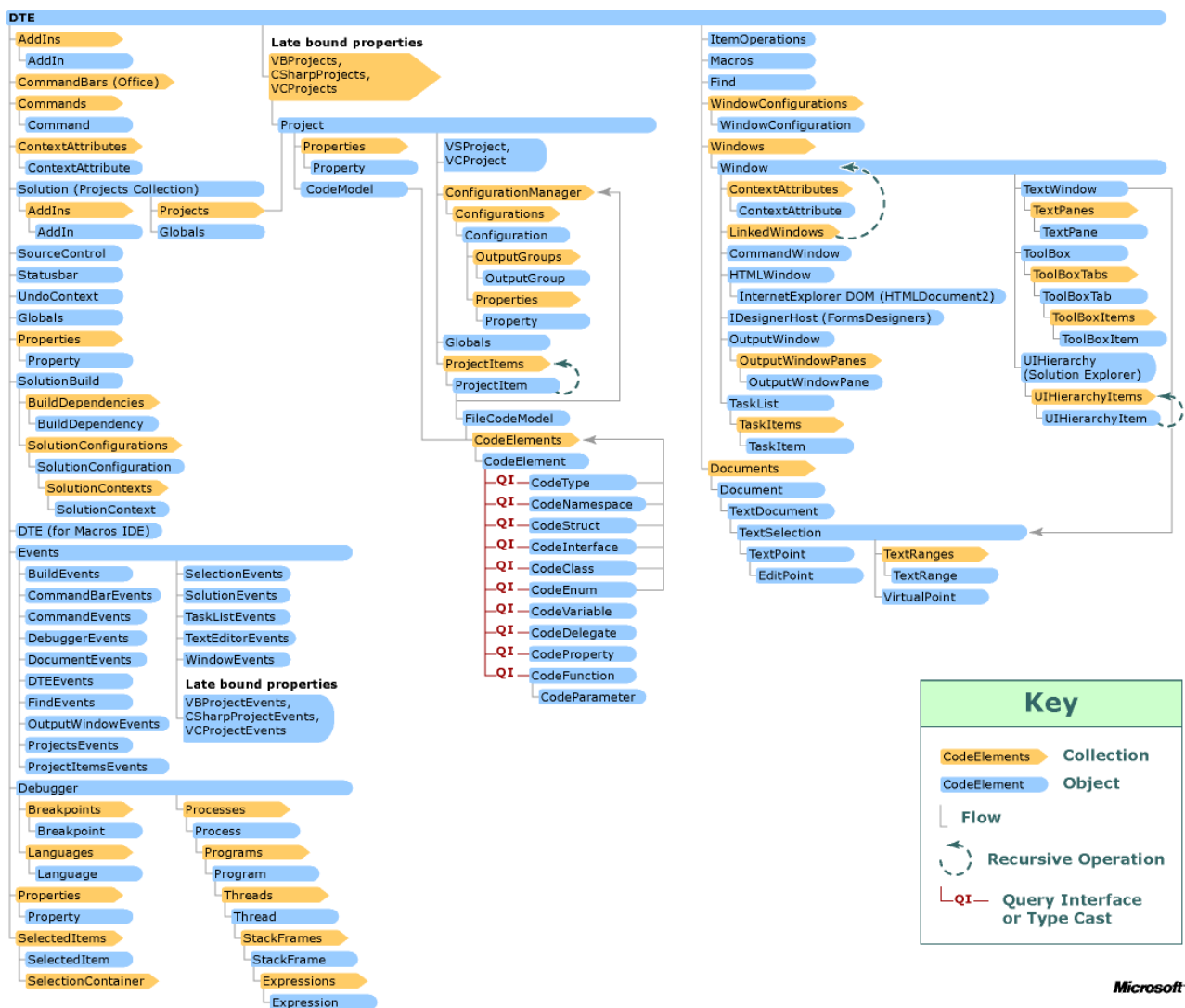
Visión general del modelo de automatización

06/04/2020 • 2 minutes to read • [Edit Online](#)

El modelo de automatización consta de un conjunto de objetos en el que puede escribir un complemento o extensión de Visual Studio. Un complemento es una aplicación que puede manipular el entorno de Visual Studio y automatizar tareas comunes. Una extensión de Visual Studio puede crear componentes personalizados de Visual Studio o agregar a la funcionalidad de componentes estándar, como el editor de texto.

Objetos en el modelo de automatización

El modelo de automatización consta de grupos relacionados de objetos que controlan las principales facetas del entorno común. En el diagrama siguiente se muestra el amplio conjunto de objetos de Visual Studio que componen el modelo de automatización.



Para obtener más información, vea [Extender el entorno de Visual Studio](#).

El entorno proporciona un modelo para diferentes áreas funcionales. Por ejemplo, hay un modelo de código para varios elementos que puede encontrar en el código. Hay un modelo de documento para varios elementos de documento. Un área, el área de proyecto, es de particular interés para los proveedores de VSPackage. Es probable que desee que los nuevos tipos de proyecto Visual C++ Visual Basic contribuyan al modelo de automatización de la misma manera que el modelo de automatización y contribuye a ellos. Ese proceso se describe en [Proporcionar automatización para VSPackages](#).

Lugares donde puede considerar la ampliación del modelo de automatización del entorno:

- proyecto
- Documento
- Código
- Compilar

Para obtener más información sobre la automatización, vea [Automatización y extensibilidad para Visual Studio](#). Este documento y los documentos a los que proporciona vínculos, le ayudan a tomar decisiones con respecto a cómo debe proporcionar automatización para el VSPackage.

Vea también

- [Cómo: Crear un complemento](#)

Provisión de automatización de VSPackages

06/04/2020 • 6 minutes to read • [Edit Online](#)

Hay dos formas principales de proporcionar automatización para los VSPackages: mediante la implementación de objetos específicos de VSPackage e mediante la implementación de objetos de automatización estándar. Generalmente, estos se utilizan juntos para extender el modelo de automatización del entorno.

Objetos específicos de VSPackage

Algunos lugares dentro del modelo de automatización requieren que proporcione objetos de automatización que son únicos para el VSPackage. Por ejemplo, los nuevos proyectos requieren objetos distintos que solo proporciona el VSPackage. Los nombres de estos objetos se introducen en `DTE` el registro y se obtienen mediante llamadas al objeto de entorno.

Objetos específicos de VSPackage también se pueden obtener cuando un consumidor de automatización utiliza el objeto proporcionado a través de la `Object` propiedad de un objeto estándar. Por ejemplo, `Window` el objeto `Object` estándar tiene una `Windows.Object` propiedad, conocida comúnmente como la propiedad. Cuando los consumidores llaman a la `Window.Object` en una ventana implementada en el VSPackage, se devuelve un objeto de automatización específico de su propio diseño.

Proyectos

VSPackages puede extender el modelo de automatización para nuevos tipos de proyecto a través de sus propios objetos específicos de VSPackage. El propósito principal de proporcionar nuevos objetos de automatización para `VCProject` el `VSPackage` es diferenciar los objetos de proyecto únicos de un objeto o un objeto. Esta diferenciación es útil cuando desea proporcionar una manera de seleccionar o iterar el tipo de proyecto aparte de otros tipos de proyecto, en caso de que aparezcan en paralelo en una solución. Para obtener más información, vea [Exponer objetos de proyecto](#).

Events

La arquitectura de eventos del entorno ofrece otro lugar para anexas sus propios objetos específicos de VSPackage. Por ejemplo, al crear sus propios objetos de evento únicos, puede ampliar el modelo de eventos del entorno para los proyectos. Es posible que desee proporcionar sus propios eventos cuando se agrega un nuevo elemento a su propio tipo de proyecto. Para obtener más información, consulte [Exposición de eventos](#).

Window (Objetos)

Windows puede devolver un objeto de automatización específico de VSPackage al entorno cuando se llama. Implementar un objeto que se `IVsExtensibleObject` `IExtensibleObject` deriva `IDispatch` de, o que entrega las propiedades, extendiendo el objeto de ventana en el que se encuentra. Por ejemplo, puede usar este enfoque para proporcionar automatización para un control incrustado en un marco de ventana. La semántica de este objeto y cualquier otro objeto que pueda extender son suyos para diseñar. Para obtener más información, consulte [Cómo: Proporcionar automatización para Windows](#).

Páginas de opciones en el menú Herramientas

Puede crear páginas para ampliar el modelo de automatización Herramientas y Opciones mediante la implementación de páginas y la adición de información al registro para crear sus propias opciones. A continuación, se puede llamar a las páginas a través del modelo de objetos de entorno como cualquier otra página de opciones. Si el diseño de la característica que va a agregar al entorno a través de VSPackages requiere páginas de opciones, a continuación, debe agregar la compatibilidad con automatización también. Para obtener más información, consulte [Compatibilidad con automatización para páginas de opciones](#).

Objetos de automatización estándar

Para ampliar la automatización de proyectos, también se `IDispatch` implementan objetos de automatización estándar (derivados de `ISolution`) que están al lado de los otros objetos de proyecto e implementan métodos y propiedades estándar. Entre los ejemplos de objetos estándar se incluyen los `Projects` `Project` objetos de proyecto que se insertan en la jerarquía de soluciones, como `ProjectItem` y `ProjectItems`. Cada nuevo tipo de proyecto debe implementar estos objetos (y posiblemente otros dependiendo de la semántica del proyecto).

En cierto sentido, estos objetos proporcionan la ventaja opuesta de los objetos de proyecto específicos de `VSPackage`. Los objetos de automatización estándar permiten que el proyecto se utilice de forma generalizada como cualquier otro proyecto que admita los mismos objetos. Por lo tanto, un complemento `Project` que `ProjectItem` se escribe en general y los objetos pueden funcionar con proyectos de cualquier tipo. Para obtener más información, vea [Modelado de proyectos](#).

Exponer objetos de proyecto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Los tipos de proyecto personalizados pueden proporcionar objetos de automatización para permitir el acceso al proyecto mediante interfaces de automatización. Se espera que cada tipo [Project](#) de proyecto proporcione [Solution](#) el objeto de automatización estándar desde el que se tiene acceso , que contiene una colección de todos los proyectos que están abiertos en el IDE. Se espera que cada elemento del [ProjectItem](#) proyecto se `Project.ProjectItems` exponga mediante un objeto al que se accede con `.` Además de estos objetos de automatización estándar, los proyectos pueden optar por ofrecer objetos de automatización específicos del proyecto.

Puede crear objetos de automatización de nivel raíz personalizados a `DTE.<customObjectName>` los `DTE.GetObject("<customObjectName>")` que pueda tener acceso a enlaces en tiempo de enlace desde el objeto DTE raíz mediante `.` Por ejemplo, Visual C++ crea una colección de proyectos específica del `DTE.VCProjects` `DTE.GetObject("VCProjects")` proyecto C++ denominada *VCProjects* a los que puede tener acceso mediante `.` También puede crear `Project.Object` un , que es único `Project.CodeModel` para el tipo de proyecto, un , `ProjectItem` que se `ProjectItem.Object` puede `ProjectItem.FileCodeModel` consultar para su objeto más derivado, y un , que expone y un archivo .

Es una convención común para que los proyectos expongan una colección de proyectos personalizada específica del proyecto. Por ejemplo, Visual C++ crea una colección de proyectos específica `DTE.VCProjects` `DTE.GetObject("VCProjects")` de C++ a la que puede acceder mediante `.` También puede crear `Project.Object` un , que es único `Project.CodeModel` para el tipo de proyecto, un , `ProjectItem` que se `ProjectItem.Object` puede `ProjectItem.FileCodeModel` consultar para su objeto más derivado, un , que expone y un archivo .

Para contribuir con un objeto específico de VSPackage para un proyecto

1. Agregue las claves adecuadas al archivo *.pkgdef* del VSPackage.

Por ejemplo, aquí está la configuración *.pkgdef* para el proyecto de lenguaje C++:

```
[${RootKey}\Packages\{F1C25864-3097-11D2-A5C5-00C04F7968B4}\Automation]
"VCProjects"=""

[${RootKey}\Packages\{F1C25864-3097-11D2-A5C5-00C04F7968B4}\AutomationEvents]
"VCProjectEngineEventsObject"=""
```

2. Implemente el código [GetAutomationObject](#) en el método, como en el ejemplo siguiente.

```

STDMETHODIMP CVsPackage::GetAutomationObject(
/* [in] */ LPCOLESTR      pszPropName,
/* [out] */ IDispatch **  ppIDispatch)
{
    ExpectedPtrRet(pszPropName);
    ExpectedPtrRet(ppIDispatch);
    *ppIDispatch = NULL;

    if (m_fZombie)
        return E_UNEXPECTED;

    if (_wcsicmp(pszPropName, g_wszAutomationProjects) == 0)
    {
        return GetAutomationProjects(ppIDispatch);
    }
    else if (_wcsicmp(pszPropName, g_wszAutomationProjectsEvents) == 0)
    {
        return CAutomationEvents::GetAutomationEvents(ppIDispatch);
    }
    else if (_wcsicmp(pszPropName, g_wszAutomationProjectItemsEvents) == 0)
    {
        return CAutomationEvents::GetAutomationEvents(ppIDispatch);
    }
    return E_INVALIDARG;
}

```

En el `g_wszAutomationProjects` código, es el nombre de la colección de proyectos. El `GetAutomationProjects` método crea un objeto `Projects` que implementa `IDispatch` la interfaz y devuelve un puntero al objeto que realiza la llamada, como se muestra en el ejemplo de código siguiente.

```

HRESULT CVsPackage::GetAutomationProjects(/* [out] */ IDispatch ** ppIDispatch)
{
    ExpectedPtrRet(ppIDispatch);
    *ppIDispatch = NULL;

    if (!m_srpAutomationProjects)
    {
        HRESULT hr = CACProjects::CreateInstance(&m_srpAutomationProjects);
        IfFailRet(hr);
        ExpectedExprRet(m_srpAutomationProjects != NULL);
    }
    return m_srpAutomationProjects.CopyTo(ppIDispatch);
}

```

Elija un nombre único para el objeto de automatización. Los conflictos de nombres son impredecibles y las colisiones hacen que un nombre de objeto en conflicto se descarte arbitrariamente si varios tipos de proyecto usan el mismo nombre. Debe incluir su nombre corporativo o algún aspecto único de su nombre de producto en el nombre del objeto de automatización.

El `Projects` objeto de colección personalizado es un punto de entrada de conveniencia para la parte restante del modelo de automatización de proyectos. El objeto de proyecto [Solution](#) también es accesible desde la colección de proyectos. Después de crear el código adecuado y las `Projects` entradas del Registro que proporcionan a los consumidores objetos de colección, la implementación debe proporcionar los objetos estándar restantes para el modelo de proyecto. Para obtener más información, consulte [Modelado de proyectos](#).

Vea también

- [GetAutomationObject](#)

Modelado de proyectos

06/04/2020 • 2 minutes to read • [Edit Online](#)

El siguiente paso para proporcionar automatización para el proyecto `Projects` `ProjectItems` es implementar los objetos de proyecto estándar: las colecciones; los `Project` `ProjectItem` objetos y; y los objetos restantes únicos para su implementación. Estos objetos estándar se definen en el archivo `Dteinternal.h`. En el ejemplo `BscPrj` se proporciona una implementación de los objetos estándar. Puede utilizar estas clases como modelos para crear sus propios objetos de proyecto estándar que se encuentran en paralelo con objetos de proyecto de otros tipos de proyecto.

Un consumidor de automatización presume `Solution` `<UniqueProjName>`) poder `ProjectItems` `n` llamar a `(` y `)` donde `n` es un número de índice para obtener un proyecto específico en la solución. Realizar esta llamada de automatización hace que el entorno llame `GetProperty` a la jerarquía de proyecto adecuada, pasando `VSITEMID_ROOT` como el parámetro `ItemID` y `VSHPROPID_ExtObject` como parámetro `VSHPROPID`.

`IVsHierarchy::GetProperty` devuelve `IDispatch` un puntero al objeto `Project` de automatización que proporciona la interfaz principal, que ha implementado.

A continuación se `IVsHierarchy::GetProperty` muestra la sintaxis de .

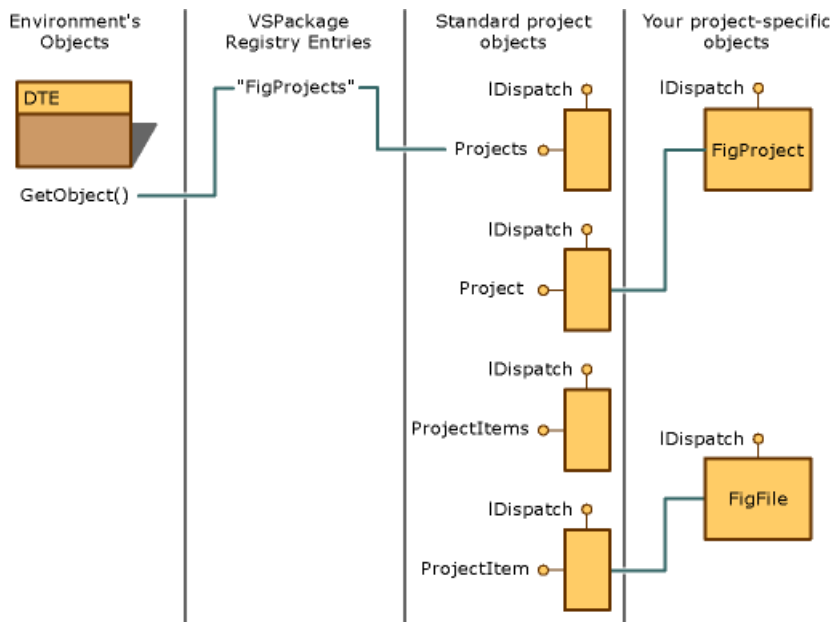
```
HRESULT GetProperty (
    VSITEMID itemid,
    VSHPROPID propid,
    VARIANT *pvar
);
```

Los proyectos admiten el anidamiento y usan colecciones para crear grupos de elementos de proyecto. La jerarquía se ve así.

```
Projects
|- Project
    |- ProjectItems (a collection of ProjectItem)
        |- ProjectItem (single object) or ProjectItems (another collection)
```

Anidar significa `ProjectItem` que un `ProjectItems` objeto puede ser `ProjectItems` colección al mismo tiempo porque una colección puede contener los objetos anidados. El proyecto básico ejemplo no muestra este anidamiento. Al implementar `Project` el objeto, participa en la estructura de árbol que caracteriza el diseño del modelo de automatización general.

La automatización del proyecto sigue la ruta de acceso en el diagrama siguiente.



de Visual Studio Automatización de

proyectos

Si no implementa `Project` un objeto, el entorno `Project` seguirá devolviendo un objeto genérico que contenga solo el nombre del proyecto.

Vea también

- [Projects](#)
- [ProjectItem](#)
- [ProjectItems](#)

Exponer eventos en el SDK de Visual Studio

06/04/2020 • 6 minutes to read • [Edit Online](#)

Visual Studio permite originar eventos mediante la automatización. Se recomienda obtener eventos para proyectos y elementos de proyecto.

Los consumidores de automatización `Events` recuperan `GetObject` los eventos `GetObject("EventObjectName")` del objeto o (por ejemplo,). El entorno `IDispatch::Invoke` llama `DISPATCH_METHOD` mediante `DISPATCH_PROPERTYGET` el uso de los indicadores o para devolver un evento.

El siguiente proceso explica cómo se devuelven los eventos específicos de `VSPackage`.

1. Se inicia el entorno.
2. Lee del registro todos los nombres de valor en las claves **Automation**, **AutomationEventsy AutomationProperties** de todos los `VSPackages` y almacena esos nombres en una tabla.
3. Un consumidor de automatización `DTE.Events.AutomationProjectsEvents` llama, en este ejemplo, o `DTE.Events.AutomationProjectItemsEvents`.
4. El entorno busca el parámetro de cadena en la tabla y carga el `VSPackage` correspondiente.
5. El entorno `GetAutomationObject` llama al método mediante el nombre pasado en la llamada; en este `AutomationProjectsEvents` ejemplo, o `AutomationProjectItemsEvents`.
6. El `VSPackage` crea un objeto raíz `get_AutomationProjectsEvents` `get_AutomationProjectItemEvents` que tiene métodos como y, a continuación, devuelve un puntero `IDispatch` al objeto.
7. El entorno llama al método adecuado en función del nombre pasado a la llamada de automatización.
8. El `get_` método crea otro objeto de evento basado `IConnectionPointContainer` en `IConnectionPoint` `IDispatch` que `IDispatchpointer` implementa la interfaz y la interfaz y devuelve un al objeto.

Para exponer un evento mediante la `GetAutomationObject` automatización, debe responder y observar las cadenas que agregue al registro. En el ejemplo De proyecto básico, las cadenas son *BscProjectsEvents* y *BscProjectItemsEvents*.

Entradas de registro del ejemplo de proyecto básico

En esta sección se muestra dónde agregar valores de eventos de automatización al registro.

[HKEY_LOCAL_MACHINE\SOFTWARE, Microsoft, VisualStudio, 8,0,\ >Paquetes <PkgGUID, AutomationEvents]

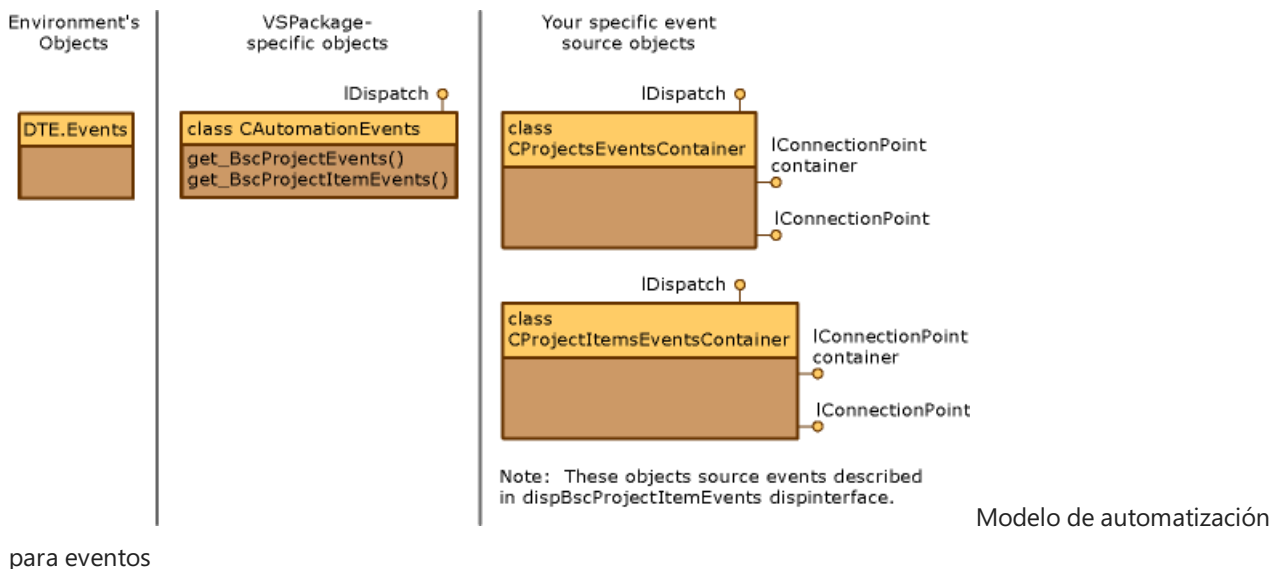
AutomationProjectEvents: devuelve `AutomationProjectEvents` el objeto.

AutomationProjectItemEvents: devuelve `AutomationProjectItemsEvents` el objeto.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
Predeterminado (no)	REG_SZ	No utilizado	Sin usar. Puede utilizar el campo de datos para la documentación.

NOMBRE	TIPO	INTERVALO	DESCRIPCIÓN
<i>AutomationProjectsEvents</i>	REG_SZ	Nombre del objeto de evento.	Solo el nombre de clave es relevante. Puede utilizar el campo de datos para la documentación. Este ejemplo procede del ejemplo de proyecto básico.
<i>AutomationProjectItemEvents</i>	REG_SZ	Nombre del objeto de evento	Solo el nombre de clave es relevante. Puede utilizar el campo de datos para la documentación. Este ejemplo procede del ejemplo de proyecto básico.

Cuando un consumidor de automatización solicita cualquiera de los objetos de evento, cree un objeto raíz que tenga métodos para cualquier evento que admita el VSPackage. El entorno llama `get_` al método adecuado en este objeto. Por ejemplo, `DTE.Events.AutomationProjectsEvents` si se `get_AutomationProjectsEvents` llama, se invoca el método en el objeto raíz.



La `CProjectEventsContainer` clase representa el objeto de origen `CProjectItemsEventsContainer` para `BscProjectsEvents` el objeto de origen para `BscProjectItemsEvents`.

En la mayoría de los casos, debe devolver un nuevo objeto para cada solicitud de evento porque la mayoría de los objetos de evento toman un objeto de filtro. Cuando se desencadena el evento, compruebe este filtro para comprobar que se llama al controlador de eventos.

AutomationEvents.h y *AutomationEvents.cpp* contienen declaraciones e implementaciones de las clases de la tabla siguiente.

CLASE	DESCRIPCIÓN
<code>CAutomationEvents</code>	Implementa un objeto raíz de evento, recuperado del <code>DTE.Events</code> objeto.
<code>CProjectsEventsContainer</code> y <code>CProjectItemsEventsContainer</code>	Implemente los objetos de origen de eventos que desencadenan los eventos correspondientes.

En el ejemplo de código siguiente se muestra cómo responder a una solicitud de un objeto de evento.

```
STDMETHODIMP CVsPackage::GetAutomationObject(
    /* [in] */ LPCOLESTR pszPropName,
    /* [out] */ IDispatch ** ppIDispatch)
{
    ExpectedPtrRet(ppIDispatch);
    *ppIDispatch = NULL;

    if (_wcsicmp(pszPropName, g_wszAutomationProjects) == 0)
        //Is the requested name our Projects object?
    {
        return GetAutomationProjects(ppIDispatch);
        // Gets our Projects object.
    }
    else if (_wcsicmp(pszPropName, g_wszAutomationProjectsEvents) == 0)
        //Is the requested name our ProjectsEvents object?
    {
        return CAutomationEvents::GetAutomationEvents(ppIDispatch);
        // Gets our ProjectEvents object.
    }
    else if (_wcsicmp(pszPropName, g_wszAutomationProjectItemsEvents) == 0) //Is the requested name our
    ProjectsItemsEvents object?
    {
        return CAutomationEvents::GetAutomationEvents(ppIDispatch);
        // Gets our ProjectItemsEvents object.
    }
    return E_INVALIDARG;
}
```

En el código `g_wszAutomationProjects` anterior, es el nombre de `g_wszAutomationProjectsEvents` la colección de proyectos (*FigProjects*), (*FigProjectsEvents*) y `g_wszAutomationProjectItemsEvents` (*FigProjectItemEvents*) son los nombres de eventos de proyecto y eventos de elementos de proyecto que se originan desde la implementación de VSPackage.

Los objetos de evento se recuperan de la misma ubicación central, el `DTE.Events` objeto. De esta manera, todos los objetos de evento se agrupan para que un usuario final no tenga que examinar todo el modelo de objetos para buscar un evento específico. Esto también le permite proporcionar sus objetos VSPackage específicos, en lugar de requerir que implemente su propio código para eventos de todo el sistema. Sin embargo, para el usuario final, `ProjectItem` que debe encontrar un evento para la interfaz, no está inmediatamente claro desde dónde se recupera ese objeto de evento.

Vea también

- [GetAutomationObject](#)

Soporte de automatización para páginas de opciones

06/04/2020 • 2 minutes to read • [Edit Online](#)

VSPackages puede proporcionar **opciones** personalizadas cuadros de diálogo Visual Studio para el **herramientas** menú (**páginas** de opciones de herramientas) en y puede ponerlos a disposición del modelo de automatización.

Opciones de herramientas (páginas)

Para crear un **opciones** de herramientas página, un VSPackage debe proporcionar una implementación [GetPropertyPage](#) de control de usuario devuelto al entorno a través de la implementación de VSPackage del método. (O, para el código [GetPropertyPage](#) administrado, el método.)

Es opcional, pero muy recomendable, permitir el acceso a esta nueva página a través del modelo de automatización. Puede hacerlo con los siguientes pasos:

1. Extienda [Properties\[String, String\]](#) el objeto a través de la implementación de un IDispatch-objeto derivado.
2. Devuelve una implementación del [GetAutomationObject](#) método (o para el código administrado el [GetAutomationObject](#) método) al objeto derivado de IDispatch.
3. Cuando un consumidor [Properties\[String, String\]](#) de automatización llama al método en [GetAutomationObject](#) una página de propiedades **Option** personalizada, el entorno usa el método para obtener la implementación de automatización de una página **Opciones** de herramientas personalizada.
4. A continuación, se usa el objeto [Property](#) de [Properties\[String, String\]](#) automatización del VSPackage para proporcionar cada uno devuelto por .

Para obtener un ejemplo de implementación de una página **de opciones** de herramientas personalizadas, vea [ejemplos de VSSDK](#).

Vea también

- [Exponer objetos de proyecto](#)

Provisión de automatización para código

06/04/2020 • 2 minutes to read • [Edit Online](#)

No es necesario crear un modelo de automatización para el código. El SDK de entorno no proporciona un ejemplo para hacerlo. Para obtener información sobre [CodeModel](#) los modelos de código, consulte el objeto.

Para implementar un modelo de código, debe implementar las interfaces que determine la estructura de datos interna. Los objetos deben derivarse de la `IDispatch` clase.

Los objetos que [CodeModel](#) [FileCodeModel](#) se extienden [Project](#) y , están disponibles en el objeto y tienen el siguiente aspecto:

- [CodeModel](#)
- [FileCodeModel](#)

Puede optar por implementar `CodeModel` solo `FileCodeModel` la interfaz o `Project` la [ProjectItem](#) del objeto que devuelve de su y objetos. Proporcione cualquier funcionalidad de esta interfaz que sea adecuada para el sistema de proyectos.

Si desea agregar características, como métodos o propiedades, que `CodeModel` `FileCodeModel` no están disponibles en el estándar y las interfaces, cree su propia interfaz que herede del estándar. Asegúrese de documentarlo con su sistema de proyecto para que los usuarios finales sepan que lo buscan. Devuelve la interfaz estándar, pero `QueryInterface` el usuario puede llamar al método o convertir a la interfaz si se sabe que existe.

Vea también

- [Información general del modelo de automatización](#)

Cómo: Proporcionar automatización para Windows

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede proporcionar automatización para ventanas de documentos y herramientas. Proporcionar automatización es recomendable siempre que desee que los objetos de automatización estén disponibles en una ventana y el entorno no proporcione ya un objeto de automatización listo, como lo hace con una lista de tareas.

Automatización de ventanas de herramientas

El entorno proporciona automatización en una [Window](#) ventana de herramientas devolviendo un objeto estándar como se explica en el procedimiento siguiente:

1. Llame [GetProperty](#) al método a través del entorno con `_VSFPROPID.VSFPROPID_ExtWindowObject` como parámetro `Window` para obtener el objeto.
2. Cuando un llamador solicita un objeto de automatización [Object](#) específico de `QueryInterface` `IExtensibleObject` `VSPackage` para la ventana de herramientas a través de , el entorno llama a , [IVsExtensibleObject](#), o las `IDispatch` interfaces. Ambos `IExtensibleObject` `IVsExtensibleObject` y [GetAutomationObject](#) proporcionar un método.
3. Cuando el entorno, `GetAutomationObject` a `NULL` continuación, llama al método que pasa , responder pasando de vuelta el objeto específico de `VSPackage`.
4. Si `QueryInterface` llama `IExtensibleObject` `IVsExtensibleObject` y produce un `QueryInterface` error, el entorno llama a `IDispatch` .

Automatización de ventanas de documentos

Un [Document](#) objeto estándar también está disponible en el entorno, aunque [Document](#) un `IExtensibleObject` editor puede `GetAutomationObject` tener su propia implementación del objeto mediante la implementación de la interfaz y la respuesta a .

Además, un editor puede proporcionar un objeto de [Object](#) automatización específico `IVsExtensibleObject` de `IExtensibleObject` `VSPackage`, recuperado a través del método, mediante la implementación de las interfaces o. Los [ejemplos de VSSDK](#) aportan un objeto de automatización específico del documento RTF.

Vea también

- [IVsExtensibleObject](#)

Uso del modelo de automatización

06/04/2020 • 2 minutes to read • [Edit Online](#)

Después de conectar el VSPackage a la automatización, [GetObject](#) puede obtener [_DTE](#) las propiedades y métodos mediante una llamada al método en el objeto, pasando una cadena que representa el objeto que desea recuperar.

Obtención de objetos de proyecto

Los siguientes son dos ejemplos de código que muestran cómo un consumidor de automatización obtiene los objetos de automatización del proyecto. Para obtener información sobre cómo obtener el objeto DTE, consulte [Cómo: Obtener referencias a los objetos DTE y DTE2](#).

```
Sub DoAutomation()  
    Dim MyProjects As Projects  
    MyProjects = DTE.GetObject("AcmeProject")  
End Sub
```

```
void DoAutomation(void)  
{  
    CComQIPtr<Projects> pMyPkg; // Use an IDispatch-derived object type.  
    pMyPkg = pDTE->GetObject("AcmeProjects");  
  
    // The '=' performs a Query Interface.  
    // Assumes pDTE is already available as a global.  
    // Use pMyPkg to access your projects object's properties and methods.  
}
```

En este punto, puede usar los objetos de proyecto estándar que forman parte de un VSPackage específico para desplazarse por el modelo de jerarquía.

En el ejemplo de código siguiente se muestra cómo obtener un objeto personalizado que es una propiedad de un tipo de proyecto personalizado.:

```
Dim MyPrj As Project  
Dim MyPrjItem As ProjectItem  
Dim objMyObject as MyExtendedObject  
  
MyPrj = MyProjects.Item(1) 'use the Projects collection to get a project  
objMyObject = MyPrj.Object 'You call .Object to get to special Project  
                           'implementation  
objMyObject.MySpecialMethodOrProperty
```

El código siguiente enumera los nombres de Visual Studio todas las propiedades en el entorno **General** opción en el **herramientas** menú:

```
dim objDTE
dim objEnv
set objDTE = CreateObject("VisualStudio.DTE")
set objEnv = objDTE.Properties("Environment", "General")
for each obj in ObjEnv
MsgBox obj.Name
Next
```

Vea también

- [GetObject](#)

Automatización para objetos Configuration y SelectedItem

06/04/2020 • 2 minutes to read • [Edit Online](#)

Puede automatizar los procesos de compilación y elemento seleccionado en Visual Studio.

Automatización para compilaciones

La compilación o configuración tiene un modelo [IVsCfgProvider](#) de automatización que se proporciona al implementar . Para obtener más información, vea [Descripción de las configuraciones de compilación](#).

Si crea un VSPackage y desea controlar las opciones de configuración, debe usar el modelo de automatización.

Automatización para SelectedItem

No es necesario proporcionar una `SelectedItem` implementación para el objeto porque Visual Studio contiene una implementación estándar. Sin embargo, `SelectedItem` puede implementar el objeto si lo prefiere. Debe implementar un objeto `SelectedItem` que contenga la interfaz y [GetPropertyPage](#) devolver `VSITEMID` una respuesta a una llamada al método con establecido en `__VSHPROPID.VSHPROPID_ExtSelectedItem`.

Vea también

- [GetPropertyPage](#)
- [Contribuir al modelo de automatización](#)
- [Descripción de las configuraciones de compilación](#)

Introducción a soluciones

06/04/2020 • 4 minutes to read • [Edit Online](#)

Una solución es una agrupación de uno o varios proyectos que trabajan juntos para crear una aplicación. La información de proyecto y estado de la solución se almacena en dos archivos de solución diferentes. El archivo de [solución \(.sln\)](#) está basado en texto y se puede colocar bajo control de código fuente y compartirse entre usuarios. El archivo de opción de [usuario de solución \(.suo\)](#) es binario. Como resultado, el archivo .suo no se puede colocar bajo control de código fuente y contiene información específica del usuario.

Cualquier VSPackage puede escribir en cualquier tipo de archivo de solución. Debido a la naturaleza de los archivos, hay dos interfaces diferentes implementadas para escribir en ellos. La [IVsPersistSolutionProps](#) interfaz escribe información de texto en [IVsPersistSolutionOpts](#) el archivo .sln y la interfaz escribe secuencias binarias en el archivo .suo.

NOTE

Un proyecto no tiene que escribir explícitamente una entrada para sí mismo en el archivo de solución; el entorno lo controla para el proyecto. Por lo tanto, a menos que desee agregar contenido adicional específicamente al archivo de solución, no es necesario registrar el VSPackage de esta manera.

Cada VSPackage que admite la persistencia [IVsSolutionPersistence](#) de la solución usa tres interfaces, la [IVsPersistSolutionProps](#) interfaz, que se implementa por el entorno y llama por el VSPackage y [IVsPersistSolutionOpts](#), que se implementan por el VSPackage. La [IVsPersistSolutionOpts](#) interfaz solo debe implementarse si el VSPackage debe escribir información privada en el archivo .suo.

Cuando se abre una solución, se lleva a cabo el siguiente proceso.

1. El entorno lee la solución.
2. Si el entorno [CLSID](#) encuentra un , carga el VSPackage correspondiente.
3. Si se carga un VSPackage, el entorno llama [QueryInterface](#) a la [IVsPackage](#) interfaz para la interfaz que requiere el VSPackage.
 - Al leer desde un archivo .sln, el entorno requiere [QueryInterface](#) [IVsPersistSolutionProps](#) .
 - Al leer desde un archivo .suo, el entorno requiere [QueryInterface](#) [IVsPersistSolutionOpts](#) .

La información específica relacionada con el uso de estos archivos se puede encontrar en la [solución \(.Sln\) Opciones de usuario de archivo y solución \(.Suo\) Archivo](#).

NOTE

Si desea crear una nueva configuración de solución que consta de las configuraciones de dos proyectos y excluir un tercero de la compilación, debe usar la interfaz de usuario de páginas de propiedades o la automatización. No puede cambiar las configuraciones del administrador de compilación de soluciones y sus [SolutionBuild](#) propiedades directamente, pero puede manipular el administrador de compilación de soluciones mediante la clase de DTE en el modelo de automatización. Para obtener más información acerca de la configuración de soluciones, vea Configuración de la [solución](#).

Vea también

- [IVsPackage](#)

- [IVsPersistSolutionOpts](#)
- [IVsPersistSolutionProps](#)
- [IVsSolutionPersistence](#)

Solución (.sln) archivo

06/04/2020 • 12 minutes to read • [Edit Online](#)

Una solución es una estructura para organizar proyectos en Visual Studio. La solución mantiene la información de estado de los proyectos en dos archivos:

- Archivo .sln (basado en texto, compartido)
- Archivo .suo (opciones de solución binarias específicas del usuario)

Para obtener más información acerca de los archivos .suo, vea Opciones de usuario de la [solución \(. Suo\) Archivo](#).

Si el VSPackage se carga como resultado de que se hace [ReadSolutionProps](#) referencia en el archivo .sln, el entorno llama para leer en el archivo .sln.

El archivo .sln contiene información basada en texto que el entorno utiliza para buscar y cargar los parámetros nombre-valor para los datos persistentes y el proyecto VSPackages al que hace referencia. Cuando un usuario abre una solución, `Project` el `postSolution` entorno recorre el `preSolution` archivo, e información del archivo .sln para cargar la solución, los proyectos dentro de la solución y cualquier información persistente asociada a la solución.

El archivo de cada proyecto contiene información adicional leída por el entorno para rellenar la jerarquía con los elementos de ese proyecto. El proyecto controla la persistencia de los datos de jerarquía. Los datos no se almacenan normalmente en el archivo .sln, aunque puede escribir intencionalmente información del proyecto en el archivo .sln si decide hacerlo. Para obtener más información acerca de la persistencia, vea [Persistencia](#) del proyecto y [Abrir y guardar elementos](#) de proyecto .

Encabezado del archivo

El encabezado de un archivo .sln tiene este aspecto:

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 15
VisualStudioVersion = 15.0.26730.15
MinimumVisualStudioVersion = 10.0.40219.1
```

Definiciones

Microsoft Visual Studio Solution File, Format Version 12.00

Encabezado estándar que define la versión del formato de archivo.

Visual Studio 15

La versión principal de Visual Studio que (más recientemente) guardó este archivo de solución. Esta información controla el número de versión en el icono de la solución.

VisualStudioVersion = 15.0.26730.15

La versión completa de Visual Studio que (más recientemente) guardó el archivo de solución. Si el archivo de solución se guarda mediante una versión más reciente de Visual Studio que tiene la misma versión principal, este valor no se actualiza para disminuir la renovación en los archivos de solución.

MinimumVisualStudioVersion = 10.0.40219.1

La versión mínima (más antigua) de Visual Studio que puede abrir este archivo de solución.


```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio Version 16
VisualStudioVersion = 16.0.28701.123
MinimumVisualStudioVersion = 10.0.40219.1
```

Definiciones

Microsoft Visual Studio Solution File, Format Version 12.00

Encabezado estándar que define la versión del formato de archivo.

Visual Studio Version 16

La versión principal de Visual Studio que (más recientemente) guardó este archivo de solución. Esta información controla el número de versión en el icono de la solución.

VisualStudioVersion = 16.0.28701.123

La versión completa de Visual Studio que (más recientemente) guardó el archivo de solución. Si el archivo de solución se guarda mediante una versión más reciente de Visual Studio que tiene la misma versión principal, este valor no se actualiza para disminuir la renovación en el archivo.

MinimumVisualStudioVersion = 10.0.40219.1

La versión mínima (más antigua) de Visual Studio que puede abrir este archivo de solución.

Cuerpo del archivo

El cuerpo de un archivo .sln `GlobalSection` consta de varias secciones etiquetadas, como esta:

```
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "Project1", "Project1.vbproj", "{8CDD8387-B905-44A8-B5D5-07BB50E05BEA}"
EndProject
Global
    GlobalSection(SolutionNotes) = postSolution
EndGlobalSection
    GlobalSection(SolutionConfiguration) = preSolution
        ConfigName.0 = Debug
        ConfigName.1 = Release
    EndGlobalSection
    GlobalSection(ProjectDependencies) = postSolution
EndGlobalSection
    GlobalSection(ProjectConfiguration) = postSolution
        {8CDD8387-B905-44A8-B5D5-07BB50E05BEA}.Debug.ActiveCfg = Debug|x86
        {8CDD8387-B905-44A8-B5D5-07BB50E05BEA}.Debug.Build.0 = Debug|x86
        {8CDD8387-B905-44A8-B5D5-07BB50E05BEA}.Release.ActiveCfg = Release|x86
        {8CDD8387-B905-44A8-B5D5-07BB50E05BEA}.Release.Build.0 = Release|x86
    EndGlobalSection
    GlobalSection(ExtensibilityGlobals) = postSolution
EndGlobalSection
    GlobalSection(ExtensibilityAddIns) = postSolution
EndGlobalSection
EndGlobal
```

Para cargar una solución, el entorno realiza la siguiente secuencia de tareas:

1. El entorno lee la sección Global del archivo .sln y procesa todas las secciones marcadas `preSolution`. En este archivo de ejemplo, hay una instrucción de este tipo:

```
GlobalSection(SolutionConfiguration) = preSolution
    ConfigName.0 = Debug
    ConfigName.1 = Release
```

Cuando el entorno `GlobalSection('name')` lee la etiqueta, asigna el nombre a un `VSPackage` mediante el registro. El nombre de la clave debe existir en el registro en `[HKLM<raíz del Registro de ID de aplicación]`. El valor predeterminado de las claves es el GUID de paquete (REG_SZ) del `VSPackage` que escribió las entradas.

2. El entorno carga el `QueryInterface` `VSPackage`, llama `IVsPersistSolutionProps` al `VSPackage` `ReadSolutionProps` para la interfaz y llama al método con los datos de la sección para que el `VSPackage` pueda almacenar los datos. El entorno repite este `preSolution` proceso para cada sección.
3. El entorno recorre en iteración los bloques de persistencia del proyecto. En este caso, hay un proyecto.

```
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "Project1",  
    "Project1.vbproj", "{8CDD8387-B905-44A8-B5D5-07BB50E05BEA}"  
EndProject
```

Esta instrucción contiene el GUID de proyecto único y el GUID de tipo de proyecto. El entorno usa esta información para buscar el archivo de proyecto o los archivos que pertenecen a la solución y el `VSPackage` necesario para cada proyecto. El GUID del `IVsProjectFactory` proyecto se pasa a cargar el `VSPackage` específico relacionado con el proyecto, a continuación, el proyecto se carga mediante el `VSPackage`. En este caso, el `VSPackage` que se carga para este proyecto es Visual Basic.

Cada proyecto puede conservar un identificador de instancia de proyecto único para que se pueda tener acceso a él según sea necesario para otros proyectos de la solución. Idealmente, si la solución y los proyectos están bajo control de código fuente, la ruta de acceso al proyecto debe ser relativa a la ruta de acceso a la solución. Cuando se carga la solución por primera vez, los archivos de proyecto no pueden estar en el equipo del usuario. Al tener el archivo de proyecto almacenado en el servidor en relación con el archivo de solución, es relativamente sencillo encontrar el archivo de proyecto y copiarlo en el equipo del usuario. A continuación, copia y carga el resto de los archivos necesarios para el proyecto.

4. En función de la información contenida en la sección de proyecto del archivo `.sln`, el entorno carga cada archivo de proyecto. A continuación, el propio proyecto es responsable de rellenar la jerarquía del proyecto y cargar los proyectos anidados.
5. Después de procesar todas las secciones del archivo `.sln`, la solución se muestra en el Explorador de soluciones y está lista para su modificación por parte del usuario.

Si cualquier `VSPackage` que implementa un proyecto en `OnProjectLoadFailure` la solución no se puede cargar, se llama al método y todos los demás proyectos de la solución tienen la oportunidad de omitir los cambios que podría haber realizado durante la carga. Si se producen errores de análisis, se conserva tanta información como sea posible con los archivos de solución y el entorno muestra un cuadro de diálogo que advierte al usuario de que la solución está dañada.

Cuando se guarda o se `QuerySaveSolutionProps` cierra la solución, se llama al método y se pasa a la jerarquía para ver si se han realizado cambios en la solución que se deben escribir en el archivo `.sln`. Un valor nulo, `QuerySaveSolutionProps` pasado `VSQUERYSAVESLNPORPS`, indica que la información se conserva para la solución. Si el valor no es null, la información persistente es para `IVsHierarchy` un proyecto específico, determinado por el puntero a la interfaz.

Si hay información que se `IVsSolutionPersistence` va a guardar, `SaveSolutionProps` se llama a la interfaz con un puntero al método. A `WriteSolutionProps` continuación, el entorno llama al método para `IPropertyBag` recuperar los pares nombre-valor de la interfaz y escribir la información en el archivo `.sln`.

`SaveSolutionProps` y `WriteSolutionProps` el entorno llama recursivamente a los objetos `IPropertyBag` para recuperar la información que se guardará de la interfaz hasta que se hayan introducido todos los cambios en el archivo `.sln`. De este modo, puede asegurarse de que la información se conservará con la solución y estará disponible la próxima vez que se abra la solución.

Cada VSPackage cargado se enumera para ver si tiene algo que guardar en el archivo .sln. Es sólo en el momento de la carga que se consultan las claves del registro. El entorno conoce todos los paquetes cargados porque están en memoria en el momento en que se guarda la solución.

Solo el archivo .sln contiene `preSolution` `postSolution` entradas en las secciones y. No hay secciones similares en el archivo .suo, ya que la solución necesita esta información para cargar correctamente. El archivo .suo contiene opciones específicas del usuario, como notas privadas, que no están diseñadas para compartirse ni colocarse bajo control de código fuente.

Vea también

- [IVsPersistSolutionProps](#)
- [Archivo de opciones de usuario de la solución \(.Suo\)](#)
- [Soluciones](#)

Archivo de opciones de usuario de la solución (.Suo)

06/04/2020 • 3 minutes to read • [Edit Online](#)

El archivo de opciones de usuario de la solución (.suo) contiene opciones de solución por usuario. Este archivo no se debe proteger en el control de código fuente.

El archivo de opciones de usuario de la solución (.suo) es un archivo de almacenamiento estructurado o compuesto almacenado en formato binario. Guardar información de usuario en secuencias con el nombre de la secuencia es la clave que se usará para identificar la información en el archivo .suo. El archivo de opciones de usuario de la solución se usa para almacenar la configuración de preferencias del usuario y se crea automáticamente cuando Visual Studio guarda una solución.

Cuando el entorno abre un archivo .suo, enumera todos los VSPackages cargados actualmente. Si un VSPackage [IVsPersistSolutionOpts](#) implementa la interfaz, [LoadUserOptions](#) el entorno llama al método en el VSPackage pidiéndole que cargue todos sus datos desde el archivo .suo.

Es responsabilidad del VSPackage saber qué secuencias podría haber escrito en el archivo .suo. Para cada secuencia que escribió, el VSPackage [LoadPackageUserOpts](#) llama al entorno a través de [LoadPackageUserOpts](#) para cargar una secuencia determinada que se identifica mediante la clave, que es el nombre de la secuencia. A continuación, el entorno vuelve a llamar al VSPackage para `IStream` leer esa secuencia concreta pasando el nombre de la secuencia y un puntero al [LoadPackageUserOpts](#) método.

En ese momento, se `LoadUserOptions` realiza otra llamada para ver si hay otra sección del archivo .suo que se debe leer. Este proceso continúa hasta que el entorno ha leído y procesado todas las secuencias de datos del archivo .suo.

Cuando se guarda o cierra la [SavePackageSolutionProps](#) solución, el [SaveUserOptions](#) entorno llama al método con un puntero al método. Un `IStream` que contiene la información binaria [WriteUserOptions](#) que se va a guardar se pasa al `SaveUserOptions` método, que luego escribe la información en el archivo .suo y llama al método de nuevo para ver si hay otra secuencia de información para escribir en el archivo .suo.

Estos dos `SaveUserOptions` métodos, y `WriteUserOptions`, se llaman recursivamente para cada secuencia de información que `IVsSolutionPersistence` se va a guardar en el archivo .suo, pasando el puntero a `IStream`. Se llaman recursivamente para permitir la escritura de varias secuencias en el archivo .suo. De este modo, la información del usuario se conserva con la solución y se garantiza que estará allí la próxima vez que se abra la solución.

Vea también

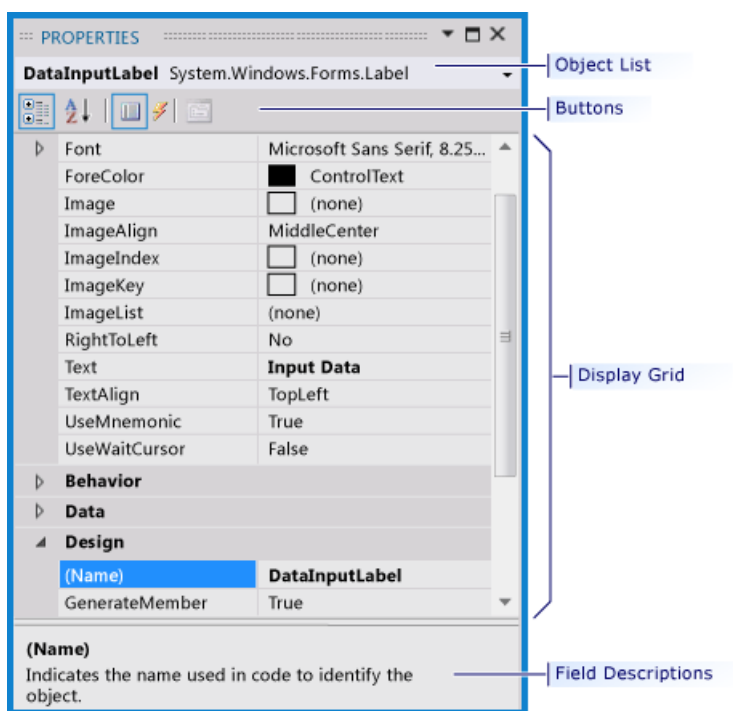
- [IVsPersistSolutionOpts](#)
- [Soluciones](#)

Extender propiedades

06/04/2020 • 4 minutes to read • [Edit Online](#)

La Visual Studio ventana **Propiedades** es un explorador de propiedades universal Visual Studio para componentes COM y COM+ y admite todos los productos. La **Properties** ventana Propiedades `TypeInfo` funciona con información de tipo y metadatos COM+ para enumerar las propiedades en tiempo de diseño del objeto seleccionado actualmente en cualquier otra ventana del entorno de desarrollo integrado (IDE).

La ventana **Propiedades**, que se puede abrir presionando **F4** en el teclado o seleccionando **Ventana Propiedades** en el menú **Ver**, se utiliza para ver y editar propiedades y eventos en tiempo de diseño independientes de la configuración de los objetos seleccionados. Las propiedades dependientes de la configuración, asociadas a soluciones y proyectos, se muestran en [las páginas de propiedades](#). Para obtener más información, [administrar opciones](#) de configuración .



Ventana Propiedades

Esta sección proporciona información detallada que se relaciona con las áreas individuales de la ventana **Propiedades** y las interfaces que debe implementar y llamar para rellenar la ventana.

En esta sección

- [Descripción general de la ventana Propiedades](#)

Explica el propósito de la ventana **Propiedades** en relación con la ventana de herramientas y la ventana del documento.

- [Directiva de plantilla y la ventana Propiedades](#)

Describe cómo un proyecto está contenido en un proyecto de plantilla de empresa y cómo ese proyecto de plantilla de empresa puede aplicar la directiva.

- [Propiedades campos de ventana e interfaces](#)

Explica la base de la selección que determina qué información se muestra en la ventana **Propiedades**.

- [Lista de objetos de la ventana Propiedades](#)

Describe el propósito de la lista de objetos de ventana **Propiedades**, que describe cómo, cuando un objeto diferente de esta lista desencadena una llamada, se informa al entorno de que se ha seleccionado un nuevo objeto.

- [Botones de la ventana Propiedades](#)

Explica el propósito de los cuatro botones predeterminados que se muestran en la barra de herramientas de la ventana **Propiedades**.

- [Cuadrícula de visualización de propiedades](#)

Explica dónde se encuentran los nombres de propiedad y los campos de valores de propiedad en la cuadrícula.

Secciones relacionadas

- [Tipos de proyecto](#)

Describe los proyectos como los Visual Studio bloques de creación del IDE.

- [Compilar y generar](#)

Describe cómo puede usar Visual Studio la plataforma para probar y depurar aplicaciones continuamente a medida que las compila.

- [IDispatch](#)

Describe la `IDispatch` interfaz, que se diseñó primero para admitir la automatización, proporcionando un mecanismo enlazado en tiempo de ejecución de que se va a tiempo para tener acceso y recuperar información sobre los métodos y propiedades de un objeto.

- [Administración de la configuración de la aplicación \(.NET\)](#)

Proporciona información general sobre la configuración de la aplicación que le permite configurar la aplicación para que los valores de propiedad se almacenen en un archivo de configuración externo en lugar del código compilado de la aplicación.

- [Soluciones y proyectos](#)

Explica Visual Studio cómo se administran de forma eficaz los elementos, como referencias, conexiones de datos, carpetas y archivos que requiere el esfuerzo de desarrollo a través de soluciones y proyectos.

- [Ampliar otras partes de Visual Studio](#)

Explica cómo usar los servicios de Visual Studio para crear elementos de interfaz de usuario que coincidan con el resto de servicios de Visual Studio.

Información general sobre la ventana Propiedades

06/04/2020 • 5 minutes to read • [Edit Online](#)

La ventana **Propiedades** se utiliza para mostrar las propiedades de los Visual Studio objetos seleccionados en los dos tipos principales de ventanas disponibles en el entorno de desarrollo integrado (IDE). Estos dos tipos de ventanas son:

- Ventanas de herramientas como el Explorador de soluciones, la vista de clases y el explorador de objetos
- Ventanas de documentos que contienen editores y diseñadores como diseñador de formularios, editor XML y editor HTML

Uso de la ventana Propiedades

La ventana **Propiedades** muestra las propiedades de uno o varios elementos seleccionados. Si se seleccionan varios elementos, se muestra la intersección de todas las propiedades de todos los objetos seleccionados.

Los eventos relacionados con un objeto seleccionado dentro de una ventana de diseño de formulario o un editor HTML mediante metadatos COM+ se muestran en la ventana **Propiedades**. Por ejemplo, puede seleccionar un botón y mostrar `OnClick` sus eventos asociados, como un evento, que se puede vincular a ese botón.

Los eventos que se muestran en la ventana **Propiedades** se utilizan principalmente con objetos enlazados al código. Si está editando un formato de archivo que no tiene nada que ver con el código, no va a tener ningún evento. Los eventos solo se muestran en la ventana **Propiedades** cuando hay un enlace entre el código en ejecución y determinados eventos asociados a objetos específicos. Un ejemplo de esto sería el código detrás de un objeto seleccionado que se ejecuta cuando se activa ese objeto.

En la tabla siguiente se enumeran las interfaces principales utilizadas por la ventana **Propiedades**.

NOMBRE DE LA INTERFAZ	DESCRIPCIÓN
ICategorizeProperties	Proporciona una lista de categorías a la ventana Propiedades y asigna cada propiedad a una categoría.
Interfaz IDispatch	Expone los métodos y propiedades de un objeto a herramientas de programación y otras aplicaciones que admiten la automatización.
IProvidePropertyBuilder	Proporciona los botones de puntos suspensivos (...) <i>denominados generadores</i> que abren ventanas de cuadro de diálogo modal implementadas por el propio objeto. Se utiliza cuando el usuario no escribe fácilmente un valor en un campo de texto. Por ejemplo, se puede utilizar para abrir un selector de color que determina el valor RGB por usted.
ISelectionContainer	Proporciona acceso a los objetos utilizados para actualizar la información que se muestra en la ventana Propiedades . ISelectionContainerVSPackages implementa para cada ventana que contiene objetos seleccionables con propiedades relacionadas que se mostrarán.
ITypeInfo	Proporciona información sobre el tipo de un objeto, como métodos de una interfaz y campos de una estructura.

NOMBRE DE LA INTERFAZ	DESCRIPCIÓN
IVsMonitorSelection	Permite VSPackages para recibir notificación de eventos de selección y para recuperar información sobre la jerarquía de proyecto actual, elemento, valor de elemento y contexto de interfaz de usuario de comandos.
IVsMultiItemSelect	Proporciona el entorno con acceso a varias selecciones.
IVsPerPropertyBrowsing	Se utiliza para proporcionar nombres localizados en algunas propiedades que se muestran en la ventana Propiedades .
IVsSelectionEvents	Notifica a los paquetes VSPackage registrados que se han producido cambios en la selección, el valor de elemento o el contexto de la interfaz de usuario de comandos actuales.
IVsTrackSelectionEx	Notifica al entorno de un cambio en la selección actual y proporciona acceso a la información de jerarquía y de elemento relacionada con la nueva selección.

Para obtener `IDispatch` más información sobre , consulte la biblioteca MSDN.

Vea también

- [Extensión de propiedades](#)
- [Interfaces y campos de la ventana Propiedades](#)

Directiva de plantilla y ventana Propiedades

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando un proyecto está contenido dentro de un proyecto de plantilla de empresa, ese proyecto de plantilla de empresa puede aplicar la directiva. La directiva de plantilla se convierte en un sistema de restricción que se puede usar para establecer valores predeterminados para propiedades, ocultar propiedades, agregar propiedades, etc.

El uso de la directiva de **Properties** plantilla para controlar [IVsPerPropertyBrowsing](#) la visualización de información en la ventana Propiedades es diferente de la implementación de [IVsPerPropertyBrowsing](#). [IVsPerPropertyBrowsing](#) controla las propiedades de los objetos en el nivel de componente, mientras que la directiva de plantilla se puede utilizar para restringir las propiedades de objeto en el nivel de solución o proyecto. En otras palabras,

- Implemente los [IVsPerPropertyBrowsing](#) métodos para determinar lo que se muestra en la ventana **Propiedades** para objetos específicos
- Use la directiva de plantilla en el nivel de solución y proyecto para determinar lo que se muestra en la ventana **Propiedades** para los objetos especificados anteriormente

El uso de la directiva de plantilla para restringir selectivamente propiedades específicas en la ventana **Propiedades** cuando se selecciona un elemento de proyecto de un tipo especificado en el **Explorador** de soluciones puede ser beneficioso para todos los miembros del equipo de desarrollo que trabajan en un proyecto. Por ejemplo, mediante la directiva de plantilla, puede configurar toda la información de cadena de conexión en una base de datos para los desarrolladores y hacer que la cadena de conexión sea de solo lectura. De este modo, puede proporcionar una forma sencilla de asegurarse de que cada desarrollador utiliza la ruta de acceso correcta para el acceso a los datos.

Vea también

- [IVsPerPropertyBrowsing](#)
- [Extensión de propiedades](#)

Interfaces y campos de la ventana Propiedades

06/04/2020 • 13 minutes to read • [Edit Online](#)

El modelo de selección para determinar qué información se muestra en la ventana **Propiedades** se basa en la ventana que tiene el foco en el IDE. Cada ventana y objeto dentro de la ventana seleccionada pueden tener su objeto de contexto de selección insertado en el contexto de selección global. El entorno actualiza el contexto de selección global con valores de un marco de ventana cuando esa ventana tiene el foco. Cuando el foco cambia, también lo hace el contexto de selección.

Seguimiento de la selección en el IDE

El marco o sitio de ventana, propiedad del `STrackSelectionIDE`, tiene un servicio llamado `ISTrackSelection`. Los pasos siguientes muestran cómo se implementa un cambio en una selección, causado por el usuario cambiando el foco a otra ventana abierta o seleccionando un elemento de proyecto diferente en el **Explorador** de soluciones, para cambiar el contenido que se muestra en la ventana **Propiedades**.

1. El objeto creado por el `VSPackage` que se `QueryService` encuentra `ISTrackSelection` en la ventana seleccionada llama para tener invoke `ISTrackSelection`.
2. El contenedor de selección, proporcionado por `ISelectionContainer` la ventana seleccionada, crea su propio objeto. Cuando cambia la selección, `OnSelectChange` el `VSPackage` llama para notificar a los agentes de escucha en el entorno, incluida la **propiedades** ventana, del cambio. También proporciona acceso a la jerarquía y a la información de elementos relacionada con la nueva selección.
3. Al `OnSelectChange` llamarlo y pasarle `VSHPROPID_BrowseObject` los elementos `ISelectionContainer` de jerarquía seleccionados en el parámetro, se rellena el objeto.
4. Un objeto derivado de la `IDispatch` se devuelve para `__VSHPROPID.VSHPROPID_BrowseObject` para el elemento solicitado y el `ISelectionContainer` entorno lo envuelve en un (consulte el paso siguiente). Si se produce un error en `IVsHierarchy::GetProperty` la llamada, el entorno realiza una segunda llamada a , pasándole el contenedor de selección `__VSHPROPID.VSHPROPID_SelContainer` que el artículo de jerarquía o los artículos proporcionan.

El proyecto `VSPackage` `ISelectionContainer` no crea porque la ventana proporcionada por el entorno `VSPackage` `ISelectionContainer` que lo implementa (por ejemplo, El **Explorador** de soluciones) construye en su nombre.

5. El entorno invoca los `ISelectionContainer` métodos de obtener `IDispatch` los objetos basados en la interfaz para rellenar la ventana **Propiedades**.

Cuando se cambia un valor en el `IVsTrackSelectionEx::OnElementValueChangeEx` `IVsTrackSelectionEx::OnSelectionChangeEx` **propiedades** ventana, `VSPackages` implementan y para notificar el cambio al valor del elemento. A continuación, `IVsUIShell` el `IConnectionPointContainer` entorno invoca o para mantener la información mostrada en la ventana **Propiedades** sincronizada con los valores de propiedad. Para obtener más información, vea Actualización de valores de [propiedad en la ventana Propiedades](#).

Además de seleccionar un elemento de proyecto diferente en el **Explorador** de soluciones para mostrar las propiedades relacionadas con ese elemento, también puede elegir un objeto diferente dentro de un formulario o ventana de documento mediante la lista desplegable disponible en la ventana **Propiedades**. Para obtener más información, consulte Lista de [objetos](#) de ventana de propiedades .

Puede cambiar la forma en que se muestra la información en la cuadrícula de la ventana **Propiedades** de alfabética a categórica y, si está disponible, también puede abrir una página de propiedades para un objeto seleccionado haciendo clic en los botones adecuados de la ventana **Propiedades**. Para obtener más información, vea [Botones](#) de ventana de propiedades y [páginas](#) de propiedades .

Por último, la parte inferior de la ventana **Propiedades** también contiene una descripción del campo seleccionado en la cuadrícula de la ventana **Propiedades**. Para obtener más información, consulte [Obtención de descripciones de campos desde la ventana Propiedades](#).

Actualización de los valores de propiedad en la ventana Propiedades

Existen dos maneras de mantener la ventana **Propiedades** sincronizada con los cambios de valores de propiedad. La primera es llamar a la interfaz [IVsUIShell](#), que proporciona acceso a la funcionalidad básica basada en ventanas, incluidos el acceso y la creación de ventanas de documentos y herramientas proporcionadas por el entorno. Los pasos siguientes describen este proceso de sincronización.

Actualización de los valores de propiedad mediante IVsUIShell

Para actualizar los valores de propiedad mediante la interfaz [IVsUIShell](#)

1. Llame a [IVsUIShell](#) (a través del servicio [SVsUIShell](#)) cada vez que paquetes [VSPackage](#), proyectos, o editores necesiten crear o enumerar las ventanas de documentos o herramientas.
2. Implementar [RefreshPropertyBrowser](#) para mantener la ventana **Propiedades** sincronizada con los cambios de propiedad de un proyecto [OnChanged](#) (o cualquier otro objeto seleccionado que esté examinando la ventana **Propiedades**) sin implementar [IConnectionPointContainer](#) ni desencadenar eventos.
3. Implemente los métodos de [IVsHierarchyAdviseHierarchyEvents](#) y [UnadviseHierarchyEvents](#) para establecer y deshabilitar, respectivamente, la notificación de cliente de los eventos de jerarquía sin requerir que la jerarquía implemente [IConnectionPointContainer](#).

Actualización de los valores de propiedad mediante IConnection

La segunda manera de mantener la ventana **Propiedades** sincronizada con los cambios de valores de propiedad consiste en implementar `IConnection` en el objeto conectable para indicar la existencia de las interfaces de salida. Si desea localizar el nombre de propiedad, derive su objeto de [ICustomTypeDescriptor](#). La implementación de [ICustomTypeDescriptor](#) puede modificar los descriptores de propiedad que devuelve y cambiar el nombre de una propiedad. Para localizar la descripción, cree un atributo que se derive de [DescriptionAttribute](#) e invalide la propiedad `Description`.

Consideraciones de la implementación de la interfaz IConnection

1. `IConnection` proporciona acceso a un subobjeto enumerador con la interfaz [IEnumConnectionPoints](#). También proporciona acceso a todos los subobjetos del punto de conexión, cada uno de los cuales implementa la interfaz [IConnectionPoint](#).
2. Cualquier objeto de búsqueda es responsable de implementar un evento [IPropertyNotifySink](#). La ventana **Propiedades** aconsejará sobre el evento establecido a través de `IConnection` .
3. Un punto de conexión controla cuántas conexiones (una o más) permite en su implementación de [Advise](#). Un punto de conexión que permita solo una interfaz puede devolver [E_NOTIMPL](#) desde el método [EnumConnections](#).
4. Un cliente puede llamar a la interfaz `IConnection` para obtener acceso a un subobjeto enumerador con la interfaz [IEnumConnectionPoints](#). A continuación, se puede llamar a la interfaz [IEnumConnectionPoints](#) para enumerar los puntos de conexión de cada identificador de interfaz (IID) de salida.
5. `IConnection` también se puede llamar para obtener acceso a los subobjetos del punto de conexión con la interfaz [IConnectionPoint](#) para cada IID saliente. A [IConnectionPoint](#) través de la interfaz, un cliente inicia o

termina un bucle de aviso con el objeto conectable y la propia sincronización del cliente. El cliente también [IConnectionPoint](#) puede llamar a la [IEnumConnections](#) interfaz para obtener un objeto enumerador con la interfaz para enumerar las conexiones que conoce.

Obtención de descripciones de campo desde la ventana Propiedades

En la parte inferior de la ventana **Propiedades** , un área de descripción muestra información relacionada con el campo de la propiedad seleccionada. Esta característica está activada de forma predeterminada. Si quiere ocultar el campo de descripción, haga clic con el botón derecho en la ventana **Propiedades** y haga clic en **Descripción**. Al hacerlo, también se quita la marca de verificación junto al título **Descripción** de la ventana de menú. Puede volver a mostrar el campo siguiendo los mismos pasos para volver a activar **Descripción** .

La información del campo descripción proviene de [ITypeInfo](#). Cada método, interfaz, coclase, etc. puede tener un atributo `helpstring` sin localizar en la biblioteca de tipos. La ventana **Propiedades** recupera [GetDocumentation](#) la cadena de .

Para especificar cadenas de ayuda localizadas

1. Agregue el atributo `helpstringdll` a la instrucción de la biblioteca en la biblioteca de tipos (`typelib`).

NOTE

Este paso es opcional si la biblioteca de tipos está en un archivo de biblioteca de objetos (.olb).

2. Especifique atributos `helpstringcontext` para las cadenas. También puede especificar atributos `helpstring` .

Estos atributos son distintos de los atributos `helpfile` y `helpcontext` , que se encuentran en los temas de Ayuda del archivo .chm real.

Para recuperar la información de descripción que se mostrará para [GetDocumentation2](#) el nombre de propiedad resaltado, `lcid` la ventana **Propiedades** llama a la propiedad seleccionada, especificando el atributo deseado para la cadena de salida. Internamente, [ITypeInfo](#) busca el archivo .dll especificado en el atributo `helpstringdll` y llama a la función `DLLGetDocumentation` en dicho archivo .dll con el contexto especificado y el atributo `lcid` .

La signatura y la implementación de `DLLGetDocumentation` SON:

```
STDAPI DLLGetDocumentation
(
    ITypeLib * /* ptlib */,
    ITypeInfo * /* ptinfo */,
    LCID /* lcid */,
    DWORD dwCtx,
    BSTR * pbstrHelpString
);
```

La función `DLLGetDocumentation` debe ser una exportación definida en el archivo .def para la DLL.

Internamente, se crea un archivo .olb, que en realidad es una DLL. Esta DLL contiene un recurso, el archivo de biblioteca de tipos (.tlb) y una función exportada, `DLLGetDocumentation` .

En el caso de los archivos .olb, el atributo `helpstringdll` es opcional porque es el mismo archivo que contiene el propio archivo .tlb.

Para obtener cadenas que aparezcan en el panel **Descripciones** , por tanto, lo mínimo que debe hacer es especificar el atributo `helpstring` en la biblioteca de tipos. Si quiere que estas cadenas se localicen, debe

especificar el atributo `helpstringdll` (opcional) y el atributo `helpstringcontext` (obligatorio) e implementar `DLLGetDocumentation` .

No hay interfaces adicionales que deban implementarse al obtener información localizada a través del atributo `helpstringcontext` y la función `DLLGetDocumentation` del archivo .idl.

Otra manera de obtener el nombre y la descripción localizados de una propiedad es implementar [GetLocalizedPropertyInfo](#). Para obtener más información sobre la implementación de este método, consulte [Properties Window Fields and Interfaces](#).

Vea también

- [Extensión de propiedades](#)

Lista de objetos de la ventana Propiedades

06/04/2020 • 2 minutes to read • [Edit Online](#)

La lista de objetos de la ventana **Propiedades** es una lista desplegable que permite cambiar la selección a otros objetos disponibles en una o varias ventanas seleccionadas. Al seleccionar un objeto diferente de esta [SelectObjects](#) lista, se desencadena una llamada para informar al entorno de que se ha seleccionado un nuevo objeto. A continuación, se cambia la información que se muestra en la ventana **Propiedades** para mostrar las propiedades asociadas al objeto recién seleccionado.

La lista de objetos

La lista de objetos consta de dos campos: el nombre del objeto (que se muestra en negrita) y el tipo de objeto.

El nombre de objeto que se muestra a la izquierda del tipo `Name` de objeto [IProvideClassInfo](#) en negrita se recupera del propio objeto mediante la propiedad proporcionada por la interfaz. [GetClassInfo](#), el único [IProvideClassInfo](#) método [ITypeInfo](#) en , devuelve para la coclase de esa interfaz. La **Properties** ventana [IProvideClassInfo](#) **Propiedades** se utiliza para obtener el nombre de la coclase, que se muestra como el nombre del objeto en la lista desplegable.

Si el objeto no `Name` tiene una propiedad, no se muestra un nombre en el área Nombre de la lista de objetos. Puede agregar una propiedad Name al objeto si desea que el nombre se muestre en la lista de objetos.

Si el objeto COM [IProvideClassInfo](#) no implementa , la ventana **Propiedades** muestra el nombre de la interfaz en lugar del nombre del objeto en el lado izquierdo de la lista.

Vea también

- [Extensión de propiedades](#)

Botones de la ventana Propiedades

06/04/2020 • 3 minutes to read • [Edit Online](#)

Según el idioma de desarrollo y el tipo de producto, algunos botones se muestran de forma predeterminada en la barra de herramientas de la ventana **Propiedades**. En todos los casos, se muestran los botones **Clasificado**, **Alfabetizado**, **Propiedades** y **Páginas** de propiedades. También se muestra el botón **Eventos** en Visual C. y Visual Basic. En determinados proyectos de Visual C++, se muestran los botones **Vc++ Messages** y **VC Overrides**. Es posible que se muestren botones adicionales para otros tipos de proyecto. Para obtener más información acerca de los botones de la ventana **Propiedades**, vea [Ventana Propiedades](#).

Implementación de botones de ventana de propiedades

Al hacer clic en el **botón Clasificado**, Visual Studio llama a la [ICategorizeProperties](#) interfaz en el objeto que tiene el foco para ordenar sus propiedades por categoría. [ICategorizeProperties](#) se implementa en `IDispatch` el objeto que se presenta en la ventana **Propiedades**.

Hay 11 categorías de propiedades predefinidas, que tienen valores negativos. Puede definir categorías personalizadas, pero le recomendamos que les asigne valores positivos para distinguirlas de las categorías predefinidas.

El [MapPropertyToCategory](#) método devuelve el valor de categoría de propiedad adecuado para la propiedad especificada. El [GetCategoryName](#) método devuelve una cadena que contiene el nombre de categoría. Solo tiene que proporcionar compatibilidad con los valores de categoría personalizados porque Visual Studio conoce los valores de categoría de propiedad estándar.

Al hacer clic en el botón **Alphabetized**, las propiedades se muestran en orden alfabético por nombre. Los nombres se `IDispatch` recuperan según un algoritmo de ordenación localizado.

Cuando la ventana **Propiedades** está abierta, el botón **Propiedades** se muestra automáticamente como seleccionado. En otras partes del entorno, se muestra el mismo botón y puede hacer clic en él para mostrar la ventana **Propiedades**.

El botón **Páginas** `ISpecifyPropertyPages` de propiedades no está disponible si no se implementa para el objeto seleccionado. Las páginas de propiedades muestran propiedades dependientes de la configuración que normalmente están asociadas con soluciones y proyectos, pero también se pueden asociar a elementos de proyecto (por ejemplo, en Visual C++).

NOTE

No puede agregar botones de barra de herramientas a la ventana **Propiedades** mediante código no administrado. Para agregar un botón de barra de herramientas, debe crear un objeto administrado que derive de [PropertyTab](#).

Vea también

- [Extensión de propiedades](#)

Cuadrícula de visualización de propiedades

06/04/2020 • 6 minutes to read • [Edit Online](#)

La ventana **Propiedades** muestra campos dentro de una cuadrícula. La columna izquierda contiene los nombres de propiedad; la columna derecha contiene los valores de propiedad.

Trabajar con la cuadrícula

La lista de dos columnas muestra las propiedades independientes de la configuración que se pueden cambiar en tiempo de diseño y su configuración actual. Tenga en cuenta que es posible que no se muestren todas las propiedades. Una propiedad se puede establecer como oculta, [HideProperty](#) por ejemplo, mediante la implementación del método. En concreto, para ocultar propiedades que tienen propiedades secundarias:

1. Establezca `pfDisplay` el [DisplayChildProperties](#) parámetro `FALSE` en .
2. Establezca `pfHide` el [HideProperty](#) parámetro `TRUE` en .

Para insertar información en la ventana [ISelectionContainer](#) **Propiedades**, el IDE usa .

[ISelectionContainer](#) VSPackages llama para cada ventana que contiene objetos seleccionables con propiedades relacionadas que se mostrarán en el **propiedades** ventana. **Implementación** del Explorador [ISelectionContainer](#) `GetProperty` de soluciones de llamadas mediante `__VSHPROPID.VSHPROPID_BrowseObject` en la jerarquía del proyecto para adquirir los objetos que se pueden examinar en la jerarquía.

Si el VSPackage no admite `__VSHPROPID.VSHPROPID_BrowseObject`, el IDE `GetProperty` intenta usar el valor de `__VSHPROPID.VSHPROPID_SelContainer` que el artículo de jerarquía o los artículos proporcionan.

El proyecto VSPackage no [ISelectionContainer](#) necesita crear porque el paquete de ventana proporcionado por IDE [ISelectionContainer](#) que lo implementa (por ejemplo, El **Explorador** de soluciones) se construye en su nombre.

[ISelectionContainer](#) consta de tres métodos a los que llama el IDE:

- [CountObjects](#) contiene el número de objetos seleccionados para mostrarse en la ventana **Propiedades**.
- [GetObjects](#) devuelve `IDispatch` los objetos seleccionados para mostrarse en la ventana **Propiedades**.
- [SelectObjects](#) hace posible que cualquiera de [GetObjects](#) los objetos devueltos por sea seleccionado por el usuario. Esto permite que el VSPackage para actualizar visualmente la selección que se muestra al usuario en la interfaz de usuario.

La ventana **Propiedades** extrae `IDispatch` información de los objetos para recuperar las propiedades que se están examinando. El explorador `IDispatch` de propiedades utiliza para preguntar al `ITypeInfo` objeto qué propiedades `IDispatch::GetTypeInfo` admite mediante la consulta, que se obtiene de . A continuación, el explorador utiliza estos valores para rellenar la ventana **Propiedades** y cambiar los valores de las propiedades individuales que se muestran en la cuadrícula. La información de propiedades se mantiene dentro del propio objeto.

Dado que los `IDispatch` objetos devueltos admiten , el llamador `IDispatch::Invoke` `ITypeInfo::Invoke` puede obtener información como el nombre del objeto llamando a o con un identificador de envío predefinido (DISPID) que representa la información deseada. Los DISPID declarados son negativos para asegurarse de que no entren en conflicto con los identificadores definidos por el usuario.

La ventana **Propiedades** muestra diferentes tipos de campos en función de los atributos de propiedades específicas de un objeto seleccionado. Estos campos incluyen cuadros de edición, listas desplegables y vínculos a

cuadros de diálogo de editor personalizados.

- Los valores contenidos en una lista [GetObjects](#) enumerada `IDispatch` se recuperan mediante una consulta en . Los valores obtenidos de una lista enumerada se pueden cambiar en la cuadrícula de propiedades haciendo doble clic en el nombre del campo o haciendo clic en el valor y seleccionando el nuevo valor de la lista desplegable. Para las propiedades que tienen la configuración predefinida de las listas enumeradas, haga doble clic en el nombre de la propiedad en la lista Propiedades recorre las opciones disponibles. Para las propiedades predefinidas con solo dos opciones, como true/false, haga doble clic en el nombre de propiedad para cambiar entre las opciones.
- Si [HasDefaultValue](#) `false` es , lo que indica que se ha cambiado el valor, el valor se muestra en negrita. [CanResetPropertyValue](#) se utiliza para determinar si el valor se puede restablecer al valor original. Si es así, puede volver al valor predeterminado haciendo clic con el botón derecho en el valor y seleccionando **Restablecer** en el menú que se muestra. De lo contrario, debe volver a cambiar el valor al valor predeterminado manualmente. [IVsPerPropertyBrowsing](#) También permite localizar y ocultar los nombres de las propiedades que se muestran durante el tiempo de diseño, pero no afecta a los nombres de propiedad que se muestran durante el tiempo de ejecución.
- Al hacer clic en el botón de puntos suspensivos (...) se muestra una lista de valores de propiedad que el usuario puede seleccionar (por ejemplo, un selector de color o una lista de fuentes). [IProvidePropertyBuilder](#) proporciona estos valores.

Vea también

- [Extender propiedades](#)

Compatibilidad con las propiedades de proyecto y configuración

06/04/2020 • 8 minutes to read • [Edit Online](#)

La **Properties** ventana Propiedades Visual Studio del entorno de desarrollo integrado (IDE) puede mostrar las propiedades de proyecto y configuración. Puede proporcionar una página de propiedades para su propio tipo de proyecto para que el usuario pueda establecer propiedades para la aplicación.

Al seleccionar un nodo de proyecto en el **Explorador** de soluciones y, a continuación, hacer clic en **Propiedades** en el menú **Proyecto**, puede abrir un cuadro de diálogo que incluya las propiedades de proyecto y configuración. En Visual C# Visual Basic y, y tipos de proyecto derivados de estos idiomas, este cuadro de diálogo aparece como una página con fichas en el cuadro de [diálogo General, Entorno, Opciones](#). Para obtener más información, vea [No en compilación: Tutorial: exponer propiedades de proyecto y configuración \(C#\)](#).

Managed Package Framework for Projects (MPFProj) proporciona clases auxiliares para crear y administrar un nuevo sistema de proyectos. Puede encontrar el código fuente y las instrucciones de compilación en [MPF for Projects - Visual Studio 2013](#).

Persistencia de las propiedades de proyecto y configuración

Las propiedades de proyecto y configuración se conservan en un archivo de proyecto que tiene cualquier extensión de nombre de archivo asociada al tipo de proyecto, por ejemplo, .csproj, .vbproj y .myproj. Los proyectos de lenguaje suelen usar un archivo de plantilla para generar el archivo de proyecto. Sin embargo, en realidad hay varias maneras de asociar tipos de proyecto y plantillas. Para obtener más información, consulte Descripción del directorio de [plantillas \(. Vmdir\) Archivos](#).

Las propiedades de proyecto y configuración se crean agregando elementos al archivo de plantilla. Estas propiedades están disponibles para cualquier proyecto creado mediante el tipo de proyecto que usa esta plantilla. Visual C# proyectos y MPFProj usan el esquema [Not in Build: MSBuild Overview](#) para los archivos de plantilla. Estos archivos tienen una sección PropertyGroup para cada configuración. Las propiedades de los proyectos normalmente se conservan en la primera sección PropertyGroup, que tiene un argumento Configuration establecido en una cadena nula.

El código siguiente muestra el inicio de un archivo de proyecto básico de MSBuild.

```
<Project MSBuildVersion="2.0" DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Name>SomeProjectSix</Name>
    <SchemaVersion>2.0</SchemaVersion>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Debug' ">
    <Optimize>>false</Optimize>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)' == 'Release' ">
    <Optimize>>true</Optimize>
```

En este archivo `<Name>` `<SchemaVersion>` de proyecto y `<Optimize>` son propiedades de proyecto y es una propiedad de configuración.

Es responsabilidad del proyecto conservar las propiedades de proyecto y configuración del archivo de proyecto.

NOTE

Un proyecto puede optimizar la persistencia conservando solo los valores de propiedad que difieren de sus valores predeterminados.

Compatibilidad con las propiedades de proyecto y configuración

La `Microsoft.VisualStudio.Package.SettingsPage` clase implementa páginas de propiedades de proyecto y configuración. La implementación `SettingsPage` predeterminada de ofrece propiedades públicas a un usuario en una cuadrícula de propiedades genéricas. El `Microsoft.VisualStudio.Package.HierarchyNode.GetPropertyPageGuids` método selecciona las `SettingsPage` clases derivadas de las cuadrículas de propiedades del proyecto. El `Microsoft.VisualStudio.Package.ProjectNode.GetConfigPropertyPageGuids` método selecciona las `SettingsPage` clases derivadas de las cuadrículas de propiedades de configuración. El tipo de proyecto debe invalidar estos métodos para seleccionar las páginas de propiedades adecuadas.

La `SettingsPage` clase `Microsoft.VisualStudio.Package.ProjectNode` y la clase ofrecen estos métodos para conservar las propiedades de proyecto y configuración:

- `Microsoft.VisualStudio.Package.ProjectNode.GetProjectProperty` y `Microsoft.VisualStudio.Package.ProjectNode.SetProjectProperty` persista las propiedades del proyecto.
- `Microsoft.VisualStudio.Package.SettingsPage.GetConfigProperty` y `Microsoft.VisualStudio.Package.SettingsPage.SetConfigProperty` conservar las propiedades de configuración.

NOTE

Las implementaciones `Microsoft.VisualStudio.Package.SettingsPage` de `Microsoft.VisualStudio.Package.ProjectNode` las `Microsoft.Build.BuildEngine` clases y usan los métodos (MSBuild) para obtener y establecer las propiedades de proyecto y configuración desde el archivo de proyecto.

La clase de `SettingsPage` la `Microsoft.VisualStudio.Package.SettingsPage.ApplyChanges` `Microsoft.VisualStudio.Package.SettingsPage.BindProperties` que deriva debe implementar y conservar las propiedades de proyecto o configuración del archivo de proyecto.

ProvideObjectAttribute y ruta del registro

Las clases `SettingsPage` derivadas de están diseñadas para compartirse entre `VSPackages`. Para que sea posible que un `VSPackage` `SettingsPage` cree una `Microsoft.VisualStudio.Shell.ProvideObjectAttribute` clase derivada de `Microsoft.VisualStudio.Shell.Package`, agregue a una clase derivada de .

```
[ProvideObject(typeof(MyProjectPropertyPage))]  
[Guid("e815d8ad-49bf-427d-99c8-009db3e56ab9")]  
public sealed class MyPackage : Package
```

```
<ProvideObject(GetType(MyProjectPropertyPage))> _  
<Guid("e815d8ad-49bf-427d-99c8-009db3e56ab9")> _  
Public Class MyPackage  
    Inherits Package
```

El `VSPackage` al que está asociado el atributo no es importante. Cuando se registra un Visual Studio `VSPackage` con , el identificador de clase (CLSID) de [CreateInstance](#) cualquier objeto que se puede crear se registra para que una llamada a puede crearlo.

La ruta de acceso del Registro de [UserRegistryRoot](#) un objeto que se puede crear se determina combinando , la palabra, CLSID y el guid del tipo de objeto. Si `MyProjectPropertyPage` la clase tiene un guid de 3c693da2-5bca-49b3-bd95-ffe0a39dd723 y el UserRegistryRoot es HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 8.0Exp, a continuación, la ruta de acceso del Registro sería HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 8.0Exp, CLSID, 3c693da2-5bca-49b3-bd95-ffe0a39dd723.

Atributos y diseño de propiedades de proyecto y configuración

Los [CategoryAttribute](#) [DisplayNameAttribute](#) atributos [DescriptionAttribute](#) , y , y determinan el diseño, el etiquetado y la descripción de las propiedades de proyecto y configuración en una página de propiedades genérica. Estos atributos determinan la categoría, el nombre para mostrar y la descripción de la opción, respectivamente.

NOTE

Los atributos equivalentes, SRCategory, LocDisplayName y SRDescription, usan recursos de cadena para la localización y se definen en [MPF para Proyectos - Visual Studio 2013](#).

Observe el fragmento de código siguiente:

```
Private _myConfigProp As String
Private _isDirty As Boolean

Public Property IsDirty As Boolean
    Get
        Return _isDirty
    End Get
    Set(ByVal value As Boolean)
        _isDirty = value
    End Set
End Property

<Category("My Category")> _
<DisplayName("My Config Property")> _
<Description("My Description")> _
Public Property MyConfigProp() As String
    Get
        Return _myConfigProp
    End Get
    Set(ByVal value As String)
        _myConfigProp = value
        IsDirty = True
    End Set
End Property
```

```
public bool IsDirty { get; set; }
private string myConfigProp;

[Category("My Category")]
[DisplayName("My Config Property")]
[Description("My Description")]
public string MyConfigProp
{
    get { return myConfigProp; }
    set { myConfigProp = value; IsDirty = true; }
}
```

La `MyConfigProp` propiedad de configuración aparece en la página de propiedades de configuración como **My Config Property** en la categoría **My Category**. Si la opción está seleccionada, la descripción, **Mi descripción**, aparece en el panel de descripción.

Vea también

- [Adición y eliminación de páginas de propiedades](#)
- [Proyectos](#)
- [Archivos de descripción del directorio de plantilla \(.Vmdir\)](#)

Opciones y páginas de opciones

06/04/2020 • 8 minutes to read • [Edit Online](#)

Al hacer clic en **Opciones** en el menú **Herramientas**, se abre el cuadro de diálogo **Opciones**. Las opciones de este cuadro de diálogo se conocen colectivamente como páginas de opciones. El control de árbol del panel de navegación incluye categorías de opciones y cada categoría tiene páginas de opciones. Al seleccionar una página, sus opciones aparecen en el panel derecho. Estas páginas permiten cambiar los valores de las opciones que determinan el estado de un VSPackage.

Soporte para páginas de opciones

La [Package](#) clase proporciona compatibilidad para crear páginas de opciones y categorías de opciones. La [DialogPage](#) clase implementa una página de opciones.

La implementación [DialogPage](#) predeterminada de ofrece sus propiedades públicas a un usuario en una cuadrícula genérica de propiedades. Puede personalizar este comportamiento reemplazando varios métodos en la página para crear una página de opciones personalizadas que tenga su propia interfaz de usuario (UI). Para obtener más información, consulte [Creación de una página de opciones](#).

La [DialogPage](#) clase [IProfileManager](#) implementa, que proporciona persistencia para las páginas de opciones y también para la configuración de usuario. Las implementaciones predeterminadas de los [LoadSettingsFromStorage](#) métodos y [SaveSettingsToStorage](#) conservan los cambios de propiedad en una sección de usuario del registro si la propiedad se puede convertir a y desde una cadena.

Ruta del registro de la página de opciones

De forma predeterminada, la ruta de acceso del Registro [UserRegistryRoot](#) de las propiedades administradas por una página de opciones se determina combinando, la palabra `DialogPage` y el nombre de tipo de la clase de página de opciones. Por ejemplo, una clase de página de opciones podría definirse de la siguiente manera.

```
namespace Company.OptionsPage
{
    public class OptionsPageGeneral : DialogPage
    {
    }
}
```

```
Namespace Company.OptionsPage
    Public Class OptionsPageGeneral
        Inherits DialogPage
    End Class
End Namespace
```

Si [UserRegistryRoot](#) es `HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 8.0Exp`, los pares de nombre de propiedad y valor son subclaves de `HKEY_CURRENT_USER, Software, Microsoft, VisualStudio, 8.0Exp, DialogPage, Company.OptionsPage.OptionsPageGeneral`.

La ruta de acceso del Registro [ApplicationRegistryRoot](#) de la propia página de opciones se determina combinando, la palabra `ToolsOptionsPages`, y la categoría de página de opciones y el nombre. Por ejemplo, si la página Opciones personalizadas tiene [ApplicationRegistryRoot](#) la categoría `Mis páginas de opciones` y está `HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 8.0Exp`, la página de opciones tiene la clave del

Registro, HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 8.0Exp, ToolsOptionsPages, Mis páginas de opciones y Personalizada.

Atributos y diseño de página De herramientas/opciones

El [ProvideOptionPageAttribute](#) atributo determina la agrupación de páginas de opciones personalizadas en categorías en el árbol de navegación del cuadro de diálogo **Opciones**. El [ProvideOptionPageAttribute](#) atributo asocia una página de opciones con el VSPackage que proporciona la interfaz. Observe el fragmento de código siguiente:

```
[ProvideOptionPage(typeof(OptionsPageGeneral), "My Option Pages", "General", 101, 106, true)]
[ProvideOptionPage(typeof(OptionsPageCustom), "My Option Pages", "Custom", 101, 107, true)]
[Guid("B0002DC2-56EE-4931-93F7-70D6E9863940")]
public class MyPackage : Package
```

```
<ProvideOptionPage(GetType(OptionsPageGeneral), "My Option Pages", "General", 101, 106, True),
  ProvideOptionPage(GetType(OptionsPageCustom), "My Option Pages", "Custom", 101, 107, True),
  Guid("B0002DC2-56EE-4931-93F7-70D6E9863940")>
Public Class MyPackage
    Inherits Package
```

Esto declara que MyPackage proporciona dos páginas de opciones, OptionsPageGeneral y OptionsPageCustom. En el cuadro de diálogo **Opciones**, ambas páginas de opciones aparecen en la categoría **Mis páginas** de opciones como **General** y **Personalizado**, respectivamente.

Atributos de opción y diseño

La interfaz de usuario (UI) que proporciona la página determina la apariencia de las opciones en una página de opciones personalizadas. El diseño, el etiquetado y la descripción de las opciones de una página de opciones genéricas están determinados por los siguientes atributos:

- [CategoryAttribute](#) determina la categoría de la opción.
- [DisplayNameAttribute](#) determina el nombre para mostrar de la opción.
- [DescriptionAttribute](#) determina la descripción de la opción.

NOTE

Los atributos equivalentes, SRCategory, LocDisplayName y SRDescription, usan recursos de cadena para la localización y se definen en el ejemplo de [proyecto administrado](#).

Observe el fragmento de código siguiente:

```
[Category("My Options")]
[DisplayName("Integer Option")]
[Description("My integer option")]
public int OptionInteger { get; set; }
```

```

Private _optionInt As Integer

<Category("My Options"),
DisplayName("Integer Option"),
Description("My integer option")>
Public Property OptionInteger() As Integer
    Get
        Return _optionInt
    End Get
    Set(ByVal value As Integer)
        _optionInt = value
    End Set
End Property

```

La opción OptionInteger aparece en la página de opciones como **Opción entero** en la categoría **Mis opciones**. Si la opción está seleccionada, la descripción, **Mi opción de entero**, aparece en el cuadro de descripción.

Acceso a las páginas de opciones desde otro VSPackage

Un VSPackage que hospeda y administra una página de opciones se puede tener acceso mediante programación desde otro VSPackage mediante el modelo de automatización. Por ejemplo, en el código siguiente se registra un VSPackage como hospedaje de una página de opciones.

```

[ProvideOptionPage(typeof(MyOptionPage), "My Category", "My Grid Page", 0, 0, true)]
[Guid("6bb6942e-014c-489e-a612-a935680f703d")]
public sealed class MyToolsOptions : Package

```

```

<ProvideOptionPage(GetType(MyOptionPage), "My Category", "My Grid Page", 0, 0, True)>
<Guid("6bb6942e-014c-489e-a612-a935680f703d")>
Public NotInheritable Class MyToolsOptions
    Inherits Package

```

El siguiente fragmento de código obtiene el valor de OptionInteger de MyOptionPage:

```

DTE dte = (DTE)GetService(typeof(DTE));
EnvDTE.Properties props = dte.get_Properties("My Category", "My Grid Page");
int n = (int)props.Item("OptionInteger").Value;

```

```

Dim dte As DTE = CType(GetService(GetType(DTE)), DTE)
Dim props As EnvDTE.Properties = dte.get_Properties("My Category", "My Grid Page")
Dim n As Integer = CInt(Fix(props.Item("OptionInteger").Value))

```

Cuando [ProvideOptionPageAttribute](#) el atributo registra una página de opciones, la página `SupportsAutomation` se registra `true` en la clave AutomationProperties si el argumento del atributo es `.`. Automatización examina esta entrada del Registro para buscar el VSPackage asociado y, a continuación, la automatización tiene acceso a la propiedad a través de la página de opciones hospedadas, en este caso, Mi página de cuadrícula.

La ruta de acceso del Registro [ApplicationRegistryRoot](#) de la propiedad de automatización se determina combinando , la palabra, AutomationProperties y la categoría y el nombre de la página de opciones. Por ejemplo, si la página de opciones tiene la categoría [ApplicationRegistryRoot](#) Mi categoría, el nombre de mi página de cuadrícula y , HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 8.0Exp, la propiedad de automatización tiene la clave del Registro, HKEY_LOCAL_MACHINE, software, Microsoft, VisualStudio, 8.0Exp, AutomationProperties, mi categoría, mi página de cuadrícula.

NOTE

El nombre canónico, Mi Category.My página de cuadrícula, es el valor de la subclave Nombre de esta clave.

Crear páginas de opciones

06/04/2020 • 6 minutes to read • [Edit Online](#)

En Visual Studio el marco de paquete [DialogPage](#) administrado, Visual Studio las clases derivadas de ampliar el IDE **agregando** opciones páginas en el **herramientas** menú.

Un objeto que implementa una determinada página de [ProvideOptionPageAttribute](#) opción de **herramientas** está asociado con VSPackages específicos por el objeto.

Dado que el entorno crea una instancia del objeto que implementa una página **de opciones** de herramientas determinada cuando el IDE muestra esa página determinada:

- Un **opciones** de herramientas página debe implementarse en su propio objeto y no en el objeto que implementa un VSPackage.
- Un objeto no puede implementar varias páginas **Opciones** de herramientas.

Regístrese como proveedor de páginas Opciones de herramientas

Un VSPackage que admite la configuración de usuario a través **de opciones** de herramientas páginas indica los objetos que proporcionan estas páginas de opciones de **herramientas** mediante la aplicación de instancias de [ProvideOptionPageAttribute](#) aplicado a la [Package](#) implementación.

Debe haber una [ProvideOptionPageAttribute](#) instancia [DialogPage](#) de cada tipo derivado que implemente una página **Opciones** de herramientas.

Cada instancia [ProvideOptionPageAttribute](#) de utiliza el tipo que implementa la página **Opciones** de herramientas, las cadenas que contienen la categoría y la subcategoría utilizadas para identificar una página **Opciones** de herramientas e información de recursos para registrar el tipo como una página **Opciones** de herramientas.

Estado de la página Opciones de herramientas de persistencia

Si se registra una implementación de página **Opciones** de herramientas con compatibilidad con automatización habilitada, el IDE conserva el estado de la página junto con todas las demás páginas **Opciones** de herramientas.

Un VSPackage puede administrar su [ProvideProfileAttribute](#) propia persistencia mediante . Solo se debe utilizar uno u otro método de persistencia.

Implementar la clase DialogPage

Un objeto que proporciona la implementación de un VSPackage de un [DialogPage](#) tipo derivado puede aprovechar las siguientes características heredadas:

- Una ventana de interfaz de usuario predeterminada.
- Un mecanismo de persistencia [ProvideProfileAttribute](#) predeterminado disponible si se [SupportsProfiles](#) aplica a `true` la [ProvideOptionPageAttribute](#) clase o si la propiedad se establece en para la que se aplica a la clase.
- Soporte de automatización.

El requisito mínimo para **Tools Options** un objeto [DialogPage](#) que implementa una página Opciones de herramientas mediante es la adición de propiedades públicas.

Si la clase se registra correctamente como proveedor de **páginas Opciones** de herramientas, sus propiedades públicas están disponibles en la sección **Opciones** del menú **Herramientas** en forma de cuadrícula de propiedades.

Todas estas características predeterminadas se pueden invalidar. Por ejemplo, para crear una interfaz de usuario más **Window** sofisticada solo se requiere reemplazar la implementación predeterminada de .

Ejemplo

Lo que sigue es una simple implementación "Hola mundo" de una página de opciones. Agregar el código siguiente a un proyecto predeterminado creado por la plantilla de paquete de Visual Studio con la opción **Comando** de menú seleccionada demostrará adecuadamente la funcionalidad de la página de opciones.

Descripción

La siguiente clase define una página de opciones mínima "Hola mundo". Cuando se abre, el `HelloWorld` usuario puede establecer la propiedad pública en una cuadrícula de propiedades.

Código

```
using System;
using Microsoft.VisualStudio.Shell;

namespace Company.UIUserSettingsToolsOptionsPages
{
    class HelloWorldOptions : DialogPage
    {
        bool helloWorld = true;
        public bool HelloWorld
        {
            get { return helloWorld; }
            set { helloWorld = value; }
        }
    }
}
```

```
Imports System
Imports Microsoft.VisualStudio.Shell

Namespace Company.UIUserSettingsToolsOptionsPages
    Class HelloWorldOptions
        Inherits DialogPage
        Private m_helloWorld As Boolean = True
        Public Property HelloWorld() As Boolean
            Get
                Return m_helloWorld
            End Get
            Set(ByVal value As Boolean)
                m_helloWorld = value
            End Set
        End Property
    End Class
End Namespace
```

Descripción

Aplicar el siguiente atributo a la clase de paquete hace que la página de opciones esté disponible cuando se carga el paquete. Los números son identificadores de recursos arbitrarios para la categoría y la página, y el valor booleano al final especifica si la página admite la automatización.

Código

```
[ProvideOptionPage(typeof(HelloWorldOptions),
    "Hello World Category", "Hello World Page",
    1000, 1001, false)]
```

```
<PackageRegistration(UseManagedResourcesOnly:=True)> _
<DefaultRegistryRoot("Software\Microsoft\VisualStudio\9.0")> _
<InstalledProductRegistration(False, "#110", "#112", "1.0", IconResourceID:=400)> _
<ProvideLoadKey("Standard", "1.0", "Package Name", "Company", 1)> _
<ProvideMenuResource(1000, 1)> _
<Guid(GuidList.guidPkgString)> _
<ProvideOptionPage(GetType(HelloWorldOptions), "Hello World Category", "Hello World Page", 1000, 1001, False)>
_
Public NotInheritable Class UIUserSettingsToolsOptionsPagesPackage
    Inherits Package
```

Descripción

El siguiente controlador de eventos muestra un resultado en función del valor del conjunto de propiedades en la página de opciones. Utiliza el [GetDialogPage](#) método con el resultado explícitamente convertido en el tipo de página de opción personalizada para tener acceso a las propiedades expuestas por la página.

En el caso de un proyecto generado por la `MenuItemCallback` plantilla de paquete, llame a esta función desde la función para adjuntarla al comando predeterminado agregado al menú **Herramientas**.

Código

```
private void ShowHelloWorld(object sender, EventArgs e)
{
    var hw = GetDialogPage(typeof(HelloWorldOptions))
        as HelloWorldOptions;
    if (hw.HelloWorld == true)
    {
        MessageBox.Show("Hello World!");
    }
    else MessageBox.Show("Goodbye.");
}
```

```
Private Sub ShowHelloWorld(ByVal sender As Object, ByVal e As EventArgs)
    Dim hw = TryCast(GetDialogPage(GetType(HelloWorldOptions)), HelloWorldOptions)
    If hw.HelloWorld = True Then
        MessageBox.Show("Hello World!")
    Else
        MessageBox.Show("Goodbye.")
    End If
End Sub
```

Vea también

- [Ampliar la configuración y las opciones del usuario](#)
- [Soporte de automatización para páginas de opciones](#)

Compatibilidad con la configuración de usuario

06/04/2020 • 6 minutes to read • [Edit Online](#)

Un VSPackage puede definir una o varias categorías de configuración, que son grupos de variables de estado que persisten cuando un usuario elige el comando **Importar/Exportar configuración** en el menú **Herramientas**. Para habilitar esta persistencia, utilice las Visual Studio SDKAPI de configuración en el archivo .

Una entrada del Registro que se conoce como un punto de configuración personalizado y un GUID define la categoría de configuración de un VSPackage. Un VSPackage puede admitir varias categorías de configuración, cada una definida por un punto de configuración personalizado.

- Las implementaciones de la configuración basada en [IVsUserSettings](#) ensamblados de interoperabilidad (mediante la interfaz) deben crear un punto de configuración personalizado editando el registro o utilizando un script de registrador (archivo .rgs). Para obtener más información, consulta [Creating Registrar Scripts](#).
- El código que usa Managed Package Framework (MPF) debe [ProvideProfileAttribute](#) crear puntos de configuración personalizados adjuntando un al VSPackage para cada punto de configuración personalizado.

Si un único VSPackage admite varios puntos de configuración personalizados, cada punto de configuración personalizado [ProvideProfileAttribute](#) se implementa mediante una clase independiente y cada uno está registrado por una instancia única de la clase. Por lo tanto, una configuración que implementa la clase puede admitir más de una categoría de configuración.

Detalles de la entrada del registro de puntos de configuración personalizada

Los puntos de configuración personalizados se crean en una entrada del Registro en \<CSPName> la \<CSPName> siguiente ubicación: HKLM, Software, Microsoft, VisualStudio*<Versión>, *donde está el nombre del punto de configuración personalizado que admite VSPackage y * <Version> * es la versión de Visual Studio, por ejemplo, 8.0.*

NOTE

La ruta de acceso de la raíz de HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, Visual Studio VisualStudio*<Versión>* se puede invalidar con una raíz alternativa cuando se inicializa el entorno de desarrollo integrado (IDE). Para obtener más información, consulte [Conmutadores de línea de comandos](#).

La estructura de la entrada del registro se ilustra a continuación:

HKLM, Software, Microsoft, \VisualStudio*<Versión>* de Usuario.

<CSPName> > '#12345'

Paquete á "XXXXXX XXXX XXXX XXXXXXXXXX"

Categoría "YYYYYA AAAA AAAA

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAA

Paquete de Recursos de la Imagen zzzzZZZZZZZZZZZz ZZZZZZZZZZ'

AlternateParent - CategoryName

NOMBRE	TIPO	DATA	DESCRIPCIÓN
(Es el valor predeterminado).	REG_SZ	Nombre del punto de configuración personalizado	El nombre de <code><CSPName</code> la clave,>, es el nombre no localizado del punto de configuración personalizado. Para las implementaciones basadas en MPF, el nombre <code>categoryName</code> <code>objectName</code> de la ProvideProfileAttribute clave <code>categoryName_objectName</code> se obtiene combinando los argumentos y del constructor en . La clave puede estar vacía o puede contener el identificador de referencia de la cadena localizada en un archivo DLL satélite. Este valor se obtiene <code>objectNameResourceID</code> del ProvideProfileAttribute argumento al constructor.
Paquete	REG_SZ	GUID	Guid del VSPackage que implementa el punto de configuración personalizado. Implementaciones basadas en ProvideProfileAttribute MPF mediante la <code>objectType</code> clase, use el Type argumento del constructor que contiene el VSPackage y la reflexión para obtener este valor.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
Category	REG_SZ	GUID	GUID que identifica la categoría de configuración. Para las implementaciones basadas en ensamblados de interoperabilidad, este Visual Studio valor puede ser ExportSettings un ImportSettings GUID elegido arbitrariamente, que el IDE pasa a los métodos y. Todas las implementaciones de estos dos métodos deben comprobar sus argumentos GUID. Para las implementaciones basadas en MPF, Type este GUID Visual Studio se obtiene mediante la de la clase que implementa el mecanismo de configuración.
ResourcePackage	REG_SZ	GUID	Opcional. Ruta de acceso a DLL satélite que contiene cadenas localizadas si el VSPackage de implementación no las proporciona. MPF usa la reflexión para obtener el ProvideProfileAttribute recurso correcto VSPackage, por lo que la clase no establece este argumento.

NOMBRE	TIPO	DATA	DESCRIPCIÓN
AlternateParent	REG_SZ	Nombre de la carpeta en la página Opciones de herramientas que contiene este punto de configuración personalizado.	Opcional. Debe establecer este valor solo si Tools Options una implementación de configuración admite Visual Studio SDK páginas de opciones de herramientas que usan el mecanismo de persistencia en el mecanismo en lugar del mecanismo en lugar del modelo de automatización para guardar el estado. En estos casos, el valor de <code>topic</code> la clave <code>topic.sub-topic</code> AlternateParent es la sección de la cadena utilizada para identificar la página ToolsOptions determinada. Por ejemplo, para la <code>"TextEditor.Basic"</code> página ToolsOptions el <code>"TextEditor"</code> valor de AlternateParent sería . Cuando ProvideProfileAttribute se genera el punto de configuración personalizado, es el mismo que el nombre de la categoría.

Conceptos básicos del servicio

06/04/2020 • 6 minutes to read • [Edit Online](#)

Un servicio es un contrato entre dos VSPackages. Un VSPackage proporciona un conjunto específico de interfaces para otro VSPackage para consumir. Visual Studioes en sí misma una colección de VSPackages que proporciona servicios a otros VSPackages.

Por ejemplo, puede usar el servicio SVsActivityLog para obtener una interfaz IVsActivityLog, que puede usar para escribir en el registro de actividad. Para obtener más información, consulte [Cómo: Usar el registro de actividad](#).

Visual Studio también proporciona algunos servicios integrados que no están registrados. VSPackages puede reemplazar los servicios integrados u otros proporcionando una invalidación de servicio. Solo se permite una invalidación de servicio para cualquier servicio.

Los servicios no tienen capacidad de detección. Por lo tanto, debe conocer el identificador de servicio (SID) de un servicio que desea consumir y debe saber qué interfaces proporciona. La documentación de referencia para el servicio proporciona esta información.

- VSPackages que proporcionan servicios se denominan proveedores de servicios.
- Los servicios que se proporcionan a otros VSPackages se denominan servicios globales.
- Los servicios que solo están disponibles para el VSPackage que los implementa, o para cualquier objeto que crea, se denominan servicios locales.
- Los servicios que reemplazan los servicios integrados o los servicios proporcionados por otros paquetes se denominan invalidaciones de servicio.
- Los servicios, o invalidaciones de servicio, se cargan a petición, es decir, el proveedor de servicios se carga cuando otro VSPackage solicita el servicio que proporciona.
- Para admitir la carga bajo demanda, un proveedor Visual Studio de servicios registra sus servicios globales con `IServiceProvider`. Para obtener más información, consulte [Cómo: proporcionar un servicio](#).
- Después de obtener un servicio, utilice [QueryInterface](#) (código no administrado) o conversión (código administrado) para obtener la interfaz deseada, por ejemplo:

```
TryCast(GetService(GetType(SVsActivityLog)), IVsActivityLog)
```

```
GetService(typeof(SVsActivityLog)) as IVsActivityLog;
```

- El código administrado hace referencia a un servicio por su tipo, mientras que el código no administrado hace referencia a un servicio por su GUID.
- Cuando Visual Studio carga un VSPackage, pasa un proveedor de servicios al VSPackage para dar el VSPackage acceso a los servicios globales. Esto se conoce como "sentar" el VSPackage.
- VSPackages pueden ser proveedores de servicios para los objetos que crean. Por ejemplo, un formulario puede enviar una solicitud de un servicio Visual Studio de color a su marco, que podría pasar la solicitud a `IServiceProvider`.
- Los objetos administrados que están profundamente anidados, o que no están sitios en absoluto, pueden requerir [GetGlobalService](#) acceso directo a los servicios globales.

Usar GetGlobalService

A veces es posible que necesite obtener un servicio desde una ventana de herramientas o un contenedor de control que no se ha ubicado, o bien se ha ubicado con un proveedor de servicios que no conoce el servicio que desea. Por ejemplo, es posible que desee escribir en el registro de actividad desde dentro de un control. Para obtener más información acerca de estos y otros escenarios, vea [Cómo: Solucionar problemas de servicios](#).

Puede obtener la mayoría de los [GetGlobalService](#) servicios de Visual Studio llamando al método estático.

[GetGlobalService](#) se basa en un proveedor de servicios en caché que se inicializa la primera vez que se encuentra cualquier VSPackage derivado de Package. Debe garantizar que se cumple esta condición o, de lo contrario, estar preparado para un servicio nulo.

Afortunadamente, [GetGlobalService](#) funciona correctamente la mayor parte del tiempo.

- Si un VSPackage proporciona un servicio conocido sólo para otro VSPackage, el VSPackage que solicita el servicio se encuentra antes de que se cargue el VSPackage que proporciona el servicio.
- Si un VSPackage crea una ventana de herramientas, el VSPackage se encuentra antes de crear la ventana de herramientas.
- Si un contenedor de controles está hospedado por una ventana de herramientas creada por un VSPackage, el VSPackage se encuentra antes de crear el contenedor de control.

Para obtener un servicio desde una ventana de herramientas o un contenedor de control

- Inserte este código en el constructor, la ventana de herramientas o el contenedor de control:

```
IVsActivityLog log = Package.GetGlobalService(typeof(SVsActivityLog)) as IVsActivityLog;  
if (log == null) return;
```

```
Dim log As IVsActivityLog = TryCast(Package.GetGlobalService(GetType(SVsActivityLog)), IVsActivityLog)  
If log Is Nothing Then  
    Return  
End If
```

Este código obtiene un servicio SVsActivityLog y lo convierte en una interfaz IVsActivityLog, que se puede usar para escribir en el registro de actividad. Para obtener un ejemplo, vea [Cómo: Usar el registro de actividad](#).

Vea también

- [Lista de servicios disponibles](#)
- [Uso y prestación de servicios](#)
- [Conversiones de tipos](#)
- [Conversión](#)

Lista de servicios disponibles

06/04/2020 • 26 minutes to read • [Edit Online](#)

Visual Studio y el SDK de Visual Studio admite los siguientes servicios. Algunos paquetes ofrecen sus propios servicios que no se enumeran aquí, por ejemplo, los servicios de lenguaje no tienen un único GUID de servicio. Debe usar el nombre del idioma para buscar el GUID del servicio de lenguaje en el registro.

Utilice los GUID de servicio enumerados aquí u obtenidos de alguna otra fuente (por ejemplo, servicios de idioma) para obtener la interfaz principal o las interfaces que se muestran con cada servicio.

Los Servicios

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SBindHost	IBindHost	Sí	Sí	Utilizado por VSPackages IBindHost para obtener una interfaz de un control ActiveX para facilitar las transferencias de datos asincrónicas.
SDTE	DTE	No	Sí	Obtiene el objeto de extensibilidad de tiempo de diseño (DTE) utilizado para la automatización. C/C++ ID: SID_SDTE
SCodeNavigate	ICodeNavigate	Sí	Sí	Implementado por un diseñador de formularios para mostrar el controlador de eventos predeterminado para un control.
SContainerDispatch	IDispatch	Sí	Sí	Permite un VSPackage para tener acceso a la interfaz de automatización de otro VSPackage o un control.
SExtendedTypeLib	IExtendedTypeLib	Sí	Sí	Habilita un VSPackage para agregar o crear una biblioteca de tipos extendido.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SDirList	IDirList	No	Sí	Proporciona acceso a la lista de listas con nombre de un contenedor; por ejemplo, la lista de directorios para buscar como se muestra en el cuadro de diálogo Buscar y reemplazar en la lista desplegable Buscar en. El IDirList objeto se puede leer y escribir en.
SVsPackageDynamicToolOwner	IVsPackageDynamicToolOwner	Sí	Sí	Permite que un VSPackage tenga sus propias ventanas de herramientas que se muestran u ocultan dinámicamente.
SLicensedClassManager	ILicensedClassManager	Sí	Sí	Habilita un VSPackage Visual Studio para indicar a las clases que requiere especificando una lista de claves de licencia.
SLocalRegistry	ILocalRegistry ILocalRegistry2	Sí	Sí	Permite que un VSPackage para tener Visual Studio acceso al registro en relación con el subárbol del registro local.
SOleComponentManager	IOleComponentManager	Sí	Sí	Proporciona servicios de coordinación de componentes, como bucles de mensajes, bucles de teclado y notificaciones de eventos.
SOleComponentUIManager	IOleComponentUIManager	Sí	Sí	Permite que un VSPackage tener acceso a Visual Studio los diversos elementos de interfaz de usuario (UI) de , como ayuda, barra de estado y eventos de interfaz de usuario.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SOleInPlaceComponent	IInPlaceComponent	Sí	Sí	Habilita un VSPackage para integrar su Visual Studio interfaz de usuario con la interfaz de usuario de .
SOleInPlaceComponentSite	IInPlaceComponentSite	Sí	Sí	Habilita un VSPackage para controlar los cambios de interfaz de usuario que son específicos de las herramientas.
SOleUndoManager	IUndoManager	Sí	Sí	Permite a un VSPackage tener acceso al administrador de deshacer de un contenedor para participar en la pila de deshacer de ese contenedor o para tener acceso a la pila de deshacer de ese contenedor.
SProfferService	IProfferService	Sí	Sí	Habilita un VSPackage para ofrecer sus propios servicios.
SProfferTypeLib	IProfferTypeLib	Sí	Sí	Permite a un diseñador de formularios hacer que una biblioteca de tipos esté disponible como referencia.
STrackSelection	ITrackSelection	Sí	Sí	Proporciona acceso a las selecciones en un contenedor de selección. Utilizado por un diseñador de formularios.
SUIHostCommandDispatcher	ICommandTarget	Sí	Sí	Permite que un VSPackage participe en la cadena del controlador de comandos y controle los comandos en nombre del entorno de desarrollo integrado (IDE) o en sí mismo.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SUIHostLocale	IUIHostLocale	Sí	Sí	Proporciona acceso a la información de configuración regional de la interfaz de usuario del host.
SVsActivityLog	IVsActivityLog	No	Sí	Habilita un VSPackage para registrar mensajes de alto nivel cuando el registro está activado.
SVsAddProjectItemDlg	IVsAddProjectItemDlg	Sí	Sí	Proporciona acceso a la Agregar elemento de proyecto cuadro de diálogo, lo que permite VSPackages para implementar su propia Agregar elemento opción de menú.
SVsAddWebReferenceDlg	IVsAddWebReferenceDlg	Sí	Sí	Muestra el cuadro de diálogo Agregar referencia .
SVsAppCommandLine	IVsAppCommandLine	Sí	Sí	Habilita un VSPackage para determinar si se dio un modificador de línea de comandos a devenv.exe.
SVsCallBrowser	IVsCallBrowser	No	Sí	Habilita un VSPackage para crear un nuevo explorador de llamadas utilizado en la depuración.
SVsClassView	IVsClassView	Sí	Sí	Habilita un VSPackage para sincronizar la vista de clases a un objeto específico.
SVsCmdNameMapping	IVsCmdNameMapping	Sí	Sí	Proporciona compatibilidad para asignar nombres de comandos a GUID y volver a determinar y determinar los nombres de todos los comandos y nombres disponibles.
SVsCodeDefView	IVsCodeDefView	No	Sí	Permite que un VSPackage manipule la vista de definición de código .

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsCodeShareHandler	IVsCodeShareHandler	Sí	Sí	Servicio interno. No debe usarse.
SVsCodeWindow	IVsCodeWindow	Sí	Sí	Proporciona acceso a una ventana de código que puede contener uno o varios documentos.
SVsCodeWindowManager	IVsCodeWindowManager	Sí	Sí	Habilita un VSPackage para agregar cambios a la ventana de código, como barras desplegables.
SVsCommandWindow	IVsCommandWindow IVsCommandWindow2	Sí	Sí	Permite que un VSPackage ejecute un comando a través de la ventana de comandos y, de lo contrario, interactúe con la ventana de comandos .
SVsCommandWindowCollection	IVsCommandWindowCollection	No	Sí	Permite que un VSPackage manipule Visual Studio lista de ventanas de comandos mantenidas por .
SVsComplusLibrary	IVsLibraryReferenceManager	Sí	Sí	Permite un VSPackage para proporcionar información de exploración al Examinador de objetos .
SVsComponentSelectorDlg	IVsComponentSelectorDlg	No	Sí	Habilita un VSPackage para admitir el Agregar referencia opción, que permite a un usuario seleccionar componentes externos para agregar al proyecto.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsComponentSelectorDlg2	IVsComponentSelectorDlg2	No	Sí	Habilita un VSPackage para admitir el Agregar referencia opción, que permite a un usuario seleccionar componentes externos para agregar al proyecto. Esta versión del cuadro de diálogo permite rellenar previamente la lista de componentes antes de que se muestre.
SVsConfigurationManagerDlg	IVsConfigurationManagerDlg	No	Sí	Muestra el cuadro de diálogo Administrador de configuración .
SVsCreateAggregateProject	IVsCreateAggregateProject	No	Sí	Habilita un VSPackage para crear un proyecto que contiene una colección de otros proyectos.
SVsDebuggableProtocol	IVsDebuggableProtocol	Sí	Sí	Habilita un VSPackage para actualizar la lista de protocolos depurables utilizados por el IDE para iniciar determinados motores de depuración.
SVsDebugLaunch	IVsDebugLaunch	Sí	Sí	Habilita un VSPackage para admitir el inicio de un depurador.
SVsDiscoveryService	IVsDiscoveryService	Sí	Sí	Habilita un VSPackage para crear una sesión de detección que se usa para detectar servicios web.
SVsEnumHierarchyItemsFactory	IVsEnumHierarchyItemsFactory	Sí	Sí	Proporciona un generador IVsEnumHierarchyItemsFactory para crear objetos utilizados para enumerar sobre jerarquías especificadas (proyectos).

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsErrorList	IVsErrorList	No	Sí	Proporciona métodos adicionales para manipular la ventana de tareas Generar lista de errores. En concreto, lleva la ventana de tareas Generar lista de errores a la vanguardia y fuerza que se muestren todos los errores.
SVsExternalFilesManager	IVsExternalFilesManager	Sí	Sí	Proporciona acceso al nodo de proyecto Archivos varios de la solución actual.
SVsFileChange		Sí	Sí	Obsoleto. Utilice SVsFileChangeEx el servicio en su lugar.
SVsFileChangeEx	IVsFileChangeEx	Sí	Sí	Permite un VSPackage para obtener acceso a varios eventos de cambio de archivo desencadenados por el IDE.
SVsFilterAddProjectItemDlg	IVsFilterAddProjectItemDlg	Sí	Sí	Permite un VSPackage para filtrar los elementos que aparecen en el Agregar elemento cuadro de diálogo.
SVsFilterKeys	IVsFilterKeys	Sí	Sí	Habilita un VSPackage para realizar el filtrado de teclado avanzado.
SVsFontAndColorCacheManager	IVsFontAndColorCacheManager	No	Sí	Proporciona acceso al conjunto de cachés Visual Studio para fuentes y colores para actualizar o borrar una caché determinada o todas las cachés.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsFontAndColorStorage	IVsFontAndColorStorage IVsFontAndColorUtilities	Sí	Sí	Permite que un VSPackage manipule Visual Studio la configuración de fuente y color mantenida por . Además, este servicio proporciona acceso a una colección de métodos de utilidad para manipular datos de fuente y color.
SVsGeneralOutputWindowPane	IVsOutputWindowPane	Sí	Sí	Proporciona acceso al panel ventana de salida general, creándolo según sea necesario.
SVsHelpService	IVsHelpSystem	Sí	Sí	Proporciona acceso al sistema de ayuda.
SVsHTMLConverter	IVsHTMLConverter	Sí	Sí	Utilizado por Visual Studio el depurador para controlar HTML para dar formato a su salida.
SVsIME	IVsIME	Sí	Sí	Proporciona acceso a la API del Editor de métodos de entrada (IME) desde un VSPackage.
SVsIntegratedHelp	SVsHelp	Sí	Sí	Proporciona acceso Visual Studio al sistema de ayuda para el acceso a palabras clave o URL, así como control de navegación a través de un archivo de ayuda. Este servicio solo está disponible si Visual Studio la Ayuda está integrada en el IDE y no se ejecuta como un programa externo.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsIntelliMouseHandler	IVsIntelliMouseHandler	Sí	Sí	Permite que un VSPackage obtenga acceso a la funcionalidad de IntelliMouse, como el uso de la rueda del mouse y el control de los mapas de bits de desplazamiento y panorámica cuando se hace clic en la rueda del mouse.
SVsIntellisenseEngine	IVsIntellisenseEngine	No	Sí	Permite que un nodo de jerarquía de proyectos cargue o descargue archivos como parte de la compatibilidad con operaciones de IntelliSense. El proceso de carga y descarga desencadena eventos que pueden afectar a lo que se muestra en la información sobre herramientas de IntelliSense para el proyecto.
SVsIntellisenseProjectHost	IVsIntellisenseProjectHost	No	Sí	Permite que un nodo de jerarquía de proyectos proporcione IVsIntellisenseProject información sobre los proyectos de IntelliSense anidados (que implementan la interfaz) que se pueden mostrar en una información sobre herramientas de IntelliSense.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsIntellisenseProjectManager	IVsIntellisenseProjectManager	No	Sí	Permite que un nodo de jerarquía de proyectos avise a los agentes de escucha de eventos, como cambios en las referencias o la configuración, que pueden afectar a lo que se muestra en la información sobre herramientas de IntelliSense. Diseñado para ser utilizado con idiomas contenidos.
SVsInvisibleEditorManager	IVsInvisibleEditorManager	Sí	Sí	Permite un VSPackage para registrar un editor "invisible", es decir, un editor que proporciona funcionalidad de edición completa, pero no es visible para los usuarios.
SVsLanguageFilter	IVsTextViewFilter	Sí	Sí	Habilita un VSPackage para proporcionar información adicional a la vista de texto, como sugerencias de datos y la extensión de las palabras.
SVsLaunchPad	IVsLaunchPad	Sí	Sí	Permite a un VSPackage ejecutar scripts por lotes temporales, ejecutar un programa de línea de comandos cuya salida se envía a un panel de salida y analizar los mensajes de advertencia y error estándar que se envían a una ventana de error.
SVsLaunchPadFactory	IVsLaunchPadFactory	Sí	Sí	Proporciona un generador IVsLaunchPad para crear objetos.
SVsLinkedUndoTransactionManager	IVsLinkedUndoTransactionManager	Sí	Sí	Proporciona acceso al administrador de deshacer vinculado.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsMenuEditor	IVsMenuEditorFactory	Sí	Sí	Permite que un diseñador de formularios acceda al editor de menús compartidos. IVsMenuEditorFactory se puede IVsMenuEditor consultar para .
SVsMonitorUserContext	IVsMonitorUserContext	Sí	Sí	Habilita un VSPackage para crear una "bolsa de contexto", que se usa para asociar palabras clave de ayuda para un contexto determinado.
SVsObjBrowser	IVsObjBrowser	Sí	Sí	Permite que un VSPackage navegue a un objeto específico en el Examinador de objetos .
SVsObjectManager	IVsObjectManager	Sí	Sí	Permite a un VSPackage registrar Visual Studio su administrador de biblioteca con para administrar objetos como espacios de nombres, clases y enumeraciones.
SVsObjectSearch	IVsObjectSearch	Sí	Sí	Habilita un VSPackage para buscar un objeto determinado.
SVsOpenProjectOrSolutionDlg	IVsOpenProjectOrSolutionDlg	No	Sí	Habilita un VSPackage para Visual Studio usar el cuadro de diálogo estándar para abrir un proyecto o solución.
SVsOutputWindow	IVsOutputWindow	Sí	Sí	Habilita un VSPackage para crear paneles de salida adicionales en el general salida ventana.
SVsParseCommandLine	IVsParseCommandLine	Sí	Sí	Habilita un implementador IOleCommandTarget de la interfaz para analizar las líneas de comandos.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsPathVariableResolver	IVsPathVariableResolver	No	Sí	Proporciona una manera de resolver variables que son específicas Visual Studio y que están incrustadas en las rutas de acceso para producir una ruta de acceso final.
SVsPreviewChangesService	IVsPreviewChangesService	No	Sí	Muestra el cuadro de diálogo Vista previa de cambios que se utiliza en la refactorización de código.
SVsProfileDataManager	IVsProfileDataManager	No	Sí	Proporciona acceso al administrador Visual Studio de perfiles del que permite importar y exportar datos de configuración, así como mostrar una interfaz de usuario de la configuración del perfil del usuario actual.
SVsProfilesManagerUI	IVsProfilesManagerUI	No	Sí	Muestra un cuadro de diálogo que muestra la configuración del perfil del usuario actual.
SVsPropertyPageFrame	IVsPropertyPageFrame	Sí	Sí	Habilita un VSPackage para invalidar qué página de propiedades se muestra inicialmente en el propiedades ventana.
SVsQueryEditQuerySave	IVsQueryEditQuerySave2	No	Sí	Usado por VSPackages para informar a un proveedor de control de código fuente que un archivo está a punto de cambiarse en la memoria o guardar.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsRegisterDebugTargetProvider	IVsRegisterProjectDebugTargetProvider	No	Sí	Permite que un proyecto de VSPackage invalide mediante programación el destino que se iniciará en un depurador.
SVsRegisterEditors	IVsRegisterEditors	Sí	Sí	Habilita un VSPackage para registrar un generador de editores con el IDE.
SVsRegisterFindScope	IVsRegisterFindScope	No	Sí	Habilita un VSPackage para registrar un ámbito de búsqueda para el buscar en archivos cuadro de diálogo.
SVsRegisterPriorityCommandTarget	IVsRegisterPriorityCommandTarget	Sí	Sí	Permite que un VSPackage se registre como un controlador de comandos de alta prioridad, lo que permite que el VSPackage para ver todos los comandos. Utilícelo con moderación, si es que lo haga.
SVsRegisterProjectTypes	IVsRegisterProjectTypes	Sí	Sí	Habilita un VSPackage para registrar tipos de proyecto con el IDE.
SVsResourceManager	IVsResourceManager	No	Sí	Permite un VSPackage para cargar recursos administrados y no administrados desde archivos DLL satélite.
SVsResourceView	IVsResourceView	Sí	Sí	Utilice SVsClassView el servicio en su lugar.
SVsRunningDocumentTable	IVsRunningDocumentTable	Sí	Sí	Proporciona acceso a la tabla de documentos en ejecución (RDT) del IDE que realiza un seguimiento de todos los documentos abiertos actualmente.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsSccManager	IVsSccManager2	No	Sí	Permite VSPackages para registrarse con un proveedor de control de código fuente para que puedan participar en el control de código fuente.
SVsSccToolsOptions	IVsSccToolsOptions	Sí	Sí	Habilita un VSPackage para obtener y establecer opciones de proveedor de control de código fuente.
SVsSettingsReader	IVsSettingsReader	No	Sí	Proporciona acceso de lectura a la configuración del perfil del usuario.
SVsShell	IVsShell	Sí	Sí	Permite un VSPackage para interactuar directamente con y manipular otros VSPackages.
SVsShellDebugger	IVsDebugger	Sí	Sí	Proporciona acceso Visual Studio al depurador.
SVsShellMonitorSelection	IVsMonitorSelection	Sí	Sí	Permite un VSPackage para tener acceso a la selección actual y administrar contextos de interfaz de usuario de comandos.
SVSMDCodeDomProvider	IVSMDCodeDomProvider	No	Sí	Proporciona acceso a un proveedor de modelo de objetos de documento de código (DOM) que se puede usar en código nativo.
SVSMDDesignerService	IVSMDCodeDomCreator IVSMDDesignerService	No	Sí	Proporciona acceso a la compatibilidad del IDE para diseñadores de formularios administrados. Se <code>IVSMDCodeDomCreator</code> puede usar para crear proveedores DOM de código.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVSMDPropertyBrowser	IVSMDPropertyBrowser	No	Sí	Proporciona acceso al servicio de ventanas de propiedades del diseñador.
SVSMDTypeResolutionService	IVSMDTypeResolutionService	No	Sí	Proporciona acceso a una interfaz que puede devolver un ITypeResolutionService objeto que se puede usar en código nativo.
SVsSmartOpenScope	IVsSmartOpenScope	No	Sí	Proporciona una manera de abrir un ámbito en un ensamblado, teniendo en cuenta el bloqueo según sea necesario.
SVsSolution	IVsSolution	Sí	Sí	Proporciona acceso de nivel superior a la solución actual.
SVsSolutionBuildManager	IVsSolutionBuildManager	Sí	Sí	Permite que un VSPackage interactúe con el proceso de compilación de una solución.
SVsSolutionObject	IVsSolution	Sí	Sí	Use el servicio SVsSolution en su lugar.
SVsSolutionPersistence	IVsSolutionPersistence	Sí	Sí	Habilita un VSPackage para almacenar y recuperar información del archivo .sln de la solución actual.
SVsSQLCLRReferences	IVsSQLCLRReferences	No	Sí	Proporciona la capacidad de agregar y actualizar referencias en ensamblados de código administrado.
SVsStartPageDownload	IVsStartPageDownload	No	Sí	Proporciona acceso al servicio de descarga de la página de inicio de Visual Studio 2017 para iniciar y detener el servicio de descarga en un subproceso en segundo plano.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsStatusBar	IVsStatusBar	Sí	Sí	Proporciona acceso a la barra de estado del IDE.
SVsStrongNameKeys	IVsStrongNameKeys	No	Sí	Proporciona acceso a métodos para crear nombres de clave seguros y archivos de claves con contraseñas que se usan en la firma de ensamblados de código administrado.
SVsStructuredFileIO	IVsStructuredFileIO	Sí	Sí	Habilita un VSPackage para proporcionar compatibilidad para guardar datos en varios formatos.
SVsTaskList	IVsTaskList	Sí	Sí	Proporciona acceso a la ventana Lista de tareas del IDE.
SVsTextImageUtilities	IVsTextImageUtilities	No	Sí	Proporciona utilidades para cargar y guardar archivos de texto.
SVsTextManager	IVsTextManager IVsHiddenTextManager	Sí	Sí	Proporciona acceso a todos los búferes de texto, así como a las sesiones de texto oculto (para regiones ocultas) que están disponibles en el IDE.
SVsTextOut	IVsTextOut	Sí	Sí	Proporciona una versión de <code>TextOut</code> la función Win32 para escribir texto en un contexto de dispositivo (requiere un identificador de controlador de dominio).
SVsTextSpanSet	IVsTextSpanSet	Sí	Sí	Proporciona acceso a una lista de intervalos de texto en una imagen de texto o búfer. Este servicio se implementa normalmente en un contenedor de documentos y hace referencia al documento actual.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsThreadedWaitDialog	IVsThreadedWaitDialog	No	Sí	Habilita un VSPackage para mostrar un cuadro de diálogo que espera en un subproceso diferente (utilizado para esperar tareas en segundo plano).
SVsThreadPool	IVsThreadPool	No	Sí	Habilita un VSPackage para iniciar tareas Visual Studio en segundo plano que, a continuación, se mantienen mediante .
SVsToolbox	IVsToolbox	Sí	Sí	Proporciona acceso al cuadro de herramientas del IDE.
SVsToolboxActiveXDataProvider	IVsToolboxDataProvider	Sí	Sí	Habilita un VSPackage para obtener información de elementos de Cuadro de herramientas .
SVsToolboxDataProviderRegistry	IVsToolboxDataProviderRegistry	No	Sí	Permite que un VSPackage registre un proveedor de datos de cuadro de herramientas sin incurrir en el costo de rendimiento de precargar todo el cuadro de herramientas .
SVsToolsOptions	IVsToolsOptions	No	Sí	Habilita un VSPackage para determinar si el cuadro de diálogo Opciones está abierto y para actualizar la visibilidad de todas las opciones página.
SVsTrackProjectDocuments	IVsTrackProjectDocuments2 IVsTrackProjectDocuments3	No	Sí	Permite un VSPackage para supervisar los cambios en los archivos de un proyecto y proporcionar control por lotes sobre un proveedor de control de código fuente.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsTrackSelectionEx	IVsTrackSelectionEx	Sí	Sí	Habilita un VSPackage para informar al IDE de los cambios en una selección que pueden afectar al elemento de proyecto seleccionado actualmente.
SVsUIHierWinClipboardHelper	IVsUIHierWinClipboardHelper	Sí	Sí	Permite que una jerarquía (como un proyecto VSPackage) para coordinar el uso del portapapeles con otras jerarquías.
SVsUIShell	IVsUIShell	Sí	Sí	Proporciona acceso a los elementos de interfaz de usuario del IDE, como ventanas de herramientas y ventanas de documentos.
SVsUIShellDocumentWindowMgr	IVsUIShellDocumentWindowMgr	Sí	Sí	Permite un VSPackage para restaurar las posiciones de todas las ventanas en función del contenido de una secuencia de datos o para guardar la posición de todas las ventanas en una secuencia. Casi nunca se usa.
SVsUIShellOpenDocument	IVsUIShellOpenDocument	Sí	Sí	Permite un VSPackage para abrir documentos de numerosas maneras y para determinar quién es el propietario de qué documento.
SVsUpgradeLogger	IVsUpgradeLogger	No	Sí	Utilizado por los IVsProjectUpgradeViaFactory implementadores de la interfaz para notificar mensajes de error e información.
SVsWebBrowsingService	IVsWebBrowsingService	Sí	Sí	Habilita un VSPackage para crear y controlar una sesión de exploración web.

SERVICIO	INTERFAZ	VISUAL STUDIO	VISUAL STUDIO 2005	DESCRIPCIÓN
SVsWebFavorites	IVsWebFavorites	Sí	Sí	Habilita un VSPackage para agregar a la lista de favoritos del usuario.
SVsWebPreview	IVsWebPreview	Sí	Sí	Habilita un VSPackage para obtener una vista previa de una página Web, normalmente en una ventana secundaria.
SVsWebURLMRU	IVsWebURLMRU	Sí	Sí	Habilita un VSPackage para agregar una dirección URL a la lista de direcciones URL de uso más reciente (MRU) y para obtener una lista de todas las direcciones URL de la lista MRU.
SVsWindowFrame	IVsWindowFrame	Sí	Sí	Habilita un VSPackage para obtener el marco de ventana en el que el paquete o parte del paquete podría estar situado.
SVsXMLMemberIndex Service	IVsXMLMemberIndex Service	Sí	Sí	Proporciona acceso a archivos de documentación con formato XML asociados a un archivo de metadatos determinado.

Vea también

- [Uso y prestación de servicios](#)

Extensibilidad del depurador de Visual Studio

06/04/2020 • 3 minutes to read • [Edit Online](#)

Visual Studio incluye un depurador de código fuente totalmente interactivo, que proporciona una herramienta eficaz y fácil de usar para realizar el seguimiento de errores en el programa. El depurador tiene compatibilidad completa con Visual Basic, C, C/C++ y JavaScript. Sin embargo, con el Visual Studio SDK, que está disponible en el Centro de descarga de [Microsoft](#), otros lenguajes de programación se pueden admitir en el depurador con las mismas características enriquecidas.

El Visual Studio depurador es el front-end común (es decir, la interfaz de usuario) para los componentes de depuración que, a su vez, son específicos del idioma que se está depurando. Para los nuevos idiomas, todo lo Visual Studio que es necesario para la compatibilidad con el depurador es crear los componentes back-end necesarios, como un motor de depuración (DE). Este punto es Visual Studio SDK donde entra el entrar.

Incluye Visual Studio SDK una referencia completa Visual Studio a todos los elementos necesarios para crear una nueva DE. Además, hay ejemplos y tutoriales que le ayudarán a empezar.

Para obtener una muestra completa de un sistema de proyecto de lenguaje con compatibilidad con depuración, consulte el [ejemplo IronPython](#).

En las secciones siguientes se describe Visual Studio SDK cómo ampliar el depurador mediante el archivo .

En esta sección

[Empezar](#) Describe lo Visual Studio que ofrece la depuración y cómo instalar el SDK.

Crear un motor de [depuración personalizado](#) Documenta el proceso DE personalizado, desde la preparación del programa para un DE hasta la desasociación del DE.

[Escribir un evaluador](#) de expresiones CLR Explica si debe escribir un evaluador de expresiones.

Elija una estrategia de [implementación del motor](#) de depuración Describe cómo implementar su DE.

[Referencia](#) Documenta Visual Studio la API de depuración.

[Muestras](#) Contiene vínculos a un ejemplo de evaluador de expresiones de Common Language Runtime y un ejemplo de motor de depuración.

Elija una estrategia de implementación del motor de depuración

06/04/2020 • 3 minutes to read • [Edit Online](#)

Utilice la arquitectura en tiempo de ejecución para determinar la estrategia de implementación del motor de depuración (DE). Puede crear el motor de depuración en proceso en el programa que está depurando. Cree el motor de depuración en proceso en el Administrador de depuración de sesión (SDM) de Visual Studio. O bien, cree el motor de depuración fuera de proceso para ambos. Las siguientes pautas deben ayudarlo a elegir entre estas tres estrategias.

Directrices

Si bien es posible que el DE esté fuera de proceso tanto para el SDM como para el programa que está depurando, normalmente no hay razón para hacerlo. Las llamadas a través de los límites del proceso son relativamente lentas.

Los motores de depuración ya se proporcionan para el entorno de tiempo de ejecución nativo de Win32 y para el entorno de tiempo de ejecución de lenguaje común. Si usted tiene que substituir el DE para cualquier entorno, usted debe crear el DE en proceso con el SDM.

De lo contrario, puede crear el DE en proceso en el SDM o en proceso en el programa que está depurando. Deberá tener en cuenta si el evaluador de expresiones de la DE requiere acceso frecuente al almacén de símbolos del programa. O bien, si el almacén de símbolos se puede cargar en la memoria para un acceso rápido. Además, considere los siguientes enfoques:

- Si no hay muchas llamadas entre el evaluador de expresiones y el almacén de símbolos, o si el almacén de símbolos se puede leer en el espacio de memoria SDM, cree el DE en proceso en el SDM. Debe devolver el CLSID del motor de depuración al SDM cuando se asocia al programa. El SDM utiliza este CLSID para crear una instancia en proceso del DE.
- Si el DE debe llamar al programa para acceder al almacén de símbolos, cree el DE en proceso con el programa. En este caso, el programa crea la instancia de LA.

Vea también

- [Extensibilidad del depurador de Visual Studio](#)

Ejemplos de depuración de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

Dos ejemplos muestran cómo ampliar Visual Studio SDKel depurador de Visual Studio mediante .

- El ejemplo del evaluador de expresiones del Visual Studio SDKdepurador, para los lenguajes administrados, se incluye en el archivo . En Visual Studio SDK la ventana, haga clic en la pestaña **Ejemplos** , haga clic en **Todo** o en **C-** en la lista de idiomas y, a continuación, haga clic en **Depurador**.
- El ejemplo del motor de depuración de Visual Studio, para idiomas nativos, está disponible en MSDN Code Gallery. Para obtener más información, vea Ejemplo del motor de [depuración](#) de Visual Studio en el sitio Web de Microsoft.

Crear un motor de depuración personalizado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un motor de depuración (DE) es un componente que permite la depuración de arquitecturas en tiempo de ejecución determinadas. Normalmente, solo hay una implementación DE por entorno en tiempo de ejecución.

NOTE

Aunque hay implementaciones DE independientes para Transact-SQL y JScript, VBScript y JScript comparten una sola DE.

Un DE trabaja con el intérprete o el sistema operativo para proporcionar servicios de depuración como control de ejecución, puntos de interrupción y evaluación de expresiones. Estos servicios se implementan a través de las interfaces DE y pueden hacer que el depurador pase entre diferentes modos operativos. Para obtener más información, consulte [Modos operativos](#).

La creación de un DE consta de los siguientes pasos:

1. Registrar un DE con Visual Studio
2. Habilitar la depuración de un programa
3. Implementar el control de ejecución y la evaluación del estado
4. Envío de eventos
5. Configurar la terminación y desmontaje

En esta sección

[Registrar un motor de depuración personalizado](#) Explica los pasos necesarios para registrar un motor de depuración con Visual Studio para que se pueda usar.

[Habilitar la depuración de un programa](#) Explica que antes de que su DE pueda depurar un programa, primero debe iniciar el DE o adjuntarlo a un programa existente.

[Implementar el control de ejecución y la evaluación del estado](#) Describe por qué la depuración de una aplicación requiere la implementación de características de control de ejecución.

[Enviar eventos](#) Describe la comunicación entre el depurador y la DE como un modelo de eventos basado en DCOM.

[Configurar la terminación y desmontaje](#) Explica cómo lograr la terminación normal, lo que significa que no hay puntos de interrupción, excepciones, errores en tiempo de ejecución o bucles infinitos en la aplicación que se va a depurar.

[Eventos del depurador de llamadas](#) Documenta el orden de llamada de los eventos que se producen en una sesión de depuración.

[Cómo: Depurar un motor de depuración personalizado](#) Explica cómo depurar un DE personalizado.

Vea también

- [Extensibilidad del depurador de Visual Studio](#)

Registrar un motor de depuración personalizado

06/04/2020 • 3 minutes to read • [Edit Online](#)

El motor de depuración debe registrarse como un generador de clases, siguiendo las convenciones COM, así como registrarse con Visual Studio a través de la subclave del Registro de Visual Studio.

NOTE

Puede encontrar un ejemplo de cómo registrar un motor de depuración en el ejemplo `TextInterpreter`, que se compila como parte del Tutorial: Creación de un motor de [depuración mediante ATL COM](#).

Proceso del servidor DLL

Normalmente, un motor de depuración se configura en su propio archivo DLL como un servidor COM. Como tal, el motor de depuración debe registrar el CLSID de su generador de clases con COM antes de que Visual Studio pueda tener acceso a él. A continuación, el motor de depuración debe registrarse con Visual Studio para establecer las propiedades (también conocidas como métricas) que admite el motor de depuración. La elección de las métricas escritas en la subclave del Registro de Visual Studio depende de las características que admite el motor de depuración.

[Las aplicaciones auxiliares](#) del SDK para la depuración describen no solo las ubicaciones del Registro necesarias para registrar un motor de depuración; también describe la biblioteca *dbgmetric.lib*, que contiene una serie de funciones y declaraciones útiles para los desarrolladores de C++ que facilitan la manipulación del registro.

Ejemplo

En el ejemplo siguiente (del ejemplo `TextInterpreter`) se muestra cómo utilizar la `SetMetric` función (de *dbgmetric.lib*), para registrar un motor de depuración con Visual Studio. Las métricas que se pasan también se definen en *dbgmetric.lib*.

NOTE

`TextInterpreter` es un motor de depuración básico; no configura (y por lo tanto no registra) ninguna otra característica. Un motor de depuración más `SetMetric` completo tendría una lista completa de llamadas o su equivalente, una para cada característica que admite el motor de depuración.

```
// Define base registry subkey to Visual Studio.
static const WCHAR strRegistrationRoot[] = L"Software\\Microsoft\\VisualStudio\\8.0";

HRESULT CTextInterpreterModule::RegisterServer(BOOL bRegTypeLib, const CLSID * pCLSID)
{
    SetMetric(metricTypeEngine, __uuidof(Engine), metricName, L"Text File", false, strRegistrationRoot);
    SetMetric(metricTypeEngine, __uuidof(Engine), metricCLSID, CLSID_Engine, false, strRegistrationRoot);
    SetMetric(metricTypeEngine, __uuidof(Engine), metricProgramProvider, CLSID_MsProgramProvider, false,
strRegistrationRoot);

    return base::RegisterServer(bRegTypeLib, pCLSID);
}
```

Vea también

- [Creación de un motor de depuración personalizado](#)
- [Ayudantes del SDK para la depuración](#)
- [Tutorial: Creación de un motor de depuración mediante ATL COM](#)

Habilitar la depuración de un programa

06/04/2020 • 2 minutes to read • [Edit Online](#)

Antes de que el motor de depuración (DE) pueda depurar un programa, primero debe iniciar el DE o adjuntarlo a un programa existente.

En esta sección

[Obtener un puerto](#) Describe cómo obtener un puerto como primer paso para habilitar la depuración de un programa.

[Registrar el programa](#) Explica el siguiente paso para habilitar la depuración de un programa: registrarlo con el puerto. Una vez registrado, el programa se puede depurar mediante el proceso de asociación o depuración Just-In-Time (JIT).

[Adjuntar al programa](#) Explica el siguiente paso: adjuntar el depurador al programa.

[Colocación basada en el lanzamiento](#) Describe los datos adjuntos basados en el inicio a un programa, que es automático al iniciarse por el SDM.

[Enviar los eventos requeridos](#) Le guía a través de los eventos necesarios al crear un motor de depuración (DE) y adjuntarlo a un programa.

Secciones relacionadas

[Creación de un motor](#) de depuración personalizado Define un motor de depuración (DE) y describe los servicios implementados a través de las interfaces DE y cómo pueden hacer que el depurador pase entre diferentes modos operativos.

Obtener un puerto

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un puerto representa una conexión a una máquina en la que se ejecutan los procesos. Esa máquina podría ser la máquina local o una máquina remota (que posiblemente podría estar ejecutando un sistema operativo no basado en Windows; consulte [Puertos](#) para obtener más información).

Un puerto se representa mediante el [IDebugPort2](#) interfaz. Se utiliza para obtener información sobre los procesos que se ejecutan en la máquina a la que está conectado el puerto.

Un motor de depuración necesita acceso a un puerto para registrar los nodos de programa con el puerto y satisfacer las solicitudes de información del proceso. Por ejemplo, si el motor de depuración implementa la interfaz [IDebugProgramProvider2](#), la implementación del método [GetProviderProcessData](#) podría solicitar al puerto la información de proceso necesaria que se va a devolver.

Visual Studio proporciona el puerto necesario al motor de depuración y obtiene este puerto de un proveedor de puertos. Si se adjunta un programa (ya sea desde el depurador o debido a una excepción, que desencadena el cuadro de diálogo Just-in-Time [JIT]), se da al usuario la opción de transporte (otro nombre para un proveedor de puertos) para usar. De lo contrario, si el usuario inicia el programa desde el depurador, el sistema del proyecto especifica el proveedor de puertos que se va a utilizar. En cualquier caso, Visual Studio crea una instancia del proveedor de puertos, representado por un [IDebugPortSupplier2](#) interfaz y solicita un nuevo puerto mediante una llamada a [AddPort](#) con un [IDebugPortRequest2](#) interfaz. Este puerto entonces se pasa al motor de depuración de una forma u otra.

Ejemplo

Este fragmento de código muestra cómo utilizar el puerto proporcionado a [LaunchSuspended](#) para registrar un nodo de programa en [ResumeProcess](#). Los parámetros no directamente relacionados con este concepto se han omitido para mayor claridad.

NOTE

En este ejemplo se usa el puerto para iniciar y reanudar el proceso y se supone que la interfaz [IDebugPortEx2](#) se implementa en el puerto. Esta no es de ninguna manera la única manera de realizar estas tareas, y es posible que el puerto ni siquiera esté involucrado que no sea tener el programa [IDebugProgramNode2](#) dado a él.

```

// This is an IDebugEngineLaunch2 method.
HRESULT CDebugEngine::LaunchSuspended(/* omitted parameters */,
                                       IDebugPort2 *pPort,
                                       /* omitted parameters */,
                                       IDebugProcess2**ppDebugProcess)
{
    // do stuff here to set up for a launch (such as handling the other parameters)
    ...

    // Now get the IPortNotify2 interface so we can register a program node
    // in CDebugEngine::ResumeProcess.
    CComPtr<IDebugDefaultPort2> spDefaultPort;
    HRESULT hr = pPort->QueryInterface(&spDefaultPort);
    if (SUCCEEDED(hr))
    {
        CComPtr<IDebugPortNotify2> spPortNotify;
        hr = spDefaultPort->GetPortNotify(&spPortNotify);
        if (SUCCEEDED(hr))
        {
            // Remember the port notify so we can use it in ResumeProcess.
            m_spPortNotify = spPortNotify;

            // Now launch the process in a suspended state and return the
            // IDebugProcess2 interface
            CComPtr<IDebugPortEx2> spPortEx;
            hr = pPort->QueryInterface(&spPortEx);
            if (SUCCEEDED(hr))
            {
                // pass on the parameters we were given (omitted here)
                hr = spPortEx->LaunchSuspended(/* omitted parameters */,ppDebugProcess)

            }
        }
    }
    return(hr);
}

HRESULT CDebugEngine::ResumeProcess(IDebugProcess2 *pDebugProcess)
{
    // Make a program node for this process
    HRESULT hr;
    CComPtr<IDebugProgramNode2> spProgramNode;
    hr = this->GetProgramNodeForProcess(pProcess, &spProgramNode);
    if (SUCCEEDED(hr))
    {
        hr = m_spPortNotify->AddProgramNode(spProgramNode);
        if (SUCCEEDED(hr))
        {
            // resume execution of the process using the port given to us earlier.
            // (Querying for the IDebugPortEx2 interface is valid here since
            // that's how we got the IDebugPortNotify2 interface in the first place.)
            CComPtr<IDebugPortEx2> spPortEx;
            hr = m_spPortNotify->QueryInterface(&spPortEx);
            if (SUCCEEDED(hr))
            {
                hr = spPortEx->ResumeProcess(pDebugProcess);
            }
        }
    }
    return(hr);
}

```

Vea también

- [Registro del programa](#)
- [Habilitación de un programa que se va a depurar](#)

- Proveedores portuarios
- Puertos

Registrar el programa

06/04/2020 • 3 minutes to read • [Edit Online](#)

Después de que el motor de depuración ha adquirido un puerto, representado por una interfaz [IDebugPort2](#), el siguiente paso para permitir que se va a depurar el programa es registrarlo con el puerto. Una vez registrado, el programa está disponible para la depuración por uno de los siguientes medios:

- El proceso de asociación, que permite al depurador obtener el control de depuración completo de una aplicación en ejecución.
- Depuración Just-In-Time (JIT), que permite la depuración posterior al hecho de un programa que se ejecuta independientemente de un depurador. Cuando la arquitectura en tiempo de ejecución detecta un error, se notifica al depurador antes de que el sistema operativo o el entorno de tiempo de ejecución libere la memoria y los recursos del programa con errores.

Procedimiento de registro

Para registrar su programa

1. Llame al método [AddProgramNode](#) implementado por el puerto.

`IDebugPortNotify2::AddProgramNode` requiere un puntero a un [IDebugProgramNode2](#) interfaz.

Normalmente, cuando el sistema operativo o el entorno en tiempo de ejecución carga un programa, crea el nodo del programa. Si se pide al motor de depuración (DE) que cargue el programa, la DE crea y registra el nodo del programa.

En el ejemplo siguiente se muestra el motor de depuración que inicia el programa y lo registra con un puerto.

NOTE

Este ejemplo de código no es la única manera de iniciar y reanudar un proceso; este código es principalmente un ejemplo de registro de un programa con un puerto.

```

// This is an IDebugEngineLaunch2 method.
HRESULT CDebugEngine::LaunchSuspended(/* omitted parameters */,
                                       IDebugPort2 *pPort,
                                       /* omitted parameters */,
                                       IDebugProcess2**ppDebugProcess)
{
    // do stuff here to set up for a launch (such as handling the other parameters)
    ...

    // Now get the IPortNotify2 interface so we can register a program node
    // in CDebugEngine::ResumeProcess.
    CComPtr<IDebugDefaultPort2> spDefaultPort;
    HRESULT hr = pPort->QueryInterface(&spDefaultPort);
    if (SUCCEEDED(hr))
    {
        CComPtr<IDebugPortNotify2> spPortNotify;
        hr = spDefaultPort->GetPortNotify(&spPortNotify);
        if (SUCCEEDED(hr))
        {
            // Remember the port notify so we can use it in ResumeProcess.
            m_spPortNotify = spPortNotify;

            // Now launch the process in a suspended state and return the
            // IDebugProcess2 interface
            CComPtr<IDebugPortEx2> spPortEx;
            hr = pPort->QueryInterface(&spPortEx);
            if (SUCCEEDED(hr))
            {
                // pass on the parameters we were given (omitted here)
                hr = spPortEx->LaunchSuspended(/* omitted parameters */,ppDebugProcess)
            }
        }
    }
    return(hr);
}

HRESULT CDebugEngine::ResumeProcess(IDebugProcess2 *pDebugProcess)
{
    // Make a program node for this process
    HRESULT hr;
    CComPtr<IDebugProgramNode2> spProgramNode;
    hr = this->GetProgramNodeForProcess(pProcess, &spProgramNode);
    if (SUCCEEDED(hr))
    {
        hr = m_spPortNotify->AddProgramNode(spProgramNode);
        if (SUCCEEDED(hr))
        {
            // resume execution of the process using the port given to us earlier.
            // (Querying for the IDebugPortEx2 interface is valid here since
            // that's how we got the IDebugPortNotify2 interface in the first place.)
            CComPtr<IDebugPortEx2> spPortEx;
            hr = m_spPortNotify->QueryInterface(&spPortEx);
            if (SUCCEEDED(hr))
            {
                hr = spPortEx->ResumeProcess(pDebugProcess);
            }
        }
    }
    return(hr);
}

```

Vea también

- [Conseguir un puerto](#)

- [Habilitación de un programa que se va a depurar](#)

Adjuntar al programa

06/04/2020 • 4 minutes to read • [Edit Online](#)

Después de registrar los programas con el puerto adecuado, debe adjuntar el depurador al programa que desea depurar.

Elija cómo adjuntar

Hay tres maneras en que el administrador de depuración de sesión (SDM) intenta asociarse al programa que se está depurando.

1. Para los programas que inicia el motor de depuración a través del método [LaunchSuspended](#) (típico de los lenguajes interpretados, por ejemplo), el SDM obtiene la interfaz [IDebugProgramNodeAttach2](#) del objeto [IDebugProgramNode2](#) asociado al programa al que se adjunta. Si el SDM [IDebugProgramNodeAttach2](#) puede obtener la interfaz, el SDM entonces llama al método [OnAttach](#). El [IDebugProgramNodeAttach2::OnAttach](#) método [S_OK](#) vuelve a indicar que no se adjuntó al programa y que se pueden realizar otros intentos para adjuntar al programa.
2. Si el SDM puede obtener el [IDebugProgramEx2](#) interfaz desde el programa que se adjunta a, el SDM llama a la [Attach](#) método. Este enfoque es típico de los programas que fueron lanzados remotamente por el proveedor del puerto.
3. Si el programa no [IDebugProgramNodeAttach2::OnAttach](#) se [IDebugProgramEx2::Attach](#) puede asociar a través de los métodos o, el SDM carga el motor de depuración (si aún no está cargado) llamando a la [CoCreateInstance](#) función y, a continuación, llama al método [Attach](#). Este enfoque es típico de los programas lanzados localmente por un proveedor portuario.

También es posible que un proveedor [IDebugEngine2::Attach](#) de puertos personalizado llame al [IDebugProgramEx2::Attach](#) método en la implementación del método por parte del proveedor de puertos personalizado. Normalmente, en este caso, el proveedor de puertos personalizado inicia el motor de depuración en el equipo remoto.

El archivo adjunto se logra cuando el administrador de depuración de sesión (SDM) llama al método [Attach](#).

Si ejecuta la DE en el mismo proceso que la aplicación que se va a depurar, debe implementar los siguientes métodos de [IDebugProgramNode2](#):

- [GetHostName](#)
- [GetHostPid](#)
- [GetProgramName](#)

Después [IDebugEngine2::Attach](#) de llamar al método, siga [IDebugEngine2::Attach](#) estos pasos en la implementación del método:

1. Enviar un [IDebugEngineCreateEvent2](#) objeto de evento al SDM. Para obtener más información, consulte [Envío de eventos](#).
2. Llame a la [GetProgramId](#) método en el [IDebugProgram2](#) objeto que se pasó al [IDebugEngine2::Attach](#) método.

Esto devuelve [GUID](#) un que se utiliza para identificar el programa. Debe [GUID](#) almacenarse en el objeto que representa el programa local a la [IDebugProgram2::GetProgramId](#) DE y [IDebugProgram2](#) debe devolverse

cuando se llama al método en la interfaz.

NOTE

Si implementa `IDebugProgramNodeAttach2` la interfaz, `GUID` el programa `IDebugProgramNodeAttach2::OnAttach` se pasa al método. Esto `GUID` se utiliza para `GUID` el programa `IDebugProgram2::GetProgramId` devuelto por el método.

3. Enviar un `IDebugProgramCreateEvent2` objeto de evento para `IDebugProgram2` notificar al SDM que el objeto local se creó para representar el programa a la DE. Para obtener más información, consulte [Envío de eventos](#).

NOTE

No es el `IDebugProgram2` mismo objeto que `IDebugEngine2::Attach` se pasó al método. El objeto `IDebugProgram2` pasado anteriormente solo lo reconoce el puerto y es un objeto independiente.

Vea también

- [Archivo adjunto basado en el lanzamiento](#)
- [Envío de eventos](#)
- [LaunchSuspended](#)
- [IDebugProgram2](#)
- [IDebugProgramCreateEvent2](#)
- [IDebugProgramNodeAttach2](#)
- [OnAttach](#)
- [IDebugProgramNode2](#)
- [GetProgramId](#)
- [IDebugProgramEx2](#)
- [Attach](#)
- [Attach](#)

Archivo adjunto basado en el lanzamiento

06/04/2020 • 2 minutes to read • [Edit Online](#)

El archivo adjunto basado en el inicio a un programa es automático. Cuando el SDM inicia el proceso que hospeda el programa, los datos adjuntos basados en el lanzamiento siguen una ruta similar a la del método de datos adjuntos manual. Para obtener más información, consulte [Adjuntar al programa](#).

El proceso de fijación

La principal diferencia es la secuencia de eventos que siguen a la llamada **Attach**, como se indica a continuación:

1. Enviar un `IDebugEngineCreateEvent2` objeto de evento al SDM. Para obtener más información, consulte [Enviar eventos](#).
2. Llame `IDebugProgram2::GetProgramId` al método en el `IDebugProgram2` interfaz pasada a la **Attach** método.
3. Enviar un `IDebugProgramCreateEvent2` objeto de evento para notificar al SDM que el local `IDebugProgram2` objeto se creó para representar el programa a la DE.
4. Enviar un `IDebugThreadCreateEvent2` objeto de evento para notificar al SDM que se crea un nuevo subproceso para el proceso que se inició.

Vea también

- [Enviar los eventos requeridos](#)
- [Habilitar la depuración de un programa](#)

Enviar los eventos requeridos

06/04/2020 • 2 minutes to read • [Edit Online](#)

Use este procedimiento para enviar eventos necesarios.

Proceso para el envío de eventos requeridos

Los siguientes eventos son necesarios, en este orden, al crear un motor de depuración (DE) y adjuntarlo a un programa:

1. Envíe un objeto de evento [IDebugEngineCreateEvent2](#) al Administrador de depuración de sesión (SDM) cuando se inicialice la DE para depurar uno o varios programas en un proceso.
2. Cuando se adjunta el programa que se va a depurar, envíe un [IDebugProgramCreateEvent2](#) objeto de evento al SDM. Este evento puede ser un evento de parada, dependiendo del diseño del motor.
3. Si el programa está asociado a cuando se inicia el proceso, envíe un [IDebugThreadCreateEvent2](#) objeto de evento al SDM para notificar al IDE del nuevo subproceso. Este evento puede ser un evento de parada, dependiendo del diseño del motor.
4. Enviar un [IDebugLoadCompleteEvent2](#) objeto de evento al SDM cuando el programa que se está depurando ha terminado de cargarse o cuando se completa la asociación al programa. Este evento debe ser un evento de detención.
5. Si se inicia la aplicación que se va a depurar, envíe un objeto de evento [IDebugEntryPointEvent2](#) al SDM cuando se ejecute la primera instrucción de código en la arquitectura en tiempo de ejecución. Este evento siempre es un evento de detención. Al entrar en la sesión de depuración, el IDE se detiene en este evento.

NOTE

Muchos lenguajes utilizan inicializadores globales o funciones externas precompiladas (desde la biblioteca DE CRT o `_Main`) al principio de su código. Si el idioma del programa que está depurando contiene cualquiera de estos tipos de elementos antes del punto de entrada `main` inicial, `winMain` este código se ejecuta y se envía el evento de punto de entrada cuando se alcanza el punto de entrada del usuario, como `main` o `.`.

Vea también

- [Habilitación de un programa que se va a depurar](#)

Control de ejecución y evaluación del estado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Depurar una aplicación requiere implementar características de control de ejecución como entrar en funciones, detener en puntos de interrupción y continuar la ejecución. La depuración de Visual Studio basa su control de ejecución en eventos enviados entre componentes del depurador.

En esta sección

Control del programa Enumera las siguientes rutinas que se producen en el nivel de programa: establecer la siguiente instrucción, ejecutar, paso a paso, continuar, suspender y reanudar.

Métodos relacionados con el punto de interrupción Define los tipos enlazados y pendientes de puntos de interrupción que admite Visual Studio.

Evaluación de pila de llamadas Describe la implementación de los métodos que permiten la visualización de los marcos de pila de la pila de llamadas durante el modo de interrupción.

Evaluación de expresiones Explica cómo el motor de depuración (DE), la evaluación de expresiones (EE) y el administrador de depuración de sesión participan en el análisis y la evaluación de una expresión introducida en una de las ventanas del IDE.

Eventos de control Describe la interfaz utilizada para enviar eventos durante la ejecución controlada del programa.

Control del programa

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la depuración de Visual Studio, todas las siguientes rutinas paso a paso y continuas se producen en el nivel de programa:

- Establecer la siguiente instrucción, es decir, establecer el equipo en la siguiente instrucción que se ejecutará en un entorno de marco determinado
- Ejecutando, es decir, continuando saliendo del modo de paso a paso
- Pasar a la siguiente instrucción
- Continuar con el modo de paso actual
- Suspensión de los hilos contenidos por el programa
- Reanudación de los hilos contenidos en el programa

NOTE

La visualización de la pila de llamadas se implementa en el nivel de subproceso. Para enumerar la información de marco al ver la pila de llamadas para un subproceso, debe implementar todos los métodos de la [IEnumDebugFrameInfo2](#) interfaz.

Métodos de control del programa

En la tabla siguiente se muestran los métodos de [IDebugProgram2](#) que se deben implementar para un motor de depuración mínimamente funcional (DE) y control de ejecución.

MÉTODO	DESCRIPCIÓN
IDebugProgram2::Execute	Continúa ejecutando todos los subprocesos contenidos en un programa desde un estado detenido. Necesario para el control de ejecución.
IDebugProgram2::Continue	Continúa ejecutando todos los subprocesos contenidos en un programa desde un estado detenido. Necesario para el control de ejecución.
IDebugProgram2::Step	Realiza un paso en el subproceso especificado. Continúa ejecutando todos los demás subprocesos contenidos en el programa. Necesario para el control de ejecución.

Para los programas multiproceso, también debe implementar el [IDebugProgram2::EnumThreads](#) método y todos los métodos de la [IEnumDebugThreads2](#) interfaz.

Vea también

- [Control de ejecución y evaluación del estado](#)

Métodos relacionados con el punto de interrupción

06/04/2020 • 5 minutes to read • [Edit Online](#)

Un motor de depuración (DE) debe admitir la configuración de puntos de interrupción. La depuración de Visual Studio admite los siguientes tipos de puntos de interrupción:

- Bound

Solicitado a través de la interfaz de usuario y enlazado correctamente a una ubicación de código especificada

- Pending

Solicitado a través de la interfaz de usuario, pero aún no vinculado a las instrucciones reales

Discusión

Por ejemplo, se produce un punto de interrupción pendiente cuando las instrucciones aún no están cargadas. Cuando se carga el código, los puntos de interrupción pendientes intentan enlazar se enlazan al código en la ubicación prescrita, es decir, para insertar instrucciones de interrupción en el código. Los eventos se envían al administrador de depuración de sesión (SDM) para indicar el enlace correcto o para notificar que hubo errores de enlace.

Un punto de interrupción pendiente también administra su propia lista interna de puntos de interrupción enlazados correspondientes. Un punto de interrupción pendiente puede provocar la inserción de muchos puntos de interrupción en el código. La interfaz de usuario de depuración de Visual Studio muestra una vista de árbol de los puntos de interrupción pendientes y sus puntos de interrupción enlazados correspondientes.

La creación y el uso de puntos de interrupción pendientes requieren la implementación del método [IDebugEngine2::CreatePendingBreakpoint](#), así como los métodos siguientes de las interfaces [IDebugPendingBreakpoint2](#).

MÉTODO	DESCRIPCIÓN
CanBind	Determina si un punto de interrupción pendiente especificado puede enlazarse a una ubicación de código.
Bind	Enlaza un punto de interrupción pendiente especificado a una o varias ubicaciones de código.
GetState	Obtiene el estado de un punto de interrupción pendiente.
GetBreakpointRequest	Obtiene la solicitud de punto de interrupción utilizada para crear un punto de interrupción pendiente.
Habilitar	Alterna el estado habilitado de un punto de interrupción pendiente.
EnumBoundBreakpoints	Enumera todos los puntos de interrupción enlazados desde un punto de interrupción pendiente.

MÉTODO	DESCRIPCIÓN
EnumErrorBreakpoints	Enumera todos los puntos de interrupción de error resultantes de un punto de interrupción pendiente.
Eliminar	Elimina un punto de interrupción pendiente y todos los puntos de interrupción enlazados desde él.

Para enumerar los puntos de interrupción enlazados y los puntos de interrupción de error, debe implementar todos los métodos de [IEnumDebugBoundBreakpoints2](#) y de [IEnumDebugErrorBreakpoints2](#).

Los puntos de interrupción pendientes que se enlazan a una ubicación de código requieren la implementación de los siguientes [métodos IDebugBoundBreakpoint2](#).

MÉTODO	DESCRIPCIÓN
GetPendingBreakpoint	Obtiene el punto de interrupción pendiente que contiene un punto de interrupción.
GetState	Obtiene el estado de un punto de interrupción enlazado.
GetBreakpointResolution	Obtiene la resolución de punto de interrupción que describe un punto de interrupción.
Habilitar	Habilita o deshabilita un punto de interrupción.
Eliminar	Elimina un punto de interrupción enlazado.

La resolución y la información de solicitud requieren la implementación de los siguientes [métodos IDebugBreakpointResolution2](#).

MÉTODO	DESCRIPCIÓN
GetBreakpointType	Obtiene el tipo del punto de interrupción representado por una resolución.
GetResolutionInfo	Obtiene la información de resolución de punto de interrupción que describe un punto de interrupción.

La resolución de errores que pueden producirse durante el enlace requiere la implementación de los siguientes [IDebugErrorBreakpoint2](#) métodos.

MÉTODO	DESCRIPCIÓN
GetPendingBreakpoint	Obtiene el punto de interrupción pendiente que contiene un punto de interrupción de error.
GetBreakpointResolution	Obtiene la resolución de error de punto de interrupción que describe un punto de interrupción de error.

La resolución de errores que pueden producirse durante el enlace también requiere los siguientes métodos de [IDebugErrorBreakpointResolution2](#).

MÉTODO	DESCRIPCIÓN
GetBreakpointType	Obtiene el tipo de un punto de interrupción.
GetResolutionInfo	Obtiene la información de resolución de un punto de interrupción.

Para ver el código fuente en un punto de interrupción es necesario implementar los métodos de [IDebugStackFrame2::GetDocumentContext](#) y/o los métodos de [IDebugStackFrame2::GetCodeContext](#).

Vea también

- [Control de ejecución y evaluación del estado](#)

Evaluación de pila de llamadas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para ver los marcos de pila de la pila de llamadas durante el modo de interrupción, debe implementar el [enumFrameInfo](#) método.

Métodos de evaluación

Para un motor de depuración simple (DE), puede haber solamente una trama de pila. Para examinar el marco de pila durante el modo de interrupción, debe implementar los métodos siguientes de [IDebugStackFrame2](#).

MÉTODO	DESCRIPCIÓN
GetCodeContext	Obtiene el contexto de código para un marco de pila. El contexto de código representa el puntero de instrucción actual en un marco de pila.
GetDocumentContext	Obtiene el contexto del documento para un marco de pila. El contexto del documento representa la ubicación actual en el código fuente de un marco de pila. Necesario para ver el código fuente cuando se detiene en un programa.

Estos métodos requieren la implementación de varias interfaces y métodos relacionados con el contexto. Por lo tanto, debe implementar el [método GetDocumentContext](#) y los métodos siguientes de [IDebugDocumentContext2](#).

MÉTODO	DESCRIPCIÓN
GetStatementRange	Obtiene el intervalo de instrucciones de archivo de un contexto de documento.

Para enumerar contextos de código, debe implementar todos los métodos de [IEnumDebugCodeContexts2](#).

Vea también

- [Control de ejecución y evaluación del estado](#)

Evaluación de expresiones (SDK de depuración de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Durante el modo de interrupción, el IDE debe evaluar expresiones simples que impliquen varias variables de programa. Para realizar su evaluación, el motor de depuración (DE) debe analizar y evaluar una expresión que se ha especificado en una de las ventanas del IDE.

Las expresiones se crean con el [IDebugExpressionContext2::ParseText](#) método y representado por el resultante [IDebugExpression2](#) interfaz.

El [IDebugExpression2](#) interfaz se implementa mediante el DE y llama a su [EvalAsync](#) método para devolver un [IDebugProperty2](#) interfaz al IDE, con el fin de mostrar los resultados de la evaluación de expresión en el IDE. [IDebugProperty2::GetPropertyInfo](#) devuelve una estructura que se usa para colocar el valor de una expresión en un **inspección** ventana o en el **locales** ventana.

El paquete de depuración o el administrador de depuración de sesión (SDM) llama a [IDebugExpression2::EvaluateAsync](#) o [EvaluateSync](#) para obtener una interfaz [IDebugProperty2](#) que representa el resultado de la evaluación. [IDebugProperty2](#) tiene métodos que devuelven el nombre, el tipo y el valor de la expresión. Esta información aparece en varias ventanas del depurador.

Uso de la evaluación de expresiones

Para usar la evaluación de expresiones, debe implementar el [método IDebugExpressionContext2::ParseText](#) y todos los métodos de la interfaz [IDebugExpression2](#), como se muestra en la tabla siguiente.

MÉTODO	DESCRIPCIÓN
EvaluateAsync	Evalúa una expresión de forma asincrónica.
Aborta	Finaliza la evaluación de expresiones asincrónicas.
EvaluateSync	Evalúa una expresión de forma sincrónica.

La evaluación sincrónica y asincrónica requiere implementar el [método IDebugProperty2::GetPropertyInfo](#). La evaluación de expresiones asincrónicas requiere la implementación de [IDebugExpressionEvaluationCompleteEvent2](#).

Vea también

- [Control de ejecución y evaluación del estado](#)

Eventos de control

06/04/2020 • 2 minutes to read • [Edit Online](#)

Debe enviar eventos durante la ejecución controlada del programa. Todos los eventos se envían mediante el [IDebugEvent2](#) interfaz y tienen atributos que requieren implementar el [IDebugEvent2::GetAttributes](#) método.

Métodos adicionales

Algunos eventos requieren la implementación de métodos adicionales, como se indica a continuación:

- Enviar el [IDebugEngineCreateEvent2](#) interfaz cuando se inicializa el motor de depuración (DE) requiere que implemente el [IDebugEngineCreateEvent2::GetEngine](#) método.
- El control de ejecución requiere la implementación de eventos de control como el [IDebugBreakEvent2](#) y [IDebugStepCompleteEvent2](#) interfaces. [IDebugBreakEvent2](#) solo es necesario para los saltos asíncronos.
- Paso a paso en funciones requiere la implementación de la [IDebugStepCompleteEvent2](#) interfaz y sus métodos.

Los eventos derivados de puntos de interrupción requieren la implementación de las interfaces [IDebugBreakpointErrorEvent2](#), [IDebugBreakpointEvent2](#) y [IDebugBreakpointBoundEvent2](#), así como los métodos [IDebugBreakpointBoundEvent2::GetPendingBreakpoint](#) y [EnumBoundBreakpoints](#).

La evaluación de expresiones asíncronas requiere que implemente la interfaz [IDebugExpressionEvaluationCompleteEvent2](#) y sus métodos [IDebugExpressionEvaluationCompleteEvent2::GetExpression](#) y [GetResult](#).

Los eventos sincrónicos requieren la implementación de la [IDebugEngine2::ContinueFromSynchronousEvent](#) método.

Para que el motor escriba la salida de estilo de cadena, debe implementar el [IDebugOutputStringEvent2::GetString](#) método.

Vea también

- [Control de ejecución y evaluación del estado](#)

Envío de eventos

06/04/2020 • 2 minutes to read • [Edit Online](#)

El mecanismo de comunicación entre el depurador y el motor de depuración (DE) es un modelo de eventos basado en DCOM. Los eventos se envían como objetos COM y cada evento tiene parámetros que especifican:

- El DE que llamó al evento.
- Una descripción de lo que pasó.
- La información de proceso, programa y subproceso que identifica el contexto de donde se produjo el evento. El proceso no se envía para los eventos enviados desde un DE.
- El tipo de evento que indica si el evento es sincrónico o asincrónico.

Todos los eventos de depuración se envían mediante el método [IDebugEventCallback2::Event](#).

En esta sección

[Fuentes de eventos](#) Explica las dos fuentes de eventos: el motor de depuración (DE) y el administrador de depuración de sesión (SDM).

[Tipos de eventos admitidos](#) Describe los tipos de eventos admitidos actualmente: asincrónicos y sincrónicos.

[Descripciones de eventos](#) Define los eventos y las razones de su uso.

Secciones relacionadas

[Creación de un motor](#) de depuración personalizado Describe cómo funciona un DE con el intérprete o el sistema operativo para proporcionar servicios de depuración.

Orígenes de eventos (SDK de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Hay dos fuentes de eventos: el motor de depuración (DE) y el Administrador de depuración de sesión (SDM). Los eventos enviados desde un DE tienen un motor que no es NULL, mientras que los eventos enviados desde el SDM tienen un motor NULL.

Ejemplo

En el ejemplo siguiente se muestra cómo enviar el `IDebugProgramCreateEvent2` desde el DE al SDM.

```

CDebugProgramCreateEvent* pProgramCreateEvent = new CDebugProgramCreateEvent();
if (FAILED(pCallback->Event(m_pEngine, NULL, m_pProgram, NULL, pProgramCreateEvent,
IID_IDebugProgramCreateEvent2, EVENT_ASYNCHRONOUS)))
{
    // Handle failure here.
}
]

CEvent * pProgCreate = new CEvent(IID_IDebugProgramCreateEvent2, EVENT_ASYNCHRONOUS);
pProgCreate->SendEvent(pCallback, m_pEngine, (IDebugProgram2 *)this, NULL);

HRESULT CEvent::SendEvent(IDebugEventCallback2 *pCallback, IDebugEngine2 *pEngine, IDebugProgram2 *pProgram,
IDebugThread2 *pThread) {
    HRESULT hr;

    if (m_dwAttrib & EVENT_STOPPING)
    {
        hr = SendStoppingEvent(pCallback, pEngine, pProgram, pThread);
    }
    else if (m_dwAttrib & EVENT_SYNCHRONOUS)
    {
        hr = SendSynchronousEvent(pCallback, pEngine, pProgram, pThread);
    }
    else
    {
        assert(m_dwAttrib == 0);
        hr = SendAsynchronousEvent(pCallback, pEngine, pProgram, pThread);
    }

    return hr;
}

HRESULT CEvent::SendAsynchronousEvent(IDebugEventCallback2 *pCallback, IDebugEngine2 *pEngine, IDebugProgram2
*pProgram, IDebugThread2 *pThread) {

    HRESULT hr;

    // Make sure the CEvent object running this code is not deleted until the code completes.
    AddRef();

    pCallback->Event(pEngine, NULL, pProgram, pThread, (IDebugEvent2 *)this, m_riid, m_dwAttrib);

    // No error recovery here.
    hr = S_OK;

    Release();
    return hr;
}

```

Vea también

- [Envío de eventos](#)

Tipos de eventos admitidos

06/04/2020 • 5 minutes to read • [Edit Online](#)

La depuración de Visual Studio admite actualmente los siguientes tipos de eventos:

- Eventos asíncronos

Notifique al administrador de depuración de sesión (SDM) y al IDE que el estado de la aplicación que se está depurando está cambiando. Estos eventos se procesan en el tiempo libre del SDM y el IDE. No se envía ninguna respuesta al motor de depuración (DE) una vez que se procesa el evento. Las interfaces [IDebugOutputStringEvent2](#) e [IDebugMessageEvent2](#) son ejemplos de eventos asíncronos.

- Eventos sincrónicos

Notifique al SDM y al IDE que el estado de la aplicación que se está depurando está cambiando. La única diferencia entre estos eventos y eventos asíncronos es que se envía una respuesta mediante el método [ContinueFromSynchronousEvent](#).

Enviar un evento sincrónico es útil si necesita que su DE continúe el procesamiento después de que el IDE reciba y procese el evento.

- Eventos de detención sincrónicos o eventos de detención

Notifique al SDM y al IDE que la aplicación que se está depurando ha dejado de ejecutar código. Cuando se envía un evento de detención mediante el método [Event](#), se requiere el parámetro [IDebugThread2](#). Los eventos de detención se continúan mediante una llamada a uno de los métodos siguientes:

- [Ejecutar](#)
- [Step](#)
- [Continuar](#)

Las interfaces [IDebugBreakpointEvent2](#) y [IDebugExceptionEvent2](#) son ejemplos de detener eventos.

NOTE

No se admiten eventos de detención asíncronos. Es un error enviar un evento de detención asíncrono.

Discusión

La implementación real de eventos depende del diseño de su DE. El tipo de cada evento enviado viene determinado por sus atributos, que se establecen al diseñar la DE. Por ejemplo, un DE puede enviar un [IDebugProgramCreateEvent2](#) como un evento asíncrono, mientras que otro puede enviarlo como un evento de detención.

En la tabla siguiente se especifica qué parámetros de programa y subprocesso son necesarios para qué eventos, así como tipos de eventos. Cualquier evento puede ser sincrónico. Ningún evento debe ser sincrónico.

NOTE

El [IDebugEngine2](#) interfaz es necesaria para todos los eventos.

EVENTO	DEBUGPROGRAM2	DEBUGTHREAD2	DETENER EVENTOS
IDebugActivateDocumentEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugBreakEvent2	Obligatorio	Obligatorio	Sí
IDebugBreakpointBoundEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugBreakpointErrorEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugBreakpointUnboundEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugBreakpointEvent2	Obligatorio	Obligatorio	Sí
IDebugCanStopEvent2	Obligatorio	Obligatorio	No
IDebugDocumentTextEvents2	No permitida	No permitida	No
IDebugEngineCreateEvent2	No permitida	No permitida	No
IDebugEntryPointEvent2	Obligatorio	Obligatorio	Sí
IDebugErrorEvent2	Permitido, pero no requerido	Permitido, pero no requerido	Puede ser
IDebugExceptionEvent2	Obligatorio	Obligatorio	Sí
IDebugExpressionEvaluationCompleteEvent2	Permitido, pero no requerido	Permitido, pero no requerido	Puede ser
IDebugInterceptExceptionCompleteEvent2	Obligatorio	Obligatorio	Sí
IDebugLoadCompleteEvent2	Obligatorio	Obligatorio	Sí
IDebugMessageEvent2	Permitido, pero no requerido	Permitido, pero no requerido	Puede ser
IDebugModuleLoadEvent2	Obligatorio	Permitido, pero no requerido	No
IDebugOutputStringEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugProgramCreateEvent2	Obligatorio	Permitido, pero no requerido	No
IDebugProgramDestroyEvent2	Obligatorio	Permitido, pero no requerido	No
IDebugPropertyCreateEvent2	Obligatorio	Permitido, pero no requerido	No

EVENTO	IDeBUGPROGRAM2	IDeBUGTHREAD2	DETENER EVENTOS
IDebugPropertyDestroyEvent2	Obligatorio	Permitido, pero no requerido	No
IDebugReturnValueEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugStopCompleteEvent2	Obligatorio	Obligatorio	Sí
IDebugStepCompleteEvent2	Obligatorio	Obligatorio	Sí
IDebugSymbolSearchEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No
IDebugThreadCreateEvent2	Obligatorio	Obligatorio	No
IDebugThreadDestroyEvent2	Obligatorio	Obligatorio	No
IDebugThreadNameChangeEvent2	Permitido, pero no requerido	Permitido, pero no requerido	No

Vea también

- [Envío de eventos](#)

Descripciones de eventos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Cada tipo de evento tiene un propósito específico.

Eventos y las razones de su uso

EVENTO	DESCRIPCIÓN
Activar eventos de documentos	Se produce cuando el motor de depuración (DE) desea que el IDE abra o lleve un documento al primer plano.
Eventos de error de punto de interrupción o de punto de interrupción	Se envía cuando se enlaza un punto de interrupción o cuando un punto de interrupción no se puede enlazar y se devuelve un error.
Eventos sin enlazar punto de interrupción	Se produce cuando un punto de interrupción enlazado se desenlaza del código.
Puede detener eventos	Se envía al IDE para determinar si el usuario desea detenerse en un punto especificado en el código.
Eventos de punto de interrupción	Se produce cuando se alcanza un punto de interrupción de datos o código.
Eventos de texto del documento	Se produce cuando se cambia el texto de un documento. Estos eventos no se <code>IDebugEventCallback2::Event</code> envían a través del método.
El motor crea eventos	Se envía cuando se crea un motor por primera vez.
Eventos de punto de entrada	Se envía cuando el programa que se está depurando ha ejecutado su código de inicialización y ha alcanzado su primer punto de entrada de usuario.
Eventos de excepción	Se envía cuando un programa en ejecución alcanza una excepción.
Evaluación de expresiones eventos completos	Se envía cuando se completa la evaluación de expresiones asíncronas.
Buscar eventos de símbolo	Se envía siempre que el DE necesita pedir al usuario que encuentre símbolos para un módulo.
Cargar eventos completos	Se envía solo cuando se completa la carga inicial del programa y el primer código está a punto de ejecutarse en el programa.
Eventos de mensajes	Se envía cuando se envían mensajes a los usuarios.
Eventos de carga de módulos	Se envía cuando se carga o descarga un nuevo módulo.

EVENTO	DESCRIPCIÓN
Eventos de cadena de salida	Se envía cuando el programa escribe la salida de depuración.
Crear y destruir eventos	Se envía para anunciar la creación o destrucción de procesos, programas, propiedades, sesiones y subprocesos para que el IDE de Visual Studio pueda realizar un seguimiento del estado de los programas que se están depurando.
Eventos completos paso a paso	Se envía cuando se completa un paso.
Eventos de cambio de nombre de subproceso	Se envía cuando el usuario cambia el nombre de un subproceso.
Eventos de cambio de nombre del programa	Se envía cuando el usuario cambia el nombre de un programa.

Vea también

- [Envío de eventos](#)

Terminación y desmontaje

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la siguiente sección se describe la terminación normal.

Discusión

Después de que el [IDebugLoadCompleteEvent2](#) o [IDebugEntryPointEvent2](#) interfaz continúa, si no hay puntos de interrupción, excepciones, errores en tiempo de ejecución o bucles infinitos en la aplicación que se va a depurar, el programa que se está depurando se ejecuta hasta su finalización. Este proceso es una terminación normal.

Debe enviar un [IDebugProgramDestroyEvent2](#) para implementar la terminación normal. La terminación normal requiere ejecutar el [método IDebugProgramDestroyEvent2::GetExitCode](#).

Vea también

- [Creación de un motor de depuración personalizado](#)

Eventos del depurador de llamadas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los eventos en las sesiones de depuración se producen en un orden específico.

Discusión

Para comprender el patrón de llamadas entre el motor de depuración (DE) y el administrador de depuración de sesión (SDM), lo siguiente representa el orden de llamada de los eventos que se producen en una sesión de depuración típica:

1. [Adjuntar y separar a un programa](#)
2. [Lanzamiento del depurador](#)
3. [Terminar un programa](#)
4. [Creación de un punto de interrupción](#)
5. [Cuando un punto de interrupción se une o se desvincula](#)
6. [Errores de punto de interrupción](#)
7. [Llegada a un punto de interrupción](#)
8. [Eliminación de un punto de interrupción](#)
9. [Entrar en modo de interrupción](#)
10. [Paso en modo de interrupción](#)
11. [Evaluación de expresiones en modo de interrupción](#)
12. [Excepciones](#)

Vea también

- [Creación de un motor de depuración personalizado](#)

Adjuntar y separar a un programa

06/04/2020 • 2 minutes to read • [Edit Online](#)

La asociación del depurador requiere el envío de la secuencia correcta de métodos y eventos con los atributos adecuados.

Secuencia de métodos y eventos

1. El administrador de depuración de sesión (SDM) llama al método [OnAttach](#).

En función del modelo de proceso `IDebugProgramNodeAttach2::OnAttach` del motor de depuración (DE), el método devuelve uno de los métodos siguientes, que determina lo que sucede a continuación.

Si `S_FALSE` se devuelve, el motor de depuración se ha asociado correctamente al programa. De lo contrario, se llama al método [Attach](#) para completar el proceso de conexión.

Si `S_OK` se devuelve, el DE se cargará en el mismo proceso que el SDM. El SDM realiza las siguientes tareas:

- a. Llama a [GetEngineInfo](#) para obtener la información del motor de la DE.
- b. Co-crea el DE.
- c. Llamadas [Adjuntar](#).

2. El DE envía un [IDebugEngineCreateEvent2](#) al `EVENT_SYNC` SDM con un atributo.
3. El DE envía un [IDebugProgramCreateEvent2](#) al `EVENT_SYNC` SDM con un atributo.
4. El DE envía un [IDebugLoadCompleteEvent2](#) al `EVENT_SYNC_STOP` SDM con un atributo.

Separarse de un programa es un proceso simple de dos pasos, de la siguiente manera:

5. Las llamadas SDM [se separan](#).
6. La DE envía un [IDebugProgramDestroyEvent2](#).

Vea también

- [Llamar a eventos del depurador](#)

Inicie el depurador

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para iniciar el depurador es necesario enviar la secuencia correcta de métodos y eventos con sus atributos adecuados.

Secuencias de métodos y eventos

1. Se llama al administrador de depuración de sesión (SDM) eligiendo el menú **Depurar** y, a continuación, seleccionando **Iniciar**. Para obtener más información, consulte [Iniciar un programa](#).

2. El SDM llama al método [OnAttach](#).

3. En función del modelo de proceso `IDebugProgramNodeAttach2::OnAttach` del motor de depuración (DE), el método devuelve uno de los métodos siguientes, que determina lo que sucede a continuación.

Si `S_FALSE` se devuelve, el motor de depuración (DE) se cargará en el proceso de la máquina virtual.

o bien

Si `S_OK` devuelve, el DE se cargará en proceso del SDM. A continuación, el SDM realiza las siguientes tareas:

a. Llama a [GetEngineInfo](#) para obtener la información del motor de la DE.

b. Co-crea el DE.

c. Llamadas [Adjuntar](#).

4. El DE envía un [IDebugEngineCreateEvent2](#) al `EVENT_SYNC` SDM con un atributo.

5. El DE envía un [IDebugProgramCreateEvent2](#) al `EVENT_SYNC` SDM con un atributo.

6. El DE envía un [IDebugThreadCreateEvent2](#) al `EVENT_SYNC` SDM con un atributo.

7. El DE envía un [IDebugLoadCompleteEvent2](#) al `EVENT_SYNC` SDM con un atributo.

8. El DE envía un [IDebugEntryPointEvent2](#) al `EVENT_SYNC` SDM con un atributo.

Vea también

- [Llamar a eventos del depurador](#)
- [Lanzamiento de un programa](#)

Terminar un programa

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la siguiente sección se describe la terminación de un único programa con un subproceso.

Proceso de terminación

1. El DE envía un [IDebugThreadDestroyEvent2](#) con un [válido IDebugThread2](#).
2. El DE envía un [IDebugProgramDestroyEvent2](#) con un [iDebugProgram2](#)válido.

El IDE entra en modo de diseño. El motor de depuración o el entorno en tiempo de ejecución llama a [IDebugPortNotify2::RemoveProgramNode](#) para quitar el programa del puerto.

Vea también

- [Llamar a eventos del depurador](#)

Crear un punto de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

A continuación se describe el proceso de creación de un punto de interrupción.

Métodos en la creación de puntos de interrupción

Cuando se carga el módulo necesario para enlazar un punto de interrupción, el administrador de depuración de sesión (SDM) llama a los métodos siguientes:

1. [IDebugPendingBreakpoint2::Enable](#)
2. [IDebugPendingBreakpoint2::Virtualize](#)
3. [IDebugPendingBreakpoint2::CanBind](#)

NOTE

CanBind se llama sólo cuando un usuario hace un punto de interrupción desde el **punto** de interrupción ventana.

4. [IDebugPendingBreakpoint2::Bind](#)
5. [IDebugPendingBreakpoint2::EnumBoundBreakpoints](#)

Vea también

- [Eventos del depurador de llamadas](#)

Cuando un punto de interrupción se une o se vuelve sin enlazar

06/04/2020 • 2 minutes to read • [Edit Online](#)

Cuando un punto de interrupción no se puede enlazar en el momento en que se realiza una llamada a la [IDebugPendingBreakpoint2::CanBind](#) método, la hora de enlace y la hora de creación del punto de interrupción son diferentes.

Métodos llamados

El administrador de depuración de sesión (SDM) llama a los métodos siguientes:

1. [IDebugEngine2::CreatePendingBreakpoint](#). El DE devuelve un [IDebugPendingBreakpoint2](#).
2. [IDebugPendingBreakpoint2::Enable](#).
3. [IDebugPendingBreakpoint2::Virtualize](#).
4. El [IDebugPendingBreakpoint2::Bind](#) método y devuelve S_OK. El DE envía un [IDebugBreakpointBoundEvent2](#) o [IDebugBreakpointErrorEvent2](#).
5. [IDebugBreakpointBoundEvent2::GetPendingBreakpoint](#) y [IDebugBreakpointBoundEvent2::EnumBoundBreakpoints](#) métodos para comprobar y obtener los puntos de interrupción enlazados.

Vea también

- [Llamar a eventos del depurador](#)

Errores de punto de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

A continuación se describe el proceso cuando un punto de interrupción intenta enlazar al código, pero se produce un error.

Solucionar un error de punto de interrupción

1. El motor de depuración (DE) envía un [IDebugBreakpointErrorEvent2](#) al administrador de depuración de sesión (SDM).
2. El SDM llama a [IDebugBreakpointErrorEvent2::GetErrorBreakpoint](#) ([IDebugErrorBreakpoint2**](#) `ppErrorBP`) para obtener el punto de interrupción de error.
3. El SDM llama a [IDebugErrorBreakpoint2::GetPendingBreakpoint](#) para obtener el punto de interrupción pendiente desde el que se originó el punto de interrupción de error.
4. El SDM llama a [IDebugErrorBreakpoint2::GetBreakpointResolution](#) para obtener el motivo por el que el punto de interrupción de error no se pudo enlazar.

Vea también

- [Llamar a eventos del depurador](#)

Llegar a un punto de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

La siguiente sección describe el proceso cuando el motor de depuración (DE) alcanza un punto de interrupción mientras se ejecuta o se pasa por pasos:

Solucionar problemas de punto de interrupción de hit

1. La DE envía un [IDebugBreakpointEvent2](#) interfaz como un `EVENT_SYNC_STOP`.
2. El Administrador de depuración de sesión (SDM) llama a [IDebugBreakpointEvent2::EnumBreakpoints](#) para obtener el punto de interrupción que se alcanzó.

Vea también

- [Eventos del depurador de llamadas](#)

Eliminación de un punto de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

A continuación se describe el proceso al eliminar un punto de interrupción pendiente:

Proceso de eliminación

El Administrador de depuración de sesión (SDM) llama a la [IDebugPendingBreakpoint2::Delete](#) método para quitar el punto de interrupción pendiente y todos los puntos de interrupción enlazados de él.

NOTE

Un único punto de interrupción enlazado también se puede eliminar mediante una llamada a [IDebugBoundBreakpoint2::Delete](#).

Vea también

- [Eventos del depurador de llamadas](#)

Entrar en modo de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

La siguiente información describe el proceso que se produce cuando se encuentra un punto de interrupción después de entrar en una función, ejecutar en la línea de código fuente que tiene el cursor en él o ejecutar en un punto de interrupción.

Proceso de modo de interrupción

1. El motor de depuración (DE) envía [IDebugBreakpointEvent2](#), [IDebugExceptionEvent2](#) o cualquier otro evento de detención para hacer que el IDE entre en modo de interrupción.
2. El SDM consigue la información de la pila de llamadas del hilo, como sigue:
 - [IDebugThread2::EnumFrameInfo](#)
 - [IEnumDebugFrameInfo2::GetCount](#)
 - [IEnumDebugFrameInfo2::Next](#)
 - [IDebugStackFrame2::GetDocumentContext](#) para obtener la información del código fuente
 - [IDebugDocumentContext2::GetName](#) para obtener el nombre de archivo
 - [IDebugDocumentContext2::GetStatementRange](#) para obtener el intervalo de instrucciones
 - [IDebugStackFrame2::GetCodeContext](#) para obtener información de memoria

Vea también

- [Llamar a eventos del depurador](#)

Paso en modo de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la siguiente sección se describe el proceso que se produce cuando el depurador está en modo de interrupción y debe recorrer el código paso a paso:

Proceso de paso a paso

1. Llame a Los argumentos [IDebugProgram2::Step](#) con [STEPKIND](#) y [STEPUNIT](#) para ejecutar un paso.
2. Cuando finalice el paso, envíe un [IDebugStepCompleteEvent2](#) como un evento de detención.

Vea también

- [Llamar a eventos del depurador](#)

Evaluación de expresiones en modo de interrupción

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la siguiente sección se describe el proceso que se produce cuando el depurador está en modo de interrupción y debe llevar a cabo la evaluación de expresiones.

Proceso de evaluación de expresiones

A continuación se indican los pasos básicos implicados en la evaluación de una expresión:

1. El Administrador de depuración de sesión (SDM) llama a [IDebugStackFrame2::GetExpressionContext](#) para obtener una interfaz de contexto de expresión, [IDebugExpressionContext2](#).
2. A continuación, el SDM llama a [IDebugExpressionContext2::ParseText](#) con la cadena que se va a analizar.
3. Si [ParseText](#) no devuelve S_OK, se devuelve el motivo del error.

-de lo contrario-

Si [ParseText](#) devuelve S_OK, el SDM puede llamar a [IDebugExpression2::EvaluateSync](#) o [IDebugExpression2::EvaluateAsync](#) para obtener un valor final de la expresión analizada.

- Cuando `IDebugExpression2::EvaluateSync` se utiliza, la interfaz de devolución de llamada dada comunica el proceso en curso de la evaluación. El valor final se devuelve en un [IDebugProperty2](#) interfaz.
- Cuando `IDebugExpression2::EvaluateAsync` se utiliza, la interfaz de devolución de llamada dada comunica el proceso en curso de la evaluación. Una vez completada la evaluación, [EvaluateAsync](#) envía una interfaz [IDebugExpressionEvaluationCompleteEvent2](#) a través de la devolución de llamada. Con esta interfaz de eventos, el valor final se produce con [GetResult](#).

Vea también

- [Eventos del depurador de llamadas](#)

Control de excepciones (SDK de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

A continuación se describe el proceso que se produce cuando se producen excepciones.

Proceso de manejo de excepciones

1. Cuando se produce una excepción por primera vez, pero antes de que se controla mediante el controlador de excepciones en el programa que se está depurando, el motor de depuración (DE) envía un `IDebugExceptionEvent2` al administrador de depuración de sesión (SDM) como un evento de detención. Se `IDebugExceptionEvent2` envía si solo la configuración de la excepción (especificada en el cuadro de diálogo Excepciones del paquete de depuración) especifica que el usuario desea detenerse en las notificaciones de excepción de primera oportunidad.
2. El SDM llama a `IDebugExceptionEvent2::GetException` para obtener la propiedad de excepción.
3. El paquete de depuración llama a `IDebugExceptionEvent2::CanPassToDebuggee` para determinar qué opciones presentar al usuario.
4. El paquete de depuración pregunta al usuario cómo controlar la excepción abriendo un cuadro de diálogo de excepción de primera oportunidad.
5. Si el usuario decide continuar, el SDM llama a `IDebugExceptionEvent2::CanPassToDebuggee`.
 - Si el método devuelve `S_OK`, llama a `IDebugExceptionEvent2::PassToDebuggee`.o bien
Si el método devuelve `S_FALSE`, el programa que se está depurando tiene una segunda oportunidad para controlar la excepción.
6. Si el programa que se está depurando no tiene ningún `IDebugExceptionEvent2` controlador para una excepción de segunda oportunidad, la DE envía un al SDM como `EVENT_SYNC_STOP`.
7. El paquete de depuración pregunta al usuario cómo controlar la excepción abriendo un cuadro de diálogo de excepción de primera oportunidad.
8. El paquete de depuración llama a `IDebugExceptionEvent2::CanPassToDebuggee` para determinar qué opciones presentar al usuario.
9. El paquete de depuración pregunta al usuario cómo controlar la excepción abriendo un cuadro de diálogo de excepción de segunda oportunidad.
10. Si el método devuelve `IDebugExceptionEvent2::PassToDebuggee` `S_OK`, llama a .

Vea también

- [Eventos del depurador de llamadas](#)

Cómo: Depurar un motor de depuración personalizado

06/04/2020 • 6 minutes to read • [Edit Online](#)

Un tipo de proyecto inicia el motor [DebugLaunch](#) de depuración (DE) desde el método. Esto significa que el DE se lanza bajo Visual Studio el control de la instancia de control del tipo de proyecto. Sin embargo, Visual Studio esa instancia de no puede depurar el DE. Lo que sigue son los pasos que le permiten depurar su DE personalizado.

NOTE

: en el procedimiento "Depurar un motor de depuración personalizado", debe esperar a que se inicie la DE antes de poder adjuntarla. Si coloca un cuadro de mensaje cerca del principio de su DE que aparece cuando se inicia el DE, puede adjuntar en ese punto y, a continuación, desactive el cuadro de mensaje para continuar. De esa manera, puede ver todos los eventos DE.

WARNING

Debe tener instalada la depuración remota antes de intentar los siguientes procedimientos. Consulte [Depuración remota](#) para obtener más información.

Depurar un motor de depuración personalizado

1. Inicie *msvsmon.exe*, el Monitor de depuración remota.
2. En el menú **Herramientas** de *msvsmon.exe*, seleccione **Opciones** para abrir el cuadro de diálogo **Opciones**.
3. Seleccione la opción "sin autenticación" y haga clic en **Aceptar**.
4. Inicie una Visual Studio instancia y abra el proyecto DE personalizado.
5. Inicie una segunda Visual Studio instancia del proyecto personalizado que inicia el DE (para el desarrollo, esto suele estar en el subárbol del registro experimental que se configura cuando se instala VSIP).
6. En esta segunda Visual Studio instancia de , cargue un archivo de origen desde el proyecto personalizado e inicie el programa que se va a depurar. Espere unos momentos para permitir que el DE se cargue o espere hasta que se golpee un punto de interrupción.
7. En la primera Visual Studio instancia de (con el proyecto DE), seleccione **Adjuntar al proceso** en el menú **Depurar**.
8. En el cuadro de diálogo **Asociar al proceso**, cambie **Transporte a remoto (nativo solo sin autenticación)**.
9. Cambie el **Calificador** por el nombre de su máquina (tenga en cuenta que hay un historial de entradas, por lo que debe escribir este nombre solo una vez).
10. En la lista **Procesos disponibles**, seleccione la instancia de su DE que se está ejecutando y haga clic en el botón **Adjuntar**.
11. Después de que los símbolos se hayan cargado en su DE, coloque puntos de interrupción en el código DE.
12. Cada vez que detenga y reinicie el proceso de depuración, repita los pasos del 6 al 10.

Depurar un tipo de proyecto personalizado

1. Comience Visual Studio en el subárbol del Registro normal y cargue el proyecto de tipo de proyecto (esto es el origen del tipo de proyecto, no una creación de instancias del tipo de proyecto).
2. Abra las propiedades del proyecto y vaya a la página **Depurar**. Para el **comando**, escriba Visual Studio la ruta de acceso al IDE (de Visual Studio forma predeterminada, se trata de *[unidad]*).
3. En **Command Arguments** `/rootsuffix exp`, escriba para el subárbol del Registro experimental (creado cuando se instaló VSIP).
4. Haga clic en **Aceptar** para aceptar los cambios.
5. Inicie el tipo de proyecto presionando **F5**. Esto lanza una segunda Visual Studio instancia de .
6. En este punto, puede colocar puntos de interrupción en el código fuente del tipo de proyecto.
7. En la segunda Visual Studio instancia de , cargue o cree una nueva instancia del tipo de proyecto. Durante la carga o la creación, los puntos de interrupción pueden ser alcanzados.
8. Depurar el tipo de proyecto.
9. Si decide depurar el proceso de iniciar un DE, puede realizar los pasos del procedimiento "Depurar un motor de depuración personalizado" para adjuntar lo desenmarcha después de iniciarlo. Esto proporciona tres instancias de Visual Studio ejecución: una para el origen de tipo de proyecto, una segunda para el tipo de proyecto con instancias y una tercera asociada a la DE.

Vea también

- [Creación de un motor de depuración personalizado](#)

Comience con la extensibilidad del depurador

06/04/2020 • 2 minutes to read • [Edit Online](#)

Proporciona Visual Studio SDK la información que necesita para crear y personalizar los Visual Studio componentes del depurador que se usan para depurar programas desde el entorno.

Visual Studio depuración ha agregado mejoras derivadas de las extensas pruebas de usabilidad realizadas en depuradores anteriores. Visual Studio puede usar Visual Studio la depuración para recorrer paso a paso una aplicación en varios idiomas, o puede implementar la edición sobre la marcha de variables durante la depuración de aplicaciones y soluciones multilenguaje.

Visual Studio la depuración se ejecuta fuera de proceso con el programa que se está depurando y, por lo tanto, es menos intrusiva en el espacio de proceso de la aplicación. Por lo tanto, es más fácil escribir componentes que interactúan con el depurador sin afectar al programa de depuración.

Para utilizar Visual Studio SDK mejor el , debe estar familiarizado con los siguientes elementos:

- El Visual Studio entorno de desarrollo integrado (IDE)
- El lenguaje de programación C++
- ATL COM

En esta sección

[Hoja de ruta para ampliar el depurador](#) Describe el proceso de implementación de la depuración en el producto, según el compilador y su salida.

[Componentes del depurador](#) Proporciona información general Visual Studio sobre los componentes de depuración, que incluyen el motor de depuración (DE), el evaluador de expresiones (EE) y el controlador de símbolos (SH).

[Conceptos del depurador](#) Describe los principales conceptos arquitectónicos de depuración.

[Contextos del depurador](#) Explica cómo funciona el motor de depuración (DE) simultáneamente dentro de los contextos de evaluación de código, documentación y expresión. Describe, para cada uno de los tres contextos, la ubicación, la posición o la evaluación relevantes para ella.

[Tareas de depuración](#) Contiene vínculos a varias tareas de depuración, como iniciar un programa y evaluar expresiones.

Hoja de ruta para ampliar el depurador

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta documentación proporciona información de Visual Studio guía y Visual Studio SDK referencia para ampliar el depurador con el archivo .

Visual Studio la documentación de depuración incluye ejemplos, una referencia completa y varios escenarios representativos que muestran las formas típicas de personalizar el depurador.

El compilador y su salida determinan lo que se necesita para configurar la depuración en el producto. Si el compilador:

- Tiene como destino el sistema operativo nativo de Windows y escribe un *archivo .PDB*, puede depurar programas con el motor de depuración Visual Studio de código nativo (DE), que está integrado en . No es necesario implementar un evaluador de expresiones o DE. El evaluador de expresiones se escribe para la sintaxis del lenguaje de programación C++.
- Produce la salida del lenguaje intermedio de Microsoft (MSIL), puede depurar programas con Visual Studio el motor de depuración de código administrado DE, que también está integrado en . Por lo tanto, solo necesita implementar un evaluador de expresiones. Se proporciona un evaluador de expresiones de ejemplo. Para obtener más información, vea los temas siguientes:

[Evaluación de expresiones](#)

[Evaluación de expresiones](#)

[Contexto de evaluación de expresiones](#)

[Evaluación de expresiones en modo de interrupción](#)

[Escribir un evaluador de expresiones de Common Language Runtime](#)

- Se dirige a un sistema operativo propietario o a algún otro entorno en tiempo de ejecución, debe escribir su propio DE. Se proporciona un tutorial que crea un DE simple mediante ATL COM. Para obtener más información, vea los temas siguientes:

[Crear un motor de depuración personalizado](#)

[Tutorial: Cree un motor de depuración con ATL COM](#)

[Implementar un proveedor portuario](#)

[Muestras](#)

Vea también

- [Introducción](#)

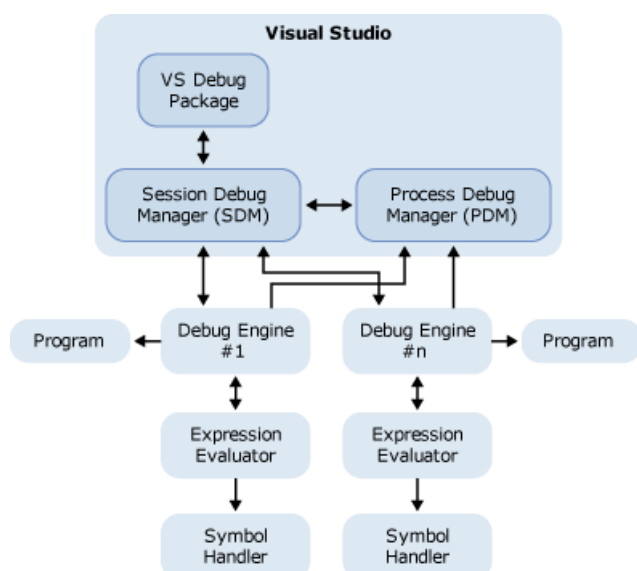
Componentes del depurador

06/04/2020 • 5 minutes to read • [Edit Online](#)

El Visual Studio depurador se implementa como un VSPackage y administra toda la sesión de depuración. La sesión de depuración consta de los siguientes elementos:

- **Paquete de depuración:** El Visual Studio depurador proporciona la misma interfaz de usuario independientemente de lo que se está depurando.
- **Administrador de depuración de sesión (SDM):** Proporciona una interfaz de Visual Studio programación coherente al depurador para la administración de una variedad de motores de depuración. Se implementa por Visual Studio.
- **Administrador de depuración de procesos (PDM):** Administra, para todas las Visual Studio instancias en ejecución de , una lista de todos los programas que pueden ser o se están depurando. Se implementa por Visual Studio.
- **Motor de depuración (DE):** Es responsable de supervisar un programa que se está depurando, comunicar el estado del programa en ejecución al SDM y PDM e interactuar con el evaluador de expresiones y el proveedor de símbolos para proporcionar un análisis en tiempo real del estado de la memoria y las variables de un programa. Se implementa por Visual Studio (para los idiomas que admite) y proveedores de terceros que desean admitir su propio tiempo de ejecución.
- **Evaluador de expresiones (EE):** Proporciona compatibilidad con la evaluación dinámica de variables y expresiones proporcionadas por el usuario cuando un programa se ha detenido en un punto determinado. Se implementa por Visual Studio (para los idiomas que admite) y proveedores de terceros que quieren admitir sus propios idiomas.
- **Proveedor de símbolos (SP):** También denominado controlador de símbolos, asigna los símbolos de depuración de un programa a una instancia en ejecución del programa para que se pueda proporcionar información significativa (como la depuración de nivel de código fuente y la evaluación de expresiones). Se implementa mediante Visual Studio (para los símbolos de Common Language Runtime [CLR] y el formato de archivo de símbolos de Base de datos de programa [PDB]) y por proveedores externos que tienen su propio método propietario de almacenamiento de información de depuración.

En el diagrama siguiente se muestra la relación entre estos elementos del depurador de Visual Studio.



En esta sección

[Paquete de depuración](#) Describe el paquete de depuración, que se ejecuta en el Visual Studio shell y controla toda la interfaz de usuario.

[Administrador de depuración de procesos](#) Proporciona una visión general de las características de PDM, que es el administrador de los procesos que se pueden depurar.

[Administrador de depuración de sesión](#) Define el SDM, que proporciona una vista unificada de la sesión de depuración al IDE. El SDM administra el DE.

[Motor de depuración](#) Documenta los servicios de depuración que proporciona la DE.

[Modos operativos](#) Proporciona una visión general de los tres modos en los que el IDE puede operar: modo de diseño, modo de ejecución y modo de interrupción. También se discuten los mecanismos de transición.

[Evaluador de expresiones](#) Explica el propósito de EE en tiempo de ejecución.

[Proveedor de símbolos](#) Describe cómo, en la implementación, el proveedor de símbolos evalúa variables y expresiones.

[Visualizador de tipos y visor personalizado](#) Describe qué son un visualizador de tipos y un visor personalizado y qué función desempeña el evaluador de expresiones al admitir ambos.

Secciones relacionadas

[Conceptos del depurador](#) Describe los principales conceptos arquitectónicos de depuración.

[Contextos del depurador](#) Explica cómo funciona la DE simultáneamente dentro de los contextos de evaluación de código, documentación y expresión. Describe, para cada uno de los tres contextos, la ubicación, la posición o la evaluación relevantes para ella.

[Tareas de depuración](#) Contiene vínculos a varias tareas de depuración, como iniciar un programa y evaluar expresiones.

Vea también

- [Introducción](#)

Paquete de depuración

06/04/2020 • 2 minutes to read • [Edit Online](#)

El paquete de depuración se ejecuta en el shell de Visual Studio y controla toda la interfaz de usuario. Consume las interfaces de depuración de Visual Studio y se comunica con el administrador de depuración de sesión (SDM).

Los eventos de interrupción enviados a través del SDM cambian el depurador del modo de ejecución al modo de interrupción y cambian el foco al programa donde se produjo la interrupción. El paquete de depuración realiza un seguimiento del marco de pila y el subproceso de la información que le envían los eventos.

El paquete de depuración no tiene dependencias de entorno de idioma o en tiempo de ejecución. No es necesario implementar o modificar el paquete de depuración.

El paquete de depuración se implementa mediante *vsdebug.dll*.

Vea también

- [Administrador de depuración de sesión](#)
- [Marcos de pila](#)
- [Hilos](#)
- [Componentes del depurador](#)

Administrador de depuración de procesos

06/04/2020 • 2 minutes to read • [Edit Online](#)

El administrador de depuración de procesos (PDM) es un componente de Visual Studio que administra programas y procesos, haciéndolos disponibles para el administrador de depuración de sesión y los motores de depuración.

PDM administra todos los procesos que se pueden depurar. Para ser depurado, un programa debe estar registrado en PDM. Este registro se realiza en el momento en que se inicia el programa, ya sea por un puerto o un motor de depuración.

Vea también

- [Procesos](#)
- [Motor de depuración](#)
- [Puertos](#)
- [Programas](#)
- [Componentes del depurador](#)

Administrador de depuración de sesión

06/04/2020 • 3 minutes to read • [Edit Online](#)

El administrador de depuración de sesión (SDM) administra cualquier número de motores de depuración (DE) que están depurando cualquier número de programas en varios procesos en cualquier número de máquinas. Además de ser un multiplexor del motor de depuración, el SDM proporciona una vista unificada de la sesión de depuración al IDE.

Operación del administrador de depuración de sesión

El administrador de depuración de sesión (SDM) administra el DE. Puede haber más de un motor de depuración ejecutándose en una máquina al mismo tiempo. Para multiplexar los DE, el SDM ajusta una serie de interfaces de los DE y los expone al IDE como una sola interfaz.

Para aumentar el rendimiento, algunas interfaces no se multiplexan. En lugar, se utilizan directamente del DE, y las llamadas a estas interfaces no pasan a través del SDM. Por ejemplo, las interfaces utilizadas con contextos de memoria, código y documento no se multiplexan, porque hacen referencia a una instrucción, memoria o documento específico según un programa específico depurado por un DE específico. Ningún otro DE necesita estar involucrado en ese nivel de comunicación.

Esto no es cierto en todos los contextos. Las llamadas a la interfaz de contexto de evaluación de expresiones pasan por el SDM. Durante la evaluación de expresiones, el SDM ajusta el [IDebugExpression2](#) interfaz que proporciona al IDE porque cuando se evalúa esa expresión, puede implicar varios DE que están depurando programas en el mismo proceso que podrían ejecutarse en el mismo subproceso.

El SDM actúa típicamente como un mecanismo de delegación, pero podría actuar como un mecanismo de difusión. Por ejemplo, durante la evaluación de expresiones, el SDM actúa como un mecanismo de difusión para notificar a todos los DE que pueden ejecutar código en un subproceso especificado. De forma similar, cuando el SDM recibe un evento de detención, transmite a los programas que deben dejar de ejecutarse. Cuando se llama a un paso, el SDM difunde a los programas que pueden continuar ejecutándose. Los puntos de interrupción también se transmiten a cada DE.

El SDM no realiza un seguimiento del programa, el subproceso o el marco de pila actuales. La información de proceso, programa y subproceso se envía al SDM junto con eventos de depuración específicos.

Vea también

- [Motor de depuración](#)
- [Componentes del depurador](#)
- [Contextos del depurador](#)

Motor de depuración

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un motor de depuración (DE) funciona con el intérprete o el sistema operativo para proporcionar servicios de depuración como control de ejecución, puntos de interrupción y evaluación de expresiones. El DE es responsable de supervisar el estado de un programa que se está depurando. Para lograr esto, la DE utiliza los métodos que están disponibles para él en el tiempo de ejecución admitido, ya sea desde la CPU o desde las API proporcionadas por el tiempo de ejecución.

Por ejemplo, Common Language Runtime (CLR) proporciona mecanismos para supervisar un programa en ejecución a través de las interfaces ICorDebugXXX. Un DE que admite CLR usa las interfaces ICorDebugXXX adecuadas para realizar un seguimiento de un programa de código administrado que se está depurando. A continuación, comunica cualquier cambio de estado al administrador de depuración Visual Studio de sesión (SDM), que reenvía dicha información al IDE.

NOTE

Un motor de depuración tiene como destino un tiempo de ejecución específico, es decir, el sistema en el que se ejecuta el programa que se está depurando. CLR es el tiempo de ejecución del código administrado y el tiempo de ejecución de Win32 es para aplicaciones nativas de Windows. Si el lenguaje que crea puede tener Visual Studio como destino uno de estos dos tiempos de ejecución, ya proporciona los motores de depuración necesarios. Todo lo que tiene que implementar es un evaluador de expresiones.

Funcionamiento del motor de depuración

Los servicios de supervisión se implementan a través de las interfaces DE y pueden hacer que el paquete de depuración pase entre diferentes modos operativos. Para obtener más información, consulte [Modos operativos](#). Normalmente, solo hay una implementación DE por entorno en tiempo de ejecución.

NOTE

Aunque hay implementaciones DE independientes JScript para Transact-SQL y VBScript, JScript comparte una sola DE.

Visual Studio la depuración permite que los motores de depuración se ejecuten de dos maneras: en el mismo proceso que el shell o en el mismo proceso que el programa de destino que se está depurando. Este último formulario suele producirse cuando el proceso que se está depurando es en realidad un script que se ejecuta bajo un intérprete. El motor de depuración debe tener un conocimiento íntimo del intérprete para supervisar el script. En este caso, el intérprete es en realidad un tiempo de ejecución; los motores de depuración son para implementaciones en tiempo de ejecución específicas. Además, la implementación de una única DE se puede dividir entre los límites del proceso y de la máquina (por ejemplo, la depuración remota).

El DE expone Visual Studio las interfaces de depuración. Toda la comunicación es a través de COM. Si el DE se carga en proceso, fuera de proceso o en otro equipo, no afecta a la comunicación de componentes.

El DE funciona con un componente de evaluador de expresiones para permitir que la DE de ese tiempo de ejecución determinado comprenda la sintaxis de las expresiones. El DE también funciona con un componente de controlador de símbolos para tener acceso a la información de depuración simbólica generada por el compilador de lenguaje. Para obtener más información, consulte Evaluador de [expresiones](#) y Proveedor de [símbolos](#).

Vea también

- [Componentes del depurador](#)
- [Evaluador de expresiones](#)
- [Proveedor de símbolos](#)

Modos operativos

06/04/2020 • 5 minutes to read • [Edit Online](#)

Hay tres modos en los que el IDE puede funcionar, de la siguiente manera:

- [Modo de diseño](#)
- [Modo de ejecución](#)
- [Modo de interrupción](#)

La forma en que las transiciones del motor de depuración personalizado (DE) entre estos modos es una decisión de implementación que requiere que esté familiarizado con los mecanismos de transición. El DE puede o no implementar directamente estos modos. Estos modos son realmente modos de paquete de depuración que cambian en función de la acción del usuario o eventos de la DE. Por ejemplo, la transición del modo de ejecución al modo de interrupción es instigada por un evento de detención de la DE. La transición de interrupción al modo de ejecución o modo de paso es instigada por el usuario que realiza operaciones como Paso o Ejecutar. Para obtener más información acerca de las transiciones DE, vea [Control de ejecución](#).

Modo de diseño

El modo de diseño es el estado de no ejecución de la depuración de Visual Studio, durante el cual puede establecer características de depuración en la aplicación.

Solo se utilizan algunas características de depuración durante el modo de diseño. Un desarrollador puede elegir establecer puntos de interrupción o crear expresiones de inspección. El DE nunca se carga o se llama mientras el IDE está en modo de diseño. La interacción con el DE tiene lugar solo durante los modos de ejecución y interrupción.

Modo de ejecución

El modo de ejecución se produce cuando un programa se ejecuta en una sesión de depuración en el IDE. La aplicación se ejecuta hasta la terminación, hasta que se alcanza un punto de interrupción o hasta que se produce una excepción. Cuando la aplicación se ejecuta hasta la terminación, la DE pasa al modo de diseño. Cuando se alcanza un punto de interrupción o se produce una excepción, la DE pasa al modo de interrupción.

Modo de interrupción

El modo de interrupción se produce cuando se suspende la ejecución del programa de depuración. El modo de interrupción ofrece al desarrollador una instantánea de la aplicación en el momento de la interrupción y permite al desarrollador analizar el estado de la aplicación y cambiar la forma en que se ejecutará la aplicación. El desarrollador puede ver y editar código, examinar o modificar datos, reiniciar la aplicación, finalizar la ejecución o continuar la ejecución desde el mismo punto.

El modo de interrupción se introduce cuando la DE envía un evento de detención sincrónico. Eventos de detención sincrónicos, también denominados detener eventos, notificar al administrador de depuración de sesión (SDM) y el IDE que la aplicación que se está depurando ha dejado de ejecutar código. Las interfaces [IDebugBreakpointEvent2](#) e [IDebugExceptionEvent2](#) son ejemplos de eventos de detención.

Los eventos de detención se continúan mediante una llamada a uno de los métodos siguientes, que realizan la transición del depurador del modo de interrupción al modo de ejecución o paso:

- [Ejecutar](#)
- [Step](#)
- [Continuar](#)

Modo paso

El modo de paso se produce cuando el programa pasa a la siguiente línea de código, o dentro, encima o fuera de una función. Se ejecuta un paso llamando al método [Step](#). Este método `DWORD` requiere s que especifican las enumeraciones [STEPUNIT](#) y [STEPKIND](#) como parámetros de entrada.

Cuando el programa pasa correctamente a la siguiente línea de código o a una función, o se ejecuta en el cursor o a un punto de interrupción establecido, la DE vuelve automáticamente al modo de interrupción.

Vea también

- [Control de ejecución](#)

Evaluador de expresiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los evaluadores de expresiones (EE) examinan la sintaxis de un lenguaje para analizar y evaluar variables y expresiones en tiempo de ejecución, lo que permite que el usuario las vea cuando el IDE está en modo de interrupción.

Usar evaluadores de expresiones

Las expresiones se crean mediante el método [ParseText](#), como se indica a continuación:

1. El motor de depuración (DE) implementa el [IDebugExpressionContext2](#) interfaz.
2. El paquete de `IDebugExpressionContext2` depuración obtiene un objeto de un `IDebugStackFrame2::ParseText` [IDebugStackFrame2](#) interfaz y, a continuación, llama al método en él para obtener un [IDebugExpression2](#) objeto.
3. El paquete de depuración llama al método [EvaluateSync](#) o al método [EvaluateAsync](#) para obtener el valor de la expresión. `IDebugExpression2::EvaluateAsync` se llama desde la ventana Comando/Inmediato. Todos los demás componentes de la interfaz de usuario llaman a `IDebugExpression2::EvaluateSync`.
4. El resultado de la evaluación de expresiones es un objeto [IDebugProperty2](#), que contiene el nombre, el tipo y el valor del resultado de la evaluación de expresiones.

Durante la evaluación de expresiones, EE requiere información del componente del proveedor de símbolos. El proveedor de símbolos proporciona la información simbólica utilizada para identificar y comprender la expresión analizada.

Cuando se completa la evaluación de expresiones asincrónicas, la DE envía un evento asincrónico a través del administrador de depuración de sesión (SDM) para notificar al IDE que la evaluación de expresiones se ha completado. Además, el resultado de la evaluación `IDebugExpression2::EvaluateSync` se devuelve de la llamada al método.

Notas de implementación

Los Visual Studio motores de depuración esperan hablar con el evaluador de expresiones mediante interfaces de Common Language Runtime (CLR). Como resultado, un evaluador de Visual Studio expresiones que funciona con los motores de depuración debe admitir CLR (se puede encontrar una Kit de desarrollo de software de Windows (SDK) lista completa de todas las interfaces de depuración de CLR en [debugref.doc](#), que forma parte de la extensión).

Vea también

- [Componentes del depurador](#)

Proveedor de símbolos

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una implementación del evaluador de expresiones debe tener acceso a la información de depuración simbólica generada por el compilador de lenguaje para evaluar variables y expresiones. Lo hace consumiendo las interfaces de un proveedor de símbolos (SP), también denominado controlador de símbolos.

Visual Studio proporciona SPs para código administrado, así como código nativo mediante el formato de archivo de símbolos de Base de datos de programa (PDB). A menos que haya una gran necesidad de que el programa utilice símbolos almacenados Visual Studio en un formato personalizado, se recomienda que utilice los SP proporcionados por .

Notas de implementación

Los Visual Studio motores de depuración esperan hablar con los SP mediante interfaces de Common Language Runtime (CLR). Como resultado, un SP que va a trabajar con los motores de depuración de Visual Studio debe admitir CLR. Puede encontrar una lista completa de todas las interfaces de depuración Kit de desarrollo de software de Windows (SDK) de CLR en [debugref.doc](#), que forma parte del archivo .

Si el SP solo funcionará con el motor de depuración personalizado, puede implementar el SP como considere adecuado en función de las necesidades del motor de depuración.

Vea también

- [Componentes del depurador](#)

Visualizador de tipos y visor personalizado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un visualizador de tipos es un componente que muestra un fragmento de datos en un formato específico. El formato depende enteramente de quién implementa el visualizador, ya sea el usuario final o un proveedor externo de visualizadores.

Un visor personalizado es la parte de un evaluador de expresiones personalizado que muestra un fragmento de datos en un formato específico. Este formato depende totalmente del implementador del visor personalizado, lo que significa que el formato depende del implementador del evaluador de expresiones (EE).

Compatibilidad con visualizadores de tipos en un evaluador de expresiones

Un EE admite visualizadores de tipos mediante la compatibilidad con un conjunto de interfaces accesibles para los visualizadores: interfaces como [IEEVisualizerService](#) e [IEEVisualizerDataProvider](#). Sin embargo, eE no es responsable de implementar el visualizador de tipo en sí: el EE simplemente permite a los visualizadores externos el acceso a su información de tipo. Estos visualizadores pueden enviarse junto con el EE e instalar en el lugar adecuado en Visual Studio, proporcionado por otro proveedor de terceros o incluso por el usuario final.

Compatibilidad con visores personalizados en un evaluador de expresiones

Un EE también puede admitir visores personalizados en los que el propio EE proporciona el código para ver el tipo de datos. Un visor personalizado implementa el [IDebugCustomViewer](#) interfaz, que controla todas las tareas de mostrar los datos en el formato que se desee; el visor tiene control total sobre la pantalla e incluso puede permitir que los datos se modifiquen. Los visores personalizados suministrados por EE vienen con el EE cuando se envía el producto.

Vea también

- [Componentes del depurador](#)
- [Evaluador de expresiones](#)
- [Motor de depuración](#)
- [IDebugCustomViewer](#)
- [IEEVisualizerService](#)
- [IEEVisualizerDataProvider](#)

Conceptos del depurador

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para compilar en el paquete de depuración de Visual Studio, debe estar familiarizado con los conceptos arquitectónicos utilizados en el diseño del paquete.

En esta sección

Sesión de depuración Explica el rol de una sesión en la arquitectura de depuración.

Servidores Define lo que es un servidor en términos de arquitectura de depuración, tanto en términos abstractos como físicos.

Proveedores portuarios Define lo que es un proveedor de puertos en términos de arquitectura de depuración.

Puertos Define lo que es un puerto en términos de arquitectura de depuración.

Procesos Define lo que es un proceso en términos de arquitectura de depuración.

Nodos de programa Define un nodo de programa en términos de arquitectura de depuración, incluida la forma en que puede identificarse a sí mismo y el proceso en el que se ejecuta.

Programas Define un programa en términos de arquitectura de depuración.

Roscas Define las características de los subprocesos en términos de arquitectura de depuración.

Marcos de pila Define un marco de pila en términos de arquitectura de depuración. Un marco de pila es una abstracción de una pila que proporciona el contexto de ejecución de un subproceso.

Módulos Define un módulo, en términos de arquitectura de depuración, como un contenedor físico de código, como un archivo ejecutable o un archivo DLL.

Puntos de interrupción Define los tres tipos de puntos de interrupción (pendientes, enlazados y errores) en términos de arquitectura de depuración.

Secciones relacionadas

Contextos del depurador Explica cómo funciona el motor de depuración (DE) simultáneamente dentro de los contextos de evaluación de código, documentación y expresión. Describe, para cada uno de los tres contextos, la ubicación, la posición o la evaluación relevantes para ella.

Componentes del depurador Proporciona información general de los componentes de depuración de Visual Studio, que incluyen el motor de depuración (DE), el evaluador de expresiones (EE) y el controlador de símbolos (SH).

Tareas de depuración Contiene vínculos a varias tareas de depuración, como iniciar un programa y evaluar expresiones.

Sesión de depuración

06/04/2020 • 2 minutes to read • [Edit Online](#)

En términos de la arquitectura del depurador, una *sesión*:

- Es el conjunto de todos los procesos, relacionados o no relacionados, que se han lanzado o adjuntado desde una única instancia del IDE.
- Se implementa mediante un administrador de depuración de sesión (SDM). Para obtener más información, vea [Administrador de depuración](#) de sesiones .

Vea también

- [Administrador de depuración de sesión](#)
- [Conceptos del depurador](#)

Servidores (Visual Studio SDK)

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un *servidor*:

- Es un contenedor de puertos y proveedores de puertos y comunica puertos y proveedores de puertos al administrador de depuración de sesión (SDM) y a los motores de depuración.
- Puede identificarse por su nombre y enumerar sus puertos y proveedores de puertos.
- Se representa mediante una interfaz [IDebugCoreServer2](#), que solo implementa Visual Studio (una instancia de un servidor para cada instancia de Visual Studio en ejecución).

Vea también

- [Puertos](#)
- [Proveedores portuarios](#)
- [Conceptos del depurador](#)
- [IDebugCoreServer2](#)

Proveedores portuarios

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un proveedor de *puertos*:

- Está contenido por un servidor y proporciona puertos a petición a ese servidor.
- Puede agregar y quitar puertos del servidor contenedor.
- Puede enumerar todos los puertos que ha proporcionado al servidor.
- Se representa mediante un [IDebugPortSupplier2](#) interfaz, que se registra con Visual Studio a través del registro. Esta interfaz se puede obtener llamando a [GetPortSupplier](#).

Visual Studio proporciona un proveedor de puertos predeterminado y un puerto predeterminado. Si es necesario implementar un puerto personalizado, también es necesario implementar un proveedor de puertos personalizado para proporcionar esos puertos personalizados.

Vea también

- [Servidores](#)
- [Puertos](#)
- [Conceptos del depurador](#)
- [IDebugPortSupplier2](#)
- [GetPortSupplier](#)

Puertos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un *puerto*:

- Es un contenedor para un conjunto de procesos que se ejecutan en un servidor. Por ejemplo, un puerto puede representar una conexión a un dispositivo basado en Windows CE mediante un cable serie o a una máquina en red no DCOM. Un puerto especial, denominado puerto local, contiene todos los procesos que se ejecutan en el equipo local.
- Puede identificarse por nombre o identificador.
- Puede enumerar todos los procesos que se ejecutan en el puerto e iniciar y finalizar estos procesos.
- Se representa mediante un [IDebugPort2](#) interfaz, que se crea pasando un [IDebugPortRequest2](#) argumento a [AddPort](#).

Visual Studio proporciona un puerto predeterminado que controla todos los procesos basados en Windows, tanto nativos como administrados. Se debe configurar un puerto personalizado para las conexiones con dispositivos externos que no están basados en Windows. Para proporcionar estos puertos personalizados, también debe configurar un proveedor de puertos personalizado.

Vea también

- [Servidores](#)
- [Procesos](#)
- [Conceptos del depurador](#)
- [IDebugPort2](#)
- [IDebugPortRequest2](#)
- [AddPort](#)

Procesos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un *proceso*:

- Es un contenedor para un conjunto de programas. Es muy análogo a un proceso de Windows, que es un contenedor para un conjunto de subprocesos.
- Puede identificarse por nombre, identificador o identificador físico.
- Puede enumerar todos los programas en ejecución (y sus subprocesos).
- Puede describirse a sí mismo, el puerto en el que se ejecuta y la máquina que lo contiene.
- Puede crear uno o más programas, terminar cualquiera de los programas que crea o hacer que un programa se detenga.
- Se representa mediante un [IDebugProcess2](#) interfaz, que se crea cuando se inicia el proceso. Un proceso es iniciado por el administrador de depuración de sesión (SDM) o [LaunchSuspended](#).

El paquete de depuración puede adjuntar un motor de depuración (DE) a un proceso mediante una llamada a [Attach](#), lo que significa que el DE se asocia a todos los programas posibles que se ejecutan en el proceso que puede controlar. Por ejemplo, si Common Language Runtime DE se asocia a un proceso, solo se adjunta a los programas que ejecutan código administrado.

Vea también

- [Programs](#)
- [Hilos](#)
- [Conceptos del depurador](#)
- [Paquete de depuración](#)
- [Motor de depuración](#)
- [IDebugProcess2](#)
- [LaunchSuspended](#)
- [Attach](#)

Nodos de programa

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un nodo de *programa*:

- Es una descripción ligera de un programa.
- Puede identificarse a sí mismo y el proceso en el que se está ejecutando. Un nodo de programa se puede asociar, separar y describir el motor de depuración (DE) que lo creó, si existe.
- Se representa mediante un [IDebugProgramNode2](#) interfaz, normalmente creado por un DE o puerto. Los nodos de programa se agregan a un puerto llamando a [AddProgramNode](#). Cuando se agrega un nodo de programa a un puerto, se agrega al proceso que contiene el programa que representa este nodo de programa.

En algún momento después de iniciar una sesión de depuración, dependiendo de la implementación del paquete de depuración, los nodos del programa se utilizan para crear los programas correspondientes. Cuando se consulta un proceso para sus programas, se enumeran los programas, uno para cada nodo de programa.

Antes de adjuntar un programa, el IDE solo necesita una descripción ligera del programa. Esta información se puede obtener del nodo del programa. Una vez que se adjunta el programa, el IDE muestra información más detallada, como una lista de todos los subprocessos que se ejecutan en el programa. Esta información se obtiene del propio programa.

Vea también

- [Programs](#)
- [Procesos](#)
- [Motor de depuración](#)
- [Conceptos del depurador](#)
- [IDebugProgramNode2](#)
- [AddProgramNode](#)

Programas

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un *programa*:

- Es un contenedor para un conjunto de subprocesos y un conjunto de módulos. Un programa no tiene una sola analogía en el sistema operativo Windows.

Un programa es una especie de subproceso. Por ejemplo, al depurar un sitio Web, un script se puede ver como un programa. Mientras que un script se ejecuta en el proceso del motor de scripting, independientemente de otros scripts, también tiene su propio conjunto de subprocesos. Un motor de depuración (DE) se asocia a un programa y no a un proceso o un subproceso.

- Puede identificarse a sí mismo y el proceso en el que se está ejecutando. Un programa se puede adjuntar, separar y describir el DE que lo creó, si existe. Un programa también puede ejecutarse, detenerse, continuar y terminarse.
- Puede enumerar todos sus subprocesos. Un programa también puede proporcionar su propia secuencia de desensamblado y puede enumerar todos los contextos de código de una posición de documento determinada.
- Se representa mediante un `IDebugProgram2` interfaz, creado antes de que se adjunta el programa, o como parte del proceso de asociación, dependiendo de la implementación. Cuando un puerto enumera los programas de un proceso, cada programa se crea de acuerdo con una interfaz `IDebugProgramNode2` correspondiente que se pasa como argumento a `AddProgramNode`. Mientras que los `IDebugProgram2` motores de depuración también crean interfaces para representar programas, estos programas no se crean de acuerdo con un nodo de programa. Las `IDebugProgramNode2` interfaces creadas por un DE se utilizan para la depuración real, mientras que las creadas por un puerto se utilizan solo para detectar qué programas se ejecutan en un proceso.

Vea también

- [Procesos](#)
- [Nodos de programa](#)
- [Módulos](#)
- [Conceptos del depurador](#)
- [Motor de depuración](#)
- [Posición del documento](#)
- [Contexto del código](#)
- [IDebugProgram2](#)
- [IDebugProgramNode2](#)
- [AddProgramNode](#)

Subprocesos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un *subproceso*:

- Es la unidad fundamental de cálculo. Un subproceso ejecuta secuencialmente sus instrucciones en el contexto de una pila de llamadas única, pasando de un contexto de código al siguiente.
- Puede identificarse a sí mismo y el programa en el que se está ejecutando. Los subprocesos se pueden nombrar, suspender y reanudar. Un subproceso también puede enumerar sus marcos de pila asociados y, en algunas condiciones, se puede mover a otro marco de pila. Dado el contexto de un marco de pila, un subproceso puede devolver su subproceso lógico asociado, si existe. Un subproceso tiene propiedades, como un recuento de suspensión, que se pueden mostrar en la ventana **Subprocesos** del IDE.
- Se representa mediante una interfaz [IDebugThread2](#), normalmente creada por un motor de depuración (DE) o una máquina virtual como consecuencia de la ejecución de un programa.

Vea también

- [Programs](#)
- [Marcos de pila](#)
- [Motor de depuración](#)
- [Conceptos del depurador](#)
- [Administrador de depuración de sesión](#)

Marcos de pila

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la arquitectura del depurador, un marco de *pila*:

- Es una abstracción de una pila que proporciona el contexto de ejecución de un subproceso. Un subproceso siempre se ejecuta dentro de una función. Un marco de pila contiene las variables locales de la función y los argumentos para ella. Para depurar con Visual Studio, el lenguaje o entorno que se está depurando debe admitir marcos de pila.
- Puede identificarse y describirse a sí mismo, y puede devolver el subproceso asociado. Un marco de pila también puede devolver el contexto de código que representa el puntero de instrucción actual y los contextos de evaluación de documentación y expresión asociados.
- Tiene propiedades que describen el nombre, el tipo y el valor de las variables y argumentos locales, y que aparecen en varias ventanas de depuración IDE.
- Se representa mediante un [IDebugStackFrame2](#) interfaz, normalmente creado por un motor de depuración (DE) o máquina virtual como consecuencia de la ejecución de un subproceso.

Vea también

- [Contextos del depurador](#)
- [Conceptos del depurador](#)
- [Motor de depuración](#)
- [IDebugStackFrame2](#)

Módulos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En cuanto a la arquitectura del depurador, un *módulo*:

- Es un contenedor físico de código, como un archivo ejecutable o un archivo DLL.
- Puede recargar sus símbolos y describirse a sí mismo. Las descripciones del módulo se muestran en la ventana Módulos del IDE.
- Se representa mediante un [IDebugModule2](#) interfaz, creado por un motor de depuración para describir el módulo.

Vea también

- [Conceptos del depurador](#)
- [IDebugModule2](#)

Puntos de interrupción (Visual Studio SDK)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Hay tres tipos de puntos de interrupción: pendiente, enlazado y error.

Un punto de interrupción pendiente:

- Es una abstracción que contiene toda la información necesaria para enlazar un punto de interrupción a uno o varios contextos de código en uno o varios programas. Cada vez que un programa que se está depurando hace que el código se cargue, el motor de depuración comprueba todos los puntos de interrupción pendientes para ver si se pueden enlazar.

Un punto de interrupción pendiente nunca se enlaza al código, sino que se recopila y se dice que contiene todos los puntos de interrupción enlazados que genera.

- Se representa mediante un [IDebugPendingBreakpoint2](#) interfaz.

Un punto de interrupción enlazado:

- Es una abstracción para un punto de interrupción asociado o enlazado a un único contexto de código. Cada punto de interrupción enlazado se genera en respuesta a un punto de interrupción pendiente. Sin embargo, un punto de interrupción pendiente puede generar más de un punto de interrupción enlazado.

Cuando se descarga código, un punto de interrupción enlazado se puede desenlazar y descartar.

- Se representa mediante un [IDebugBoundBreakpoint2](#) interfaz.

Un punto de interrupción de error:

- Es una abstracción para describir un error al intentar enlazar un punto de interrupción pendiente a un contexto de código. Un punto de interrupción de error describe un error en la ubicación o en la propia expresión de punto de interrupción. Para obtener más información, vea Puntos de [interrupción](#) de enlace .

El error de punto de interrupción puede ser un error o una advertencia.

- Se representa mediante un [IDebugErrorBreakpoint2](#) interfaz.

Vea también

- [Programs](#)
- [Conceptos del depurador](#)
- [Contexto del código](#)
- [IDebugBoundBreakpoint2](#)
- [IDebugPendingBreakpoint2](#)
- [IDebugErrorBreakpoint2](#)

Contextos del depurador

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio la depuración, el motor de depuración (DE) funciona simultáneamente dentro de varios contextos distintos, como se indica a continuación:

- El contexto de código, que describe la ubicación actual en la secuencia de ejecución de un programa.
- El contexto o la posición de la documentación, que describe la posición actual dentro de un documento de origen.
- El contexto de evaluación de expresiones, que describe el contexto en el que se llevará a cabo la evaluación de expresiones.

En esta sección

Contexto del código Describe el contexto de código como una dirección en la secuencia de instrucciones de un programa en las arquitecturas de tiempo de ejecución de hoy en día frente a los lenguajes no tradicionales, donde el código puede no estar representado por instrucciones, pero algunos otros medios.

Posición del documento Define la posición del documento en la depuración de Visual Studio mediante una abstracción de una posición en un archivo de origen como se conoce al IDE.

Contexto del documento Describe qué contexto de documento representa en la depuración de Visual Studio en relación con un archivo de origen. También se describe cómo el controlador de símbolos asigna un contexto de código al contexto de documentación.

Contexto de **evaluación de expresiones** Proporciona información sobre un contexto de evaluación de expresiones en Visual Studio. Por ejemplo, un contexto de evaluación de expresiones asociado a un marco de pila proporciona el contexto para evaluar variables locales, parámetros de método y miembros de clase.

Secciones relacionadas

Conceptos de depuración Describe los principales conceptos arquitectónicos de depuración.

Componentes de depuración Proporciona información general de los componentes de depuración de Visual Studio, que incluyen el motor de depuración (DE), el evaluador de expresiones (EE) y el controlador de símbolos (SH).

Tareas de depuración Contiene vínculos a varias tareas de depuración, como iniciar un programa y evaluar expresiones.

Contexto del código

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio la depuración, un contexto de **código**:

- Proporciona una abstracción de una posición en el código como se conoce al motor de depuración (DE). Para la mayoría de las arquitecturas en tiempo de ejecución actuales, un contexto de código se puede considerar como una dirección en la secuencia de instrucciones de un programa. Para los lenguajes no tradicionales, donde el código puede no estar representado por instrucciones, un contexto de código puede estar representado por otros medios.
- Describe la posición actual en la secuencia de ejecución del programa que está depurando.
- Sólo existe cuando un programa se ha detenido en un punto de interrupción.
- Tiene un contexto de documento asociado.
- Se implementa mediante un [IDebugCodeContext2](#) interfaz.

Vea también

- [Contexto del documento](#)
- [Contextos del depurador](#)

Posición del documento

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio la depuración, una posición de *documento*:

- Proporciona una abstracción de una posición en un archivo de origen como se conoce al IDE. Para la mayoría de los idiomas actuales, una posición de documento se puede considerar como una posición en un archivo de origen.
- Describe una posición en un documento de origen en un motor de depuración.
- Se implementa mediante un [IDebugDocumentPosition2](#) interfaz.

Vea también

- [Contexto del código](#)
- [Contexto del documento](#)
- [Proveedor de símbolos](#)
- [Interfaces del proveedor de símbolos](#)
- [Contextos del depurador](#)

Contexto del documento

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio la depuración, un contexto de *documento*:

- Representa una posición en un archivo de origen. Para los idiomas donde el archivo de origen puede no estar presente, un contexto de documento identifica una posición en un documento generado normalmente por el entorno en tiempo de ejecución. Por ejemplo, un motor de scripting podría generar un documento a partir de script. Para obtener más información, consulte [Posición del documento](#).
- Describe una posición en un documento de origen que corresponde a un contexto de código. El controlador de símbolos asigna un contexto de código al contexto de documentación, utilizando la información generada por un compilador o intérprete.
- Se implementa mediante un [IDebugDocumentContext2](#) interfaz.

Vea también

- [Contexto del código](#)
- [Proveedor de símbolos](#)
- [Interfaces del proveedor de símbolos](#)
- [Contextos del depurador](#)

Contexto de evaluación de expresiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Visual Studio la depuración, un contexto de evaluación de **expresiones**:

- Representa un contexto para la evaluación de expresiones. Por lo general, un contexto de evaluación corresponde al ámbito léxico dentro del cual se evalúan variables, parámetros, funciones y métodos. Por ejemplo, un contexto de evaluación de expresiones asociado a un marco de pila proporcionará el contexto para evaluar variables locales, parámetros de método y miembros de clase (si corresponde).
- Existe cuando un programa se ha detenido en un punto de interrupción. La propia expresión es una estructura de datos que representa una expresión analizada que está lista para enlazar y evaluar dentro del contexto especificado.

Con más detalle, las expresiones se crean mediante el [ParseText](#) método. Cuando se evalúa una expresión, genera una cadena imprimible que contiene el nombre y el tipo de variable o argumento y su valor. Esta cadena se muestra en la ventana Inspección o en la ventana Locales del IDE.

Dado `BSTR` un y un `IDebugExpressionContext2` interfaz, un motor de depuración (DE) puede crear un `IDebugExpression2` interfaz mediante el análisis de una expresión. Dada `IDebugExpression2` una interfaz, la DE puede obtener un valor a través de la evaluación de expresiones sincrónicas o asincrónicas. Este valor, junto con el nombre y el tipo de la variable o argumento, se envía al IDE para su presentación.

Vea también

- [Interfaces de evaluación de expresiones](#)
- [Contextos del depurador](#)

Tareas de depuración

06/04/2020 • 3 minutes to read • [Edit Online](#)

Para depurar un programa, debe iniciarse y debe adjuntarse un motor de depuración (DE) o, de lo contrario, el DE debe adjuntarse a un programa iniciado anteriormente. Una vez asociado, el DE debe generar ciertos eventos de inicio. En respuesta, el paquete de depuración intenta enlazar los puntos de interrupción establecidos en el IDE. Cuando el programa alcanza un punto de interrupción enlazado, se detiene y espera la entrada del usuario.

En esta sección

[Problemas de seguridad](#) Describe los pasos de seguridad necesarios para depurar un programa.

[Iniciar un programa](#) Proporciona instrucciones paso a paso sobre cómo especificar un DE, que llama al sistema operativo para iniciar el programa.

[Adjuntar directamente a un programa](#) Describe el proceso utilizado para depurar un programa en un proceso que ya se está ejecutando.

Enviar eventos de [inicio después](#) de un lanzamiento Enumera los eventos que tienen lugar una vez que la DE se adjunta al programa, hasta que el programa está en su punto de entrada principal y está listo para la depuración.

[Control de ejecución](#) Explica cómo el DE normalmente envía un evento de punto de entrada, un evento load-complete o un evento de detención, dependiendo de las circunstancias.

[Enlazar puntos de interrupción](#) Describe cómo, si el usuario establece un punto de interrupción, el IDE formula la solicitud y solicita a la sesión de depuración que cree el punto de interrupción.

[Evaluar expresiones](#) Explica cómo se crean las expresiones y qué sucede cuando se evalúa una expresión.

[Visualizar y ver datos](#) Explica cómo el evaluador de expresiones (EE) admite visualizadores de tipos y visores personalizados.

Secciones relacionadas

[Conceptos del depurador](#) Describe los principales conceptos arquitectónicos de depuración.

[Componentes del depurador](#) Proporciona información general de los componentes de depuración de Visual Studio, que incluyen el DE, EE y el controlador de símbolos (SH).

[Contextos del depurador](#) Explica cómo funciona la DE simultáneamente dentro de los contextos de evaluación de código, documentación y expresión. Describe, para cada uno de los tres contextos, la ubicación, la posición o la evaluación relevantes para ella.

Vea también

[Introducción](#)

Problemas de seguridad

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para depurar un programa mediante Visual Studio, los únicos permisos necesarios son los mismos que un desarrollador necesita para ejecutar el programa. Esto incluye la depuración remota para la mayoría de las situaciones. Algunas situaciones, que implican otros servicios, como el Servicio de Internet Information Service, pueden requerir un mayor nivel de permisos.

Mientras se ejecuta Visual Studio, el Administrador de depuración de procesos (PDM) realiza un seguimiento de los procesos de depuración en el equipo local. De forma remota, el desarrollador inicia un programa llamado *msvsmon.exe* para controlar la depuración remota y hacer que PDM esté disponible. (*msvsmon.exe* no es un servicio y debe iniciarse manualmente para habilitar la depuración remota en ese equipo.) Cuando Visual Studio (o *msvsmon.exe*) no se está ejecutando, no se realiza un seguimiento de ningún proceso para la depuración.

Un desarrollador puede depurar programas que iniciaron sin permisos especiales. El desarrollador puede incluso depurar procesos iniciados por otra persona si esa otra persona es miembro del mismo grupo de seguridad. Y, para habilitar la depuración remota, sólo es necesario copiar los archivos necesarios en el equipo remoto e iniciar *msvsmon.exe*. Para obtener más información, consulte [Depuración remota](#).

Vea también

- [Tareas de depuración](#)
- [Administrador de depuración de procesos](#)
- [Depuración remota](#)

Iniciar un programa

06/04/2020 • 6 minutes to read • [Edit Online](#)

Los usuarios que desean depurar un programa pueden presionar **F5** para ejecutar el depurador desde el IDE. Esto comienza una serie de eventos que en última instancia dan lugar a que el IDE se conecte a un motor de depuración (DE), que a su vez está conectado, o conectado, al programa de la siguiente manera:

1. El IDE llama primero al paquete de proyecto para obtener la configuración de depuración de proyecto activa de la solución. La configuración incluye el directorio inicial, las variables de entorno, el puerto en el que se ejecutará el programa y la DE que se utilizará para crear el programa, si se especifica. Esta configuración se pasa al paquete de depuración.
2. Si se especifica un DE, el DE llama al sistema operativo para iniciar el programa. Como consecuencia del lanzamiento del programa, se carga el entorno en tiempo de ejecución del programa. Por ejemplo, si un programa se escribe en MSIL, se invocará Common Language Runtime para ejecutar el programa.

o bien

Si no se especifica un DE, el puerto llama al sistema operativo para iniciar el programa, lo que hace que el entorno de tiempo de ejecución del programa se cargue.

NOTE

Si se utiliza un DE para iniciar un programa, es probable que se adjunte el mismo DE al programa.

3. Dependiendo de si el DE o el puerto iniciaron el programa, el DE o el entorno en tiempo de ejecución crea una descripción del programa, o nodo, y notifica al puerto que el programa se está ejecutando.

NOTE

Se recomienda que el entorno en tiempo de ejecución cree el nodo de programa, ya que el nodo de programa es una representación ligera de un programa que se puede depurar. No es necesario cargar un DE completo solo para crear y registrar un nodo de programa. Si la DE está diseñada para ejecutarse en el proceso del IDE, pero no se está ejecutando ningún IDE, debe haber un componente que pueda agregar un nodo de programa al puerto.

El programa recién creado, junto con cualquier otro programa, relacionado o no relacionado, iniciado o asociado desde el mismo IDE, componen una sesión de depuración.

Mediante programación, cuando el usuario Visual Studio presiona por primera vez **F5**, el paquete de depuración de `DebugLaunch` 's llama al paquete `VsDebugTargetInfo2` de proyecto (que está asociado con el tipo de programa que se inicia) a través del método, que a su vez rellena una estructura con la configuración de depuración del proyecto activo de la solución. Esta estructura se devuelve al paquete de `LaunchDebugTargets2` depuración a través de una llamada al método. A continuación, el paquete de depuración crea una instancia del administrador de depuración de sesión (SDM), que inicia el programa que se está depurando y los motores de depuración asociados.

Uno de los argumentos pasados al SDM es el GUID de la DE que se usará para iniciar el programa.

Si el GUID `GUID_NULL` DE no lo es, el SDM crea automáticamente el DE y, a continuación, llama a su `LaunchSuspended` método para iniciar el programa. Por ejemplo, si un programa está

`IDebugEngineLaunch2::LaunchSuspended` escrito `CreateProcess` en `ResumeThread` código nativo,

probablemente llamará y (funciones Win32) para ejecutar el programa.

Como consecuencia del lanzamiento del programa, se carga el entorno de tiempo de ejecución del programa. El DE o el entorno en tiempo de ejecución, a continuación, crea un [IDebugProgramNode2](#) interfaz para describir el programa y pasa esta interfaz a [AddProgramNode](#) para notificar al puerto que el programa se está ejecutando.

Si `GUID_NULL` se pasa, el puerto inicia el programa. Una vez que el programa se está `IDebugProgramNode2` ejecutando, el entorno en `IDebugPortNotify2::AddProgramNode` tiempo de ejecución crea una interfaz para describir el programa y lo pasa a . Esto notifica al puerto que el programa se está ejecutando. A continuación, el SDM asocia el motor de depuración al programa en ejecución.

En esta sección

[Notificar al puerto](#) Explica lo que sucede después de que se inicia un programa y se notifica al puerto.

[Adjuntar después de un lanzamiento](#) Documentos cuando la sesión de depuración está lista para adjuntar el DE al programa.

Secciones relacionadas

[Tareas de depuración](#) Contiene vínculos a varias tareas de depuración, como iniciar un programa y evaluar expresiones.

Notificar al puerto

06/04/2020 • 3 minutes to read • [Edit Online](#)

Después de iniciar un programa, el puerto debe ser notificado, de la siguiente manera:

1. Cuando un puerto recibe un nuevo nodo de programa, envía un evento de creación de programa a la sesión de depuración. El evento lleva conlleva una interfaz que representa el programa.
2. La sesión de depuración consulta el programa para el identificador de un motor de depuración (DE) que se puede asociar a.
3. La sesión de depuración comprueba si el DE está en la lista de DE permitidos para ese programa. La sesión de depuración obtiene esta lista de la configuración del programa activo de la solución, que el paquete de depuración le pasó originalmente.

El DE debe estar en la lista permitida, o de lo contrario el DE no se adjuntará al programa.

Mediante programación, cuando un puerto recibe por primera vez un nuevo nodo de programa, crea una interfaz `IDebugProgram2` para representar el programa.

NOTE

Esto no debe confundirse con la `IDebugProgram2` interfaz creada más adelante por el motor de depuración (DE).

El puerto envía un evento de creación de programa `IDebugProgramCreateEvent2` al administrador `IConnectionPoint` de depuración de sesión (SDM) mediante una interfaz COM.

NOTE

Esto no debe confundirse con la `IDebugProgramCreateEvent2` interfaz, que es enviada más adelante por el DE.

Junto con la propia interfaz de eventos, el puerto envía las interfaces `IDebugPort2`, `IDebugProcess2` e `IDebugProgram2`, que representan el puerto, el proceso y el programa, respectivamente. El SDM llama a `IDebugProgram2::GetEngineInfo` para obtener el GUID de la DE que puede depurar el programa. El GUID se obtuvo originalmente de la `IDebugProgramNode2` interfaz.

El SDM marca para ver si el DE está en la lista de DE permitidos. El SDM obtiene esta lista de la configuración del programa activo de la solución, que el paquete de depuración le pasó originalmente. El DE debe estar en la lista permitida, o de lo contrario no se adjuntará al programa.

Una vez que se conoce la identidad del DE, el SDM está listo para adjuntarlo al programa.

Vea también

- [Lanzamiento de un programa](#)
- [Adjuntar después de un lanzamiento](#)
- [Tareas de depuración](#)

Adjuntar después de un lanzamiento

06/04/2020 • 5 minutes to read • [Edit Online](#)

Después de que se inicie un programa, la sesión de depuración está lista para adjuntar el motor de depuración (DE) a dicho programa.

Decisiones de diseño

Dado que la comunicación es más fácil dentro de un espacio de direcciones compartido, debe elegir entre dos enfoques de diseño: establecer la comunicación entre la sesión de depuración y la DE. O bien, establezca la comunicación entre el DE y el programa. Elija entre lo siguiente:

- Si tiene más sentido configurar la comunicación entre la sesión de depuración y el DE, la sesión de depuración crea automáticamente el DE y pide que el DE se asocie al programa. Este diseño deja la sesión de depuración y DE juntas en un espacio de direcciones, y el entorno y el programa en tiempo de ejecución juntos en otro.
- Si tiene más sentido configurar la comunicación entre el DE y el programa, el entorno en tiempo de ejecución crea automáticamente el DE. Este diseño deja el SDM en un espacio de direcciones y el DE, el entorno en tiempo de ejecución y el programa juntos en otro. Este diseño es típico de un DE que se implementa con un intérprete para ejecutar lenguajes con scripts.

NOTE

La forma en que el DE se asocia al programa depende de la implementación. La comunicación entre el DE y el programa también depende de la implementación.

Implementación

Mediante programación, cuando el Administrador de depuración de sesión (SDM) recibe por primera vez el [objeto IDebugProgram2](#) que representa el programa que se va a iniciar, llama al método [Attach](#), pasándole un objeto [IDebugEventCallback2](#), que se usa más adelante para pasar eventos de depuración al SDM. A

`IDebugProgram2::Attach` continuación, el método llama al método [OnAttach](#). Para obtener más información sobre cómo `IDebugProgram2` el SDM recibe la interfaz, consulte [Notificación del puerto](#).

Si la DE necesita ejecutarse en el mismo espacio de direcciones que el programa que está depurando:

`IDebugProgramNodeAttach2::OnAttach` dado `S_FALSE` que la DE suele formar parte de un intérprete que ejecuta un script, el método devuelve `S_FALSE`. La devolución indica que completó el proceso de conexión.

Sin embargo, si el DE se ejecuta en `IDebugProgramNodeAttach2::OnAttach` el `S_OK` espacio de direcciones del SDM: el método devuelve `S_OK`, o el `IDebugProgramNodeAttach2` interfaz no se implementa en absoluto en el `IDebugProgramNode2` objeto asociado con el programa que está depurando. En este caso, el [Attach](#) se llama finalmente al método para completar la operación de conexión.

En este último caso, debe llamar al `IDebugProgram2` método [GetProgramId](#) en el `GUID` objeto que se pasó al `IDebugEngine2::Attach` método, almacenar el objeto de programa local y devolverlo cuando se llame posteriormente al `IDebugProgram2::GetProgramId` método en este objeto. El `GUID` se utiliza para identificar el programa de forma única a través de los diversos componentes de depuración.

En el caso `IDebugProgramNodeAttach2::OnAttach` de `S_FALSE` que `GUID` el método devuelva, el que se `IDebugProgramNodeAttach2::OnAttach` va a `GUID` usar para el programa se pasa a ese método y es el método que

establece el objeto de programa local.

El DE ahora está asociado al programa y listo para enviar cualquier evento de inicio.

Vea también

- [Adjuntar directamente a un programa](#)
- [Notificar al puerto](#)
- [Tareas de depuración](#)
- [IDebugEventCallback2](#)
- [IDebugProgram2](#)
- [Attach](#)
- [GetProgramId](#)
- [IDebugProgramNode2](#)
- [IDebugProgramNodeAttach2](#)
- [OnAttach](#)
- [Attach](#)

Adjuntar directamente a un programa

06/04/2020 • 3 minutes to read • [Edit Online](#)

Los usuarios que desean depurar programas en un proceso que ya se está ejecutando suelen seguir este proceso:

1. En el IDE, elija el comando **Depurar procesos** del menú **Herramientas**.

Aparecerá el cuadro de diálogo **Procesos**.

2. Elija un proceso y haga clic en el botón **Adjuntar**.

Aparece el cuadro de diálogo **Asociar al proceso**, que enumera todos los motores de depuración (DE) instalados en el equipo.

3. Especifique los DE que se utilizarán para depurar el proceso seleccionado y, a continuación, haga clic en **Aceptar**.

El paquete de depuración inicia una sesión de depuración y le pasa la lista de DE. La sesión de depuración a su vez pasa esta lista, junto con una función de devolución de llamada, al proceso seleccionado y, a continuación, pide al proceso que enumere sus programas en ejecución.

Mediante programación, en respuesta a la solicitud del usuario, el paquete de depuración crea una instancia del administrador de depuración de sesión (SDM) y le pasa la lista de DE seleccionados. Junto con la lista, el paquete de depuración pasa el SDM un [IDebugEventCallback2](#) interfaz. El paquete de depuración pasa la lista de DE al proceso seleccionado llamando a [IDebugProcess2::Attach](#). A continuación, el SDM llama a [IDebugProcess2::EnumPrograms](#) en el puerto para enumerar los programas que se ejecutan en el proceso.

A partir de este momento, cada motor de depuración se adjunta a un programa exactamente como se detalla en [Adjuntar después de un lanzamiento](#), con dos excepciones.

Para mayor eficiencia, los DE que se implementan para compartir un espacio de direcciones con el SDM se agrupan de modo que cada DE tenga un conjunto de programas a los que se adjuntará. En este caso, [IDebugProcess2](#) llama a [IDebugEngine2::Attach](#) y le pasa una matriz de programas para adjuntar a.

La segunda excepción es que los eventos de inicio enviados por un DE adjunto a un programa que ya se está ejecutando no suelen incluir el evento de punto de entrada.

Vea también

- [Envío de eventos de inicio después de un lanzamiento](#)
- [Tareas de depuración](#)

Enviar eventos de inicio después de un lanzamiento

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una vez que el motor de depuración (DE) se asocia al programa, envía una serie de eventos de inicio de nuevo a la sesión de depuración.

Los eventos de inicio enviados de vuelta a la sesión de depuración incluyen:

- Un evento de creación de motores.
- Un evento de creación de programas.
- Eventos de creación de subprocesos y carga de módulos.
- Un evento load complete, enviado cuando se carga el código y está listo para ejecutarse, pero antes de que se ejecute cualquier código.

NOTE

Cuando se continúa con este evento, se inicializan las variables globales y se ejecutan las rutinas de inicio.

- Posibles otros eventos de creación de subprocesos y carga de módulos.
- Un evento de punto de entrada, que indica que el **Main** programa `WinMain` ha alcanzado su punto de entrada principal, como `Main` o `.`. Normalmente, este evento no se envía si la DE se adjunta a un programa que ya se está ejecutando.

Mediante programación, el DE primero envía el Administrador de depuración de sesión (SDM) un [IDebugEngineCreateEvent2](#) interfaz, que representa un evento de creación de motor, seguido de un [IDebugProgramCreateEvent2](#), que representa un evento de creación de programa.

Estos eventos suelen ir seguidos de uno o varios eventos de creación de subprocesos [IDebugThreadCreateEvent2](#) y eventos de carga de módulo [IDebugModuleLoadEvent2](#).

Cuando el código se carga y está listo para ejecutarse, pero antes de que se ejecute cualquier código, el DE envía el SDM un [IDebugLoadCompleteEvent2](#) carga completa eventos. Por último, si el programa aún no se está ejecutando, la DE envía un evento de punto de entrada [IDebugEntryPointEvent2](#), lo que indica que el programa ha alcanzado su punto de entrada principal y está listo para la depuración.

Vea también

- [Control de ejecución](#)
- [Tareas de depuración](#)

Control de ejecución

06/04/2020 • 7 minutes to read • [Edit Online](#)

El motor de depuración (DE) normalmente envía uno de los siguientes eventos como el último evento de inicio:

- El evento de punto de entrada, si se adjunta a un programa recién lanzado
- El evento load complete, si se adjunta a un programa que ya se está ejecutando

Ambos eventos detienen eventos, lo que significa que el DE espera una respuesta del usuario mediante el IDE. Para obtener más información, consulte [Modos operativos](#).

Detener el evento

Cuando se envía un evento de detención a la sesión de depuración:

1. El programa y el subproceso que contienen el puntero de instrucción actual se pueden obtener desde la interfaz de eventos.
2. El IDE determina el archivo de código fuente actual y la posición, que se muestra como resaltado en el editor.
3. La sesión de depuración normalmente responde a este primer evento de detención llamando al método **Continue** del programa.
4. A continuación, el programa se ejecuta hasta que encuentra una condición de detención, como golpear un punto de interrupción. En ese caso, el DE envía un evento de punto de interrupción a la sesión de depuración. El evento de punto de interrupción es un evento de detención y el DE espera de nuevo una respuesta del usuario.
5. Si el usuario elige entrar, pasar o salir de una función, el IDE solicita **Step** a la sesión de depuración que llame al método del programa. A continuación, el IDE pasa la unidad de paso (instrucción, instrucción o línea) y el tipo de paso (ya sea para entrar, superar o salir de la función). Cuando se completa el paso, el DE envía un evento complete del paso a la sesión de depuración, que es un evento de detención.

o bien

Si el usuario elige continuar la ejecución desde el puntero de instrucción actual, el IDE solicita a la sesión de depuración que llame al método **Execute** del programa. El programa reanuda la ejecución hasta que encuentra la siguiente condición de detención.

o bien

Si la sesión de depuración debe omitir un evento de detención determinado, la sesión de depuración llama al método **Continue** del programa. Si el programa entraba, entraba o salía de una función cuando se encontraba con la condición de detención, continúa el paso.

Mediante programación, cuando el DE encuentra una condición de detención, envía eventos de detención como [IDebugLoadCompleteEvent2](#) o [IDebugEntryPointEvent2](#) al administrador de depuración de sesión (SDM) mediante una [interfaz IDebugEventCallback2](#). El DE pasa el [IDebugProgram2](#) y [IDebugThread2](#) interfaces que representan el programa y el subproceso que contiene el puntero de instrucción actual. El SDM llama a [IDebugThread2::EnumFrameInfo](#) para obtener el marco de pila superior y llama a [IDebugStackFrame2::GetDocumentContext](#) para obtener el contexto de documento asociado con el puntero de instrucción actual. Este contexto de documento suele ser un nombre de archivo de código fuente, una

línea y un número de columna. El IDE los usa para resaltar el código fuente que contiene el puntero de instrucción actual.

El SDM normalmente responde a este primer evento de detención llamando a [IDebugProgram2::Continue](#). A continuación, el programa se ejecuta hasta que encuentra una condición de detención, como golpear un punto de interrupción, en cuyo caso la DE envía una [interfaz IDebugBreakpointEvent2](#) al SDM. El evento de punto de interrupción es un evento de detención y el DE espera de nuevo una respuesta del usuario.

Si el usuario elige entrar, pasar o salir de una función, el IDE solicita al SDM que llame a [IDebugProgram2::Step](#). A continuación, el IDE pasa [STEPUNIT](#) (instrucción, instrucción o línea) y [STEPKIND](#), es decir, si se debe entrar, superar o salir de la función. Cuando se completa el paso, el DE envía un [IDebugStepCompleteEvent2](#) interfaz al SDM, que es un evento de detención.

Si el usuario decide continuar la ejecución desde el puntero de instrucción actual, el IDE pide al SDM que llame a [IDebugProgram2::Execute](#). El programa reanuda la ejecución hasta que encuentra la siguiente condición de detención.

Si el paquete de depuración debe omitir un evento de detención determinado, el paquete de depuración llama al SDM, que llama a [IDebugProgram2::Continue](#). Si el programa entraba, entraba o salía de una función cuando se encontraba con la condición de detención, continúa el paso. Esto implica que el programa mantiene un estado de paso a paso, para que sepa cómo continuar.

Las llamadas que [Step](#) realiza el SDM a [Execute](#) y [Continue](#) son asincrónicas, lo que significa que el SDM espera que la llamada vuelva rápidamente. Si el DE envía al SDM un evento [Step](#) de detención en el mismo subproceso antes de que [Ejecutar](#) o [Continuar](#) devuelva, el SDM se bloquea.

Vea también

- [Tareas de depuración](#)

Enlazar puntos de interrupción

06/04/2020 • 6 minutes to read • [Edit Online](#)

Si el usuario establece un punto de interrupción, quizás presionando **F9**, el IDE formula la solicitud y solicita a la sesión de depuración que cree el punto de interrupción.

Establecer un punto de interrupción

Establecer un punto de interrupción es un proceso de dos pasos, porque es posible que el código o los datos afectados por el punto de interrupción aún no estén disponibles. En primer lugar, se debe describir el punto de interrupción y, a continuación, a medida que el código o los datos estén disponibles, debe estar enlazado a ese código o datos, como se indica a continuación:

1. El punto de interrupción se solicita a los motores de depuración (DE) relevantes y, a continuación, el punto de interrupción se enlaza al código o a los datos a medida que esté disponible.
2. La solicitud de punto de interrupción se envía a la sesión de depuración, que la envía a todos los DE relevantes. Cualquier DE que elija controlar el punto de interrupción crea un punto de interrupción pendiente correspondiente.
3. La sesión de depuración recopila los puntos de interrupción pendientes y los devuelve al paquete de depuración (el componente de depuración de Visual Studio).
4. El paquete de depuración solicita a la sesión de depuración que enlace el punto de interrupción pendiente al código o a los datos. La sesión de depuración envía esta solicitud a todos los DE relevantes.
5. Si el DE puede enlazar el punto de interrupción, envía un evento enlazado de punto de interrupción a la sesión de depuración. Si no es así, envía un evento de error de punto de interrupción en su lugar.

Puntos de interrupción pendientes

Un punto de interrupción pendiente puede enlazarse a varias ubicaciones de código. Por ejemplo, una línea de código fuente para una plantilla C++ puede enlazarse a cada secuencia de código generada a partir de la plantilla. La sesión de depuración puede usar un evento enlazado a punto de interrupción para enumerar los contextos de código enlazados a un punto de interrupción en el momento en que se envió el evento. Más contextos de código se pueden enlazar más adelante, por lo que la DE puede enviar varios eventos enlazados de punto de interrupción para cada solicitud de enlace. Sin embargo, un DE debe enviar solo un evento de error de punto de interrupción por solicitud de enlace.

Implementación

Mediante programación, el paquete de depuración llama al administrador de depuración de sesión (SDM) y le proporciona una interfaz `IDebugBreakpointRequest2` que ajusta una estructura `BP_REQUEST_INFO`, que describe el punto de interrupción que se va a establecer. Aunque los puntos de interrupción pueden ser de muchos formularios, en última instancia se resuelven en un contexto de código o datos.

El SDM pasa esta llamada a cada DE relevante mediante una llamada a su `CreatePendingBreakpoint` método. Si el DE elige controlar el punto de interrupción, crea y devuelve un `IDebugPendingBreakpoint2` interfaz. El SDM recoge estas interfaces y las pasa de `IDebugPendingBreakpoint2` nuevo al paquete de depuración como una sola interfaz.

Hasta ahora, no se han generado eventos.

A continuación, el paquete de depuración intenta enlazar el punto de interrupción pendiente al código o a los datos mediante una llamada a [Bind](#), que se implementa mediante la DE.

Si el punto de interrupción está enlazado, el DE envía un [IDebugBreakpointBoundEvent2](#) interfaz de eventos al paquete de depuración. El paquete de depuración utiliza esta interfaz para enumerar todos los contextos de código (o el contexto de datos único) enlazados al punto de interrupción mediante una llamada a [EnumBoundBreakpoints](#), que devuelve una o varias [interfaces IDebugBoundBreakpoint2](#). El [GetBreakpointResolution](#) interfaz devuelve un [IDebugBreakpointResolution2](#) interfaz y [GetResolutionInfo](#) devuelve un [BP_RESOLUTION_INFO](#) unión que contiene el código o el contexto de datos.

Si el DE no puede enlazar el punto de interrupción, envía una única interfaz de eventos [IDebugBreakpointErrorEvent2](#) al paquete de depuración. El paquete de depuración recupera el tipo de error (error o advertencia) y el mensaje informativo llamando a [GetErrorBreakpoint](#), seguido de [GetBreakpointResolution](#) y [GetResolutionInfo](#). Esto devuelve una estructura [BP_ERROR_RESOLUTION_INFO](#) que contiene el tipo de error y el mensaje.

Si un DE controla un punto de interrupción `BPET_TYPE_ERROR` pero no puede enlazarlo, devuelve un error de tipo . El paquete de depuración responde mostrando un cuadro de diálogo de error y el IDE coloca un glifo de exclamación dentro del glifo de punto de interrupción a la izquierda de la línea de código fuente.

Si un DE controla un punto de interrupción, no puede enlazarlo, pero alguna otra DE podría enlazarlo, devuelve una advertencia. El IDE responde colocando un glifo de pregunta dentro del glifo de punto de interrupción a la izquierda de la línea de código fuente.

Vea también

- [Tareas de depuración](#)

Evaluar expresiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

Las expresiones se crean a partir de cadenas que se pasan desde las **ventanas Autos, Watch, QuickWatch o Immediate**. Cuando se evalúa una expresión, genera una cadena imprimible que contiene el nombre y el tipo de variable o argumento y su valor. Esta cadena se muestra en la ventana IDE correspondiente.

Implementación

Las expresiones se evalúan cuando un programa se ha detenido en un punto de interrupción. La expresión en sí se representa mediante un `IDebugExpression2` interfaz, que representa una expresión analizada que está lista para el enlace y la evaluación dentro del contexto de evaluación de expresión dado. El marco de pila determina el contexto de evaluación de expresiones, que el motor de depuración (DE) proporciona mediante la implementación de la `IDebugExpressionContext2` interfaz.

Dado una cadena de usuario y un `IDebugExpressionContext2` interfaz, un motor de depuración (DE) puede obtener un `IDebugExpression2` interfaz pasando la cadena de usuario a la `IDebugExpressionContext2::arse :PText` método. El `IDebugExpression2` interfaz que se devuelve contiene la expresión analizada lista para la evaluación.

Con `IDebugExpression2` la interfaz, la DE puede obtener el valor de la expresión mediante la evaluación de expresiones sincrónicas o asincrónicas, mediante `IDebugExpression2::EvaluateSync` o `IDebugExpression2::EvaluateAsync`. Este valor, junto con el nombre y el tipo de la variable o argumento, se envía al IDE para su presentación. El valor, el nombre y el tipo se representan mediante una interfaz `IDebugProperty2`.

Para habilitar la evaluación de expresiones, un DE debe implementar el `IDebugExpression2` y `IDebugExpressionContext2` interfaces. La evaluación sincrónica y asincrónica requiere la implementación de la `IDebugProperty2::GetPropertyInfo` método.

Vea también

- [Marcos de pila](#)
- [Contexto de evaluación de expresiones](#)
- [Tareas de depuración](#)

Visualización y visualización de datos

06/04/2020 • 6 minutes to read • [Edit Online](#)

Los visualizadores de tipos y los visores personalizados presentan los datos de una manera que es rápidamente significativa para un desarrollador. El evaluador de expresiones (EE) puede admitir visualizadores de tipos de terceros, así como proporcionar sus propios visores personalizados.

Visual Studio determina cuántos visualizadores de tipos y visores personalizados están asociados con el tipo del objeto mediante una llamada a la `GetCustomViewerCount` método. Si hay al menos un visualizador de tipo o un visor personalizado disponible, Visual Studio llama al método `GetCustomViewerList` para recuperar una lista de esos visualizadores y visores (en realidad, una lista de `s` que implementa los visualizadores y visores) y los presenta al usuario.

Compatibilidad con visualizadores de tipos

Hay una serie de interfaces que EE debe implementar para admitir visualizadores de tipos. Estas interfaces se pueden dividir en dos amplias categorías: interfaces que enumeran los visualizadores de tipos y las interfaces que tienen acceso a los datos de propiedad.

Visualizadores de tipo de listado

El EE admite la lista de visualizadores de tipos en su implementación de `IDebugProperty3::GetCustomViewerCount` y `IDebugProperty3::GetCustomViewerList`. Estos métodos pasan la llamada a los métodos correspondientes `GetCustomViewerCount` y `GetCustomViewerList`.

El `IEEVisualizerService` se obtiene mediante una llamada a `CreateVisualizerService`. Este método requiere el `IDebugBinder3` interfaz, que se obtiene de la `IDebugBinder` interfaz pasa a `EvaluateSync`. `IEEVisualizerServiceProvider::CreateVisualizerService` también requiere el `IDebugSymbolProvider` y `IDebugAddress` interfaces, que se pasaron a `IDebugParsedExpression::EvaluateSync`. La interfaz final `IEEVisualizerService` necesaria para crear la interfaz es la interfaz `IEEVisualizerDataProvider`, que implementa EE. Esta interfaz permite realizar cambios en la propiedad que se visualiza. Todos los datos de propiedad se encapsulan en un `IDebugObject` interfaz, que también se implementa mediante el EE.

Acceso a los datos de propiedad

El acceso a los datos de propiedad se realiza a través de la interfaz `IPropertyProxyEESide`. Para obtener esta interfaz, Visual Studio llama a `QueryInterface` en el objeto de propiedad para obtener el `IPropertyProxyProvider` interfaz (implementada en `IPropertyProxyEESide` el mismo objeto que implementa el `IDebugProperty3` interfaz) y, a continuación, llama a la `GetPropertyProxy` método para obtener la interfaz.

Todos los datos pasados `IPropertyProxyEESide` dentro y fuera de la interfaz se encapsulan en la interfaz `IEEDataStorage`. Esta interfaz representa una matriz de bytes y se implementa mediante Visual Studio y EE. Cuando se van a cambiar los datos de `IEEDataStorage` una propiedad, Visual Studio crea un objeto que contiene `IEEDataStorage` los nuevos datos y llama a `CreateReplacementObject` con ese objeto de datos para obtener un nuevo objeto que, a su vez, se pasa a `InPlaceUpdateObject` para actualizar los datos de la propiedad. `IPropertyProxyEESide::CreateReplacementObject` permite que EE cree una instancia `IEEDataStorage` de su propia clase que implementa la interfaz.

Apoyo a los espectadores personalizados

El `DBG_ATTRIB_VALUE_CUSTOM_VIEWER` indicador se `dwAttrib` establece en el campo de la estructura `DEBUG_PROPERTY_INFO` (devuelta por una llamada a `GetPropertyInfo`) para indicar que el objeto tiene un visor

personalizado asociado. Cuando se establece esta marca, Visual Studio obtiene el [IDebugProperty3](#) interfaz de la [IDebugProperty2](#) interfaz mediante [QueryInterface](#).

Si el usuario selecciona un visor personalizado, Visual Studio crea `CLSID` una instancia

`IDebugProperty3::GetCustomViewerList` del visor personalizado mediante el visor proporcionado por el método. Visual Studio, a continuación, llama a [DisplayValue](#) para mostrar el valor al usuario.

Normalmente, `IDebugCustomViewer::DisplayValue` presenta una vista de solo lectura de los datos. Para permitir cambios en los datos, EE debe implementar una interfaz personalizada que admita el cambio de datos en un objeto de propiedad. El `IDebugCustomViewer::DisplayValue` método utiliza esta interfaz personalizada para admitir el cambio de datos. El método busca la interfaz personalizada en la `IDebugProperty2` interfaz pasada como `pDebugProperty` argumento.

Compatibilidad con visualizadores de tipos y visores personalizados

Un EE puede admitir visualizadores de tipos y visores personalizados en los métodos [GetCustomViewerCount](#) y [GetCustomViewerList](#). En primer lugar, el EE agrega el número de visores personalizados que proporciona al valor devuelto por el [GetCustomViewerCount](#) método. En segundo lugar, el `CLSID` EE anexa las s de sus propios visores personalizados a la lista devuelta por el [GetCustomViewerList](#) método.

Vea también

- [Tareas de depuración](#)
- [Visualizador de tipos y visor personalizado](#)

Escribir un evaluador de expresiones de Common Language Runtime

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

El evaluador de expresiones (EE) es la parte de un motor de depuración (DE) que controla la sintaxis y la semántica del lenguaje de programación que produjo el código que se está depurando. Las expresiones deben evaluarse en el contexto de un lenguaje de programación. Por ejemplo, en algunos idiomas, la expresión "A+B" significa "la suma de A y B." En otros idiomas, la misma expresión podría significar "A o B." Por lo tanto, se debe escribir un EE independiente para cada lenguaje de programación que genera código de objeto que se va a depurar en el IDE de Visual Studio.

Algunos aspectos del paquete de depuración de Visual Studio deben interpretar el código en el contexto del lenguaje de programación. Por ejemplo, cuando la ejecución se detiene en un punto de interrupción, se deben evaluar y mostrar todas las expresiones que el usuario haya escrito en una ventana **Inspección**. El usuario puede cambiar el valor de una variable local escribiendo una expresión en una ventana **Inspección** o en la ventana **Inmediato**.

En esta sección

[Common Language Runtime y evaluación de expresiones](#) Explica que cuando se integra el lenguaje de programación propietario en el IDE de Visual Studio, escribir un EE capaz de evaluar expresiones en el contexto del lenguaje propietario le permite compilar en un lenguaje intermedio de Microsoft (MSIL) sin escribir un motor de depuración.

[Arquitectura del evaluador](#) de expresiones Describe cómo implementar las interfaces EE necesarias y llamar al proveedor de símbolos de Common Language Runtime (SP) y a las interfaces de enlazador.

[Registrar un evaluador](#) de expresiones Tenga en cuenta que el EE debe registrarse como un generador de clases con common Language Runtime y entornos de tiempo de ejecución de Visual Studio.

[Implementar un evaluador](#) de expresiones Describe cómo el proceso de evaluación de una expresión incluye el motor de depuración (DE), el proveedor de símbolos (SP), el objeto de enlazador y el evaluador de expresiones (EE).

[Mostrar locales](#) Describe cómo, cuando se detiene la ejecución, el paquete de depuración llama a la DE para obtener una lista de variables y argumentos locales.

[Evaluar una expresión de ventana de inspección](#) Documenta cómo el paquete de depuración de Visual Studio llama a la DE para determinar el valor actual de cada expresión en su lista de inspección.

[Cambiar el valor de un local](#) Explica que al cambiar el valor de un local, cada línea de la ventana Locales tiene un objeto asociado que proporciona el nombre, el tipo y el valor actual de un local.

[Implementar visualizadores de tipos y visores personalizados](#) Explica qué interfaz debe implementarse mediante el componente para admitir visualizadores de tipos y visores personalizados.

Veá también

[Extensibilidad del depurador de Visual Studio](#)

Common Language Runtime y evaluación de expresiones

06/04/2020 • 5 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, consulte Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Los compiladores, como Visual Basic y C- (pronunciado C-sharp), que se dirigen a Common Language Runtime (CLR), producen Microsoft Intermediate Language (MSIL), que más adelante se compila en código nativo. CLR proporciona un motor de depuración (DE) para depurar el código resultante. Si planea integrar el lenguaje de programación propietario en el IDE de Visual Studio, puede elegir compilar en MSIL y, por lo tanto, no tendrá que escribir su propio DE. Sin embargo, tendrá que escribir un evaluador de expresiones (EE) que sea capaz de evaluar expresiones en el contexto de su lenguaje de programación.

Discusión

Las expresiones de lenguaje informático generalmente se analizan para generar un conjunto de objetos de datos y un conjunto de operadores utilizados para manipularlos. Por ejemplo, la expresión "A+B" se puede analizar para aplicar el operador de adición (+) a los objetos de datos "A" y "B", lo que puede dar lugar a otro objeto de datos. El conjunto total de objetos de datos, operadores y sus asociaciones se representan con mayor frecuencia en un programa como un árbol, con los operadores en los nodos del árbol y los objetos de datos en las ramas. Una expresión que se ha desglosado en forma de árbol a menudo se denomina árbol analizado.

Una vez analizada una expresión, se llama a un proveedor de símbolos (SP) para evaluar cada objeto de datos. Por ejemplo, si "A" se define en más de un método, la pregunta "¿Qué A?" debe ser respondido antes de que el valor de A pueda ser determinado. La respuesta devuelta por el SP es algo así como "El tercer elemento en el quinto marco de pila" o "La A que está 50 bytes más allá del inicio de la memoria estática asignada a este método."

Además de producir MSIL para el propio programa, los compiladores CLR también pueden producir información de depuración muy descriptiva que se escribe en un archivo De base de datos de programa (.pdb). Siempre que un compilador de lenguaje propietario genere información de depuración en el mismo formato que los compiladores CLR, el SP de CLR puede identificar los objetos de datos con nombre de ese lenguaje. Una vez que se ha identificado un objeto de datos con nombre, EE utiliza un objeto de enlazador para asociar (o enlazar) el objeto de datos al área de memoria que contiene el valor de ese objeto. A continuación, la DE puede obtener o establecer un nuevo valor para el objeto de datos.

Un compilador propietario puede proporcionar información de `ISymbolWriter` depuración de CLR llamando a la `System.Diagnostics.SymbolStore` interfaz (que se define en .NET Framework en el espacio de nombres). Al compilar en MSIL y escribir información de depuración a través de estas interfaces, un compilador propietario puede usar CLR DE y SP. Esto simplifica en gran medida la integración de un lenguaje propietario en el IDE de Visual Studio.

Cuando el CLR DE llama a la EE propietaria para evaluar una expresión, el DE proporciona el EE con interfaces a un SP y un objeto de enlazador. Por lo tanto, escribir un motor de depuración basado en CLR significa que solo es necesario implementar las interfaces de evaluador de expresiones adecuadas; CLR se encarga del enlace y el control de símbolos por usted.

Vea también

- [Escribir un evaluador de expresiones CLR](#)

Arquitectura del evaluador de expresiones

06/04/2020 • 6 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

La integración de un lenguaje propietario en el paquete de depuración de Visual Studio significa que debe configurar las interfaces de evaluador de expresiones (EE) necesarias y llamar al proveedor de símbolos en tiempo de ejecución (SP) de lenguaje común e interfaces de enlazador. Los objetos SP y binder, junto con la dirección de ejecución actual, son el contexto en el que se evalúan las expresiones. La información que estas interfaces producen y consumen representa los conceptos clave en la arquitectura de un EE.

Información general

Analizar la expresión

Al depurar un programa, las expresiones se evalúan por varias razones, pero siempre cuando el programa que se está depurando se ha detenido en un punto de interrupción (ya sea un punto de interrupción colocado por el usuario o uno causado por una excepción). Es en este momento cuando Visual Studio obtiene un marco de pila, representado por el `IDebugStackFrame2` interfaz, desde el motor de depuración (DE). Visual Studio, a continuación, llama a `GetExpressionContext` para obtener el `IDebugExpressionContext2` interfaz. Esta interfaz representa un contexto en el que se pueden evaluar expresiones; `ParseText` es el punto de entrada al proceso de evaluación. Hasta este punto, todas las interfaces son implementadas por el DE.

Cuando `IDebugExpressionContext2::ParseText` se llama, el DE crea una instancia del EE asociado con el idioma del archivo de origen donde se produjo el punto de interrupción (el DE también crea una instancia de la SH en este punto). El EE se representa mediante el `IDebugExpressionEvaluator` interfaz. A continuación, el DE llama a `Parse` para convertir la expresión (en forma de texto) en una expresión analizada, lista para la evaluación. Esta expresión analizada se representa mediante el `IDebugParsedExpression` interfaz. Normalmente, la expresión se analiza y no se evalúa en este momento.

La DE crea un objeto que implementa el `IDebugExpression2` interfaz, coloca `IDebugExpression2` el `IDebugExpressionContext2::ParseText` `IDebugParsedExpression` objeto en el objeto y devuelve el objeto desde .

Evaluar la expresión

Visual Studio llama a `EvaluateSync` o `EvaluateAsync` para evaluar la expresión analizada. Ambos métodos llaman a `IDebugExpression2::EvaluateSync` `EvaluateSync` (`IDebugExpression2::EvaluateAsync` llama al método inmediatamente, mientras que llama al método a través de un subproceso en segundo plano) para evaluar la expresión analizada y devolver un `IDebugProperty2` interfaz que representa el valor y el tipo de la expresión analizada. `IDebugParsedExpression::EvaluateSync` utiliza el SH, la dirección y el enlazador proporcionados para convertir `IDebugProperty2` la expresión analizada en un valor real, representado por la interfaz.

Por ejemplo

Después de alcanzar un punto de interrupción en un programa en ejecución, el usuario elige ver una variable en el cuadro de diálogo **Inspección rápida**. Este cuadro de diálogo muestra el nombre de la variable, su valor y su tipo. Normalmente, el usuario puede cambiar el valor.

Cuando se muestra el cuadro de diálogo **Inspección rápida**, el nombre de la variable que se examina se envía como texto a [ParseText](#). Esto devuelve un [IDebugExpression2](#) objeto que representa la expresión analizada, en este caso, la variable. A continuación, se llama `IDebugProperty2` a `EvaluateSync` para generar un objeto que representa el valor y el tipo de la variable, así como su nombre. Es esta información la que se muestra.

Si el usuario cambia el valor de la variable, [SetValueAsString](#) se llama con el nuevo valor, que cambia el valor de la variable en la memoria para que se usará cuando el programa reanude la ejecución.

Consulte Visualización de [valores locales](#) para obtener más detalles sobre este proceso de visualización de los valores de las variables. Consulte [Cambiar el valor de un local](#) para obtener más detalles sobre cómo se cambia el valor de una variable.

En esta sección

[Contexto de evaluación](#) Proporciona los argumentos que se pasan cuando el DE llama a EE.

[Interfaces del evaluador](#) de expresiones clave Describe las interfaces cruciales necesarias al escribir un EE, junto con el contexto de evaluación.

Vea también

- [Escribir un evaluador de expresiones CLR](#)
- [Mostrando locales](#)
- [Cambiar el valor de un local](#)

Contexto de evaluación

06/04/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de expresiones CLR y Ejemplo de evaluador de expresiones administradas.

Cuando el motor de depuración (DE) llama al evaluador de expresiones (EE), tres argumentos que se pasan a `EvaluateSync` determinan el contexto para buscar y evaluar símbolos, como se muestra en la tabla siguiente.

Argumentos

ARGUMENTO	DESCRIPCIÓN
<code>pSymbolProvider</code>	Una interfaz <code>IDebugSymbolProvider</code> que especifica el controlador de símbolos (SH) que se usará para identificar el símbolo.
<code>pAddress</code>	Una interfaz <code>IDebugAddress</code> que especifica el punto de ejecución actual. Esta interfaz busca el método que contiene el código que se está ejecutando.
<code>pBinder</code>	Una interfaz <code>IDebugBinder</code> que busca el valor y el tipo de un símbolo con su nombre.

`IDebugParsedExpression::EvaluateSync` devuelve un `IDebugProperty2` interfaz que representa el valor resultante y su tipo.

Vea también

- [Interfaces del evaluador de expresiones clave](#)
- [Mostrando locales](#)
- [EvaluateSync](#)
- [IDebugProperty2](#)
- [IDebugSymbolProvider](#)
- [IDebugAddress](#)
- [IDebugBinder](#)

Interfaces del evaluador de expresiones clave

06/04/2020 • 3 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, consulte Evaluadores de expresiones CLR y Ejemplo de evaluador de expresiones administradas.

Al escribir un evaluador de expresiones (EE), junto con el contexto de evaluación, debe estar familiarizado con las interfaces siguientes.

Descripciones de la interfaz

- [IDebugAddress](#)

Tiene un único método, [GetAddress](#), que obtiene una estructura de datos que representa el punto actual de ejecución. Esta estructura de datos es uno de los tres argumentos que el motor de depuración (DE) pasa al método [EvaluateSync](#) para evaluar una expresión. Esta interfaz normalmente la implementa el proveedor de símbolos.

- [IDebugBinder](#)

Tiene el [bind](#) método, que obtiene el área de memoria que contiene el valor actual de un símbolo. Dado el método contenedor, representado por un [IDebugObject](#) objeto y el propio símbolo, `IDebugBinder::Bind` representado por un [IDebugField](#) objeto, devuelve el valor del símbolo. `IDebugBinder` normalmente es implementado por el DE.

- [IDebugField](#)

Representa un tipo de datos simple. Para tipos más complejos, como matrices y métodos, use las derivadas [IDebugArrayField](#) y [IDebugMethodField](#) interfaces, respectivamente. [IDebugContainerField](#) es otra interfaz derivada importante que representa símbolos que contienen otros símbolos, como métodos o clases. La `IDebugField` interfaz (y sus derivados) normalmente la implementa el proveedor de símbolos.

Un `IDebugField` objeto se puede utilizar para buscar el nombre y el tipo de un símbolo y, junto con un [IDebugBinder](#) objeto, se puede utilizar para buscar su valor.

- [IDebugObject](#)

Representa los bits reales del valor en tiempo de ejecución de un símbolo. [Bind](#) toma un [IDebugField](#) objeto, que representa un símbolo y devuelve un [IDebugObject](#) objeto. El [GetValue](#) método devuelve el valor del símbolo en un búfer de memoria. Un DE normalmente implementa esta interfaz para representar el valor de una propiedad en la memoria.

- [IDebugExpressionEvaluator](#)

Esta interfaz representa el propio evaluador de expresiones. El método de clave es [Parse](#), que devuelve una interfaz [IDebugParsedExpression](#).

- [IDebugParsedExpression](#)

Esta interfaz representa una expresión analizada lista para evaluarse. El método clave es [EvaluateSync](#) que

devuelve un `IDebugProperty2` que representa el valor y el tipo de la expresión.

- [IDebugProperty2](#)

Esta interfaz representa un valor y su tipo y es el resultado de una evaluación de expresiones.

Vea también

- [Contexto de evaluación](#)

Registrar un evaluador de expresiones

06/04/2020 • 7 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

El evaluador de expresiones (EE) debe registrarse como un generador de clases con el entorno COM de Windows y Visual Studio. Un EE se configura como un archivo DLL para que se inyecte en el espacio de direcciones del motor de depuración (DE) o en el espacio de direcciones de Visual Studio, dependiendo de la entidad que cree una instancia de EE.

Evaluador de expresiones de código administrado

Un EE de código administrado se implementa como una biblioteca de clases, que es un archivo DLL que se registra con el entorno COM, normalmente iniciado por una llamada al programa VSIP, *regpkg.exe*. El proceso real de escribir las claves del Registro para el entorno COM se controla automáticamente.

Un método de la clase [ComRegisterFunctionAttribute](#) principal se marca con `RegisterClass`, lo que indica que se debe llamar al método cuando el archivo DLL se está registrando con COM. Este método de `RegisterClass` registro, a menudo llamado `RegisterClass`, realiza la tarea de registrar el archivo DLL con Visual Studio. Un `UnregisterClass` correspondiente (marcado [ComUnregisterFunctionAttribute](#) con el `UnregisterClass`), deshace los efectos de cuando se desinstala el archivo DLL. Las mismas entradas de registro se realizan como para un EE escrito en código no administrado; la única diferencia es que no `SetEEMetric` hay ninguna función auxiliar como hacer el trabajo por usted. A continuación se muestra un ejemplo del proceso de registro y cancelación del registro.

Ejemplo

La siguiente función muestra cómo un código administrado EE se registra y anula el registro con Visual Studio.


```

namespace EEMC
{
    [GuidAttribute("462D4A3D-B257-4AEE-97CD-5918C7531757")]
    public class EEMCClass : IDebugExpressionEvaluator
    {
        #region Register and unregister.
        private static Guid guidMyLang = new Guid("462D4A3E-B257-4AEE-97CD-5918C7531757");
        private static string languageName = "MyC";
        private static string eeName = "MyC Expression Evaluator";

        private static Guid guidMicrosoftVendor = new Guid("994B45C4-E6E9-11D2-903F-00C04FA302A1");
        private static Guid guidCOMPlusOnlyEng = new Guid("449EC4CC-30D2-4032-9256-EE18EB41B62B");
        private static Guid guidCOMPlusNativeEng = new Guid("92EF0900-2251-11D2-B72E-0000F87572EF");

        /// <summary>
        /// Register the expression evaluator.
        /// Set "project properties/configuration properties/build/register for COM interop" to true.
        /// </summary>
        [ComRegisterFunctionAttribute]
        public static void RegisterClass(Type t)
        {
            // Get Visual Studio version (set by regpkg.exe)
            string hive = Environment.GetEnvironmentVariable("EnvSdk_RegKey");
            string s = @"SOFTWARE\Microsoft\VisualStudio\"
                + hive
                + @"\AD7Metrics\ExpressionEvaluator";

            RegistryKey rk = Registry.LocalMachine.CreateSubKey(s);
            if (rk == null) return;

            rk = rk.CreateSubKey(guidMyLang.ToString("B"));
            rk = rk.CreateSubKey(guidMicrosoftVendor.ToString("B"));
            rk.SetValue("CLSID", t.GUID.ToString("B"));
            rk.SetValue("Language", languageName);
            rk.SetValue("Name", eeName);

            rk = rk.CreateSubKey("Engine");
            rk.SetValue("0", guidCOMPlusOnlyEng.ToString("B"));
            rk.SetValue("1", guidCOMPlusNativeEng.ToString("B"));
        }
        /// <summary>
        /// Unregister the expression evaluator.
        /// </summary>
        [ComUnregisterFunctionAttribute]
        public static void UnregisterClass(Type t)
        {
            // Get Visual Studio version (set by regpkg.exe)
            string hive = Environment.GetEnvironmentVariable("EnvSdk_RegKey");
            string s = @"SOFTWARE\Microsoft\VisualStudio\"
                + hive
                + @"\AD7Metrics\ExpressionEvaluator\"
                + guidMyLang.ToString("B");
            RegistryKey key = Registry.LocalMachine.OpenSubKey(s);
            if (key != null)
            {
                key.Close();
                Registry.LocalMachine.DeleteSubKeyTree(s);
            }
        }
    }
}

```

Evaluador de expresiones de código no administrado

El archivo DLL de `DllRegisterServer` EE implementa la función para registrarse con el entorno COM, así como

NOTE

Puede encontrar el código de registro de ejemplo de código MyCEE en el archivo *dllentry.cpp*, que se encuentra en la instalación de VSIP en EnVSDK- MyCPkgs-MyCEE.

Proceso del servidor DLL

Al registrar el EE, el servidor DLL:

1. Registra su generador `CLSID` de clases según las convenciones COM normales.
2. Llama a `SetEEMetric` la función auxiliar para registrar con Visual Studio las métricas EE que se muestran en la tabla siguiente. La `SetEEMetric` función y las métricas especificadas de la siguiente manera forman parte de la biblioteca *dbgmetric.lib*. Consulte Aplicaciones auxiliares del [SDK para la depuración](#) para obtener más información.

MÉTRICA	DESCRIPCIÓN
<code>metricCLSID</code>	<code>CLSID</code> de la fábrica de clases EE
<code>metricName</code>	Nombre del EE como una cadena visible
<code>metricLanguage</code>	El nombre del idioma que el EE está diseñado para evaluar
<code>metricEngine</code>	<code>GUID</code> s de los motores de depuración (DE) que funcionan con este EE

NOTE

El `metricLanguage` `GUID` identifica el idioma por nombre, `guidLang` pero `SetEEMetric` es el argumento para que seleccione el idioma. Cuando el compilador genera el archivo de `guidLang` información de depuración, debe escribir el adecuado para que el DE sepa qué EE usar. El DE normalmente pide al `GUID` proveedor de símbolos para este idioma, que se almacena en el archivo de información de depuración.

3. Se registra en Visual Studio mediante la creación de\claves en HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudioX.Y, donde X.Y es la versión de Visual Studio con la que registrarse.

Ejemplo

La siguiente función muestra cómo un código no administrado (C++) EE se registra y anula el registro con Visual Studio.

```
/*-----
Registration
-----*/
#ifdef LREGKEY_VISUALSTUDIOROOT
    #define LREGKEY_VISUALSTUDIOROOT L"Software\\Microsoft\\VisualStudio\\8.0"
#endif

static HRESULT RegisterMetric( bool registerIt )
{
    // check where we should register
    const ULONG cchBuffer = _MAX_PATH;
    WCHAR wszRegistrationRoot[cchBuffer];
    DWORD cchFreeBuffer = cchBuffer - 1;
    wcsncpy(wszRegistrationRoot, LREGKEY_VISUALSTUDIOROOT_NOVERSION);
```

```

wscat(wszRegistrationRoot, L"\\");

// this is Environment SDK specific
// we check for EnvSdk_RegKey environment variable to
// determine where to register
DWORD cchDefRegRoot = 1strlenW(LREGKEY_VISUALSTUDIROOT_NOVERSION) + 1;
cchFreeBuffer = cchFreeBuffer - cchDefRegRoot;
DWORD cchEnvVarRead = GetEnvironmentVariableW(
    /* LPCTSTR */ L"EnvSdk_RegKey", // environment variable name
    /* LPTSTR */ &wszRegistrationRoot[cchDefRegRoot], // buffer for variable value
    /* DWORD */ cchFreeBuffer); // size of buffer
if (cchEnvVarRead >= cchFreeBuffer)
    return E_UNEXPECTED;
// If the environment variable does not exist then we must use
// LREGKEY_VISUALSTUDIROOT which has the version number.
if (0 == cchEnvVarRead)
    wcscpy(wszRegistrationRoot, LREGKEY_VISUALSTUDIROOT);

if (registerIt)
{
    SetEEMetric(guidMyLang,
        guidMicrosoftVendor,
        metricCLSID,
        CLSID_MycEE,
        wszRegistrationRoot );
    SetEEMetric(guidMyLang,
        guidMicrosoftVendor,
        metricName,
        GetString(IDS_INFO_MYCDESCRIPTION),
        wszRegistrationRoot );
    SetEEMetric(guidMyLang,
        guidMicrosoftVendor,
        metricLanguage, L"MyC",
        wszRegistrationRoot);

    GUID engineGuids[2];
    engineGuids[0] = guidCOMPlusOnlyEng;
    engineGuids[1] = guidCOMPlusNativeEng;
    SetEEMetric(guidMyLang,
        guidMicrosoftVendor,
        metricEngine,
        engineGuids,
        2,
        wszRegistrationRoot);
}
else
{
    RemoveEEMetric( guidMyLang,
        guidMicrosoftVendor,
        metricCLSID,
        wszRegistrationRoot);
    RemoveEEMetric( guidMyLang,
        guidMicrosoftVendor,
        metricName,
        wszRegistrationRoot );
    RemoveEEMetric( guidMyLang,
        guidMicrosoftVendor,
        metricLanguage,
        wszRegistrationRoot );
    RemoveEEMetric( guidMyLang,
        guidMicrosoftVendor,
        metricEngine,
        wszRegistrationRoot );
}

return S_OK;
}

```

Vea también

- [Escribir un evaluador de expresiones CLR](#)
- [Ayudantes del SDK para la depuración](#)

Implementar un evaluador de expresiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

La evaluación de una expresión es una interacción compleja entre el motor de depuración (DE), el proveedor de símbolos (SP), el objeto de enlazador y el evaluador de expresiones (EE). Estos cuatro componentes están conectados por interfaces que son implementadas por un componente y consumidas por otro.

El EE toma una expresión de la DE en forma de una cadena y la analiza o evalúa. El EE ejecuta las siguientes interfaces, que son consumidas por el DE:

- [IDebugExpressionEvaluator](#)
- [IDebugParsedExpression](#)

El EE llama al objeto de enlazador, proporcionado por el DE, para obtener el valor de símbolos y objetos. El EE consume las siguientes interfaces, que son implementadas por el DE:

- [IDebugObject](#)
- [IDebugArrayObject](#)
- [IDebugFunctionObject](#)
- [IDebugPointerObject](#)
- [IDebugManagedObject](#)
- [IEnumDebugObjects](#)
- [IDebugBinder](#)

El EE ejecuta [IDebugProperty2](#). `IDebugProperty2` proporciona el mecanismo para describir el resultado de una evaluación de expresión, como una variable local, una primitiva o un objeto a Visual Studio, que, a continuación, muestra la información adecuada en la ventana **Locales**, **Inspección Inmediato**.

El SP es dado al EE por el DE cuando pide información. El SP ejecuta interfaces que describen direcciones y campos, como las interfaces siguientes y sus derivados:

- [IDebugSymbolProvider](#)
- [IDebugAddress](#)
- [IDebugField](#)

El EE consume todas estas interfaces.

En esta sección

[Estrategia de implementación del evaluador](#) de expresiones Define un proceso de tres pasos para la estrategia de implementación del evaluador de expresiones (EE).

Vea también

- [Escribir un evaluador de expresiones CLR](#)

Estrategia de implementación del evaluador de expresiones

06/04/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Un enfoque para crear rápidamente un evaluador de expresiones (EE) es implementar primero el código mínimo necesario para mostrar variables locales en la ventana **Locales**. Es útil tener en cuenta que cada línea de la ventana **Locales** muestra el nombre, el tipo y el valor de una variable local y que los tres están representados por un objeto `IDebugProperty2`. El nombre, el tipo y el valor de `IDebugProperty2` una variable local se obtiene de un objeto mediante una llamada a su `GetPropertyInfo` método. Para obtener más información sobre cómo mostrar variables locales en la ventana **Locales**, consulte [Visualización de variables locales](#).

Discusión

Una posible secuencia de implementación comienza con la implementación de `IDebugExpressionEvaluator`. Los métodos `Parse` y `GetMethodProperty` deben implementarse para mostrar los locales. Llamar `IDebugExpressionEvaluator::GetMethodProperty` devuelve `IDebugProperty2` un objeto que representa un método: es decir, un `IDebugMethodField` objeto. Los propios métodos no se muestran en la ventana **Locales**.

El `EnumChildren` método debe implementarse a continuación. El motor de depuración (DE) llama a este método para `IDebugProperty2::EnumChildren` `guidFilter` obtener `guidFilterLocalsPlusArgs` una lista de variables y argumentos locales pasando un argumento de `IDebugProperty2::EnumChildren` llama a `EnumArguments` y `EnumLocals`, combinando los resultados en una sola enumeración. Consulte [Mostrar configuraciones regionales](#) para obtener más detalles.

Vea también

- [Implementar un evaluador de expresiones](#)
- [Mostrar locales](#)

Mostrar locales

06/04/2020 • 3 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

La ejecución siempre tiene lugar en el contexto de un método, también conocido como método contenedor o método actual. Cuando se detiene la ejecución, Visual Studio llama al motor de depuración (DE) para obtener una lista de variables y argumentos locales, denominados colectivamente los valores locales del método. Visual Studio muestra estos valores locales y sus valores en la ventana **Locales**.

Para mostrar los locales, el DE llama a la [GetMethodProperty](#) método que pertenece a la EE y le proporciona un contexto de evaluación, es decir, un proveedor de símbolos (SP), la dirección de ejecución actual y un objeto de enlazador. Para obtener más información, consulte [Contexto de evaluación](#). Si la llamada se

`IDebugExpressionEvaluator::GetMethodProperty` realiza correctamente, el método devuelve un [IDebugProperty2](#) objeto, que representa el método que contiene la dirección de ejecución actual.

El DE llama a [EnumChildren](#) para obtener un [IEnumDebugPropertyInfo2](#) objeto, que se filtra para devolver solo los valores locales y se enumera para generar una lista de [DEBUG_PROPERTY_INFO](#) estructuras. Cada estructura contiene el nombre, el tipo y el valor de un local. El tipo y el valor se almacenan como cadenas formateadas, adecuadas para su visualización. El nombre, el tipo y el valor se muestran normalmente juntos en una línea de la ventana **Locales**.

NOTE

Las ventanas **Inspección rápida** y **Inspección** también muestran variables con el mismo formato de nombre, valor y tipo. Sin embargo, esos valores se obtienen `IDebugProperty2::EnumChildren` llamando a [GetPropertyInfo](#) en lugar de .

En esta sección

[Ejemplo de implementación de locales](#) Utiliza ejemplos para recorrer paso a paso el proceso de implementación de locales.

Secciones relacionadas

[Contexto de evaluación](#) Explica que cuando el motor de depuración (DE) llama al evaluador de expresiones (EE), pasa tres argumentos.

Vea también

[Escribir un evaluador de expresiones CLR](#)

Ejemplo de implementación de locales

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

A continuación se muestra información general sobre cómo Visual Studio obtiene los locales de un método del evaluador de expresiones (EE):

1. Visual Studio llama a [GetDebugProperty](#) del motor de depuración (DE) para obtener un [Objeto IDebugProperty2](#) que representa todas las propiedades del marco de pila, incluidos los locales.
2. `IDebugStackFrame2::GetDebugProperty` llama a [GetMethodProperty](#) para obtener un objeto que describe el método dentro del cual se produjo el punto de interrupción. El DE proporciona un proveedor de símbolos ([IDebugSymbolProvider](#)), una dirección ([IDebugAddress](#)) y un enlazador ([IDebugBinder](#)).
3. `IDebugExpressionEvaluator::GetMethodProperty` llama a [GetContainerField](#) con el objeto proporcionado [IDebugAddress](#) para obtener un [IDebugContainerField](#) que representa el método que contiene la dirección especificada.
4. Se `IDebugContainerField` consulta la interfaz para el [IDebugMethodField](#) interfaz. Es esta interfaz la que da acceso a los locales del método.
5. `IDebugExpressionEvaluator::GetMethodProperty` crea una instancia `CFieldProperty` de una clase (llamada en el ejemplo) que ejecuta la `IDebugProperty2` interfaz para representar los locales del método. El `IDebugMethodField` objeto se `CFieldProperty` coloca en `IDebugSymbolProvider` este `IDebugAddress` objeto `IDebugBinder` junto con los objetos `,` y `.`
6. Cuando `CFieldProperty` se inicializa el objeto, [GetInfo](#) se llama en el `IDebugMethodField` objeto para obtener un [FIELD_INFO](#) estructura que contiene toda la información que se puede mostrar sobre el propio método.
7. `IDebugExpressionEvaluator::GetMethodProperty` devuelve `CFieldProperty` el objeto `IDebugProperty2` como un objeto.
8. Visual Studio llama a [EnumChildren](#) en el objeto devuelto `IDebugProperty2` con el filtro `guidFilterLocalsPlusArgs`, que devuelve un [IEnumDebugPropertyInfo2](#) objeto que contiene los locales del método. Esta enumeración se rellena mediante llamadas a [EnumLocals](#) y [EnumArguments](#).
9. Visual Studio llama a [Next](#) para obtener una estructura de [DEBUG_PROPERTY_INFO](#) para cada local. Esta estructura contiene un `IDebugProperty2` puntero a una interfaz para un local.
10. Visual Studio llama a [GetPropertyInfo](#) para cada local para obtener el nombre, el valor y el tipo del local. Esta información se muestra en la ventana **Locales**.

En esta sección

[Implementar GetMethodProperty](#) Describe una implementación de [GetMethodProperty](#).

[Enumerar locales](#) Describe cómo el motor de depuración (DE) realiza una llamada para enumerar variables o argumentos locales.

[Obtener propiedades locales](#) Describe cómo la DE realiza una llamada para obtener el nombre, el tipo y el valor de uno o varios locales.

[Obtener valores locales](#) Describe cómo obtener el valor de local, que requiere los servicios de un objeto de enlazador proporcionado por el contexto de evaluación.

[Evaluar a los lugareños](#) Explica cómo se evalúan los locales.

Secciones relacionadas

[Contexto de evaluación](#) Proporciona los argumentos que se pasan cuando la DE llama al evaluador de expresiones (EE).

[Muestra MyCEE](#) Muestra un enfoque de implementación para crear un evaluador de expresiones para el lenguaje MyC.

Vea también

- [Mostrando locales](#)

Implementar GetMethodProperty

06/04/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

Visual Studio llama a [GetDebugProperty](#) del motor de depuración (DE), que a su vez llama a [GetMethodProperty](#) para obtener información sobre el método actual en el marco de pila.

Esta implementación de `IDebugExpressionEvaluator::GetMethodProperty` realiza las siguientes tareas:

1. Llama a [GetContainerField](#), pasando el objeto [IDebugAddress](#). El proveedor de símbolos (SP) devuelve un [IDebugContainerField](#) que representa el método que contiene la dirección especificada.
2. Obtiene el [IDebugMethodField](#) `IDebugContainerField` del archivo.
3. Crea una instancia `CFieldProperty` de una clase (llamada en este ejemplo) `IDebugMethodField` que implementa la interfaz [IDebugProperty2](#) y contiene el objeto devuelto desde el SP.
4. Devuelve `IDebugProperty2` la interfaz del `CFieldProperty` objeto.

Código administrado

En este ejemplo `IDebugExpressionEvaluator::GetMethodProperty` se muestra una implementación de en código administrado.

```
namespace EEMC
{
    [GuidAttribute("462D4A3D-B257-4AEE-97CD-5918C7531757")]
    public class EEMCClass : IDebugExpressionEvaluator
    {
        public HRESULT GetMethodProperty(
            IDebugSymbolProvider symbolProvider,
            IDebugAddress address,
            IDebugBinder binder,
            int includeHiddenLocals,
            out IDebugProperty2 property)
        {
            IDebugContainerField containerField = null;
            IDebugMethodField methodField = null;
            property = null;

            // Get the containing method field.
            symbolProvider.GetContainerField(address, out containerField);
            methodField = (IDebugMethodField) containerField;

            // Return the property of method field.
            property = new CFieldProperty(symbolProvider, address, binder, methodField);
            return COM.S_OK;
        }
    }
}
```

Código no administrado

En este ejemplo `IDebugExpressionEvaluator::GetMethodProperty` se muestra una implementación de código no administrado.

```
[CPP]
STDMETHODIMP CExpressionEvaluator::GetMethodProperty(
    in IDebugSymbolProvider *pprovider,
    in IDebugAddress        *paddress,
    in IDebugBinder         *pbinder,
    in BOOL                 includeHiddenLocals,
    out IDebugProperty2     **pproperty
)
{
    if (pprovider == NULL)
        return E_INVALIDARG;

    if (pproperty == NULL)
        return E_INVALIDARG;
    else
        *pproperty = 0;

    HRESULT hr;
    IDebugContainerField* pcontainer = NULL;

    hr = pprovider->GetContainerField(paddress, &pcontainer);
    if (FAILED(hr))
        return hr;

    IDebugMethodField* pmethod = NULL;
    hr = pcontainer->QueryInterface( IID_IDebugMethodField,
        reinterpret_cast<void**>(&pmethod));
    pcontainer->Release();
    if (FAILED(hr))
        return hr;

    CFieldProperty* pfieldProperty = new CFieldProperty( pprovider,
                                                         paddress,
                                                         pbinder,
                                                         pmethod );

    pmethod->Release();
    if (!pfieldProperty)
        return E_OUTOFMEMORY;

    hr = pfieldProperty->Init();
    if (FAILED(hr))
    {
        pfieldProperty->Release();
        return hr;
    }

    hr = pfieldProperty->QueryInterface( IID_IDebugProperty2,
        reinterpret_cast<void**>(pproperty));
    pfieldProperty->Release();

    return hr;
}
```

Vea también

- [Ejemplo de implementación de locales](#)

Enumerar locales

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de expresiones CLR y Ejemplo de evaluador de expresiones administradas.

Cuando Visual Studio está listo para rellenar el **locales** ventana, llama a `EnumChildren` en el `IDebugProperty2` objeto devuelto desde `GetMethodProperty` (consulte [implementar GetMethodProperty](#)).

`IDebugProperty2::EnumChildren` devuelve un `IEnumDebugPropertyInfo2` objeto.

La `IDebugProperty2::EnumChildren` implementación realiza las siguientes tareas:

1. Garantiza que esto representa un método.
2. Utiliza `guidFilter` el argumento para determinar qué método llamar a la `IDebugMethodField` objeto. Si `guidFilter` es igual a:
 - a. `guidFilterLocals`, llame a `EnumLocals` para obtener un objeto `IEnumDebugFields`.
 - b. `guidFilterArgs`, llame a `EnumArguments` para obtener un `IEnumDebugFields` objeto.
 - c. `guidFilterLocalsPlusArgs`, sinteteten una `IDebugMethodField::EnumLocals` `IDebugMethodField::EnumArguments` enumeración que combine los resultados de y . Esta síntesis está `CEnumMethodField` representada por la clase .
3. Crea una instancia `CEnumPropertyInfo` de una clase (llamada `IEnumDebugPropertyInfo2` en este `IEnumDebugFields` ejemplo) que implementa la interfaz y contiene el objeto.
4. Devuelve `IEnumDebugProperty2Info2` la interfaz del `CEnumPropertyInfo` objeto.

Código administrado

En este ejemplo `IDebugProperty2::EnumChildren` se muestra una implementación de en código administrado.

```
namespace EEMC
{
    public class CFieldProperty : IDebugProperty2
    {
        public HRESULT EnumChildren (
            uint                dwFields,
            uint                radix,
            ref Guid            guidFilter,
            ulong               attribFilter,
            string              nameFilter,
            uint                timeout,
            out IEnumDebugPropertyInfo2 properties)
        {
            properties = null;
            IEnumDebugFields fields = null;

            // If this field is a method...
            if (0 != ((uint) fieldKind & (uint) FIELD_KIND.FIELD_TYPE_METHOD))
            {
                if (guidFilter == Guid.Empty)
                {
                    // If no filter, use the default filter.
                    fields = EnumLocals(dwFields, radix, 0, 0, 0, 0, 0, out properties);
                }
                else if (guidFilter == Guid.NewGuid())
                {
                    // If a filter is provided, use it.
                    fields = EnumArguments(dwFields, radix, guidFilter, 0, 0, 0, 0, out properties);
                }
                else if (guidFilter == Guid.NewGuid())
                {
                    // If a filter is provided, use it.
                    fields = EnumLocalsPlusArgs(dwFields, radix, guidFilter, 0, 0, 0, 0, out properties);
                }
            }
            else
            {
                // If this field is a variable, use the default filter.
                fields = EnumLocals(dwFields, radix, 0, 0, 0, 0, 0, out properties);
            }

            if (fields != null)
            {
                properties = new CEnumPropertyInfo(fields, this);
            }

            return S_OK;
        }
    }
}
```

```

IDebugMethodField methodField = (IDebugMethodField) field;

// Enumerate parameters.
if (guidFilter == FilterGuids.guidFilterArgs)
{
    methodField.EnumParameters(out fields);
}
// Enumerate local variables.
else if (guidFilter == FilterGuids.guidFilterLocals)
{
    methodField.EnumLocals(address, out fields);
}
// Enumerate all local variables, including invisible compiler temps.
else if (guidFilter == FilterGuids.guidFilterAllLocals)
{
    methodField.EnumAllLocals(address, out fields);
}
// Enumerate "this", if any, and all parameters and local variables.
else if (guidFilter == FilterGuids.guidFilterLocalsPlusArgs)
{
    IDebugClassField fieldThis = null;
    IEnumDebugFields parameters = null;
    IEnumDebugFields locals = null;

    methodField.GetThis(out fieldThis);
    methodField.EnumParameters(out parameters);
    methodField.EnumLocals(address, out locals);

    CEnumMethodField enumMethodField =
        new CEnumMethodField(fieldThis, parameters, locals);
    fields = (IEnumDebugFields) enumMethodField;
}
// Enumerate only "this".
else if (guidFilter == FilterGuids.guidFilterThis)
{
    IDebugClassField fieldThis = null;
    methodField.GetThis(out fieldThis);

    CEnumMethodField enumMethodField =
        new CEnumMethodField(fieldThis, null, null);
    fields = (IEnumDebugFields) enumMethodField;
}
else throw new COMException();// E_FAIL
}
// Wrap a property enumerator around the field enumerator.
CEnumPropertyInfo propertiesInfo =
    new CEnumPropertyInfo(provider, address, binder, radix, fields,
        (DEBUGPROP_INFO_FLAGS) dwFields);

properties = (IEnumDebugPropertyInfo2) propertiesInfo;
return COM.S_OK;
}
}
}

```

Código no administrado

En este ejemplo `IDebugProperty2::EnumChildren` se muestra una implementación de código no administrado.

```
STDMETHODIMP CFieldProperty::EnumChildren(
    in DEBUGPROP_INFO_FLAGS    infoFlags,
    in DWORD                    radix,
    in REFGUID                  guidFilter,
    in DBG_ATTRIB_FLAGS         attribFilter,
    in LPCOLESTR                 pszNameFilter,
    in DWORD                     timeout,
```

```

        out IEnumDebugPropertyInfo2 ** ppchildren )
{
    if (ppchildren == NULL)
        return E_INVALIDARG;
    else
        *ppchildren = 0;

    //get enumeration
    HRESULT hr;
    IEnumDebugFields* pfields = NULL;

    if (m_fieldKind & FIELD_TYPE_METHOD)
    {
        //-----
        // A Method

        IDebugMethodField* pmethod = NULL;

        //enumerate the requested properties
        hr = m_field->QueryInterface( IID_IDebugMethodField,
            reinterpret_cast<void*>(&pmethod) );
        if (FAILED(hr))
            return hr;

        if (guidFilter == guidFilterArgs)
        {
            hr = pmethod->EnumParameters( &pfields );
        }
        else if (guidFilter == guidFilterLocals)
        {
            hr = pmethod->EnumLocals( m_address, &pfields );
        }
        else if (guidFilter == guidFilterAllLocals)
        {
            hr = pmethod->EnumAllLocals( m_address, &pfields );
        }
        else if (guidFilter == guidFilterLocalsPlusArgs)
        {
            //we create a special enumerator for this
            IDebugClassField* pfieldThis = NULL;
            IEnumDebugFields* pparameters = NULL;
            IEnumDebugFields* plocals = NULL;

            pmethod->GetThis( &pfieldThis );
            pmethod->EnumParameters( &parameters );
            pmethod->EnumLocals( m_address, &plocals );

            CEnumMethodField* penumMethodField =
                new CEnumMethodField( pfieldThis, pparameters, plocals );
            if (pfieldThis != NULL)
                pfieldThis->Release();
            if (pparameters != NULL)
                pparameters->Release();
            if (plocals != NULL)
                plocals->Release();
            if (!penumMethodField)
            {
                hr = E_OUTOFMEMORY;
            }
            else
            {
                hr = penumMethodField->QueryInterface( IID_IEnumDebugFields,
                    reinterpret_cast<void*>(&pfields) );
                penumMethodField->Release();
            }
        }
        else if (guidFilter == guidFilterThis )
        {
            IDebugClassField* pfieldThis = NULL;

```

```

        IDebugClassField* pfieldThis = NULL;

        hr = pmethod->GetThis( &pfieldThis );
        if (SUCCEEDED(hr))
        {
            CEnumMethodField* penumMethodField =
                new CEnumMethodField( pfieldThis, NULL, NULL );
            pfieldThis->Release();
            if (!penumMethodField)
            {
                hr = E_OUTOFMEMORY;
            }
            else
            {
                hr = penumMethodField->QueryInterface( IID_IEnumDebugFields,
                    reinterpret_cast<void**>(&pfields) );
                penumMethodField->Release();
            }
        }
    }
    else
    {
        hr = E_FAIL;
    }

    pmethod->Release();
    if (hr != S_OK)
        return hr;
}

//create a property info enumeration around the field enumerator
CEnumPropertyInfo* pproperties =
    new CEnumPropertyInfo( m_provider, m_address, m_binder,
                          radix, pfields, infoFlags );

if (pfields != NULL)
    pfields->Release();
if (pproperties == NULL)
    return E_OUTOFMEMORY;

hr = pproperties->QueryInterface( IID_IEnumDebugPropertyInfo2,
    reinterpret_cast<void**>(ppchildren) );
pproperties->Release();

return hr;
}

```

Vea también

- [Ejemplo de implementación de locales](#)
- [Implementar GetMethodProperty](#)
- [Contexto de evaluación](#)

Obtener propiedades locales

06/04/2020 • 3 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Visual Studio llama a [EnumChildren](#) para obtener un [IEnumDebugPropertyInfo2](#) objeto que proporciona acceso a todos los locales que se mostrarán en el **locales** ventana. Visual Studio, a continuación, llama a [Next](#) para obtener la información que se mostrará para cada local. En este ejemplo, [CEnumPropertyInfo](#) la [IEnumDebugPropertyInfo2](#) clase implementa la interfaz.

Esta implementación de [IEnumDebugPropertyInfo2::Next](#) realiza las siguientes tareas:

1. Borra la matriz donde se va a almacenar la información.
2. Llama a [Next](#) para cada local, almacenando el [DEBUG_PROPERTY_INFO](#) devuelto en la matriz que se va a devolver. El [IEnumDebugFields](#) objeto se [CEnumPropertyInfo](#) proporcionó cuando se crea una instancia de esta clase.

Código administrado

En este ejemplo [IEnumDebugPropertyInfo2::EnumChildren](#) se muestra una implementación de para los locales de un método en código administrado.

```

namespace EEMC
{
    public class CEnumMethodField : IEnumDebugFields
    {
        public HRESULT Next(
            uint                count,
            DEBUG_PROPERTY_INFO[] properties,
            out uint            fetched)
        {
            if (count > properties.Length)
                throw new COMException();

            // Zero out the array.
            for (int i= 0; i < count; i++)
            {
                properties[i].bstrFullName = "";
                properties[i].bstrName = "";
                properties[i].bstrType = "";
                properties[i].bstrValue = "";
                properties[i].dwAttrib = 0;
                properties[i].dwFields = 0;
                properties[i].pProperty = null;
            }
            fetched = 0;

            // COM interop.
            HRESULT hr;
            uint innerFetched;
            IDebugField[] field = new IDebugField[1];

            while (fetched < count)
            {
                field[0] = null;
                innerFetched = 0;

                // Get next field.
                if (fetched < fieldCount)
                {
                    hr = fields.Next(1, field, ref innerFetched);
                    // No more fields.
                    else return COM.S_FALSE;

                    if (hr != COM.S_OK || innerFetched != 1 || field[0] == null)
                        throw new COMException("CEnumPropertyInfo.Next");

                    // Get property from field.
                    CFieldProperty fieldProperty =
                        new CFieldProperty(provider, address, binder, field[0]);

                    DEBUG_PROPERTY_INFO[] property =
                        new DEBUG_PROPERTY_INFO[1];
                    fieldProperty.GetPropertyInfo((uint) infoFlags, radix, 0, null, 0, property);
                    properties[fetched++] = property[0];
                }
                return COM.S_OK;
            }
        }
    }
}

```

Código no administrado

En este ejemplo `IEnumDebugPropertyInfo2::EnumChildren` se muestra una implementación de para los locales de un método en código no administrado.

```

STDMETHODIMP CEnumPropertyInfo::Next(
    in  ULONG          count,
    out DEBUG_PROPERTY_INFO* pelements,
    out ULONG*         pfetched )
{
    if (pfetched)
        *pfetched = 0;
    if (!pelements)
        return E_INVALIDARG;
    else
        memset( pelements, 0, count * sizeof(DEBUG_PROPERTY_INFO));

    HRESULT hr = S_OK;
    ULONG idx = 0;
    while (idx < count)
    {
        ULONG          fetchedFields;
        IDebugField* pfield;

        //get the next field
        hr = m_fields->Next( 1, &pfield, &fetchedFields );
        if (FAILED(hr))
            return hr;
        if (fetchedFields != 1)
            return E_FAIL;
        idx++;

        //create a CFieldProperty to retrieve the DEBUG_PROPERTY_INFO
        CFieldProperty* pproperty =
            new CFieldProperty( m_provider, m_address, m_binder, pfield );
        pfield->Release();
        if (!pproperty)
            return E_OUTOFMEMORY;

        hr = pproperty->Init();
        if (FAILED(hr))
        {
            pproperty->Release();
            return hr;
        }

        hr = pproperty->GetPropertyInfo( m_infoFlags,
                                         m_radix,
                                         0,
                                         NULL,
                                         0,
                                         pelements + idx - 1);

        pproperty->Release();
        if (FAILED(hr))
            return hr;
    }

    if (pfetched)
        *pfetched = idx;
    return hr;
}

```

Veá también

- [Ejemplo de implementación de locales](#)
- [Enumeración de locales](#)

Obtener valores locales

06/04/2020 • 8 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

Para obtener el valor de un local, Visual Studio llama a [GetPropertyInfo](#) para ese local. En esta implementación, la clase `CFieldProperty` implementa el `IDebugProperty2` interfaz para cada local.

Esta implementación de `IDebugProperty2::GetPropertyInfo` realiza las siguientes tareas:

1. Obtiene el nombre, la propiedad y los atributos del local de la estructura `FIELD_INFO` relleno cuando se crea una instancia e inicialización de la clase.
2. Obtiene el tipo local de la `IDebugField` objeto.
3. Obtiene el valor local del `IDebugField` objeto. Este campo está enlazado a la ubicación de memoria del local mediante el `IDebugBinder` objeto y el valor se obtiene de la resultante `IDebugObject` objeto.
4. Devuelve todas las propiedades solicitadas en una estructura `DEBUG_PROPERTY_INFO`.

Código administrado

En este ejemplo `IDebugProperty2::GetPropertyInfo` se muestra una implementación de para el local de un método en código administrado. También muestra una función auxiliar, `Field.GetType`, que se utiliza para obtener el tipo del campo. `Field.GetValue` se muestra en [Evaluar locales](#). La función `Field.MapModifiersToAttributes` auxiliar (no se muestra) simplemente convierte los indicadores `FIELD_MODIFIERS` de un campo en valores `DBG_ATTRIB_FLAGS`.

```
namespace EEMC
{
    public class CFieldProperty : IDebugProperty2
    {
        public HRESULT GetPropertyInfo(
            uint          dwFields,
            uint          radix,
            uint          timeout,
            IDebugReference2[] refArgs,
            uint          argCount,
            DEBUG_PROPERTY_INFO[] infoArray)
        {
            DEBUGPROP_INFO_FLAGS flags = (DEBUGPROP_INFO_FLAGS) dwFields;
            DEBUGPROP_INFO_FLAGS infoFields = DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_NONE;

            // Initialize the structure.
            DEBUG_PROPERTY_INFO info = infoArray[0];
            info.pProperty = null;

            // Fill in the full name, if requested.
            if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_FULLNAME))
            {
                info.bstrFullName = fieldInfo.bstrFullName;
                infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_FULLNAME;
            }
        }
    }
}
```

```

    }
    // Fill in the name, if requested.
    if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_NAME))
    {
        info.bstrName = fieldInfo.bstrName;
        infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_NAME;
    }
    // Fill in the type, if requested.
    if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_TYPE))
    {
        if (fieldType == null)
            fieldType = Field.GetType(field, out fieldSize);
        if (fieldType == null)
            info.bstrType = "unknown";
        else
            info.bstrType = fieldType.ToString();
        infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_TYPE;
    }
    // The property associated with this property is this property.
    if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_PROP))
    {
        info.pProperty = this;
        infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_PROP;
    }
    // Get the property value, if requested.
    if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_VALUE))
    {
        if (fieldType == null)
            fieldType = Field.GetType(field, out fieldSize);
        if (fieldType == null)
            info.bstrValue = "?";
        else
        {
            object o = Field.GetValue(binder, field, fieldType, fieldSize);
            if (o == null)
                info.bstrValue = "?";
            else
                info.bstrValue = o.ToString();
        }
        infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_VALUE;
    }
    // Get the property attributes, if requested.
    if (0 != (flags & DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_ATTRIB))
    {
        info.dwAttrib =
            (ulong) Field.MapModifiersToAttributes(
                (FIELD_MODIFIERS) fieldInfo.dwModifiers,
                fieldKind);
        infoFields |= DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_ATTRIB;
    }
    info.dwFields = (uint) infoFields;
    infoArray[0] = info;
    return COM.S_OK;
}

}

//-----

internal class Field
{
    internal static Type GetType(IDebugField field, out uint size)
    {
        size = 0;

        IDebugField fieldType = null;
        field.GetType(out fieldType);
        if (fieldType == null) return null;

        // Get field size.

```

```

        fieldType.GetSize(out size);

// Get approximate type name.
FIELD_INFO[] fieldTypeInfo = new FIELD_INFO[1];
fieldType.GetInfo((uint) FIELD_INFO_FIELDS.FIF_NAME,
    fieldTypeInfo);

// Map approximate name to type name.
switch (fieldTypeInfo[0].bstrName)
{
    case "whole":
        switch (size)
        {
            case 1: return typeof(sbyte);
            case 2: return typeof(short);
            case 4: return typeof(int);
            case 8: return typeof(long);
        }
        break;
    case "uwhole":
        switch (size)
        {
            case 1: return typeof(byte);
            case 2: return typeof(char);
            case 4: return typeof(uint);
            case 8: return typeof(ulong);
        }
        break;
    case "real":
        switch (size)
        {
            case 4: return typeof(float);
            case 8: return typeof(double);
        }
        break;
    case "bool": return typeof(bool);
    case "string": return typeof(string);
}
return null;
}
}

```

Código no administrado

En este ejemplo `IDebugProperty2::GetPropertyInfo` se muestra una implementación de para el local de un método en código no administrado. También muestra dos funciones auxiliares `FieldGetType` y `FieldGetValue` que se usan para obtener el tipo y el valor del campo, respectivamente. Las `VARIANT`s se utilizan para el valor `VARIANT` y el tipo del campo como un puede controlar una amplia variedad de tipos de valor. En esta `FieldGetValue` implementación, devuelve un `IDebugField` objeto que más `FieldGetPrimitiveValue` tarde se convierte en un valor en una llamada a (que se muestra en [Evaluar valores locales](#)).

```

STDMETHODIMP CFieldProperty::GetPropertyInfo(
    in  DEBUGPROP_INFO_FLAGS infoFlags,
    in  DWORD                 radix,
    in  DWORD                 timeout,
    in  IDebugReference2**    ppargs,
    in  DWORD                 argCount,
    out DEBUG_PROPERTY_INFO*  ppropertyInfo
)
{
    if (!ppropertyInfo)
        return E_INVALIDARG;
    else
        memset(ppropertyInfo, 0, sizeof(*ppropertyInfo));
}

```

```

if (infoFlags & DEBUGPROP_INFO_FULLNAME)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_FULLNAME;
    ppropertyInfo->bstrFullName = SysAllocString( m_fieldInfo.bstrFullName );
}

if (infoFlags & DEBUGPROP_INFO_NAME)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_NAME;
    ppropertyInfo->bstrName = SysAllocString( m_fieldInfo.bstrName );
}

if (infoFlags & DEBUGPROP_INFO_TYPE)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_TYPE;

    VARIANT type;
    if (SUCCEEDED(FieldGetType( m_field, &type )))
    {
        // Convert the variant's type to a string
        VariantTypeToString( type, &(ppropertyInfo->bstrType) );
        VariantClear(&type);
    }
    if (ppropertyInfo->bstrType == NULL)
        ppropertyInfo->bstrType = SysAllocString( GetString(IDS_MSG_UNKNOWNTYPE) );
}

if (infoFlags & DEBUGPROP_INFO_PROP)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_PROP;
    QueryInterface( IID_IDebugProperty2,
        reinterpret_cast<void*>(&(ppropertyInfo->pProperty)) );
}

if (infoFlags & DEBUGPROP_INFO_VALUE)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_VALUE;

    //only show primitive values
    VARIANT value;
    if (SUCCEEDED(FieldGetValue(m_field, &value)))
    {
        VariantValueToString( radix, m_binder, value,
            &(ppropertyInfo->bstrValue) );
        VariantClear(&value);
    }

    if (ppropertyInfo->bstrValue == NULL)
        ppropertyInfo->bstrValue = SysAllocString( GetString(IDS_MSG_UNKNOWNVALUE) );
}

if (infoFlags & DEBUGPROP_INFO_VALUE_AUTOEXPAND)
{
    // AUTOEXPAND is ignored in this example
}

if (infoFlags & DEBUGPROP_INFO_ATTRIB)
{
    ppropertyInfo->dwFields |= DEBUGPROP_INFO_ATTRIB;

    FIELD_MODIFIERS    modifiers = m_fieldInfo.dwModifiers;
    DBG_ATTRIB_FLAGS    attrib    = DBG_ATTRIB_NONE;

    //access
    if (modifiers & FIELD_MOD_ACCESS_PUBLIC)
        attrib |= DBG_ATTRIB_ACCESS_PUBLIC;
    if (modifiers & FIELD_MOD_ACCESS_PRIVATE)

```

```

        attrib |= DBG_ATTRIB_ACCESS_PRIVATE;
    if (modifiers & FIELD_MOD_ACCESS_PROTECTED)
        attrib |= DBG_ATTRIB_ACCESS_PROTECTED;
    if (modifiers & FIELD_MOD_FINAL)
        attrib |= DBG_ATTRIB_ACCESS_FINAL;

    //constant
    if (modifiers & FIELD_MOD_CONSTANT)
        attrib |= DBG_ATTRIB_VALUE_READONLY;

    //storage
    if (m_fieldKind & FIELD_SYM_GLOBAL)
        attrib |= DBG_ATTRIB_STORAGE_GLOBAL;
    if (modifiers & FIELD_MOD_STATIC)
        attrib |= DBG_ATTRIB_STORAGE_STATIC;

    //type modifier
    if (modifiers & FIELD_MOD_VIRTUAL)
        attrib |= DBG_ATTRIB_TYPE_VIRTUAL;
    if (modifiers & FIELD_MOD_CONSTANT)
        attrib |= DBG_ATTRIB_TYPE_CONSTANT;
    if (modifiers & FIELD_MOD_SYNCHRONIZED)
        attrib |= DBG_ATTRIB_TYPE_SYNCHRONIZED;
    if (modifiers & FIELD_MOD_VOLATILE)
        attrib |= DBG_ATTRIB_TYPE_VOLATILE;

    //type
    if (m_fieldKind & FIELD_TYPE_METHOD)
        attrib |= DBG_ATTRIB_METHOD;
    if (m_fieldKind & FIELD_TYPE_PROP)
        attrib |= DBG_ATTRIB_PROPERTY;
    if (m_fieldKind & FIELD_TYPE_CLASS)
        attrib |= DBG_ATTRIB_CLASS;
    if (m_fieldKind & FIELD_TYPE_INTERFACE)
        attrib |= DBG_ATTRIB_INTERFACE;
    if (m_fieldKind & FIELD_TYPE_INNERCLASS)
        attrib |= DBG_ATTRIB_INNERCLASS;
    if (m_fieldKind & FIELD_KIND_SYMBOL)
        attrib |= DBG_ATTRIB_DATA;

    //set the debug attributes
    ppropertyInfo->dwAttrib = attrib;
}

return S_OK;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Helper functions

struct PrimitiveTypeInfo
{
    LPCOLESTR  pszName;
    UINT       size;
    VARTYPE    vt;
};

PrimitiveTypeInfo primitiveTypeTable[] =
{
    { OLE("string"),      0,    VT_BSTR },
    { OLE("whole"),       1,    VT_I1  },
    { OLE("whole"),       2,    VT_I2  },
    { OLE("whole"),       4,    VT_I4  },
    { OLE("whole"),       8,    VT_I8  },
    { OLE("uwhole"),      1,    VT_UI1 },
    { OLE("uwhole"),      2,    VT_UI2 },
    { OLE("uwhole"),      4,    VT_UI4 },
    { OLE("uwhole"),      8,    VT_UI8 },
    { OLE("real"),        4,    VT_R4  },

```



```

{ OLE("real"),      4,    VT_R4 },
{ OLE("real"),      8,    VT_R8 },
{ OLE("bool"),      1,    VT_BOOL },
{ OLE("bool"),      2,    VT_BOOL },
{ OLE("bool"),      4,    VT_BOOL },

{ OLE("System.String"), 0,  VT_BSTR },
{ OLE("System.SByte"),  1,  VT_I1 },
{ OLE("System.Int16"),  2,  VT_I2 },
{ OLE("System.Int32"),  4,  VT_I4 },
{ OLE("System.Int64"),  8,  VT_I8 },
{ OLE("System.Byte"),   1,  VT_UI1 },
{ OLE("System.Char"),   1,  VT_UI2 },
{ OLE("System.UInt16"), 2,  VT_UI2 },
{ OLE("System.UInt32"), 4,  VT_UI4 },
{ OLE("System.UInt64"), 8,  VT_UI8 },
{ OLE("System.Single"), 4,  VT_R4 },
{ OLE("System.Double"), 8,  VT_R8 },
{ OLE("System.Boolean"), 1,  VT_BOOL },
{ OLE("System.Boolean"), 2,  VT_BOOL },
{ OLE("System.Boolean"), 4,  VT_BOOL },

{ NULL, 0, VT_EMPTY }
};

HRESULT FieldGetType( in IDebugField* pfield, out VARIANT* pvarType )
{
    HRESULT hr;

    if (pfield == NULL)
        return E_INVALIDARG;
    if (pvarType == NULL)
        return E_INVALIDARG;
    else
        *pvarType = 0;

    //get type size and name
    DWORD      fieldTypeSize;
    FIELD_INFO  fieldTypeInfo;
    IDebugField* pfieldType = NULL;

    hr = pfield->GetType( &pfieldType );
    if (FAILED(hr))
        return hr;

    hr = pfieldType->GetSize( &fieldTypeSize );
    if (FAILED(hr))
    {
        pfieldType->Release();
        return hr;
    }

    hr = pfieldType->GetInfo( FIF_NAME, &fieldTypeInfo );
    if (FAILED(hr))
    {
        pfieldType->Release();
        return hr;
    }

    //check for primitive types
    memset( pvarType, 0, sizeof(*pvarType) );
    VariantInit(pvarType);

    for (PrimitiveTypeInfo* pprimTypeInfo = primitiveTypeTable;
         pprimTypeInfo->pszName != NULL;
         pprimTypeInfo++)
    {
        if (pprimTypeInfo->size == fieldTypeSize &&
            (wcscmp(pprimTypeInfo->pszName,fieldTypeInfo.bstrName) == 0))
        {

```

```

        {
            pvarType->vt = pprimTypeInfo->vt;
            break;
        }
    }

    //VT_UNKNOWN is used for all other (structured) types
    if (pvarType->vt == VT_EMPTY)
    {
        pvarType->vt = VT_UNKNOWN;
        pvarType->punkVal = pfieldType;
        pvarType->punkVal->AddRef();
    }

    if (fieldTypeInfo.bstrName != NULL)
        SysFreeString(fieldTypeInfo.bstrName);
    pfieldType->Release();
    return S_OK;
}

//-----

HRESULT FieldGetValue( in IDebugField* pfield, out VARIANT* pvarValue )
{
    if (pvarValue == NULL)
        return E_INVALIDARG;
    else
        *pvarValue = 0;
    if (pfield == NULL)
        return E_INVALIDARG;

    //we delay getting the primitive value by just setting VT_UNKNOWN
    pvarValue->vt = VT_UNKNOWN;
    pvarValue->punkVal = pfield;
    pvarValue->punkVal->AddRef();

    return S_OK;
}

```

Vea también

- [Ejemplo de implementación de locales](#)
- [Obtener propiedades locales](#)
- [Contexto de evaluación](#)

Evaluar a los lugareños

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

[GetPropertyInfo](#) se llama para obtener el valor de un local, así como el nombre y el tipo de la ubicación local. Puesto que el valor de un local depende del estado actual del programa, el valor local debe obtenerse de la memoria. El [IDebugBinder](#) objeto se utiliza para enlazar el [IDebugField](#) objeto que representa el local a la ubicación adecuada en la memoria que contiene el valor. Esta ubicación en la memoria se representa mediante un [IDebugObject](#) objeto.

Esta funcionalidad de recuperar el valor de un local se encapsula en una función auxiliar que realiza las siguientes tareas:

1. Enlaza el `IDebugField` objeto a la `IDebugObject` memoria para obtener un objeto.
2. Obtiene el valor de la memoria. Este valor se representa como una serie de bytes.
3. Da formato al valor en función del tipo local.
4. Devuelve un objeto genérico que contiene el valor local. En C-, esto `object` es un , y en `VARIANT` C++ , esto es un .

Código administrado

Se trata de una implementación de una función que recupera el valor de un local en código administrado.

```

namespace EEMC
{
    internal class Field
    {
        internal static object GetValue(
            IDebugBinder binder,
            IDebugField field,
            Type t,
            uint size)
        {
            if (t == null || size == 0) return null;

            IDebugObject debugObject = null;
            binder.Bind(null, field, out debugObject);

            byte[] buffer = new byte[size];
            for (int i = 0; i < size; i++) buffer[i] = 0;

            debugObject.GetValue(buffer, size);

            if (t == typeof(sbyte)) return (sbyte) buffer[0];
            if (t == typeof(short)) return BitConverter.ToInt16(buffer, 0);
            if (t == typeof(int)) return BitConverter.ToInt32(buffer, 0);
            if (t == typeof(long)) return BitConverter.ToInt64(buffer, 0);
            if (t == typeof(byte)) return buffer[0];
            if (t == typeof(char)) return BitConverter.ToChar(buffer, 0);
            if (t == typeof(uint)) return BitConverter.ToUInt32(buffer, 0);
            if (t == typeof(ulong)) return BitConverter.ToUInt64(buffer, 0);
            if (t == typeof(float)) return BitConverter.ToSingle(buffer, 0);
            if (t == typeof(double)) return BitConverter.ToDouble(buffer, 0);
            if (t == typeof(bool)) return BitConverter.ToBoolean(buffer, 0);
            if (t == typeof(string)) return BitConverter.ToString(buffer, 0);
            return null;
        }
    }
}

```

Código no administrado

Se trata de una implementación de una función que recupera el valor de un local en código no administrado.

`FieldGetType` se muestra en Obtención de [valores locales](#).

```

HRESULT FieldGetPrimitiveValue(
    in IDebugBinder* pbinder,
    in IDebugField* pfield,
    out VARIANT* pvarValue
)
{
    if (pvarValue == NULL)
        return E_INVALIDARG;
    else
        *pvarValue = 0;

    if (pfield == NULL)
        return E_INVALIDARG;

    if (pbinder == NULL)
        return E_INVALIDARG;

    HRESULT hr;
    UINT valueSize = 0;
    BYTE* pvalueBits = NULL;
    IDebugObject* pobject = NULL;

    //get the value as bits

```

```

//get the value as bits
hr = pbinder->Bind( NULL, pfield, &pobject );
if (FAILED(hr))
    return hr;

hr = pobject->GetSize( &valueSize );
if (FAILED(hr))
{
    pobject->Release();
    return hr;
}

pvalueBits = reinterpret_cast<BYTE *>(malloc(valueSize * sizeof(BYTE)));
if (!pvalueBits)
{
    pobject->Release();
    return E_OUTOFMEMORY;
}

hr = pobject->GetValue( pvalueBits, valueSize );
pobject->Release();
if (FAILED(hr))
{
    free(pvalueBits);
    return hr;
}

//get the type
VARIANT    valueType;

hr = FieldGetType( pfield, &valueType );
if (FAILED(hr))
{
    free(pvalueBits);
    return hr;
}

//copy a primitive value
switch (valueType.vt)
{
case VT_BSTR:
    {
        pvarValue->vt = VT_BSTR;
        if (valueSize == 0)
            pvarValue->bstrVal = SysAllocString( OLE("") );
        else
            pvarValue->bstrVal =
                SysAllocStringByteLen( reinterpret_cast<char*>(pvalueBits),
                                        valueSize );
    }

case VT_BOOL:
case VT_I1:
case VT_I2:
case VT_I4:
case VT_I8:
case VT_UI1:
case VT_UI2:
case VT_UI4:
case VT_UI8:
case VT_R4:
case VT_R8:
    pvarValue->vt = valueType.vt;

    if (valueSize > 8)
        valueSize = 8;
    memcpy( &(pvarValue->iVal), pvalueBits, valueSize );
    break;
}

```

```
case VI_VOID:
case VT_EMPTY:
    pvarValue->vt = valueType.vt;
    break;

default:
    //not a primitive type
    VariantClear(&valueType);
    free(pvalueBits);
    return E_FAIL;
}

free(pvalueBits);
VariantClear(&valueType);
return S_OK;
}
```

Vea también

- [Ejemplo de implementación de locales](#)
- [Obtener valores locales](#)
- [Contexto de evaluación](#)

Evaluar una expresión de ventana de inspección

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

Cuando se detiene la ejecución, Visual Studio llama al motor de depuración (DE) para determinar el valor actual de cada expresión en su lista de inspección. La DE evalúa cada expresión mediante un evaluador de expresiones (EE) y Visual Studio muestra su valor en la ventana **Inspección**.

A continuación se muestra una descripción general de cómo se evalúa una expresión de lista de inspección:

1. Visual Studio llama a [GetExpressionContext](#) de la DE para obtener un contexto de expresión que se puede usar para evaluar expresiones.
2. Para cada expresión de la lista de inspección, Visual Studio llama a [ParseText](#) para convertir el texto de la expresión en una expresión analizada.
3. `IDebugExpressionContext2::ParseText` llama a [Parse](#) para realizar el trabajo real de analizar el texto y generar un [IDebugParsedExpression](#) objeto.
4. `IDebugExpressionContext2::ParseText` crea un [objeto IDebugExpression2](#) `IDebugParsedExpression` y coloca el objeto en él. Este `IDebugExpression2` objeto, a continuación, se devuelve a Visual Studio.
5. Visual Studio llama a [EvaluateSync](#) para evaluar la expresión analizada.
6. `IDebugExpression2::EvaluateSync` pasa la llamada a [EvaluateSync](#) para realizar la evaluación real y generar un [IDebugProperty2](#) objeto que se devuelve a Visual Studio.
7. Visual Studio llama a [GetPropertyInfo](#) para obtener el valor de la expresión que, a continuación, se muestra en la lista de inspección.

Analizar y luego evaluar

Dado que el análisis de una expresión compleja puede tardar mucho más que evaluarla, el proceso de evaluación de una expresión se divide en dos pasos: 1) analizar la expresión y 2) evaluar la expresión analizada. De esta manera, la evaluación puede producirse muchas veces, pero la expresión debe analizarse una sola vez. La expresión analizada intermedia se devuelve desde el EE en un [IDebugParsedExpression](#) objeto que a su vez se encapsula y se devuelve desde el DE como un [IDebugExpression2](#) objeto. El `IDebugExpression` objeto aplaza toda `IDebugParsedExpression` la evaluación al objeto.

NOTE

No es necesario que un EE se adhiera a este proceso de dos pasos aunque Visual Studio asume esto; EE puede analizar y evaluar en el mismo paso cuando se llama a [EvaluateSync](#) (así es como funciona el ejemplo MyCEE, por ejemplo). Si su lenguaje puede formar expresiones complejas, es posible que desee separar el paso de análisis del paso de evaluación. Esto puede aumentar el rendimiento en el depurador de Visual Studio cuando se muestran muchas expresiones de inspección.

En esta sección

[Ejemplo de implementación de la evaluación de expresiones](#) Utiliza el ejemplo MyCEE para recorrer paso a paso el proceso de evaluación de expresiones.

[Evaluación de una expresión de reloj](#) Explica lo que sucede después de un análisis de expresión correcto.

Secciones relacionadas

[Contexto de evaluación](#) Proporciona los argumentos que se pasan cuando el motor de depuración (DE) llama al evaluador de expresiones (EE).

Vea también

[Escribir un evaluador de expresiones CLR](#)

Ejemplo de implementación de la evaluación de expresiones

06/04/2020 • 3 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Para una expresión de ventana **De inspección**, Visual Studio llama a [ParseText](#) para generar un [IDebugExpression2](#) objeto. `IDebugExpressionContext2::ParseText` crea una instancia de un evaluador de expresiones (EE) y llama a [Parse](#) para obtener un [IDebugParsedExpression](#) objeto.

Realiza `IDebugExpressionEvaluator::Parse` las siguientes tareas:

1. [Sólo C++] Analiza la expresión para buscar errores.
2. Crea una instancia `CParsedExpression` de una clase (llamada en este ejemplo) que ejecuta la `IDebugParsedExpression` interfaz y almacena en la clase la expresión que se va a analizar.
3. Devuelve `IDebugParsedExpression` la interfaz del `CParsedExpression` objeto.

NOTE

En los ejemplos siguientes y en el ejemplo MyCEE, el evaluador de expresiones no separa el análisis de la evaluación.

Código administrado

El código siguiente muestra `IDebugExpressionEvaluator::Parse` una implementación de en código administrado. Esta versión del método aplaza el análisis a [EvaluateSync](#), ya que el código para analizar también se evalúa al mismo tiempo (consulte [Evaluar una expresión de inspección](#)).

```

namespace EEMC
{
    public class CParsedExpression : IDebugParsedExpression
    {
        public HRESULT Parse(
            string          expression,
            uint            parseFlags,
            uint            radix,
            out string      errorMessage,
            out uint        errorPosition,
            out IDebugParsedExpression parsedExpression)
        {
            errorMessage = "";
            errorPosition = 0;

            parsedExpression =
                new CParsedExpression(parseFlags, radix, expression);
            return COM.S_OK;
        }
    }
}

```

Código no administrado

El código siguiente es `IDebugExpressionEvaluator::Parse` una implementación de código no administrado. Este método llama a `Parse` una función auxiliar, para analizar la expresión y comprobar si hay errores, pero este método omite el valor resultante. La evaluación formal se aplaza a [EvaluateSync](#), donde se analiza la expresión mientras se evalúa (consulte [Evaluar una expresión de inspección](#)).

```

STDMETHODIMP CExpressionEvaluator::Parse(
    in     LPCOLESTR      pszExpression,
    in     PARSEFLAGS     flags,
    in     UINT           radix,
    out    BSTR           *pbstrErrorMessages,
    inout  UINT           *perrorCount,
    out    IDebugParsedExpression **pparsedExpression
)
{
    if (pbstrErrorMessages == NULL)
        return E_INVALIDARG;
    else
        *pbstrErrorMessages = 0;

    if (pparsedExpression == NULL)
        return E_INVALIDARG;
    else
        *pparsedExpression = 0;

    if (perrorCount == NULL)
        return E_INVALIDARG;

    HRESULT hr;
    // Look for errors in the expression but ignore results
    hr = ::Parse( pszExpression, pbstrErrorMessages );
    if (hr != S_OK)
        return hr;

    CParsedExpression* pparsedExpr = new CParsedExpression( radix, flags, pszExpression );
    if (!pparsedExpr)
        return E_OUTOFMEMORY;

    hr = pparsedExpr->QueryInterface( IID_IDebugParsedExpression,
                                     reinterpret_cast<void*>(&pparsedExpression) );
    pparsedExpr->Release();

    return hr;
}

```

Vea también

- [Evaluar una expresión de ventana de inspección](#)
- [Evaluar una expresión Watch](#)

Evaluar una expresión de reloj

06/04/2020 • 4 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Cuando Visual Studio está listo para mostrar el valor de una expresión de inspección, llama a [EvaluateSync](#), que a su vez llama a [EvaluateSync](#). Este proceso produce un [IDebugProperty2](#) objeto que contiene el valor y el tipo de la expresión.

En esta `IDebugParsedExpression::EvaluateSync` implementación de , la expresión se analiza y evalúa al mismo tiempo. Esta implementación realiza las siguientes tareas:

1. Analiza y evalúa la expresión para generar un objeto genérico que contiene el valor y su tipo. En C-, esto se `object` representa como un `while` en C++, esto se representa como un `VARIANT` .
2. Crea una instancia `CValueProperty` de una clase (llamada `IDebugProperty2` en este ejemplo) que implementa la interfaz y almacena en la clase el valor que se va a devolver.
3. Devuelve `IDebugProperty2` la interfaz del `CValueProperty` objeto.

Código administrado

Se trata de `IDebugParsedExpression::EvaluateSync` una implementación del código administrado en. El método `Tokenize` auxiliar analiza la expresión en un árbol de análisis. La función `EvalToken` auxiliar convierte el token en un valor. La función `FindTerm` auxiliar atraviesa recursivamente el árbol de análisis, llamando a `EvalToken` cada nodo que representa un valor y aplicando cualquier operación (suma o resta) en la expresión.

```

namespace EEMC
{
    public class CParsedExpression : IDebugParsedExpression
    {
        public HRESULT EvaluateSync(
            uint evalFlags,
            uint timeout,
            IDebugSymbolProvider provider,
            IDebugAddress address,
            IDebugBinder binder,
            string resultType,
            out IDebugProperty2 result)
        {
            HRESULT retval = COM.S_OK;
            this.evalFlags = evalFlags;
            this.timeout = timeout;
            this.provider = provider;
            this.address = address;
            this.binder = binder;
            this.resultType = resultType;

            try
            {
                IDebugField field = null;
                // Tokenize, then parse.
                tokens = Tokenize(expression);
                result = new CValueProperty(
                    expression,
                    (int) FindTerm(EvalToken(tokens[0], out field),1),
                    field,
                    binder);
            }
            catch (ParseException)
            {
                result = new CValueProperty(expression, "Huh?");
                retval = COM.E_INVALIDARG;
            }
            return retval;
        }
    }
}

```

Código no administrado

Se trata de `IDebugParsedExpression::EvaluateSync` una implementación del código no administrado en. La función `Evaluate` auxiliar analiza y evalúa la `VARIANT` expresión, devolviendo un holding el valor resultante. La función `VariantValueToProperty` auxiliar `VARIANT` agrupa `CValueProperty` el en un objeto.

```

STDMETHODIMP CParsedExpression::EvaluateSync(
    in  DWORD                evalFlags,
    in  DWORD                dwTimeout,
    in  IDebugSymbolProvider* pprovider,
    in  IDebugAddress*       paddress,
    in  IDebugBinder*        pbinder,
    in  BSTR                 bstrResultType,
    out IDebugProperty2**    ppproperty )
{
    // dwTimeout parameter is ignored in this implementation.
    if (pprovider == NULL)
        return E_INVALIDARG;

    if (paddress == NULL)
        return E_INVALIDARG;

```

```

    if (pbinder == NULL)
        return E_INVALIDARG;

    if (pproperty == NULL)
        return E_INVALIDARG;
    else
        *pproperty = 0;

    HRESULT hr;
    VARIANT value;
    BSTR bstrErrorMessage = NULL;
    hr = ::Evaluate( pprovider,
                    paddress,
                    pbinder,
                    m_expr,
                    &bstrErrorMessage,
                    &value );

    if (hr != S_OK)
    {
        if (bstrErrorMessage == NULL)
            return hr;

        //we can display better messages ourselves.
        HRESULT hrLocal = S_OK;
        VARIANT varType;
        VARIANT varErrorMessage;

        VariantInit( &varType );
        VariantInit( &varErrorMessage );
        varErrorMessage.vt = VT_BSTR;
        varErrorMessage.bstrVal = bstrErrorMessage;

        CValueProperty* valueProperty = new CValueProperty();
        if (valueProperty != NULL)
        {
            hrLocal = valueProperty->Init(m_expr, varType, varErrorMessage);
            if (SUCCEEDED(hrLocal))
            {
                hrLocal = valueProperty->QueryInterface( IID_IDebugProperty2,
                    reinterpret_cast<void*>(pproperty) );
            }
        }

        VariantClear(&varType);
        VariantClear(&varErrorMessage); //frees BSTR
        if (!valueProperty)
            return hr;
        valueProperty->Release();
        if (FAILED(hrLocal))
            return hr;
    }
    else
    {
        if (bstrErrorMessage != NULL)
            SysFreeString(bstrErrorMessage);

        hr = VariantValueToProperty( pprovider,
                                    paddress,
                                    pbinder,
                                    m_radix,
                                    m_expr,
                                    value,
                                    pproperty );

        VariantClear(&value);
        if (FAILED(hr))
            return hr;
    }

    return S_OK;

```

```
}
```

Vea también

- [Evaluar una expresión de ventana de inspección](#)
- [Ejemplo de implementación de la evaluación de expresiones](#)

Cambiar el valor de un local

06/04/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, consulte [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

Cuando se escribe un nuevo valor en el campo de valor de la ventana **Locales**, el paquete de depuración pasa la cadena, tal como está escrita, al evaluador de expresiones (EE). El EE evalúa esta cadena, que puede contener un valor simple o una expresión, y almacena el valor resultante en el local asociado.

Esta es una visión general del proceso de cambio del valor de un local:

1. Después de que el usuario escribe el nuevo valor, Visual Studio llama a [SetValueAsString](#) en el [IDebugProperty2](#) objeto asociado a la local.
2. `IDebugProperty2::SetValueAsString` realiza las siguientes tareas:
 - a. Evalúa la cadena para generar un valor.
 - b. Enlaza el objeto [IDebugField](#) asociado para obtener un objeto [IDebugObject](#).
 - c. Convierte el valor en una serie de bytes.
 - d. Llama a [SetValue](#) para colocar los bytes del valor en la memoria para que el programa que se está depurando pueda tener acceso a ellos.
3. Visual Studio actualiza la presentación **Locales** (consulte [Mostrar configuraciones regionales](#) para obtener más información).

Este procedimiento también se utiliza para cambiar el valor de una variable en la ventana **Inspección**, excepto que es el `IDebugProperty2` objeto asociado con el valor del local que se utiliza en lugar del `IDebugProperty2` objeto asociado al propio local.

En esta sección

[Ejemplo de implementación de valores cambiantes](#) Utiliza el ejemplo MyCEE para recorrer paso a paso el proceso de cambio de valores.

Vea también

- [Escribir un evaluador de expresiones CLR](#)
- [Mostrando locales](#)

06/04/2020 • 6 minutes to read • [Edit Online](#)

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea [Evaluadores de expresiones CLR](#) y [Ejemplo de evaluador de expresiones administradas](#).

`IDebugProperty2` objeto contiene el nombre, el valor y el tipo del local. Cuando un usuario cambia el valor de un local, Visual Studio llama a `SetValueAsString` para actualizar el valor de la local en memoria. En este ejemplo, el local `CFieldProperty` se representa mediante `IDebugProperty2` la clase que implementa la interfaz.

Para las expresiones **Watch** y **QuickWatch**, el `CValueProperty` valor que se va a cambiar se representa mediante la clase en el ejemplo MyCEE. Sin embargo, `IDebugProperty2::SetValueAsString` la implementación de es la misma que se muestra aquí.

La implementación de `IDebugProperty2::SetValueAsString` realiza las siguientes tareas:

1. Evalúa la expresión para generar un valor.
2. Enlaza el asociado `IDebugField` objeto a su ubicación de memoria y produce un `IDebugObject` objeto.
3. Convierte el valor en una serie de bytes.
4. Llama a `SetValue` para almacenar los bytes en memoria.

El código siguiente es `IDebugProperty2::SetValueAsString` una implementación de en código administrado.

```
namespace EEMC
{
    public class CFieldProperty : IDebugProperty2
    {
        public HRESULT SetValueAsString(
            string pszValue,
            uint dwRadix,
            uint dwTimeout)
        {
            HRESULT hr = COM.E_NOTIMPL;
            uint flags = (uint)enum_PARSEFLAGS.PARSE_EXPRESSION;
            CParsedExpression parsedExpression =
                new CParsedExpression(flags, dwRadix, pszValue);
            IDebugProperty2 value;
            parsedExpression.EvaluateSync(flags,
                                           dwTimeout,
                                           null,
                                           null,
                                           null,
                                           "string",
                                           out value);
```

```

        if (value != null)
        {
            DEBUG_PROPERTY_INFO[] dpi = new DEBUG_PROPERTY_INFO[1];
            hr = value.GetPropertyInfo(
                (uint)enum_DEBUGPROP_INFO_FLAGS.DEBUGPROP_INFO_VALUE,
                dwRadix,
                dwTimeout,
                null,
                0,
                dpi);
            if (hr == COM.S_OK)
            {
                hr = Field.SetValue(binder,
                                    field,
                                    dpi[0].bstrValue);
            }
        }
        return hr;
    }
}

```

//-----

```

internal class Field
{
    internal static HRESULT SetValue(
        IDebugBinder binder,
        IDebugField field,
        string bstrValue)
    {
        HRESULT hr = COM.E_FAIL;
        uint fieldSize = 0;
        Type fieldType = GetType(field, out fieldSize);
        if (fieldType != null)
        {
            FIELD_INFO[] fi = new FIELD_INFO[1];
            hr = field.GetInfo((uint)enum_FIELD_INFO_FIELDS.FIF_MODIFIERS, fi);
            if (hr != COM.S_OK ||
                (fi[0].dwModifiers & (uint)enum_FIELD_MODIFIERS.FIELD_MOD_CONSTANT) != 0)
            {
                // Couldn't get field info or field is constant and can't be changed.
                return COM.E_FAIL;
            }
            IDebugObject valueObject;
            IntPtr pBuffer = new IntPtr();
            int bufferSize = 0;
            binder.Bind(null, field, out valueObject);
            if (valueObject != null)
            {
                if (fieldType == typeof(sbyte))
                {
                    sbyte value = Convert.ToSByte(bstrValue);
                    bufferSize = 1;
                    pBuffer = Marshal.AllocCoTaskMem(bufferSize);
                    Marshal.StructureToPtr(value, pBuffer, false);
                }

                if (fieldType == typeof(short))
                {
                    System.Int16 value = Convert.ToInt16(bstrValue);
                    bufferSize = 2;
                    pBuffer = Marshal.AllocCoTaskMem(bufferSize);
                    Marshal.StructureToPtr(value, pBuffer, false);
                }

                if (fieldType == typeof(int))
                {
                    System.Int32 value = Convert.ToInt32(bstrValue);
                    bufferSize = 4;
                }
            }
        }
    }
}

```

```

        bufferSize = 4;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(long))
    {
        System.Int64 value = Convert.ToInt64(bstrValue);
        bufferSize = 8;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(byte))
    {
        byte value = Convert.ToByte(bstrValue);
        bufferSize = 1;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(char))
    {
        char value = Convert.ToChar(bstrValue);
        bufferSize = 1;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(bool))
    {
        bool value = Convert.ToBoolean(bstrValue);
        bufferSize = 1;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(uint))
    {
        System.UInt32 value = Convert.ToUInt32(bstrValue);
        bufferSize = 4;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(ulong))
    {
        System.UInt64 value = Convert.ToUInt64(bstrValue);
        bufferSize = 8;
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(float))
    {
        float value = Convert.ToSingle(bstrValue);
        bufferSize = sizeof(float);
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

    if (fieldType == typeof(double))
    {
        double value = Convert.ToDouble(bstrValue);
        bufferSize = sizeof(double);
        pBuffer = Marshal.AllocCoTaskMem(bufferSize);
        Marshal.StructureToPtr(value, pBuffer, false);
    }

```

```

        if (fieldType == typeof(string))
        {
            bufferSize = bstrValue.Length;
            pBuffer = Marshal.StringToCoTaskMemAuto(bstrValue);
        }
        if (bufferSize != 0)
        {
            byte[] byteBuffer = new byte[bufferSize];
            for (int i = 0; i < bufferSize; i++)
            {
                byteBuffer[i] = Marshal.ReadByte(pBuffer, i);
            }
            Marshal.FreeCoTaskMem(pBuffer);
            hr = valueObject.SetValue(byteBuffer, (uint)bufferSize);
        }
    }
}
return hr;
}
}
}
}

```

Código no administrado

El código siguiente es `IDebugProperty2::SetValueAsString` una implementación de en código administrado. La función `FieldCoerceValueType` auxiliar (no `VARIANT` se muestra) fuerza a a ser un `FieldSetValue` tipo específico y se asegura de que el valor es uno de los tipos que puede controlar.

```

STDMETHODIMP CFieldProperty::SetValueAsString(
    in LPCOLESTR pszValueStr,
    in DWORD      radix,
    in DWORD      timeout
)
{
    HRESULT hr;

    //evaluate the value
    VARIANT value;
    hr = Evaluate( m_provider,
                  m_address,
                  m_binder,
                  pszValueStr,
                  NULL,
                  &value );
    if (FAILED(hr))
        return hr;
    if (hr == S_FALSE)
        return E_FAIL;    //parse failed

    //copy the bits
    hr = FieldSetValue( m_binder, m_field, value );

    return hr;
}

//-----

HRESULT FieldSetValue(
    in IDebugBinder* pbinder,
    in IDebugField* pfield,
    in VARIANT&      rawValue )
{
    if (pfield == NULL)
        return E_INVALIDARG;

    if (pbinder == NULL)

```

```

if (pobject == NULL)
    return E_INVALIDARG;

HRESULT      hr      = S_OK;
IDebugObject* pobject = NULL;
VARIANT      value;
VariantInit(&value);

//check the type
hr = FieldCoerceValueType( pbinder, pfield, rawValue, &value );
if (FAILED(hr))
    goto fail;
if (hr == S_FALSE)
{
    VariantClear(value);
    return E_FAIL;
}

//get the object
hr = pbinder->Bind( NULL, pfield, &pobject );
if (FAILED(hr))
{
    pobject->Release();
    VariantClear(value);
    return hr;
}

//set the value
switch (value.vt)
{
    case VT_BSTR:
    {
        if (value.bstrVal == NULL)
        {
            LPOLESTR pszEmptyStr = OLE("");
            hr = pobject->SetValue( reinterpret_cast<BYTE*>(pszEmptyStr),
                                   sizeof(OLECHAR) );
        }
        else
            hr = pobject->SetValue( reinterpret_cast<BYTE*>(value.bstrVal),
                                   (SysStringLen(value.bstrVal)+1) * sizeof(wchar_t));
    }

    case VT_BOOL:
    case VT_I1:
    case VT_UI1:
        hr = pobject->SetValue( reinterpret_cast<BYTE*>(&(value.byref)), 1 );
        break;

    case VT_I2:
    case VT_UI2:
        hr = pobject->SetValue( reinterpret_cast<BYTE*>(&(value.byref)), 2 );
        break;

    case VT_I4:
    case VT_UI4:
    case VT_R4:
        hr = pobject->SetValue( reinterpret_cast<BYTE*>(&(value.byref)), 4 );
        break;

    case VT_I8:
    case VT_UI8:
    case VT_R8:
        hr = pobject->SetValue( reinterpret_cast<BYTE*>(&(value.byref)), 8 );
        break;

    case VT_VOID:
    case VT_EMPTY:
        hr = E_FAIL;
        break;
}

```

```

break;

case VT_UNKNOWN:
{
    //this is also a field (structured type)
    if (value.punkVal == NULL)
    {
        pobject->Release();
        VariantClear(value);
        return E_FAIL;
    };

    IDebugField* valueField;
    hr = value.punkVal->QueryInterface( IID_IDebugField,
        reinterpret_cast<void*>(&valueField) );
    if (FAILED(hr))
    {
        pobject->Release();
        VariantClear(value);
        return hr;
    };

    //for MyC we simply copy the bits
    IDebugObject* valueObject;
    hr = pbinder->Bind( NULL, valueField, &valueObject );
    valueField->Release();
    if (FAILED(hr))
    {
        pobject->Release();
        VariantClear(value);
        return hr;
    };

    BYTE* pvalueBits;
    UINT valueSize;
    hr = valueObject->GetSize( &valueSize );
    if (FAILED(hr))
    {
        valueObject->Release();
        pobject->Release();
        VariantClear(value);
        return hr;
    };

    pvalueBits = NALLOC(BYTE,valueSize+1);
    if (!pvalueBits)
    {
        valueObject->Release();
        pobject->Release();
        VariantClear(value);
        return E_OUTOFMEMORY;
    };

    hr = valueObject->GetValue( pvalueBits, valueSize );
    valueObject->Release();
    if (FAILED(hr))
    {
        free(pvalueBits);
        pobject->Release();
        VariantClear(value);
        return hr;
    }

    hr = pobject->SetValue( pvalueBits, valueSize );
    free(pvalueBits);
    if (FAILED(hr))
    {
        pobject->Release();
        VariantClear(value);

```

```
        return hr;
    }

    break;
}

default:
    //not a primitive type
    hr = E_FAIL;
    break;
}

VariantClear( &value );
pobject->Release();
return hr;
}
```

Vea también

- [Cambiar el valor de un local](#)
- [Contexto de evaluación](#)

Implementar visualizadores de tipos y visores personalizados

06/04/2020 • 3 minutes to read • [Edit Online](#)

IMPORTANT

En Visual Studio 2015, esta forma de implementar evaluadores de expresiones está en desuso. Para obtener información sobre la implementación de evaluadores de expresiones CLR, vea Evaluadores de [expresiones CLR](#) y Ejemplo de evaluador de [expresiones administradas](#).

Los visualizadores de tipos y los visores personalizados permiten a un usuario ver datos de un tipo determinado de una manera más significativa que un simple volcado hexadecimal de números. Un evaluador de expresiones (EE) puede asociar visores personalizados con tipos específicos de datos o variables. Estos visores personalizados son implementados por EE. El EE también puede admitir visualizadores de tipo externo, que podrían provenir de otro proveedor externo o incluso del usuario final.

Discusión

Visualizadores de tipo

Visual Studio solicita una lista de visualizadores de tipos y visores personalizados para cada objeto que se mostrará en una ventana de inspección. Un evaluador de expresiones (EE) proporciona dicha lista para cada tipo para el que desea admitir visualizadores de tipos y visores personalizados. Las llamadas a [GetCustomViewerCount](#) y [GetCustomViewerList](#) inician todo el proceso de acceso a visualizadores de tipos y visores personalizados (consulte [Visualización y visualización](#) de datos para obtener más información sobre la secuencia de llamada).

Visores personalizados

Los visores personalizados se implementan en el EE para un tipo de datos específico y se representan mediante el [IDebugCustomViewer](#) interfaz. Un visor personalizado no es tan flexible como un visualizador de tipos, ya que solo está disponible cuando se ejecuta el EE que implementa ese visor personalizado determinado. Implementar un visor personalizado es más sencillo que implementar la compatibilidad con visualizadores de tipos. Sin embargo, la compatibilidad con visualizadores de tipos proporciona la máxima flexibilidad al usuario final para visualizar sus datos. El resto de esta discusión se refiere únicamente a los visualizadores de tipos.

Interfaces

EE implementa las siguientes interfaces para admitir visualizadores de tipos, que visualde Visual Studio:

- [IEEVisualizerDataProvider](#)
- [IPropertyProxyEESide](#)
- [IPropertyProxyProvider](#)
- [IEEDataStorage](#)
- [IDebugProperty3](#)
- [IDebugObject](#)

EE consume las siguientes interfaces para admitir visualizadores de tipos:

- [IEEVisualizerService](#)
- [IEEVisualizerServiceProvider](#)
- [IDebugBinder3](#)

Vea también

- [Escribir un evaluador de expresiones CLR](#)
- [Visualización y visualización de datos](#)
- [IDebugCustomViewer](#)

Implementar un proveedor portuario

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un proveedor de puertos suministra puertos a petición al administrador de depuración de sesión (SDM). Se debe implementar un proveedor de puertos al depurar en una máquina que no sea DCOM o cuando un nuevo dispositivo requiere soporte técnico. Por ejemplo, para proporcionar depuración a un teléfono celular, puede configurar un proveedor de puertos que proporcione puertos, que se conectan al teléfono celular (quizás mediante IR o una conexión de celda) y enumera los procesos y programas que se ejecutan en el teléfono.

Para depurar programas en equipos basados en Windows (incluida la depuración remota), Visual Studio proporciona proveedores de puertos para procesos nativos y DeCommon Language Runtime (CLR), por lo que no es necesario configurar su propio proveedor de puertos en esos casos.

En esta sección

[Implementar y registrar un proveedor portuario](#) Describe cómo el SDM interactúa con el proveedor de puertos y sus puertos.

[Interfaces de proveedores de puertos requeridas](#) Documenta las interfaces que debe implementar para obtener un proveedor de puertos.

Secciones relacionadas

[Conceptos del depurador](#) Describe los principales conceptos arquitectónicos de depuración.

Vea también

[Extensibilidad del depurador de Visual Studio](#)

Implementar y registrar un proveedor portuario

06/04/2020 • 3 minutes to read • [Edit Online](#)

La función de un proveedor de puertos es realizar un seguimiento y suministrar puertos, que a su vez gestionan los procesos. Cuando es necesario crear un puerto, se crea una instancia del proveedor de puertos mediante `CoCreate` con el GUID del proveedor de puertos (el administrador de depuración de sesión [SDM] usará el proveedor de puertos seleccionado por el usuario o el proveedor de puerto especificado por el sistema de proyecto). El SDM entonces llama a `CanAddPort` para ver si se puede agregar algún puerto. Si se puede agregar un puerto, se solicita un nuevo puerto llamando a `AddPort` y pasándole un `IDebugPortRequest2` que describe el puerto. `AddPort` devuelve un nuevo puerto representado por un `IDebugPort2` interfaz.

Discusión

Un puerto es creado por un proveedor de puertos, que está asociado a una máquina o servidor de depuración. Un servidor enumera sus proveedores de puertos a través de la `EnumPortSuppliers` método y un proveedor de puertos enumera sus puertos a través de la `EnumPorts` método.

Además del registro COM típico, un proveedor de puertos debe registrarse en Visual Studio colocando su CLSID y el nombre en ubicaciones de registro específicas. Una función auxiliar `SetMetric` de SDK de depuración denominada controla esta tarea: se llama una vez para cada elemento que se va a registrar, por lo tanto:

```
SetMetric(metrictypePortSupplier,
    <GUID of your port supplier>,
    metricCLSID,
    <CLSID of your port supplier>,
    false,
    NULL)
SetMetric(metrictypePortSupplier,
    <GUID of your port supplier>,
    metricName,
    <name of your port supplier>,
    false,
    NULL);
```

Un proveedor de puertos `RemoveMetric` se anula el registro llamando a sí mismo llamando a (otra función auxiliar del SDK de depuración) una vez para cada elemento que se registró, por lo tanto:

```
RemoveMetric(metrictypePortSupplier,
    <GUID of your port supplier>,
    metricCLSID,
    NULL);
RemoveMetric(metrictypePortSupplier,
    <GUID of your port supplier>,
    metricName,
    NULL);
```

NOTE

Las [aplicaciones auxiliares del SDK para la depuración](#) `SetMetric` y `RemoveMetric` son funciones estáticas definidas en `dbgmetric.h` y compiladas en `ad2de.lib`. Las `metrictypePortSupplier` `metricCLSID` aplicaciones, `metricName`, y helper también se definen en `dbgmetric.h`.

Un proveedor de puertos puede proporcionar su nombre y GUID a través de los métodos [GetPortSupplierName](#) y [GetPortSupplierId](#), respectivamente.

Vea también

- [Implementar un proveedor portuario](#)
- [Ayudantes del SDK para la depuración](#)
- [Proveedores portuarios](#)

Interfaces de proveedores de puertos requeridas

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un proveedor de puertos debe implementar el [IDebugPortSupplier2](#) interfaz. [IDebugPortSupplier2](#)

Un proveedor de puertos suministra puertos y los implementa. Por lo tanto, debe ejecutar las siguientes interfaces:

- [IDebugPort2](#)

Describe el puerto y enumera todos los procesos que se ejecutan en el puerto.

- [IDebugPortEx2](#)

Proporciona para iniciar y finalizar procesos en el puerto.

- [IDebugPortNotify2](#)

Proporciona un mecanismo para que los programas que se ejecutan dentro del contexto de este puerto le notifiquen la creación y destrucción de nodo de programa. Para obtener más información, consulte [Nodos de programa](#).

- [IConnectionPointContainer](#)

Proporciona un punto de conexión para [IDebugPortEvents2](#).

Operación de proveedores portuarios

El receptor [IDebugPortEvents2](#) recibe notificaciones cuando el proceso y los programas se crean y destruyen en un puerto. Se requiere un puerto para enviar [IDebugProcessCreateEvent2](#) cuando se crea un proceso y [IDebugProcessDestroyEvent2](#) cuando se destruye un proceso en el puerto. También es necesario un puerto para enviar [IDebugProgramCreateEvent2](#) cuando se crea un programa y [IDebugProgramDestroyEvent2](#) cuando se destruye un programa en un proceso que se ejecuta en el puerto.

Normalmente, un puerto envía eventos de creación y destrucción de programas en respuesta a los métodos [AddProgramNode](#) y [RemoveProgramNode](#), respectivamente.

Dado que un puerto puede iniciar y terminar tanto los procesos físicos como los programas lógicos, el motor de depuración también debe implementar las siguientes interfaces:

- [IDebugProcess2](#)

Describe el proceso físico. Al menos deben implementarse los siguientes métodos:

- [EnumPrograms](#)
- [GetName](#)
- [GetServer](#)
- [GetPhysicalProcessId](#)
- [GetProcessId](#)
- [GetAttachedSessionName](#)

- [IDebugProcessEx2](#)

Proporciona una manera para que el SDM se adjunte y se desvincule de un proceso.

- [IDebugProgram2](#)

Describe el programa lógico. Al menos deben implementarse los siguientes métodos:

- [GetName](#)
- [GetProcess](#)
- [GetProgramId](#)

- [IDebugProgramEx2](#)

Proporciona una manera para que el SDM se asocie a este programa.

Vea también

- [Implementación de un proveedor portuario](#)

Internos de extensión paralela para .NET Framework

06/04/2020 • 2 minutes to read • [Edit Online](#)

En esta sección se describen los tipos internos, métodos y campos de clases que le ayudan a implementar un depurador personalizado para las extensiones paralelas de .NET Framework.

En esta sección

[Clase de tarea](#) Describe los miembros de [System.Threading.Tasks.Task](#) datos internos de la clase.

[Clase TaskScheduler](#) Describe los miembros de [System.Threading.Tasks.TaskScheduler](#) datos internos de la clase.

[Clase ContingentProperties](#) Describe los miembros de `System.Threading.Tasks.ContingentProperties` datos internos de la clase.

[Estructura AsyncTaskMethodBuilder](#) Describe los miembros [AsyncTaskMethodBuilder](#) internos de la estructura.

[AsyncTaskMethodBuilder<TResult> estructura](#) Describe los [AsyncTaskMethodBuilder<TResult>](#) miembros internos de la estructura.

[Estructura AsyncVoidMethodBuilder](#) Describe los miembros [AsyncVoidMethodBuilder](#) internos de la estructura.

Vea también

- [System.Threading.Tasks.Task](#)
- [System.Threading.Tasks.TaskScheduler](#)
- [Extensibilidad del depurador de Visual Studio](#)
- [Programación en paralelo](#)

Clase de tarea - miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este artículo se describen los miembros internos de la [System.Threading.Tasks.Task](#) clase que le ayudan a implementar un depurador personalizado. Para obtener información general sobre [Task](#) esta clase, consulte el artículo de referencia.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class public auto ansi System.Threading.Tasks.Task
    extends System.Object
    implements System.Threading.IThreadPoolWorkItem,
               System.IAsyncResult,
               System.IDisposable,
               System.Threading.ICancelableOperation
```

Miembros

Métodos

NOMBRE	DESCRIPCIÓN
SetNotificationForWaitCompletion (Método)	Establece o borra el bit de estado de TASK_STATE_WAIT_COMPLETION_NOTIFICATION.
NotifyDebuggerOfWaitCompletion (Método)	Método de marcador de posición utilizado como destino de punto de interrupción por el depurador.

Fields

NOMBRE	DESCRIPCIÓN
m_action	El delegado que representa el código Task que se va a ejecutar en el objeto.
m_contingentProperties	Almacena propiedades adicionales Task del objeto.
m_parent	El campo de System.Threading.Tasks.Task respaldo de la propiedad primaria.
m_stateFlags	Almacena información sobre el Task estado actual del objeto.
m_stateObject	Objeto que representa los datos que usará la acción.

NOMBRE	DESCRIPCIÓN
<code>m_taskId</code>	El campo de <code>System.Threading.Tasks.Task.Id</code> respaldo de la propiedad.
<code>s_taskIdCounter</code>	El siguiente identificador <code>Task</code> disponible para un objeto.
<code>TASK_STATE_CANCELED</code>	Indica que la tarea cancelada antes de que alcanzara el estado de ejecución o que la tarea confirmara su cancelación y se completó sin excepción.
<code>TASK_STATE_EXECUTED</code>	Indica que la tarea se está ejecutando.
<code>TASK_STATE_FAULTED</code>	Indica que la tarea se ha completado debido a una excepción no controlada.
<code>TASK_STATE_RAN_TO_COMPLETION</code>	Indica que la tarea ha completado la ejecución correctamente.
<code>TASK_STATE_WAITING_ON_CHILDREN</code>	Indica que la tarea ha terminado de ejecutar su delegado y está esperando implícitamente a que finalicen las tareas secundarias adjuntas.

Observaciones

Los siguientes métodos internos son útiles para `Task` un motor de depurador porque marcan la entrada a la ejecución de código:

- `Execute`
- `ExecuteEntry`
- `ExecuteWithThreadLocal`
- `Finish`
- `InnerInvoke`
- `InternalWait`

Vea también

- `System.Threading.Tasks.Task`
- [Internos de extensión paralela para .NET Framework](#)

m_action campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

El delegado que representa el código [Task](#) que se va a ejecutar en el objeto.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field assembly object m_action
```

Observaciones

Este es `action` el [Task](#) parámetro en el constructor.

Vea también

- [Clase de tarea: miembros internos](#)

m_contingentProperties campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Almacena propiedades adicionales [Task](#) del objeto.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field assembly class System.Threading.Tasks.Task/ContingentProperties  
modreq(System.Runtime.CompilerServices.IsVolatile) m_contingentProperties
```

Observaciones

Vea también

- [Clase de tarea: miembros internos](#)

m_parent campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

El campo de respaldo para [Task](#) el elemento primario de la tarea.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field private initonly class System.Threading.Tasks.Task m_parent
```

Vea también

- [Clase de tarea: miembros internos](#)

m_stateFlags campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Almacena información sobre el [Task](#) estado actual del objeto.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field assembly int32 modreq(System.Runtime.CompilerServices.IsVolatile) m_stateFlags
```

Observaciones

Normalmente se [System.Threading.Tasks.Task.Status](#) utiliza la propiedad para tener acceso a este valor.

Este miembro puede ser cualquier combinación de los siguientes valores:

- [TASK_STATE_EXECUTED](#)
- [TASK_STATE_FAULTED](#)
- [TASK_STATE_CANCELED](#)
- [TASK_STATE_WAITING_ON_CHILDREN](#)
- [TASK_STATE_RAN_TO_COMPLETION](#)

Vea también

- [Clase de tarea: miembros internos](#)

m_stateObject campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Objeto que representa los datos que utilizará la acción.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field assembly object m_stateObject
```

Observaciones

Este es `state` el [Task](#) parámetro en el constructor. También es el campo [System.Threading.Tasks.Task.AsyncState](#) de respaldo de la propiedad.

Vea también

- [Clase de tarea: miembros internos](#)

m_taskId campo

06/04/2020 • 2 minutes to read • [Edit Online](#)

El campo de [System.Threading.Tasks.Task.Id](#) respaldo de la propiedad.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field private int32 m_taskId
```

Observaciones

Vea también

- [Clase de tarea: miembros internos](#)

s_taskIdCounter (Campo)

06/04/2020 • 2 minutes to read • [Edit Online](#)

El siguiente identificador [Task](#) disponible para un objeto.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly int32 s_taskIdCounter
```

Observaciones

Vea también

- [Clase de tarea: miembros internos](#)

campo TASK_STATE_CANCELED

06/04/2020 • 2 minutes to read • [Edit Online](#)

La tarea se canceló antes de que alcanzara el estado de ejecución o confirmó su cancelación y se completó sin excepción.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly literal int32 TASK_STATE_CANCELED = int32(0x00800000)
```

Observaciones

Si el campo [m_stateFlags](#) contiene [Status](#) este [System.Threading.Tasks.TaskStatus](#) valor, la propiedad devuelve .

Vea también

- [Clase de tarea: miembros internos](#)

campo TASK_STATE_EXECUTED

06/04/2020 • 2 minutes to read • [Edit Online](#)

La tarea se está ejecutando pero aún no se ha completado.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly literal int32 TASK_STATE_EXECUTED = int32(0x00020000)
```

Observaciones

Si el campo [m_stateFlags](#) contiene [Status](#) este [System.Threading.Tasks.TaskStatus](#) valor, la propiedad devuelve .

Vea también

- [Clase de tarea: miembros internos](#)

campo TASK_STATE_FAULTED

06/04/2020 • 2 minutes to read • [Edit Online](#)

La tarea se completó debido a una excepción no controlada.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly literal int32 TASK_STATE_FAULTED = int32(0x00400000)
```

Observaciones

Si el campo [m_stateFlags](#) contiene [Status](#) este [System.Threading.Tasks.TaskStatus](#) valor, la propiedad devuelve .

Vea también

- [Clase de tarea: miembros internos](#)

campo TASK_STATE_RAN_TO_COMPLETION

06/04/2020 • 2 minutes to read • [Edit Online](#)

La tarea terminó de ejecutarse correctamente.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly literal int32 TASK_STATE_RAN_TO_COMPLETION = int32(0x02000000)
```

Observaciones

Si el campo [m_stateFlags](#) contiene [Status](#) este [System.Threading.Tasks.TaskStatus](#) valor, la propiedad devuelve .

Vea también

- [Clase de tarea: miembros internos](#)

campo TASK_STATE_WAITING_ON_CHILDREN

06/04/2020 • 2 minutes to read • [Edit Online](#)

La tarea ha terminado de ejecutar su delegado y está esperando implícitamente a que se completen las tareas secundarias adjuntas.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field static assembly literal int32 TASK_STATE_WAITING_ON_CHILDREN = int32(0x01000000)
```

Observaciones

Si el campo [m_stateFlags](#) contiene [Status](#) este [System.Threading.Tasks.TaskStatus](#) valor, la propiedad devuelve .

Vea también

- [Clase de tarea: miembros internos](#)

SetNotificationForWaitCompletion (Método)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Establece o borra el bit de estado de TASK_STATE_WAIT_COMPLETION_NOTIFICATION.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Sintaxis

```
internal void SetNotificationForWaitCompletion(bool enabled)
```

Parámetros

`enabled`

`true` para establecer el bit; `false` para desajustar el bit.

Excepciones

Observaciones

El depurador establece este bit para ayudar a salir de un cuerpo de método asincrónico. Si `enabled` `true` es, se debe llamar a este método solo en una tarea que aún no se ha completado. Cuando `enabled` `false` es, se puede llamar a este método en las tareas completadas. En cualquier caso, solo se debe usar para tareas de estilo de promesa.

Requisitos

Vea también

- [Clase de tarea: miembros internos](#)

NotifyDebuggerOfWaitCompletion (método)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Método de marcador de posición utilizado como destino de punto de interrupción por el depurador. Este método no debe estar insertado ni optimizado.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Sintaxis

```
private void NotifyDebuggerOfWaitCompletion()
```

Observaciones

Todas las operaciones de combinación con una tarea deben llamar a este método si se establece su bit de notificación del depurador.

Requisitos

Vea también

- [Clase de tarea: miembros internos](#)

TaskScheduler (Clase) - Miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este artículo se describen los miembros internos de la [System.Threading.Tasks.TaskScheduler](#) clase que le ayudan a implementar un depurador personalizado. Para obtener información general sobre [TaskScheduler](#) esta clase, consulte el artículo de referencia.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class public abstract auto ansi beforefieldinit System.Threading.Tasks.TaskScheduler
    extends System.Object
```

Miembros

Métodos

NOMBRE	DESCRIPCIÓN
GetScheduledTasksForDebugger	Recupera una matriz de todas las tareas programadas.
GetTaskSchedulersForDebugger	Recupera una matriz TaskScheduler de todos los objetos que están activos actualmente.

Observaciones

Vea también

- [System.Threading.Tasks.TaskScheduler](#)
- [Internos de extensión paralela para .NET Framework](#)

GetScheduledTasksForDebugger (Método)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Recupera una matriz de todas las tareas programadas.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.method assembly hideby sig instance class System.Threading.Tasks.Task[] GetScheduledTasksForDebugger() cil managed
```

Valor devuelto

Matriz de todas las tareas programadas. Cada tarea se está ejecutando o ha terminado de ejecutarse.

Observaciones

Este método no es seguro para subprocesos y no debe [TaskScheduler](#) usarlo simultáneamente con otras instancias de . Llame a este método desde un depurador solo cuando el depurador haya suspendido todos los demás subprocesos.

Vea también

- [Clase TaskScheduler](#)

GetTaskSchedulersForDebugger (Método)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Recupera una matriz [TaskScheduler](#) de todos los objetos que están activos actualmente.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.method assembly hideby sig static class System.Threading.Tasks.TaskScheduler[] GetTaskSchedulersForDebugger()  
cil managed
```

Valor devuelto

Matriz de [TaskScheduler](#) todos los objetos [AppDomain](#) que están activos actualmente en este archivo .

Observaciones

Este método no es seguro para subprocessos y no debe [TaskScheduler](#) usarlo simultáneamente con otras instancias de . Llame a este método desde un depurador solo cuando el depurador haya suspendido todos los demás subprocessos.

Vea también

- [Clase TaskScheduler](#)

Clase ContingentProperties - miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

Contiene propiedades adicionales [Task](#) para un objeto.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class auto ansi nested assembly beforefieldinit ContingentProperties
    extends System.Object
```

Miembros

Fields

NOMBRE	DESCRIPCIÓN
m_children	La lista de tareas secundarias que se registran con esta tarea.

Observaciones

.NET Framework inicializa los campos de esta clase solo cuando son necesarios.

Vea también

- [Internos de extensión paralela para .NET Framework](#)

campo m_children

06/04/2020 • 2 minutes to read • [Edit Online](#)

La lista de tareas secundarias que se registran con esta tarea.

Espacio de nombres: [System.Threading.Tasks](#)

Ensamblado: mscorlib (en *mscorlib.dll*)

Dado que no se puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field public class System.Collections.Generic.List`1<class System.Threading.Tasks.Task> m_children
```

Observaciones

Mientras se ejecuta la tarea, solo el subproceso que ejecuta la tarea debe tener acceso a esta matriz.

Si se completa la tarea, otros subprocesos pueden tener acceso a este campo siempre y cuando no le agreguen nada ni quiten nada de él.

Vea también

- [Clase ContingentProperties](#)

Estructura AsyncTaskMethodBuilder - miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este tema se describen los miembros internos de la [AsyncTaskMethodBuilder](#) clase. Para obtener información general sobre [AsyncTaskMethodBuilder](#) esta clase, consulte el tema de referencia.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class public sequential ansi sealed beforefieldinit System.Runtime.CompilerServices.AsyncTaskMethodBuilder
    extends System.ValueType
    implements System.Runtime.CompilerServices.IAsyncMethodBuilder
```

Miembros internos

NOMBRE	DESCRIPCIÓN
Propiedad ObjectIdForDebugger	Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.
m_builder campo	Representa el objeto de generador genérico al que se delega esta instancia no genérica.

Vea también

- [AsyncTaskMethodBuilder](#)
- [Internos de extensión paralela para .NET Framework](#)

Propiedad

AsyncTaskMethodBuilder.ObjectIdForDebugger

06/04/2020 • 2 minutes to read • [Edit Online](#)

Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
private object ObjectIdForDebugger
```

Vea también

- [AsyncTaskMethodBuilder<TResult>](#)
- [Internos de extensión paralela para .NET Framework](#)

Campo AsyncTaskMethodBuilder.m_builder

06/04/2020 • 2 minutes to read • [Edit Online](#)

Representa el objeto de generador genérico al que se delega esta instancia no genérica.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field private valuetype System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1<valuetype  
System.Threading.Tasks.VoidTaskResult> m_builder
```

Observaciones

Vea también

- [AsyncTaskMethodBuilder](#)
- [Internos de extensión paralela para .NET Framework](#)

Estructura DeSasyncTaskMethodBuilder<TResult> - miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este tema se describen los miembros internos de la [AsyncTaskMethodBuilder<TResult>](#) clase. Para obtener información general sobre [AsyncTaskMethodBuilder<TResult>](#) esta clase, consulte el tema de referencia.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class public sequential ansi sealed beforefieldinit
System.Runtime.CompilerServices.AsyncTaskMethodBuilder`1<TResult>
    extends System.ValueType
    implements System.Runtime.CompilerServices.IAsyncMethodBuilder
```

Miembros internos

NOMBRE	DESCRIPCIÓN
Propiedad ObjectIdForDebugger	Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.
m_task campo	Representa la tarea compilada inicializada perezosamente.

Vea también

- [AsyncTaskMethodBuilder<TResult>](#)
- [Internos de extensión paralela para .NET Framework](#)

AsyncTaskMethodBuilder<TResult>. Propiedad ObjectIdForDebugger

06/04/2020 • 2 minutes to read • [Edit Online](#)

Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
private object ObjectIdForDebugger
```

Vea también

- [AsyncTaskMethodBuilder<TResult>](#)
- [Internos de extensión paralela para .NET Framework](#)

Campo `.m_task<de>AsyncTaskMethodBuilder TResult`

06/04/2020 • 2 minutes to read • [Edit Online](#)

Representa la tarea compilada inicializada perezosamente.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field private class System.Threading.Tasks.Task`1<TResult> m_task
```

Vea también

- [AsyncTaskMethodBuilder<TResult>](#)
- [Internos de extensión paralela para .NET Framework](#)

Estructura AsyncVoidMethodBuilder - miembros internos

06/04/2020 • 2 minutes to read • [Edit Online](#)

En este tema se describen los miembros internos de la [AsyncVoidMethodBuilder](#) clase. Para obtener información general sobre [AsyncVoidMethodBuilder](#) esta clase, consulte el tema de referencia.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a estos miembros internos desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.class public sequential ansi sealed beforefieldinit System.Runtime.CompilerServices.AsyncVoidMethodBuilder
    extends System.ValueType
    implements System.Runtime.CompilerServices.IAsyncMethodBuilder
```

Miembros internos

NOMBRE	DESCRIPCIÓN
Propiedad ObjectIdForDebugger	Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.
campo m_objectIdForDebugger	Representa el objeto inicializado perezosamente utilizado por el depurador para identificar de forma única este generador.

Vea también

- [AsyncVoidMethodBuilder](#)
- [Internos de extensión paralela para .NET Framework](#)

Propiedad

AsyncVoidMethodBuilder.ObjectIdForDebugger

06/04/2020 • 2 minutes to read • [Edit Online](#)

Obtiene un objeto que se puede usar para identificar de forma única este generador en el depurador.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
private object ObjectIdForDebugger
```

Vea también

- [AsyncVoidMethodBuilder](#)
- [Internos de extensión paralela para .NET Framework](#)

Campo

AsyncVoidMethodBuilder.m_objectIdForDebugger

06/04/2020 • 2 minutes to read • [Edit Online](#)

Representa el objeto inicializado perezosamente utilizado por el depurador para identificar de forma única este generador.

Espacio de nombres: [System.Runtime.CompilerServices](#)

Ensamblado: mscorlib (en mscorlib.dll)

Dado que no puede tener acceso a este miembro interno desde .NET Framework, se proporciona la sintaxis siguiente en Common Intermediate Language (CIL).

Sintaxis

```
.field private object m_objectIdForDebugger
```

Vea también

- [AsyncVoidMethodBuilder](#)
- [Internos de extensión paralela para .NET Framework](#)

Control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

En esta sección se describen las opciones Visual Studio para implementar el control de código fuente como una característica integrada de , ya sea a través de un complemento de control de código fuente o un control de código fuente VSPackage.

En esta sección

- [Essentials](#)

Presenta la información importante que debe tener para empezar a trabajar con el control de código fuente.

- [Información general](#)

Una visión general de las dos opciones disponibles para implementar el control de código fuente.

- [Creación de un complemento de control de código fuente](#)

Describe cómo crear un complemento de control de código fuente Visual Studio que proporciona funcionalidad de control de código fuente a través de la interfaz de usuario (UI) del control de código fuente.

- [Creación de un VSPackage de control de código fuente](#)

Describe cómo crear un control de código fuente VSPackage que no solo proporciona Visual Studio funcionalidad de control de código fuente, pero se puede usar para personalizar la interfaz de usuario del control de código fuente.

Secciones relacionadas

- [Complementos de control de código fuente](#)

Referencia para la API de complemento de Control de código fuente.

- [Ampliación de proyectos](#)

Describe cómo usar los proyectos y soluciones de Visual Studio para organizar los archivos de código y de recursos, así como la implementación del control de código fuente.

Conceptos básicos de la integración del control de código fuente

06/04/2020 • 3 minutes to read • [Edit Online](#)

Visual Studio admite dos tipos de integración de control de código fuente: un complemento de control de código fuente que proporciona funcionalidad básica y se compila mediante la API de complemento de Control de código fuente (anteriormente conocida como la API de MSSCCI) y una solución de integración de control de código fuente basada en VSPackage que proporciona una funcionalidad más sólida.

Complemento control de código fuente

Un complemento de Control de código fuente se escribe como un archivo DLL que implementa la API de complemento de Control de código fuente. La funcionalidad de integración de registro y control de código fuente se proporciona a través de la API. Este enfoque es más fácil de implementar que un Visual Studio control de código fuente VSPackage y usa la interfaz de usuario (UI) para la mayoría de las operaciones de control de código fuente.

Para implementar un complemento de control de código fuente mediante la API de complemento de Control de código fuente, siga estos pasos:

1. Cree un archivo DLL que implemente las funciones especificadas en [Complementos de Control de código fuente](#).
2. Registre el archivo DLL realizando las entradas de registro adecuadas, como se describe en [Cómo: instalar un complemento de Control de código fuente](#).
3. Cree una interfaz de usuario auxiliar y muésela cuando se lo solicite el paquete de adaptador de Control de código fuente (el componente que controla la Visual Studio funcionalidad de control de código fuente a través de complementos de control de código fuente).

Para obtener más información, consulte [Creación de un complemento de control de código fuente](#).

Control de código fuente VSPackage

Una implementación de VSPackage de control de Visual Studio código fuente le permite desarrollar un reemplazo personalizado para la interfaz de usuario del control de código fuente. Este enfoque proporciona un control completo sobre la integración del control de código fuente, pero requiere que proporcione los elementos de la interfaz de usuario e implemente las interfaces de control de código fuente que, de lo contrario, se proporcionarían bajo el enfoque de complemento.

Para implementar un control de código fuente VSPackage, debe:

1. Cree y registre su propio control de código fuente VSPackage, como se describe en [Registro y selección](#).
2. Reemplace la interfaz de usuario de control de código fuente predeterminada por la interfaz de usuario personalizada. Consulte [Interfaz de usuario personalizada](#).
3. Especifique los pictogramas que se van a usar y controle los eventos de glifo del Explorador de soluciones. Consulte [Control de pictogramas](#).
4. Controlar eventos de edición de consulta y guardado de consulta, como se muestra en [Query Edit Query Save](#).

Para obtener más información, vea [Crear un VSPackage de control de código fuente](#).

Vea también

- [Información general](#)
- [Creación de un complemento de control de código fuente](#)
- [Creación de un VSPackage de control de código fuente](#)

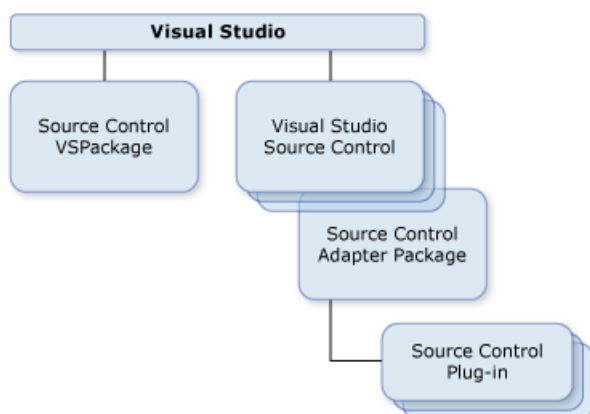
Información general de la integración del control de código fuente

06/04/2020 • 10 minutes to read • [Edit Online](#)

En esta sección se comparan las dos formas de integrar seingen en el control de código fuente de Visual Studio; un complemento de control de código fuente y un VSPackage que proporciona una solución de control de código fuente y resalta las nuevas características de control de código fuente. Visual Studio permite la conmutación manual entre VSPackages de control de código fuente y complementos de control de código fuente, así como conmutación automática basada en solución.

Integración del control de código fuente

Visual Studio admite dos tipos de opciones de integración de control de código fuente. En todas Visual Studio las versiones de , todavía puede integrar un complemento basado en la API de complemento de Control de código fuente (anteriormente también conocida como la API de MSSCCI), que proporciona funcionalidad básica de control de código fuente mientras usa la interfaz de usuario (UI) del control de código fuente de Visual Studio. Un control de código fuente VSPackage, por otro Visual Studio SDK lado, proporciona una nueva ruta de acceso de integración profunda adecuada para la integración de control de código fuente que exige un alto nivel de sofisticación y autonomía en su modelo de control de código fuente.



Complemento control de código fuente

Todas las versiones de Visual Studio admiten la especificación de api de complemento de Control de código fuente versión 1.2 como una ruta de acceso de integración. Un implementador de complemento de control de código fuente escribe un archivo DLL que implementa las funciones de la API de complemento de Control de código fuente para la integración y el registro del control de código fuente, tal como se describe en Creación de [un complemento](#) de control de código fuente . En este enfoque, el entorno de desarrollo Visual Studio integrado (IDE) usa la interfaz de usuario para los cuadros de diálogo, como la protección, la desprotección, las páginas de propiedades de herramientas/opciones, las barras de herramientas y los glifos de control de código fuente. La estricta adherencia a la API de complemento de Control de código fuente asegura una fácil integración en Visual Studio y una experiencia sin problemas para el usuario. Esto significa que el complemento de control de código fuente debe implementar la mayoría de las funciones y devoluciones de llamada detalladas en la API.

Para implementar un complemento de control de código fuente mediante la API de complemento de Control de código fuente, siga estos pasos:

1. Cree un archivo DLL que implemente las funciones especificadas en [Complementos](#) de Control de código fuente .

2. Registre el archivo DLL realizando las entradas de registro adecuadas (descritas en [Cómo: instalar un complemento](#) de Control de código fuente).
3. Crear una interfaz de usuario auxiliar y mostrar cuando se le solicite el paquete de adaptador de Control de código fuente (el componente de Visual Studio que controla la funcionalidad de control de código fuente a través de complementos de control de código fuente)

En respuesta a un comando de control de código fuente, el IDE de Visual Studio presenta una interfaz de usuario estándar para las operaciones básicas y, a continuación, pasa la información al complemento de control de código fuente a través de las funciones definidas en la API de complemento de Control de código fuente. Para las opciones avanzadas, se puede llamar al complemento de control de código fuente para presentar su propia interfaz de usuario, por ejemplo, la exploración de un proyecto controlado por código fuente. Esto significa que al usuario se le pueden presentar dos estilos posiblemente diferentes de la interfaz de usuario al tratar con el control de código fuente: la interfaz de usuario que presenta Visual Studio y la interfaz de usuario que presenta el complemento de control de código fuente. Esto es más notable con las operaciones avanzadas de control de código fuente.

Inconvenientes para implementar un complemento de control de código fuente

- Para las características avanzadas, el usuario puede ver dos estilos diferentes de interfaces, lo que conduce a una posible confusión.
- El complemento de control de código fuente se limita al modelo de control de código fuente implicado por la API de complemento de Control de código fuente.
- La API de complemento de Control de código fuente puede ser demasiado restrictiva para algunos escenarios de control de código fuente.

Ventajas de implementar un complemento de control de código fuente

- Visual Studio proporciona toda la interfaz de usuario para todas las operaciones básicas de control de código fuente para que el complemento de control de código fuente no tenga que implementar una interfaz de usuario potencialmente compleja.
- Debido a la API estricta, el complemento de control de código fuente puede interactuar fácilmente con programas de control de código fuente externos para proporcionar una funcionalidad más extensa; A Visual Studio no le importa demasiado cómo se logra la funcionalidad del control de código fuente, solo que se realiza de acuerdo con la API de complemento de Control de código fuente.
- Es más fácil implementar un complemento de control de código fuente que un control de código fuente VSPackage.

Control de código fuente VSPackage

Visual Studio SDK permite una integración profunda en Visual Studio con control total de la funcionalidad de control de código fuente y reemplazo completo de la interfaz de usuario de control de código fuente proporcionada por Visual Studio. Un control de código fuente VSPackage se registra con Visual Studio y proporciona funcionalidad de control de código fuente. Aunque varios VSPackages de control de código fuente se pueden registrar con Visual Studio, solo uno de ellos puede estar activo en cualquier momento. Un control de código fuente VSPackage tiene control total sobre la funcionalidad y la apariencia del control de código fuente en Visual Studio mientras está activo. Todos los demás VSPackages de control de código fuente que se pueden registrar en el sistema están inactivos y no mostrará ninguna interfaz de usuario en absoluto.

Implementar un control de código fuente VSPackage requiere una estrategia de "todo o nada". El creador de un control de código fuente VSPackage debe invertir una cantidad significativa de esfuerzo en la implementación de un número de interfaces de control de código fuente y nuevos elementos de interfaz de usuario (cuadros de diálogo, menús y barras de herramientas) para cubrir toda la funcionalidad de control de código fuente. Consulte [crear un VSPackage de control de código fuente](#) para obtener más detalles.

Inconvenientes para implementar un control de código fuente VSPackage

- El VSPackage debe implementar una serie de interfaces complejas para integrarse correctamente con Visual Studio.
- El VSPackage debe proporcionar toda la interfaz de usuario necesaria para el control de código fuente; Visual Studio no proporcionará asistencia en esta área.
- Un control de código fuente VSPackage está íntimamente vinculado a Visual Studio y no puede funcionar con programas independientes, por lo que la funcionalidad no se puede compartir tan fácilmente con una versión externa del programa de control de código fuente.

Ventajas de implementar un control de código fuente VSPackage

- Dado que el VSPackage tiene control total sobre la interfaz de usuario y la funcionalidad del control de código fuente, el usuario se presenta con una interfaz perfecta para el control de código fuente.
- El VSPackage no se limita a un modelo de control de código fuente determinado.

Vea también

- [Control de código fuente](#)
- [Creación de un complemento de control de código fuente](#)
- [Creación de un VSPackage de control de código fuente](#)
- [Novedades del control de código fuente](#)

Novedades del Control de código fuente para el SDK de Visual Studio 2015

06/04/2020 • 3 minutes to read • [Edit Online](#)

En Visual Studio SDKel , puede proporcionar una solución de control de código fuente profundamente integrada mediante la implementación de un control de código fuente VSPackage. En esta sección se describen las características del control de código fuente VSPackages y proporciona información general de los pasos de implementación.

El control de código fuente VSPackage

Visual Studio admite dos tipos de soluciones de control de código fuente. En todas las versiones de Visual Studio, todavía puede integrar un complemento basado en API de complemento de Control de código fuente. También puede crear un VSPackage para el control Visual Studio SDK de código fuente que proporciona una integración profunda, ruta de acceso adecuada para las soluciones de control de código fuente que requieren un alto nivel de sofisticación y autonomía.

Un VSPackage puede agregar casi cualquier tipo de funcionalidad a Visual Studio. Un control de código fuente VSPackage proporciona una característica de control de código fuente completa para Visual Studio, desde la interfaz de usuario presentada al usuario a la comunicación back-end con el sistema de control de código fuente.

Implementar un control de código fuente VSPackage requiere una estrategia de "todo o nada". El creador de un control de código fuente VSPackage debe invertir una cantidad significativa de esfuerzo en la implementación de una serie de interfaces de control de código fuente y nuevos elementos de interfaz de usuario (cuadros de diálogo, menús y barras de herramientas) para cubrir toda la funcionalidad de control de código fuente, así como las interfaces necesarias de cualquier paquete para integrar se integran correctamente con Visual Studio.

Los pasos siguientes proporcionan una visión general de lo que se necesita para implementar un paquete de control de código fuente. Para obtener más información, vea [Crear un VSPackage](#) de control de código fuente .

1. Cree un VSPackage que ofrezca un servicio de control de código fuente privado.
2. Implemente las interfaces en los servicios relacionados con el control de [IVsQueryEditQuerySave2](#) código [IVsSccProvider](#) fuente que ofrece Visual Studio (por ejemplo, la interfaz y la).
3. Registre el control de código fuente VSPackage.
4. Implemente toda la interfaz de usuario del control de código fuente, incluidos los elementos de menú, los cuadros de diálogo, las barras de herramientas y los menús contextuales.
5. Todos los eventos relacionados con el control de código fuente se pasan al control de código fuente VSackage cuando está activo y debe controlarse por el VSPackage.
6. El control de código fuente VSPackage debe [IVsSolutionEvents3](#) escuchar eventos como los que implementan la interfaz, [IVsTrackProjectDocuments2](#) así como eventos de documento de proyecto de seguimiento (TPD) (según lo implementado por la interfaz) y tomar las medidas necesarias.

Vea también

- [IVsQueryEditQuerySave2](#)
- [IVsSccProvider](#)
- [IVsSolutionEvents3](#)

- [IVsTrackProjectDocuments2](#)
- [Información general](#)
- [Creación de un VSPackage de control de código fuente](#)

Crear un complemento de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

El SDK de Visual Studio proporciona recursos que Visual Studio le permiten agregar la capacidad de control de código fuente al entorno de desarrollo integrado (IDE). Le permite usar cualquier archivo DLL de complemento que cumpla con la API de complemento de Control de código fuente que se describe en esta documentación.

En esta sección

- [Introducción](#)

Describe cómo instalar un complemento de control de código fuente y resalta las versiones de API de complemento de Control de código fuente disponibles actualmente.

- [Architecture](#)

Utiliza un diagrama de arquitectura para explicar la integración de un complemento de control de código fuente con el Visual Studio IDE.

- [Guía de prueba](#)

Proporciona instrucciones sobre cómo probar la instalación y el funcionamiento de un complemento de control de código fuente.

Secciones relacionadas

- [Crear un control de código fuente VSPackage](#)

Describe cómo crear un control de código fuente VSPackage que no Visual Studio solo proporciona funcionalidad de control de código fuente, sino que reemplaza la interfaz de usuario del control de código fuente.

- [Complementos de control de código fuente](#)

Proporciona una lista completa de todos los elementos de la API de complemento de Control de código fuente.

- [Control de código fuente](#)

Describe las opciones para implementar el control Visual Studio de código fuente como una característica integrada de .

Introducción a los complementos de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para crear un complemento de control de código fuente, debe crear un archivo DLL que implemente las funciones definidas en la API de complemento de Control de código fuente y, a continuación, registrar el archivo DLL con Visual Studio para que esté disponible para su uso en el control de versiones de código fuente.

Hay tres versiones de la API de complemento de Control de código fuente (versiones 1.1, 1.2 y 1.3) disponibles para los complementos de control de código fuente. La API de complemento de Control de código fuente documentada aquí es la versión 1.3. Fue diseñado para ser totalmente compatible con los plug-ins de control de código fuente que admiten las versiones 1.1 y 1.2. La sección [Novedades de la API de complemento de Control de código fuente versión 1.3](#) detalla las nuevas características admitidas en la versión más reciente de la API de complemento de Control de código fuente.

En esta sección

- [Cómo: Instalar un complemento de control de código fuente](#)

Describe cómo realizar las entradas del Registro necesarias para conectar un archivo DLL de control de código fuente.

- [Novedades de la API de complemento de Control de código fuente versión 1.3](#)

Proporciona una breve descripción general de los cambios realizados en la API de complemento de Control de código fuente en la versión 1.3.

- [Novedades de la API de complemento de Control de código fuente versión 1.2](#)

Proporciona una breve descripción general de los cambios realizados en la API de complemento de Control de código fuente en la versión 1.2.

Secciones relacionadas

- [Complementos de control de código fuente](#)

Proporciona una lista completa de todos los elementos de la API de complemento de Control de código fuente.

- [Crear un complemento de control de código fuente](#)

Define el SDK de complemento de Control de código fuente y describe los recursos incluidos.

Cómo: Instalar un complemento de control de código fuente

06/04/2020 • 13 minutes to read • [Edit Online](#)

La creación de un complemento de control de código fuente implica tres pasos:

1. Cree un archivo DLL con las funciones definidas en la sección de referencia de la API de complemento de Control de código fuente de esta documentación.
2. Implemente las funciones definidas por la API del complemento de Control de código fuente. Cuando Visual Studio lo solicite, haga que las interfaces y los cuadros de diálogo estén disponibles desde el complemento.
3. Registre el archivo DLL realizando las entradas de registro adecuadas.

Integración con Visual Studio

Visual Studio admite complementos de control de código fuente que se ajustan a la API de complemento de Control de código fuente.

Registrar el complemento de control de código fuente

Antes de que un entorno de desarrollo integrado (IDE) en ejecución pueda llamar al sistema de control de código fuente, primero debe buscar el archivo DLL del complemento de control de código fuente que exporta la API.

Para registrar el archivo DLL del complemento de control de código fuente

1. Agregue dos entradas bajo la clave **HKEY_LOCAL_MACHINE** en la subclave **SOFTWARE** que especifica la subclave de nombre de la empresa seguida de la subclave de nombre de producto. El patrón es ****HKEY_LOCAL_MACHINE nombre de la empresa de software \ < \ < > nombre **de producto > *valor* de entrada > = .** Las dos entradas siempre se denominan **SCCServerName** y **SCCServerPath**. Cada uno es una cadena normal.

Por ejemplo, si el nombre de la empresa es Microsoft y el producto de control de código fuente se denomina SourceSafe, esta ruta de acceso del Registro se **HKEY_LOCAL_MACHINE**. En esta subclave, la primera entrada, **SCCServerName**, es una cadena legible por el usuario que nombra el producto. La segunda entrada, **SCCServerPath**, es la ruta de acceso completa al archivo DLL del complemento de control de código fuente al que debe conectarse el IDE. A continuación se proporcionan entradas de registro de ejemplo:

ENTRADA DE REGISTRO DE EJEMPLO	VALOR DE EJEMPLO
HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, SourceSafe, SCCServerName	Microsoft Visual SourceSafe
HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, SourceSafe, SCCServerPath	c:-vss-win32-sssc.dll

NOTE

SCCServerPath es la ruta de acceso completa al complemento SourceSafe. El complemento de control de código fuente usará diferentes nombres de empresa y producto, pero las mismas rutas de acceso de entrada del Registro.

2. Las siguientes entradas de registro opcionales se pueden usar para modificar el comportamiento del complemento de control de código fuente. Estas entradas van en la misma subclave que **SccServerName** y **SccServerPath**.

- La entrada **HideInVisualStudioRegistry** se puede utilizar si no desea que el complemento de control Visual Studio de código fuente aparezca en la lista Selección de **complementos** de . Esta entrada también afectará al cambio automático al complemento de control de código fuente. Un posible uso de esta entrada es si proporciona un paquete de control de código fuente que reemplaza el complemento de control de código fuente, pero desea facilitar al usuario migrar desde el uso del complemento de control de código fuente al paquete de control de código fuente. Cuando se instala el paquete de control de código fuente, establece esta entrada del Registro, que oculta el complemento.

HideInVisualStudio es un valor DWORD y se establece en *1* para ocultar el complemento o *0* para mostrar el complemento. Si la entrada del Registro no aparece, el comportamiento predeterminado es mostrar el complemento.

- La entrada del Registro **DisableSccManager** se puede usar para deshabilitar u ocultar la opción de menú **Iniciar** < servidor de control de código fuente> que normalmente aparece en el > submenú **Control de código fuente** de archivos. Al seleccionar esta opción de menú se llama a la función **SccRunScc**. Es posible que el complemento de control de código fuente no admita un programa externo y, por lo tanto, es posible que desee deshabilitar u incluso ocultar la opción de menú **Iniciar**.

DisableSccManager es un valor DWORD y se establece en *0* para habilitar la opción de menú **Iniciar** < *1* < servidor de Control de código fuente>, se establece en *1* para deshabilitar la opción de menú y se establece en *2* para ocultar la opción de menú. Si no aparece esta entrada del Registro, el comportamiento predeterminado es mostrar la opción de menú.

ENTRADA DE REGISTRO DE EJEMPLO	VALOR DE EJEMPLO
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SourceSafe\HideInVisualStudio	1
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SourceSafe\DisableSccManager	1

3. Agregue la subclave **SourceCodeControlProvider**, en la clave **HKEY_LOCAL_MACHINE** de la subclave **SOFTWARE**.

Bajo esta subclave, la entrada del Registro **ProviderRegKey** se establece en una cadena que representa la subclave que colocó en el registro en el paso 1. El patrón es **HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\ProviderRegKey = SOFTWARE\<nombre>\>** de la empresa<nombre de producto>.

A continuación se muestra el contenido de ejemplo de esta subclave.

ENTRADA DEL REGISTRO	VALOR DE EJEMPLO
HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\ProviderRegKey	SOFTWARE-Microsoft-SourceSafe

NOTE

El complemento de control de código fuente usará los mismos nombres de subclave y entrada, pero el valor será diferente.

4. Cree una subclave denominada **InstalledSCCProviders** en la subclave **SourceCodeControlProvider** y, a continuación, coloque una entrada bajo esa subclave.

El nombre de esta entrada es el nombre legible por el usuario del proveedor (el mismo que el valor especificado para la entrada **SCCServerName**) y el valor es, una vez más, la subclave creada en el paso 1. El patrón se **HKEY_LOCAL_MACHINE, SOFTWARE, \SourceCodeControlProvider, > InstalledSCCProviders < nombre = para mostrar SOFTWARE* \ < nombre> \ > * de la empresa < nombre del producto .**

Por ejemplo:

ENTRADA DE REGISTRO DE EJEMPLO	VALOR DE EJEMPLO
HKEY_LOCAL_MACHINE, SOFTWARE, SourceCodeControlProvider, InstalledSCCProviders, Microsoft Visual SourceSafe	SOFTWARE-Microsoft-SourceSafe

NOTE

Puede haber varios complementos de control de código fuente registrados de esta manera. Así es Visual Studio como se encuentran todos los complementos basados en API de source Control instalados.

Cómo un IDE localiza el archivo DLL

El Visual Studio IDE tiene dos maneras de encontrar el archivo DLL del complemento de control de código fuente:

- Busque el complemento de control de código fuente predeterminado y conéctese a él de forma silenciosa.
- Busque todos los complementos de control de código fuente registrados, de los que el usuario elige uno.

Para buscar el archivo DLL de la primera manera, el IDE busca en la subclave **HKEY_LOCAL_MACHINE, Software, SourceCodeControlProvider**, la entrada **ProviderRegKey**. El valor de esta entrada apunta a otra subclave. A continuación, el IDE busca una entrada denominada **SccServerPath** en esa segunda subclave en **HKEY_LOCAL_MACHINE**. El valor de esta entrada apunta el IDE al archivo DLL.

NOTE

El IDE no carga archivos DLL desde rutas de acceso relativas (por ejemplo, *.-NewProvider.DLL*). Se debe especificar una ruta de acceso completa al archivo DLL (por ejemplo, *c:-Providers-NewProvider.DLL*). Esto refuerza la seguridad del IDE al impedir la carga de archivos DLL de complementos o suplantados.

Para buscar el archivo DLL de la segunda manera, el IDE busca en la subclave **HKEY_LOCAL_MACHINE, Software, SourceCodeControlProvider, InstalledSCCProviders** para todas las entradas. Cada entrada tiene un nombre y un valor. El IDE muestra una lista de estos nombres al usuario. Cuando el usuario elige un nombre, el IDE busca el valor del nombre seleccionado que apunta a una subclave. El IDE busca una entrada denominada **SccServerPath** en esa subclave en **HKEY_LOCAL_MACHINE**. El valor de esa entrada apunta el IDE al archivo DLL correcto.

Un complemento de control de código fuente debe admitir ambas formas de buscar el archivo DLL y, en

consecuencia, establece **ProviderRegKey**, sobrescribiendo cualquier configuración anterior. Lo que es más importante, debe agregarse a la lista de **InstalledSccProviders** para que el usuario pueda tener la opción de qué complemento de control de código fuente usar.

NOTE

Dado que se utiliza la **clave de HKEY_LOCAL_MACHINE**, solo se puede registrar un complemento de control Visual Studio de código fuente como el complemento de control de código fuente predeterminado en un equipo determinado (sin embargo, permite a los usuarios determinar qué complemento de control de código fuente que desean usar realmente para una solución determinada). Durante el proceso de instalación, compruebe si ya se ha establecido un complemento de control de código fuente; si es así, pregunte al usuario si debe establecer o no el nuevo complemento de control de código fuente que se está instalando como predeterminado. Durante la desinstalación, no quite otras subclaves del Registro que sean comunes a todos los complementos de control de código fuente de **HKEY_LOCAL_MACHINE** . eliminar sólo su subclave SCC en particular.

Cómo el IDE detecta la compatibilidad con la versión 1.2/1.3

¿Cómo Visual Studio detecta si un complemento admite la funcionalidad de la API de complemento de Control de código fuente versión 1.2 y 1.3? Para declarar la capacidad avanzada, el complemento de control de código fuente debe implementar la función correspondiente:

En Visual Studio primer lugar, comprueba el valor devuelto llamando a [SccGetVersion](#). Debe ser mayor o igual que 1.2.

A Visual Studio continuación, determina si la nueva funcionalidad `1pSccCaps` concreta se admite examinando el argumento en [ScclInitialize](#).

Si se cumplen ambas condiciones, se puede llamar a las nuevas funciones admitidas en las versiones 1.2 y 1.3.

Vea también

- [Introducción a los complementos de control de código fuente](#)

Qué's Nuevo en la versión 1.3 de la API de complemento de Control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

La versión 1.3 de la API de complemento de Control de código fuente presenta las siguientes funciones nuevas para proporcionar un control más avanzado.

Cambios

Las siguientes funciones son nuevas en la versión 1.3 de la API de complemento de Control de código fuente:

FUNCIÓN	INFORMACIÓN GENERAL
SccGetExtendedCapabilities	Permite notificar bits de capacidad adicionales
SccEnumChangedFiles	Permite el examen de archivos que tienen versiones más recientes en la base de datos de control de versiones que en el disco local
SccQueryChanges	Permite examinar el estado de los cambios de nombre (renombrar, adiciones y eliminaciones) para los archivos especificados
SccPopulateDirList	Permite el examen de directorios y archivos en la base de datos de control de versiones
SccAddFilesFromSCC	Agrega una lista especificada de archivos de la base de datos de control de versiones al proyecto actual
SccBackgroundGet	Realiza un "Get" silencioso de los archivos especificados (no se muestra ninguna interfaz de usuario)
SccGetUserOption	Permite el acceso a opciones específicas del usuario

Vea también

- [Introducción](#)
- [Novedades de la API del complemento de control de código fuente, versión 1.2](#)

Qué's Nuevo en la versión 1.2 de la API de complemento de Control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

En esta sección se destaca la funcionalidad avanzada de control de código fuente en la versión 1.2 de la API de complemento de Control de código fuente. A continuación se enumeran las nuevas características de la versión 1.2:

Cambios

- [Eliminación de los archivos de la aplicación de la aplicación de la aplicación de la aplicación](#) Explica cómo interactúa la API de complemento de Control de código fuente 1.2 con archivos temporales.
- [Aplicación de la configuración en varias conexiones](#) de proyecto Muestra cómo un complemento de control de código fuente puede usar una operación por lotes para ejecutar la misma operación de control de código fuente en varios proyectos o varios contextos de conexión.
- [Creación de carpetas de contenedor escuayera para soluciones](#) Explica cómo la versión 1.2 de la API de complemento de Control de código fuente permite a un usuario especificar un único destino de control de código fuente raíz para todos los proyectos web dentro de la solución.
- [Comparación opcional de](#) la carpeta de proyecto local con el almacén de Control de código fuente Explica cómo la API de complemento de Control de código fuente 1.2 compara las versiones de los archivos.
- [Eliminación de la información de Control de código fuente de . Proj y . Archivos Sln](#) Explica cómo la versión 1.2 de la API de complemento de Control de código fuente almacena información sobre proyectos y soluciones.

Vea también

- [Introducción](#)

Eliminación de los archivos de la aplicación de la aplicación de la aplicación

06/04/2020 • 3 minutes to read • [Edit Online](#)

En la API de complemento de control de código fuente 1.2, los archivos *de la aplicación* se han reemplazado por indicadores de capacidad y nuevas funciones que detectan si un complemento de control de código fuente admite el archivo *MSSCCPRJ* y las desprotecciones compartidas.

Archivos de la página de trabajo

Visual Studio .NET 2003 creó archivos temporales con el prefijo *.SAK*. Estos archivos se utilizan para determinar si un complemento de control de código fuente admite:

- El archivo *MSSCCPRJ.SCC*.
- Múltiples desprotecciones (compartidas).

Para los complementos que admiten funciones avanzadas proporcionadas en la API de complemento de Control de código fuente 1.2, el IDE puede detectar estas capacidades sin crear los archivos temporales mediante el uso de nuevas capacidades, indicadores y funciones, que se detallan en las secciones siguientes.

Nuevos indicadores de capacidad

`SCC_CAP_SCCFILE`

`SCC_CAP_MULTICHECKOUT`

Nuevas funciones

- [SccWillCreateSccFile](#)
- [SccIsMultiCheckoutEnabled](#)

Si un complemento de control de código fuente admite varias `SCC_CAP_MULTICHECKOUT` desprotecciones `SccIsMultiCheckOutEnabled` (compartidas), declara la capacidad e implementa la función. Esta función se llama siempre que se produce una operación de desprotección en cualquiera de los proyectos controlados por código fuente.

Si un complemento de control de código fuente admite la creación y el uso de `SCC_CAP_SCCFILE` un archivo *MSSCCPRJ.SCC*, declara la capacidad e implementa [SccWillCreateSccFile](#). Esta función se llama con una lista de archivos. La función devuelve `TRUE` or `FALSE` para cada archivo para indicar si Visual Studio debe usar un archivo *MSSCCPRJ.SCC* para él. Si el complemento de control de código fuente decide no admitir estas nuevas capacidades y funciones, puede usar la siguiente clave del Registro para deshabilitar la creación de estos archivos:

[HKEY_CURRENT_USER,Software,Microsoft,VisualStudio,8.0,SourceControl]

DoNotCreateTemporaryFilesInSourceControl = dword:00000001

NOTE

Si esta clave del Registro se establece en *dword:00000000*, equivale a que la clave no existe y Visual Studio sigue intentando crear los archivos temporales. Sin embargo, si la clave del Registro se establece en *dword:00000001*, Visual Studio no intenta crear los archivos temporales. En su lugar, se supone que el complemento de control de código fuente no admite el archivo *MSSCCPRJ.SCC* y no admite desprotecciones compartidas.

Vea también

- [Novedades de la API de complemento de Control de código fuente versión 1.2](#)

Aplicación de la configuración en varias conexiones de proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un complemento de control de código fuente creado con la versión 1.2 de la API de complemento de Control de código fuente, puede usar una operación por lotes para ejecutar la misma operación de control de código fuente en varios proyectos o varios contextos de conexión. Los lotes se pueden usar para eliminar cuadros de diálogo redundantes por proyecto de la experiencia del usuario.

Si un usuario selecciona varios elementos que pertenecen a más de una conexión en un complemento de control de código fuente creado mediante la versión 1.1 de la API de complemento de Control de código fuente (por ejemplo, dos proyectos web en equipos de recurso compartido de archivos diferentes) y los desprotege, el usuario ve el mismo cuadro de diálogo repetidamente. Este escenario se produce incluso si el usuario hace clic en la casilla de verificación **Aplicar a todos** en el cuadro de diálogo, porque el IDE restablece su estado para cada contexto de conexión.

Nueva bandera de capacidad

La `SccBeginBatch` función `SCC_CAP_BATCH` establece el indicador para indicar que una operación por lotes está en curso.

Nuevas funciones

Las siguientes funciones nuevas admiten la operación por lotes:

- [SccBeginBatch](#)
- [SccEndBatch](#)

La `SccBeginBatch` función inicia un grupo de operaciones de control de código fuente. La `SccEndBatch` función cierra el grupo. Es posible que los grupos no estén anidados.

Vea también

- [Novedades de la API de complemento de Control de código fuente versión 1.2](#)

Crear carpetas de contenedor principales para soluciones

06/04/2020 • 4 minutes to read • [Edit Online](#)

En la versión 1.2 de la API de complemento de Control de código fuente, un usuario puede especificar un único destino de control de código fuente raíz para todos los proyectos web dentro de la solución. Esta raíz única se llama una raíz súper unificada (SUR).

En la versión 1.1 de la API de complemento de Control de código fuente, si el usuario agregó una solución multiproyecto al control de código fuente, se le pidió al usuario que especificara un destino de control de código fuente para cada proyecto web.

Nuevos indicadores de capacidad

SCC_CAP_CREATESUBPROJECT

SCC_CAP_GETPARENTPROJECT

Nuevas funciones

- [SccCreateSubProject](#)
- [SccGetParentProjectPath](#)

El Visual Studio IDE casi siempre crea una carpeta SUR al agregar una solución al control de código fuente. Concretamente, lo hace en los siguientes casos:

- El proyecto es un proyecto web de recurso compartido de archivos.
- Hay diferentes unidades para el proyecto y el archivo de solución.
- Hay diferentes recursos compartidos para el proyecto y el archivo de solución.
- Los proyectos se agregaron por separado (en una solución controlada por código fuente).

En Visual Studio, se sugiere que el nombre de la carpeta SUR sea el mismo que el nombre de la solución sin la extensión. En la tabla siguiente se resume el comportamiento de las dos versiones.

CARACTERÍSTICA	SOURCE CONTROL PLUG-IN API VERSIÓN 1.1	SOURCE CONTROL PLUG-IN API VERSIÓN 1.2
Añadir solución a SCC	SccInitialize() SccGetProjPath() SccGetProjPath() SccOpenProject()	SccInitialize() SccGetProjPath() SccCreateSubProject() SccCreateSubProject() SccOpenProject()

CARACTERÍSTICA	SOURCE CONTROL PLUG-IN API VERSIÓN 1.1	SOURCE CONTROL PLUG-IN API VERSIÓN 1.2
Agregar proyecto a la solución controlada por código fuente	SccGetProjPath() OpenProject()	SccGetParentProjectPath() SccOpenProject() Nota: Visual Studio supone que una solución es un elemento secundario directo de SUR.

Ejemplos

En la tabla siguiente se enumeran dos ejemplos. En ambos casos, se solicita al Visual Studio usuario una ubicación de destino para la solución bajo control de código fuente hasta que se especifique el *user_choice* como destino. Cuando se especifica el *user_choice*, la solución y dos proyectos se agregan sin solicitar al usuario destinos de control de código fuente.

LA SOLUCIÓN CONTIENE	EN UBICACIONES DE DISCO	ESTRUCTURA PREDETERMINADA DE LA BASE DE DATOS
<i>sln1.sln</i>	<i>C:-Soluciones-sln1</i>	<i>\$/<user_choice>/sln1</i>
Web1	<i>C:-Inetpub-wwwroot-Web1</i>	<i>\$/<user_choice>/C/Web1</i>
Web2	<i>\servidor" (servidor) y "wwwroot".</i>	<i>\$/<>/Web2 user_choice 2</i>
<i>sln1.sln</i>	<i>C:-Soluciones-sln1</i>	<i>\$/<user_choice>/sln1</i>
Web1	<i>D:-Inetpub-wwwroot-Web1</i>	<i>\$/<> user_choice/D/web1</i>
Win1	<i>C:-soluciones-sln1-Win1</i>	<i>\$/<user_choice>/sln1/win1</i>

La carpeta SUR y las subcarpetas se crean independientemente de si la operación se cancela o se produce un error debido a un error. No se eliminan automáticamente en condiciones de cancelación o error.

Visual Studio de forma predeterminada, el comportamiento de la versión `SCC_CAP_CREATESUBPROJECT` 1.1 si el complemento de control de código fuente no devuelve y `SCC_CAP_GETPARENTPROJECT` los indicadores de capacidad. Además, Visual Studio los usuarios pueden optar por volver al comportamiento de la versión 1.1 estableciendo el valor de la siguiente clave en *dword:00000001*:

[HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8.0, SourceControl]

DoNotCreateSolutionRootFolderInSourceControl = *dword:00000001*

Vea también

- [Novedades de la API de complemento de Control de código fuente versión 1.2](#)

Comparación opcional de la carpeta de proyecto local con el almacén de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

En Source Control Plug-in API 1.2, la comparación entre la carpeta del proyecto local y el control de código fuente se realiza mediante las funciones [SccDirQueryInfo](#) y [SccDirDiff](#).

En el **Explorador** de soluciones, si se selecciona una carpeta en lugar de un archivo individual, el menú contextual **Comparar versiones** invoca el nuevo [SccDirQueryInfo](#) y [SccDirDiff](#) en el complemento de control de código fuente.

Nuevos indicadores de capacidad

`SCC_CAP_DIRECTORYDIFF`

`SCC_CAP_DIRECTORYCHECKOUT`

Funciones nuevas

- [SccDirDiff](#)
- [SccDirQueryInfo](#)

Se `SccDirQueryInfo` llama a `SccDirDiff` la función antes para determinar si el directorio de trabajo está controlado por código fuente. La `SccDirDiff` función muestra las diferencias entre el directorio local actual y la carpeta de control de código fuente correspondiente. Este comando solicita al complemento de control de código fuente que muestre la lista de cambios en el directorio. Un complemento de control de código fuente proporciona su propia interfaz de usuario para mostrar las diferencias.

NOTE

Esta función utiliza los mismos indicadores de comando [que SccDiff](#). Como proveedor de complementos de control de código fuente, puede optar por no admitir la operación "diff rápida" para directorios.

Vea también

- [Novedades de la API del complemento de control de código fuente, versión 1.2](#)

Eliminación de la información de control de código fuente de archivos .Proj y .Sln

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la versión 1.2 de la API del complemento de control de código fuente, la información SCC se almacena en un MSSCCPRJ. SCC. La ventaja del MSSCCPRJ. SCC es que la información de SCC no está controlada por código fuente, como en los archivos .proj y .sln.

Cambios en la versión 1.2

En los complementos de control de código fuente que se basan en la versión 1.1 de la API de complemento de Control de código fuente, la información sobre el control de código fuente se almacena en los archivos de proyecto (.proj) y solución (.sln). AuxPath especifica la ubicación de la base de datos de la información del control de código fuente y ProjName especifica la ubicación específica dentro de la base de datos. Este comportamiento puede causar problemas después de las operaciones de bifurcación, bifurcación o copia porque ProjName normalmente no sería válido después de cualquiera de estas operaciones.

En la versión 1.1 de la API de complemento de Control de código fuente, el IDE utilizaba archivos de tipo SAK para detectar si un complemento admitía MSSCCPRJ. Método SCC de almacenamiento de información de control de código fuente. La versión 1.2 de la API del complemento de control de código fuente proporciona una nueva capacidad para detectar el soporte para MSSCCPRJ. SCC sin utilizar un archivo .SAK. Para obtener más información, consulte [Eliminación de archivos de sak](#).

Vea también

- [Novedades de la API del complemento de control de código fuente, versión 1.2](#)

Arquitectura de complemento de control de código fuente

06/04/2020 • 7 minutes to read • [Edit Online](#)

Puede agregar compatibilidad con Visual Studio el control de código fuente al entorno de desarrollo integrado (IDE) implementando y adjuntando un complemento de control de código fuente. El IDE se conecta al complemento de control de código fuente a través de la API de complemento de Control de código fuente bien definida. El IDE expone las características de control de versiones del sistema de control de código fuente proporcionando una interfaz de usuario (UI) que consta de barras de herramientas y comandos de menú. El complemento de control de código fuente implementa la funcionalidad de control de código fuente.

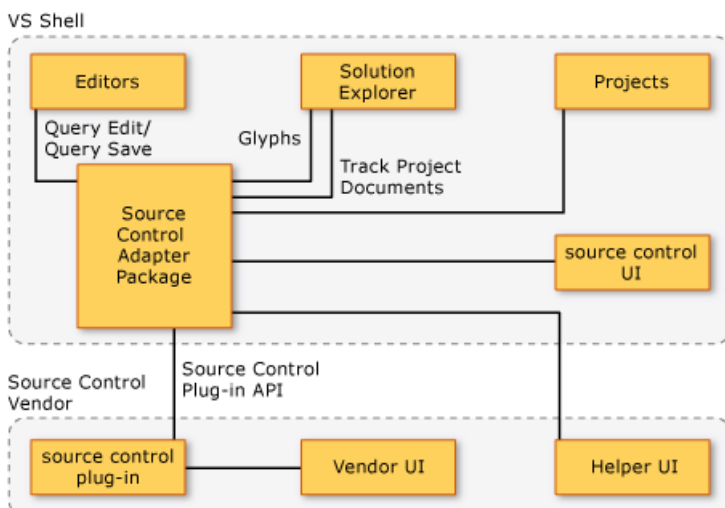
Recursos de complementos de Control de código fuente

El complemento Control de código fuente proporciona recursos para ayudar Visual Studio a crear y conectar la aplicación de control de versiones al IDE. El complemento de Control de código fuente contiene la especificación de API que debe implementar Visual Studio un complemento de control de código fuente para que se pueda integrar en el IDE. También contiene un ejemplo de código (escrito en C++) que implementa un complemento de control de código fuente de esqueleto que demuestra la implementación de funciones esenciales compatibles con la API de complemento de Control de código fuente.

La especificación de API de complemento de Control de código fuente le permite aprovechar cualquier sistema de control de código fuente de su elección si crea un archivo DLL de control de código fuente con el conjunto necesario de funciones implementadas de acuerdo con la API de complemento de Control de código fuente.

Componentes

El paquete de adaptador de control de código fuente en el diagrama es el componente del IDE que traduce la solicitud del usuario para una operación de control de código fuente en una llamada de función admitida por el complemento de control de código fuente. Para que esto suceda, el IDE y el complemento de control de código fuente deben tener un cuadro de diálogo eficaz que pase información entre el IDE y el complemento. Para que este diálogo tenga lugar, ambos deben hablar el mismo idioma. La API de complemento de Control de código fuente que se describe en esta documentación es el vocabulario común para este intercambio.



de arquitectura de control de código fuente

Diagrama de arquitectura que muestra la interacción entre VS y el complemento de control de código fuente

Como se muestra en Visual Studio el diagrama de arquitectura, el shell, etiquetado como shell VS en el diagrama,

hospeda los proyectos de trabajo del usuario y los componentes asociados, como los editores y el Explorador de soluciones. El paquete de adaptador de control de código fuente controla la interacción entre el IDE y el complemento de control de código fuente. El paquete de adaptador de control de código fuente proporciona su propia interfaz de usuario de control de código fuente. Es la interfaz de usuario de nivel superior con la que interactúa el usuario para iniciar y definir el ámbito de una operación de control de código fuente.

El complemento de control de código fuente puede tener su propia interfaz de usuario, que puede constar de dos partes como se muestra en la figura. El cuadro etiquetado "User UI" representa los elementos de interfaz de usuario personalizados que usted, como creador de complementos de control de código fuente, proporciona. Estos se muestran directamente por el complemento de control de código fuente cuando el usuario invoca una operación de control de código fuente avanzada. El cuadro etiquetado "Helper UI" es un conjunto de características de interfaz de usuario de complemento de control de código fuente que se invocan indirectamente a través del IDE. El complemento de control de código fuente pasa mensajes relacionados con la interfaz de usuario al IDE a través de funciones de devolución de llamada especiales proporcionadas por el IDE. La interfaz de usuario auxiliar facilita una integración más fluida con el IDE (a menudo mediante el uso de un botón **avanzado**) y, por lo tanto, proporciona una experiencia de usuario final más unificada.

Un complemento de control de código Visual Studio fuente no puede realizar cambios en el shell y, en consecuencia, en el paquete de adaptador de control de código fuente o en la interfaz de usuario de control de código fuente proporcionada por el IDE. Debe aprovechar al máximo la flexibilidad ofrecida a través de la implementación de las diversas funciones de API de complemento de Control de código fuente que contribuyen a una experiencia integrada para el usuario final. La sección de referencia de la documentación de la API de complemento de Control de código fuente incluye información sobre algunas capacidades avanzadas del complemento de control de código fuente. Para aprovechar estas características, el complemento de control de código fuente debe declarar sus capacidades avanzadas al IDE durante la inicialización y debe implementar funciones avanzadas específicas para cada capacidad.

Vea también

- [Complementos de control de código fuente](#)
- [Glosario](#)
- [Creación de un complemento de control de código fuente](#)

Guía de pruebas para los complementos de control de código fuente

06/04/2020 • 5 minutes to read • [Edit Online](#)

En esta sección se proporcionan instrucciones para Visual Studio probar el complemento de control de código fuente con . Se proporciona una amplia visión general de las áreas de prueba más comunes, así como algunas de las áreas más complejas que pueden ser problemáticas. Esta descripción general no pretende ser una lista exhaustiva de casos de prueba.

NOTE

Algunas correcciones de Visual Studio errores y mejoras en el IDE más reciente pueden descubrir problemas con Visual Studio los complementos de control de código fuente existentes que no se encontraban anteriormente al usar versiones anteriores de . Se recomienda encarecidamente que pruebe el complemento de control de código fuente existente para las áreas enumeradas en esta Visual Studio sección, incluso si no se han realizado cambios en el complemento desde la versión anterior de .

Preparación común

Se requiere Visual Studio una máquina con el complemento de control de código fuente de destino instalado. Una segunda máquina configurada de forma similar se puede utilizar para algunas de las pruebas De abierto desde control de código fuente.

Definición de Términos

Para los fines de esta guía de prueba, utilice las siguientes definiciones de términos:

Proyecto de cliente Cualquier Visual Studio tipo de proyecto disponible Visual Basic que Visual C# admita la integración de controles de código fuente (por ejemplo, , , o Visual C++).

Proyecto web Hay cuatro tipos de proyectos web: sistema de archivos, IIS local, sitios remotos y FTP.

- Los proyectos del sistema de archivos se crean en una ruta de acceso local, pero no requieren que Internet Information Services (IIS) se instale a medida que se accede a ellos internamente a través de una ruta de acceso UNC y se pueden colocar bajo control de código fuente desde dentro del IDE, al igual que los proyectos de cliente.
- Los proyectos locales de IIS funcionan con IIS que está instalado en el mismo equipo y se tiene acceso con una dirección URL que apunta al equipo local.
- Los proyectos de sitios remotos también se crean en servicios IIS, pero se colocan Visual Studio bajo control de código fuente en el equipo servidor IIS y no desde dentro del IDE.
- Se accede a los proyectos FTP a través de un servidor FTP remoto, pero no se pueden colocar bajo control de código fuente.

Inscripción Otro término para la solución o proyecto bajo control de código fuente.

Almacén de versiones La base de datos de control de código fuente a la que se tiene acceso a través de la API de complemento de Control de código fuente.

Zonas de prueba cubiertas en esta sección

- [Zona de prueba 1: Añadir a/Abrir desde el Control de código fuente](#)
 - Caso 1a: Agregar solución al control de código fuente
 - Caso 1b: Solución abierta desde el control de código fuente
 - Caso 1c: Agregar solución desde el Control de código fuente
- [Área de prueba 2: obtener de control de código fuente](#)
- [Zona de prueba 3: Desproteger/Deshacer pago](#)
 - Caso 3: Desproteger/Deshacer pago
 - Caso 3a: Salida
 - Caso 3b: Pago desconectado
 - Caso 3c: Edición de consultas/guardado de consultas (QEQS)
 - Caso 3d: Silent Checkout
 - Caso 3e: Deshacer pago
- [Área de prueba 4: insertar en el repositorio](#)
 - Caso 4a: Artículos modificados
 - Caso 4b: Adición de archivos
 - Caso 4c: Adición de proyectos
- [Área de prueba 5: cambiar control de código fuente](#)
 - Caso 5a: Enlazar
 - Caso 5b: Desenlazar
 - Caso 5c: Reenlazar
- [Área de prueba 6: eliminar](#)
- [Área de prueba 7: compartir](#)
- [Área de prueba 8: cambio de complementos](#)
 - Caso 8a: Cambio automático
 - Caso 8b: Cambio basado en soluciones

Vea también

- [Complementos de control de código fuente](#)

Zona de prueba 1: Añadir a/Abrir desde el Control de código fuente

06/04/2020 • 13 minutes to read • [Edit Online](#)

Este área de prueba del complemento de control de código fuente cubre la colocación de soluciones o proyectos bajo control de código fuente y la recuperación de ellos del control de código fuente.

Acceso al menú de comandos

En Visual Studio los casos de prueba se utilizan las siguientes rutas de menú del entorno de desarrollo integrado:

- Para Visual SourceSafe, abrir desde el control de código fuente: **Archivo, Abrir, Solución de proyecto**; mirar en Visual SourceSafe la ubicación.
- Para otros complementos de control de código fuente, abierto desde el control de código fuente: **Archivo, Control de código fuente, Abrir desde Control de código fuente**.
- Agregar al control de código fuente: **Archivo, Control de código fuente, Agregar solución al archivo** de control de código fuente, **Control de código fuente, Agregar proyectos seleccionados al Control de código fuente**.
- Menú contextual (proyecto/solución), **Agregar solución al control de código fuente**.
- Agregar desde el control de código fuente: **Archivo, Control de código fuente, Agregar proyecto desde Control de código fuente**.
- Para Visual SourceSafe, add from source control también está disponible desde **Archivo, Agregar, Proyecto existente**; mirar en Visual SourceSafe la ubicación.

NOTE

En esta prueba se puede utilizar una ruta de acceso de un archivo local o un IIS local (servidor web).

Comportamiento esperado

- Para cada tipo de proyecto admitido, un usuario debe poder "Agregar a" y "Abrir desde" Control de código fuente.
- Cuando se agrega un proyecto al <Control de código fuente, se crea un archivo *ProjectName*.vspssc correspondiente (archivo de sugerencia de proyecto). Contiene la lista de archivos de exclusión y la información de conexión. No elimine este archivo porque contiene información específica del proyecto.
- Cuando se agrega una solución <al control de código fuente, se crea un archivo *SolutionName*.vsscc (triple S) correspondiente. El archivo de texto contiene información de conexión y una lista de archivos de exclusión, similar al archivo de sugerencia del proyecto. Este archivo es temporal y solo existe en la base de datos de control de código fuente.
- Cuando se abre una solución <desde el control de código fuente, un *SolutionName*.vsscc (doble S) archivo que existe solo en la base de datos de control de código fuente, se crea localmente en un archivo temporal. Este archivo contiene la ruta de acceso de la carpeta de conexión de la solución al archivo de solución. Este archivo es temporal y la copia local se elimina cuando se ha completado la operación "Abrir desde control

de código fuente".

- Después de agregar un proyecto al control de código fuente, puede realizar cualquier acción de control de código fuente en él (Desproteger, Obtener, etc.).

Casos de prueba

A continuación se muestran casos de prueba específicos para el área de prueba Agregar a/Abrir desde el control de código fuente.

Caso 1a: Agregar solución al control de código fuente

Este caso de prueba se centra en agregar soluciones al control de código fuente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Agregar solución que contenga un proyecto de cliente al control de código fuente	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente (File, Source Control, Add Solution to Source Control).	Solución/Proyecto se agregó al control de código fuente.
Agregar solución que contenga un sistema de archivos o un proyecto web IIS local al control de código fuente	1. Cree un sistema de archivos o un proyecto web IIS local (utilice el botón Examinar para apuntar a la ubicación del proyecto; la ruta de acceso determina qué tipo de proyecto Web se crea). 2. Agregue la solución al control de código fuente (File, Source Control, Add Solution to Source Control).	Solución/Proyecto se agregó al control de código fuente.
Agregar solución que contenga un proyecto Web de sitio remoto al control de código fuente	1. Cree un proyecto Web de sitio remoto. 2. Agregue la solución al control de código fuente (File, Source Control, Add Solution to Source Control). 3. Hacer clic OK en FrontPage Access cuadro de diálogo de advertencia.	Solución se agregó al control de código fuente. Proyecto de sitio remoto NO está bajo control de código fuente. (Los proyectos de sitio remoto deben controlarse desde su propio servidor IIS.)
Agregue una única solución de proyecto al control de código fuente mediante Agregar proyectos seleccionados al control de código fuente .	1. Cree una única solución de proyecto. 2. Agregue solo solución al control de código fuente como una selección (Archivo, Control de código fuente, Agregar proyectos seleccionados al Control de código fuente). Si este paso se realiza correctamente, continúe con el paso siguiente. 3. Agregue el proyecto al control de código fuente como selección (Archivo, Control de código fuente, Agregar proyectos seleccionados al Control de código fuente). 4. Haga clic en Sí para agregar el proyecto a la misma ubicación. 5. Hacer clic Desproteger en Desproteger para editar cuadro de diálogo.	Result from Step 2: El proyecto y todos los archivos del proyecto tienen un indicador de control de código fuente desprotegido y una información sobre herramientas muestra "No bajo control de código fuente". Result from Step 5: El proyecto y el archivo de solución están en la misma carpeta en el control de código fuente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Cancelar la adición de una solución al control de código fuente	1. Cree una única solución de proyecto. 2. Intente agregar proyecto y solución al control de código fuente. Si este paso se realiza correctamente, continúe con el paso siguiente. 3. Cancelar después de que usted está en el sistema de control de código fuente.	Result from Step 2: El cuadro de diálogo Establecer control de código fuente de ubicación del proyecto solo aparece una vez. Result from Step 3: Agregar proyecto cancelado, proyecto/solución NO está bajo control de código fuente y todos los menús Agregar al control de código fuente todavía están disponibles.

Caso 1b. Solución abierta desde el Control de código fuente

Este caso de prueba se centra en la apertura de soluciones desde el control de código fuente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Abra una solución que contenga un proyecto de cliente desde el control de código fuente	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente. 3. Cierre la solución. 4. Abra la solución desde el control de código fuente a una nueva ubicación.	Solución/Proyecto abierto desde el control de código fuente.
Abra una solución que contenga un proyecto web local o IIS desde el control de código fuente	1. Cree un proyecto web local o IIS. 2. Agregue la solución al control de código fuente. 3. Cierre la solución. 4. Abra la solución desde el control de código fuente a una nueva ubicación.	Solución/Proyecto abierto desde el control de código fuente.
Abra una solución que contenga un proyecto Web de sitio remoto desde el control de código fuente	1. Cree un proyecto Web de sitio remoto. 2. Agregue la solución al control de código fuente. Si este paso se realiza correctamente, continúe con el paso siguiente. 3. Cierre la solución. 4. Abra la solución desde el control de código fuente a una nueva ubicación.	Result from Step 2: Web del sitio remoto NO está bajo control de código fuente. Result from Step 4: Solución abierta desde el control de código fuente. Se carga el proyecto de sitio remoto, pero NO está bajo control de código fuente.

Caso 1c: Agregar solución desde el Control de código fuente

Este caso de prueba se centra en agregar soluciones desde el control de código fuente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Añadir a la solución vacía: una solución de proyecto único	1. Cree una única solución de proyecto. 2. Agregue la solución al control de código fuente. 3. Cierre la solución. 4. Cree una segunda solución vacía. 5. Agregue la solución previamente controlada desde el control de código fuente (File, Source Control, Add Project from Source Control).	El proyecto agregado aparece en el Explorador de soluciones y está protegido.
Añadir a la solución con un solo proyecto — proyecto único	1. Cree una solución con un solo proyecto. 2. Agregue la solución al control de código fuente. 3. Cierre la solución. 4. Cree una segunda solución vacía. 5. Agregue la solución previamente controlada desde el control de código fuente (File, Source Control, Add Project from Source Control).	El proyecto agregado aparece en el Explorador de soluciones y está protegido.
Añadir a la solución: solución añadida al control de código fuente por selección	1. Cree una solución con un proyecto. 2. Agregue solo solución al control de código fuente como selección. Si este paso se realiza correctamente, continúe con el paso siguiente. 3. Cierre la solución. 4. Cree una nueva solución. 5. Agregue la solución previamente controlada desde el control de código fuente (File, Source Control, Add Project from Source Control).	Result from Step 2: Project no está bajo control de código fuente. Result from Step 5: Si la primera solución tenía elementos de solución, no se pueden agregar desde el control de código fuente, por lo que no aparecen. El proyecto de la primera solución aparece como no disponible.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 2: obtener de control de código fuente

06/04/2020 • 5 minutes to read • [Edit Online](#)

Esta área de prueba cubre los casos de prueba para recuperar elementos del almacén de versiones mediante el comando Obtener. Estos casos de prueba se pueden aplicar tanto a proyectos locales como a proyectos web.

Acceso al menú de comandos

Las Visual Studio siguientes rutas de menú del entorno de desarrollo integrado se utilizan en los casos de prueba.

Obtener la última versión:

- Archivo, Control de código fuente, Obtener la versión más reciente.
- Archivo, Obtener la última versión.
- Menú contextual, Obtener la última versión.
- Obtener: Archivo, Control de código fuente, Obtener.

Comportamiento esperado

Obtener la última versión:

Realiza una recuperación silenciosa (sin interfaz de usuario) de la versión más reciente del elemento desde el almacén de versiones.

Get:

Muestra el cuadro de diálogo **Obtener** y permite al usuario realizar cambios en el conjunto de archivos que se recuperarán, así como modificar las opciones que afectan a cómo se recuperan los archivos.

Casos de prueba

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Obtener la última versión de un archivo que NO existe localmente	1. Cree un proyecto. 2. Agregue un elemento al proyecto. 3. Ponga el proyecto bajo control de código fuente. 4. Elimine la copia local del elemento. 5. Obtener la última versión del elemento (menú contextual, obtener la última versión).	El archivo de elemento se recupera localmente.
Obtener un archivo que NO existe localmente	1. Cree un proyecto. 2. Agregue un elemento al proyecto. 3. Ponga el proyecto bajo control de código fuente. 4. Elimine la copia local del elemento. 5. Obtenga el elemento (Archivo, Control de código fuente, Obtener <elemento>).	El archivo de elemento se recupera localmente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Obtener un archivo que ha sido desprotegido exclusivamente y modificado localmente	1. Cree un proyecto. 2. Agregue un elemento al proyecto. 3. Ponga el proyecto bajo control de código fuente. 4. Echa un vistazo al elemento del proyecto exclusivamente. 5. Modifique la copia local. 6. Obtener la última versión del elemento (Archivo, Obtener la última versión del <elemento>). Si este paso se realiza correctamente, continúe con el paso siguiente. 7. Hacer clic Reemplazar botón en el cuadro de diálogo de advertencia.	Resultado del paso 6 : El cuadro de diálogo Advertencia indica que el archivo está desprotegido. Resultado del paso 7: El archivo local modificado se sustituye por la versión original del almacén de versiones. El archivo es de lectura/escritura.
Obtener y reemplazar el archivo que se desprotege, comparte y modifica localmente	1. Cree un nuevo proyecto. 2. Agregue un elemento al proyecto. 3. Ponga el proyecto bajo control de código fuente. 4. Consulte el elemento del proyecto como compartido. 5. Modifique la copia local. 6. Obtener la última versión del elemento (Archivo, Obtener la última versión del <elemento>). Si este paso se realiza correctamente, continúe con el paso siguiente. 7. Haga clic en Reemplazar en el cuadro de diálogo de advertencia.	Resultado del paso 6: El cuadro de diálogo Advertencia indica que el archivo está desprotegido. Resultado del paso 7: El archivo local modificado se sustituye por la versión original del almacén de versiones. El archivo es de lectura/escritura.
Obtener un archivo que exista localmente, igual que la versión más reciente en el almacén de versiones	1. Cree un nuevo proyecto. 2. Agregue un elemento al proyecto. 3. Ponga el proyecto bajo control de código fuente. 4. Obtenga el elemento (Archivo, Control de código fuente, Obtener <elemento>).	El archivo local no cambia.
Obtenga una solución con un proyecto	1. Cree una solución con un proyecto. 2. Coloque la solución bajo control de código fuente. 3. Elimine todos los archivos de proyecto localmente. 4. Obtenga la solución (File, Source Control, Get).	Todos los archivos eliminados se restauran localmente.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Zona de prueba 3: Desproteger/Deshacer pago

06/04/2020 • 17 minutes to read • [Edit Online](#)

Este área de prueba del complemento de control de código fuente cubre la edición y la reversión de elementos del almacén de versiones a través de los comandos **Desproteger** y **Deshacer desprotección**.

Desproteger: Marca un elemento en el almacén de versiones como desprotegido, modifica la copia local para leer/escribir.

Deshacer desprotección: Marca un elemento en el almacén de versiones como protegido, revierte la copia local al estado antes de la desprotección (dependiendo de las opciones).

Acceso al menú de comandos

Las Visual Studio siguientes rutas de menú del entorno de desarrollo integrado se utilizan en los casos de prueba.

Modificar:

- Archivo, Control de código fuente, Desproteger.
- Archivo, Desproteger.
- Menú contextual, desproteger.
- Deshacer desprotección: Archivo, Control de código fuente, Deshacer desprotección.

Comportamiento esperado común

- Después de la operación de desprotección, los archivos de destino y/o carpetas se marcan como desprotegidos en el almacén de versiones.
- El almacén de versiones atribuye la desprotección al usuario correcto.
- La hora y la fecha de la compra son correctas (según la configuración del usuario).

Casos de prueba

Los siguientes son casos de prueba específicos para el área de prueba de pago/deshacer pago.

Caso 3a: Salida

Esta sección se centra en el funcionamiento del comando check-out.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Echa un vistazo a Exclusive (COE) un proyecto de cliente	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente. 3. Echa un vistazo a todo el proyecto exclusivamente (Archivo , Desproteger).	Se produce la salida.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Desproteger exclusivo (COE) un sistema de archivos o un proyecto web IIS local	1. Establezca Conexión del servidor web en Compartir archivos en Herramientas, Opciones, Proyectos, Configuración web. 2. Cree un proyecto Web. 3. Agregue la solución al control de código fuente. 4. Echa un vistazo a todo el proyecto exclusivamente (Archivo, Control de código fuente, Desproteger).	Se produce la salida.
Desproteger elementos de solución en una solución (nuevo método para controlar otros archivos)	1. Cree una solución en blanco. 2. Agregue la solución al control de código fuente. 3. Compruebe la solución. 4. Agregue varios elementos de solución. 5. Registre todos los elementos recién agregados. 6. Seleccione varios elementos de solución. 7. Echa un vistazo a los elementos seleccionados (Menú contextual, Desproteger).	Los archivos seleccionados se desprotegen.
Desproteger la versión local (si el plug-in bajo prueba admite esta característica)	1. Usuario 1: Crear un proyecto de cliente. 2. Usuario 1: Agregue la solución al control de código fuente. 3. Usuario 2: Abra la solución desde el control de código fuente a otra ubicación. 4. Usuario 2: Echa un vistazo a un archivo. 5. Usuario 2: Modifique el archivo. 6. Usuario 2: Registre el archivo. 7. Usuario 1: Desproteger la versión local del archivo (Desproteger versión local avanzada opción en Desproteger cuadro de diálogo).	La versión local del archivo está desprotegida. Las modificaciones del usuario 2 no se aplican al archivo de usuario 1.

Caso 3b: Desconexión Desprotección

Operar en modo desconectado permite a los usuarios algún nivel de compatibilidad continua con el control de código fuente cuando no están conectados directamente a un almacén de versiones. Esto se hace almacenando localmente en caché toda la información relevante sobre la solución y los proyectos dados de alta.

Las operaciones exclusivas de desprotección solo pueden producirse mientras están conectados al almacén de control de código fuente. Las operaciones de desprotección compartidas pueden producirse en cualquier momento, ya sea conectadas o desconectadas. Por lo tanto, cuando se desconecta del almacén de versiones, solo se habilita el comando **Desproteger compartido** (COS). Mientras está desconectado, **Deshacer desprotección** está deshabilitado porque la versión anterior no se puede recuperar para reemplazar los cambios realizados por el usuario.

Cuando el usuario se vuelve a conectar al almacén de versiones, se sincronizan los estados de desprotección de todas las soluciones y proyectos dados de alta. Esto hace las actualizaciones necesarias en el almacén para las desprotecciones que el usuario ha realizado. Una vez que se ha producido la sincronización, el usuario puede seguir trabajando de forma normal (conectado).

Comportamiento esperado

- No se puede utilizar el comando **Desproteger exclusivamente** mientras está desconectado del almacén de versiones.
- No se puede utilizar el comando **Deshacer desprotección** mientras está desconectado del almacén de versiones.
- El comando **Desproteger compartido** funciona.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Mientras está desconectado, desprotege un archivo y, a continuación, conéctate para sincronizar	1. Desconecte un proyecto controlado mediante el cuadro de diálogo Cambiar control de código fuente (Archivo, Control de código fuente, Cambiar control de código fuente). 2. Revise un archivo. 3. Hacer clic Desproteger (desconectado) en el cuadro de diálogo de advertencia. 4. Edite el archivo. 5. Conéctese mediante el cuadro de diálogo Cambiar control de código fuente. 6. Obtenga la última versión del archivo editado.	Comportamiento esperado común

Caso 3c: Edición de consultas/guardado de consultas (QEQS)

Se realiza un seguimiento de los elementos bajo control de código fuente para editarlos, cambiar los cambios y guardarlos para ayudar a los usuarios a administrar fácilmente sus archivos. Cuando se edita un elemento controlado que está "protegido", QEQS intercepta el intento de edición y pregunta al usuario si desea desproteger el archivo para editarlo. Dependiendo de **las herramientas, configuración** de opciones, el usuario se ve obligado a desproteger el archivo con el fin de editar o se le puede permitir editar una copia en la memoria y desproteger más tarde. Si la configuración **Herramientas, Opciones** del usuario no está configurada para mostrar el cuadro de diálogo de desprotección y simplemente desprotegerlo, a continuación, a medida que el usuario realiza su edición, el archivo se desprotege automáticamente, siempre que sea posible.

Comportamiento esperado

- Después de la operación de desprotección, los archivos de destino y/o carpetas se marcan como desprotegidos en el almacén de versiones.
- El almacén de versiones atribuye la desprotección al usuario correcto.
- La hora y la fecha de la salida son correctas (según la configuración del usuario).
- La copia local del archivo o carpeta de destino se puede escribir.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Editar archivo de texto que está protegido	1. Cree un nuevo proyecto que contenga un archivo de texto. 2. Agregue la solución al control de código fuente. 3. Establezca Herramientas, Opciones, Control de código fuente, Permitir que los archivos se editen mientras se desmarca el disco. 4. Establecer herramientas, Opciones, Control de código fuente, Solicitar desprotección en el cuando los archivos marcados se editan cuadro combinado. 5. Establecer herramientas, Opciones, Control de código fuente, Solicitar desprotección en el cuando los archivos marcados se guardan cuadro combinado. 6. Abra el archivo de texto en el editor, intente escribir texto nuevo en el archivo. Si este paso se realiza correctamente, continúe con el paso siguiente. 7. Haga clic en Cancelar en el cuadro de diálogo Desproteger para editar. Si este paso se realiza correctamente, continúe con el paso siguiente. 8. Establezca Herramientas, Opciones, Control de código fuente, Permitir que los archivos se editen mientras se marca el disco de solo lectura. 9. Abra el archivo de proyecto en el editor, intente escribir texto nuevo en el archivo. Si este paso se realiza correctamente, continúe con el paso siguiente. 10. Haga clic en Editar en el cuadro de diálogo Desproteger para editar. Si este paso se realiza correctamente, continúe con el paso siguiente. 11. Edite el archivo de texto e intente guardarlo.	Result of step 6: Aparece el cuadro de diálogo Editar para Editar. Result of step 7: El archivo no ha cambiado. Result of step 9: Aparece el cuadro de diálogo Editar para Editar. Result of step 10: Puede editar el archivo de proyecto en la memoria. Result of step 11: Al guardar, aparece el cuadro de diálogo Desproteger al guardar.
Edite un archivo de solución que esté protegido	Repita los pasos descritos en la prueba anterior, pero en lugar de modificar un archivo de texto, modifique la solución cambiando las propiedades de la solución.	Igual que la prueba anterior
Editar un archivo de proyecto que está protegido	Repita los pasos descritos en la prueba anterior, pero en lugar de modificar un archivo de texto, modifique el proyecto cambiando las propiedades del proyecto.	Igual que la prueba anterior.

Caso 3d: Silent Check Out

Esta subárea cubre los escenarios de desprotección en los que el cuadro de diálogo **Desproteger** no aparece por

usuario **Herramientas, Opciones, Configuración de Control de código fuente.**

Comportamiento esperado

- Después de la operación de desprotección, los archivos de destino y/o carpetas se marcan como desprotegidos en el almacén de versiones.
- El almacén de versiones atribuye la desprotección al usuario correcto.
- La hora y la fecha de la salida es correcta (según la configuración del usuario).
- La copia local del archivo o carpeta de destino se puede escribir.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Pago silencioso para un archivo	1. Establezca Herramientas, Opciones, Control de código fuente en archivos de desprotección automáticamente al editar . 2. Cree un nuevo proyecto con un archivo. 3. Agregue la solución al control de código fuente. 4. Echa un vistazo al archivo.	El archivo se desprotege en silencio (sin interfaz de usuario).
Pago silencioso para un proyecto	1. Establezca Herramientas, Opciones, Control de código fuente en archivos de desprotección automáticamente al editar . 2. Cree un nuevo proyecto. 3. Agregue la solución al control de código fuente. 4. Echa un vistazo al proyecto.	El archivo se desprotege en silencio (sin interfaz de usuario).

Caso 3e: Deshacer desprotección

Deshacer desprotección se utiliza para cancelar el estado de desprotección de un archivo y evitar la comprobación de los cambios realizados en el archivo.

Comportamiento esperado

- El valor predeterminado se basa en la configuración de la versión local de **desprotección** del usuario. Si el usuario ha elegido desproteger la versión local, el valor predeterminado para deshacer la desprotección es volver siempre a la versión desprotegida.
- Tras la aceptación de la opción deshacer, los iconos del **Explorador** de soluciones se actualizan para los archivos afectados y el elemento se quita de la ventana **Registros pendientes**.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Deshacer desprotección de un único archivo que está desprotegido exclusivamente	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente. 3. Echa un vistazo a un archivo exclusivamente. 4. Modifique el archivo. 5. Deshacer desprotección (Archivo, Control de código fuente, Deshacer desprotección).	Comportamiento esperado común.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Deshacer desprotección de un único archivo que está desprotegido Compartido	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente. 3. Echa un vistazo a un archivo compartido. 4. Modifique el archivo. 5. Deshacer desprotección (Archivo, Control de código fuente, Deshacer desprotección).	Comportamiento esperado común.
Deshacer desprotección de un proyecto después de agregar archivos al proyecto	1. Cree un nuevo proyecto y agréguelo al control de código fuente. 2. Echa un vistazo al proyecto. 3. Agregue un archivo al proyecto. 4. Deshacer la compra del proyecto.	El archivo agregado se quita del proyecto en el Explorador de soluciones. El proyecto ya no se desprotege.
Deshacer desprotección de un proyecto después de eliminar archivos del proyecto	1. Cree un nuevo proyecto y agréguelo al control de código fuente. 2. Echa un vistazo al proyecto. 3. Elimine un archivo del proyecto. 4. Deshacer la compra del proyecto.	El archivo eliminado aparece en el proyecto en el Explorador de soluciones. El proyecto ya no se desprotege.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 4: insertar en el repositorio

06/04/2020 • 5 minutes to read • [Edit Online](#)

Este área de prueba del complemento de control de código fuente cubre el envío de elementos actualizados al almacén de versiones mediante el comando **Registrar**.

Acceso al menú de comandos

Las Visual Studio siguientes rutas de menú del entorno de desarrollo integrado se utilizan en los casos de prueba.

Registrar:

Archivo, Control de código fuente, Registrar.

Archivo, Registrar.

Menú contextual, **Registrar**.

Comportamiento esperado común

- Los proyectos y archivos agregados a una solución o proyecto bajo control de código fuente aparecen en el cuadro de diálogo **Registrar** y en la ventana **Checkins pendientes**.
- Después de la protección, los elementos agregados aparecen en el control de código fuente.
- Después del check-in, los elementos actualizados se versionan correctamente en la tienda.

Casos de prueba

Los siguientes son casos de prueba específicos para el área de prueba de check-in.

Caso 4a: Artículos modificados

Describe el uso de la acción de protección para actualizar un archivo bajo control de código fuente que se ha modificado.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Modificar un archivo de texto que se ha desprotegido, proteger solo el archivo (cuadro de diálogo Registrar)	1. Cree un nuevo proyecto con un archivo de texto. 2. Agregue la solución al control de código fuente. 3. Desproteger y modificar el archivo de texto. 4. Regístrese a través del cuadro de diálogo Registrar (Archivo, Control de código fuente, Registrar).	Comportamiento esperado común.
Modificar un archivo de texto que se ha desprotegido, Registrar solo archivo (ventana Checkins pendientes)	1. Cree un nuevo proyecto con un archivo de texto. 2. Agregue la solución al control de código fuente. 3. Desproteger y modificar el archivo de texto. 4. Regístrese a través de la ventana Checkins pendientes.	Comportamiento esperado común.

Caso 4b: Adición de archivos

Al agregar un archivo a un proyecto o un elemento a una solución, el proyecto o la solución también debe cambiar. Por lo tanto, el archivo principal también se desprotege y debe protegerse para completar la adición.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Agregar un archivo de texto y registrar todo (cuadro de diálogo Registrar)	1. Cree un nuevo proyecto. 2. Agregue la solución al control de código fuente. 3. Agregue un archivo de texto al proyecto. 4. Acepte la salida del proyecto si se le solicita. 5. Seleccione la solución en el Explorador de soluciones. 6. Regístrese desde el cuadro de diálogo Registrar.	Comportamiento esperado común.
Añadir un archivo de texto y registrar todo (ventana Checkins pendientes)	1. Cree un nuevo proyecto. 2. Agregue la solución al control de código fuente. 3. Agregue un archivo de texto al proyecto. 4. Acepte la salida del proyecto si se le solicita. 5. Compruebe la solución desde la ventana Checkins pendientes.	Comportamiento esperado común

Caso 4c: Adición de proyectos

Al agregar un proyecto a una solución, la solución también debe cambiar. Por lo tanto, el archivo de solución también se desprotege y debe protegerse para completar la adición.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Agregar un proyecto a una solución en blanco bajo control de código fuente (cuadro de diálogo Registrar)	1. Cree una solución en blanco. 2. Agregue la solución al control de código fuente. 3. Agregue un nuevo proyecto. 4. Acepte la salida de la solución si se le solicita. 5. Regístrese desde el cuadro de diálogo Registrar.	Comportamiento esperado común.
Agregar un proyecto a una solución en blanco bajo control de código fuente (ventana Checkins pendientes)	1. Cree una solución en blanco. 2. Agregue la solución al control de código fuente. 3. Agregue un nuevo proyecto. 4. Acepte la salida de la solución si se le solicita. 5. Compruebe la solución desde la ventana Checkins pendientes.	Comportamiento esperado común.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 5: cambiar control de código fuente

06/04/2020 • 15 minutes to read • [Edit Online](#)

Este área de prueba del complemento de control de código fuente cubre el cambio del control de código fuente mediante el comando **Cambiar control de código fuente**.

El comando Change Source Control proporciona cuatro funciones básicas para el usuario:

- **Atar:**

Permite a un usuario establecer o restablecer un vínculo de control de código fuente entre una solución o proyecto y el almacén de versiones.

- **Desvincular:**

Quita un proyecto o solución del control de código fuente por conexión.

- **Conectar/Desconectar:**

Alterna el estado conectado o sin conexión de la solución controlada, que se cubre en el área 3. Para obtener más información, consulte Zona de [prueba 3: Desproteger/Deshacer desprotección](#).

Acceso al menú de comandos

La Visual Studio siguiente ruta de menú del entorno de desarrollo integrado se utiliza en los casos de prueba.

Cambiar control de código fuente: **Archivo, Control de código fuente, Cambiar control de código fuente**.

Casos de prueba

A continuación se muestran casos de prueba específicos para el área de prueba del comando **Cambiar control de código fuente**.

Caso 5a: Enlazar

Enlazar permite al usuario agregar información de control de código fuente a los proyectos y soluciones seleccionados. Normalmente se solicita al usuario que identifique un proyecto en el control de código fuente al que se van a agregar. El usuario no puede crear un nuevo proyecto en el control de código fuente como parte de esta operación (contraste con Agregar al control de código fuente).

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Enlazar a una ubicación vacía	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Abra el cuadro de diálogo Cambiar control de código fuente (Archivo, Control de código fuente, Cambiar control de código fuente). 4. Haga clic en Desvincular. 5. Acepte el cuadro de diálogo de advertencia si aparece. 6. Seleccione todos los elementos. 7. Haga clic en Enlazar. 8. Vaya a una ubicación vacía en un almacén de control de código fuente. 9. Hacer clic OK para cerrar el cuadro de diálogo Cambiar control de código fuente. 10. Haga clic en Continuar con estos enlaces en el cuadro de diálogo de confirmación. 11. Haga clic en Aceptar en el cuadro de diálogo de advertencia si aparece. 12. Compruebe todo. Si este paso se realiza correctamente, continúe con el paso siguiente. 13. Abra la solución desde el control de código fuente a una nueva ubicación.	Result from Step 12: La solución y el proyecto se enlazan y escriben en el nuevo destino en el almacén de versiones. Los archivos de solución y proyecto están protegidos. La jerarquía del proyecto de almacén de versiones coincide con la jerarquía de carpetas del proyecto en disco. Result from Step 13: Se descargan todos los elementos del proyecto.
Enlazar a la ubicación que está sincronizada con el cliente	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Cree un duplicado de la solución y el Visual SourceSafe proyecto en el almacén de versiones (Compartir y rama si usa). 4. Abra el cuadro de diálogo Cambiar control de código fuente (Archivo, Control de código fuente, Cambiar control de código fuente). 5. Desenlazar todo. 6. Hacer clic OK para cerrar Cambiar control de código fuente cuadro de diálogo. 7. Vuelva a abrir el cuadro de diálogo Cambiar control de código fuente. 8. Seleccione todo. 9. Haga clic en Enlazar. 10. Vaya a la ubicación ramificada de la solución y el proyecto (del paso 3) 11. Hacer clic OK para cerrar el cuadro de diálogo Cambiar control de código fuente. 12. Obtenga la última recursiva para todos los artículos.	El contenido del archivo después de la obtención es el mismo que antes de la obtención.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Enlazar a una ubicación que no está sincronizada con el cliente	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Cree un duplicado de la solución y el Visual SourceSafe proyecto en el almacén de versiones (Compartir y rama si usa). 4. Modifique los archivos del proyecto ramificado en el almacén de versiones. 5. Abra el cuadro de diálogo Cambiar control de código fuente (Archivo, Control de código fuente, Cambiar control de código fuente). 6. Desenlazar todo. 7. Hacer clic OK para cerrar Cambiar control de código fuente cuadro de diálogo. 8. Vuelva a abrir el cuadro de diálogo Cambiar control de código fuente. 9. Seleccione todo. 10. Haga clic en Enlazar. 11. Vaya a la ubicación ramificada para la solución y el proyecto. 12. Hacer clic OK para cerrar el cuadro de diálogo Cambiar control de código fuente. 13. Acepte el cuadro de diálogo Advertencia si aparece. 14. Obtener la última recursiva para todos los elementos.	Los archivos que se modificaron en el paso 4 también se modifican localmente.
Solución de enlace que nunca estuvo bajo control de código fuente	1. Cree una carpeta vacía en el control de código fuente. 2. Cree un proyecto de cliente. 3. Abra el cuadro de diálogo Cambiar control de código fuente (Archivo, Control de código fuente, Cambiar control de código fuente). 4. Enlazar la solución a la ubicación vacía en el control de código fuente. 5. Hacer clic OK para cerrar el cuadro de diálogo Cambiar control de código fuente. 6. Haga clic en Continuar con estos enlaces en el cuadro de diálogo de confirmación. 7. Hacer clic OK en el cuadro de diálogo de advertencia si aparece.	Solución se agrega al control de código fuente. La solución y el proyecto están desprotegidos.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Cancelar enlace	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Abra el cuadro de diálogo Cambiar control de código fuente. 4. Desenlazar todo. 5. Hacer clic OK para cerrar el cuadro de diálogo. Si este paso se realiza correctamente, continúe con el paso siguiente. 6. Vuelva a abrir el cuadro de diálogo Cambiar control de código fuente. 7. Enlazar a una ubicación no relacionada. 8. Haga clic en Cancelar.	Result from Step 5: La solución ya no está bajo control de código fuente Result from Step 8: La solución todavía NO está bajo control de código fuente.

Caso 5b: Desenlazar

Desenlazar quita la información de control de código fuente de los proyectos y su solución. Los proyectos y la solución afectados se basan en una combinación de selección de usuario y cómo se agregaron los elementos al control de código fuente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Desenlazar solución que contiene un sistema de archivos o un proyecto web IIS local y un proyecto de cliente	1. Cree un sistema de archivos o un proyecto web IIS local. 2. Agregue la solución al control de código fuente. 3. Agregue un nuevo proyecto de cliente a la solución. 4. Acepte La desprotección de la solución si se le solicita. 5. Abra el cuadro de diálogo Cambiar control de código fuente. 6. Haga clic en Desvincular. 7. Hacer clic OK para cerrar el cuadro de diálogo. 8. Intente comprobar la solución, proyecto, elementos de solución, elementos del proyecto.	La solución y los proyectos NO están bajo control de código fuente. Los comandos del menú Control de código fuente no aparecen.
Cancelar desvincular	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Abra el cuadro de diálogo Cambiar control de código fuente. 4. Haga clic en Desenlazar todo. 5. Haga clic en Cancelar.	La solución está bajo control de código fuente.

Caso 5c: Reenlazar

Rebind es simplemente una combinación de desenlazar y enlazar: el proceso de volver a enlazar un proyecto o solución que anteriormente estaba bajo control de código fuente y no estaba enlazado.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Vuelva a enlazar la solución y los proyectos sin cerrar el cuadro de diálogo Cambiar control de código fuente	1. Cree un proyecto. 2. Agregue la solución al control de código fuente. 3. Abra el cuadro de diálogo Cambiar control de código fuente. 4. Haga clic en Desvincular. 5. Seleccione todas las filas. 6. Haga clic en Enlazar. 7. Hacer clic OK para cerrar el cuadro de diálogo Cambiar control de código fuente. 8. Acepte el pago si se le solicita.	La solución y el proyecto están bajo control de código fuente.
Volver a enlazar el proyecto solo sin cerrar el cuadro de diálogo Cambiar control de código fuente	1. Cree un proyecto. 2. Agregue solo el proyecto al control de código fuente mediante (Control de código fuente de > de archivo > Agregar proyectos seleccionados al control de código fuente. 3. Abra el cuadro de diálogo Cambiar control de código fuente. 4. Desenlazar sólo el proyecto. 5. Enlazar sólo el proyecto.	La solución permanece incontrolada. El proyecto permanece controlado.
Vuelva a enlazar la solución solo sin cerrar el cuadro de diálogo Cambiar control de código fuente	1. Cree un proyecto. 2. Agregue solo la solución al control de código fuente mediante (File, Source Control, Add Selected Projects to Source Control). 3. Abra el cuadro de diálogo Cambiar control de código fuente. 4. Desenlazar solo la solución (No cerrar Cambiar control de código fuente cuadro de diálogo.) 5. Enlazar sólo la solución. 6. Hacer clic OK para cerrar el cuadro de diálogo. 7. Echa un vistazo a los elementos de solución y solución (si los hay.)	La solución permanece controlada. El proyecto permanece incontrolado.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Reenlazar solución/proyecto solo cuando se encuentra en el mismo directorio	1. Cree un proyecto. 2. Agregue solo el proyecto al control de código fuente mediante (Archivo, Control de código fuente, Agregar proyectos seleccionados al Control de código fuente. 3. Cierre la solución. 4. Cree una nueva solución con al menos dos proyectos. 5. Agregue la solución al control de código fuente. 6. Agregue el proyecto creado en el paso 1 desde el control de código fuente. 7. Acepte la compra de la solución si se le solicita. 8. Compruebe toda la solución. 9. Abra el cuadro de diálogo Cambiar control de código fuente. 10. Seleccione el proyecto agregado (desde el paso 6) y haga clic en Desvincular. 11. Hacer clic OK para cerrar el cuadro de diálogo. 12. Acepte la compra si se le solicita. 13. Vuelva a abrir el cuadro de diálogo Cambiar control de código fuente. 14. Seleccione el proyecto agregado (desde el paso 6) y haga clic en Enlazar. 15. Seleccione la ubicación original.	La solución y los proyectos permanecen controlados.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 6: eliminar

06/04/2020 • 4 minutes to read • [Edit Online](#)

Este área de prueba del complemento de control de código fuente cubre las acciones de eliminación.

El control de código fuente responde a las acciones de eliminación en el Explorador de **soluciones**.

A continuación se muestra una lista de elementos que se pueden eliminar:

- Archivos
- Carpetas
- proyecto

Según el tipo de proyecto, es posible que tenga la opción **Quitar** el proyecto (deja los archivos en el disco) o **Eliminar** el proyecto (elimina los archivos en el disco). Cualquiera de las acciones quita el proyecto o el elemento del Explorador de **soluciones**.

Comportamiento esperado

El comportamiento esperado para los casos de prueba en el área de prueba de eliminación es:

- El elemento eliminado ya no está visible en el Explorador de **soluciones**.
- El elemento primario del proyecto o elemento eliminado se desprotege según sea necesario (posiblemente con un mensaje.)
- Después de eliminar un elemento desprotegido o agregado, NO aparece en la ventana **Checkins pendientes**.
- El elemento todavía existe dentro del almacén de control de código fuente, incluso después de la eliminación, y debe purgarse manualmente.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Eliminar un proyecto de cliente	1. Cree un proyecto de cliente. 2. Agregue la solución al control de código fuente. 3. Retire todo el proyecto de la solución	Comportamiento esperado común.
Eliminar un archivo vacío	1. Cree un proyecto de cliente. 2. Agregue un archivo de cero bytes al proyecto. 3. Agregue la solución al control de código fuente. 4. Seleccione el archivo, elimínelo.	Comportamiento esperado común.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Eliminar una carpeta con un archivo	1. Cree una solución de proyecto único. 2. Agregue una carpeta. 3. Agregue un archivo a la carpeta. 4. Agregue la solución al control de código fuente. 5. Echa un vistazo al proyecto para evitar mensajes. 6. Elimine la carpeta.	Comportamiento esperado común.
Eliminar un proyecto web del sistema de archivos	1. Cree un proyecto Web del sistema de archivos (utilice el botón Examinar para especificar una ruta UNC). 2. Agregue la solución al control de código fuente. 3. Retire todo el proyecto de la solución. 4. Repita los pasos del 1 al 3 para un proyecto Web local (ejerce diferentes rutas de acceso a través del código, pero tiene la misma interfaz externa y comportamiento).	Comportamiento esperado común.
Eliminar un archivo de un proyecto web del sistema de archivos	1. Cree un proyecto Web del sistema de archivos. 2. Agregue la solución al control de código fuente. 3. Elimine un archivo del proyecto. 4. Repita los pasos del 1 al 3 para un proyecto Web local (ejerce diferentes rutas de acceso a través del código, pero tiene la misma interfaz externa y comportamiento).	Comportamiento esperado común.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 7: compartir

06/04/2020 • 7 minutes to read • [Edit Online](#)

Esta área de prueba cubre el uso compartido de elementos entre ubicaciones mediante el comando **Compartir**.

Una operación hshare es la aparente duplicación de archivos y elementos de carpeta entre dos o más ubicaciones dentro de una jerarquía de archivos de control de código fuente. La duplicación no se produce realmente en el servidor, pero el usuario ve el mismo archivo en dos o más ubicaciones especificadas. Cada vez que se realizan cambios en cualquiera de los elementos compartidos, esos cambios aparecen en todas las demás ubicaciones compartidas.

Compartir en carpetas funciona si selecciona una carpeta con al menos un archivo bajo control de código fuente en ella. El comando share está deshabilitado en las siguientes condiciones:

- Si la carpeta seleccionada es una carpeta vacía.
- Si hay una carpeta real, pero no contiene archivos de control de código fuente.
- Si hay una carpeta virtual, si los archivos bajo control de código fuente están en ella o no.
- Si hay un proyecto Web de sitio remoto.

Acceso al menú de comandos

Las Visual Studio siguientes rutas de menú del entorno de desarrollo integrado se utilizan en los casos de prueba.

Compartir: **Compartir control** -> **Source Control** -> de código fuente de archivos .

Comportamiento esperado

- El archivo compartido aparece en la ubicación compartida.
- Al ver el historial del almacén de versiones del control de código fuente, se muestra que los archivos se comparten.
- La edición de un archivo compartido edita ambas ubicaciones del archivo.

Casos de prueba

Los siguientes son casos de prueba específicos para el área de prueba Compartir.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Compartir un archivo de un proyecto cargado bajo control de código fuente a otro proyecto cargado	1. Cree un nuevo proyecto. 2. Agregue un segundo proyecto a la solución. 3. Cree un archivo en el segundo proyecto con un nombre que no esté en el primer proyecto. 4. Agregue la solución al control de código fuente. 5. Seleccione el primer proyecto. 6. Abra el cuadro de diálogo Compartir (Compartir -> control -> Share de código fuente de archivos). 7. Comparta el archivo del segundo proyecto al primer proyecto. 8. Acepte desproteger si se le solicita.	Comportamiento esperado común.
Compartir un archivo de un proyecto a otro	1. Cree un nuevo proyecto. 2. Añádalo al control de código fuente. 3. Cierre la solución. 4. Crear un segundo proyecto (nueva solución.) 5. Agregue la solución al control de código fuente. 6. Seleccione el proyecto. 7. Abra el cuadro de diálogo Compartir (Compartir -> control -> Share de código fuente de archivos). 8. Comparta un archivo del proyecto agregado anteriormente al proyecto abierto. 9. Acepte Desproteger si se le solicita.	Comportamiento esperado común.
Compartir un archivo que no forma parte del proyecto desde el control de código fuente en el proyecto cargado actualmente	1. Cree un nuevo proyecto. 2. Agregue la solución al control de código fuente. 3. Agregue un archivo al control de código fuente que no forme parte del proyecto o la solución. 4. Seleccione el proyecto y abra el cuadro de diálogo Compartir (-> Compartir****control -> de código fuente de archivos). 5. Seleccione un archivo dentro del cuadro de diálogo Compartir que no exista dentro del proyecto o solución actual y compártalo. 6. Acepte Desproteger si se le solicita.	El almacén de control de código fuente ha realizado un Get, por lo que el archivo está ahora en la ubicación local del proyecto.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Compartir archivos dentro del mismo proyecto a una carpeta diferente	1. Seleccione Desproteger automáticamente en Control de código fuente de opciones -> de herramientas -> . 2. Cree un nuevo proyecto y agréguelo al control de código fuente. 3. Agregue una carpeta al proyecto. 4. Agregue un archivo a la carpeta y compruebe la carpeta. 5. Seleccione la carpeta. 6. Abra el cuadro de diálogo Compartir (Compartir -> control -> Share de código fuente de archivos). 7. Comparta el archivo en la carpeta seleccionada.	Comportamiento esperado común. La carpeta debe protegerse con un archivo en ella antes de que se pueda usar para compartir.
Compartir una carpeta en el proyecto cargado — Recursivo	1. Cree un nuevo proyecto. 2. Agregue la solución al control de código fuente. 3. Seleccione el proyecto. 4. Abra el cuadro de diálogo Compartir (Compartir -> control -> Share de código fuente de archivos). 5. Seleccione una carpeta. 6. Comparta la carpeta de forma recursiva en el proyecto.	Comportamiento esperado común.
Comparte varios archivos de un proyecto a otro	1. Cree un nuevo proyecto con varios archivos en él. 2. Agregue la solución al control de código fuente. 3. Cierre la solución. 4. Cree un nuevo proyecto en una nueva solución. 5. Agregue la solución al control de código fuente. 6. Seleccione el proyecto. 7. Abra el cuadro de diálogo Compartir (Compartir -> control -> Share de código fuente de archivos). 8. Comparta varios archivos del proyecto creado anteriormente con el proyecto actualmente abierto.	Comportamiento esperado común.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Área de prueba 8: cambio de complementos

06/04/2020 • 4 minutes to read • [Edit Online](#)

El Visual Studio entorno de desarrollo integrado (IDE) tiene la interfaz de usuario (UI) para cambiar el complemento de control de código fuente actual. Esta área de prueba proporciona casos de prueba para el proceso de selección del complemento que se usará para el control de código fuente de la solución.

Acceso al menú de comandos

Las Visual Studio siguientes rutas de menú del entorno de desarrollo integrado se utilizan en los casos de prueba.

- Complemento de control de código fuente actual: **Opciones de herramientas** -> **Options** -> Selección **decomplementodecontrol** -> de código fuente .
- Cambiar el enlace de control de código **fuentes**: -> **Control** -> de código fuente de archivos **Cambiar control de código fuente...**

Comportamiento esperado común

Cambiar el complemento de control de código fuente para una solución es posible sin salir de Visual Studio o volver a cargar la solución. Además, el complemento de control de código fuente actual cambia automáticamente al que usa una solución cuando se carga esa solución.

Casos de prueba

Los siguientes son casos de prueba específicos para el área de prueba de conmutación de plug-in.

Caso 8a: Cambio automático

Comportamiento esperado

Cuando un usuario carga una solución que está bajo control de código fuente, la solución se carga automáticamente y se selecciona el complemento de control de código fuente adecuado como actual.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Cambio automático del plug-in de control de código fuente	1. Seleccione plug-in bajo prueba como actual (Opciones -> de herramientas -> Selección de plug-in de control -> de código fuente .) 2. Cree un nuevo proyecto. 3. Agregue la solución al control de código fuente. 4. Seleccione otro plug-in Visual SourceSafe (por ejemplo,). 5. Acepte el mensaje de la solución de descarga. 6. Vuelva a abrir la solución desde el disco.	La solución está abierta. Plug-in bajo prueba es el plug-in de control de código fuente actual.

Caso 8b: Cambio basado en soluciones

Comportamiento esperado

La solución puede cambiar su complemento de control de código fuente asociado.

ACCIÓN	PASOS DE PRUEBA	RESULTADOS ESPERADOS PARA VERIFICAR
Cambio de plug-in para una solución	1. Seleccione plug-in en test como actual (Tools -> Options -> Source Control -> Plug-in Selection). 2. Cree un nuevo proyecto y solución. 3. Agregue la solución al control de código fuente. 4. Desvincule la solución del control de código fuente (mediante el cuadro de diálogo Cambiar control de código fuente). 5. Seleccione otro plug-in Visual SourceSafe(por ejemplo,). 6. Vuelva a cargar la solución del disco si se descarga. 7. Agregue la solución al control de código fuente. 8. Desvincule la solución del control de código fuente (mediante el cuadro de diálogo Cambiar control de código fuente). 9. Seleccione el plug-in en prueba de nuevo. 10. Vuelva a cargar la solución del disco si se descarga. 11. Enlazar la solución a la ubicación original (mediante el cuadro de diálogo Cambiar control de código fuente).	Solución se agrega al control de código fuente mediante el complemento seleccionado.

Vea también

- [Guía de pruebas para los complementos de control de código fuente](#)

Crear un control de código fuente VSPackage

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta documentación incluye vínculos a la información general Visual Studio de la arquitectura de un paquete de control de código fuente integrado con , la API definida por las interfaces que se van a implementar y los servicios que se van a consumir, y un ejemplo que ilustra una implementación de paquete de control de código fuente simple.

Con un control de código fuente VSPackage, puede crear Visual Studio una ruta de acceso de integración profunda para que el control de código fuente se integre con . Permite que el paquete omita la Visual Studio interfaz de usuario de control de código Visual Studio fuente predeterminada hospedada por , responda a las solicitudes de control de código fuente del sistema del proyecto e interactúe con componentes como el **Explorador** de soluciones . El Visual Studio SDK permite Visual Studio a los asociados con un mecanismo Visual Studio para crear un VSPackage que se puede integrar con el uso de un modelo de servicio.

En esta sección

- [Introducción](#)

Describe el paquete de control de código fuente, que es una alternativa más Visual Studio avanzada al complemento de control de código fuente para implementar características de control de código fuente en .

- [Architecture](#)

Presenta un diagrama y explica los componentes de un paquete de control de código fuente.

- [Características](#)

Describe las diversas características de un paquete de control de código fuente.

- [Elementos de diseño](#)

Describe la estructura del VSPackage que un paquete de control de código fuente debe implementar para una integración profunda.

Secciones relacionadas

- [Crear un complemento de control de código fuente](#)

Describe cómo crear un complemento de control de código fuente Visual Studio que proporciona funcionalidad de control de código fuente en la interfaz de usuario (UI) del control de código fuente.

- [Control de código fuente](#)

Describe las opciones para implementar el control Visual Studio de código fuente como una característica integrada de .

Introducción al control de código fuente VSPackages

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un VSPackage de control de código fuente es una alternativa más avanzada al complemento de control de código fuente. Para obtener más información acerca de los complementos de control de código fuente, vea [Crear un complemento](#) de control de código fuente . Un control de código fuente VSPackage permite un control completo sobre el modelo de Visual Studio control de código fuente, características e interfaz de usuario (UI) y se integra en el entorno como un VSPackage.

En esta sección

[Determinar si se debe implementar un control de código fuente VSPackage](#)

Explica las opciones de las soluciones de control de código fuente y proporciona directrices generales para elegir una ruta de integración adecuada.

Secciones relacionadas

- [Novedades del control de código fuente](#)

Describe nuevas características mediante VSPackages con control de código fuente.

- [Crear un control de código fuente VSPackage](#)

Describe cómo crear un control de código fuente VSPackage que no solo proporciona Visual Studio funcionalidad de control de código fuente, pero se puede usar para personalizar la interfaz de usuario del control de código fuente.

Determinar si se debe implementar un control de código fuente VSPackage

06/04/2020 • 3 minutes to read • [Edit Online](#)

En esta sección se explican las opciones de complementos de control de código fuente y VSPackages de control de código fuente para ampliar las soluciones de control de código fuente y proporciona instrucciones generales sobre cómo elegir una ruta de acceso de integración adecuada.

Pequeña solución de control de código fuente con recursos limitados

Si tiene recursos limitados y no se puede cargar con la sobrecarga de escribir un paquete de control de código fuente, puede crear complementos basados en API de complementos de Control de código fuente. Para obtener más información, consulte [Registro y selección](#).

Solución de control de código fuente grande con un amplio conjunto de características

Si desea implementar una solución de control de código fuente que proporciona un modelo de control de código fuente enriquecido que no se captura adecuadamente mediante la API de complemento de Control de código fuente, puede considerar un paquete de control de código fuente como la ruta de acceso de integración. Esto se aplica especialmente si prefiere reemplazar el paquete de adaptador de Control de código fuente (que se comunica con complementos de control de código fuente y proporciona una interfaz de usuario de control de código fuente básica) con la suya propia para que pueda controlar los eventos de control de código fuente de una manera personalizada. Si ya tiene una interfaz de usuario de Visual Studio de control de código fuente satisfactoria y desea conservar esa experiencia en , la opción de paquete de control de código fuente le permite hacer justo eso. El paquete de control de código fuente no Visual Studio es genérico y está diseñado únicamente para su uso con IDE.

Si desea implementar una solución de control de código fuente que proporciona flexibilidad y un control más completo sobre la lógica de control de código fuente y la interfaz de usuario, puede preferir la ruta de integración del paquete de control de código fuente. Puede:

1. Registre su propio control de código fuente VSPackage (consulte [Registro y selección](#)).
2. Reemplace la interfaz de usuario de control de código fuente predeterminada por la interfaz de usuario personalizada (consulte [Interfaz de usuario personalizada](#)).
3. Especifique los pictogramas que se van a usar y controle los eventos de glifo del Explorador de soluciones (consulte [Control de pictogramas](#)).
4. Controlar eventos de edición de consulta y guardado de consultas (consulte [Guardar consulta de edición de consultas](#)).

Vea también

- [Crear un complemento de control de código fuente](#)

Arquitectura de VSPackage de control de código fuente

06/04/2020 • 6 minutes to read • [Edit Online](#)

Un paquete de control de código fuente Visual Studio es un VSPackage que usa los servicios que proporciona el IDE. A cambio, un paquete de control de código fuente proporciona su funcionalidad como un servicio de control de código fuente. Además, un paquete de control de código fuente es una alternativa más Visual Studio versátil que un complemento de control de código fuente para integrar el control de código fuente en .

Un complemento de control de código fuente que implementa la API de complemento de Control de código fuente cumple con un contrato estricto. Por ejemplo, un complemento no Visual Studio puede reemplazar la interfaz de usuario (UI) predeterminada. Además, la API de complemento de Control de código fuente no habilita un complemento para implementar su propio modelo de control de código fuente. Sin embargo, un paquete de control de código fuente supera ambas limitaciones. Un paquete de control de código fuente tiene Visual Studio un control completo sobre la experiencia de control de código fuente de un usuario. Además, un paquete de control de código fuente puede usar su propio modelo de control de código fuente y lógica, y puede definir todas las interfaces de usuario relacionadas con el control de código fuente.

Componentes del paquete de control de código fuente

Como se muestra en Visual Studio el diagrama de arquitectura, un componente denominado el Código auxiliar Visual Studio de control de código fuente es un VSPackage que integra un paquete de control de código fuente con .

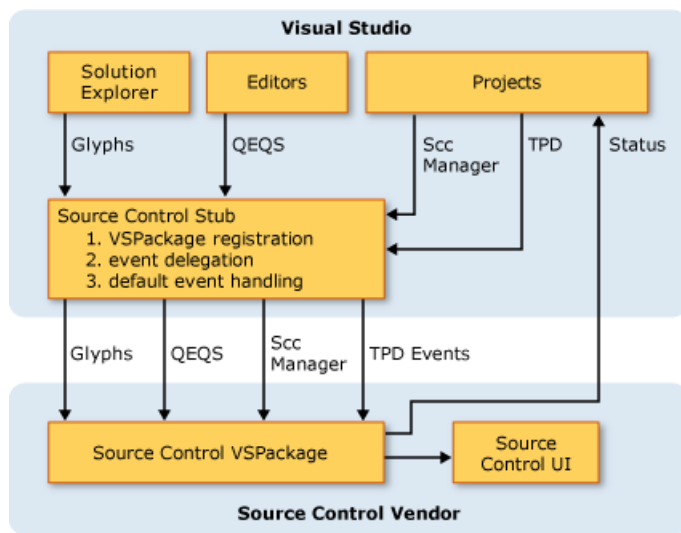
El Código auxiliar de Control de código fuente controla las siguientes tareas.

- Proporciona la interfaz de usuario común que se requiere para el registro del paquete de control de código fuente.
- Carga un paquete de control de código fuente.
- Establece un paquete de control de código fuente como activo/inactivo.

El Código Auxiliar de Control de código fuente busca el servicio activo para el paquete de control de código fuente y enruta todas las llamadas de servicio entrantes desde el IDE a ese paquete.

El paquete de adaptador de control de Visual Studio código fuente es un paquete de control de código fuente especial que proporciona. Este paquete es el componente central para admitir complementos de control de código fuente basados en la API de complemento de Control de código fuente. Cuando un complemento de control de código fuente es el complemento activo, el Código auxiliar de control de código fuente envía sus eventos al paquete de adaptador de Control de código fuente. A su vez, el paquete de adaptador de control de código fuente se comunica con el complemento de control de código fuente mediante el uso de la API de complemento de Control de código fuente y también proporciona una interfaz de usuario predeterminada que es común para todos los complementos de control de código fuente.

Cuando un paquete de control de código fuente es el paquete activo, por otro lado, el Código auxiliar de control de código fuente se comunica directamente con el paquete mediante el uso de las Visual Studio SDK interfaces de paquete de control de código fuente. El paquete de control de código fuente es responsable de hospedar su propia interfaz de usuario de control de código fuente.



Para un paquete de Visual Studio control de código fuente, no proporciona código de control de código fuente ni una API para la integración. Compare esto con el enfoque descrito en Creación de [un complemento](#) de control de código fuente donde el complemento de control de código fuente tiene que implementar un conjunto rígido de funciones y devoluciones de llamada.

Al igual que cualquier VSPackage, un paquete de control `CoCreateInstance` de código fuente es un objeto COM que se puede crear mediante `CoCreateInstance`. El VSPackage se pone Visual Studio a disposición del IDE mediante la implementación `IVsPackage` de `IVsPackage`. Cuando se ha creado una instancia, un VSPackage `IServiceProvider` recibe un puntero de sitio y una interfaz que proporciona el VSPackage acceso a los servicios e interfaces disponibles en el IDE.

Escribir un paquete de control de código fuente basado en VSPackage requiere más experiencia en programación avanzada que escribir un complemento basado en API de complemento de Control de código fuente.

Vea también

- [IVsPackage](#)
- [Introducción](#)

Características de VSPackage de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

En esta sección se describen las diversas características de un control de código fuente VSPackage. Describe los detalles de registro y selección de dicho VSPackage y describe tres de las principales características relacionadas con el control de código fuente: control de eventos Query-Edit Query-Save (QEQS), reemplazo de glifos e interfaz de usuario personalizada (UI) para las funciones de control de código fuente.

En esta sección

- [Registro y selección](#)

Describe los mecanismos de registro y selección del paquete.

- [Edición y guardado de consultas](#)

Explica el rol de query-Edit Query-Save eventos y cómo se controlan mediante el control de código fuente VSPackage.

- [Control de glifos](#)

Describe los niveles de control de glifo y cómo implementarlos.

- [Personalización de la interfaz de usuario](#)

Describe los elementos de interfaz de usuario que puede especificar un control de código fuente VSPackage.

Secciones relacionadas

- [Creación de un VSPackage de control de código fuente](#)

Describe cómo crear un control de código fuente VSPackage que no solo proporciona Visual Studio funcionalidad de control de código fuente, pero se puede usar para personalizar la interfaz de usuario del control de código fuente.

Registro y selección (VSPackage de control de código fuente)

06/04/2020 • 12 minutes to read • [Edit Online](#)

Un control de código fuente VSPackage Visual Studio debe estar registrado para exponerlo a la . Si se registra más de un control de código fuente VSPackage, el usuario puede seleccionar qué VSPackage para cargar en los momentos adecuados. Vea [VSPackages](#) para obtener más detalles sobre VSPackages y cómo registrarlos.

Registro de un paquete de control de código fuente

El paquete de control de Visual Studio código fuente se registra para que el entorno pueda encontrarlo y consultar sus características admitidas. Esto es de acuerdo con un esquema de carga de retraso en el que se crea una instancia de un paquete solo cuando sus características o comandos son necesarios o se solicitan explícitamente.

LOS VSPackages colocan la información en una clave del Registro\específica de la versión, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudioX.Y, donde *X* es el número de versión principal y *Y* es el número de versión secundaria. Esta práctica proporciona la capacidad de admitir la instalación Visual Studio en paralelo de varias versiones de .

La Visual Studio interfaz de usuario (UI) admite la selección entre varios complementos de control de código fuente instalados (a través del paquete de adaptador de control de código fuente) así como VSPackages de control de código fuente. Solo puede haber un complemento de control de código fuente activo o VSPackage a la vez. Sin embargo, como se describe a continuación, el IDE permite cambiar entre complementos de control de código fuente y VSPackages a través de un mecanismo de intercambio de paquetes basado en solución automática. Hay algunos requisitos por parte del control de código fuente VSPackage para habilitar este mecanismo de selección.

Entradas del Registro

Un paquete de control de código fuente necesita tres GUID privados:

- GUID del paquete: este es el GUID principal del paquete que contiene la implementación del control de código fuente (denominada ID_Package en esta sección).
- GUID de Control de código fuente: este es un GUID para el control de código fuente VSPackage utilizado para registrarse con el código auxiliar de Control de código fuente de Visual Studio y también se usa como un GUID de contexto de interfaz de usuario de comando. El GUID del servicio de control de código fuente se registra en el GUID del control de código fuente. En el ejemplo, el GUID del control de código fuente se denomina ID_SccProvider.
- GUID del servicio de control de código fuente: este es el GUID de servicio privado utilizado por Visual Studio (denominado SID_SccPkgService en esta sección). Además de esto, el paquete de control de código fuente debe definir otros GUID para VSPackages, ventanas de herramientas, etc.

Las siguientes entradas del Registro deben realizarse mediante un control de código fuente VSPackage:

NOMBRE DE CLAVE	ENTRADAS
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ VisualStudio\ X.Y\ SourceControlProviders\	(predeterminado) - rg_sz:-ID_SccProvider-

NOMBRE DE CLAVE	ENTRADAS
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ VisualStudio\ X.Y\ SourceControlProviders\ {ID_SccProvider}\	(predeterminado) - <rg_sz: nombre descriptivo del paquete> Servicio: rg_sz:SID_SccPkgService
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ VisualStudio\ X.Y\ SourceControlProviders\ {ID_SccProvider}\ Name\	(predeterminado): rg_sz:<id de recurso para el nombre localizado> Paquete srg_sz:-ID_Package
HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ VisualStudio\ X.Y\ SolutionPersistence\ <PackageName>\	(predeterminado) - rg_sz: ID_Package
(Tenga en cuenta <code>SourceCodeControl1</code> que el nombre Visual Studio de clave, , <ya está utilizado por y no está disponible como una opción para PackageName>.)	

Selección de un paquete de control de código fuente

Varios complementos basados en API de complemento de Control de código fuente y VSPackages de control de código fuente se pueden registrar simultáneamente. El proceso de selección de un complemento de Visual Studio control de código fuente o VSPackage debe asegurarse de que carga el complemento o VSPackage en el momento adecuado y puede aplazar la carga de componentes innecesarios hasta que sean necesarios. Además, Visual Studio debe quitar toda la interfaz de usuario de otros VSPackages inactivos, incluidos los elementos de menú, cuadros de diálogo y barras de herramientas y mostrar la interfaz de usuario para el VSPackage activo.

Visual Studiocarga un control de código fuente VSPackage cuando se realiza cualquiera de las siguientes operaciones:

- La solución se abre (cuando la solución está bajo control de código fuente).

Cuando se abre una solución o proyecto bajo control de código fuente, el IDE hace que se cargue el control de código fuente VSPackage que se designó para esa solución.

- Se ejecuta cualquiera de los comandos de menú del control de código fuente VSPackage.

Un control de código fuente VSPackage debe cargar los componentes que necesita solo cuando realmente se van a usar (también conocido como carga retrasada).

Intercambio automático de VSPackage basado en soluciones

Puede intercambiar manualmente el control de Visual Studio código fuente VSPackages a través del cuadro de diálogo **Opciones** en la categoría **Control de código fuente**. El intercambio automático de paquetes basado en soluciones significa que un paquete de control de código fuente que se ha designado para una solución determinada se establece automáticamente en activo cuando se abre esa solución. Cada paquete de [SetActive](#) control [SetInactive](#) de código fuente debe implementar y . Visual Studiocontrola el conmutador entre ambos complementos de control de código fuente (implementación de la API de complemento de Control de código fuente) y VSPackages de control de código fuente.

El paquete de adaptador de Control de código fuente se usa para cambiar a cualquier complemento basado en API de complemento de Control de código fuente. El proceso de cambiar al paquete de adaptador de control de código fuente intermedio y determinar qué complemento de control de código fuente debe establecerse en activo o inactivo es transparente para el usuario. El paquete de adaptador siempre está activo cuando cualquier complemento de control de código fuente está activo. Cambiar entre dos complementos de control de código fuente equivale a simplemente cargar y descargar el archivo DLL del complemento. Cambiar a un control de código fuente VSPackage, sin embargo, implica interactuar con el IDE para cargar el VSPackage adecuado.

Se llama a un control de código fuente VSPackage cuando se abre cualquier solución y la clave del Registro para el VSPackage está en el archivo de solución. Cuando se abre Visual Studio la solución, busca el valor del Registro y carga el control de código fuente adecuado VSPackage. Todos los VSPackages de control de código fuente deben tener las entradas del Registro descritas anteriormente. Una solución que está bajo control de código fuente se marca como asociada a un control de código fuente determinado VSPackage. VSPackages de control [IVsSolutionPersistence](#) de código fuente debe implementar el para habilitar el intercambio de VSPackage basado en solución automática.

Interfaz de usuario de Visual Studio para la selección y conmutación de paquetes

Visual Studio proporciona una interfaz de usuario para el control de código fuente VSPackage y selección de complementos en el **opciones** cuadro de diálogo en el **Control de código fuente** categoría. Permite al usuario seleccionar el complemento de control de código fuente activo o VSPackage. Una lista desplegable incluye:

- Todos los paquetes de control de código fuente instalados
- Todos los complementos de control de código fuente instalados
- Una opción "ninguno", que deshabilita el control de código fuente

Solo está visible la interfaz de usuario de la opción de control de código fuente activo. La selección de VSPackage oculta la interfaz de usuario para el VSPackage anterior y muestra la interfaz de usuario para el nuevo. El VSPackage activo se selecciona por usuario. Si un usuario tiene Visual Studio varias copias de abierto simultáneamente, cada uno puede usar un VSPackage activo diferente. Si varios usuarios han iniciado sesión en el mismo Visual Studio equipo, cada usuario puede tener instancias independientes de abierto, cada uno con un VSPackage activo diferente. Cuando varias instancias de Visual Studio son cerradas por un usuario, el control de código fuente VSPackage que estaba activo para la última solución abierta se convierte en el control de código fuente predeterminado VSPackage, que se establecerá activo en el reinicio.

A diferencia Visual Studio de las versiones anteriores de , un reinicio IDE ya no es la única manera de cambiar VSPackages de control de código fuente. La selección de VSPackage es automática. Cambiar paquetes requiere privilegios de usuario de Windows (no administrador o usuario avanzado).

Vea también

- [IVsSolutionPersistence](#)
- [Características](#)
- [Creación de un complemento de control de código fuente](#)
- [VSPackages](#)

Editar/guardar consulta (VSPackage de control de código fuente)

06/04/2020 • 5 minutes to read • [Edit Online](#)

Visual Studio los editores pueden transmitir eventos de query Edit Query Save (EQQS). Visual StudioSource Control Stub implementa el servicio EQQS, de modo que sea el destinatario de eventos EQQS. Estos eventos, a continuación, se delegan en el control de código fuente activo actualVSPackage. El control de código fuente [IVsQueryEditQuerySave2](#) activo VSPackage implementa el y sus métodos. Normalmente se `IVsQueryEditQuerySave2` llama a los métodos de la interfaz inmediatamente antes de editar un documento por primera vez e inmediatamente antes de guardar un documento.

Eventos QueryEditQuerySave

El control de código fuente VSPackage debe `IVsQueryEditQuerySave2` controlar los eventos EQQS mediante la implementación de la interfaz y los métodos necesarios. A continuación se muestra una breve descripción de los dos métodos que el VSPackage debe implementar como mínimo. La implementación real debe estar de acuerdo con la lógica del modelo de control de código fuente.

Método QueryEditFiles

Se [QueryEditFiles](#) llama cuando cualquier proyecto o editor desea modificar un archivo. Idealmente, se llama a este método *antes* de modificar el archivo y cuando se guarda un archivo. Cuando se `IVsQueryEditQuerySave2::QueryEditFiles` invoca, el método comprueba si los archivos especificados están bajo control de código fuente, si es necesario desprotegerlos y si se pueden volver a cargar. Si las circunstancias impiden que `IVsQueryEditQuerySave2::QueryEditFiles` los archivos se puedan editar, el método indica al programa que realiza la llamada que cancele la edición. También es posible que el autor de la llamada especifique un modo de invocación. En el modo "silencioso", este método toma acción solo si no hace que aparezca ninguna interfaz de usuario. Si la interfaz de usuario es inevitable, se debe devolver una marca para indicar el problema.

El método se comporta de forma transaccional; es decir, si la edición se cancela en un solo archivo, la edición se cancela para todos los archivos. Por el contrario, si se permite la edición, se permite para todos los archivos. Si este método permite editar una vez para un conjunto determinado de archivos, siempre debe permitir la edición en llamadas posteriores para el mismo conjunto de archivos. El bucle allow-edit continúa hasta que los archivos se cierran, se guardan y se vuelven a cargar; hasta que sus atributos (propiedades) cambien; o hasta que se cambie el paquete de control de código fuente. Los casos a `IVsQueryEditQuerySave2::QueryEditFiles` tener en cuenta en la implementación del método incluyen varios archivos, archivos especiales, cancelar desde el usuario y ediciones en memoria.

Método QuerySaveFiles

Se [QuerySaveFiles](#) llama cuando cualquier proyecto o editor necesita guardar un conjunto de archivos. Cuando se `IVsQueryEditQuerySave2::QuerySaveFiles` invoca, el método comprueba si los archivos especificados son de solo lectura y si están bajo control de código fuente. Si es necesario desproteger los archivos, la llamada se delega en el paquete de control de código fuente. Si las circunstancias impiden que `IVsQueryEditQuerySave2::QuerySaveFiles` los archivos se guarden, el método debe indicar al editor que cancele el guardado. Al igual `IVsQueryEditQuerySave2::QueryEditFiles` que con el método, es posible que el autor de la llamada especifique un modo de invocación. En el modo "silencioso", este método toma acción solo si no hace que aparezca ninguna interfaz de usuario. Si la interfaz de usuario es inevitable, se debe devolver una marca para indicar el problema.

Este método debe comportarse de forma transaccional; es decir, si el guardado se cancela en un solo archivo, el guardado se cancela para todos los archivos. Por el contrario, si se permite guardar, debe permitirse para todos los

archivos. Al igual `IVsQueryEditQuerySave2::QueryEditFiles` que con el método, los casos a tener en cuenta en la implementación del `IVsQueryEditQuerySave2::QuerySaveFiles` método incluyen varios archivos, archivos especiales, cancelar desde el usuario y las ediciones en memoria.

Vea también

- [IVsQueryEditQuerySave2](#)

Control de pictograma (control de código fuente VSPackage)

06/04/2020 • 4 minutes to read • [Edit Online](#)

Parte de la integración profunda disponible para el control de código fuente VSPackages es la capacidad de mostrar sus propios glifos para indicar el estado de los elementos bajo control de código fuente.

Niveles de control de glifos

Un glifo de estado es un icono que indica el estado actual de un elemento cuando se muestra, por ejemplo, en el **Explorador** de soluciones o en la vista de **clases**. Un control de código fuente VSPackage puede ejercer dos niveles de control de glifo. Puede limitar la elección de glifos a un conjunto Visual Studio predefinido de glifos proporcionado por el IDE, o puede definir un conjunto personalizado de glifos que se mostrarán.

Conjunto predeterminado de glifos

Para determinar los glifos de estado asociados a un elemento en el **Explorador** [GetSccGlyph](#) de soluciones, un proyecto solicita el glifo de estado del control de código fuente mediante el archivo. Un control de código fuente VSPackage puede decidir mantener la elección de glifos limitados a glifos predefinidos proporcionados por el IDE. En este caso, el VSPackage devuelve una matriz de valores que representan las enumeraciones de glifo que se definen en *vsshell.idl*. Para obtener más información, vea [VsStatelcon](#). Se trata de un conjunto predefinido de pictogramas establecido por el IDE, como un candado para el glifo protegido y una marca de verificación para el glifo desprotegido.

Conjunto personalizado de glifos

Un control de código fuente VSPackage puede usar sus propios glifos para una apariencia única cuando se instala. Cuando un nuevo control de código fuente VSPackage está activo, debe ser capaz de empezar a usar sus propios glifos, incluso si un control de código fuente anterior VSPackage todavía está cargado pero inactivo. En este modo, el control de código fuente VSPackage todavía puede Visual Studio usar los iconos existentes con el fin de mantener un aspecto coherente con si lo desea.

El [SVsSccManager](#) servicio admite [IVsSccGlyphs](#) una interfaz, que el VSPackage puede implementar opcionalmente y que se pedirá por el IDE. Cuando el IDE realiza Visual Studio una solicitud, a su vez intentará obtener esta interfaz desde el control de código fuente registrado actualmente VSPackage. Si la interfaz existe en el VSPackage registrado, la solicitud del IDE para glifos personalizados se realiza correctamente; de lo Visual Studio contrario, el IDE utiliza su conjunto predeterminado de glifos.

El [GetCustomGlyphList](#) método se Visual Studio utiliza para obtener una lista de imágenes que muestran varios estados de control de código fuente. El control de código fuente VSPackage devuelve al IDE un identificador a la lista de imágenes para sus glifos personalizados. El IDE realiza una copia de la lista de imágenes en este punto y la usa más adelante para elegir los glifos que se van a mostrar. Si no se admite la `IVsSccGlyphs::GetCustomGlyphList` nueva `E_NOTIMPL` interfaz o el método devuelve, el IDE Visual Studio obtiene sus glifos de la lista predeterminada de glifos proporcionados por.

Vea también

- [IVsSccGlyphs](#)
- [VsStatelcon](#)
- [SVsSccManager](#)

Interfaz de usuario personalizada (control de código fuente VSPackage)

06/04/2020 • 7 minutes to read • [Edit Online](#)

Un VSPackage declara sus elementos de menú y sus estados predeterminados a través de la tabla de comandos de Visual Studio (.vsct) archivo. El Visual Studio entorno de desarrollo integrado (IDE) muestra los elementos de menú en sus estados predeterminados hasta que se carga el VSPackage. Posteriormente, [QueryStatus](#) se llama al método para habilitar o deshabilitar los elementos de menú.

Un VSPackage puede establecer una clave del Registro para que el VSPackage se puede cargar automáticamente en función de un contexto de interfaz de usuario de comando (UI), aunque normalmente un control de código fuente VSPackage debe cargar a petición en lugar de simplemente cambiar a un contexto de interfaz de usuario determinado. Para obtener más información acerca de la clave del Registro **AutoLoadPackages**, vea [Administrar VSPackages](#).

Interfaz de usuario de VSPackage

Un paquete de control de código fuente se implementa Visual Studio como un VSPackage y no usa ninguna interfaz de usuario de . Cada control de código fuente VSPackage debe especificar sus propios elementos de interfaz de usuario, como elementos de menú, grupos de menús, ventanas de herramientas, barras de herramientas y cualquier interfaz de usuario necesaria para establecer opciones específicas para el control de código fuente VSPackage. Estos elementos de interfaz de usuario se pueden habilitar estática o dinámicamente. Elementos de interfaz de usuario estáticos se definen en un archivo .vsct y se muestran si el VSPackage está cargado o no. Los elementos dinámicos de la [vsContextNoSolution QueryStatus](#) interfaz de usuario pueden estar visibles en función de un contexto de interfaz de usuario de comando determinado, como , o como resultado de una llamada al método. La visibilidad de los elementos dinámicos de la interfaz de usuario cumple con la estrategia para la carga retrasada de VSPackages.

Restricciones de interfaz de usuario en VSPackages de control de código fuente

Dado que el control de código fuente VSPackage no se puede quitar del IDE después de que se carga, el VSPackage debe ser capaz de escribir un estado inactivo. Cuando un VSPackage recibe una notificación de que ya no está activo, el VSPackage deshabilita su interfaz de usuario e ignora cualquier interacción IDE externa. La implementación del VSPackage del [QueryStatus](#) método debe ocultar comandos cuando el VSPackage no está activo.

Cada control de código `IVsSccProvider` fuente VSPackage debe implementar la interfaz. El VSPackage debe [SetActive](#) [SetInactive](#) implementar dos métodos en la interfaz y , .

El control de código fuente VSPackage puede haberse suscrito a [IVsSolutionEvents3](#) [IVsTrackProjectDocumentsEvents2](#) varios eventos IDE, que se implementan mediante el , , etc. Además, el VSPackage puede haber implementado interfaces de [IVsSolutionPersistence](#) devolución de llamada habilitadas para el registro, como el archivo . Estas interfaces deben ser ignoradas cuando están inactivas.

La lista siguiente muestra las interfaces afectadas por el estado activo de un control de código fuente VSPackage:

- Realizar un seguimiento de los eventos de documentos del proyecto.
- Eventos de solución.

- Interfaces de persistencia de soluciones. Cuando está inactivo, los paquetes no deben escribir en archivos *.sln* y *.suo*.
- Extensores de propiedad.

El [IVsQueryEditQuerySave2](#) necesario [IVsSccManager2](#) y, y también las interfaces opcionales asociadas con el control de código fuente, no se llama cuando el control de código fuente VSPackage está inactivo.

Cuando Visual Studio se inicia Visual Studio el IDE, establece el contexto de la interfaz de usuario del comando en el identificador del identificador de VSPackage del control de código fuente predeterminado actual. Esto hace que la interfaz de usuario estática del control de código fuente activo VSPackage aparezca en el IDE sin cargar realmente el VSPackage. Visual Studio pausa para el VSPackage Visual Studio para [IVsRegisterScsiProvider](#) registrarse a través de la antes de realizar cualquier llamada al VSPackage.

En la tabla siguiente se Visual Studio describen detalles específicos sobre cómo el IDE oculta diferentes elementos de interfaz de usuario.

ELEMENTO DE LA INTERFAZ DE USUARIO	DESCRIPCIÓN
Menús y barras de herramientas	El paquete de control de código fuente debe establecer los estados de visibilidad del menú y la barra de herramientas iniciales en el identificador del paquete de control de código fuente en la sección VisibilityConstraints del archivo <i>.vsct</i> . Esto permite Visual Studio que el IDE para establecer el estado de los elementos QueryStatus de menú correctamente sin cargar el VSPackage y llamar a una implementación del método.
Ventanas de herramientas	El control de código fuente VSPackage oculta las ventanas de herramientas que posee cuando se vuelve inactiva.
Páginas de opciones específicas de VSPackage de control de código fuente	La clave del Registro HKLM-SOFTWARE-Microsoft-VisualStudio-X.Y-ToolsOptionsPages-VisibilityCmdUIContexts permite a un VSPackage establecer los contextos en los que requiere que se muestren sus páginas de opciones. Una entrada de registro bajo esta clave tendría que crearse mediante el identificador de servicio (SID) del servicio de control de código fuente y asignarle un valor DWORD de 1. Siempre que se produce un evento de interfaz de usuario en un contexto con el que se registra el control de código fuente VSPackage, se llamará al VSPackage si está activo.

Vea también

- [QueryStatus](#)
- [IVsQueryEditQuerySave2](#)
- [IVsSccManager2](#)
- [IVsRegisterScsiProvider](#)
- [vsContextNoSolution](#)
- [Administración de VSPackages](#)

Elementos de diseño de VSPackage de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los temas de esta sección describen la estructura que el control de código fuente VSPackage debe implementar para una integración profunda. También enumera las interfaces y servicios que el control de código fuente VSPackage puede implementar Visual Studio y las interfaces y servicios que el control de código fuente VSPackage puede usar desde otros componentes para admitir su modelo de control de código fuente y funcionalidad.

En esta sección

- [Estructura de VSPackage](#)

Define la estructura del control de código fuente VSPackage.

- [Interfaces y servicios relacionados](#)

Enumera las interfaces y servicios relacionados con el paquete de control de código fuente.

- [Servicios proporcionados](#)

Describe el servicio de control de código fuente proporcionado por el control de código fuente VSPackage.

Secciones relacionadas

- [Creación de un VSPackage de control de código fuente](#)

Describe cómo crear un control de código fuente VSPackage que no solo proporciona Visual Studio funcionalidad de control de código fuente, pero se puede usar para personalizar la interfaz de usuario del control de código fuente.

Estructura de VSPackage (VSPackage de control de código fuente)

06/04/2020 • 4 minutes to read • [Edit Online](#)

El SDK de paquete de Control de código fuente proporciona instrucciones para crear un VSPackage que permiten a un implementador de control de código fuente integrar su funcionalidad de control de código fuente con el entorno de Visual Studio. Un VSPackage es un componente COM que normalmente se carga a petición por el entorno de desarrollo integrado (IDE) de Visual Studio en función de los servicios que anuncia el paquete en sus entradas del Registro. Cada VSPackage [IVsPackage](#) debe implementar . Un VSPackage normalmente consume servicios ofrecidos por el IDE de Visual Studio y ofrece algunos servicios propios.

Un VSPackage declara sus elementos de menú y establece un estado de elemento predeterminado a través del archivo .vsct. El IDE de Visual Studio muestra los elementos de menú en este estado hasta que se carga el VSPackage. Posteriormente, se llama a la [QueryStatus](#) implementación del método de VSPackage para habilitar o deshabilitar los elementos de menú.

Características del paquete de control de código fuente

Un control de código fuente VSPackage está profundamente integrado en Visual Studio. La semántica de VSPackage incluye:

- Interfaz que se implementará en virtud [IVsPackage](#) de ser un VSPackage (la interfaz)
- Implementación del comando UI (archivo [IOleCommandTarget](#).vsct e implementación de la interfaz)
- Registro del VSPackage con Visual Studio.

El control de código fuente VSPackage debe comunicarse con estas otras entidades de Visual Studio:

- Proyectos
- Editores
- Soluciones
- Windows
- La tabla de documentos en ejecución

Servicios de entorno de Visual Studio que se pueden consumir

[SVsShell](#)

[SVsUIShell](#)

[SVsSolution](#)

Servicio [SVsRegisterScsiProvider](#)

[SVsQueryEditQuerySave](#)

[SVsTrackProjectDocuments](#)

[SVsSccManager](#)

Interfaces VSIP implementadas y llamadas

Un paquete de control de código fuente es un VSPackage y, por lo tanto, puede interactuar directamente con otros VSPackages que están registrados con Visual Studio. Con el fin de proporcionar la amplitud completa de la funcionalidad de control de código fuente, un control de código fuente VSPackage puede tratar con las interfaces proporcionadas por los proyectos o el shell.

Cada proyecto de Visual [IVsProject3](#) Studio debe implementarse para que se reconozca como un proyecto dentro del IDE de Visual Studio. Sin embargo, esta interfaz no está lo suficientemente especializada para el control de código fuente. Los proyectos que se espera [IVsSccProject2](#) que estén bajo control de código fuente implementan . Esta interfaz se utiliza el control de código fuente VSPackage para consultar un proyecto para su contenido y para proporcionar glifos e información de enlace (la información necesaria para establecer una conexión entre la ubicación del servidor y la ubicación de disco de un proyecto que está bajo control de código fuente).

El control de código [IVsSccManager2](#)fuente que implementa VSPackage , que a su vez permite que los proyectos se registren para el control de código fuente y recuperar sus glifos de estado.

Para obtener una lista completa de interfaces que un control de código fuente VSPackage debe tener en cuenta, vea [Servicios e interfaces relacionados](#).

Vea también

- [Elementos de diseño](#)
- [Interfaces y servicios relacionados](#)

Interfaces y servicios relacionados (VSPackage de control de código fuente)

06/04/2020 • 9 minutes to read • [Edit Online](#)

En esta sección se enumeran todas las interfaces relacionadas con VSPackage del control de código fuente en el Visual Studio SDK. El control de código fuente VSPackage implementa algunas de estas interfaces y usa otras para realizar tareas de control de código fuente.

Interfaces implementadas por y para VSPackages de Control de código fuente

Las interfaces siguientes se describen en el Visual Studio SDK, y el control de código fuente VSPackage implementa un subconjunto de ellos en función de su conjunto de características deseado. Algunas interfaces se marcan como necesarias y deben implementarse en cada control de código fuente VSPackage.

Para las interfaces que un Visual Studio paquete no implementa, proporciona una implementación predeterminada. Tenga en cuenta que la implementación predeterminada está diseñada para el caso cuando no se registra ningún VSPackage y no se controla ningún proyecto. Un control de código fuente correctamente escrito VSPackage implementa todas las interfaces necesarias en lugar de dejarlo en la implementación predeterminada de esas interfaces.

Un control de código fuente VSPackage debe implementar un servicio privado que encapsula algunas o todas las interfaces siguientes.

Las interfaces son:

- **Requerido:** la entidad adecuada (control de código fuente VSPackage, Código auxiliar de control de código fuente, proyecto) debe implementar la interfaz.
- **Recomendado:** La entidad debe implementar esta interfaz; de lo contrario, la funcionalidad de control de código fuente puede estar limitada.
- **Opcional:** la entidad puede implementar esta interfaz para proporcionar un conjunto de características más completo.

INTERFAZ	PROPÓSITO	IMPLEMENTADO POR	¿IMPLEMENTAR?
IVsQueryEditQuerySave2	Los editores llaman a esta interfaz antes de modificar o guardar un archivo. El control de código fuente VSPackage puede desproteger el archivo o denegar la operación si se produce un error en la desprotección.	Control de código fuente VSPackage	Recomendado

INTERFAZ	PROPÓSITO	IMPLEMENTADO POR	¿IMPLEMENTAR?
IVsSccManager2	Esta interfaz proporciona funcionalidad básica de control de código fuente para proyectos, como registrar y anular el registro de proyectos con control de código fuente y proporcionar compatibilidad con glifos de control de código fuente básicos.	Control de código fuente VSPackage	Obligatorio
IVsSccProject2	Esta interfaz se IVsHierarchy obtiene QueryInterface del uso de la función, o simplemente convirtiendo el objeto que se implementa <code>IVsHierarchy</code> en <code>IVsSccProject2</code> . Se utiliza para obtener los archivos bajo control de código fuente en un proyecto o para informar al proyecto del estado o la ubicación del control de código fuente actual.	proyecto	Obligatorio
IVsSccProvider	El módulo de integración utiliza esta interfaz para establecer el VSPackage activo actual.	Control de código fuente VSPackage	Obligatorio
IVsTrackProjectDocuments2	Esta interfaz se basa en un modelo de suscripción. Cualquier VSPackage puede indicar que desea recibir eventos de documento y ser advertido por el shell en los eventos que están a punto de suceder. Se implementa y controla Visual Studio mediante <code>IVsTrackProjectDocumentsEvents2</code> , que a su vez pasa eventos que implementan el VSPackage.	Código de control de código fuente Stub	Obligatorio
IVsTrackProjectDocuments3	Esta interfaz proporciona procesamiento por lotes, operaciones <code>OnQueryAddFiles</code> de lectura y escritura sincronizadas y un método avanzado.	Código de control de código fuente Stub	Obligatorio

INTERFAZ	PROPÓSITO	IMPLEMENTADO POR	¿IMPLEMENTAR?
IVsTrackProjectDocumentsEvents2	El Explorador de soluciones y los proyectos llaman a esta interfaz cuando se agregan nuevos archivos a los proyectos o cuando se cambia el nombre de los archivos y carpetas de los proyectos. El control de código fuente VSPackage puede desproteger el archivo de proyecto o cancelar la operación.	Control de código fuente VSPackage	Recomendado
IVsTrackProjectDocumentsEvents3	El Explorador de soluciones y los proyectos llaman a esta interfaz en respuesta a las llamadas realizadas a los métodos de la <code>IVsTrackProjectDocuments3</code> interfaz. El control de código fuente VSPackage puede realizar un seguimiento de las <code>OnQueryAddFiles</code> operaciones por lotes, operaciones de lectura y escritura sincronizadas y trabajar con un método más avanzado.	Control de código fuente VSPackage	Recomendado
IVsSccEnlistmentPathTranslation	Esta interfaz proporciona compatibilidad con la administración de inscripción para proyectos web.	Control de código fuente VSPackage	Recomendado
IVsSccManagerTooltip	Esta interfaz se utiliza para recuperar información sobre herramientas para los archivos controlados por código fuente en los proyectos.	Control de código fuente VSPackage	Opcional
IVsSccOpenFromSourceControl	Esta interfaz proporciona compatibilidad con la extensión de espacio de nombres.	Control de código fuente VSPackage	Opcional

INTERFAZ	PROPÓSITO	IMPLEMENTADO POR	¿IMPLEMENTAR?
IVsSccControlNewSolution	El VSPackage usa esta interfaz para integrar una extensión de espacio de nombres en el New, Openo Save cuadros de diálogo. Por lo tanto, los proyectos se pueden agregar automáticamente al control de código fuente en la creación o agregar se al control de código fuente cuando una operación de guardado está en vigor.	Control de código fuente VSPackage	Opcional
IVsSccGlyphs	El VSPackage usa esta interfaz para definir glifos adicionales como glifos de control de código fuente para nodos en el Explorador de soluciones .	Control de código fuente VSPackage	Opcional
IVsSccAddWebProjectFromSourceControl	El cuadro de diálogo Agregar para proyectos web utiliza esta interfaz. Proporciona métodos para buscar una ubicación de control de código fuente y para abrir un proyecto Web agregado previamente en el repositorio de control de código fuente en esa ubicación.	Control de código fuente VSPackage	Recomendado
IVsAsynchOpenFromScc	Esta interfaz proporciona compatibilidad para la carga asincrónica (en segundo plano) de proyectos desde el control de código fuente.	Control de código fuente VSPackage	Opcional
IVsAsynchOpenFromSccProjectEvents	Esta interfaz permite a los proyectos ver IVsAsynchOpenFromScc progreso de la carga asincrónica iniciada por .	proyecto	Opcional
IVsSccToolsOptions	Esta interfaz permite que el IDE para consultar el control de código fuente activo VSPackage. El IDE consulta el valor de la configuración del control de código fuente que tiene significado incluso cuando no hay ningún control de código fuente activo VSPackage registrado. Esta interfaz se implementa Visual Studio y se controla mediante .	Código de control de código fuente Stub	Obligatorio

INTERFAZ	PROPÓSITO	IMPLEMENTADO POR	¿IMPLEMENTAR?
IVsRegisterScsiProvider	Esta interfaz se utiliza para registrar el control de código fuente VSPackage.	Código de control de código fuente Stub	Obligatorio
SourceControl	Esta interfaz se utiliza en la automatización. Como tal, expone solo las funciones que se pueden ejecutar sin mostrar ninguna interfaz de usuario.	Control de código fuente VSPackage	Opcional
IVsPersistSolutionProps	Esta interfaz se utiliza para guardar la configuración del control de código fuente en el archivo de solución (.sln). La configuración incluye la ubicación del control de código fuente y los indicadores de estado del control de código fuente.	Control de código fuente VSPackage	Recomendado
IVsPersistSolutionOpts	Esta interfaz se utiliza para guardar la configuración del control de código fuente en el archivo de opciones de solución (.suo). Esto puede incluir la configuración de control de código fuente específica del usuario, como la ubicación de inscripción del usuario actual.	Control de código fuente VSPackage	Recomendado
IVsSolutionEvents3	Esta interfaz se utiliza para supervisar eventos con el fin de realizar operaciones como la comprobación de archivos de proyecto antes de cerrar soluciones o obtener nuevos archivos del control de código fuente al abrir un proyecto.	Control de código fuente VSPackage	Recomendado

Vea también

- [Elementos de diseño](#)

Servicios proporcionados (VSPackage de control de código fuente)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los servicios son el mecanismo principal a través del cual la funcionalidad se comparte entre VSPackages y entre el entorno de desarrollo integrado (IDE) de Visual Studio y sus VSPackages instalados. Para obtener una descripción detallada de los servicios y su importancia en el IDE de Visual Studio, vea [Uso y suministro](#) de servicios .

El Servicio de Control de Código s(Source Control Service)

Visual Studio proporciona dos capas de servicios, servicios de nivel IDE y servicios de nivel de paquete. El IDE de Visual Studio proporciona de forma nativa servicios de nivel IDE. El paquete de control de código fuente consume algunos de estos servicios. El paquete de control de código fuente como un VSPackage comparte su funcionalidad de control de código fuente proporcionando un servicio de control de código fuente privado propio. El paquete de control de código fuente encapsula el conjunto de interfaces relacionadas con el control de código fuente implementadas por él en forma de un contrato que puede usar el IDE de Visual Studio.

Vea también

- [Elementos de diseño](#)

Asistentes

06/04/2020 • 4 minutes to read • [Edit Online](#)

Después de crear un asistente, normalmente Visual Studio desea agregarlo al entorno de desarrollo integrado (IDE) para que otros puedan usarlo. A continuación, el asistente agregado aparece en los cuadros de diálogo **Agregar nuevo proyecto** o **Agregar nuevo elemento**. Para ver los cuadros de diálogo **Agregar nuevo proyecto** o **Agregar nuevo elemento**, haga clic con el botón secundario en una solución abierta en el **Explorador de soluciones**, seleccione **Agregar**, a continuación, haga clic en **Nuevo proyecto** o **Nuevo elemento**. **Add New Project**

Los asistentes de Visual Studio pueden implementarse para permitir a los usuarios seleccionar en una vista de árbol de los valores disponibles cuando abren el cuadro de diálogo **Agregar nuevo proyecto** o el cuadro de diálogo **Agregar nuevo elemento**, o cuando hacen clic con el botón derecho en un elemento del **Explorador de soluciones**.

En el asistente, puede proporcionar la opción de localizar el nombre de un nuevo proyecto o ítem, y puede determinar el icono que verán los usuarios cuando seleccionen el asistente. También puede controlar el orden en el que aparecen nuevos elementos en relación con otros elementos disponibles; los elementos no tienen que organizarse alfabéticamente.

También puede proporcionar un asistente que se inicie de forma diferente, en función de los parámetros personalizados que se pasan al asistente cuando se abre.

En los temas de esta sección de Visual Studio describen los archivos que implementa para que los cuadros de diálogo **Agregar nuevo proyecto** y **Agregar nuevo elemento** muestren el asistente entre los asistentes y plantillas disponibles y los requisitos que debe cumplir el asistente para funcionar correctamente en el IDE.

En esta sección

- [Archivos de descripción del directorio de plantilla \(.Vsdir\)](#)

Proporciona información general sobre qué archivos de descripción de directorio de plantilla y explica cómo funcionan en el IDE para mostrar carpetas, archivos .vsz del asistente y archivos de plantilla asociados a un proyecto en los cuadros de diálogo.

- [Archivo de asistente \(.Vsiz\)](#)

Explica cómo el IDE inicia los asistentes y enumera las tres partes del archivo .vsz.

- [Interfaz de asistente \(IDTWizard\)](#)

Describe la `IDTWizard` interfaz que los asistentes deben implementar para trabajar en el IDE.

- [Parámetros de contexto](#)

Explica cómo se implementan los asistentes y qué ocurre cuando el IDE pasa parámetros de contexto a la implementación.

- [Parámetros personalizados](#)

Explica cómo utilizar parámetros personalizados para controlar el funcionamiento del asistente después de iniciar el asistente.

Secciones relacionadas

- [Tipos de proyecto](#)

Proporciona vínculos a temas adicionales que ofrecen información sobre cómo diseñar nuevos tipos de proyecto.

- [Ampliación de proyectos](#)

Describe cómo usar los proyectos y soluciones de Visual Studio para organizar los archivos de código y de recursos, así como la implementación del control de código fuente.

Archivos de descripción del directorio de plantilla (.Vmdir)

06/04/2020 • 10 minutes to read • [Edit Online](#)

Un archivo de descripción de directorio de plantilla (.vsdir) es un archivo de texto que permite al entorno de desarrollo integrado (IDE) mostrar carpetas, archivos .vsz del asistente y archivos de plantilla asociados al proyecto en cuadros de diálogo. El contenido incluye un registro por archivo o carpeta. Todos los archivos .vsdir en una ubicación a la que se hace referencia se combinan, aunque generalmente solo se proporciona un archivo .vsdir para describir varias carpetas, asistentes o archivos de plantilla.

Las carpetas (subdirectorios), los archivos a los que se hace referencia en el archivo .vsdir y el propio archivo .vsdir se encuentran en el mismo directorio. Cuando el IDE ejecuta un asistente o muestra una carpeta o un archivo en los cuadros de diálogo **Nuevo proyecto** o Agregar **nuevo elemento**, el IDE examina el directorio que contiene los archivos ejecutados para determinar si hay un archivo .vsdir presente. Si se encuentra un archivo .vsdir, el IDE lo lee para determinar si contiene una entrada para la carpeta o archivo ejecutado o mostrado. Si se encuentra una entrada, el IDE usa la información en la ejecución del asistente o la visualización del contenido.

El ejemplo de código siguiente procede del archivo <SourceFiles.vmdir de la clave del Registro EnvSDK>, BscPrj, BscPrjProjectItems, Source_Files:

```
HeaderFile.h|{E59935A1-6156-11d1-87A6-00A0C91E2A46}|#125|130|#126|0|0|0|#127
SourceFile.cpp|{E59935A1-6156-11d1-87A6-00A0C91E2A46}|#122|110|#123|0|0|0|#124
```

En este caso, dos registros están en un archivo. Una nueva línea (carácter de retorno de carro) separa cada registro. Cada línea representa un tipo de archivo diferente. Un carácter de tubería (|) separa los campos de cada registro. Un único directorio puede contener varios archivos .vsdir que tienen nombres de archivo diferentes, o puede tener un archivo .vsdir para cada tipo de archivo.

Fields

En la tabla siguiente se enumeran los campos especificados para cada registro.

CAMPO	DESCRIPCIÓN
Nombre relativo de ruta de acceso (RelPathName)	El nombre de la carpeta, plantilla o archivo .vsz, como HeaderFile.h o MyWizard.vsz. Este campo también puede ser un nombre utilizado para representar una carpeta.
{clsidPackage}	GUID del VSPackage que permite el acceso a cadenas localizadas, como LocalizedName, Description, IconResourceId y SuggestedBaseName, en los recursos de la biblioteca de vínculos dinámicos por satélite (DLL) del VSPackage. IconResourceId se aplica si DLLPath no se proporciona. Nota: Este campo es opcional a menos que uno o varios de los campos anteriores sea un identificador de recursos. Este campo suele estar en blanco para los archivos .vsdir que se corresponden con asistentes de terceros que no localizan su texto.

CAMPO	DESCRIPCIÓN
LocalizedName	El nombre localizado del archivo de plantilla o asistente. Este campo puede ser una cadena o un identificador de recurso con el formato "#ResID". Este nombre se muestra en el cuadro de diálogo Agregar nuevo elemento. Nota: Si LocalizedName es un identificador de recurso, se requiere el valor de .clsidPackage.
SortPriority	Entero que representa la prioridad relativa de este archivo de plantilla o asistente. Por ejemplo, si este elemento tiene un valor de 1, este elemento se muestra junto a otros elementos con un valor de 1 y por delante de todos los elementos con un valor de ordenación de 2 o mayor. La prioridad de ordenación es relativa a los elementos del mismo directorio. Puede haber más de un archivo .vsdir en el mismo directorio. En ese caso, los elementos de todos los <i>archivos</i> .vsdir archivos en ese directorio se combinan. Los elementos con la misma prioridad se enumeran en orden lexicográfico que no distingue mayúsculas de minúsculas del nombre mostrado. La <code>_wcsicmp</code> función se utiliza para ordenar los artículos. Los elementos no descritos en los archivos .vsdir incluyen un número de prioridad mayor que el número de prioridad más alto que aparece en los archivos .vsdir. El resultado es que estos elementos están al final de la lista mostrada independientemente de su nombre.
Descripción	La descripción localizada del archivo de plantilla o asistente. Este campo puede ser una cadena o un identificador de recurso con el formato "#ResID". Esta cadena aparece en el cuadro de diálogo Nuevo proyecto o Agregar nuevo elemento cuando se selecciona el elemento.
DLLPath o {clsidPackage}	Se utiliza para cargar un icono para el archivo de plantilla o el asistente. El icono se carga como un recurso de un archivo .dll o .exe mediante el IconResourceId. Este archivo .dll o .exe se puede identificar mediante una ruta de acceso completa o mediante un GUID de un VSPackage. El archivo DLL de implementación del VSPackage se utiliza para cargar el icono (no el archivo DLL satélite).
IconResourceId	Identificador de recursos en el archivo DLL de implementación de DLL o VSPackage que determina el icono que se va a mostrar.
Banderas <code>_VSDIRFLAGS()</code>	Se utiliza para deshabilitar o habilitar los campos Nombre y Ubicación en el cuadro de diálogo Agregar nuevo elemento. El valor del campo Flags es el equivalente decimal de la combinación de indicadores de bits necesarios. Cuando un usuario selecciona un elemento en la ficha Nuevo, el proyecto determina si el campo Nombre y el campo Ubicación se muestran cuando se muestra por primera vez el cuadro de diálogo Agregar nuevo elemento. Un elemento, a través de un archivo .vsdir, solo puede controlar si los campos están habilitados frente a deshabilitados cuando se selecciona el elemento.

CAMPO	DESCRIPCIÓN
SuggestedBaseName	Representa el nombre predeterminado para el archivo, asistente o plantilla. Este campo es una cadena o un identificador de recurso con el formato "#ResID". El IDE usa este valor para proporcionar un nombre predeterminado para el elemento. Este valor base se anexa con un valor entero para que el nombre sea único, como MyFile21.asp. En la lista anterior, Description, DLLPath, IconResourceId, Flags y SuggestedBaseNumber solo se aplican a los archivos de plantilla y asistente. Estos campos no se aplican a las carpetas. Este hecho se ilustra en el código del archivo <BscPrjProjectItems de la clave del Registro EnvSDK>,BscPrj, BscPrj y BscPrjProjectItems. Este archivo contiene tres registros (uno para cada carpeta) con cuatro campos para cada registro: RelPathName, .clsidPackage, LocalizedName y SortPriority. General&#124;{E59935A1-6156-11d1-87A6-00A0C91E2A46}&#124;#110&#124;100 Source_Files&#124;{E59935A1-6156-11d1-87A6-00A0C91E2A46}&#124;#111&#124;110 Env&#124;{E59935A1-6156-11d1-87A6-00A0C91E2A46}&#124;#112&#124;120

Al crear un archivo de asistente, también debe tener en cuenta los siguientes problemas.

- Los campos opcionales para los que no existen datos significativos tienen que contener un cero (0) como marcador de posición.
- Si no se proporciona ningún nombre localizado, el nombre de ruta de acceso relativo se utiliza en el archivo del asistente.
- DLLPath reemplaza clsidPackage para la ubicación del icono.
- Si no se define ningún icono, el IDE sustituye el icono predeterminado por un archivo que tiene esa extensión.
- Si no se proporciona ningún nombre base sugerido, se utiliza 'Proyecto'.
- Si elimina los archivos, carpetas o archivos de plantilla .vsz, también debe quitar sus registros asociados del archivo .vsdir.

Vea también

- [Asistentes](#)
- [Archivo de asistente \(.Vsz\)](#)

Archivo de asistente (.Vsz)

06/04/2020 • 3 minutes to read • [Edit Online](#)

El entorno de desarrollo integrado (IDE) utiliza archivos .vsz para iniciar asistentes. Estos archivos .vsz contienen información que el IDE utiliza para determinar qué asistente llamar y qué información pasar al asistente.

Un archivo .vsz es una versión de un archivo de texto con formato .ini que no tiene secciones. La información conocida por el IDE se almacena al principio del archivo. Esto proporciona un vínculo entre el asistente que el IDE llama y los parámetros que se encuentran en el archivo .vsz que se pasará al IDE. El resto del archivo proporciona parámetros que son específicos del asistente y que el IDE debe recopilar y pasar al asistente específico.

En el ejemplo siguiente se muestra el contenido de un archivo .vsz.

```
VSWizard 8.0
Wizard=VsWizard.CBlankSiteWizard -or- {123-1234556-123334}
Param="WIZARDNAME = Wizard One"
Param="WIZARDUI = FALSE"
```

A continuación se muestran las partes del archivo .vsz.

PARTE	DESCRIPCIÓN
VSWizard	El primer parámetro del archivo es el número de versión del formato de archivo de plantilla. Este número de versión debe ser 6.0, 7.0, 7.1 o 8.0. No se pueden iniciar otros números y provocar un error de formato no válido.
Asistente	Este campo contiene el ProgID OLE del asistente o, alternativamente, una representación de cadena GUID del CLSID del asistente que es creado cocremente por el IDE.
Parámetro	Estas piezas son opcionales. Puede agregar tantos como sea necesario.

Los parámetros permiten que el archivo .vsz pase parámetros personalizados adicionales al asistente. Cada valor se pasa como un elemento de cadena en una matriz de variantes al asistente. Para obtener más información, consulte [Parámetros personalizados](#).

Para agregar un ID de configuración regional predeterminado `FALLBACK_LCID` a su archivo .vsz, especifique el valor de xxxx, donde xxxx es el identificador de configuración regional, por ejemplo, 1033 para inglés. Cuando `FALLBACK_LCID` se define el parámetro, el asistente utiliza el identificador de configuración regional de reserva proporcionado si no se encuentra el identificador actual.

Vea también

- [Parámetros personalizados](#)
- [Asistentes](#)
- [Archivos de descripción del directorio de plantilla \(.Vmdir\)](#)

Interfaz de asistente (IDTWizard)

06/04/2020 • 3 minutes to read • [Edit Online](#)

El entorno de desarrollo integrado [IDTWizard](#) (IDE) utiliza la interfaz para comunicarse con los asistentes. Los asistentes deben implementar esta interfaz para poder instalarse en el IDE.

El [Execute](#) método es el único [IDTWizard](#) método asociado a la interfaz. Los asistentes implementan este método y el IDE llama al método en la interfaz. En el ejemplo siguiente se muestra la firma del método.

```
/* IDTWizard Method */
STDMETHOD(Execute)(THIS_
    /* [in] */ IDispatch *Application,
    /* [in] */ long hwndOwner,
    /* [in] */ SAFEARRAY * *ContextParams,
    /* [in] */ SAFEARRAY * *CustomParams,
    /* [out] [in] */ wizardResult *RetVal
);
```

El mecanismo de inicio es similar para los asistentes **Nuevo proyecto** y **Agregar nuevo elemento**. Para iniciar cualquiera de [IDTWizard](#) ellos, llame a la interfaz definida en Dteinternal.h. La única diferencia es el conjunto de parámetros de contexto y personalizados que se pasan a la interfaz cuando se llama a la interfaz.

La siguiente información [IDTWizard](#) describe la interfaz que los Visual Studio asistentes deben implementar para trabajar en el IDE. El IDE [Execute](#) llama al método en el asistente, pasándole lo siguiente:

- El objeto DTE

El objeto DTE es la raíz del modelo de automatización.

- El identificador del cuadro de diálogo de `hwndOwner ([in] long)` la ventana como se muestra en el segmento de código, .

El asistente `hwndOwner` lo utiliza como elemento primario para el cuadro de diálogo del asistente.

- Parámetros de contexto pasados a la interfaz como `[in] SAFEARRAY (VARIANT)* ContextParams` variante para `SAFEARRAY` como se muestra en el segmento de código, .

Los parámetros de contexto contienen una matriz de valores que son específicos del tipo de asistente que se está iniciando y el estado actual del proyecto. El IDE pasa los parámetros de contexto al asistente. Para obtener más información, consulte [Parámetros de contexto](#).

- Parámetros personalizados pasados a la interfaz como variante `[in] SAFEARRAY (VARIANT)* CustomParams` para `SAFEARRAY` como se muestra en el segmento de código, .

Los parámetros personalizados contienen una matriz de parámetros definidos por el usuario. Un archivo .vsz pasa parámetros personalizados al IDE. Los valores están `Param=` determinados por las instrucciones. Para obtener más información, consulte [Parámetros personalizados](#).

- Los valores devueltos para la interfaz son

```
wizardResultSuccess = -1,
wizardResultFailure = 0
wizardResultCancel = 1
wizardResultBackout = 2
```

Vea también

- [Parámetros de contexto](#)
- [Parámetros personalizados](#)
- [Asistentes](#)
- [Archivo de asistente \(.Vsz\)](#)

Parámetros de contexto

06/04/2020 • 6 minutes to read • [Edit Online](#)

En Visual Studio el entorno de desarrollo integrado (IDE), puede agregar asistentes a los cuadros de diálogo **Nuevo proyecto**, **Agregar nuevo elemento** o **Agregar subproyecto**. Los asistentes agregados están disponibles en el menú **Archivo** o haciendo clic con el botón derecho en un proyecto en el **Explorador de soluciones**. El IDE pasa parámetros de contexto a la implementación del asistente. Los parámetros de contexto definen el estado del proyecto cuando el IDE llama al asistente.

El IDE inicia los **VSADDITEMOPERATION** asistentes estableciendo la marca **AddItem** en la llamada del IDE al método para el proyecto. Cuando se establece, el `IVsExtensibility::RunWizardFile` proyecto debe hacer que el método se ejecute mediante el nombre del asistente registrado o GUID y otros parámetros de contexto que el IDE le pasa.

Parámetros de contexto para el nuevo proyecto

PARÁMETRO	DESCRIPCIÓN
<code>WizardType</code>	Tipo de <code>devsWizardNewProject</code> asistente registrado () o el GUID que indica el tipo de asistente. En Visual Studio SDK la implementación, el GUID del asistente es .0F90E1D0-4999-11D1-B6D1-00A0C90F2744.
<code>ProjectName</code>	Cadena que es Visual Studio el nombre único del proyecto.
<code>LocalDirectory</code>	Ubicación local de los archivos de proyecto de trabajo.
<code>InstallationDirectory</code>	Ruta de Visual Studio acceso de directorio de la instalación es.
<code>FExclusive</code>	Indicador booleano que indica que el proyecto debe cerrar soluciones abiertas.
<code>SolutionName</code>	Nombre del archivo de solución sin la parte del directorio o la extensión <code>.sln</code> . El nombre de archivo <code>.suo</code> también se crea mediante <code>SolutionName</code> . Cuando este argumento no es una Create cadena vacía, AddFromTemplate el asistente lo usa antes de agregar el proyecto con <code>.</code> . Si este nombre es una AddFromTemplate cadena Create vacía, utilice sin llamar a <code>.</code>
<code>Silent</code>	Booleano que indica si el asistente Finish debe ejecutarse <code>TRUE</code> silenciosamente como si se hiciera clic en Finalizar ().

Parámetros de contexto para Agregar nuevo elemento

PARÁMETRO	DESCRIPCIÓN
<code>WizardType</code>	Tipo de <code>devsWizardAddItem</code> asistente registrado () o el GUID que indica el tipo de asistente. En Visual Studio SDK la implementación, el GUID para el asistente es .0F90E1D1-4999-11D1-B6D1-00A0C90F2744.

PARÁMETRO	DESCRIPCIÓN
<code>ProjectName</code>	Cadena que es Visual Studio el nombre único del proyecto.
<code>ProjectItems</code>	Ubicación local que contiene archivos de proyecto de trabajo.
<code>ItemName</code>	Nombre del elemento que se va a agregar. Este nombre es el nombre de archivo predeterminado o el nombre de archivo que el usuario escribe en el cuadro de diálogo Agregar elementos . El nombre se basa en las marcas que se establecen en el archivo <i>.vsdir</i> . El nombre puede ser un valor nulo.
<code>InstallationDirectory</code>	Ruta de Visual Studio acceso de directorio de la instalación es.
<code>Silent</code>	Booleano que indica si el asistente Finish debe ejecutarse <code>TRUE</code> silenciosamente como si se hiciera clic en Finalizar ().

Parámetros de contexto para Agregar subproyecto

PARÁMETRO	DESCRIPCIÓN
<code>WizardType</code>	Tipo de <code>vsWizardAddSubProject</code> asistente registrado () o el GUID que indica el tipo de asistente. En Visual Studio SDK la implementación, el GUID para el asistente es .0F90E1D2-4999-11D1-B6D1-00A0C90F2744.
<code>ProjectName</code>	Cadena que es Visual Studio el nombre único del proyecto.
<code>ProjectItems</code>	Puntero a <code>ProjectItems</code> la colección en la que opera el asistente. Este puntero se pasa al asistente en función de la selección de jerarquía del proyecto. Normalmente, un usuario selecciona una carpeta en la que colocar el elemento y, a continuación, llama al cuadro de diálogo Agregar elemento del proyecto.
<code>LocalDirectory</code>	Ubicación local de los archivos de proyecto de trabajo.
<code>ItemName</code>	Nombre del elemento que se va a agregar. Este nombre es el nombre de archivo predeterminado o el nombre de archivo que el usuario escribe en el cuadro de diálogo Agregar elementos . El nombre se basa en las marcas que se establecen en el archivo <i>.vsdir</i> . El nombre puede ser un valor nulo.
<code>InstallationDirectory</code>	Ruta de Visual Studio acceso del directorio de la instalación.
<code>Silent</code>	Booleano que indica si el asistente Finish debe ejecutarse <code>TRUE</code> silenciosamente como si se hiciera clic en Finalizar ().

Vea también

- [IVsProject](#)
- [IVsProject2](#)

- [Parámetros personalizados](#)
- [Asistentes](#)
- [Archivo Wizard \(.vsz\)](#)
- [Parámetros de contexto para iniciar asistentes](#)

Parámetros personalizados

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los parámetros personalizados controlan el funcionamiento de un asistente después de que se haya iniciado un asistente. Un archivo `.vsz` relacionado proporciona una matriz de parámetros definidos por el usuario empaquetados por el entorno de desarrollo integrado (IDE) y pasados al asistente como una matriz de cadenas cuando se inicia el asistente. A continuación, el asistente analiza la matriz de cadenas y utiliza la información para controlar el funcionamiento real del asistente. De esta manera, un asistente puede personalizar la funcionalidad en función del contenido del archivo `.vsz`.

Los parámetros de contexto, por otro lado, definen el estado del proyecto cuando se inicia el asistente. Para obtener más información, consulte [Parámetros de contexto](#).

A continuación se muestra un ejemplo de un archivo `.vsz` que tiene parámetros personalizados:

```
VSWIZARD 8.0
Wizard=VsWizard.VsWizard_Engine
Param="WIZARD_NAME = Sample Wizard"
Param="WIZARD_UI = FALSE"
Param="RELATIVE_PATH = VSWizards\Classwiz\ATL"
Param="PREPROCESS_FUNCTION = CanAddATLSupport"
Param="PROJECT_TYPE = CSPROJ"
```

El autor del archivo `.vsz` agrega los valores de los parámetros. Cuando un usuario selecciona **Nuevo proyecto** o **Agregar nuevo elemento** en el archivo menú o haciendo clic en un proyecto en el **Explorador** de soluciones, el IDE recopila estos valores en una matriz de cadenas. A continuación, el IDE [AddItem](#) llama [VSADDITEMOPERATION](#) al método del proyecto [RunWizardFile](#) con el conjunto de indicadores y el proyecto llama al método responsable de ejecutar el asistente y devolver el resultado.

El asistente es responsable de analizar la matriz de cadenas y actuar en las cadenas de forma adecuada. De esta manera, mediante la implementación de parámetros personalizados puede crear un asistente que realice una variedad de funciones. En otras palabras, un asistente podría tener tres archivos `.vsz` diferentes. Cada archivo pasa diferentes conjuntos de parámetros personalizados para controlar el comportamiento del asistente en varias situaciones.

Para obtener más información, consulte [Wizard \(.vsz\) file](#).

Vea también

- [IVsProject3](#)
- [Parámetros de contexto](#)
- [Asistentes](#)
- [Archivo Wizard \(.vsz\)](#)

Herramientas personalizadas

06/04/2020 • 3 minutes to read • [Edit Online](#)

Las *herramientas personalizadas* le permiten asociar una herramienta con un elemento de un proyecto y ejecutarla siempre que se guarde el archivo. Ciertas herramientas personalizadas, a veces denominadas *generadores de un solo archivo*, se utilizan con frecuencia para implementar traductores que generan código a partir de datos y viceversa. Por ejemplo, los generadores Visual C# Visual Basic de un solo archivo crean y obtienen código fuente a partir de los archivos *.settings* y *.resx*. El código fuente generado proporciona acceso fuertemente tipado a los datos de los archivos *.settings* y *.resx*. Los Visual C# Visual Basic tipos y los de proyecto admiten herramientas personalizadas; Visual C++ los tipos de proyecto no lo hacen. Sus propios tipos de proyecto también pueden admitir herramientas personalizadas.

Las herramientas personalizadas son `IVsSingleFileGenerator` componentes registrados que implementan la interfaz.

Las herramientas personalizadas `ProjectItem` están asociadas a un objeto de interfaz y son como diseñadores y editores. Una herramienta personalizada toma el `ProjectItem` archivo representado por una como entrada y escribe `DefaultExtension` un nuevo archivo cuyo nombre de archivo es proporcionado por el método.

En esta sección

- [Implementar generadores de un solo archivo](#)

Describe cómo usar `IVsSingleFileGenerator` la interfaz para implementar una herramienta personalizada.

- [Registrar generadores de archivos individuales](#)

Proporciona descripciones para todas las entradas del registro para una herramienta personalizada.

- [Exponer tipos a diseñadores visuales](#)

Explica cómo los sistemas de proyectos proporcionan compatibilidad para que los diseñadores visuales accedan a clases y tipos generados a través de archivos ejecutables portátiles temporales (PE).

- [Persistir la propiedad de un elemento de proyecto](#)

Muestra cómo conservar una propiedad de elemento de proyecto, como el autor de un archivo de origen, en el archivo de proyecto.

Referencia

`IVsSingleFileGenerator` Proporciona detalles `IVsSingleFileGenerator` sobre el , que transforma un único archivo de entrada en un único archivo de salida que se puede compilar o agregar a un proyecto.

`ProjectItem` Explica `ProjectItem` la interfaz, que representa un elemento de un proyecto.

`DefaultExtension` Proporciona detalles `DefaultExtension` sobre el método, que recupera la extensión de nombre de archivo que se proporciona al nombre de archivo de salida.

Secciones relacionadas

- [Ampliar proyectos](#)

Describe cómo usar los proyectos y soluciones de Visual Studio para organizar los archivos de código y de

recursos, así como la implementación del control de código fuente.

Implementación de generadores de un solo archivo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una herramienta personalizada, a veces denominada un único generador Visual Basic Visual C# de archivos, se puede utilizar para ampliar los sistemas y proyectos en Visual Studio. Una herramienta personalizada es un componente [IVsSingleFileGenerator](#) COM que implementa la interfaz. Con esta interfaz, una herramienta personalizada transforma un único archivo de entrada en un único archivo de salida. El resultado de la transformación puede ser código fuente o cualquier otra salida que sea útil. Dos ejemplos de archivos de código generados por herramientas personalizadas son código generado en respuesta a los cambios en un diseñador visual y los archivos generados mediante el lenguaje de descripción de servicios web (WSDL).

Cuando se carga una herramienta personalizada o se guarda el [Generate](#) archivo de entrada, el [IVsGeneratorProgress](#) sistema del proyecto llama al método y pasa una referencia a una interfaz de devolución de llamada, por lo que la herramienta puede notificar su progreso al usuario.

El archivo de salida que genera la herramienta personalizada se agrega al proyecto con una dependencia en el archivo de entrada. El sistema de proyectos determina automáticamente el nombre del archivo de salida, en [DefaultExtension](#) función de la cadena devuelta por la implementación de la herramienta personalizada de .

Una herramienta personalizada [IVsSingleFileGenerator](#) debe implementar la interfaz. Opcionalmente, las herramientas [IObjectWithSite](#) personalizadas admiten la interfaz para recuperar información de orígenes distintos del archivo de entrada. En cualquier caso, antes de poder utilizar una herramienta personalizada, Visual Studio debe registrarla en el sistema o en el registro local. Para obtener más información sobre el registro de herramientas personalizadas, consulte Registro de generadores de [archivos únicos](#).

Vea también

- [Exposición de tipos a diseñadores visuales](#)

Registro de generadores de un solo archivo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Para que una herramienta Visual Studio personalizada esté disponible Visual Studio en , debe registrarla para poder crear instancias y asociarla a un tipo de proyecto determinado.

Para registrar una herramienta personalizada

1. Registre el archivo DLL Visual Studio de la herramienta personalizada en el registro local o en el registro del sistema, en HKEY_CLASSES_ROOT.

Por ejemplo, aquí está la información de registro para la herramienta Visual Studio personalizada administrada MSDataSetGenerator, que viene con:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\14.0\CLSID\{E76D53CC-3D4F-40A2-BD4D-4F3419755476}]
@="COM+ class: Microsoft.VSDesigner.CodeGenerator.TypedDataSourceGenerator.DataSourceGeneratorWrapper"
"InprocServer32"="C:\\WINDOWS\\system32\\mscoree.dll"
"ThreadingModel"="Both"
"Class"="Microsoft.VSDesigner.CodeGenerator.TypedDataSourceGenerator.DataSourceGeneratorWrapper"
"Assembly"="Microsoft.VSDesigner, Version=14.0.0.0, Culture=Neutral, PublicKeyToken=b03f5f7f11d50a3a"
```

2. Cree una clave del Visual Studio Registro en el subárbol deseado en el GUID de generadores, donde GUID es el GUID definido por el sistema de proyecto o servicio del lenguaje específico. El nombre de la clave se convierte en el nombre de programación de la herramienta personalizada. La clave de herramienta personalizada tiene los siguientes valores:

- (Es el valor predeterminado).

Opcional. Proporciona una descripción fácil de usar de la herramienta personalizada. Este parámetro es opcional, pero se recomienda.

- CLSID

Necesario. Especifica el identificador de la biblioteca de clases [IVsSingleFileGenerator](#) del componente COM que implementa .

- GeneratesDesignTimeSource

Necesario. Indica si los tipos de archivos generados por esta herramienta personalizada están disponibles para los diseñadores visuales. El valor de este parámetro debe ser (cero) 0 para los tipos que no están disponibles para los diseñadores visuales o (uno) 1 para los tipos disponibles para los diseñadores visuales.

NOTE

Debe registrar la herramienta personalizada por separado para cada idioma para el que desea que la herramienta personalizada esté disponible.

Por ejemplo, MSDataSetGenerator se registra una vez para cada idioma:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\14.0\Generators\{164b10b9-b200-11d0-8c61-00a0c91e29d5}\MSDataSetGenerator]
@="Microsoft VB Code Generator for XSD"
"CLSID"="{E76D53CC-3D4F-40a2-BD4D-4F3419755476}"
"GeneratesDesignTimeSource"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\14.0\Generators\{fae04ec1-301f-11d3-bf4b-00c04f79efbc}\MSDataSetGenerator]
@="Microsoft C# Code Generator for XSD"
"CLSID"="{E76D53CC-3D4F-40a2-BD4D-4F3419755476}"
"GeneratesDesignTimeSource"=dword:00000001
```

Vea también

- [IVsSingleFileGenerator](#)
- [Implementación de generadores de un solo archivo](#)
- [Exposición de tipos a diseñadores visuales](#)
- [Introducción al objeto BuildManager](#)

Exponer tipos a diseñadores visuales

06/04/2020 • 4 minutes to read • [Edit Online](#)

Visual Studio debe tener acceso a las definiciones de clase y tipo en tiempo de diseño para mostrar un diseñador visual. Las clases se cargan desde un conjunto predefinido de ensamblados que incluyen el conjunto de dependencias completo del proyecto actual (referencias más sus dependencias). También puede ser necesario que los diseñadores visuales accedan a clases y tipos definidos en archivos generados por herramientas personalizadas.

Los Visual Basic Visual C# sistemas y los sistemas de proyecto proporcionan compatibilidad para acceder a clases y tipos generados a través de archivos ejecutables portátiles temporales (PE temporales). Cualquier archivo generado por una herramienta personalizada se puede compilar en un ensamblado temporal para que los tipos se puedan cargar desde esos ensamblados y exponerse a los diseñadores. La salida de cada herramienta personalizada se compila en un PE temporal independiente y el éxito o el error de esta compilación temporal depende únicamente de si se puede compilar o no el archivo generado. Aunque un proyecto puede no construirse como un todo, los PE temporales individuales pueden seguir estando disponibles para los diseñadores.

El sistema de proyectos proporciona compatibilidad completa para realizar el seguimiento de los cambios en el archivo de salida de una herramienta personalizada, siempre que estos cambios sean el resultado de ejecutar la herramienta personalizada. Cada vez que se ejecuta la herramienta personalizada, se genera un nuevo PE temporal y se envían las notificaciones adecuadas a los diseñadores.

NOTE

Dado que el archivo de generación ejecutable de programa temporal se produce en segundo plano, no se notifica ningún error al usuario si se produce un error en la compilación.

Las herramientas personalizadas que aprovechan la compatibilidad temporal con PE deben seguir las siguientes reglas:

- **GeneratesDesignTimeSource** debe establecerse en 1 en el registro.

Ninguna compilación de archivos ejecutables del programa tiene lugar sin esta configuración.

- El código generado debe estar en el mismo idioma que la configuración del proyecto global.

El PE temporal se compila independientemente de lo que [DefaultExtension](#) la herramienta personalizada notifica como la extensión solicitada en siempre que **GeneratesDesignTimeSource** se establece en 1 en el registro. La extensión no necesita ser *.vb*, *.cso* *.jsf*; puede ser cualquier extensión.

- El código generado por la herramienta personalizada debe ser válido y debe compilarse por sí solo [Generate](#) utilizando solo el conjunto de referencias presentes en el proyecto en el momento en que finaliza la ejecución.

Cuando se compila un PE temporal, el único archivo de origen proporcionado al compilador es la salida de la herramienta personalizada. Por lo tanto, una herramienta personalizada que utiliza un PE temporal debe generar archivos de salida que se pueden compilar independientemente de otros archivos del proyecto.

Vea también

- [Introducción al objeto BuildManager](#)
- [Implementar generadores de un solo archivo](#)

- [Registrar generadores de un solo archivo](#)

Utilidades VSSDK

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los temas de esta sección describen las herramientas que se envían con VSSDK.

En esta sección

- [Utilidad RegPkg](#)
- [Utilidad CreatePkgDef](#)
- [Utilidad CreateExplInstance](#)
- [Herramientas de creación de temas de color](#)
- [Herramientas de servicio de imágenes](#)

Secciones relacionadas

Utilidad RegPkg

06/04/2020 • 3 minutes to read • [Edit Online](#)

NOTE

La forma preferida de registrar paquetes en Visual Studio es mediante archivos .pkgdef. Esto permite la implementación de extensiones sin tener que tener acceso al registro del sistema, que es un requisito para la implementación de VSIX. Los archivos Pkgdef se crean mediante la [utilidad CreatePkgDef](#). Para obtener más información sobre la implementación de paquetes de Visual Studio, vea Envío de [extensiones](#) de Visual Studio .

La utilidad RegPkg.exe registra un Visual Studio VSPackage con y lo prepara para la implementación. Esta utilidad se usa en segundo plano durante el desarrollo de VSPackage. Se ejecuta como parte del proceso de compilación para que pueda compilar y ejecutar un VSPackage en el subárbol experimental.

RegPkg puede generar scripts de registro del sistema en varios formatos. Puede incorporar estos scripts en proyectos de implementación como proyectos .msi o archivos de conjunto de herramientas XML de Windows Installer.

RegPkg.exe se encuentra <normalmente en la ruta de instalación del SDK de *Visual Studio*> . RegPkg sigue esta sintaxis:

```
RegPkg [/root:<root>] [/regfile:<regfile>] [/rgsfile:<rgsfile> [/rgm]] [/vrgfile:<vrgfile>] [/codebase |  
/assembly] [/unregister] AssemblyPath
```

/root:root Realiza el registro en el

Visual Studio raíz.

/regfile:FileName Crea un archivo .reg en lugar de actualizar el registro. No se puede utilizar con /vrgfile o /rgsfile o /wixfile.

/rgsfile:FileName Crea un archivo .rgs en lugar de actualizar el registro. No se puede utilizar con /vrgfile o /regfile o /wixfile.

/vrgfile:FileName Crea un archivo .vrg en lugar de actualizar el registro. No se puede utilizar con /regfile o /rgsfile o /wixfile.

/rgm Crea un archivo .rgm además del archivo rgs. Debe combinarse con /rgsfile.

/wixfile:FileName Crea un archivo compatible con el conjunto de herramientas XML de Windows Installer en lugar de actualizar el registro. No se puede utilizar con /regfile o /rgsfile o /vrgfile.

/codebase Fuerza el registro con CodeBase en lugar de Assembly.

/assembly Fuerza el registro con Assembly en lugar de CodeBase.

/unregister Anula el registro de este paquete. No se puede utilizar con /regfile o /vrgfile o /rgsfile o /wixfile.

Vea también

- [VSPackages](#)

- [Solución de problemas del registro de paquete RegPkg](#)

Solución de problemas del registro de paquete RegPkg

06/04/2020 • 2 minutes to read • [Edit Online](#)

NOTE

La forma preferida de registrar paquetes en Visual Studio es mediante archivos .pkgdef. Esto permite la implementación de extensiones sin tener que acceder al registro del sistema. Los archivos Pkgdef se crean mediante [la utilidad CreatePkgDef](#).

Para registrar un paquete mediante Visual StudioRegPkg en , debe usar la versión de RegPkg que sea adecuada para el paquete.

Versiones reg/resctuyes relacionadas con versiones de paquetes

Hay dos versiones de RegPkg. Una versión Visual Studio incluye en . Utilice esta versión para registrar los paquetes que se han compilado mediante uno de los ensamblados siguientes:

1. Microsoft.VisualStudioShell.9.0.dll
2. Microsoft.VisualStudioShell.10.0.dll
3. Microsoft.VisualStudioShell.11.0.dll

No puede registrar paquetes que se han compilado mediante el ensamblado Microsoft.VisualStudio.Shell.dll anterior.

La versión anterior de RegPkg puede registrar paquetes que se han compilado mediante el ensamblado Microsoft.VisualStudio.Shell.dll. Sin embargo, no puede registrar paquetes compilados mediante versiones posteriores de ese ensamblado.

Vea también

- [VSPackages](#)

Utilidad CreatePkgDef

06/04/2020 • 3 minutes to read • [Edit Online](#)

Toma un archivo .dll para una extensión de Visual Studio como parámetro y crea un archivo .pkgdef para acompañar el archivo .dll. El archivo .pkgdef contiene toda la información que de otro modo se escribiría en el registro del sistema cuando se instala la extensión.

NOTE

La mayoría de las plantillas de proyecto que se incluyen en el SDK de Visual Studio crean automáticamente archivos .pkgdef como parte del proceso de compilación. Este documento está pensado para aquellos que desean crear paquetes manualmente o convertir paquetes existentes para usar la implementación .pkgdef.

Sintaxis

```
CreatePkgDef /out=<FileName> [/codebase] [/assembly] <AssemblyPath>
```

Argumentos

/out-<NombredeArchivo>

Necesario. Establece el nombre del archivo de <salida> .pkgdef en FileName .

/codebase

Opcional. Fuerza el registro con la utilidad CodeBase.

/ensamblaje

Fuerza el registro con la utilidad Asamblea.

<AssemblyPath>

La ruta de acceso del archivo .dll desde el que desea generar el archivo .pkgdef.

Observaciones

La implementación de extensiones mediante archivos .pkgdef reemplaza los requisitos del Registro de versiones anteriores de Visual Studio.

Los archivos .pkgdef deben instalarse en una de las siguientes ubicaciones:

- %localappdata%-Microsoft-Visual Studio-16.0-Extensiones\
- %vsinstalldir%-Common7-IDE-Extensiones\

Si la carpeta de instalación es %localappdata%-Microsoft-Visual\Studio-16.0-Extensions, visual Studio reconoce la extensión, pero está deshabilitada de forma predeterminada. El usuario puede habilitar la extensión mediante **Administrar extensiones**.

Si la carpeta de instalación es %vsinstalldir%-Common7-IDE-Extensions\, la extensión está habilitada de forma predeterminada.

NOTE

La herramienta **Administrar extensiones** no se puede usar para tener acceso a una extensión a menos que se instale como parte de un paquete VSIX.

Los archivos *.pkgdef* deben instalarse en una de las siguientes ubicaciones:

- *%localappdata%-Microsoft-Visual Studio-15.0-Extensiones*
- *%vsinstalldir%-Common7-IDE-Extensiones*

Si la carpeta de instalación es *%localappdata%-Microsoft-Visual\Studio-15.0-Extensions*, Visual Studio reconoce la extensión, pero está deshabilitada de forma predeterminada. El usuario puede habilitar la extensión mediante **Extensiones y actualizaciones**.

Si la carpeta de instalación es *%vsinstalldir%-Common7-IDE-Extensions*, la extensión está habilitada de forma predeterminada.

NOTE

La herramienta **Extensiones y actualizaciones** no se puede utilizar para tener acceso a una extensión a menos que se instale como parte de un paquete VSIX.

Vea también

- [Utilidad CreateExpInstance](#)

Utilidad CreateExpInstance

06/04/2020 • 2 minutes to read • [Edit Online](#)

Utilice la utilidad **CreateExpInstance** para crear, restablecer o eliminar una instancia experimental de Visual Studio. Puede usar la instancia experimental para depurar y probar extensiones de Visual Studio sin cambiar el producto subyacente.

Sintaxis

```
CreateExpInstance.exe [/Create | /Reset | /Clean] /VSInstance=VsInstance /RootSuffix=Suffix
```

Parámetros

/Crear Crea la instancia experimental.

/Reset Elimina la instancia experimental y, a continuación, crea una nueva.

/Limpieza Elimina la instancia experimental.

/VSInstance El nombre del directorio que contiene la instancia base de Visual Studio que se va a copiar.

/RootSuffix Sufijo que se va a anexar al nombre del directorio de instancia experimental.

Observaciones

Cuando se trabaja en una extensión de Visual Studio, puede presionar F5 para abrir la instancia experimental predeterminada e instalar la extensión actual. Si no hay ninguna instancia experimental disponible, Visual Studio crea una que tiene la configuración predeterminada.

La ubicación predeterminada de la instancia experimental depende del número de versión de Visual Studio. Por ejemplo, para Visual Studio 2015, la ubicación es *%localappdata%-Microsoft-VisualStudio-14.0Exp*. Todos los archivos de la ubicación del directorio se consideran parte de esa instancia. Visual Studio no cargará ninguna instancia experimental adicional a menos que el nombre del directorio se cambie a la ubicación predeterminada.

Visual Studio no tiene acceso al registro del sistema cuando abre la instancia experimental. Esto difiere de las versiones anteriores de Visual Studio, que usaban una versión experimental del subárbol del Registro.

La utilidad **CreateExpInstance** reemplaza la utilidad **VsRegEx**.

En el ejemplo siguiente se restablece la instancia experimental predeterminada de Visual Studio:

```
CreateExpInstance.exe /Reset /VSInstance-14.0 /RootSuffix-Exp
```

Vea también

- [VSPackages](#)

Herramientas de tema de color

06/04/2020 • 2 minutes to read • [Edit Online](#)

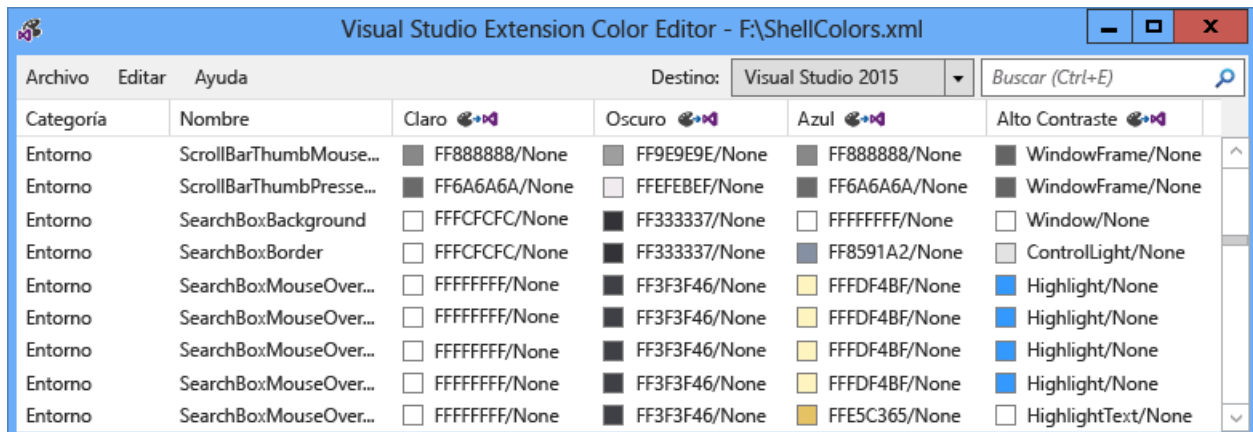
El SDK de Visual Studio proporciona varias herramientas para ayudarle a crear y administrar temas de color.

- La herramienta Editor de [colores VSIX](#) puede crear y editar colores personalizados para Visual Studio.
- La herramienta [VSIX Color Compiler](#) convierte los archivos `.xm` de tema de Visual Studio existentes en archivos `.pkgdef` para su extensión.

Editor de colores de VSIX

06/04/2020 • 18 minutes to read • [Edit Online](#)

La herramienta Editor de colores de extensión de Visual Studio puede crear y editar colores personalizados para Visual Studio. La herramienta también puede generar claves de recursos de tema para que los colores se puedan utilizar en el código. Esta herramienta es útil para crear colores para una extensión de Visual Studio que admite la creación de temas. Esta herramienta puede abrir archivos .pkgdef y .xml. Los temas de Visual Studio (archivos .vstheme) se pueden usar con el Editor de color es un archivo de extensión de Visual Studio cambiando la extensión de archivo a .xml. Además, los archivos .vstheme se pueden importar a un archivo .xml actual.



Archivos de definición del paquete

Los archivos de definición de paquetes (.pkgdef) son los archivos que definen temas. Los colores en sí se almacenan en archivos .xml de color de tema, que se compilan en un archivo .pkgdef. Los archivos .pkgdef se implementan en ubicaciones de búsqueda de Visual Studio, se procesan en tiempo de ejecución y se combinan para definir temas.

Fichas de color

Un token de color se compone de cuatro elementos:

- **Nombre de la categoría:** Agrupación lógica para un conjunto de colores. Use un nombre de categoría existente si ya hay colores que son específicos del elemento de interfaz de usuario deseado o un grupo de elementos de interfaz de usuario.
- **Nombre del token:** Un nombre descriptivo para el token de color y los conjuntos de tokens. Los conjuntos incluyen nombres de token de fondo y de primer plano (texto), así como todos sus estados, y estos deben tener un nombre para que sea fácil identificar los pares y los estados a los que se aplican.
- **Valores de color (o tonos):** Necesario para cada tema de color. Cree siempre valores de color de fondo y texto en pares. Los colores se emparejan para el fondo/primer plano para que el color del texto (en primer plano) siempre sea legible con respecto al color de fondo en el que se dibuja. Estos colores están vinculados y se usarán juntos en la interfaz de usuario. Si el fondo no está pensado para su uso con texto, no defina un color de primer plano.
- **Nombre del color del sistema:** Para su uso en pantallas de alto contraste.

Cómo utilizar la herramienta

En la medida de lo posible y, en su caso, los colores existentes de Visual Studio deben reutilizarse en lugar de crear otros nuevos. Sin embargo, para los casos en los que no se definen colores adecuados, se deben crear colores

personalizados para mantener una temática de extensión compatible.

Creación de nuevas fichas de color

Para crear colores personalizados mediante el Editor de colores de extensión de Visual Studio, siga estos pasos:

1. Determine la categoría y los nombres de token para los nuevos tokens de color.
2. Elija los tonos que utilizará el elemento de interfaz de usuario para cada tema y el color del sistema para Contraste alto.
3. Utilice el editor de color para crear nuevos tokens de color.
4. Use los colores de una extensión de Visual Studio.
5. Pruebe los cambios en Visual Studio.

Paso 1: Determine la categoría y los nombres de token para los nuevos tokens de color.

El esquema de nomenclatura preferido para un VSColor es **[Categoría] [Tipo de interfaz de usuario] [Estado]**. No utilice la palabra "color" en los nombres vscolor, ya que es redundante.

Los nombres de categoría proporcionan agrupaciones lógicas y deben definirse de la forma más estrecha posible. Por ejemplo, el nombre de una sola ventana de herramientas podría ser un nombre de categoría, pero el nombre de toda una unidad de negocio o equipo de proyecto no lo es. Agrupar las entradas en categorías ayuda a evitar confusiones entre colores con el mismo nombre.

Un nombre de token debe indicar claramente el tipo de elemento y las situaciones, o "estado", para las que se aplicará el color. Por ejemplo, el **[tipo de interfaz de usuario]** de una sugerencia de datos activa podría denominarse **"DataTip"** y el **[State]** podría llamarse **"Active,"** dando como resultado un nombre de color de **"DataTipActive."** Dado que las sugerencias de datos tienen texto, es necesario definir un color de primer plano y un color de fondo. Mediante el uso de un emparejamiento de fondo/primer plano, el editor de color creará automáticamente los colores **"DataTipActive"** para el fondo y **"DataTipActiveText"** para el primer plano.

Si la parte de la interfaz de usuario solo tiene un estado, se puede omitir la parte **[State]** del nombre. Por ejemplo, si un cuadro de búsqueda tiene un borde y no hay ningún cambio de estado que afecte al color del borde, el nombre del token de color del borde simplemente se puede llamar **"SearchBoxBorder."**

Algunos nombres de estado comunes incluyen:

- Active
- Inactivo
- MouseOver
- Mousedown
- Seleccionado
- Con foco

Ejemplos de algunos nombres de token para partes de un control de elemento de lista:

- ListItem
- ListItemBorder
- ListItemMouseOver
- ListItemMouseOverBorder

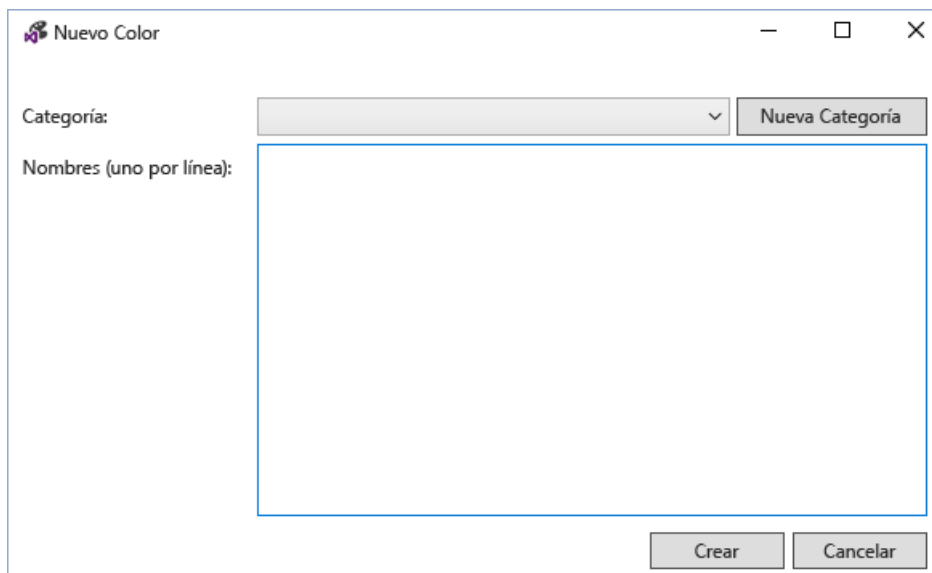
- ListItemSelected
- ListItemSelectedBorder
- ListItemDisabled
- ListItemDisabledBorder

Paso 2: Elija los tonos que el elemento de interfaz de usuario usará para cada tema y el color del sistema para contraste alto.

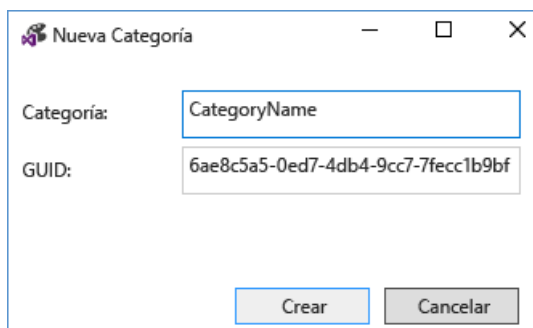
Al elegir colores personalizados para la interfaz de usuario, seleccione un elemento de interfaz de usuario existente similar y use sus colores como base. Los colores de los elementos de interfaz de usuario in-the-box se han revisado y realizado pruebas, por lo que se verán apropiados y se comportarán correctamente en todos los temas.

Paso 3: Utilice el editor de color para crear nuevos tokens de color.

Inicie el editor de color y abra o cree un nuevo archivo .xml de colores de tema personalizados. Seleccione **Editar > Nuevo color** en el menú. Esto abre un cuadro de diálogo para especificar la categoría y uno o más nombres para las entradas de color dentro de esa categoría:



Seleccione una categoría existente o seleccione **Nueva categoría** para crear una nueva categoría. Se abrirá otro cuadro de diálogo, creando un nuevo nombre de categoría:



La nueva categoría estará disponible en el menú desplegable **Categoría de nuevo color**. Después de elegir una categoría, escriba un nombre por línea para cada nuevo token de color y seleccione "Crear" cuando haya terminado:

Nuevo Color

Categoría:

CategoryName

Nueva Categoría

Nombres (uno por línea):

ColorName1

ColorName2

ColorName3

ColorName4

Crear

Cancelar

Los valores de color se muestran en pares de fondo/primer plano, con "Ninguno" que indica que el color no se ha definido. Nota: si un color no tiene un par de color de texto/color de fondo, solo es necesario definir el fondo.

CategoryName	ColorName1	☒ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno
CategoryName	ColorName2	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno
CategoryName	ColorName3	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno
CategoryName	ColorName4	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno	☐ Ninguno/ninguno

Para editar un token de color, seleccione una entrada de color para el tema (columna) de ese token. Agregue el valor de color escribiendo un valor de color hexadecimal en formato ARGB de 8 dígitos, introduciendo un nombre de color del sistema en la celda o utilizando el menú desplegable para seleccionar el color deseado a través de un conjunto de controles deslizantes de color o una lista de colores del sistema.

Categoría	Nombre	Claro	Oscuro	Azul
CategoryName	ColorName1	☐ 00FFFFFF/Ninguno	☐ Ninguno/Ninguno	☐ Ninguno/Ninguno
CategoryName	ColorName2			
CategoryName	ColorName3			
CategoryName	ColorName4			

Fondo: Personalizado

H

S

L

R

N

B

A

149

43

56

96

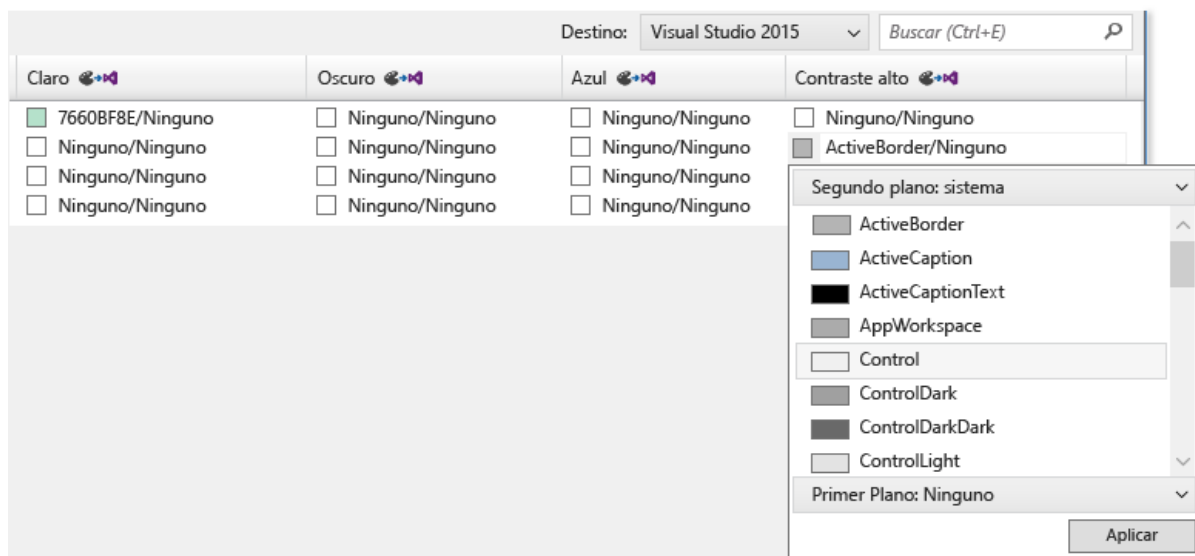
191

142

118

Primer Plano: Ninguno

Aplicar



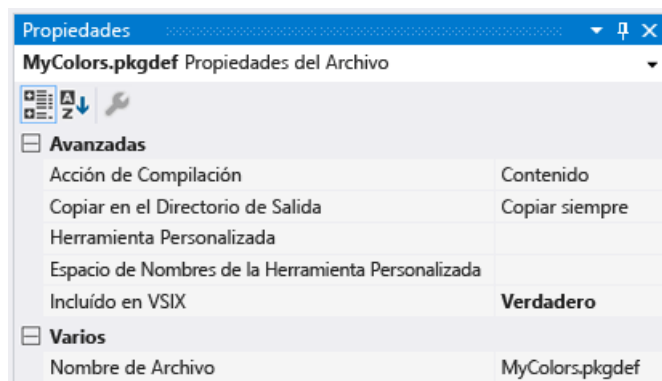
Para los componentes que no necesitan mostrar texto, introduzca solo un valor de color: el color de fondo. De lo contrario, introduzca valores para el fondo y el color del texto, separados por una barra diagonal.

Al introducir valores para Contraste alto, escriba nombres de color válidos del sistema de Windows. No introduzca valores ARGB codificados de forma rígida. Puede ver una lista de nombres de color es válidos del sistema seleccionando "Fondo: Sistema" o "Primer plano: Sistema" en los menús desplegables de valores de color. Al crear elementos que tienen componentes de texto, utilice el par de colores correcto del sistema de fondo/texto o el texto podría ser ilegible.

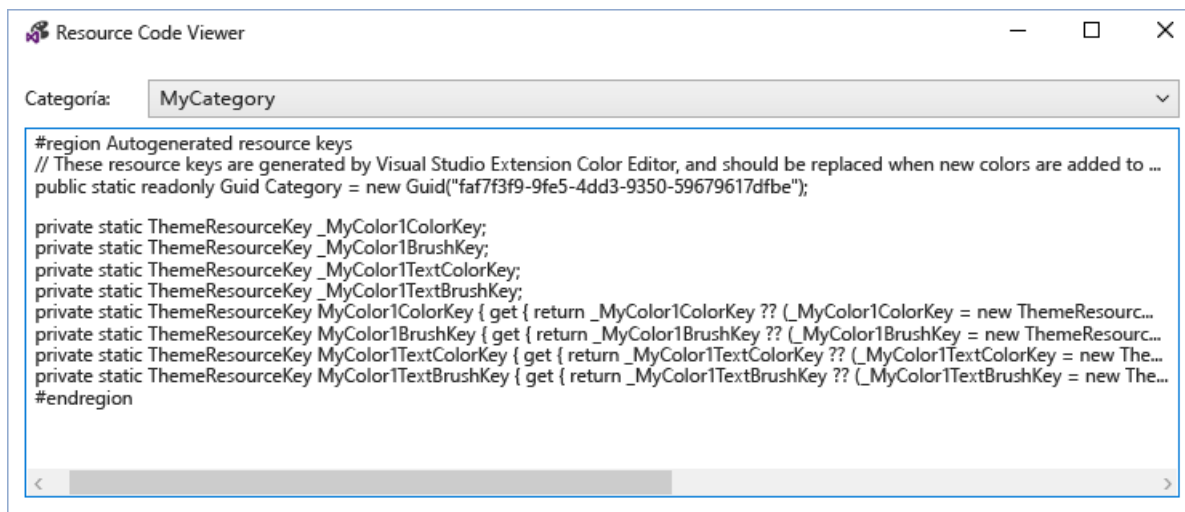
Cuando termine de crear, configurar y editar los tokens de color, guárdelos en el formato .xml o .pkgdef deseado. Los tokens de color sin fondo ni conjunto de primer plano se guardarán como colores vacíos en formato .xml, pero se descartarán en formato .pkgdef. Un cuadro de diálogo le advertirá de la posible pérdida de color si intenta guardar colores vacíos en un archivo .pkgdef.

Paso 4: Use los colores en una extensión de Visual Studio.

Después de definir los nuevos tokens de color, incluya el archivo .pkgdef en el archivo de proyecto con "Build Action" establecido en "Content" e "Include in VSIX" establecido en "True."



En el Editor de colores de extensión de Visual Studio, elija > archivo Ver código de recursos para ver el código que se usa para tener acceso a los colores personalizados en la interfaz de usuario basada en WPF.



Incluya este código en una clase estática en el proyecto. Una referencia a **Microsoft.VisualStudio.Shell.< VSVersion>.0.dll** debe agregarse al proyecto para usar el tipo **ThemeResourceKey**.

```
namespace MyCustomColors
{
    public static class MyCategory
    {
        #region Autogenerated resource keys
        // These resource keys are generated by Visual Studio Extension Color Editor, and should be replaced
        when new colors are added to this category.
        public static readonly Guid Category = new Guid("faf7f3f9-9fe5-4dd3-9350-59679617dfbe");

        private static ThemeResourceKey _MyColor1ColorKey;
        private static ThemeResourceKey _MyColor1BrushKey;
        private static ThemeResourceKey _MyColor1TextColorKey;
        private static ThemeResourceKey _MyColor1TextBrushKey;
        public static ThemeResourceKey MyColor1ColorKey { get { return _MyColor1ColorKey ?? (_MyColor1ColorKey = new ThemeResourc...
        public static ThemeResourceKey MyColor1BrushKey { get { return _MyColor1BrushKey ?? (_MyColor1BrushKey = new ThemeResourc...
        public static ThemeResourceKey MyColor1TextColorKey { get { return _MyColor1TextColorKey ?? (_MyColor1TextColorKey = new The...
        private static ThemeResourceKey MyColor1TextBrushKey { get { return _MyColor1TextBrushKey ?? (_MyColor1TextBrushKey = new The...
        #endregion
    }
}
```

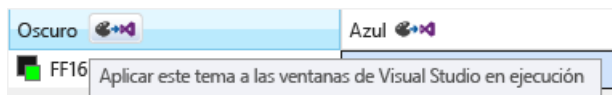
Esto permite el acceso a los colores en el código XAML y permite que la interfaz de usuario responda a los cambios de tema.

```
<UserControl x:Class="NewTestProject.TestPackageControl" Name="MyToolWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:ns="clr-namespace:MyCustomColors">
    <Grid>
        <TextBlock Background="{DynamicResource {x:Static ns:MyCategory.MyColor1BrushKey}}"
            Foreground="{DynamicResource {x:Static ns:MyCategory.MyColor1TextColorKey}}"
            >Sample Text</TextBlock>

    </Grid>
</UserControl>
```

Paso 5: pruebe los cambios en Visual Studio.

El editor de color puede aplicar temporalmente tokens de color a las instancias en ejecución de Visual Studio para ver los cambios en vivo en los colores sin volver a generar el paquete de extensión. Para ello, haga clic en el botón "Aplicar este tema a las ventanas de Visual Studio en ejecución" ubicado en el encabezado de cada columna de tema. Este tema temporal desaparecerá cuando se cierre el Editor de colores VSIX.



Para que los cambios sean permanentes, vuelva a generar y vuelva a implementar la extensión de Visual Studio después de agregar los nuevos colores al archivo .pkgdef y escribir el código que usará esos colores. Volver a generar la extensión de Visual Studio combinará los valores del Registro para los nuevos colores en el resto de los temas. A continuación, vuelva a iniciar Visual Studio, vea la interfaz de usuario y compruebe que los nuevos colores aparecen como se esperaba.

Notas

Esta herramienta está diseñada para usarse para crear colores personalizados para los temas de Visual Studio preexistentes o para editar los colores de un tema de Visual Studio personalizado. Para crear temas personalizados completos de Visual Studio, descargue la extensión Editor de temas de color de [Visual Studio](#) desde la Galería de extensiones de Visual Studio.

Salida de ejemplo

Salida de color XML

El archivo .xml generado por la herramienta será similar a esto:

```
<Themes>
  <Theme Name="Light" GUID="{de3dbbcd-f642-433c-8353-8f1df4370aba}">
    <Category Name="CategoryName" GUID="{eee9d521-dac2-48d9-9a5e-5c625ba2040c}">
      <Color Name="ColorName1">
        <Background Type="CT_RAW" Source="FFFFFFFF" />
      </Color>
      <Color Name="ColorName2">
        <Background Type="CT_RAW" Source="FFFFFFFF" />
        <Foreground Type="CT_RAW" Source="FF000000" />
      </Color>
      <Color Name="ColorName3">
        <Background Type="CT_RAW" Source="FFFF0000" />
      </Color>
      <Color Name="ColorName4">
        <Background Type="CT_RAW" Source="FF000088" />
        <Foreground Type="CT_RAW" Source="FFFFFFFF" />
      </Color>
    </Category>
  </Theme>
  <Theme Name="Dark" GUID="{1ded0138-47ce-435e-84ef-9ec1f439b749}">...</Theme>
  <Theme Name="Blue" GUID="{a4d6a176-b948-4b29-8c66-53c97a1ed7d0}">...</Theme>
  <Theme Name="HighContrast" GUID="{a5c004b4-2d4b-494e-bf01-45fc492522c7}">...</Theme>
</Themes>
```

Salida de color PKGDEF

El archivo .pkgdef generado por la herramienta será similar a esto:

```
[\$RootKey$\Themes\{de3dbbcd-f642-433c-8353-8f1df4370aba}\CategoryName]
"Data"=hex:78,00,00,00,0b,00,00,00,01,00,00,00,21,d5,e9,ee,c2,da,d9,48,9a,5e,5c,62,5b,a2,04,0c,04,00,00,00,0a,
00,00,00,43,6f,6c,6f,72,4e,61,6d,65,31,01,ff,ff,ff,ff,00,0a,00,00,00,43,6f,6c,6f,72,4e,61,6d,65,32,01,ff,ff,ff
,ff,01,00,00,00,ff,0a,00,00,00,43,6f,6c,6f,72,4e,61,6d,65,33,01,ff,00,00,ff,00,0a,00,00,00,43,6f,6c,6f,72,4e,6
1,6d,65,34,01,00,00,88,ff,01,ff,ff,ff,ff
[\$RootKey$\Themes\{1ded0138-47ce-435e-84ef-9ec1f439b749}\CategoryName]
"Data"=hex:...
[\$RootKey$\Themes\{a4d6a176-b948-4b29-8c66-53c97a1ed7d0}\CategoryName]
"Data"=hex:...
[\$RootKey$\Themes\{a5c004b4-2d4b-494e-bf01-45fc492522c7}\CategoryName]
"Data"=hex:...
```

Contenedor de claves de recursos de CTM

Las claves de recursos de color generadas por la herramienta serán similares a esto:

```
namespace MyNamespace
{
    public static class MyColors
    {
        #region Autogenerated resource keys
        // These resource keys are generated by Visual Studio Extension Color Editor, and should be replaced
        when new colors are added to this category.

        public static string ColorName1ColorKey { get { return "ColorName1ColorKey"; } }
        public static string ColorName1BrushKey { get { return "ColorName1BrushKey"; } }

        public static string ColorName2ColorKey { get { return "ColorName2ColorKey"; } }
        public static string ColorName2BrushKey { get { return "ColorName2BrushKey"; } }
        public static string ColorName2TextColorKey { get { return "ColorName2TextColorKey"; } }
        public static string ColorName2TextBrushKey { get { return "ColorName2TextBrushKey"; } }

        public static string ColorName3ColorKey { get { return "ColorName4ColorKey"; } }
        public static string ColorName3BrushKey { get { return "ColorName4BrushKey"; } }
        public static string ColorName3TextColorKey { get { return "ColorName4TextColorKey"; } }
        public static string ColorName3TextBrushKey { get { return "ColorName4TextBrushKey"; } }
        #endregion
    }
}
```

Contenedor de diccionario de recursos WPF

Las claves **ResourceDictionary** de color generadas por la herramienta serán similares a esto:

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:colors="clr-namespace:MyNamespace">

    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName1BrushKey}" Color="#FFFFFF" />
    <Color x:Key="{x:Static colors:MyColors.ColorName1ColorKey}" A="255" R="255" G="255" B="255" />

    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName2BrushKey}" Color="#FFFFFF" />
    <Color x:Key="{x:Static colors:MyColors.ColorName2ColorKey}" A="255" R="255" G="255" B="255" />
    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName2TextBrushKey}" Color="#FF0000" />
    <Color x:Key="{x:Static colors:MyColors.ColorName2TextColorKey}" A="255" R="0" G="0" B="0" />

    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName3BrushKey}" Color="#FFF000" />
    <Color x:Key="{x:Static colors:MyColors.ColorName3ColorKey}" A="255" R="255" G="0" B="0" />

    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName4BrushKey}" Color="#FF0008" />
    <Color x:Key="{x:Static colors:MyColors.ColorName4ColorKey}" A="255" R="0" G="0" B="136" />
    <SolidColorBrush x:Key="{x:Static colors:MyColors.ColorName4TextBrushKey}" Color="#FFFFFF" />
    <Color x:Key="{x:Static colors:MyColors.ColorName4TextColorKey}" A="255" R="255" G="255" B="255" />
</ResourceDictionary>
```

Compilador de colores de VSIX

06/04/2020 • 6 minutes to read • [Edit Online](#)

La herramienta Visual Studio Extension Color Compiler es una aplicación de consola que toma un archivo .xml que representa colores para los temas de Visual Studio existentes y lo encubre en un archivo .pkgdef para que esos colores se puedan usar en Visual Studio. Dado que es fácil comparar las diferencias entre los archivos .xml, esta herramienta es útil para administrar colores personalizados en el control de código fuente. También se puede enlazar a entornos de compilación para que la salida de la compilación sea un archivo .pkgdef válido.

Esquema XML de tema

Un archivo .xml de tema completo tiene este aspecto:

```
<Themes>
  <!--one or Theme elements -->
  <Theme>
    <!-- one or more Category elements -->
    <Category>
      <!-- one or more Color elements -->
      <Color>
        <!-- zero or one Background element -->
        <Background />
        <!-- zero or one Foreground element -->
        <Foreground />
      </Color>
    </Category>
  </Theme>
</Themes>
```

Tema

El <elemento Theme> define un tema completo. Un tema debe contener <al menos un elemento de categoría>. Los elementos temáticos se definen así:

```
<Theme Name="name" GUID="guid">
  <!-- one or more Category elements -->
</Theme>
```

Atributo	Definición
Nombre	[Requerido] El nombre del tema
GUID	[Requerido] GUID del tema (debe coincidir con el formato GUID)

Al crear colores personalizados para Visual Studio, esos colores deben definirse para los siguientes temas. Si no existen colores para un tema determinado, Visual Studio intenta cargar los colores que faltan desde el tema Luz.

Nombre del tema	GUID temático

Ligero	•de3dbbcd-f642-433c-8353-8f1df4370aba
Oscuro	N.o 1ded0138-47ce-435e-84ef-9ec1f439b749?
Azul	.a4d6a176-b948-4b29-8c66-53c97a1ed7d0
Contraste alto	.a4d6a176-b948-4b29-8c66-53c97a1ed7d0

Categoría

El <elemento Category> define una colección de colores en un tema. Los nombres de categoría proporcionan agrupaciones lógicas y deben definirse de la forma más estrecha posible. Una categoría debe contener <al menos un elemento Color>. Los elementos de categoría se definen así:

```
<Category Name="name" GUID="guid">
  <!-- one or more Color elements -->
</Category>
```

Atributo	Definición
Nombre	[Requerido] El nombre de la categoría
GUID	[Requerido] GUID de la categoría (debe coincidir con el formato GUID)

Color

El <elemento Color> define un color para un componente o estado de la interfaz de usuario. El esquema de nomenclatura preferido para un color es [Tipo de interfaz de usuario] [Estado]. No utilice la palabra "color", ya que es redundante. Un color debe indicar claramente el tipo de elemento y las situaciones, o "estado", para las que se aplicará el color. Un color no debe estar vacío y debe <contener uno <o ambos de un elemento de fondo> y> de primer plano. Los elementos de color se definen así:

```
<Color Name="name">
  <Background /> <!-- zero or one Background element -->
  <Foreground /> <!-- zero or one Foreground element -->
</Color>
```

Atributo	Definición
Nombre	[Requerido] El nombre del color

Antecedentes y/o Primer plano

Los <elementos <de> de fondo y> de primer plano definen el valor y el tipo de un color para el fondo o el primer plano de un elemento de interfaz de usuario. Estos elementos no tienen hijos.

```
<Background Type="type" Source="int" />
<Foreground Type="type" Source="int" />
```


Atributo	Definición
Tipo	[Requerido] El tipo del color. Puede tener uno de los valores siguientes: <i>CT_INVALID</i> : El color no es válido o no está establecido. <i>CT_RAW</i> : Un valor ARGB sin procesar. <i>CT_COLORINDEX</i> : NO USAR. <i>CT_SYSCOLOR</i> : Un color del sistema Windows de SysColor. <i>CT_VSCOLOR</i> : Un color de Visual Studio de __VSSYSCOLOREX. <i>CT_AUTOMATIC</i> : El color automático. <i>CT_TRACK_FOREGROUND</i> : NO USAR. <i>CT_TRACK_BACKGROUND</i> : NO USAR.
Source	[Requerido] El valor del color representado en hexadecimal

Todos los valores admitidos por la enumeración `__VSCOLORTYPE` son compatibles con el esquema en el `Type` atributo. Sin embargo, le recomendamos que utilice solo `CT_RAW` y `CT_SYSCOLOR`.

Todos juntos

Este es un ejemplo sencillo de un archivo `.xml` de tema válido:

```
<Themes>
  <Theme Name="Light" GUID="{de3dbbcd-f642-433c-8353-8f1df4370aba}">
    <Category Name="MyCategory" GUID="{0A96238B-70CE-4479-9170-EECEAA3FCD58}">
      <Color Name="MyActiveBorder">
        <Background Type="CT_RAW" Source="FFCCCEDB" />
      </Color>
    </Category>
  </Theme>
</Themes>
```

Cómo utilizar la herramienta

Sintaxis

El archivo `<XML VsixColorCompiler <> <archivo PkgDef> > Args opcionales`

Argumentos

Cambiar nombre	Notas	Requerido u Opcional
Sin nombre (archivo <code>.xml</code>)	Este es el primer parámetro sin nombre y es la ruta de acceso al archivo XML que se va a convertir.	Obligatorio

Sin nombre (archivo .pkgdef)	Este es el segundo parámetro sin nombre y es la ruta de salida para el archivo .pkgdef generado. Predeterminado: <Nombre de archivo XML>.pkgdef	Opcional
/noLogo	Al establecer esta marca, se detiene la impresión de la información del producto y de los derechos de autor.	Opcional
/?	Imprima la información de ayuda.	Opcional
/help	Imprima la información de ayuda.	Opcional

Ejemplos

- VsixColorCompiler D:-xml-colors.xml D:-pkgdef-colors.pkgdef
- VsixColorCompiler D:-xml-colors.xml /noLogo

Notas

- Esta herramienta requiere que se instale la última versión del tiempo de ejecución VC++.
- Solo se admiten archivos individuales. No se admite la conversión masiva a través de rutas de carpeta.

Salida de ejemplo

El archivo .pkgdef generado por la herramienta será similar a las siguientes claves:

```
[${RootKey}\Themes\{de3dbbcd-f642-433c-8353-8f1df4370aba}\Environment]
"Data"=hex:3a,00,00,00,0b,00,00,00,01,00,00,00,c3,d9,4e,62,fd,bd,fa,41,96,c3,7c,82,4e,a3,2e,3d,01,00,00,00,0c,
00,00,00,41,63,74,69,76,65,42,6f,72,64,65,72,01,cc,ce,db,ff,01,33,31,24,ff

[${RootKey}\Themes\{de3dbbcd-f642-433c-8353-8f1df4370aba}\TreeView]
"Data"=hex:38,00,00,00,0b,00,00,00,01,00,00,00,8e,f0,ec,92,13,8b,f4,4c,99,e9,ae,26,92,38,21,85,01,00,00,00,0a,
00,00,00,42,61,63,6b,67,72,6f,75,6e,64,01,f5,f5,f5,ff,01,1e,1e,1e,ff
```

Herramientas de servicio de imágenes

06/04/2020 • 2 minutes to read • [Edit Online](#)

El SDK de VS incluye varias herramientas para ayudar a los generadores de extensiones a usar el servicio de imágenes de Visual Studio.

- La herramienta [Manifiesto de recursos](#) toma una lista de recursos de imagen (archivos .png o .xaml) y genera un archivo de manifiesto de imagen.
- La herramienta [Manifiesto a código](#) toma un archivo de manifiesto de imagen y genera un archivo contenedor para usarlo en archivos C++, C, VB o .vsct.
- El Visor de biblioteca de [imágenes](#) puede cargar, editar y buscar manifiestos de imagen.

Manifest from Resources

06/04/2020 • 8 minutes to read • [Edit Online](#)

La herramienta Manifiesto desde recursos es una aplicación de consola que toma una lista de recursos de imagen (archivos .png o .xaml) y genera un archivo .imagemanifest que permite usar esas imágenes con Visual Studio Image Service. Además, esta herramienta se puede utilizar para agregar imágenes a un .imagemanifest existente. Esta herramienta es útil para agregar alto PPP y compatibilidad con temas para imágenes a una extensión de Visual Studio. El archivo .imagemanifest generado debe incluirse e implementarse como parte de una extensión de Visual Studio (.vsix).

Cómo utilizar la herramienta

Sintaxis

ManifestFromResources /resources:<Dir1>; <Img1> /assembly:< <AssemblyName>> de args opcionales

Argumentos

Cambiar nombre	Notas	Requerido u Opcional
/recursos	Una lista delimitada por punto y coma de imágenes o directorios. Esta lista siempre debe contener la lista completa de imágenes que estarán en el manifiesto. Si solo se proporciona una lista parcial, se perderán las entradas no incluidas. Si un archivo de recursos determinado es una tira de imagen, la herramienta lo dividirá en imágenes independientes antes de agregar cada subimagen al manifiesto. Si la imagen es un archivo .png, le recomendamos que formatee el nombre <de esta manera para que la herramienta pueda rellenar los atributos correctos para la imagen: Nombre>. <Ancho>. <Altura>.png.	Obligatorio
/ensamblaje	El nombre del ensamblado administrado (sin incluir la extensión) o la ruta de acceso en tiempo de ejecución del ensamblado nativo que hospeda los recursos (en relación con la ubicación en tiempo de ejecución del manifiesto).	Obligatorio

/manifest	El nombre que se va a asignar al archivo .imagemanifest generado. Esto también puede incluir una ruta de acceso absoluta o relativa para crear el archivo en una ubicación diferente. El nombre predeterminado coincide con el nombre del ensamblado. Predeterminado: <Directorio \ actual> > <Assembly .imagemanifest	Opcional
/guidName	El nombre que se va a asignar al símbolo GUID de todas las imágenes del manifiesto generado. Predeterminado: AssetsGuid	Opcional
/rootPath	La ruta de acceso raíz que debe quitarse antes de crear URI de recursos administrados. (Esta marca es para ayudar con los casos en los que la herramienta obtiene la ruta de ACCESO URI relativa incorrecta, lo que hace que los recursos no se carguen.) Predeterminado: <directorio actual>	Opcional
/recursivo	Al establecer esta marca, la herramienta busca recursivamente cualquier directorio en el argumento /resources. Si se omite esta marca, se realizará una búsqueda de directorios de solo nivel superior.	Opcional
/isNative	Establezca esta marca cuando el argumento de ensamblado sea una ruta de acceso para un ensamblado nativo. Omita esta marca cuando el argumento de ensamblado sea el nombre de un ensamblado administrado. (Consulte la sección Notas para obtener información adicional sobre esta marca.)	Opcional
/newGuids	Al establecer esta marca, se indica a la herramienta que cree un nuevo valor para el símbolo GUID de las imágenes en lugar de combinar el del manifiesto existente.	Opcional
/newIds	Al establecer esta marca, se indica a la herramienta que cree nuevos valores de símbolo de identificador para cada imagen en lugar de combinar valores del manifiesto existente.	Opcional
/noLogo	Al establecer esta marca, se detiene la impresión de la información del producto y de los derechos de autor.	Opcional

/?	Imprima la información de ayuda.	Opcional
/help	Imprima la información de ayuda.	Opcional

Ejemplos

- ManifestFromResources /resources:D:-Images /assembly:My.Assembly.Name /isNative
- ManifestFromResources /resources:D:-Imágenes-Image1.png;D:-Imágenes-Image1.xml /assembly:My.Assembly.Name /manifest:MyImageManifest.imagemanifest
- ManifestFromResources /resources:D:-Imágenes-Image1.png;D:-Imágenes-Image1.xml /assembly:My.Assembly.Name /guidName:MyImages /newGuids /newIds

Notas

- La herramienta solo admite archivos .png y .xml. Cualquier otro tipo de imagen o archivo se ignorará. Se genera una advertencia para todos los tipos no admitidos encontrados al analizar los recursos. Si no se encuentra ninguna imagen compatible cuando la herramienta haya terminado de analizar los recursos, se generará un error
- Siguiendo el formato sugerido para las imágenes .png, la herramienta establecerá el valor de tamaño/dimensión para el .png en el tamaño especificado por el formato, incluso si difiere del tamaño real de la imagen.
- El formato de anchura/altura se puede omitir para las imágenes .png, pero la herramienta leerá el ancho/alto real de la imagen y los usará para el valor de tamaño/dimensión de la imagen.
- Al ejecutar esta herramienta en la misma tira de imagen varias veces para el mismo .imagemanifest, se producirán entradas de manifiesto duplicadas, ya que la herramienta intenta dividir la tira de imagen en imágenes independientes y agregarlas al manifiesto existente.
- La combinación (omitiendo /newGuids o /newIds) solo debe realizarse para los manifiestos generados por la herramienta. Es posible que los manifiestos que se han personalizado o generado a través de otros medios no se combinen correctamente.
- Es posible que los manifiestos que se generan para ensamblados nativos deban editarse a mano después de la generación para que los símbolos de identificador coincidan con los identificadores de recursos del archivo .rc del ensamblado nativo.

Salida de ejemplo

Manifiesto de imagen simple

Un manifiesto de imagen será similar a este archivo .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file was generated by the ManifestFromResources tool.-->
<!-- Version: 14.0.15197 -->
<ImageManifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/VisualStudio/ImageManifestSchema/2014">
  <Symbols>
    <String Name="Resources" Value="/My.Assembly.Name;Component/Resources/Images" />
    <Guid Name="AssetsGuid" Value="{fb41b7ef-6587-480c-aa27-5b559d42cfc9}" />
    <ID Name="MyImage" Value="0" />
  </Symbols>
  <Images>
    <Image Guid="$(AssetsGuid)" ID="$(MyImage)">
      <Source Uri="$(Resources)/Xaml/MyImage.xaml" />
      <Source Uri="$(Resources)/Png/MyImage.16.16.png">
        <Size Value="16" />
      </Source>
    </Image>
  </Images>
  <ImageLists />
</ImageManifest>
```

Manifiesto de imagen para una tira de imagen

Un manifiesto de imagen para una tira de imagen será similar a este archivo .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file was generated by the ManifestFromResources tool.-->
<!-- Version: 14.0.15197 -->
<ImageManifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/VisualStudio/ImageManifestSchema/2014">
  <Symbols>
    <String Name="Resources" Value="/My.Assembly.Name;Component/Resources/ImageStrip" />
    <Guid Name="AssetsGuid" Value="{fb41b7ef-6587-480c-aa27-5b559d42cfc9}" />
    <ID Name="MyImageStrip_0" Value="1" />
    <ID Name="MyImageStrip_1" Value="2" />
    <ID Name="MyImageStrip" Value="3" />
  </Symbols>
  <Images>
    <Image Guid="$(AssetsGuid)" ID="$(MyImageStrip_0)">
      <Source Uri="$(Resources)/MyImageStrip_0.png">
        <Size Value="16" />
      </Source>
    </Image>
    <Image Guid="$(AssetsGuid)" ID="$(MyImageStrip_1)">
      <Source Uri="$(Resources)/MyImageStrip_1.png">
        <Size Value="16" />
      </Source>
    </Image>
  </Images>
  <ImageLists>
    <ImageList Guid="$(AssetsGuid)" ID="$(MyImageStrip)">
      <ContainedImage Guid="$(AssetsGuid)" ID="$(MyImageStrip_0)" />
      <ContainedImage Guid="$(AssetsGuid)" ID="$(MyImageStrip_1)" />
    </ImageList>
  </ImageLists>
</ImageManifest>
```

Manifiesto de imagen para recursos de imagen de ensamblado nativo

Un manifiesto de imagen para imágenes nativas será similar a este archivo .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file was generated by the ManifestFromResources tool.-->
<!-- Version: 14.0.15198 -->
<ImageManifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/VisualStudio/ImageManifestSchema/2014">
  <Symbols>
    <String Name="Resources" Value="..\Assembly\Folder\My.Assembly.Name" />
    <Guid Name="AssetsGuid" Value="{442d8739-efde-46a4-8f29-e3a1e5e7f8b4}" />
    <ID Name="MyImage1" Value="0" />
    <ID Name="MyImage2" Value="1" />
  </Symbols>
  <Images>
    <Image Guid="$(AssetsGuid)" ID="$(MyImage1)">
      <Source Uri="$(Resources)">
        <Size Value="16" />
        <NativeResource ID="$(MyImage1)" Type="PNG" />
      </Source>
    </Image>
    <Image Guid="$(AssetsGuid)" ID="$(MyImage2)">
      <Source Uri="$(Resources)">
        <Size Value="16" />
        <NativeResource ID="$(MyImage2)" Type="PNG" />
      </Source>
    </Image>
  </Images>
  <ImageLists />
</ImageManifest>
```


Manifest to Code

06/04/2020 • 7 minutes to read • [Edit Online](#)

La herramienta Manifiesto a código es una aplicación de consola que toma un archivo .imagemanifest para el servicio de imágenes de Visual Studio y genera un archivo contenedor o archivos para hacer referencia a los valores del manifiesto de imagen en archivos C++, C, VB o .vsct para las extensiones de Visual Studio. Esta herramienta genera archivos contenedor que se pueden usar para solicitar imágenes desde el servicio de imágenes de Visual Studio directamente o para pasar los valores de manifiesto a través de las API si el código no controla ninguna de sus propias interfaz de usuario y representación.

Cómo utilizar la herramienta

Sintaxis

ManifestToCode /manifest: archivo de manifiesto <de <imagen> /language: Code Language> Args opcionales>

Argumentos

Cambiar nombre	Notas	Requerido u Opcional
/manifest	La ruta de acceso al manifiesto de imagen que se usará para crear o actualizar el contenedor de código.	Obligatorio
/idioma	El idioma en el que se va a generar el contenedor de código. Valores válidos: CPP, C++, CS, CSharp, C, VB o VSCT Los valores no distinguen mayúsculas de minúsculas. Para la opción de lenguaje VSCT, se omiten las opciones /monikerClass, /classAccess y /namespace.	Obligatorio
/imageIdClass	El nombre de imageIdClass y el archivo asociado creado por la herramienta. Para la opción de idioma C++, solo se generan archivos .h. Predeterminado: <ruta de acceso del manifiesto> de MyImageIds. <Lang Ext>	Opcional
/monikerClass	El nombre de monikerClass y el archivo asociado creado por la herramienta. Para la opción de idioma C++, solo se generan archivos .h. Esto se omite para el lenguaje VSCT. Predeterminado: <Ruta de acceso del manifiesto>-MyMonikers. <Lang Ext>	Opcional

<code>/classAccess</code>	El modificador de acceso para el <code>imageIdClass</code> y el <code>monikerClass</code> . Asegúrese de que el modificador de acceso es válido para el idioma especificado. Esto se omite para la opción de lenguaje VSCT. Predeterminado: Público	Opcional
<code>/namespace</code>	Espacio de nombres definido en el contenedor de código. Esto se omite para la opción de lenguaje VSCT. '.' o '::' son separadores de espacio de nombres válidos, independientemente de la opción de idioma elegida. Predeterminado: <code>MyImages</code>	Opcional
<code>/noLogo</code>	Al establecer esta marca, se detiene la impresión de la información del producto y de los derechos de autor.	Opcional
<code>/?</code>	Imprima la información de ayuda.	Opcional
<code>/help</code>	Imprima la información de ayuda.	Opcional

Ejemplos

- `ManifestToCode /manifest:D:-MyManifest.imagemanifest /language:CSharp`
- `ManifestToCode /manifest:D:-MyManifest.imagemanifest /language:C++ /namespace:My::Namespace /imageIdClass:MyImageIds /monikerClass:MyMonikers /classAccess:friend`
- `ManifestToCode /manifest:D:-MyManifest.imagemanifest /language:VSCT /imageIdClass:MyImageIds`

Notas

- Se recomienda utilizar esta herramienta con manifiestos de imagen generados por la herramienta Manifiesto a partir de recursos.
- La herramienta solo examina las entradas de símbolos para generar los contenedores de código. Si un manifiesto de imagen no contiene símbolos, los contenedores de código generados estarán vacíos. Si hay una imagen o un conjunto de imágenes en el manifiesto de imagen que no utilizan símbolos, se excluirán del contenedor de código.

Salida de ejemplo

Envoltorios de C-

Un par de clases simples de ID de imagen y moniker de imagen para C- serán similares al código siguiente:

```
//-----
// <auto-generated>
//   This code was generated by the ManifestToCode tool.
//   Tool Version: 14.0.15198
// </auto-generated>
//-----

using System;

namespace MyImages
{
    public static class MyImageIds
    {
        public static readonly Guid AssetsGuid = new Guid("{442d8739-efde-46a4-8f29-e3a1e5e7f8b4}");

        public const int MyImage1 = 0;
        public const int MyImage2 = 1;
    }
}

//-----
// <auto-generated>
//   This code was generated by the ManifestToCode tool.
//   Tool Version: 14.0.15198
// </auto-generated>
//-----

using Microsoft.VisualStudio.Imaging.Interop;

namespace MyImages
{
    public static class MyMonikers
    {
        public static ImageMoniker MyImage1 { get { return new ImageMoniker { Guid = MyImageIds.AssetsGuid, Id = MyImageIds.MyImage1 }; } }
        public static ImageMoniker MyImage2 { get { return new ImageMoniker { Guid = MyImageIds.AssetsGuid, Id = MyImageIds.MyImage2 }; } }
    }
}
```

Envolturas C++

Un par de clases simples de ID de imagen y moniker de imagen para C++ serán similares al siguiente código:

```

//-----
// <auto-generated>
//     This code was generated by the ManifestToCode tool.
//     Tool Version: 14.0.15198
// </auto-generated>
//-----

#pragma once

#include <guiddef.h>

namespace MyImages {

class MyImageIds {
public:

    static const GUID AssetsGuid;

    static const int MyImage1 = 0;
    static const int MyImage2 = 1;

};

__declspec(selectany) const GUID MyImageIds::AssetsGuid = {0x442d8739,0xefde,0x46a4,
{0x8f,0x29,0xe3,0xa1,0xe5,0xe7,0xf8,0xb4}};

}
//-----
// <auto-generated>
//     This code was generated by the ManifestToCode tool.
//     Tool Version: 14.0.15198
// </auto-generated>
//-----

#pragma once

#include "ImageParameters140.h"
#include "MyImageIds.h"

namespace MyImages {

class MyMonikers {
public:

    static const ImageMoniker MyImage1;
    static const ImageMoniker MyImage2;

};

__declspec(selectany) const ImageMoniker MyMonikers::MyImage1 = { MyImageIds::AssetsGuid, MyImageIds::MyImage1
};
__declspec(selectany) const ImageMoniker MyMonikers::MyImage2 = { MyImageIds::AssetsGuid, MyImageIds::MyImage2
};

}

```

Contenedores de Visual Basic

Un par de clases simples de id de imagen y moniker de imagen para Visual Basic serán similares al código siguiente:

```

' -----
' <auto-generated>
'   This code was generated by the ManifestToCode tool.
'   Tool Version: 14.0.15198
' </auto-generated>
' -----

Imports System

Namespace MyImages

    Public Module MyImageIds

        Public Shared ReadOnly AssetsGuid As Guid = New Guid("{442d8739-efde-46a4-8f29-e3a1e5e7f8b4}")

        Public Const MyImage1 As Integer = 0
        Public Const MyImage2 As Integer = 1

    End Module

End Namespace
' -----
' <auto-generated>
'   This code was generated by the ManifestToCode tool.
'   Tool Version: 14.0.15198
' </auto-generated>
' -----

Imports Microsoft.VisualStudio.Imaging.Interop

Namespace MyImages

    Public Module MyMonikers

        Public Readonly Property MyImage1
            Get
                Return New ImageMoniker With {.Guid = MyImageIds.AssetsGuid, .Id = MyImageIds.MyImage1}
            End Get
        End Property

        Public Readonly Property MyImage2
            Get
                Return New ImageMoniker With {.Guid = MyImageIds.AssetsGuid, .Id = MyImageIds.MyImage2}
            End Get
        End Property

    End Module

End Namespace

```

Envoltura VSCT

Un conjunto de iDE de imagen para un archivo .vsct será similar a esto:

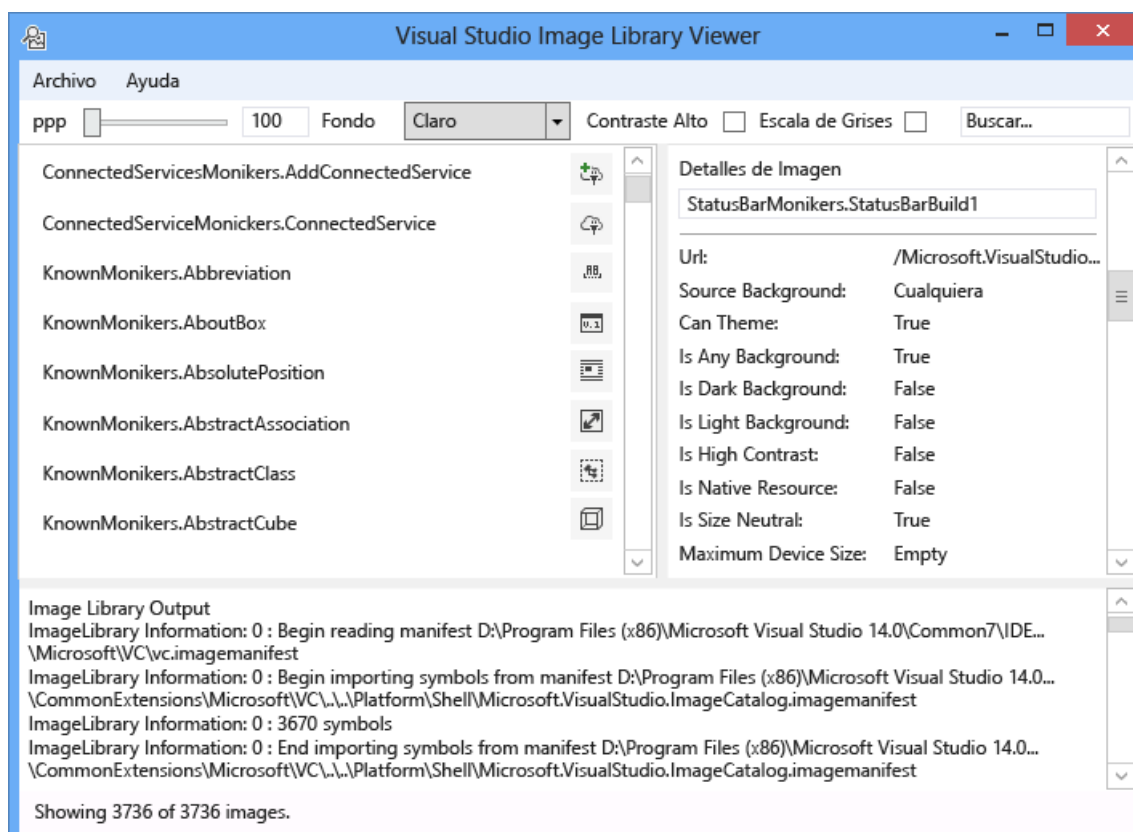
```
<?xml version='1.0' encoding='utf-8'?>
<!--
- [auto-generated]
    This code was generated by the ManifestToCode tool.
    Tool Version: 14.0.15198
- [/auto-generated]
-->
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable">
  <Symbols>
    <GuidSymbol name="AssetsGuid" value="{442d8739-efde-46a4-8f29-e3a1e5e7f8b4}">
      <IDSymbol name="MyImage1" value="0" />
      <IDSymbol name="MyImage2" value="1" />
    </GuidSymbol>
  </Symbols>
</CommandTable>
```

Visor de la biblioteca de imágenes

06/04/2020 • 18 minutes to read • [Edit Online](#)

La herramienta Visor de biblioteca de imágenes de Visual Studio puede cargar y buscar manifiestos de imagen, lo que permite al usuario manipularlos de la misma manera que Visual Studio. El usuario puede modificar el fondo, los tamaños, los PPP, el alto contraste y otros ajustes. La herramienta también muestra información de carga para cada manifiesto de imagen y muestra información de origen para cada imagen en el manifiesto de imagen. Esta herramienta es útil para:

1. Diagnóstico de errores
2. Asegurarse de que los atributos se establecen correctamente en manifiestos de imagen personalizados
3. Búsqueda de imágenes en el catálogo de imágenes de Visual Studio para que una extensión de Visual Studio pueda usar imágenes que se ajusten al estilo de Visual Studio



Apodo de imagen

Un moniker de imagen (o moniker para abreviar) es un par GUID:ID que identifica de forma única un recurso de imagen o un recurso de lista de imágenes en la biblioteca de imágenes.

Archivos de manifiesto de imagen

Los archivos de manifiesto de imagen (.imagemanifest) son archivos XML que definen un conjunto de activos de imagen, los monikers que representan esos activos y la imagen o imágenes reales que representan cada recurso. Los manifiestos de imagen pueden definir imágenes independientes o listas de imágenes para la compatibilidad con la interfaz de usuario heredada. Además, hay atributos que se pueden establecer en el activo o en las imágenes individuales detrás de cada activo para cambiar cuándo y cómo se muestran esos activos.

Esquema de manifiesto de imagen

Un manifiesto de imagen completo tiene este aspecto:

```
<ImageManifest>
  <!-- zero or one Symbols elements -->
  <Symbols>
    <!-- zero or more Guid, ID, or String elements -->
  </Symbols>
  <!-- zero or one Images elements -->
  <Images>
    <!-- zero or more Image elements -->
  </Images>
  <!-- zero or one ImageLists elements -->
  <ImageLists>
    <!-- zero or more ImageList elements -->
  </ImageLists>
</ImageManifest>
```

Symbols

Como ayuda de legibilidad y mantenimiento, el manifiesto de imagen puede utilizar símbolos para los valores de atributo. Los símbolos se definen así:

```
<Symbols>
  <Import Manifest="manifest" />
  <Guid Name="ShellCommandGuid" Value="8ee4f65d-bab4-4cde-b8e7-ac412abbda8a" />
  <ID Name="cmdidSaveAll" Value="1000" />
  <String Name="AssemblyName" Value="Microsoft.VisualStudio.Shell.UI.Internal" />
</Symbols>
```

Subelemento	Definición
Importar	Importa los símbolos del archivo de manifiesto dado para su uso en el manifiesto actual.
Guid	El símbolo representa un GUID y debe coincidir con el formato GUID.
id	El símbolo representa un identificador y debe ser un entero no negativo.
String	El símbolo representa un valor de cadena arbitrario.

Los símbolos distinguen entre mayúsculas y minúsculas y se hace referencia a ellos mediante la sintaxis \$(nombre-símbolo):

```
<Image Guid="$(ShellCommandGuid)" ID="$(cmdidSaveAll)" >
  <Source Uri="$(AssemblyName);Component/Resources/image.xaml" />
</Image>
```

Algunos símbolos están predefinidos para todos los manifiestos. Se pueden utilizar en el <atributo Uri <del elemento Source> o Import> para hacer referencia a rutas de acceso en el equipo local.

Símbolo	Descripción
CommonProgramFiles	El valor de la variable de entorno %CommonProgramFiles%
LocalAppData	El valor de la variable de entorno %LocalAppData%
ManifestFolder	La carpeta que contiene el archivo de manifiesto
MyDocuments	La ruta de acceso completa de la carpeta Mis documentos del usuario actual
ProgramFiles	El valor de la variable de entorno %ProgramFiles%
Sistema	La carpeta Windows-System32
WinDir	El valor de la variable de entorno %WinDir%

Imagen

El <elemento Image> define una imagen a la que puede hacer referencia un moniker. El GUID y el identificador tomados juntos forman el moniker de imagen. El moniker de la imagen debe ser único en toda la biblioteca de imágenes. Si más de una imagen tiene un moniker determinado, la primera encontrada al compilar la biblioteca es la que se conserva.

Debe contener al menos una fuente. Aunque las fuentes de tamaño neutro darán los mejores resultados en una amplia gama de tamaños, no son necesarias. Si se solicita al servicio una imagen de <un tamaño no definido en el elemento Image> y no hay ningún origen de tamaño neutro, el servicio elegirá el mejor origen específico del tamaño y lo escalará al tamaño solicitado.

```
<Image Guid="guid" ID="int" AllowColorInversion="true/false">
  <Source ... />
  <!-- optional additional Source elements -->
</Image>
```

Atributo	Definición
Guid	[Requerido] La parte GUID del moniker de imagen
id	[Requerido] La parte id del moniker de imagen
AllowColorInversion	[Opcional, predeterminado true] Indica si la imagen puede tener sus colores invertidos mediante programación cuando se usa en un fondo oscuro.

Origen

El <elemento Source> define un único recurso de origen de imagen (XAML y PNG).

```
<Source Uri="uri" Background="background">
  <!-- optional NativeResource element -->
</Source>
```

Atributo	Definición
Identificador URI	[Requerido] URI que define desde dónde se puede cargar la imagen. Puede tener uno de los valores siguientes: - Un URI de paquete con la autoridad application:/// - Una referencia de recursos de componentes absoluta - Una ruta de acceso a un archivo que contiene un recurso nativo
Información previa	[Opcional] Indica qué tipo de fondo está destinado a utilizar se va a utilizar el origen. Puede tener uno de los valores siguientes: - <i>Luz</i>. La fuente se puede utilizar sobre un fondo claro. - <i>Oscuro</i>. La fuente se puede utilizar sobre un fondo oscuro. - <i>HighContrast</i>. La fuente se puede utilizar en cualquier fondo en el modo de contraste alto. - <i>HighContrastLight</i>. La fuente se puede utilizar en un fondo claro en el modo de contraste alto. - <i>HighContrastDark</i>. La fuente se puede utilizar en un fondo oscuro en el modo de contraste alto. Si se omite el atributo Background, el origen se puede utilizar en cualquier fondo. Si Background es <i>Light</i>, <i>Dark</i>, <i>HighContrastLight</i> o <i>HighContrastDark</i>, los colores del origen nunca se invierten. Si Background se omite o se establece en <i>HighContrast</i>, la inversión de los colores del origen se controla mediante el atributo AllowColorInversion de la imagen.

Un <elemento Source> puede tener exactamente uno de los siguientes subelementos opcionales:

Elemento	Atributos (todos requeridos)	Definición
<Tamaño>	Value	La fuente se utilizará para imágenes del tamaño dado (en unidades de dispositivo). La imagen será cuadrada.
<> sizeRange	MinSize, MaxSize	El origen se utilizará para las imágenes de MinSize a MaxSize (en unidades de dispositivo) de forma inclusiva. La imagen será cuadrada.
<Dimensiones>	Ancho, alto	La fuente se utilizará para imágenes de la anchura y altura dadas (en unidades de dispositivo).

<> De DimensionRange	MinWidth, MinHeight, MaxWidth, MaxHeight	La fuente se utilizará para imágenes desde la anchura/altura mínima hasta la anchura/altura máxima (en unidades de dispositivo) inclusive.

Un <elemento source> también <puede tener un subelemento <nativoopcional>, que define un> de origen que se carga desde un ensamblado nativo en lugar de un ensamblado administrado.

<code><NativeResource Type="type" ID="int" /></code>		

Atributo	Definición
Tipo	[Requerido] El tipo del recurso nativo, XAML o PNG
id	[Requerido] La parte de ID entero del recurso nativo

Imagelist

El <Elemento de> ImageList define una colección de imágenes que se pueden devolver en una sola tira. La tira se construye bajo demanda, según sea necesario.

<pre><ImageList> <ContainedImage Guid="guid" ID="int" External="true/false" /> <!-- optional additional ContainedImage elements --> </ImageList></pre>		

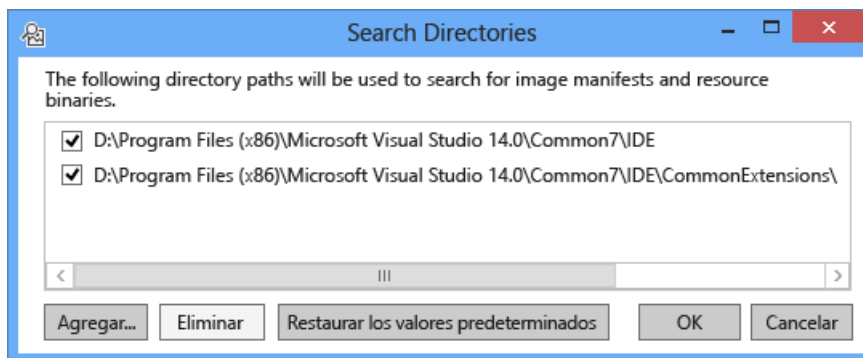
Atributo	Definición
Guid	[Requerido] La parte GUID del moniker de imagen
id	[Requerido] La parte id del moniker de imagen
Externo	[Opcional, predeterminado false] Indica si el moniker de imagen hace referencia a una imagen en el manifiesto actual.

El moniker de la imagen contenida no tiene que hacer referencia a una imagen definida en el manifiesto actual. Si la imagen contenida no se encuentra en la biblioteca de imágenes, se utilizará una imagen de marcador de posición en blanco en su lugar.

Cómo utilizar la herramienta

Validación de un manifiesto de imagen personalizado

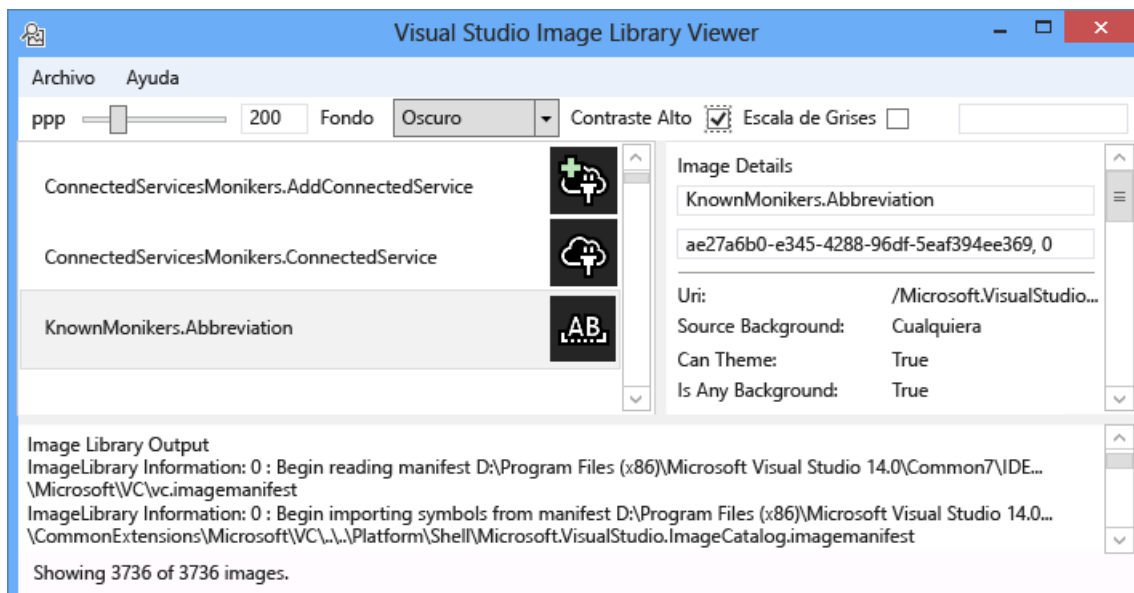
Para crear un manifiesto personalizado, se recomienda usar la herramienta ManifestFromResources para generar automáticamente el manifiesto. Para validar el manifiesto personalizado, inicie el Visor de biblioteca de imágenes y seleccione > de archivos Establecer rutas... para abrir el cuadro de diálogo Directorios de búsqueda. La herramienta usará los directorios de búsqueda para cargar manifiestos de imagen, pero también los usará para buscar los archivos .dll que contienen las imágenes en un manifiesto, así que asegúrese de incluir los directorios de manifiesto y DLL en este cuadro de diálogo.



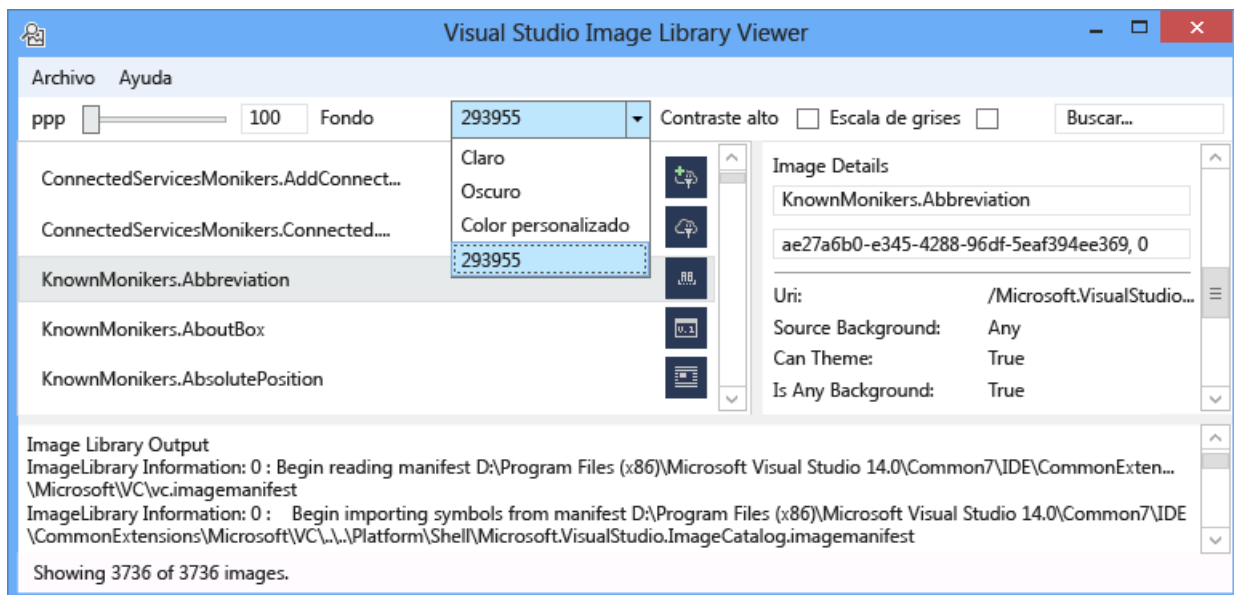
Haga clic en **Agregar...** para seleccionar nuevos directorios de búsqueda para buscar manifiestos y sus archivos DLL correspondientes. La herramienta recordará estos directorios de búsqueda, y se pueden activar o desactivar marcando o desmarcando un directorio.

De forma predeterminada, la herramienta intentará encontrar el directorio de instalación de Visual Studio y agregar esos directorios a la lista de directorios de búsqueda. Puede agregar manualmente directorios que la herramienta no encuentre.

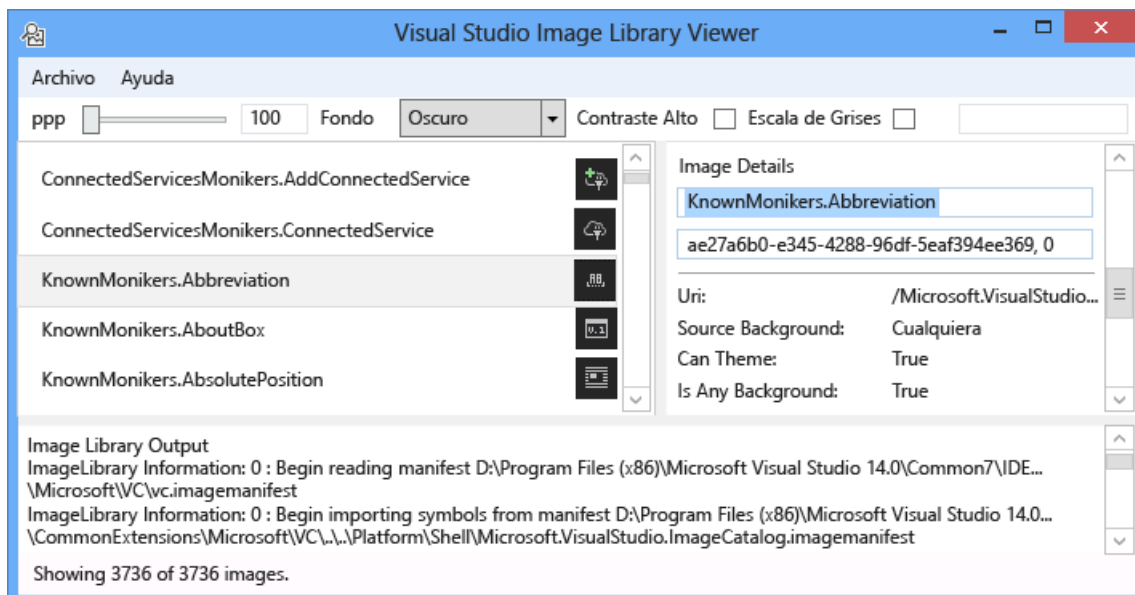
Una vez cargados todos los manifiestos, la herramienta se puede utilizar para alternar los colores de **fondo**, **DPI**, **alto contraste** escala de **grises** para las imágenes para que un usuario pueda inspeccionar visualmente los activos de imagen para comprobar que se representan correctamente para varios ajustes.



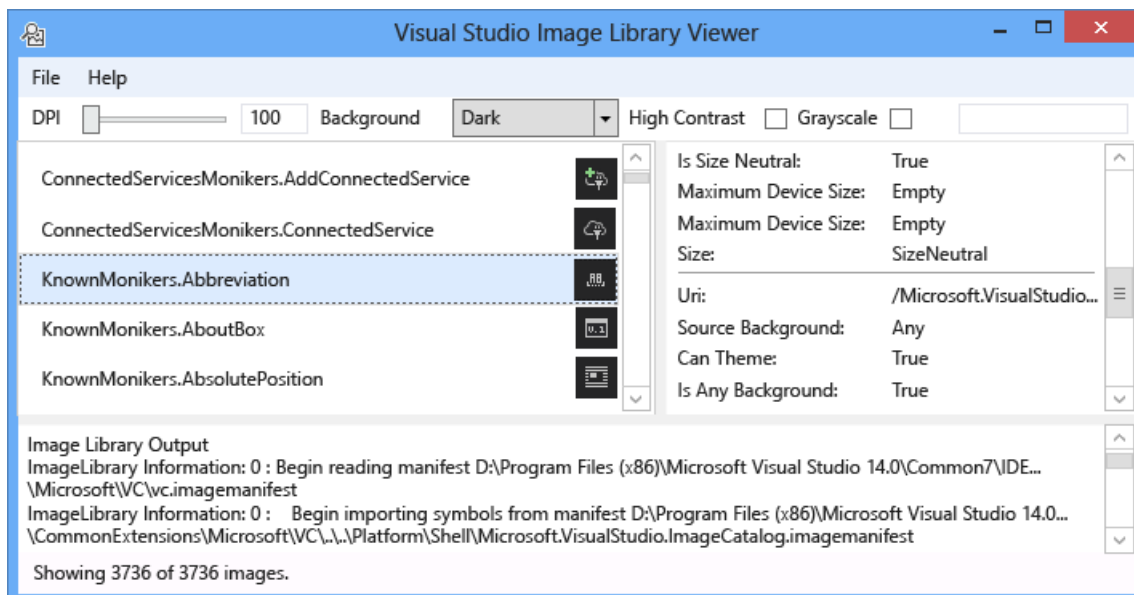
El color de fondo se puede establecer en Luz, Oscuro o un valor personalizado. Al seleccionar "Color personalizado" se abrirá un cuadro de diálogo de selección de color y se agregará ese color personalizado a la parte inferior del cuadro combinado de fondo para una fácil recuperación más adelante.



Al seleccionar un moniker de imagen, se muestra la información de cada imagen real detrás de ese moniker en el panel Detalles de imagen de la derecha. El panel también permite a los usuarios copiar un moniker por nombre o por valor GUID:ID sin formato.



La información que se muestra para cada origen de imagen incluye en qué tipo de fondo se muestra, si se puede crear temas o admite contraste alto, para qué tamaños es válido o si es de tamaño neutro y si la imagen procede de un ensamblado nativo.

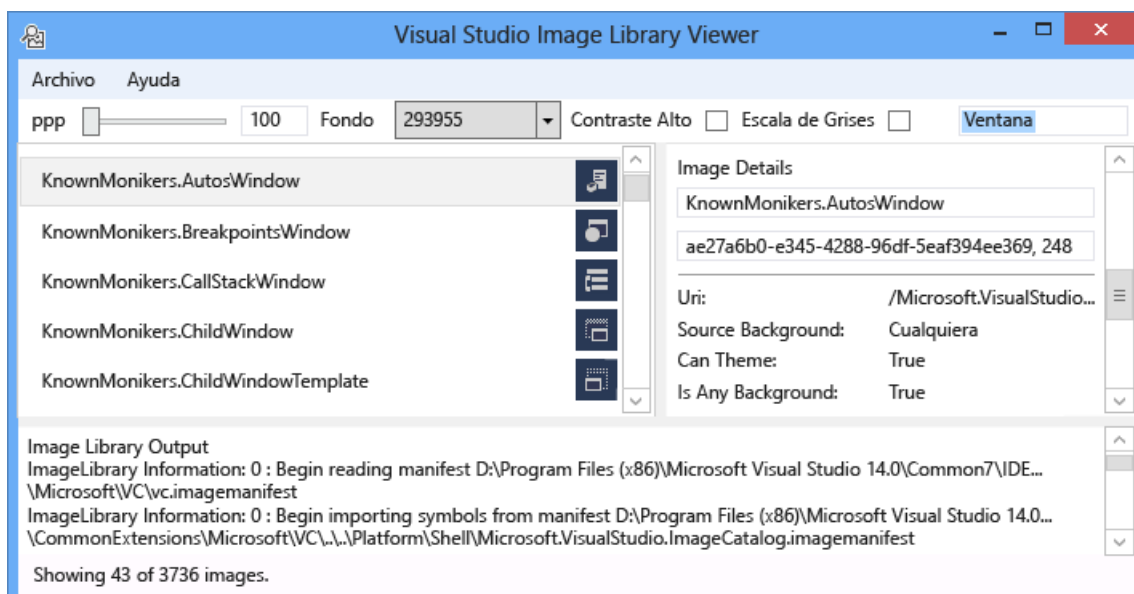


Al validar un manifiesto de imagen, se recomienda implementar el archivo DLL de manifiesto e imagen en sus ubicaciones del mundo real. Esto comprobará que las rutas de acceso relativas funcionan correctamente y que la biblioteca de imágenes puede encontrar y cargar el archivo DLL de manifiesto e imagen.

Búsqueda del catálogo de imágenes KnownMonikers

Para que coincida mejor con el estilo de Visual Studio, una extensión de Visual Studio puede usar imágenes en el catálogo de imágenes de Visual Studio en lugar de crear y usar su propio. Esto tiene la ventaja de no tener que mantener esas imágenes y garantiza que la imagen tendrá una imagen de respaldo de PPP alto, por lo que debe verse correcta en todos los valores de PPP que admite Visual Studio.

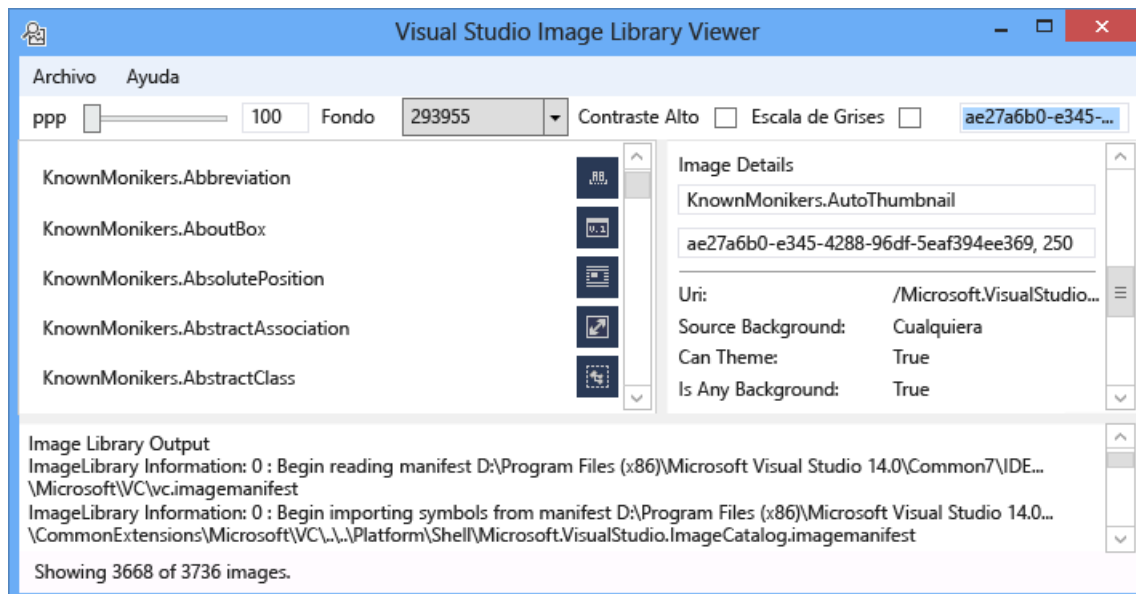
El visor de la biblioteca de imágenes permite buscar un manifiesto para que un usuario pueda encontrar el moniker que representa un recurso de imagen y usar ese moniker en el código. Para buscar imágenes, introduzca el término de búsqueda deseado en el cuadro de búsqueda y pulse Intro. La barra de estado en la parte inferior mostrará cuántas coincidencias se encontraron del total de imágenes en todos los manifiestos.



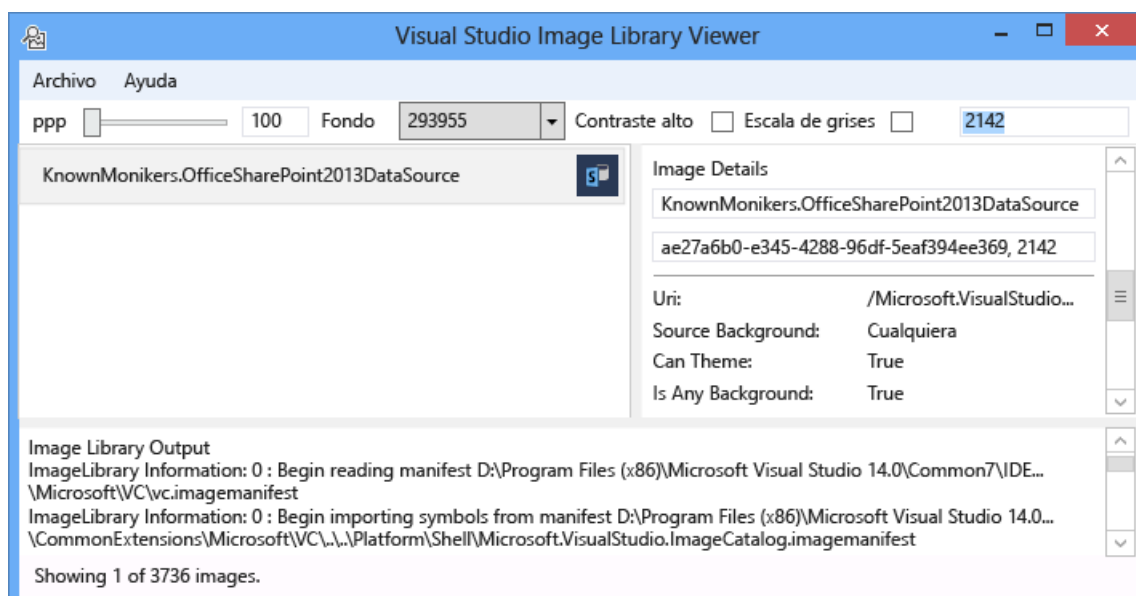
Al buscar monikers de imagen en manifiestos existentes, se recomienda buscar y usar solo los monikers del catálogo de imágenes de Visual Studio, otros monikers de acceso público intencionalmente o sus propios monikers personalizados. Si usa monikers no públicos, la interfaz de usuario personalizada podría romperse o cambiar sus imágenes de maneras inesperadas si o cuando esos monikers e imágenes no públicos se cambian o actualizan.

Además, es posible buscar por GUID. Este tipo de búsqueda es útil para filtrar la lista a un único manifiesto o una

sola subsección de un manifiesto si ese manifiesto contiene varios GUID.



Por último, la búsqueda por ID también es posible.



Notas

- De forma predeterminada, la herramienta extraerá varios manifiestos de imagen presentes en el directorio de instalación de Visual Studio. El único que tiene monikers públicamente consumibles es el **Microsoft.VisualStudio.ImageCatalog** manifiesto. GUID: ae27a6b0-e345-4288-96df-5eaf394ee369 (no invalide este GUID en un manifiesto personalizado) Tipo: KnownMonikers
- La herramienta intenta al iniciar se cargan todos los manifiestos de imagen que encuentra, por lo que la aplicación puede tardar varios segundos en aparecer realmente. También puede ser lento o no responde al cargar los manifiestos.

Salida de ejemplo

Esta herramienta no genera ninguna salida.

Instalación de VSPackages con Windows Installer

06/04/2020 • 3 minutes to read • [Edit Online](#)

La integración de Visual Studio VSPackage en requiere algo más que copiar archivos en el equipo de un usuario. El instalador de VSPackage debe instalar el VSPackage y sus Visual Studioarchivos dependientes, y registrarlos e integrarlos en . El VSPackage puede aprovechar las características de integración, como mostrar un icono en la Visual Studio pantalla de presentación y Acerca del cuadro de diálogo.

Los archivos de Microsoft Windows Installer son la forma recomendada de distribuir los VSPackages. Los paquetes de Windows Installer fáciles de usar Visual Studiose pueden ejecutar en cualquier sistema operativo Windows compatible con . Para obtener más información, consulte [Windows Installer](#).

En esta sección

- [Datos básicos de Windows Installer](#)

Proporciona información general sobre Windows Installer.

- [Escenarios de instalación de VSPackage](#)

Describe diferentes maneras en las que puede admitir instalaciones Visual Studioen paralelo de los VSPackages y .

- [Creación de un paquete de Windows Installer](#)

Proporciona información general de los pasos típicos que Visual Studiosiguen los instaladores para instalar e integrar correctamente VSPackages en .

- [Detección de requisitos del sistema](#)

Describe cómo un instalador Visual Studio puede detectar y sus componentes y cancelar la instalación si no se cumplen los requisitos de VSPackage.

- [Administración de componentes](#)

Describe cómo desarrollar un instalador que mantenga la integridad de las versiones anteriores del producto.

- [Selección del directorio de instalación de un VSPackage](#)

Resume las opciones para localizar VSPackages.

- [Registro de VSPackage](#)

Describe cómo se registran los VSPackages en el momento de la instalación y por qué el autorregistro es una mala idea.

- [Implementación de tipos de proyecto](#)

Describe cómo usar el nuevo agregador de tipo de proyecto para los tipos de proyecto de código administrado.

- [Generación de información del Registro para un instalador](#)

Explica cómo usar RegPkg.exe para generar un manifiesto de registro para un VSPackage administrado.

- [Comandos que se deben ejecutar después de la instalación](#)

Explica cómo ejecutar comandos posteriores a Visual Studiola instalación para integrar VSPackages en .

- [Desinstalación de un VSPackage con Windows Installer](#)

Describe los pasos que debe realizar el instalador cuando los usuarios desinstalan el VSPackage.

Datos básicos de Windows Installer

06/04/2020 • 7 minutes to read • [Edit Online](#)

Windows Installer instala y desinstala aplicaciones o productos de software en el equipo de un usuario, realizando estas tareas en unidades denominadas componentes de Windows Installer (a veces denominados WIC o simplemente componentes). Un GUID identifica cada WIC, que es la unidad básica de instalación y recuento de referencias para las configuraciones mediante Windows Installer.

Para obtener documentación completa de Windows Installer, consulte el tema Platform SDK, [Windows Installer](#).

Creación de un VSPackage

Windows Installer usa paquetes de instalación, que contienen información que Windows Installer necesita para instalar, desinstalar o reparar un producto y ejecutar la interfaz de usuario (UI) de instalación. Cada paquete de instalación incluye un archivo .msi, que contiene una base de datos de instalación, una secuencia de información de resumen y secuencias de datos para varias partes de la instalación. Para utilizar el instalador, debe crear una instalación. Dado que el instalador organiza las instalaciones en torno al concepto de componentes y almacena información sobre la instalación en una base de datos relacional, el proceso de creación de un paquete de instalación implica en general los siguientes pasos:

1. Planifique la creación de la configuración para admitir sus estrategias de control de versiones y en paralelo.
2. Identifique las características que se presentarán a los usuarios.
3. Organice el VSPackage y las dependencias en componentes.
4. Rellene la base de datos de instalación con información.
5. Valide el paquete de instalación.

Esta documentación se refiere principalmente al primer y tercer paso del proceso. Durante estos pasos se organizan las características de VSPackage en WICs para que Visual Studio pueda enmarcar la estrategia de control de versiones y mantenimiento para tener en cuenta las versiones posteriores de . Los tres pasos restantes se tratan en detalle en la documentación de Windows Installer en Platform SDK.

Términos clave

A continuación se muestran definiciones de términos clave relacionados con la tecnología de Windows Installer.

Archivos de recursos, claves del Registro, accesos directos, etc. que se pueden instalar en un equipo. Estos recursos se agrupan lógicamente en componentes de Windows Installer.

Componente de Windows Installer (WIC) La unidad básica de instalación que representa una agrupación lógica de recursos relacionados que se instalan y desinstalan como una unidad. Los componentes de Windows Installer se identifican mediante un identificador de componente único o GUID. Además, Windows Installer mantiene su recuento de referencias en el nivel WIC. Para obtener la máxima flexibilidad de control de versiones, no incluya más de un recurso principal, como un archivo DLL, en un WIC determinado. Tenga en cuenta que después de identificar y rellenar un WIC, darle un GUID e implementarlo, no puede cambiar su composición. Para obtener más información, consulte [Organización de aplicaciones en componentes](#).

Paquete (paquete Redist) Una unidad de implementación que consta de un archivo .msi y archivos de origen externos a los que puede apuntar este archivo. Un paquete contiene toda la información que Windows Installer necesita para ejecutar la interfaz de usuario e instalar o desinstalar la aplicación.

Archivo .msi Un archivo de almacenamiento estructurado COM que contiene las instrucciones y los datos necesarios para instalar una aplicación. Cada paquete contiene al menos un archivo .msi. El archivo .msi contiene la base de datos del instalador, una secuencia de información de resumen y, posiblemente, una o más transformaciones y archivos de origen internos. Los archivos que se van a instalar se pueden comprimir en un archivador y almacenarse en una secuencia en el archivo .msi o almacenarse, comprimirse o descomprimirse fuera del archivo .msi del medio de origen. Para obtener más información, consulte [Extensiones de archivo de Windows Installer](#).

Aplicación de reglas de Windows Installer

Dos conjuntos de reglas determinan la implementación de recursos a través de los componentes de la instalación. El propio Windows Installer mantiene un conjunto de reglas, mientras que debe aplicar el segundo conjunto como autor de la instalación.

NOTE

La aplicación de las reglas de Windows Installer solo se produce si ejecuta una validación del archivo .msi. Sin embargo, se le advierte que trate estas reglas como mejores prácticas. Para obtener más información, consulte [Validación de una base de datos de instalación y validación de paquetes](#).

Reglas aplicadas por el instalador

- Todos los archivos de un componente determinado deben instalarse en el mismo directorio. Por el contrario, los archivos instalados en carpetas separadas deben pertenecer a componentes independientes.
- Solo puede haber una ruta de acceso clave por componente. La ruta de acceso de clave es simplemente un archivo o una clave del Registro que representa todo el componente.

Responsabilidades del proveedor de componentes

- Los dos recursos que puedan enviarse por separado en versiones posteriores deben existir en componentes independientes. Los recursos deben agruparse en el mismo componente solo cuando esté seguro de que estos recursos nunca se enviarán por separado. De hecho, se recomienda que todos los recursos primarios (DLL, por ejemplo) siempre existan en WIC independientes. Para obtener más información, consulte [Definición de componentes del instalador](#).
- Ningún recurso versionado debe enviarse nunca en más de un WIC.

Vea también

- [¿Qué sucede si se rompen las reglas de componentes?](#)

Escenarios de instalación de VSPackage

06/04/2020 • 11 minutes to read • [Edit Online](#)

Es importante diseñar el instalador de VSPackage para la flexibilidad. Por ejemplo, es posible que deba lanzar una revisión de seguridad en el futuro o puede cambiar una estrategia empresarial que requiera una compatibilidad completa con el control de versiones en paralelo.

En [Compatibilidad con varias versiones de Visual Studio](#), puede leer acerca de las Visual Studio ventajas y problemas de admitir instalaciones en paralelo de instalaciones compartidas o en paralelo de su VSPackage. En resumen, en paralelo VSPackages le dan la mayor flexibilidad Visual Studio para admitir nuevas características de .

Los escenarios descritos en este tema no son sus únicas opciones, pero se presentan como prácticas recomendadas sugeridas.

Componentes, privacidad y uso compartido

Haga que sus componentes sean independientes

Una vez que identifique y `GUID` rellene un componente, asigne un archivo y despliegó el componente, no podrá cambiar su composición. Si cambia la composición de un componente, el componente resultante `GUID` debe ser un nuevo componente con un nuevo . Teniendo en cuenta estos hechos, la mayor flexibilidad de control de versiones se ofrece al hacer que cada componente sea independiente y autosuficiente. Para obtener más información acerca de las reglas que rigen los componentes, consulte [Cambiar el código de componente](#) y qué sucede si las reglas de [componentes están rotas?](#).

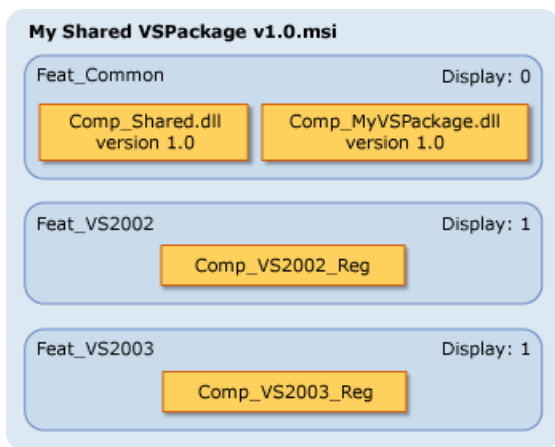
No mezcle recursos compartidos y privados en un componente

El recuento de referencias se produce en el nivel de componente. Por lo tanto, la mezcla de recursos compartidos y privados en un componente hace que sea imposible actualizar recursos privados, como un archivo ejecutable, sin sobrescribir también los recursos compartidos. Este escenario crea problemas de compatibilidad con versiones anteriores y le impide crear capacidad en paralelo.

Por ejemplo, los valores del Registro Visual Studio SDK que se usan para registrar el VSPackage con el debe mantenerse en un componente independiente de uno utilizado para registrar el VSPackage con Visual Studio. Los archivos compartidos o los valores del Registro van en otro componente.

Escenario 1: VSPackage compartido

En este escenario, un VSPackage compartido (un único binario que admite varias versiones de Visual Studio se incluye en un paquete de Windows Installer. Registrarse con cada versión de Visual Studio se controla mediante características seleccionables por el usuario. También significa que cuando se asigna a características independientes, cada componente se puede seleccionar individualmente para la instalación Visual Studio o desinstalación, poniendo al usuario en control de la integración del VSPackage en diferentes versiones de . (Consulte [Características de Windows Installer](#) para obtener más información sobre el uso de características en paquetes de Windows Installer.)



Como se muestra en la ilustración, los componentes compartidos forman parte de la función Feat_Common, que siempre se instala. Al hacer visibles las características de Feat_VS2002 y Feat_VS2003, los usuarios pueden elegir en tiempo de instalación en qué versiones de Visual Studio desean que se integre el VSPackage. Los usuarios también pueden usar el modo de mantenimiento de Windows Installer para agregar o quitar características, que en este caso agrega o quita la información de registro de VSPackage de diferentes versiones de Visual Studio.

NOTE

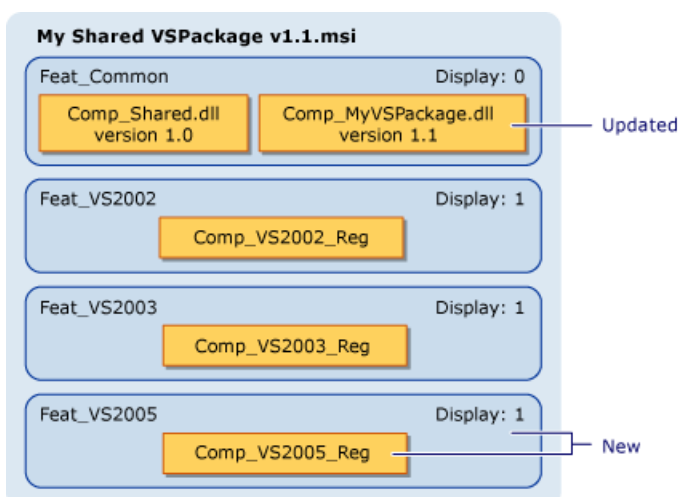
Al establecer la columna Visualización de una entidad en 0, se oculta. Un valor de columna de nivel bajo, como 1, garantiza que siempre se instalará. Para obtener más información, vea [InstallLEVEL Property](#) and [Feature Table](#).

Escenario 2: Actualización de VSPackage compartida

En este escenario, se incluye una versión actualizada del instalador de VSPackage en el escenario 1. En aras de la discusión, la actualización agrega compatibilidad con Visual Studio, pero también podría ser una revisión de seguridad más sencilla o un Service Pack de corrección de errores. Las reglas de Windows Installer para instalar componentes más recientes requieren que los componentes sin cambios que ya están en el sistema no se vuelvan a copiar. En este caso, un sistema con la versión 1.0 ya presente sobrescribirá el componente actualizado Comp_MyVSPackage.dll y permitirá a los usuarios elegir agregar la nueva característica Feat_VS2005 con su componente Comp_VS2005_Reg.

Caution

Siempre que un VSPackage se Visual Studiocomparte entre varias versiones de , es esencial que las versiones posteriores de VSPackage mantengan la compatibilidad con versiones anteriores de Visual Studio. Donde no puede mantener la compatibilidad con versiones anteriores, debe usar VSPackages privados en paralelo. Para obtener más información, vea [Compatibilidad con varias versiones de Visual Studio](#).

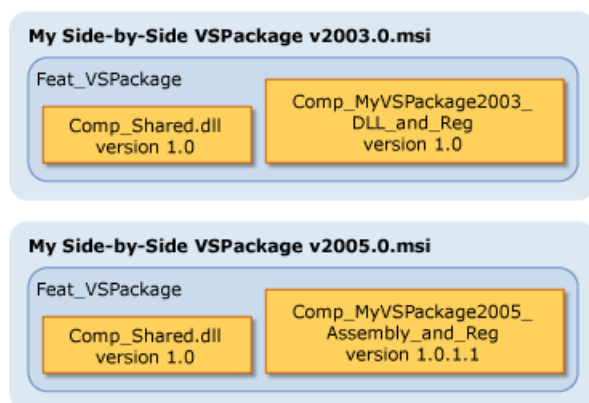


Este escenario presenta un nuevo instalador de VSPackage, aprovechando la compatibilidad de Windows Installer

para actualizaciones menores. Los usuarios simplemente instalan la versión 1.1 y actualiza la versión 1.0. Sin embargo, no es necesario tener la versión 1.0 en el sistema. El mismo instalador instalará la versión 1.1 en un sistema sin la versión 1.0. La ventaja de proporcionar actualizaciones menores de esta manera es que no es necesario pasar por el trabajo de desarrollo de un instalador de actualización y un instalador de producto completo. Un instalador realiza ambos trabajos. Una corrección de seguridad o Service Pack podrían aprovechar las revisiones de Windows Installer. Para obtener más información, consulte [Parches y actualizaciones](#).

Escenario 3: VSPackage en paralelo

Este escenario presenta dos instaladores de VSPackage: uno para cada versión de Visual Studio .NET 2003 y Visual Studio. Cada instalador instala un VSPackage en paralelo o privado (uno que se compila e instala específicamente para una versión determinada de Visual Studio). Cada VSPackage está en su propio componente. Por lo tanto, cada uno puede ser reparado individualmente con parches o versiones de mantenimiento. Dado que el archivo DLL de VSPackage ahora es específico de la versión, es seguro incluir su información de registro en el mismo componente que el archivo DLL.

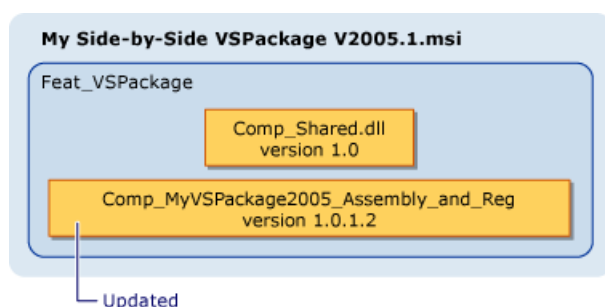


Cada instalador también incluye código que se comparte entre los dos instaladores. Si el código compartido está instalado en una ubicación común, la instalación de ambos archivos .msi instalará el código compartido solo una vez. El segundo instalador solo incrementa un recuento de referencias en el componente. El recuento de referencias garantiza que si se desinstala uno de los VSPackages, el código compartido permanecerá para el otro VSPackage. Si también se desinstala el segundo VSPackage, se quitará el código compartido.

Escenario 4: Actualización de VSPackage en paralelo

En este escenario, el VSPackage para Visual Studio sufrió de una vulnerabilidad de seguridad y debe emitir una actualización. Al igual que en el escenario 2, puede crear un nuevo archivo .msi que actualice una instalación existente para incluir la corrección de seguridad, así como implementar nuevas instalaciones con la corrección de seguridad ya en su lugar.

En este caso, el VSPackage es un VSPackage administrado instalado en la caché global de ensamblados (GAC). Al volver a generarlo para incluir la corrección de seguridad, debe cambiar la parte del número de revisión del número de versión del ensamblado. La información de registro para el nuevo número de Visual Studio versión de ensamblado sobrescribe la versión anterior, lo que hace que se cargue el ensamblado fijo.



Para obtener más información sobre la implementación de ensamblados en paralelo, vea [Simplificar la](#)

[implementación y resolver DLL Hell con .NET Framework.](#)

Vea también

- [Windows Installer](#)
- [Compatibilidad con varias versiones de Visual Studio](#)

Crear un paquete de Windows Installer

06/04/2020 • 3 minutes to read • [Edit Online](#)

Los datos controlan el modelo de Windows Installer. En lugar de escribir un script de procedimiento para copiar archivos y escribir entradas del Registro, por ejemplo, se escriben filas y columnas en tablas de base de datos que contienen datos de archivo y registro.

Entradas de base de datos

Para instalar un VSPackage, un paquete de Windows Installer debe contener entradas de base de datos para realizar las siguientes tareas:

- Busque en el sistema Visual Studio para buscar las versiones de los soportes de VSPackage (mediante tablas de Windows Installer que incluyen AppSearch, CompLocator, RegLocator, DrLocator y Signature).
- Cancele la instalación si Visual Studio no se instala ninguna versión compatible de o si no se cumple otro requisito del sistema del VSPackage (mediante la tabla LaunchCondition).
- Instale el VSPackage y los archivos dependientes (mediante el directorio, el componente y las tablas de archivos).
- Agregue la información adecuada para el VSPackage al registro (mediante la tabla del Registro).
- Integre el Visual Studio VSPackage mediante una llamada a **devenv.exe /setup** (mediante la tabla CustomAction).

Para obtener más información, consulte [Windows Installer](#).

Herramientas de configuración

Una variedad de herramientas de instalación de terceros proporcionan un entorno de desarrollo para paquetes de Windows Installer. Están disponibles las siguientes herramientas gratuitas:

- InstallShield edición limitada

Puede obtener una versión limitada de InstallShield a través del cuadro de diálogo **Nuevo proyecto** de Visual Studio. Expanda **Otros tipos de proyecto** y, a continuación, seleccione Configuración e implementación. Seleccione la plantilla InstallShield.

- Conjunto de herramientas XML de Windows Installer

El conjunto de herramientas XML (WiX) de Windows Installer compila paquetes de Windows Installer a partir de archivos de origen XML. El conjunto de herramientas WiX es un proyecto de código abierto de Microsoft. Puede descargar el código fuente y los ejecutables desde el conjunto de [herramientas WiX](#).

Para los productos Visual Studio comerciales que Visual Studio SDKse integran mediante el archivo , vea [Visual Studio Marketplace](#).

Vea también

- [Instalar VSPackages con Windows Installer](#)

Detectar los requisitos del sistema

06/04/2020 • 3 minutes to read • [Edit Online](#)

Un VSPackage no puede funcionar a menos que Visual Studio está instalado. Cuando se usa Microsoft Windows Installer para administrar la instalación de SU VSPackage, puede configurar el instalador para detectar si Visual Studio está instalado. También puede configurarlo para comprobar si el sistema tiene otros requisitos, por ejemplo, una versión determinada de Windows o una cantidad determinada de RAM.

Detectar ediciones de Visual Studio

Para determinar si está instalada una edición de Visual Studio, compruebe que el valor de la clave del Registro **Instalar** es (*REG_DWORD*) *1* en la carpeta adecuada, como se muestra en la tabla siguiente. Tenga en cuenta que hay una jerarquía de ediciones de Visual Studio:

1. Enterprise
2. Profesional
3. Comunidad

Cuando se instala una edición más reciente, se agregan las claves del registro para esa edición, así como para ediciones anteriores. Es decir, si se instala la edición Enterprise, la clave **De instalación** se establece en *1* para Enterprise, así como para las ediciones Professional y Community. Por lo tanto, solo debe comprobar la edición más reciente que necesita.

NOTE

En la versión de 64 bits del editor del Registro, las claves de 32 bits se muestran en HKEY_LOCAL_MACHINE.\ Las claves de Visual Studio se encuentran bajo HKEY_LOCAL_MACHINE, SOFTWARE,\Wow6432Node, Microsoft, DevDiv, vs, Servicio.

PRODUCTO	CLAVE
Visual Studio Enterprise 2015	HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, DevDiv, vs, servicio, 14,0, empresa
Visual Studio Professional 2015	HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, DevDiv, vs, Servicio, 14,0, profesional
Visual Studio Community 2015	HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, DevDiv, vs, servicio, 14,0, comunidad
Shell de Visual Studio 2015 (integrado y aislado)	HKEY_LOCAL_MACHINE,SOFTWARE, Microsoft, DevDiv, vs, servicio, 14,0, isoshell

Detectar cuando Visual Studio se está ejecutando

El VSPackage no se puede registrar correctamente si Visual Studio se está ejecutando cuando se instala el VSPackage. El instalador debe detectar cuándo se está ejecutando Visual Studio y, a continuación, se niega a instalar el programa. Windows Installer no permite usar entradas de tabla para habilitar dicha detección. En su lugar, debe crear una acción `EnumProcesses` personalizada, como se indica a continuación: use la función para

detectar el proceso *devenv.exe* y, a continuación, establezca una propiedad de instalador que se use en una condición de inicio o muestre condicionalmente un cuadro de diálogo que solicite al usuario que cierre Visual Studio.

Vea también

- [Instalar VSPackages con Windows Installer](#)

Gestión de componentes

06/04/2020 • 3 minutes to read • [Edit Online](#)

Las unidades de tareas en Windows Installer se conocen como componentes de Windows Installer (a veces denominados WIC o simplemente componentes). Un GUID identifica cada WIC, que es la unidad básica de instalación y recuento de referencias para las configuraciones que usan Windows Installer.

Aunque puede usar varios productos para crear el instalador de VSPackage, esta discusión supone el uso de archivos de Windows Installer (.msi). Al crear el instalador, debe administrar correctamente la implementación de archivos para que se haya quedado en todo momento el recuento de referencias correcto. Por lo tanto, diferentes versiones de su producto no interferirán ni se romperán entre sí en una combinación de escenarios de instalación y desinstalación.

En Windows Installer, el recuento de referencias se produce en el nivel de componente. Debe organizar cuidadosamente los recursos (archivos, entradas del registro, etc.) en componentes. Hay otros niveles de organización, como módulos, características y productos, que pueden ayudar en diferentes escenarios. Para obtener más información, consulte [Conceptos básicos](#) de Windows Installer .

Directrices de la configuración de creación para la instalación en paralelo

- Cree archivos y claves de registro que se comparten entre versiones en sus propios componentes.

Esto le permite consumirlos fácilmente en la próxima versión. Por ejemplo, escriba bibliotecas que estén registradas globalmente, extensiones de archivo, otros elementos registrados en HKEY_CLASSES_ROOT , etc.

- Agrupe los componentes compartidos en módulos de combinación independientes.

Esta estrategia le ayuda a crear correctamente para la instalación en paralelo en el futuro.

- Instale los archivos compartidos y las claves del Registro mediante los mismos componentes de Windows Installer en todas las versiones.

Si usa un componente diferente, los archivos y las entradas del registro se desinstalan cuando se desinstala un VSPackage versionado, pero otro VSPackage sigue instalado.

- No mezcle elementos versionados y compartidos en el mismo componente.

Al hacerlo, es imposible instalar elementos compartidos en una ubicación global y elementos versionados en ubicaciones aisladas.

- No tiene claves de registro compartidas que apunten a archivos versionados.

Si lo hace, las claves compartidas se sobrescribirán cuando se instale otro VSPackage versionado. Después de quitar la segunda versión, el archivo al que apunta la clave ha desaparecido.

Vea también

- [Elija entre VSPackages compartidos y versionados](#)
- [Escenarios de configuración de VSPackage](#)

Elija el directorio de instalación de un VSPackage

06/04/2020 • 4 minutes to read • [Edit Online](#)

Un VSPackage y sus archivos auxiliares deben estar en el sistema de archivos de un usuario. La ubicación depende de si el VSPackage está administrado o no, el esquema de control de versiones en paralelo y la elección del usuario.

VSPackages no administrados

Un VSPackage no administrado es un servidor COM que se puede instalar en cualquier ubicación. Su información de registro debe reflejar con precisión su ubicación. La interfaz de usuario (UI) del instalador debe

`ProgramFilesFolder` proporcionar una ubicación predeterminada como subdirectorio del valor de la propiedad de Windows Installer. Por ejemplo:

```
<ProgramFilesFolder>|<>|<MyCompany MyVSPackageProduct>?V1.0\
```

Se debe permitir al usuario cambiar el directorio predeterminado para dar cabida a los usuarios que mantienen una pequeña partición de arranque y prefieren instalar aplicaciones y herramientas en otro volumen.

Si el esquema en paralelo usa un VSPackage versionado, puede usar subdirectorios para almacenar diferentes versiones. Por ejemplo:

```
<ProgramFilesFolder>|<MyCompany>|<MyVSPackageProduct>|V1.0\2002\
```

```
<ProgramFilesFolder>|<MyCompany>|<MyVSPackageProduct>|V1.0\2003\
```

```
<ProgramFilesFolder>|<MyCompany>|<MyVSPackageProduct>|V1.0\2005\
```

VSPackages administrado

VSPackages administrados también se pueden instalar en cualquier ubicación. Sin embargo, debe considerar instalarlos siempre en la caché global de ensamblados (GAC) para reducir los tiempos de carga del ensamblado. Dado que los VSPackages administrados siempre son ensamblados con nombre seguro, instalarlos en la GAC significa que su comprobación de firma de nombre seguro toma solo en el momento de la instalación. Los ensamblados con nombre seguro instalados en otra parte del sistema de archivos deben tener sus firmas verificadas cada vez que se cargan. Al instalar VSPackages administrados en la GAC, use el modificador **/assembly** de la herramienta regpkg para escribir entradas del Registro que apunten al nombre seguro del ensamblado.

Si instala VSPackages administrados en una ubicación distinta de la GAC, siga el consejo anterior dado para VSPackages no administrados para elegir jerarquías de directorios. Use el modificador **/codebase** de la herramienta regpkg para escribir entradas del Registro que apunten a la ruta de acceso del ensamblado VSPackage.

Para obtener más información, vea [Registrar y anular el registro de VSPackages](#).

archivos DLL satélite

Por convención, los archivos DLL satélite de VSPackage, que contienen recursos para una configuración regional determinada, se encuentran en subdirectorios del directorio *VSPackage*. Los subdirectorios corresponden a valores de ID de configuración regional (LCID).

El [administrar VSPackages](#) artículo indica que Visual Studio las entradas del registro controlan donde realmente busca el archivo DLL satélite de un VSPackage. Sin Visual Studio embargo, intenta cargar un archivo DLL satélite en un subdirectorio denominado para un valor LCID, en el siguiente orden:

1. LCID predeterminado (Visual Studio LCID; por ejemplo, *1033* euros para inglés)
2. LCID predeterminado con el subidioma predeterminado.
3. LCID predeterminado del sistema.
4. LCID predeterminado del sistema con el subidioma predeterminado.
5. Inglés de los EE. UU. (.*-1033* o .*-0x409*).

Si el archivo DLL de VSPackage incluye recursos y la Visual Studio entrada del Registro **SatelliteDll-DIName** apunta a él, intenta cargarlos en el orden anterior.

Vea también

- [Elija entre VSPackages compartidos y versionados](#)
- [Administración de VSPackages](#)
- [Administrar el registro de paquetes](#)

Registro de VSPackage

06/04/2020 • 7 minutes to read • [Edit Online](#)

VSPackages debe Visual Studio advertir que están instalados y deben cargarse. Este proceso se realiza escribiendo información en el registro. Ese es un trabajo típico de un instalador.

NOTE

Es una práctica aceptada durante el desarrollo de VSPackage para usar el autorregistro. Sin Programa Visual Studio Industry Partner embargo, los socios no pueden enviar sus productos utilizando el autorregistro como parte de la configuración.

Las entradas del Registro en un paquete de Windows Installer generalmente se realizan en la tabla Registro. También puede registrar extensiones de archivo en la tabla Registro. Sin embargo, Windows Installer proporciona compatibilidad integrada a través del identificador de programación (ProgId), las tablas de clase, extensión y verbo. Para obtener más información, vea [Tablas de base de datos](#).

Asegúrese de que las entradas del registro están asociadas con el componente adecuado para la estrategia en paralelo elegida. Por ejemplo, las entradas del Registro para un archivo compartido deben estar asociadas con el componente de Windows Installer de ese archivo. Del mismo modo, las entradas del registro para un archivo específico de la versión deben estar asociadas con el componente de ese archivo. De lo contrario, instalar o desinstalar el Visual Studio VSPackage para una versión de podría interrumpir el VSPackage en otras versiones. Para obtener más información, vea [Compatibilidad con varias versiones de Visual Studio](#).

NOTE

La forma más fácil de administrar el registro es utilizar los mismos datos en los mismos archivos tanto para el registro de desarrollador como para el registro en tiempo de instalación. Por ejemplo, algunas herramientas de desarrollo del instalador pueden consumir archivos en formato .reg en tiempo de compilación. Si los desarrolladores mantienen archivos .reg para su propio desarrollo y depuración diaria, esos mismos archivos se pueden incluir automáticamente en el instalador. Si no puede compartir automáticamente los datos de registro, debe asegurarse de que la copia de los datos de registro del instalador esté actualizada.

Registro de VSPackages no administrados

VSPackages no administrados (incluidos los generados por la plantilla de paquete de Visual Studio) usan archivos .rgs de estilo ATL para almacenar información de registro. El formato de archivo .rgs es específico de ATL y generalmente no se puede consumir tal cual mediante una herramienta de creación de instalación. La información de registro para el instalador de VSPackage debe mantenerse por separado. Por ejemplo, los desarrolladores pueden mantener los archivos en formato .reg sincronizados con los cambios de archivo .rgs. Los archivos .reg se pueden combinar con RegEdit para el trabajo de desarrollo o consumirlos por un instalador.

Registro de VSPackages administrados

La herramienta RegPkg lee los atributos de registro de un VSPackage administrado y puede escribir la información directamente en el registro o escribir archivos de formato .reg que puede consumir un instalador.

NOTE

La herramienta RegPkg no es redistribuible y no se puede usar para registrar un VSPackage en el sistema de un usuario.

Por qué VSPackages no debe registrarse automáticamente en el momento de la instalación

Los instaladores de VSPackage no deben confiar en el autorregistro. A primera vista, mantener los valores del registro de un VSPackage sólo en el propio VSPackage parece una buena idea. Dado que los desarrolladores necesitan los valores del registro disponibles para su trabajo de rutina y pruebas, tiene sentido evitar mantener una copia independiente de los datos del registro en el instalador. El instalador puede confiar en el propio VSPackage para escribir valores del Registro.

Si bien es bueno en teoría, el autorregistro tiene varios defectos que lo hacen inadecuado para la instalación de VSPackage:

- La compatibilidad correcta con la instalación, la desinstalación, la reversión de la instalación y la reversión de la desinstalación requiere que cree cuatro acciones personalizadas para cada VSPackage administrado que se autoregistre mediante una llamada a RegPkg.
- El enfoque de la compatibilidad en paralelo puede requerir que cree cuatro acciones personalizadas que invoquen RegSvr32 o RegPkg para cada versión compatible de Visual Studio.
- Una instalación con módulos autoregistrados no se puede revertir de forma segura porque no hay forma de saber si las claves autoregistradas son utilizadas por otra característica o aplicación.
- Los archivos DLL autoregistrados a veces se vinculan a los archivos DLL auxiliares que no están presentes o son la versión incorrecta. En cambio, Windows Installer puede registrar archivos DLL mediante las tablas del Registro sin dependencia del estado actual del sistema.
- Se puede denegar el acceso al código de autorregistro a los recursos de red, como las bibliotecas de tipos, si un componente se especifica como run-from-source y aparece en la tabla SelfReg. Esto puede provocar un error en la instalación del componente durante una instalación administrativa.

Vea también

- [Windows Installer](#)
- [Registro de paquetes administrados](#)

Implementar tipos de proyecto

06/04/2020 • 2 minutes to read • [Edit Online](#)

Visual Studio SDK instala un nuevo agregador de tipo de proyecto (*ProjectAggregator2.dll*) y también un paquete de Windows Installer para su redistribución (*ProjectAggregator2.msi*). Debe usar el nuevo agregador para los tipos de proyecto de código administrado. ProjectAggregator2 se endosa Visual Studio de las limitaciones en el agregador de proyectos que impide que los tipos de proyecto de código administrado funcionen correctamente. Los pasos siguientes describen cómo cambiar el VSPackage para usar el nuevo agregador.

1. Quite el proyecto NativeHierarchyWrapper de la solución.
2. Quite los archivos binarios NativeHierarchyWrapper de la configuración.
3. Agregue *ProjectAggregator2.msi* a la configuración.

Cómo: Generar información de registro para un instalador

06/04/2020 • 4 minutes to read • [Edit Online](#)

La utilidad *RegPkg.exe* se puede usar para generar un manifiesto de registro para un VSPackage administrado. El manifiesto se puede incorporar en un paquete de instalación de Windows Installer. RegPkg también puede generar un archivo que se puede incluir en un archivo de origen de instalación basado en el conjunto de herramientas XML de [Windows Installer](#).

IMPORTANT

RegPkg genera nombres de ruta de acceso que son específicos del sistema de desarrollo, por lo que cada vez que utilice RegPkg, debe editar la salida para usar las propiedades con formato de Windows Installer adecuadas. Por ejemplo, `InprocServer32` el valor debe ser * <SystemFolder>mscorlib.dll* y las rutas de acceso deben usar * <> #filekey* y * <\$componentkey>*. Ajustar la salida de esta manera admite equipos con Windows instalado en una unidad diferente o en un directorio diferente, nombres de directorio localizados y rutas de acceso que los usuarios pueden elegir. Para obtener más información, consulte [Con formato](#) en el SDK de Windows Installer. Si sigue las convenciones de RegPkg para las rutas de acceso del sistema de desarrollo (por ejemplo, los ids de archivo del formulario *File_ <nombre>* de archivo), debe realizar menos cambios.

Para crear un manifiesto de registro

- Ejecute RegPkg con el modificador **/regfile**. Proporcione cualquier otro modificador, el nombre del archivo de salida y la ruta de acceso del VSPackage.

Por ejemplo, en el símbolo del sistema, escribiría algo similar a lo siguiente:

```
<Visual Studio SDK installation path>\VisualStudioIntegration\Tools\Bin\RegPkg /regfile:MyRegFile.reg  
MyPackage.dll
```

Para ver un manifiesto de registro

- Abra el manifiesto de registro en cualquier editor de texto.

El siguiente ejemplo es el manifiesto de registro que RegPkg crea para el servicio de lenguaje IronPython:

REGEDIT4

```
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python]
@="{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}"
"DisplayName"="131"
"IndexPath"="C:\\VSSDK80\\2006.07\\VisualStudioIntegration\\Samples\\IronPythonIntegration\\bin\\Release\\CodeSnippets\\SnippetsIndex.xml"
"LangStringId"="python"
"Package"="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}"
"ShowRoots"=dword:00000000
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python\ForceCreateDirs]
"Python"="C:\\VSSDK80\\2006.07\\VisualStudioIntegration\\Samples\\IronPythonIntegration\\bin\\Release\\CodeSnippets\\Snippets\\;%MyDocs%\Code Snippets\Python\My Code Snippets\"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python\Paths]
"Python"="C:\\VSSDK80\\2006.07\\VisualStudioIntegration\\Samples\\IronPythonIntegration\\bin\\Release\\CodeSnippets\\Snippets\\;%MyDocs%\Code Snippets\Python\My Code Snippets\"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Packages]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Packages\{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}]
@="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonPackage, IronPython.LanguageService, Version=1.0.2373.36479, Culture=neutral, PublicKeyToken=null"
"InprocServer32"="C:\\WINNT\\system32\\mscoree.dll"
"Class"="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonPackage"
"Assembly"="IronPython.LanguageService, Version=1.0.2373.36479, Culture=neutral, PublicKeyToken=null"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\File Extensions]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\File Extensions\py]
@="{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\Language Services]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Languages\Language Services\Python]
@="{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}"
"Package"="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}"
"LangResID"=dword:00000064
"ShowMatchingBrace"=dword:00000001
"CodeSense"=dword:00000001
"MatchBraces"=dword:00000001
"EnableCommenting"=dword:00000001
"ShowCompletion"=dword:00000001
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Packages]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Packages\{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}]
"ID"=dword:00000001
"MinEdition"="standard"
"ProductVersion"="1.0"
"ProductName"="Visual Studio Integration of IronPython Language Service"
"CompanyName"="Microsoft Corporation"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Services]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Services\{923b4811-26e4-4347-ac8a-244762798e1c}]
@="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}"
"Name"="IPythonLibraryManager"
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Services]
[HKEY_LOCAL_MACHINE\Software\Microsoft\VisualStudio\8.0\Services\{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}]
@="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}"
"Name"="Python"
```

Para crear un archivo de inclusión XML de Windows Installer

- Ejecute RegPkg con el modificador **/wixfile**. Proporcione cualquier otro modificador, el nombre del archivo de salida y la ruta de acceso del VSPackage.

Por ejemplo, en el símbolo del sistema, escribiría algo similar a lo siguiente:

```
<Visual Studio SDK installation path>\VisualStudioIntegration\Tools\Bin\RegPkg /codebase
/wixfile:IronPython.LanguageService.wxi ..\bin\Release\IronPython.LanguageService.dll
```

Para ver un archivo de inclusión XML de Windows Installer

- Abra el archivo de inclusión XML de Windows Installer en cualquier editor de texto.

El siguiente ejemplo es el archivo de inclusión que RegPkg crea para el servicio de lenguaje IronPython:

```
<Include>

  <Registry Root="HKLM"
Key="Software\Microsoft\VisualStudio\8.0\Languages\IntellisenseProviders\IronPythonCodeProvider">
  <Registry Name="GUID" Value="{9c1807ea-d222-4775-afa8-c092c580e451}" Type="string" />
  <Registry Name="AddItemLanguageName" Value="Iron Python" Type="string" />
  <Registry Name="DefaultExtension" Value=".py" Type="string" />
  <Registry Name="ShortLanguageName" Value="IronPython;Python" Type="string" />
  <Registry Name="TemplateFolderName" Value="IronPython" Type="string" />
</Registry>

  <Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python"
Value="{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}" Type="string">
  <Registry Name="DisplayName" Value="131" Type="string" />
  <Registry Name="IndexPath"
Value="C:\VSSDK80\2006.08\VisualStudioIntegration\Samples\IronPythonIntegration\Setup\
[$ComponentPath]\CodeSnippets\SnippetsIndex.xml" Type="string" />
  <Registry Name="LangStringId" Value="python" Type="string" />
  <Registry Name="Package" Value="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}" Type="string" />
  <Registry Name="ShowRoots" Value="0" Type="integer" />
</Registry>

  <Registry Root="HKLM"
Key="Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python\ForceCreateDirs">
  <Registry Name="Python"
Value="C:\VSSDK80\2006.08\VisualStudioIntegration\Samples\IronPythonIntegration\Setup\
[$ComponentPath]\CodeSnippets\Snippets\;%MyDocs%\Code Snippets\Python\My Code Snippets\
Type="string" />
</Registry>

  <Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Languages\CodeExpansions\Python\Paths">
  <Registry Name="Python"
Value="C:\VSSDK80\2006.08\VisualStudioIntegration\Samples\IronPythonIntegration\Setup\
[$ComponentPath]\CodeSnippets\Snippets\;%MyDocs%\Code Snippets\Python\My Code Snippets\
Type="string" />
</Registry>

  <Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Languages\File Extensions\ .py" Value="
{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}" Type="string" />

  <Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Languages\Language Services\Python"
Value="{ae8ce01a-b3ff-4c19-8c80-54669c197f2c}" Type="string">
  <Registry Name="Package" Value="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}" Type="string" />
  <Registry Name="LangResID" Value="100" Type="integer" />
  <Registry Name="ShowCompletion" Value="1" Type="integer" />
  <Registry Name="ShowMatchingBrace" Value="1" Type="integer" />
  <Registry Name="CodeSense" Value="1" Type="integer" />
  <Registry Name="MatchBraces" Value="1" Type="integer" />
  <Registry Name="EnableCommenting" Value="1" Type="integer" />
  <Registry Name="DefaultToInsertSpaces" Value="1" Type="integer" />
</Registry>

  <Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Packages\{1b05e2b4-7c21-4f63-910e-
29fe55eb5f8b}" Value="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonPackage,
IronPython.LanguageService, Version=1.0.2394.27719, Culture=neutral, PublicKeyToken=null" Type="string">
  <Registry Name="InprocServer32" Value="[SystemFolder]mscoree.dll" Type="string" />
```

```

    <Registry Name="Class"
Value="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonPackage" Type="string" />
    <Registry Name="CodeBase" Value="#File_IronPython.LanguageService.dll" Type="string" />
    <Registry Name="ID" Value="1" Type="integer" />
    <Registry Name="MinEdition" Value="standard" Type="string" />
    <Registry Name="ProductVersion" Value="1.0" Type="string" />
    <Registry Name="ProductName" Value="Visual Studio Integration of IronPython Language Service"
Type="string" />
    <Registry Name="CompanyName" Value="Microsoft Corporation" Type="string" />
</Registry>

<Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\CLSID\{9c1807ea-d222-4775-afa8-
c092c580e451}"
Value="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonIntellisenseProvider"
Type="string">
    <Registry Name="InprocServer32" Value="[SystemFolder]mscoree.dll" Type="string" />
    <Registry Name="Class"
Value="Microsoft.Samples.VisualStudio.IronPythonLanguageService.PythonIntellisenseProvider"
Type="string" />
    <Registry Name="CodeBase" Value="#File_IronPython.LanguageService.dll" Type="string" />
    <Registry Name="ThreadingModel" Value="Both" Type="string" />
</Registry>

<Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Services\{923b4811-26e4-4347-ac8a-
244762798e1c}" Value="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}" Type="string">
    <Registry Name="Name" Value="IPythonLibraryManager" Type="string" />
</Registry>

<Registry Root="HKLM" Key="Software\Microsoft\VisualStudio\8.0\Services\{ae8ce01a-b3ff-4c19-8c80-
54669c197f2c}" Value="{1b05e2b4-7c21-4f63-910e-29fe55eb5f8b}" Type="string">
    <Registry Name="Name" Value="Python" Type="string" />
</Registry>
</Include>

```

Vea también

- [Registrar VSPackages](#)
- [VSPackages](#)

Comandos que se deben ejecutar después de la instalación

06/04/2020 • 5 minutes to read • [Edit Online](#)

Si implementa la extensión a través de un archivo *.msi*, debe ejecutar **devenv /setup** como parte de la instalación para que Visual Studio detecte las extensiones.

NOTE

La información de este tema se aplica a la búsqueda de *devenv.exe* con Visual Studio 2008 y versiones anteriores. Para obtener información acerca de cómo detectar *devenv.exe* con versiones posteriores de Visual Studio, vea [Detectar requisitos del sistema](#).

Encontrar devenv.exe

Puede localizar *devenv.exe* de cada versión Visual Studio a partir de los valores del Registro que escriben los instaladores, mediante la tabla RegLocator y las tablas de AppSearch para almacenar los valores del Registro como propiedades. Para obtener más información, consulte [Detectar requisitos del sistema](#).

Filas de tabla RegLocator para localizar devenv.exe de diferentes versiones de Visual Studio

SIGNATURE	ROOT	CLAVE	NOMBRE	TIPO
RL_DevenvExe_2002	2	SOFTWARE-Microsoft-VisualStudio-7.0-Setup-VS	EnvironmentPath	2
RL_DevenvExe_2003	2	SOFTWARE-Microsoft-VisualStudio-7.1-Setup-VS	EnvironmentPath	2
RL_DevenvExe_2005	2	SOFTWARE-Microsoft-VisualStudio-8.0-Setup-VS	EnvironmentPath	2
RL_DevenvExe_2008	2	SOFTWARE-Microsoft-VisualStudio-9.0-Setup-VS	EnvironmentPath	2

Filas de tabla de AppSearch para las filas de tabla RegLocator correspondientes

PROPIEDAD	SIGNATURE
DEVENV_EXE_2002	RL_DevenvExe_2002
DEVENV_EXE_2003	RL_DevenvExe_2003

PROPIEDAD	SIGNATURE
DEVENV_EXE_2005	RL_DevenvExe_2005
DEVENV_EXE_2008	RL_DevenvExe_2008

Por ejemplo, el instalador de Visual Studio escribe el valor del Registro de **HKEY_LOCAL_MACHINE, SOFTWARE, Microsoft, VisualStudio, 9,0, Configuración, VS, EnvironmentPath**, como *C:- VS2008, Common7, IDE, devenv.exe*, una ruta de acceso completa al archivo ejecutable que debe ejecutar el instalador.

NOTE

Dado que la columna Type de la tabla RegLocator es 2, no es necesario especificar información de versión adicional en la tabla Signature.

Ejecutar devenv.exe

Después de que la acción estándar de AppSearch se ejecuta en el instalador, cada propiedad de la tabla AppSearch tiene un valor que apunta al archivo *devenv.exe* para la versión correspondiente de Visual Studio. Si alguno de los valores del Registro especificados no está presente, porque esa versión de Visual Studio no está instalada, la propiedad especificada se establece en null.

Windows Installer admite la ejecución de un ejecutable al que apunta una propiedad a través del tipo de acción personalizado 50. La acción personalizada debe incluir las `msidbCustomActionTypeInScript` opciones de ejecución en `msidbCustomActionTypeCommit` script, (1024) y (512), para Visual Studio asegurarse de que el VSPackage se ha instalado correctamente antes de integrarlo en . Para obtener más información, consulte [Tabla CustomAction](#) y Opciones de ejecución de [acción personalizada en script](#).

Las acciones personalizadas de tipo 50 especifican la propiedad que contiene el ejecutable como el valor de la columna Origen y los argumentos de línea de comandos de la columna Destino.

Filas de tabla CustomAction para ejecutar devenv.exe

ACCIÓN	TIPO	SOURCE	DESTINO
CA_RunDevenv2002	1586	DEVENV_EXE_2002	/setup
CA_RunDevenv2003	1586	DEVENV_EXE_2003	/setup
CA_RunDevenv2005	1586	DEVENV_EXE_2005	/setup
CA_RunDevenv2008	1586	DEVENV_EXE_2008	/setup

Las acciones personalizadas se deben crear en la tabla InstallExecuteSequence para programarlas para su ejecución durante la instalación. Utilice la propiedad correspondiente en cada fila de la columna Condición Visual Studio para evitar que se ejecute la acción personalizada si esa versión de no está instalada en el sistema.

NOTE

Las propiedades con valores `False` null se evalúan cuando se usan en condiciones.

El valor de la columna Secuencia para cada acción personalizada depende de otros valores de secuencia del paquete de Windows Installer. Los valores de secuencia deben ser tales que las acciones personalizadas

devenv.exe se ejecuten lo más cerca posible de inmediatamente antes de la acción estándar InstallFinalize.

InstallExecuteSequence tabla para programar las acciones personalizadas **devenv.exe**

ACCIÓN	CONDICIÓN	SECUENCIA
CA_RunDevenv2002	DEVENV_EXE_2002	6602
CA_RunDevenv2003	DEVENV_EXE_2003	6603
CA_RunDevenv2005	DEVENV_EXE_2005	6605
CA_RunDevenv2008	DEVENV_EXE_2008	6608

Vea también

- [Instalar VSPackages con Windows Installer](#)

Desinstalación de un VSPackage con Windows Installer

06/04/2020 • 3 minutes to read • [Edit Online](#)

En su mayor parte, Windows Installer puede desinstalar el VSPackage simplemente "deshacer" lo que hizo para instalar el VSPackage. Las acciones personalizadas descritas en [Comandos que deben ejecutarse después](#) de la instalación también deben ejecutarse después de una desinstalación. Dado que las llamadas a devenv.exe se producen justo antes de la acción estándar InstallFinalize para la instalación y la desinstalación, las entradas de tabla CustomAction e InstallExecuteSequence sirven en ambos casos.

NOTE

Ejecutar `devenv /setup` después de desinstalar un paquete MSI.

Como regla general, si agrega acciones personalizadas a un paquete de Windows Installer, debe controlar esas acciones durante la desinstalación y la reversión. Si agrega acciones personalizadas para registrar automáticamente el VSPackage, por ejemplo, también debe agregar acciones personalizadas para anular el registro.

NOTE

Es posible que un usuario instale el VSPackage Visual Studio y, a continuación, desinstale las versiones con las que está integrado. Puede ayudar a asegurarse de que la desinstalación de VSPackage funciona en Visual Studio ese escenario eliminando las acciones personalizadas que ejecutan código con dependencias en .

Manejo de las condiciones de lanzamiento en el momento de la desinstalación

La acción estándar LaunchConditions lee las filas de la tabla LaunchCondition para mostrar mensajes de error si no se cumplen las condiciones. Como las condiciones de lanzamiento se usan generalmente para garantizar que se cumplen los `NOT Installed` requisitos del sistema, generalmente puede omitir las condiciones de lanzamiento durante la desinstalación agregando la condición, , a la fila LaunchConditions de la tabla LaunchCondition.

Una alternativa es `OR Installed` agregar a las condiciones de lanzamiento que no son importantes durante la desinstalación. Esto garantiza que la condición siempre será verdadera durante la desinstalación y, por lo tanto, no mostrará el mensaje de error de condición de inicio.

NOTE

`Installed` es la propiedad que Windows Installer establece cuando detecta que el VSPackage ya se ha instalado en el sistema.

Vea también

- [Windows Installer](#)
- [Detección de requisitos del sistema](#)

SDK del Visor de Ayuda de Microsoft

06/04/2020 • 68 minutes to read • [Edit Online](#)

Este artículo contiene las siguientes tareas para los integradores de Visual Studio Help Viewer:

- Creación de un tema (soporte F1)
- Creación de un paquete de marca de contenido del Visor de ayuda
- Despliegue de un conjunto de artículos
- Agregar ayuda al shell de Visual Studio (integrado o aislado)
- Recursos adicionales

Crear un tema (soporte F1)

En esta sección se proporciona información general sobre los componentes de un tema presentado, los requisitos del tema, una breve descripción de cómo crear un tema (incluidos los requisitos de compatibilidad con F1) y, por último, un tema de ejemplo con su resultado representado.

Descripción general del tema del visor de ayuda

Cuando se llama a un tema para la representación, el Visor de ayuda obtiene los elementos del paquete de personalización de marca que están asociados con el tema en el momento de la instalación o la última actualización, junto con el tema XHTML, y combina los dos para dar como resultado la vista de contenido presentada (datos de marca + datos de tema). El paquete de personalización de marca contiene logotipos, compatibilidad con comportamientos de contenido y texto de personalización de marca (copyright, etc.). Consulte "Creación de paquete de personalización de marca" a continuación para obtener más información sobre los elementos del paquete de personalización de marca. En caso de que no haya ningún paquete de personalización de marca asociado al tema, el Visor de Ayuda usará el paquete de personalización de marca de reserva ubicado en la raíz de la aplicación Visor de ayuda (Branding_en-US.mshc).

Ayudar a los requisitos del tema del espectador

Para representarse correctamente en el Visor de ayuda, el contenido del tema sin formato debe ser W3C Basic 1.1 XHTML.

Un tema normalmente contiene dos secciones:

- Metadatos (consulte Referencia de metadatos de contenido): datos sobre el tema, por ejemplo, el identificador único del tema, el valor de la palabra clave, el identificador de TDC del tema, el identificador de nodo primario, etc.
- Contenido corporal: compatible con W3C Basic 1.1 XHTML, que incluye comportamientos de contenido admitidos (área plegable, fragmento de código, etc. A continuación se muestra una lista completa).

Controles compatibles con El paquete de personalización de marca de Visual Studio:

- Vínculos
- CodeSnippet
- CollapsibleArea
- Miembro heredado

- LanguageSpecificText

Cadenas de idioma admitidas (no distinguen entre mayúsculas y minúsculas):

- javascript
- csharp o c #
- cplusplus o visualc++ o c++
- jscript
- visualbasic o vb
- f o fsharp o fs
- otro - una cadena que representa un nombre de idioma

Creación de un tema del Visor de ayuda

Cree un nuevo documento XHTML denominado ContosoTopic4.htm e incluya la etiqueta de título (abajo).

```
<html>
<head>
<title>Contoso Topic 4</title>
</head>

<body>

</body>
</html>
```

A continuación, agregue datos para definir cómo se presentará el tema (automarcado o no), cómo hacer referencia a este tema para F1, donde existe este tema dentro de la TDT, su identificador (para referencia de vínculo por otros temas), etc. Consulte la tabla "Metadatos de contenido" a continuación para obtener una lista completa de los metadatos admitidos.

- En este caso, usaremos nuestro propio paquete de personalización de marca, una variante del paquete de personalización de marca visor de ayuda de Visual Studio.
- Agregue el metanombre y el valor de F1 ("Microsoft.Help.F1" content" ContosoTopic4") que coincidirá con el valor F1 proporcionado en el contenedor de propiedades IDE. (Consulte la sección Soporte de F1 para obtener más información.) Este es el valor que coincide con la llamada F1 desde dentro del IDE para mostrar este tema cuando se elige F1 en el IDE.
- Agregue el identificador del tema. Esta es la cadena que usan otros temas para vincular a este tema. Es el identificador del Visor de ayuda para este tema.
- Para el TD.TC., agregue el nodo primario de este tema para definir dónde aparecerá este nodo de TC de tema.
- Para la TD.T., agregue el orden de nodo de este tema. Cuando el nodo primario tiene un número de nodos secundarios, defina en el orden de los nodos secundarios la ubicación de este tema. Por ejemplo, este tema es el número 4 de 4 temas secundarios.

Ejemplo de sección de metadatos:

```

<html>
<head>
<title>Contoso Topic 4</title>

<meta name="SelfBranded" content="false" />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="Microsoft.Help.Id" content="ContosoTopic4" />
<meta name="Microsoft.Help.F1" content=" ContosoTopic4" />
    <meta name="Language" content="en-us" />
<meta name="Microsoft.Help.TocParent" content="ContosoTopic0" />
<meta name="Microsoft.Help.TocOrder" content="4" />

</head>

<body>

</body>
</html>

```

El cuerpo del tema

El cuerpo (sin incluir el encabezado y el pie de página) del tema contendrá vínculos de página, una sección de nota, un área contraíble, un fragmento de código y una sección de texto específico del idioma. Consulte la sección de personalización de marca para obtener información sobre las áreas del tema presentado.

1. Agregue una etiqueta de título de tema: `<div class="title">Contoso Topic 4</div>`

2. Agregue una sección de notas: `<div class="alert"> add your table tag and text </div>`

3. Agregue un área plegable:

```
<CollapsibleArea Expanded="1" Title="Collapsible Area Test Heading"> add text </CollapsibleArea>
```

4. Agregue un fragmento de código:

```
<CodeSnippet EnableCopyCode="true" Language="CSharp" ContainsMarkup="false" DisplayLanguage="C#" > a
block of code </CodeSnippet>
```

5. Añadir texto específico del

```
<LanguageSpecificText devLangcs="CS" devLangvb="VB" devLangcpp="C++" devLangnu="F#" /> idioma
devLangnu=
devLangnu="Fortran" muestra Fortran cuando el fragmento de código DisplayLanguage - Fortran
```

6. Añadir enlaces de página: `<a href="ms-xhelp:///?Id=ContosoTopic1">Main Topic</a>`

NOTE

Nota: para el nuevo "Lenguaje de visualización" no compatible (ejemplo, F, Cobol, Fortran) la coloración del código en el fragmento de código será monocroma.

Ejemplo de tema del visor de ayuda El código ilustra cómo definir metadatos, un fragmento de código, un área contraíble y texto específico del lenguaje.

```

<?xml version="1.0" encoding="utf-8"?>
<html>
<head>
<title>Contoso Topic 4</title>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="Microsoft.Help.Id" content="ContosoTopic4" />
<meta name="Microsoft.Help.F1" content=" ContosoTopic4" />
    <meta name="Language" content="en-us" />
<meta name="Microsoft.Help.TocParent" content="ContosoTopic0" />

```

```

<meta name="Microsoft.Help.TocOrder" content="ContosoTopic4" />
<meta name="Microsoft.Help.TocOrder" content="4" />
<meta name="SelfBranded" content="false" />
</head>

<body>
<div class="title">Contoso Topic 4</div>

    <div id="header">
<table id="bottomTable" cellpadding="0" cellspacing="0" width="100%">
    <tr id="headerTableRow2"><td align="left">
        <span id="nsrTitle">Contoso Topic 1</span>
    </td>
<td align="right">
</td></tr>
<tr id="headerTableRow1"><td align="left">
        <span id="runningHeaderText">Contoso Widgets & Sprockets</span>
    </td></tr>
</table>
</div>

<h2>Table of Contents</h2>

<ul class="toc">
<li class="tocline1"><a href="#introduction" target="_self">1.0 Introduction</a></li>
<li class="tocline1"><a href="#seealso" target="_self">See Also</a></li>
</ul>

<div class="topic">

<div id="mainSection">
<div id="mainBody">

<a name="introduction"></a>
<h2>1.0 Introduction</h2>
<p>[This documentation is for sample purposes only.]</p>

<p>Contoso Topic 1 contains examples of:
<ul>
<li>Collapsible Area</li>
<li>Bookmark ("See also")</li>
<li>Code Snippets from Branding Package</li>
</ul>
</p>
<div class="alert"><table><tr><th>
<strong>Note </strong></th></tr>
<tr><td>
<p>This is an example of a <span class="label">Note </span>section.
Call out important items for your reader in this <span class="label">Note </span>box.
</p></td></tr>
</table>
</div>
</div>

<CollapsibleArea Expanded="1" Title="Collapsible Area Test Heading">

    <a id="sectionToggle0"><!--</a>

<div>
Example of Collapsible Area
<br/>
Lorem ipsum dolor sit amet...
</div>
</CollapsibleArea>

<div id="snippetGroup" >
<CodeSnippet EnableCopyCode="true" Language="VisualBasic" ContainsMarkup="false" DisplayLanguage="Visual
Basic" >
Private Sub ToolStripRenderer1_RenderGrip(sender as Object, e as ToolStripGripRenderEventArgs) _ Handles
ToolStripRenderer1.RenderGrip

```

```

ToolStripRenderer1.RenderGrip
Dim messageBoxVB as New System.Text.StringBuilder()
    messageBoxVB.AppendFormat("{0} = {1}", "GripBounds", e.GripBounds)
    messageBoxVB.AppendLine()
    ...
    MessageBox.Show(messageBoxVB.ToString(),"RenderGrip Event")
End Sub
</CodeSnippet>

<CodeSnippet EnableCopyCode="true" Language="CSharp" ContainsMarkup="false" DisplayLanguage="C#" >
private void ToolStripRenderer1_RenderGrip(Object sender, ToolStripGripRenderEventArgs e)
{
System.Text.StringBuilder messageBoxCS = new System.Text.StringBuilder();
messageBoxCS.AppendFormat("{0} = {1}", "GripBounds", e.GripBounds );
messageBoxCS.AppendLine();
...
MessageBox.Show(messageBoxCS.ToString(), "RenderGrip Event" );
}
</CodeSnippet>

<CodeSnippet EnableCopyCode="true" Language="fsharp" ContainsMarkup="false" DisplayLanguage="F#" >
some F# code
</CodeSnippet>
</div>

<h4 class="subHeading">Example of code specific text</h4>Language = <LanguageSpecificText devLangcs="CS"
devLangvb="VB" devLangcpp="C++" devLangnu="F#" />

<a name="seealso"></a>
<h1 class="heading">See Also</h1>

    <div id="seeAlsoSection" class="section">
    <div class="seeAlsoStyle">
        <a href="ms-xhelp:///Id=ContosoTopic1">Main Topic</a>
    </div>
    </div>
</div>
</div>
</body>
</html>

```

Soporte F1

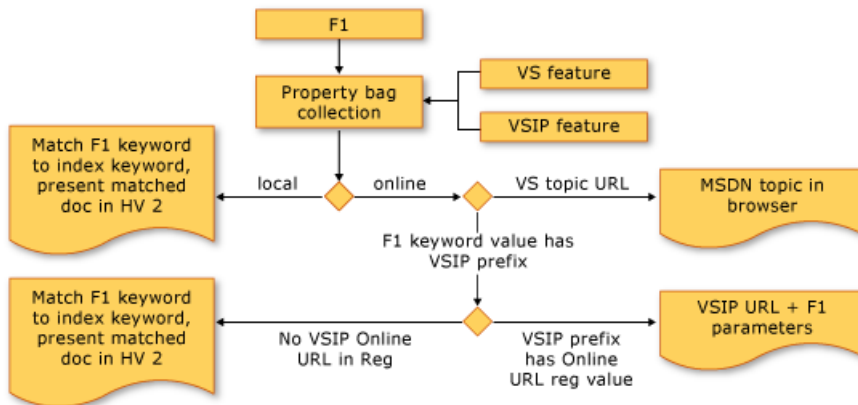
En Visual Studio, al seleccionar F1 se generan los valores proporcionados desde la colocación del cursor dentro del IDE y se rellena una "bolsa de propiedades" con los valores proporcionados (en función de la ubicación del cursor). Cuando el cursor está sobre la característica x, la característica x está activa/en foco y rellena la bolsa de propiedades con valores. Cuando se selecciona F1, el contenedor de propiedades se rellena y el código de Visual Studio F1 busca si el origen de la Ayuda predeterminado de los clientes es local o en línea (en línea es el valor predeterminado), crea la cadena adecuada en función de la configuración de los usuarios (en línea es el valor predeterminado) - shell execute (consulte la Guía del administrador de ayuda para los parámetros de inicio de exe) con parámetros para el visor de ayuda local + palabras clave del contenedor de propiedades si la ayuda local es la predeterminada , o la URL de MSDN con la palabra clave en la lista de parámetros.

Si se devuelven tres cadenas para F1, denominada cadena de varios valores, tome el primer término, busque un hit y, si se encuentra, hemos terminado; si no, pase a la siguiente cadena. El orden importa. La presentación de las palabras clave de varios valores debe ser la cadena más larga a la cadena más corta. Para comprobar esto en el caso de las palabras clave de varios valores, mire la cadena de URL F1 en línea, que incluirá la palabra clave elegida.

En Visual Studio 2012, intencionalmente creamos una división más fuerte entre en línea y sin conexión, de modo que si la configuración del usuario era para Online, simplemente pasamos la solicitud F1 directamente a nuestro servicio de consulta en línea en MSDN en lugar de enrutar a través del agente de biblioteca de ayuda que teníamos en Visual Studio 2010. A continuación, nos basamos en un estado de "contenido de proveedor instalado

- true" para determinar si se debe hacer algo diferente en ese contexto. Si es true, realizamos esta lógica de análisis y enrutamiento en función de lo que desee admitir para sus clientes. Si es false, entonces simplemente vamos a MSDN. Si la configuración del usuario es Local, todas las llamadas van al motor de ayuda local.

Diagrama de flujo F1:



Cuando el origen de contenido de ayuda predeterminado del Visor de Ayuda se establece en línea (Iniciar en el explorador):

- Las características de Visual Studio Partner (VSP) emiten un valor al contenedor de propiedades F1 (prefijo.keyword de bolsa de propiedades y dirección URL en línea para el prefijo que se encuentra en el registro): F1 envía un VSP URL+ parámetros al explorador.
- Características de Visual Studio (editor de idioma, elementos de menú específicos de Visual Studio, etc.): F1 envía una dirección URL de Visual Studio al explorador.

Cuando el origen de contenido de ayuda predeterminado del Visor de Ayuda se establece en Ayuda local (Iniciar en el Visor de Ayuda):

- Características de VSP donde la palabra clave coincide entre el contenedor de propiedades F1 y el índice de tienda local (es decir, el prefijo de contenedor de propiedades.keyword , el valor que se encuentra en el índice de almacén local): F1 representa el tema en el Visor de ayuda.
- Características de Visual Studio (sin opción para que el VSP invalide el contenedor de propiedades emitido desde las características de Visual Studio): F1 representa un tema de Visual Studio en el Visor de ayuda.

Establezca los siguientes valores del Registro para habilitar la reserva F1 para el contenido de la Ayuda del proveedor. F1 Fallback significa que el Visor de Ayuda está configurado para buscar contenido de Ayuda F1 en línea y el contenido del proveedor se instala localmente en el disco duro de los usuarios. El Visor de ayuda debe buscar la Ayuda local para el contenido aunque la configuración predeterminada sea para la ayuda en línea.

1. Establezca el valor **VendorContent** en la clave del Registro de la Ayuda 2.3:

- Para sistemas operativos de 32 bits:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Help\v2.3\Catalogs\VisualStudio15

"VendorContent"-dword:00000001

- Para sistemas operativos de 64 bits:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Ayuda\v2,3,Catálogos, VisualStudio15

"VendorContent"-dword:00000001

2. Registre el espacio de nombres del asociado en la clave del Registro de la Ayuda 2.3:

- Para sistemas operativos de 32 bits:

HKEY_LOCAL_MACHINE\SOFTWARE, Microsoft, Ayuda, v2.3, Espacio de nombres de <de socios |>

"ubicación" "fuera de línea"

- Para sistemas operativos de 64 bits:

HKEY_LOCAL_MACHINE\SOFTWARE, Wow6432Node, Microsoft, Ayuda, v2.3, <espacio de nombres de la |<de trabajo>

"ubicación" "fuera de línea"

Análisis de espacio de nombres nativo base

Para activar el análisis de espacios de nombres nativos base, en el registro agregue un nuevo DWORD con el nombre de: BaseNativeNamespaces y establezca su valor en 1 (en la clave de catálogo que desea admitir). Por ejemplo, si desea usar el catálogo de Visual Studio, puede agregar la clave a la ruta de acceso:

HKEY_LOCAL_MACHINE\SOFTWARE, Wow6432Node, Microsoft, Ayuda, v2.3, Catálogos, VisualStudio15

Cuando se encuentra una palabra clave F1 con el formato HEADER/METHOD, se analizará el carácter '/', lo que dará como resultado la siguiente construcción:

- HEADER: será el espacio de nombres que se puede utilizar para registrarse en el registro
- METHOD: esto se convertirá en la palabra clave que se pasa a través.

Por ejemplo, dada una biblioteca personalizada denominada CustomLibrary y un método denominado MyTestMethod, `CustomLibrary/MyTestMethod` cuando una solicitud F1 entra en ella se formateará como .

A continuación, un usuario puede registrar CustomLibrary como espacio de nombres en el subárbol Partners y proporcionar la clave de ubicación que desee y la palabra clave que se pasa a la consulta será MyTestMethod.

Habilitar la herramienta de depuración de ayuda en el IDE

Agregue la siguiente clave y valor del Registro:

HKEY_CURRENT_USER\Software, Microsoft, VisualStudio, 15.0, Ayuda dinámica

HKEY_CURRENT_USER\Software, Microsoft, VisualStudio, 16.0, Ayuda dinámica

Valor: Mostrar salida de depuración en datos comerciales: Sí

En el IDE, en el elemento de menú Ayuda, seleccione **Depurar** contextode ayuda .

Metadatos de contenido

En la tabla siguiente, cualquier cadena que aparece entre corchetes es un marcador de posición que debe reemplazarse por un valor reconocido. Por ejemplo, <en el nombre meta "Microsoft.Help.Locale" content "[código de idioma]" />, "[código de idioma]" debe reemplazarse por un valor como "en-us".

PROPIEDAD (REPRESENTACIÓN HTML)	DESCRIPCIÓN

PROPIEDAD (REPRESENTACIÓN HTML)	DESCRIPCIÓN
<code><metaname"Microsoft.Help.Locale" content"[idioma-código]" /></code>	Establece una configuración regional para este tema. Si esta etiqueta se utiliza en un tema, debe usarse una sola vez y debe insertarse encima de cualquier otra etiqueta de Ayuda de Microsoft. Si no se utiliza esta etiqueta, el texto del cuerpo del tema se indiza mediante el separador de palabras asociado a la configuración regional del producto, si se especifica; de lo contrario, se utiliza el separador de palabras en-us. Esta etiqueta cumple con ISO C RFC 4646. Para asegurarse de que la Ayuda de Microsoft funciona correctamente, use esta propiedad en lugar del atributo Language general.
<code><metaname"Microsoft.Help.TopicLocale" content"[idioma-código]" /></code>	Establece una configuración regional para este tema cuando también se usan otras configuraciones regionales. Si esta etiqueta se utiliza en un tema, debe usarse una sola vez. Utilice esta etiqueta cuando el catálogo contenga contenido en más de un idioma. Varios temas de un catálogo pueden tener el mismo identificador, pero cada uno debe especificar un TopicLocale único. El tema que especifica un TopicLocale que coincide con la configuración regional del catálogo es el tema que se muestra en la tabla de contenido. Sin embargo, todas las versiones de idioma del tema se muestran en los resultados de búsqueda.
<code><título>[Título]</título></code>	Especifica el título de este tema. Esta etiqueta es necesaria y debe usarse una sola vez en un tema. Si el cuerpo del tema no <contiene un div de título> sección, este título se muestra en el tema y en la tabla de contenido.
<code><meta name "Microsoft.Help.Keywords" content "[aKeywordPhrase]" /></code>	Especifica el texto de un vínculo que se muestra en el panel de índice del Visor de Ayuda. Cuando se hace clic en el vínculo, se muestra el tema. Puede especificar varias palabras clave de índice para un tema o puede omitir esta etiqueta si no desea que los vínculos a este tema aparezcan en el índice. Las palabras clave "K" de versiones anteriores de la Ayuda se pueden convertir a esta propiedad.
<code><metaname"Microsoft.Help.Id" content"[TopicID]" /></code>	Establece el identificador de este tema. Esta etiqueta es necesaria y debe usarse una sola vez en un tema. El identificador debe ser único entre los temas del catálogo que tienen la misma configuración regional. En otro tema, puede crear un vínculo a este tema mediante este identificador.
<code><metaname"Microsoft.Help.F1" content "[System.Windows.Controls.Primitives.IRecyclingItemContainer Generator]" /></code>	Especifica la palabra clave F1 para este tema. Puede especificar varias palabras clave F1 para un tema o puede omitir esta etiqueta si no desea que este tema se muestre cuando un usuario de la aplicación presione F1. Normalmente, solo se especifica una palabra clave F1 para un tema. Las palabras clave "F" de versiones anteriores de la Ayuda se pueden convertir a esta propiedad.
<code><meta name "Descripción" content"[descripción del tema]" /></code>	Proporciona un breve resumen del contenido de este tema. Si esta etiqueta se utiliza en un tema, debe usarse una sola vez. La biblioteca de consultas tiene acceso a esta propiedad directamente; no se almacena en el archivo de índice.

PROPIEDAD (REPRESENTACIÓN HTML)	DESCRIPCIÓN
metaname"Microsoft.Help.TocParent" content"[parent_Id]"/>	Especifica el tema primario de este tema en la tabla de contenido. Esta etiqueta es necesaria y debe usarse una sola vez en un tema. El valor es el Microsoft.Help.Id del elemento primario. Un tema puede tener una sola ubicación en la tabla de contenido. "-1" se considera el identificador de tema para la raíz de TCA. En Visual Studio 2013, esa página es la página de inicio del Visor de ayuda. Esta es la misma razón por la que agregamos específicamente TocParent-1 a algunos temas para asegurarnos de que aparecen en el nivel superior. La página de inicio del Visor de ayuda es una página del sistema, por lo que no se puede reemplazar. Si un VSP intenta agregar una página con un identificador de -1, puede agregarse al conjunto de contenido, pero el Visor de Ayuda siempre usará la página del sistema - Inicio del Visor de Ayuda
<metaname"Microsoft.Help.TocOrder" content="[número entero positivo]"/>	Especifica dónde en la tabla de contenido aparece este tema en relación con sus temas del mismo nivel. Esta etiqueta es necesaria y debe usarse una sola vez en un tema. El valor es un entero. Un tema que especifica un entero de valor inferior aparece encima de un tema que especifica un entero de mayor valor.
<metaname"Microsoft.Help.Product" content="[código de producto]"/>	Especifica el producto que se describe en este tema. Si esta etiqueta se utiliza en un tema, debe usarse una sola vez. Esta información también se puede proporcionar como un parámetro que se pasa al indizador de ayuda.
<metaname"Microsoft.Help.ProductVersion" content="[número de versión]"/>	Especifica la versión del producto que se describe en este tema. Si esta etiqueta se utiliza en un tema, debe usarse una sola vez. Esta información también se puede proporcionar como un parámetro que se pasa al indizador de ayuda.
<metaname"Microsoft.Help.Category" content="[string]"/>	Utilizado por los productos para identificar subsecciones de contenido. Puede identificar varias subsecciones para un tema o puede omitir esta etiqueta si no desea que los vínculos identifiquen ninguna subsección. Esta etiqueta se utiliza para almacenar los atributos de TargetOS y TargetFrameworkMoniker cuando un tema se convierte desde una versión anterior de la Ayuda. El formato del contenido es AttributeName:AttributeValue.
<meta name"Microsoft.Help.TopicVersion content="[número de versión del tema]"/>	Especifica esta versión del tema cuando existen varias versiones en un catálogo. Dado que no se garantiza que Microsoft.Help.Id sea único, esta etiqueta es necesaria cuando existe más de una versión de un tema en un catálogo, por ejemplo, cuando un catálogo contiene un tema para .NET Framework 3.5 y un tema para .NET Framework 4 y ambos tienen la misma Microsoft.Help.Id.

PROPIEDAD (REPRESENTACIÓN HTML)	DESCRIPCIÓN
<metaname"SelfBranded" content"[TRUE o FALSE]"/>	Especifica si este tema usa el paquete de personalización de marca de inicio del Administrador de bibliotecas de ayuda o un paquete de personalización de marca específico para el tema. Esta etiqueta debe ser TRUE o FALSE. Si es TRUE, el paquete de personalización de marca para el tema asociado reemplaza el paquete de personalización de marca que se establece cuando se inicia el Administrador de la biblioteca de ayuda para que el tema se represente según lo previsto, incluso si difiere de la representación de otro contenido. Si es FALSE, el tema actual se representa según el paquete de personalización de marca que se establece cuando se inicia el Administrador de bibliotecas de ayuda. De forma predeterminada, el Administrador de bibliotecas de ayuda asume que la automarca es false a menos que la variable SelfBranded se declare como TRUE; por lo tanto, usted <no tiene que declarar el nombre meta "SelfBranded" content "FALSE"/>.

Crear un paquete de marca

La versión de Visual Studio abarca varios productos de Visual Studio diferentes, incluidos los shells aislados e integrados para los socios de Visual Studio. Cada uno de estos productos requiere cierto grado de soporte de personalización de marca de contenido de Ayuda basado en temas, único en el producto. Por ejemplo, los temas de Visual Studio deben tener una presentación de marca coherente, mientras que SQL StudioSQL Studio, que ajusta el Shell ISO, requiere su propia personalización de marca de contenido de Ayuda única para cada tema. Un socio de Shell integrado puede desear que sus temas de Ayuda estén dentro del contenido de Ayuda del producto primario de Visual Studio mientras mantiene su propia marca de tema.

Los paquetes de personalización de marca los instala el producto que contiene el Visor de ayuda. Para productos de Visual Studio:

- > Branding_ <El paquete de idioma del Visor de ayuda (por ejemplo, C:-Archivos de programa (x86)-Microsoft Help Viewer-v2.3) del Visor de ayuda (por ejemplo: C:-Archivos de programa (x86)-Microsoft Help Viewer-v2.3) del paquete de idioma del Visor de ayuda. Esto se utiliza para los casos en los que el paquete de personalización de marca del producto no está instalado (no se ha instalado ningún contenido) o en los que el paquete de personalización de marca instalado está dañado. Los elementos de Visual Studio (logotipo y comentarios) se omiten cuando se usa el paquete de personalización de marca de reserva raíz de la aplicación.
- Cuando se instala contenido de Visual Studio desde el servicio de paquete de contenido, también se instala un paquete de personalización de marca (por primera vez escenario de instalación de contenido). Si hay una actualización del paquete de personalización de marca, la actualización se instala cuando se produce la siguiente actualización de contenido o la acción de instalación del paquete adicional.

El Visor de Ayuda de Microsoft admite la personalización de marca de temas en función de los metadatos del tema.

- Donde los metadatos del tema definen la marca de sí mismos, representan el tema tal como está, no hacen nada (en lo que respecta a la personalización de marca).
- Donde los metadatos del tema definen la marca propia: false, use el paquete de personalización de marca asociado con el valor de metadatos TopicVendor.
- Donde los metadatos del tema definen el nombre< "Microsoft.Help.TopicVendor" nombre del paquete de personalización de marca de contenido en el proveedor MSHA>, use el paquete de personalización de marca definido en el valor de contenido.

- Dentro del catálogo de Visual Studio, hay una aplicación de prioridad de paquetes de personalización de marca. Primero se aplica la personalización de marca predeterminada de Visual Studio y, a continuación, si se define en los metadatos del tema y se admite con el paquete de personalización de marca asociado (como se define en la msha de instalación), la personalización de marca definida por el proveedor se aplica como invalidación.

Los elementos de marca suelen dividirse en tres categorías principales:

- Elementos de encabezado (por ejemplo, enlace de comentarios, texto de exención de responsabilidad condicional, logotipo)
- Comportamientos de contenido (por ejemplo, elementos de texto de control expand/collapse y elementos de fragmento de código)
- Elementos de pie de página (ejemplo Copyright)

Los elementos considerados como elementos de marca incluyen (detallado en esta especificación):

- Logotipo de catálogo/producto (ejemplo, Visual Studio)
- Enlace de comentarios y elementos de correo electrónico
- Texto de exención de responsabilidad
- Texto de derechos de autor

Los archivos auxiliares del paquete de personalización de marca del Visor de Ayuda de Visual Studio incluyen:

- Gráficos (logos, iconos, etc.)
- Branding.js - archivos de script que admiten comportamientos de contenido
- Branding.xml: cadenas que se usan de forma coherente en el contenido del catálogo. Nota: para los elementos de texto de localización <de Visual Studio en el archivo branding.xml, incluya _locID " valor único> "
- Branding.css - definiciones de estilo para la coherencia de la presentación
- Printing.css - definiciones de estilo para una presentación impresa coherente

Como se indicó anteriormente, los paquetes de personalización de marca están asociados con el tema:

- Cuando SelfBranded - false se define en los metadatos, el tema hereda el paquete de personalización de marca del catálogo
- O cuando SelfBranded es false y hay un paquete de personalización de marca único definido en el MSHA y disponible cuando el contenido está instalado

Para vsP que implementan paquetes de personalización de marca personalizados (contenido de VSP, SelfBranded-True), una manera de continuar es comenzar con el paquete de personalización de marca de reserva (instalado con el Visor de ayuda) y cambiar el nombre del archivo según corresponda. La <Branding_ locale>.mshc es un archivo zip con la extensión de archivo cambiada a .mshc, así que simplemente cambie la extensión de .mshc a .zip y extraiga el contenido. Consulte a continuación los elementos del paquete de personalización de marca y modifíquelo según corresponda (por ejemplo, cambie el logotipo al logotipo de VSP y la referencia al logotipo en el archivo Branding.xml, actualice Branding.xml por detalles de VSP, etc.).

Cuando se realicen todas las modificaciones, cree un archivo zip que contenga los elementos de personalización de marca deseados y cambie la extensión a .mshc.

Para asociar el paquete de personalización de marca personalizado, cree MSHA, que contiene la referencia al archivo mshc de personalización de marca junto con el contenido mshc (que contiene los temas). Vea a

continuación "MSHA" para saber cómo crear un MSHA básico.

El archivo Branding.xml contiene una lista de elementos que se usan <para representar de forma coherente elementos específicos de un tema cuando el tema contiene el nombre meta"Microsoft.Help.SelfBranded" content"false"/>. La lista de elementos de Visual Studio en el archivo Branding.xml se muestra a continuación. Esta lista está diseñada para usarse como plantilla para los adoptantes de ISO Shell, donde modifican estos elementos (por ejemplo, logotipo, comentarios y derechos de autor) para satisfacer sus propias necesidades de marca de productos.

Nota: las variables que se indican por la palabra "-n" tienen dependencias de código - quitar o cambiar estos valores causará errores y posiblemente bloqueo de la aplicación. Los identificadores de localización (ejemplo_locID"codesnippet.n") se incluyen en el paquete de personalización de marca de Visual Studio.

Branding.xml

Característica:	CollapsibleArea
Uso:	Expandir contrae el texto del control de contenido
Elemento	Value
ExpandText	Expanda
CollapseText	Contraer
Característica:	CodeSnippet
Uso:	Texto de control de fragmento de código. Nota: El contenido del fragmento de código con el espacio "No rompedor" se cambiará al espacio.
Elemento	Value
CopyToClipboard	Copiar al Portapapeles
ViewColorizedText	Ver coloreado
CombinedVbTabDisplayLanguage	Visual Basic (ejemplo)
VbDeclaration	Declaración
VbUsage	Uso
Característica:	Comentarios, pie de página y logotipo
Uso:	Proporcione un control de comentarios para que el cliente proporcione comentarios sobre el tema actual a través de correo electrónico. Texto de copyright para el contenido. Definición del logotipo.
Elemento	Valor (estas cadenas se pueden modificar para satisfacer las necesidades del adoptante de contenido.)

Copyright	© 2013 Microsoft Corporation. Todos los derechos reservados.
SendFeedback	<un href="{0} {1} "><Enviar comentarios /a> sobre este tema a Microsoft.
FeedbackLink	
LogoTitle	Visual Studio 2013
LogoFileName	vs_logo_bk.gif
LogoFileNameHC	vs_logo_wh.gif
Característica:	Renuncia
Uso:	Conjunto de descargos de responsabilidad específicos para contenido traducido por máquina.
Elemento	Value
MT_Editable	Este artículo fue traducido a máquina. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
MT_NonEditable	Este artículo fue traducido a máquina. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
MT_QualityEditable	Este artículo fue traducido manualmente. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
MT_QualityNonEditable	Este artículo fue traducido manualmente. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
MT_BetaContents	Este artículo fue traducido por máquina para una versión preliminar. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
MT_BetaRecycledContents	Este artículo fue traducido manualmente para una versión preliminar. Si tiene una conexión a Internet, seleccione "Ver este tema en línea" para ver esta página en modo editable con el contenido original en inglés al mismo tiempo.
Característica:	LinkTable
Uso:	Soporte para enlaces de temas en línea

Elemento	Value
LinkTableTitle	Tabla de enlaces
TopicEnuLinkText	Vea la<versión en inglés /a> de este tema que está disponible en el equipo.
TopicOnlineLinkText	Vea este <tema un{0}href" ">{1} en línea
OnlineText	En línea
Característica:	Control de audio de vídeo
Uso:	Mostrar elementos y texto para contenido de vídeo
Elemento	Value
MultiMediaNotSupported	Internet Explorer 9 o superior {0} debe estar instalado para admitir contenido.
Videotext	mostrar vídeo
AudioText	transmisión de audio
OnlineVideoLinkText	Para ver el vídeo {0} <asociado a este{1}tema, haga clic en un href" ">{2}aquí.
OnlineAudioLinkText	Para escuchar el audio asociado {0} <a este{1}tema, {2} <haga clic en un href" ">aquí /a>.
Característica:	Control de contenido no instalado
Uso:	Elementos de texto (cadenas) utilizados para la representación de contentnotinstalled.htm
Elemento	Value
ContentNotInstalledTitle	No se encontró contenido en el equipo.
ContentNotInstalledDownloadContentText	Para descargar contenido <en su{0}ordenador, un href" ">{1} haga clic en la pestaña<Administrar /a>.
ContentNotInstalledText	No hay contenido instalado en el equipo. Consulte el administrador para obtener información sobre la instalación de contenido de la Ayuda local.
Característica:	Tema no encontrado control
Uso:	Elementos de texto (cadenas) utilizados para la representación de topicnotfound.htm
Elemento	Value

TopicNotFoundTitle	No se puede encontrar el tema solicitado en el equipo.
TopicNotFoundViewOnlineText	El tema que solicitó no se <encontró en{0} {1} su equipo, <pero puede un href " ">ver el tema en línea /a>.
TopicNotFoundDownloadContentText	Consulte el panel de navegación <para obtener{0}vínculos a temas similares, o un href" ">{1} haga clic en la pestaña<Administrar /a> para descargar contenido en el equipo.
TopicNotFoundText	El tema que solicitó no se encontró en su equipo.
Característica:	Tema Control dañado
Uso:	Elementos de texto (cadenas) utilizados para la representación de topiccorrupted.htm
Elemento	Value
TopicCorruptedTitle	No se puede mostrar el tema solicitado.
TopicCorruptedViewOnlineText	Visor de Ayuda no puede mostrar el tema solicitado. Puede haber un error en el contenido del tema o una dependencia del sistema subyacente.
Característica:	Control de la página de inicio
Uso:	Texto que admite la visualización del contenido del nodo de nivel superior del Visor de ayuda.
Elemento	Value
HomePageTitle	Ayuda al espectador a inicio
HomePageIntroduction	Bienvenido al Visor de Ayuda de Microsoft, una fuente esencial de información para todos los usuarios de herramientas, productos, tecnologías y servicios de Microsoft. El Visor de ayuda le da acceso a información de cómo hacer y hacer referencia, código de ejemplo, artículos técnicos y mucho más. Para encontrar el contenido que necesita, examine la tabla de contenido, use la búsqueda de texto completo o navegue por el contenido mediante el índice de palabras clave.
HomePageContentInstallText	<p <>br /><Utilice a{0} {1} href"">Administrar<contenido</una pestaña> para hacer lo siguiente: ul> - Agregar contenido a su equipo. <>> Compruebe si hay actualizaciones de su contenido local. <>> Eliminar contenido de su ordenador. <>/ul> </p>
HomePageInstalledBooks	Libros instalados
HomePageNoBooksInstalado	No se encontró contenido en el equipo.

HomePageHelpSettings	Configuración del contenido de la ayuda
HomePageHelpSettingsText	<p>Su configuración actual es ayuda local. El Visor de Ayuda muestra el contenido que ha instalado en el equipo.
Para cambiar el origen del contenido de <la Ayuda,{0}>en la barra<de menús de Visual Studio, elija span style=" ">Help, Set Help Preference /span>.
</p>
Megabyte	MB

branding.js

El archivo branding.js contiene JavaScript utilizado por los elementos de personalización de marca del Visor de Ayuda de Visual Studio. A continuación se muestra una lista de los elementos de personalización de marca y la función JavaScript compatible. Todas las cadenas que se van a localizar para este archivo se definen en la sección "Cadenas localizables" en la parte superior de este archivo. Se ha creado un archivo ICL para cadenas loc dentro del archivo branding.js.

Característica de marca	Función JavaScript	Descripción
Var...		Definición de variables
Obtener el idioma del código de usuario	setUserPreferenceLang	asigna un índice a lenguaje de código
Establecer y obtener valores de cookies	getCookie, setCookie	
Miembro heredado	changeMembersLabel	Expandir/contraer miembro heredado
Cuando SelfBranded-False	Onload	Lea la cadena de consulta para comprobar si se trata de una solicitud de impresión. Establezca todos los códigos para enfocar la pestaña preferida por el usuario. Si se trata de una solicitud de impresión, establezca isPrinterFriendly en true. Compruebe si hay modo de alto contraste.
Fragmento de código	addSpecificTextLanguageTagSet	
	getIndexFromDevLang	
	ChangeTab	
	setCodesnippetLang	
	setCurrentLang	
	CopyToClipboard	
CollapsibleArea	addToCollapsibleControlSet	escribir todo el objeto de control contraíble en la lista.

	CA_Click	en función del estado del área contraíble, define qué imagen y texto presentar
Soporte de contraste para logo	isBlackBackground()	Se llama para determinar si el fondo es negro. Sólo es preciso cuando está en modo de alto contraste.
	isHighContrast()	utilizar un palmo de color para detectar el modo de alto contraste
	onHighContrast(negro)	Se llama cuando se detecta un alto contraste
Funcionalidad LST		
	addToLanSpecTextIdSet(id)	
	updateLST(currentLang)	
	getDevLangFromCodeSnippet(lang)	
Funcionalidad MultiMedia	subtítulo (comienzo, fin, texto, estilo)	
	findAllMediaControls(normalizedId)	
	getActivePlayer(normalizedId)	
	captionsOnOff(id)	
	toSeconds(t)	
	getAllComments(node)	
	styleRectify(styleName, styleValue)	
	showCC(id)	
	subtítulo (id)	

ARCHIVOS HTM

El paquete de personalización de marca contiene un conjunto de archivos HTM que admiten escenarios para comunicar información clave a los usuarios de contenido de Ayuda, por ejemplo, una página de inicio que contiene una sección que describe qué conjuntos de contenido están instalados y páginas que le dicen al usuario cuándo no se pueden encontrar temas en el conjunto local de temas. Estos archivos HTM se pueden modificar por producto. Los proveedores de ISO Shell pueden tomar el paquete de personalización de marca predeterminado y cambiar el comportamiento y el contenido de estas páginas para que se ajusten a sus necesidades. Estos archivos hacen referencia a su respectivo paquete de personalización de marca para que las etiquetas de marca obtengan el contenido correspondiente del archivo branding.xml.

Archivo	Uso	Fuente de contenido mostrada
homepage.htm	Esta es una página que muestra el contenido instalado actualmente y cualquier otro mensaje apropiado para presentar al usuario acerca de su contenido. Este archivo tiene el atributo de metadatos adicional "Microsoft.Help.Id" content"-1" que coloca este contenido en la parte superior del contenido local TOC.	
	<META_HOME_PAGE_TITLE_ADD />	Branding.xml, <etiqueta HomePageTitle>
	<HOME_PAGE_INTRODUCTION_SECTION_ADD />	Branding.xml, <etiqueta HomePageIntroduction>
	<HOME_PAGE_CONTENT_INSTALL_SECTION_ADD />	Branding.xml, <etiqueta HomePageContentInstallText>
	<HOME_PAGE_BOOKS_INSTALLED_SECTION_ADD />	Encabezado sección Branding.xml etiqueta<HomePageInstalledBooks>, los <datos generados desde la aplicación, HomePageNoBooksInstalled> cuando no hay libros instalados.
	<HOME_PAGE_SETTINGS_SECTION_ADD />	Sección de encabezado <Branding.xml etiqueta HomePageHelpSettings>, texto <de sección HomePageHelpSettingsText>.
topiccorrupted.htm	Cuando existe un tema en el conjunto local, pero por alguna razón no se puede mostrar (contenido dañado).	
	<META_TOPIC_CORRUPTED_TITLE_ADD />	Branding.xml, <etiqueta TopicCorruptedTitle>
	TOPIC_CORRUPTED_SECTION_ADD />	Branding.xml, <etiqueta TopicCorruptedViewOnlineText>
topicnotfound.htm	Cuando no se encuentra un tema en el conjunto de contenido local, ni está disponible en línea	
	<META_TOPIC_NOT_FOUND_TITLE_ADD />	Branding.xml, <etiqueta TopicNotFoundTitle>
	<META_TOPIC_NOT_FOUND_ID_ADD />	Branding.xml, <etiqueta TopicNotFoundViewOnlineText <> + TopicNotFoundDownloadContentText>
	<TOPIC_NOT_FOUND_SECTION_ADD />	Branding.xml, <etiqueta TopicNotFoundText>

contentnotinstalled.htm	Cuando no hay ningún contenido local instalado para el producto.	
	<META_CONTENT_NOT_INSTALLED_TITLE_ADD />	Branding.xml, <etiqueta ContentNotInstalledTitle>
	<META_CONTENT_NOT_INSTALLED_ID_ADD />	Branding.xml, <etiqueta ContentNotInstalledDownloadContentText>
	CONTENT_NOT_INSTALLED_SECTION_ADD de </>	Branding.xml, <etiqueta ContentNotInstalledText>

Archivos CSS

El paquete de personalización de marca del Visor de Ayuda de Visual Studio contiene dos archivos css para admitir la presentación de contenido coherente de la Ayuda de Visual Studio:

- Branding.css - contiene elementos css para la representación donde SelfBranded=false
- Printer.css - contiene elementos css para la representación donde SelfBranded=false

Los archivos Branding.css incluyen definiciones para la presentación de temas de Visual Studio (caveat es que el archivo branding.css contenido en la configuración regional de Branding_>.mshc del servicio de paquete puede cambiar).

Archivos gráficos

El contenido de Visual Studio muestra un logotipo de Visual Studio, así como otros gráficos. La lista completa de archivos gráficos en el paquete de personalización de marca visor de Ayuda de Visual Studio se muestra a continuación.

Archivo	Uso	Ejemplos
clear.gif	Se utiliza para renderizar el área plegable	
footer_slice.gif	Presentación del pie de página	
info_icon.gif	Se utiliza al mostrar información	Renuncia de responsabilidades
online_icon.gif	Este icono debe asociarse con enlaces en línea	
tabLeftBD.gif	Se utiliza para representar el contenedor de fragmentos de código	
tabRightBD.gif	Se utiliza para representar el contenedor de fragmentos de código	

vs_logo_bk.gif	Se utiliza para las referencias de logotipo <de contraste normal tal como se define en la etiqueta Branding.xml LogoFileName> . Para los productos de Visual Studio, el nombre del logotipo es vs_logo_bk.gif.	
vs_logo_wh.gif	Se utiliza para referencias de logotipos <altos normales tal como se define en la etiqueta Branding.xml LogoFileNameHC> . Para los productos de Visual Studio, el nombre del logotipo es vs_logo_wh.gif.	
ccOff.png	Gráfico de subtítulos	
ccOn.png	Gráfico de subtítulos	
ImageSprite.png	Se utiliza para renderizar el área plegable	expandido o colapso gráfico

Implementar un conjunto de temas

Este es un tutorial sencillo y rápido para crear un conjunto de implementación de contenido del Visor de Ayuda compuesto por un archivo MSHA y el conjunto de cabinas o MSHCs que contienen los temas. El MSHA es un archivo XML que describe un conjunto de cabs o archivos MSHC. El Visor de Ayuda puede leer el MSHA para obtener una lista de contenido (el archivo . CAB o . MSHC) disponibles para la instalación local.

Esto es sólo una imprimación que describe el esquema XML muy básico para el Visor de ayuda MSHA. Hay un ejemplo de implementación debajo de esta breve descripción general y ejemplo HelpContentSetup.msha.

El nombre de la MSHA, a los efectos de este primer, es HelpContentSetup.msha (el nombre del archivo puede ser cualquier cosa, con la extensión . MSHA). HelpContentSetup.msha (ejemplo a continuación) debe contener una lista de las cabinas o MSHC disponibles. El tipo de archivo debe ser coherente dentro de MSHA (no admite una combinación de tipos de archivo MSHA y CAB). Para cada CAB o MSHC, <debe haber una clase div "paquete">... </div> (véase el ejemplo a continuación).

Nota: en el ejemplo de implementación siguiente, hemos incluido el paquete de personalización de marca. Esto es fundamental para incluir para obtener los elementos de representación de contenido de Visual Studio y los comportamientos de contenido necesarios.

Ejemplo de archivo HelpContentSetup.msha: (Reemplazar "nombre de conjunto de contenido 1" y "nombre de conjunto de contenido 2", etc. por sus nombres de archivo.)

```

<html>
<head />
<body class="vendor-book">
<div class="details">
<span class="vendor">Your Company</span>
<span class="locale">en-us</span>
<span class="product">Your Company Help Content</span>
<span class="name">Your Company Help Content</span>
</div>
<div class="package-list">
<div class="package">
<span class="name">Your_Company _Content_Set_1</span>
<span class="deployed">True</span>
<a class="current-link" href="Your_Company _Content_Set_1.mshc">Your_Company _Content_Set_1.mshc </a>
</div>
<div class="package">
<span class="name">Your_Company _Content_Set_2</span>
<span class="deployed">True</span>
<a class="current-link" href=" Your_Company _Content_Set_2.mshc " > Your_Company _Content_Set_2.mshc </a>
</div>

```

1. Crear carpeta local, algo así como "C:-SampleContent"
2. En este ejemplo, usaremos archivos MSHC para contener los temas. Un MSHC es un zip con la extensión de archivo cambiado de .zip a .MSHC.
3. Cree el siguiente HelpContentSetup.msha como un archivo de texto (el bloc de notas se utilizó para crear el archivo) y guárdelo en la carpeta indicada anteriormente (consulte el paso 1).

La clase "Branding" existe y es única. El mshc de branding se incluye en este primer para que el contenido instalado tenga personalización de marca, y los comportamientos de contenido contenidos en los MSHC tendrán los elementos de soporte adecuados contenidos en el paquete de personalización de marca. Sin esto, se producirán errores cuando el sistema busque elementos de soporte que no formen parte del contenido rasgado (instalado).

Para obtener el paquete de personalización de marca de Visual Studio, copie el archivo Branding_en-US.mshc en c:-Archivos de programa (x86)-Microsoft Help Viewer-v2.3- en la carpeta de trabajo.

```

<html>
<head />
<body class="vendor-book">
<div class="details">
<span class="vendor">Your Company</span>
<span class="locale">en-us</span>
<span class="product">Your Company Help Content</span>
<span class="name">Your Company Help Content</span>
</div>
<div class="package-list">
<div class="package">
<span class="name">Your_Company _Content_Set_1</span>
<span class="deployed">True</span>
<a class="current-link" href="Your_Company _Content_Set_1.mshc">Your_Company _Content_Set_1.mshc </a>
</div>
<div class="package">
<span class="name">Your_Company _Content_Set_2</span>
<span class="deployed">True</span>
<a class="current-link" href=" Your_Company _Content_Set_2.mshc " > Your_Company _Content_Set_2.mshc </a>
</div>
<div class="package">
<span class="packageType">branding</span>
<span class="name">Branding_en-US</span>
<span class="deployed">True</span>
<a class="current-link" href="Branding_en-US.mshc">Branding_en-US.mshc</a>
</div>
</div>
</body>
</html>

```

Resumen

Usar y ampliar los pasos anteriores permitirá a los VSP implementar sus conjuntos de contenido para el Visor de Ayuda de Visual Studio.

Agregar ayuda al Shell de Visual Studio (integrado y aislado)

Introducción

En este tutorial se muestra cómo incorporar contenido de Ayuda en una aplicación de Shell de Visual Studio y, a continuación, implementarlo.

Requisitos

1. Visual Studio 2013
2. [Visual Studio 2013 Aislado Shell Redist](#)

Información general

El Visual Studio 2013 Shell es una Visual Studio 2013 versión del IDE en la que puede basar una aplicación. Estas aplicaciones contienen el Shell aislado junto con las extensiones que cree. Use plantillas de proyecto de Shell Visual Studio 2013 aislado, que se incluyen en el SDK, para crear extensiones.

Los pasos básicos para crear una aplicación basada en Shell aislado y su Ayuda:

1. Obtenga Visual Studio 2013 el Shell ISO redistribuible (una descarga de Microsoft).
2. En Visual Studio, cree una extensión de Ayuda basada en el Shell aislado, por ejemplo, la extensión de Ayuda de Contoso que se describe más adelante en este tutorial.
3. Envuelva la extensión y el Shell ISO redistribuible en un MSI de implementación (una configuración de aplicación). Este tutorial no incluye un paso de configuración.

Cree un almacén de contenido de Visual Studio. Para el escenario Shell integrado, cambie Visual Studio12 por el nombre del catálogo de productos de la siguiente manera:

- Cree la carpeta C:-ProgramData-Microsoft-HelpLibrary2-Catalogs-VisualStudio15.
- Cree un archivo denominado CatalogType.xml y agréguelo a la carpeta. El archivo debe contener las siguientes líneas de código:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogType>UserManaged</catalogType>
```

Defina el almacén de contenido en el registro. Para el Shell integrado, cambie VisualStudio15 al nombre del catálogo de productos:

- HKLM-SOFTWARE-Wow6432Node-Microsoft-Help-v2.3-Catalogs-VisualStudio15
Clave: LocationPath Valor de cadena: C:-ProgramData-Microsoft-HelpLibrary2-Catalogs-VisualStudio15-
- HKLM-SOFTWARE-Wow6432Node-Microsoft-Help-v2.3-Catalogs-VisualStudio15-en-US
Clave: Valor de Visual Studio 2013 cadena CatalogName: Documentación

Crear el proyecto

Para crear una extensión de vaciado aislado:

1. En Visual Studio, en **Archivo**, elija **Nuevo proyecto**, en Otros tipos de **proyecto**, **Extensibilidad**, a continuación, elija **Visual Studio Shell Isolated**. Asigne al `ContosoHelpShell` proyecto el nombre) para crear un proyecto de extensibilidad basado en la plantilla Shell aislado de Visual Studio.
2. En el Explorador de soluciones, en el proyecto ContosoHelpShellUI, en la carpeta Archivos de recursos, abra ApplicationCommands.vsct. Asegúrese de que esta línea está comentada (busque "No_Help"):

```
<!-- <define name="No_HelpMenuCommands"/> -->
```
3. Elija la clave F5 para compilar y ejecutar **Debug**. En la instancia experimental del IDE de Shell aislado, elija el menú **Ayuda**. Asegúrese de que aparecen los comandos **Ver ayuda**, **Agregar y quitar contenido de ayuda** y **Establecer preferencia de ayuda**.
4. En el Explorador de soluciones, en el proyecto ContosoHelpShell, en la carpeta Personalización de Shell, abra ContosoHelpShell.pkgdef. Para definir el catálogo de Ayuda de Contoso, agregue las siguientes líneas:

```
[$RootKey$\Help]
"Product"="Contoso"
"Catalog"="Contoso"
"Version"="100"
"BrandingPackage"="ContosoBrandingPackage.mshc"
```

5. En el Explorador de soluciones, en el proyecto ContosoHelpShell, en la carpeta Personalización de Shell, abra ContosoHelpShell.Application.pkgdef. Para habilitar la Ayuda de F1, agregue las siguientes líneas:

```
// F1 Help Provider

[$RootKey$\HelpProviders\{C99BDC23-FF29-46bf-9658-ADD634CCAED8}]
"Name"="13407"
"Package"="{DA9FB551-C724-11d0-AE1F-00A0C90FFFC3}"
@="Help3 Provider"
[$RootKey$\HelpProviders]
@="{C99BDC23-FF29-46bf-9658-ADD634CCAED8}"
[$RootKey$\Services\{C99BDC23-FF29-46bf-9658-ADD634CCAED8}]
"Name"="Help3 Provider"
@="{4A791146-19E4-11D3-B86B-00C04F79F802}"
```

- En el Explorador de soluciones, en el menú contextual de la solución ContosoHelpShell, elija el elemento de menú **Propiedades**. En **Propiedades de configuración**, seleccione **Configuration Manager**. En la columna **Configuración**, cambie cada valor de "Depurar" a "Liberar".
- Compile la solución. Esto crea un conjunto de archivos en una carpeta de versión, que se usará en la siguiente sección.

Para probar esto como si estuviera implementado:

- En el equipo en el que va a implementar Contoso, instale el Shell ISO descargado (desde arriba).
- Cree una carpeta en .Archivos de programa (x86) y asímóquela. Contoso
- Copie el contenido de la carpeta de la versión ContosoHelpShell en la carpeta .archivos de programa (x86)-Contoso.
- Inicie el Editor del Registro seleccionando Regedit **Ejecutar** en el menú **Inicio** e introduciendo . En el editor del Registro, elija **Archivos**, a continuación, **Importar**. Vaya a la carpeta del proyecto ContosoHelpShell. En la subcarpeta ContosoHelpShell, elija ContosoHelpShell.reg.
- Crear un almacén de contenido:

Para Shell ISO: cree un almacén de contenido de Contoso C:-ProgramData-Microsoft-HelpLibrary2-Catalogs-ContosoDev12

Para Visual Studio 2013 el Shell integrado, cree la carpeta C:-ProgramData-Microsoft-HelpLibrary2-Catalogs-VisualStudio15

- Cree CatalogType.xml y agréguelo al almacén de contenido (paso anterior) que contiene:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogType>UserManaged</catalogType>
```

- Agree las siguientes claves del Registro:

HKLM-SOFTWARE-Wow6432Node-Microsoft-Help-v2.3-Catalogs-VisualStudio15Key: LocationPath Valor de cadena:

Para ISO Shell:

C:\ProgramData\Microsoft\HelpLibrary2\Catalogs\VisualStudio15

Visual Studio 2013 Shell integrado:

C:\ProgramData\Microsoft\HelpLibrary2\Catalogs\VisualStudio15en-US

Clave: Valor de Visual Studio 2013 cadena CatalogName: Documentación. Para ISO Shell, este es el nombre del catálogo.

8. Copie el contenido (cabs o MSHC y MSHA) en una carpeta local.

9. Ejemplo de línea de comandos de Shell integrado para probar el almacén de contenido. Para ISO Shell, cambie el catálogo y lanzarlos los valores según corresponda para que coincidan con el producto.

```
"C:-Archivos de programa (x86)-Microsoft Help Viewer-v2.3-HlpViewer.exe" /catalogName VisualStudio15 /helpQuery method-"page&id-ContosoTopic0" /launchingApp Microsoft,VisualStudio,12.0
```

10. Inicie la aplicación Contoso (desde la raíz de la aplicación Contoso). En ISO Shell, elija el elemento de menú **Ayuda** y cambie **Establecer preferencia** de ayuda para usar la **Ayuda local**.

11. En el shell, elija el elemento de menú **Ayuda** y, a continuación, **Ver ayuda**. El visor de Ayuda local debe iniciarse. Elija la pestaña **Manage Content (Administrar contenido)**. En **Origen de instalación**, elija el botón de opción **Disco**. Elija el botón ... y vaya a la carpeta local que contiene el contenido de Contoso (copiado en la carpeta local en el paso anterior). Elija HelpContentSetup.msha. Contoso ahora debe aparecer como un libro en las selecciones de libros. Elija **Addy**, a continuación, elija el botón **Update** (esquina inferior derecha).

12. En el IDE de Contoso, elija la tecla F1 para probar la funcionalidad de F1.

Recursos adicionales

Para la API de tiempo de ejecución, consulte API de ayuda de [Windows](#).

Para obtener más información sobre cómo aprovechar la API de ayuda, consulte Ejemplos de código del Visor de [ayuda](#).

Puede enviar sugerencias de características en [la Comunidad de desarrolladores](#).

Soporte técnico para Visual Studio SDK

06/04/2020 • 4 minutes to read • [Edit Online](#)

Hay varias opciones de soporte técnico disponibles para los clientes que tienen preguntas sobre cómo ampliar el entorno de desarrollo integrado (IDE) de Visual Studio.

Soporte gratuito

Grupos de noticias de Microsoft

Los miembros del equipo de desarrollo de Visual Studio y otros desarrolladores que amplían Visual Studio suelen participar en el grupo de noticias [microsoft.public.vstudio.extensibility](#). Este grupo de noticias es un grupo de noticias administrado. Si es un suscriptor universal de MSDN, asegúrese de registrar un alias sin spam para que Microsoft pueda responder.

Foros de MSDN

Los [foros](#) de MSDN proporcionan otra forma de chatear con compañeros y expertos. La interfaz gráfica de usuario (UI) de los foros está diseñada para que sean más fáciles de usar que los grupos de noticias. El foro de [extensibilidad](#) de Visual Studio es un buen lugar para empezar a buscar respuestas a preguntas de extensibilidad. Siempre busque antes de hacer una pregunta porque otra persona puede haber hecho la misma pregunta.

Galería de Visual Studio

[Visual Studio Marketplace](#) es el lugar para las extensiones y el código de ejemplo. Puede encontrar herramientas que mejoran la productividad y la experiencia del usuario con Visual Studio, aplicaciones de ejemplo que muestran cómo usar VSPackages y Managed Extensibility Framework (MEF) y otros tipos de ejemplos.

Ejemplos de SDK de Visual Studio

Puede encontrar ejemplos de código abierto de extensiones de SDK de VS en GitHub en [Visual Studio Samples](#). Este repositorio de GitHub contiene ejemplos que ilustran varias características extensibles en Visual Studio.

Soporte incluido

Soporte técnico de Visual Studio

Parte del soporte de Visual Studio incluye con la compra de productos. Para obtener más información, vea [Compatibilidad con Visual Studio](#). Microsoft mantiene un equipo de soporte Visual Studio técnico dedicado para responder preguntas sobre la extensibilidad.

Soporte de suscripción a MSDN

Los paquetes de suscripción de MSDN también pueden incluir cierta compatibilidad. Para obtener más información, consulte [Compatibilidad con suscriptores](#) de MSDN.

Soporte técnico de pago

Soporte de herramientas para desarrolladores

El soporte también se puede comprar. Una variedad de ofertas de soporte técnico están disponibles en la categoría Herramientas de desarrollo en los Centros de ayuda y soporte técnico de [Microsoft](#).

Los clientes que Visual Studio SDK deseen usar para desarrollar paquetes complejos pueden considerar la compra de un contrato de soporte técnico Premier con Microsoft. Para obtener más información, consulte [Soporte técnico de Microsoft Premier](#). Para las pequeñas empresas, [el soporte avanzado para socios](#) puede ser una alternativa atractiva a una oferta tradicional de Premier Support.

NOTE

El soporte estándar no se puede utilizar para la consulta o para obtener asesoramiento.

Las preguntas sobre las ofertas de soporte Premier junto vsipsvs@microsoft.com con el programa VSIP se pueden enviar a . Las preguntas sobre el nuevo programa advantg@microsoft.com Partner Advantage se pueden enviar a .

Puede encontrar información sobre las solicitudes de soporte individuales en Solicitudes de [soporte individuales](#).

Vea también

- [Hable con nosotros](#)

Referencia del SDK de Visual Studio

06/04/2020 • 7 minutes to read • [Edit Online](#)

Esta sección contiene información acerca de los espacios de nombres de Visual Studio, espacios de nombres relacionados y otras áreas de interés para los desarrolladores que usan el SDK de Visual Studio.

En esta sección

- [Microsoft.VisualStudio.TextManager.Interop](#) Interfaces de interoperabilidad heredadas utilizadas para el editor y los servicios de lenguaje.
- [Microsoft.VisualStudio.Editor](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Language.Intellisense](#) Clases utilizadas en el nuevo editor para IntelliSense.
- [Microsoft.VisualStudio.Language.StandardClassification](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Text](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Text.Adornments](#) Clases utilizadas en el nuevo editor para adornos.
- [Microsoft.VisualStudio.Text.Classification](#) Clases utilizadas en el nuevo editor para la clasificación.
- [Microsoft.VisualStudio.Text.Differencing](#) Clases utilizadas en el nuevo editor para la diferenciación.
- [Microsoft.VisualStudio.Text.Document](#) Clases utilizadas en el nuevo editor para documentos.
- [Microsoft.VisualStudio.Text.Editor](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Text.Editor.DragDrop](#) Clases utilizadas en el nuevo editor para arrastrar y soltar.
- [Microsoft.VisualStudio.Text.Editor.OptionsExtensionMethods](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Text.Formatting](#) Clases utilizadas en el nuevo editor para dar formato.
- [Microsoft.VisualStudio.Text.IncrementalSearch](#) Clases utilizadas en el nuevo editor para la búsqueda incremental.
- [Microsoft.VisualStudio.Text.Operations](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.Text.Outlining](#) Clases utilizadas en el nuevo editor para esquematización.
- [Microsoft.VisualStudio.Text.Projection](#) Clases utilizadas en el nuevo editor para la proyección.
- [Microsoft.VisualStudio.Text.Tagging](#) Clases utilizadas en el nuevo editor para el etiquetado.
- [Microsoft.VisualStudio.Utilities](#) Clases utilizadas en el nuevo editor.
- [Microsoft.VisualStudio.PlatformUI](#)
- [Microsoft.VisualStudio.Shell.Interop](#)
- [Microsoft.VisualStudio](#) Clases para constantes y ayudantes de Visual Studio.
- [Microsoft.VisualStudio.CommandBars](#) Clases para barras de comandos de Visual Studio.
- [Microsoft.VisualStudio.ComponentModelHost](#) Clases utilizadas para Managed Extensibility Framework (MEF) en Visual Studio.

- [Microsoft.VisualStudio.Designer.Interfaces](#) Interfaces utilizadas con diseñadores de Visual Studio.
- [Microsoft.VisualStudio.ManagedInterfaces.ProjectDesigner](#) Interfaces utilizadas con el diseñador de proyectos de Visual Studio.
- [Microsoft.VisualStudio.ManagedInterfaces.Publish](#) Clases utilizadas para publicar aplicaciones.
- [Microsoft.VisualStudio.OLE.Interop](#) Interfaces de interoperabilidad utilizadas en Visual Studio para componentes OLE.
- [Microsoft.VisualStudio.Package](#) Clases utilizadas para los servicios de lenguaje administrado de Visual Studio.
- [Microsoft.VisualStudio.PlatformUI](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.PlatformUI.OleComponentSupport](#) Solo para uso interno de Microsoft.
- [Microsoft.VisualStudio.ProjectAggregator](#) Clases utilizadas para proyectos de Visual Studio.
- [Microsoft.VisualStudio.Settings](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell.Design](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell.Design.Serialization](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell.Design.Serialization.CodeDom](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell.Flavor](#) Clases utilizadas para los proyectos de Visual Studio.
- [Microsoft.VisualStudio.Shell.Interop](#) Interfaces de interoperabilidad utilizadas para el shell de Visual Studio.
- [Microsoft.VisualStudio.Shell.Settings](#) Clases utilizadas en el shell de Visual Studio.
- [Microsoft.VisualStudio.VSHelp](#) Clases utilizadas para ayuda.
- [VSLangProj](#) Clases utilizadas para proyectos de servicio de lenguaje.
- [XamlGeneratedNamespace](#) Solo para uso interno de Microsoft.
- [Microsoft.VisualStudio.ManagedInterfaces9](#) Interfaces utilizadas para Visual Studio.
- [Microsoft.VisualStudio.WCFReference.Interop](#) Clases utilizadas para Windows Communication Framework.
- [Microsoft.VisualStudio.TestTools.Vsip](#) Clases utilizadas para herramientas de prueba.
- [EnvDTE](#) Se utiliza para la automatización de Visual Studio.
- [Extensibility](#) Se utiliza para la automatización de Visual Studio.
- [EnvDTE80](#) Se utiliza para la automatización de Visual Studio.
- [EnvDTE90](#) Se utiliza para la automatización de Visual Studio.
- [EnvDTE90a](#) Se utiliza para la automatización de Visual Studio.
- [EnvDTE100](#) Se utiliza para la automatización de Visual Studio.
- [Microsoft.VisualStudio.VCCodeModel](#) Se utiliza para la automatización de proyectos de Visual C++.
- [Microsoft.VisualStudio.VCProject](#) Clases utilizadas para la automatización de proyectos de Visual C++.
- [Microsoft.VisualStudio.VCProjectEngine](#) Se utiliza para la automatización de proyectos de Visual C++.
- [Microsoft.VisualStudio.VsWizard](#) Se utiliza para el Asistente para Visual Studio.

- [VSLangProj](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VSLangProj2](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VSLangProj80](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VslangProj90](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VslangProj100](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VSLangProj110](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [VSLangProj140](#) Se utiliza para la automatización de proyectos de Visual Basic y de C.
- [Microsoft.VisualStudio.Language.CallHierarchy](#) Se utiliza para la automatización de Visual Basic y de C.
- [Microsoft.VisualStudio.Language.NavigateTo.Interfaces](#) Se utiliza para la automatización de Visual Basic y de C.
- [Microsoft.VisualStudio.Threading](#) Se usa para el subprocesamiento de Visual Studio.
- [Espacio de nombres Microsoft.VisualStudio.ConnectedServices](#) Se usa para los servicios conectados de Visual Studio.
- [IntelliSenseHostFlags](#) enumeración Especifica los indicadores de host de IntelliSense.
- [Referencia de esquema XML de VSCT](#) Proporciona una tabla de elementos de esquema de tabla de comandos de Visual Studio, con elementos secundarios y atributos permitidos para cada uno.
- [GUID y constantes](#) Proporciona una lista de GUID utilizados en todo el SDK.
- [Constantes COM en código administrado](#) Proporciona una lista de identificadores para los elementos de interfaz de usuario utilizados en todo el SDK de entorno.
- [Complementos de control de código fuente](#) Vínculos a temas de la API de complemento de Control de código fuente.
- [Interrupciones de línea de comandos](#) Vínculos a temas que describen cómo los desarrolladores pueden automatizar varias tareas desde la línea de comandos.
- [Manejo de errores y valores devueltos](#) Explica la arquitectura de error para VSPackages, que es la misma que la empleada en otra programación de Windows.
- [Objetos](#) Proporciona una lista de los objetos que se utilizan en el entorno.
- [Glosario](#) Proporciona una lista de términos útiles y sus definiciones para su uso al leer la documentación del SDK de Visual Studio.
- [Microsoft.Build.BuildEngine](#) Se utiliza para MSBuild.
- [Microsoft.Build.Construction](#) Se utiliza para MSBuild.
- [Microsoft.Build.Conversion](#) Se utiliza para MSBuild.
- [Microsoft.Build.Debugging](#) Se utiliza para MSBuild.
- [Microsoft.Build.Evaluation](#) Se utiliza para MSBuild.
- [Microsoft.Build.Exceptions](#) Se utiliza para MSBuild.
- [Microsoft.Build.Execution](#) Se utiliza para MSBuild.
- [Microsoft.Build.Framework](#) Se utiliza para MSBuild.

- [Microsoft.Build.Framework.XamlTypes](#)Se utiliza para MSBuild.
- [Microsoft.Build.Logging](#)Se utiliza para MSBuild.
- [Microsoft.Build.Tasks](#)Se utiliza para MSBuild.
- [Microsoft.Build.Tasks.Deployment.Bootstrapper](#)Se utiliza para MSBuild.
- [Microsoft.Build.Tasks.Deployment.ManifestUtilities](#)Se utiliza para MSBuild.
- [Microsoft.Build.Tasks.Hosting](#)Se utiliza para MSBuild.
- [Microsoft.Build.Tasks.Xaml](#)Se utiliza para MSBuild.
- [Microsoft.Build.Utilities](#)Se utiliza para MSBuild.

Secciones relacionadas

[El SDK](#) de Visual Studio incluye documentación, ejemplos y código para ayudarle a desarrollar productos que se integran con Visual Studio.

IntelliSenseHostFlags

06/04/2020 • 2 minutes to read • [Edit Online](#)

Especifica marcas de host de IntelliSense.

Sintaxis

```
enum IntellisenseHostFlags
{
    IHF_READONLYCONTEXT      = 0x00000001
    IHF_NOSEPARATESUBJECT    = 0x00000002
    IHF_SINGLELINESUBJECT    = 0x00000004
    IHF_FORCECOMMITTOCONTEXT = 0x00000008
    IHF_OVERTYPE              = 0x00000010
};
```

Parámetros

MIEMBROS	DESCRIPCIÓN
<code>IHF_READONLYCONTEXT</code>	El búfer de contexto es de solo lectura.
<code>IHF_NOSEPARATESUBJECT</code>	No hay texto de asunto. El búfer de contexto contiene <code>!IHF_READONLYCONTEXT</code> IntelliSense-target (implica).
<code>IHF_SINGLELINESUBJECT</code>	El texto del asunto no es compatible con varias líneas.
<code>IHF_FORCECOMMITTOCONTEXT</code>	Igual a <code>CanCommitIntoReadOnlyBuffer</code> .
<code>IHF_OVERTYPE</code>	La edición (en el asunto o el contexto) debe realizarse en modo de sobrescribir.

Requisitos

SingleFileeditor.idl

Vea también

- [Microsoft.VisualStudio.TextManager.Interop](#)

Referencia de esquema XML de VSCT

06/04/2020 • 4 minutes to read • [Edit Online](#)

Proporciona una tabla de elementos de esquema del compilador de tablas de comandos, con elementos secundarios y atributos permitidos para cada uno.

Un archivo de configuración de tabla de comandos basado en XML (.vsct) define los elementos de comando que un VSPackage proporciona al entorno de desarrollo integrado (IDE). Estos elementos incluyen elementos de menú, menús, barras de herramientas y cuadros combinados.

NOTE

El compilador VSCT puede ejecutar un preprocesador en el archivo .vsct. Dado que normalmente se trata del preprocesador C++, puede definir incluye y macros que tienen la misma sintaxis que se utiliza en los archivos C++. Ejemplos de esto se proporcionan en el archivo .vsct que el Asistente para **nuevo proyecto** crea para un proyecto de VSPackage.

Elementos opcionales

Algunos elementos VSCT son opcionales. Si `Parent` no se especifica un argumento, `Group_Undefined:0` estará implícita. Si `Icon` no se especifica un argumento, se implicará `guidOfficelcon:msotcidNolcon`. Cuando se define una tecla de método abreviado, la emulación, que normalmente no se utiliza, es opcional.

Los elementos de mapa de bits se pueden incrustar en `href` tiempo de compilación especificando la ubicación de la franja de mapa de bits en el argumento. La franja de mapa de bits se copia durante la combinación en lugar de extraerse de los recursos del archivo DLL. Cuando `href` se proporciona un `usedList` argumento, el argumento se convierte en opcional y se consideran todas las ranuras de la franja de mapa de bits utilizadas.

Todos los valores GUID e ID deben definirse mediante nombres simbólicos. Estos nombres se pueden definir en <archivos de encabezado o en símbolos VSCT> secciones. Los nombres simbólicos deben <ser locales, incluidos <a través de Incluir> elementos o a los que haga referencia los elementos de> Externo. Un nombre simbólico se importa desde un <archivo de encabezado especificado en un elemento Extern> si sigue el patrón simple de `#define SYMBOL VALUE`. El valor puede ser otro símbolo siempre y cuando ese símbolo se haya definido previamente. Las definiciones GUID deben seguir el formato OLE o C++. Los valores de ID pueden ser dígitos decimales o hexadecimales precedidos por 0x, como se muestra en las siguientes líneas:

- N.o 6D484634-E53D-4a2c-ADCB-55145C9362C8
- á 0x6d484634, 0xe53d, 0x4a2c, á 0xad, 0xcb, 0x55, 0x14, 0x5c, 0x93, 0x62, 0xc8 á á

Se pueden utilizar comentarios XML, pero las herramientas de interfaz gráfica de usuario (GUI) de ida y vuelta pueden descartarlos. Se garantiza <que el contenido de los elementos de anotación> se mantenga independientemente del formato.

Jerarquía del esquema

Un archivo .vsct tiene los siguientes elementos principales.

- [Elemento CommandTable](#)
- [Elemento Externo](#)

- [Incluir elemento](#)
- [Definir elemento](#)
- [Elemento Comandos](#)
- [Elemento CommandPlacements](#)
- [VisibilityConstraints elemento](#)
- [Elemento KeyBindings](#)
- [Elemento UsedCommands](#)
- [Elemento primario](#)
- [Elemento Icono](#)
- [Elemento Strings](#)
- [Elemento Command Flag](#)
- [Elemento Símbolos](#)
- [Atributos condicionales](#)

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Enrutamiento de comandos en VSPackages](#)

Elemento CommandTable

06/04/2020 • 2 minutes to read • [Edit Online](#)

CommandTable es el elemento raíz del archivo `.vsct`. Este es el archivo que define el diseño real y el tipo de los comandos que un VSPackage proporciona al IDE. Los comandos pueden incluir elementos de menú, menús, barras de herramientas y cuadros combinados. Para obtener más información, vea [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#).

Sintaxis

```
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable"
xmlns:xs="http://www.w3.org/2001/XMLSchema" >
  <Extern>... </Extern>
  <Include>... </Include>
  <Define>... </Define>
  <Commands>... </Commands>
  <CommandPlacements>... </CommandPlacements>
  <VisibilityConstraints>... </VisibilityConstraints>
  <KeyBindings>... </KeyBindings>
  <UsedCommands>... </UsedCommands>
  <Symbols>... </Symbols>
</CommandTable>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
xmlns	Necesario. Espacios de nombres XML: xmlns=http://schemas.microsoft.com/VisualStudio/2005-10-18/CommandTable xmlns:xs="http://www.w3.org/2001/XMLSchema"
language	Opcional. El atributo language se puede utilizar para <especificar el idioma predeterminado de todas las cadenas> elementos de la tabla de comandos. Si no se especifica el idioma, se utilizará el idioma del proceso actual: "en-nosotros"

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Externo	Opcional. Contiene directivas de preprocesador para el compilador.

ELEMENTO	DESCRIPCIÓN
Incluir elemento	Opcional. Contiene rutas de acceso a los archivos que se van a incluir en la compilación.
Definir elemento	Opcional. Define un símbolo dado su nombre y valor.
Elemento Comandos	Opcional. El elemento primario que define todos los comandos para el VSPackage que contiene todos los demás elementos.
Elemento CommandPlacements	Opcional. Define dónde se van a colocar los comandos en la barra de comandos.
VisibilityConstraints elemento	Opcional. Determina la visibilidad estática de comandos y barras de herramientas.
Elemento KeyBindings	Opcional. Especifica las combinaciones de teclas de método abreviado, si las hay, para los comandos.
Elemento UsedCommands	Opcional. Permite que un VSPackage implemente opcionalmente su propia versión de funcionalidad originalmente compatible con otros VSPackages.
Elemento Símbolos	Opcional. Contiene los datos de símbolos -- GUID, IDs, etc.- para el compilador.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
None	

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Externo

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento Extern hace referencia a cualquier archivo de encabezado externo (.h) para combinar con el archivo .vsct en tiempo de compilación. Los archivos que se van a combinar deben estar en la ruta de acceso Include dada al compilador VSCT o a la que hace referencia un [elemento Include](#). Los archivos pueden ser otros archivos .vsct o archivos de encabezado C++.

Las definiciones de los archivos de encabezado deben tener el formato "#define [Símbolo] [Valor]" El valor puede ser otro símbolo si está definido previamente. Las definiciones se pueden utilizar en instrucciones condicionales de elementos de comando. Cualquier símbolo que no se utilice realmente será descartado.

CommandTable Element Extern Element

Sintaxis

```
<Extern href="stdidcmd.h" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
href	Necesario. La ruta de acceso al archivo de encabezado: href"stdidcmd.h"
Condición	Opcional. Consulte Atributos condicionales .
language	Opcional. El idioma predeterminado < de todas las cadenas> elementos de la tabla de comandos: "en-nosotros"

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Ninguno.	Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos (es decir, elementos de menú, menús, barras de herramientas y cuadros combinados) que un VSPackage proporciona al IDE.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<CommandTable xmlns="http://schemas.microsoft.com/VisualStudio/2005-10-
  18/CommandTable" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <Extern href="C:\VSCore\vscommon\inc\vsshlids.h"/>
  ...
  <Commands package="guidMyPackage">
</CommandTable>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)
- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Incluir elemento

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento Include especifica un archivo que se puede ubicar en la ruta de acceso de inclusión proporcionada para insertarlo en el archivo actual. Todos los símbolos y tipos definidos pasarán a formar parte del resultado compilado.

Sintaxis

```
<Include href="stdidcmd.h" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
href	Necesario. La ruta de acceso al archivo de encabezado: href"stdidcmd.h"
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Ninguno.	Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos (es decir, elementos de menú, menús, barras de herramientas y cuadros combinados) que un VSPackage proporciona al IDE.

Ejemplo

```
<Include href="PackagePlacements.vsct"/>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Definir elemento

06/04/2020 • 2 minutes to read • [Edit Online](#)

Define un nombre de símbolo y un par de valores. Este símbolo se puede evaluar mediante atributos condicionales. Para obtener más información, consulte [Atributos condicionales](#). Consulte también el [elemento Símbolos](#).

Sintaxis

```
<Define name="Mode" value="Standard" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
name	Necesario. El nombre del símbolo: nombre "Modo"
value	Necesario. El valor del símbolo: valor "Estándar"
Condición	Opcional. Para obtener más información, consulte Atributos condicionales .

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos que un VSPackage proporciona al entorno de desarrollo integrado (IDE). Por ejemplo, elementos de menú, menús, barras de herramientas y cuadros combinados.

Ejemplo

```
<Define name="DEMO_UI"/>  
<Define name="MODE" value="Standard"/>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Commands

06/04/2020 • 2 minutes to read • [Edit Online](#)

Representa la colección de comandos de la barra de herramientas de VSPackage. La colección puede tener hasta cinco subsecciones, como se indica a continuación: menús, grupos, botones, combos y mapas de bits.

Cada elemento secundario de subsección, por ejemplo, <Menu>, se identifica mediante un identificador de comando único que es un PAR de identificador esunidy y GUID. El GUID identifica el "conjunto de comandos" y se utiliza para agrupar comandos relacionados lógicamente. El VSPackage debe definir su propio conjunto de comandos para evitar colisiones con los IDs de comando definidos por otros VSPackages.

Sintaxis

```
<Commands package="GuidMyPackage" >
  <Menus>... </Menus>
  <Groups>... </Groups>
  <Buttons>... </Buttons>
  <Combos>... </Combos>
  <Bitmaps>... </Bitmaps>
</Commands>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
paquete	GUID que identifica el VSPackage que proporciona los comandos. Por ejemplo, package-"guidVsPackage1Pkg".

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Menús	Define todos los menús que implementa un VSPackage.
Elemento Grupos	Contiene entradas que definen los grupos de comandos en un VSPackage.
Elemento Buttons	Elementos De botón de grupos.
Elemento Mapas de bits	Agrupar elementos de mapa de bits.
Elemento Combos	Agrupar elementos Combo.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan los comandos que un VSPackage proporciona al IDE. Los elementos posibles son elementos de menú, menús, barras de herramientas y cuadros combinados.

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar un [elemento Commands](#).

```
<Commands package="guidMyPackage">
  <Menus>
    <Menu Condition="'%(DEBUG)' != 'true'"
      guid="cmdSetGuidMyProductCommands" id="menuIDMainMenu"
      priority="0x0000" type="Menu">
      <Annotation>
        <Documentation>this is an annotation</Documentation>
        <AppInfo>
          <CustomData>
            <CustomSubElement>Some data</CustomSubElement>
          </CustomData>
        </AppInfo>
      </Annotation>
      <CommandFlag>AlwaysCreate</CommandFlag>
      <Strings>
        <ButtonText>MainMenu</ButtonText>
      </Strings>
    </Menu>
  </Menus>
</Commands>
```

Vea también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Elemento Menús

06/04/2020 • 2 minutes to read • [Edit Online](#)

Define todos los menús y barras de herramientas que implementa un VSPackage.

Sintaxis

```
<Menus>
  <Menu>... </Menu>
  <Menu>... </Menu>
</Menus>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Menús	Define todos los menús y barras de herramientas que implementa un VSPackage.
Elemento de menú	Representa un único menú o barra de herramientas.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos en el VSPackage.

Ejemplo

```
<Commands package="guidMyPackage">
  <Menus>
    <Menu Condition="'%(DEBUG)' != 'true'"
      guid="cmdSetGuidMyProductCommands" id="menuIDMainMenu"
      priority="0x0000" type="Menu">
      <Annotation>
        <Documentation>this is an annotation</Documentation>
        <AppInfo>
          <CustomData>
            <CustomSubElement>Some data</CustomSubElement>
          </CustomData>
        </AppInfo>
      </Annotation>
      <CommandFlag>AlwaysCreate</CommandFlag>
      <Strings>
        <ButtonText>MainMenu</ButtonText>
      </Strings>
    </Menu>
  </Menus>
</Commands>
```

Consulte también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Elemento de menú

06/04/2020 • 9 minutes to read • [Edit Online](#)

Define un elemento de menú. Estos son los seis tipos de menús: Contexto, Menu, MenuController, MenuControllerLatched, Toolbar y ToolWindowToolbar.

Sintaxis

```
<Menu guid="guidMyCommandSet" id="MyCommand" priority="0x100" type="button">
  <Parent>... </Parent>
  <CommandFlag>... </CommandFlag>
  <Strings>... </Strings>
</Menu>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID.
id	Necesario. ID del identificador de comando GUID/ID.
priority	Opcional. Valor numérico que especifica la posición relativa de un menú en un grupo de menús.
ToolbarPriorityInband	Opcional. Valor numérico que especifica la posición relativa de una barra de herramientas en una banda cuando se acopla la ventana.
type	Opcional. Un valor enumerado que especifica el tipo de elemento. Si no está presente, el tipo predeterminado es Menú. Context Un menú contextual que se muestra cuando un usuario hace clic con el botón derecho en una ventana. Un menú contextual tiene las siguientes características: - No utiliza los campos Padre y Prioridad cuando el menú se va a mostrar como un menú contextual. - Se puede utilizar como un submenú y también como un menú contextual. En este caso, se respetan los campos ID de grupo y Prioridad. - No siempre está disponible. Solo se muestra un menú contextual cuando se cumplen las siguientes condiciones: - Se muestra la ventana que lo aloja. - Un controlador de mouse en el <code>VSPackage</code> detecta un clic

ATRIBUTO	DESCRIPCION
	Un controlador de menú en el VSPackage detecta un clic derecho en la ventana y, a continuación, llama a un método que controla el comando. - El menú contextual se ShowContextMenu muestra llamando al método ShowContextMenu (el enfoque recomendado) o al método. Menú Proporciona un menú desplegable. Un menú desplegable tiene las siguientes características: - Respeta al Padre en su definición. - Debe tener un grupo Padre o un CommandPlacement en un grupo. - Puede ser un submenú en cualquier otro tipo de menú. - Se muestra automáticamente cada vez que se muestra su menú principal. - No requiere la implementación de ningún código VSPackage para que se muestre. MenuController Proporciona un menú desplegable de botones divididos, que normalmente se utiliza en las barras de herramientas. Un menú MenuController tiene las siguientes características: - Debe estar contenido en otro menú a través de Parent o CommandPlacement. - Respeta al Padre en su definición. - Puede tener cualquier tipo de menú como su padre. - Se hace automáticamente disponible cada vez que se muestra su menú principal. - No requiere soporte programático para que se muestre el menú. En el botón de menú se muestra un comando del menú de botones divididos. El comando mostrado tiene una de las siguientes características: - Es el último comando que se utilizó si el comando todavía se muestra y se habilita. - Es el primer comando mostrado. MenuControllerLatched Proporciona un menú desplegable de botones divididos para el que se puede especificar un comando como la selección predeterminada marcando el comando como bloqueado. Un comando bloqueado es un comando que se marca en el menú como seleccionado, normalmente mostrando una marca de verificación. Un comando se puede marcar como bloqueado si tiene el indicador <code>QueryStatus</code> OLECMDF_LATCHED establecido IOleCommandTarget en él en una implementación del método de la interfaz. Un menú MenuControllerLatched tiene las siguientes características: - Debe estar contenido en otro menú a través de un grupo Padre o CommandPlacement. - Respeta al Padre en su definición. - Puede tener cualquier tipo de menú como su padre. - Se hace disponible cada vez que se muestra su menú principal. - No requiere soporte programático para que se muestre el menú. En el botón de menú se muestra un comando del menú de botones divididos. El comando mostrado tiene una de las

ATRIBUTO	siguientes características: DESCRIPCIÓN
	- Es el primer comando mostrado que está bloqueado. - Es el primer comando mostrado. Barra de herramientas Proporciona una barra de herramientas. Una barra de herramientas tiene las siguientes características: - Ignora al padre en su definición. - No se puede hacer un submenú de ningún grupo, ni siquiera mediante CommandPlacement. - Siempre se puede mostrar haciendo clic en Barras de herramientas en el menú Ver. - Se puede mostrar mediante un VisibilityItem. - No requiere ningún código para crearlo. Para obtener un ejemplo sobre cómo crear una barra de herramientas, consulte Agregar una barra de herramientas . ToolWindowToolBar Proporciona una barra de herramientas que se adjunta a una ventana de herramientas específica, al igual que una barra de herramientas se adjunta al entorno de desarrollo. - Ignora al padre en su definición. - No se puede hacer un submenú de ningún grupo, ni siquiera mediante CommandPlacement. - Se muestra solo cuando se muestra la ventana de herramientas que hospeda la barra de herramientas y el VSPackage agrega explícitamente la barra de herramientas a la ventana de herramientas. Esto se realiza normalmente cuando se crea la ventana de herramientas obteniendo la propiedad host de la barra de herramientas (representada por la IVsToolWindowToolBarHost interfaz) desde el marco de la ventana de herramientas y, a continuación, llamando al AddToolBar método.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Parent	Opcional. El elemento primario del elemento de menú.
CommandFlag	Necesario. Consulte Elemento de indicador de comando. Los valores commandFlag válidos para un menú son los siguientes: - AlwaysCreate - DefaultDocked - DefaultInvisible: esta marca no afecta a la visualización de las barras de herramientas. - DontCache - DynamicVisibility: esta marca no afecta a la visualización de barras de herramientas. - IconAndText - NoCustomize - NotInTBList - NoToolBarClose - TextChanges - TextIsAnchorCommand

ELEMENTO	DESCRIPCIÓN
Cadenas	Necesario. Consulte Elemento Cadenas . El <code>ButtonText</code> elemento secundario debe definirse.
Anotación	Comentario opcional.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Menús	Define todos los menús que implementa un VSPackage.

Ejemplo

```
<Menu guid="cmdGuidWidgetCommands" id="menuIDEditWidget"
  priority="0x0002" type="Menu">
  <Parent guid="cmdSetGuidWidgetCommands" id="groupIDFileEdit">
    <CommandFlag>AlwaysCreate</CommandFlag>
    <Strings>
      <ButtonText>Edit Widget</ButtonText>
    </Strings>
  </Parent>
</Menu>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Groups

06/04/2020 • 2 minutes to read • [Edit Online](#)

Contiene entradas que definen los grupos de comandos de un VSPackage.

Sintaxis

```
<Groups>
  <Group>... </Group>
  <Group>... </Group>
</Groups>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Group, elemento	Representa un único grupo de comandos.
Elemento Grupos	Contiene entradas que definen los grupos de comandos de un VSPackage.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos de la barra de herramientas de VSPackage.

Ejemplo

```
<Groups>
  <Group guid="cmdSetGuidWidgetCommands" id="groupIDFileEdit">
    <Parent guid="guidSHLMainMenu" id="IDM_VS_TOOL_MAINMENU"/>
  </Group>
</Groups>
```

Consulte también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Group, elemento

06/04/2020 • 2 minutes to read • [Edit Online](#)

Define un grupo de comandos de VSPackage.

Sintaxis

```
<Group guid="guidMyCommandSet" id="MyGroup" priority="0x101">  
  <Parent>... </Parent>  
</Group>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID.
id	Necesario. ID del identificador de comando GUID/ID.
priority	Opcional. Valor numérico que especifica la prioridad.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Parent	Opcional. El elemento primario del botón.
Anotación	Comentario opcional.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Grupos	Contiene entradas que definen los grupos de comandos de un VSPackage.

Ejemplo

```
<Group guid="cmdSetGuidWidgetCommands" id="groupIDFileEdit">  
  <Parent guid="guidSHLMainMenu" id="IDM_VS_TOOL_MAINMENU"/>  
</Group>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Buttons

06/04/2020 • 2 minutes to read • [Edit Online](#)

Elementos [Button](#) de grupos, que representan comandos individuales.

Sintaxis

```
<Buttons>
  <Button>... </Button>
  <Button>... </Button>
</Buttons>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Buttons	Elementos De botón de grupos.
Elemento Button	Define un comando con el que el usuario puede interactuar.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos de la barra de herramientas de VSPackage.

Ejemplo

```
<Buttons>
  <Button guid="guidMenuAndCommandsCmdSet" id="cmdidMyCommand" priority="0x100" type="Button">
    <Parent guid="guidMenuAndCommandsCmdSet" id="MyMenuGroup"/>
    <Icon guid="guidGenericCmdBmp" id="bmpArrow"/>
    <Strings>
      <ButtonText>C# Command Sample</ButtonText>
    </Strings>
  </Button>
</Buttons>
```

Consulte también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Elemento Button

06/04/2020 • 2 minutes to read • [Edit Online](#)

Define un elemento con el que el usuario puede interactuar. Los botones pueden ser de diferentes tipos: Button, MenuButton y SplitDropDown.

Sintaxis

```
<Button guid="guidMyCommandSet" id="MyCommand" priority="0x102" type="button">
  <Parent>... </Parent>
  <Icon>... </Icon>
  <CommandFlag>... </CommandFlag>
  <Strings>... </Strings>
</Button>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID.
id	Necesario. ID del identificador de comando GUID/ID.
priority	Opcional. Valor numérico que especifica la prioridad.
type	Opcional. Un valor enumerado que especifica el tipo de botón. Si no se da, utiliza Button. Botón Un comando estándar que aparece en las barras de herramientas (normalmente como un botón icónico), menús y menús contextuales. MenuButton Elemento de menú que no ejecuta un comando, pero genera otro menú. SplitDropDown Controles, como los botones Deshacer y Rehacer de la barra de herramientas estándar de Microsoft Word.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento primario	Opcional. El elemento primario del botón.
Elemento Icono	Opcional. El icono asociado con el botón.
Elemento de indicador de comando	Necesario. Los valores CommandFlag válidos para un Button son los siguientes. - AllowParams - CommandWellOnly - DefaultDisabled - PredeterminadoInvisible - DontCache - DynamicItemStart - DinámicaVisibilidad - FixMenuController - IconAndText - NoButtonCustomize - NoPersonalizar - NoKeyCustomize - NoShowOnMenuController - Picto - PostExec - ProfferedCmd - RoutetoDocs - TextCascadeUseBtn - TextMenuUseButton - TextChanges - TextChangesButton - TextContextUseButton - TextMenuCtrlUseMenu - TextMenuUseButton - TextOnly
Elemento Strings	Necesario. El elemento secundario ButtonText debe definirse.

ELEMENTO	DESCRIPCIÓN
Anotación	Comentario opcional.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Buttons	Elementos De botón de grupos.

Ejemplo

En el ejemplo siguiente se define un botón en un archivo *.vsct*.

```
<Button guid="guidMenuTextCmdSet" id="cmdidMyCommand" priority="0x0100" type="Button">
  <Parent guid="guidMenuTextCmdSet" id="MyMenuGroup" />
  <Icon guid="guidImages" id="bmpPic1" />
  <CommandFlag>TextChanges</CommandFlag>
  <Strings>
    <CommandName>cmdidMyCommand</CommandName>
    <ButtonText>My Command name</ButtonText>
  </Strings>
</Button>
```

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Mapas de bits

06/04/2020 • 2 minutes to read • [Edit Online](#)

Agrupar elementos de elemento de mapa de bits.

Sintaxis

```
<Bitmaps>
  <Bitmap>... </Bitmap>
  <Bitmap>... </Bitmap>
</Bitmaps>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Mapas de bits	Agrupar elementos de mapa de bits.
Elemento de mapa de bits	Define un mapa de bits.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos de la barra de herramientas de VSPackage.

Ejemplo

```
<Bitmaps>
  <Bitmap guid="guidWidgetIcons" href="WidgetToolbarIcons_32.bmp" />
  <Bitmap guid="guidWidgetIcons2" resID="IDBMP_WIDGETICONS"
    usedList="1, 2, 3, 4"/>
</Bitmaps>
```

Consulte también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Elemento de mapa de bits

06/04/2020 • 3 minutes to read • [Edit Online](#)

Define un mapa de bits. El mapa de bits se carga desde un recurso o desde un archivo.

Sintaxis

```
<Bitmap guid="guidImages" href="images\MyImage.bmp" usedList="bmp1, bmp2, bmp3" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID. El atributo guid de un mapa de bits no está asociado a ningún VSPackage u otro grupo de comandos. Debe ser único para la definición de mapa de bits y no debe utilizarse para ningún otro propósito.
Resnum	ID del identificador de comando GUID/ID. Se requiere el atributo resID o href. El atributo resID es un identificador de recurso entero que determina la franja de mapa de bits que se va a cargar durante la combinación de la tabla de comandos. Cuando se carga la tabla de comandos, los mapas de bits especificados por el identificador de recurso se cargarán desde el recurso del mismo módulo.
usedList	Necesario si el atributo resID está presente. Selecciona las imágenes disponibles en la tira de mapa de bits.
href	Ruta de acceso al mapa de bits. Se requiere el atributo resID o href. Se busca la ruta de inclusión para el archivo de imagen indicado, que está incrustado en el binario resultante. Durante la combinación de la tabla de comandos, se copia la imagen y no se requiere ninguna búsqueda o carga de recursos adicionales. Si el atributo usedList no está presente, todas las imágenes de la tira están disponibles. Nota: Las imágenes se pueden suministrar en uno de varios formatos que incluyen <i>.bmp</i>, <i>.png</i>, <i>.gif</i>. Las versiones anteriores del compilador no admiten imágenes de mapa de bits de 32 bits que tenían información alfa para la transparencia parcial. La solución alternativa para estas versiones es utilizar el formato <i>.png</i>.

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Mapas de bits	Agrupar elementos de mapa de bits.

Ejemplo

```
<Bitmap guid="guidWidgetIcons" href="WidgetToolbarIcons_32.bmp" />
<Bitmap guid="guidWidgetIcons2" resID="IDBMP_WIDGETICONS"
  usedList="1, 2, 3, 4"/>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Combos

06/04/2020 • 2 minutes to read • [Edit Online](#)

Agrupar [elementos de elemento s Combo](#).

Sintaxis

```
<Combos>
  <Combo>... </Combo>
  <Combo>... </Combo>
</Combos>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento Combos	Agrupar elementos Combo.
Elemento combinado	Define los comandos que aparecen en un cuadro combinado.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos de la barra de herramientas de VSPackage.

Ejemplo

```
<Combos>
  <Combo guid="guidWidgetPackage" id="cmdidInsertOptions"
    defaultWidth="100" idCommandList="cmdidGetInsertOptionsList">
    <CommandFlag>DynamicVisibility</CommandFlag>
    <Strings>
      <ButtonText>Select Insert Options</ButtonText>
    </Strings>
  </Combo>

  <Combo guid="guidWidgetPackage" id="cmdidInsertOptions"
    priority="0x0500" type="DropDownCombo" defaultWidth="100"
    idCommandList="cmdidGetInsertOptionsList">
    <Parent guid="cmdSetGuidWidgetCommands" id="groupIDFileEdit">
    <CommandFlag>DynamicVisibility</CommandFlag>
    <Strings>
      <ButtonText>Select Insert Options</ButtonText>
    </Strings>
  </Combo>
</Combos>
```

Consulte también

- [Cómo VSPackages agregan elementos de interfaz de usuario](#)
- [Comandos, menús y barras de herramientas](#)

Elemento combinado

06/04/2020 • 3 minutes to read • [Edit Online](#)

Define los comandos que aparecen en un cuadro combinado. Hay cuatro tipos de cuadros combinados, como se indica a continuación: DropDownCombo, DynamicCombo, IndexCombo y MRUCombo.

Sintaxis

```
<combo guid="guidMyCommandSet" id="MyCommand" defaultWidth="20" idCommandList="MyCommandListID"
priority="0x102" type="DropDownCombo">
  <Parent>... </Parent>
  <CommandFlag>... </CommandFlag>
  <Strings>... </Strings>
</combo>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID.
id	Necesario. ID del identificador de comando GUID/ID.
defaultWidth	Necesario. Entero que especifica un ancho de píxel para el cuadro combinado.
idCommandList	Necesario. Un identificador que se envía al destino del comando activo para recuperar la lista de elementos que se mostrarán en el cuadro combinado. El identificador estará en el mismo ámbito GUID que el control.
priority	Opcional. Valor numérico que especifica la prioridad.

ATRIBUTO	DESCRIPCIÓN
type	Opcional. Un valor enumerado que especifica el tipo de botón. Si no se da, utiliza Button. DropDownCombo El VSPackage es responsable de rellenar el contenido de este cuadro combinado. El usuario no puede escribir nada en el cuadro de texto de esta lista desplegable. DynamicCombo El VSPackage es responsable de rellenar el contenido de este cuadro combinado. El usuario puede editar este combo y también seleccionar elementos en él. IndexCombo Lo mismo que DynamicCombo excepto que genera el índice del elemento en lugar de su texto. MRUCombo Lleno por el entorno de desarrollo integrado (IDE) en nombre del VSPackage. El usuario puede editar en este cuadro combinado. El IDE recuerda hasta las últimas 16 entradas por cuadro combinado. Cuando el usuario selecciona algo en el cuadro combinado o escribe algo nuevo, el IDE notifica el VSPackage adecuado.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Parent	Opcional. El elemento primario del botón.

ELEMENTO	DESCRIPCIÓN
CommandFlag	Necesario. Consulte Elemento de indicador de comando. Los valores CommandFlag válidos para un Button son los siguientes. - CaseSensitive - CommandWellOnly - DefaultDisabled - PredeterminadoInvisible - DinámicaVisibilidad - FilterKeys - IconAndText - NoAutoComplete - NoButtonCustomize - NoPersonalizar - NoKeyCustomize - StretchHorizontally
Cadenas	Necesario. Consulte Elemento Cadenas. El elemento secundario ButtonText debe definirse.
Anotación	Comentario opcional.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Comandos	Representa la colección de comandos de la barra de herramientas de VSPackage.

Ejemplo

```
<Combo guid="guidWidgetPackage" id="cmdidInsertOptions"
  defaultWidth="100" idCommandList="cmdidGetInsertOptionsList">
  <CommandFlag>DynamicVisibility</CommandFlag>
  <Strings>
    <ButtonText>Select Insert Options</ButtonText>
  </Strings>
</Combo>

<Combo guid="guidWidgetPackage" id="cmdidInsertOptions"
  priority="0x0500" type="DropDownCombo" defaultWidth="100"
  idCommandList="cmdidGetInsertOptionsList">
  <Parent guid="cmdSetGuidWidgetCommands" id="groupIDFileEdit">
  <CommandFlag>DynamicVisibility</CommandFlag>
  <Strings>
    <ButtonText>Select Insert Options</ButtonText>
  </Strings>
</Combo>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento CommandPlacements

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento CommandPlacements agrupa elementos CommandPlacement y otras agrupaciones CommandPlacements.

El elemento CommandPlacements es opcional. Si no se deben incluir comandos, grupos o menús en una ubicación secundaria, no es necesario incluir esta sección en el archivo `.vsct`.

Sintaxis

```
<CommandPlacements>
  <CommandPlacement>... </CommandPlacement>
  <CommandPlacement>... </CommandPlacement>
</CommandPlacements>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
CommandPlacements	Agrupar commandPlacement elementos y otras agrupaciones CommandPlacements.
Elemento CommandPlacement	Permite incluir botones, grupos y menús en más de un grupo o menú.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos.

Ejemplo

```
<CommandPlacements>
  <CommandPlacement guid="guidWidgetPackage" id="cmdidInsertOptions"
    priority="0x0300">
    <Parent guid="cmdGuidWidgetCommands" id="menuIDEditWidget"/>
  </CommandPlacement>
</CommandPlacements>
```

Consulte también

- [Elemento CommandPlacement](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento CommandPlacement

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento CommandPlacement permite incluir botones, grupos y menús en más de un grupo o menú. Mediante el uso de la CommandPlacement elemento, no es necesario redefinir completamente estos elementos para modificar el aspecto de una interfaz de usuario.

Para obtener más información, consulte [Crear grupos de botones reutilizables](#).

Sintaxis

```
<CommandPlacement guid="guidMyCommandSet" id="MyCommand" priority="0x001" >
  <Parent>... </Parent>
</CommandPlacement>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. El guid del conjunto de comandos , tal como se define en el elemento Símbolos .
id	Necesario. El identificador del menú, grupo o comando que se <code>Symbols Element</code> va a colocar, tal como se define en el archivo .
priority	Necesario. Determina la posición visual del elemento en su elemento primario.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Parent	Necesario. El menú o grupo que hospeda el elemento que se va a colocar.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandPlacements	Especifica grupos de CommandPlacements y CommandPlacement elementos.

Ejemplo

```
<CommandPlacements>
  <CommandPlacement guid="guidWidgetPackage" id="cmdidInsertOptions"
    priority="0x0300">
    <Parent guid="cmdGuidWidgetCommands" id="menuIDEditWidget"/>
  </CommandPlacement>
</CommandPlacements>
```

Consulte también

- [Elemento CommandPlacements](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

VisibilityConstraints elemento

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento VisibilityConstraints determina la visibilidad estática de grupos de comandos y barras de herramientas. La visibilidad se controla Visual Studio primero mediante el entorno de desarrollo integrado (IDE) sin cargar el VSPackage.

Sintaxis

```
<VisibilityConstraints>
  <VisibilityItem />
  <VisibilityItem />
</VisibilityConstraints>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento VisibilityItem	Determina la visibilidad estática de comandos y barras de herramientas.
VisibilityConstraints	Determina la visibilidad estática de grupos de comandos y barras de herramientas.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan los comandos (por ejemplo, elementos de menú, menús, barras de herramientas y cuadros combinados) que un VSPackage proporciona al IDE.

Ejemplo

```
<VisibilityConstraints>
  <VisibilityItem guid="cmdSetGuidMyProductCommands"      id="cmdidAddWidget"
    context="guidNotViewSourceMode"/>
</VisibilityConstraints>
```


Consulte también

- [Elemento VisibilityItem](#)
- [Tabla de comandos de Visual Studio \(. Vsct\)](#)

Elemento VisibilityItem

06/04/2020 • 3 minutes to read • [Edit Online](#)

El `VisibilityItem` elemento determina la visibilidad estática de comandos y barras de herramientas. Cada entrada identifica un comando o menú, y también un contexto de interfaz de usuario de comandos asociado. Visual Studio detecta comandos, menús y barras de herramientas, y su visibilidad, sin cargar los `VSPackages` que los definen. El IDE `IsCmdUIContextActive` utiliza el método para determinar si un contexto de interfaz de usuario de comando está activo.

Después de cargar el `VSPackage`, Visual Studio espera que la visibilidad de comandos sea determinada por el `VSPackage` en lugar de el `VisibilityItem`. Para determinar la visibilidad del comando, `BeforeQueryStatus` puede implementar `QueryStatus` el controlador de eventos o el método, dependiendo de cómo haya implementado el comando.

Un comando o menú `VisibilityItem` que tiene un elemento solo aparece cuando el contexto asociado está activo. Puede asociar un único comando, menú o barra de herramientas con uno o varios contextos de interfaz de usuario de comandos incluyendo una entrada para cada combinación de comando-contexto. Si un comando o menú está asociado a varios contextos de interfaz de usuario de comandos, el comando o menú está visible cuando cualquiera de los contextos de interfaz de usuario de comandos asociados está activo.

El `VisibilityItem` elemento solo se aplica a comandos, menús y barras de herramientas, no a grupos. Un elemento que no `VisibilityItem` tiene un elemento relacionado es visible siempre que su menú primario está activo.

Sintaxis

```
<VisibilityItem
  guid ="cmdGuidMyProductCommands"
  id="cmdidAddWidget"
  context="guidNotViewSourceMode"/>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. Guid del identificador de comando GUID/ID.
id	Necesario. El identificador del identificador de comando GUID/ID.
context	Necesario. El contexto de la interfaz de usuario en el que está visible el comando.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

None

Elementos primarios

ELEMENTO	DESCRIPCIÓN
VisibilityConstraints elemento	El <code>VisibilityConstraints</code> elemento determina la visibilidad estática de grupos de comandos y barras de herramientas.

Observaciones

Los contextos estándar de la interfaz de usuario de Visual Studio se definen en la *ruta* [UIContextGuids](#) [UIContextGuids80](#) de instalación del SDK de Visual Studio . En la [VSConstants](#) clase se define un conjunto más completo de contextos de interfaz de usuario.

Ejemplo

```
<VisibilityConstraints>
  <VisibilityItem guid="cmdSetGuidMyProductCommands"      id="cmdidAddWidget"
    context="guidNotViewSourceMode" />
</VisibilityConstraints>
```

Consulte también

- [IsCmdUIContextActive](#)
- [BeforeQueryStatus](#)
- [VSConstants](#)
- [UIContextGuids](#)
- [UIContextGuids80](#)
- [VisibilityConstraints elemento](#)
- [Tabla de comandos de Visual Studio \(. Vsct\) Archivos](#)

Elemento KeyBindings

06/04/2020 • 2 minutes to read • [Edit Online](#)

El keyBindings elemento agrupa KeyBinding elementos y otras KeyBindings agrupaciones.

Sintaxis

```
<KeyBindings>
  <KeyBinding>... </KeyBinding>
  <KeyBinding>... </KeyBinding>
</KeyBindings>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento KeyBinding	Especifica los métodos abreviados de teclado para los comandos.
Keybindings	Agrupa KeyBinding elementos y otras keyBindings agrupaciones.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos.

Ejemplo

```
<KeyBindings>
  <KeyBinding guid="guidWidgetPackage" id="cmdidUpdateWidget"
    editor="guidWidgetEditor" key1="VK_F5"/>
  <KeyBinding guid="guidWidgetPackage" id="cmdidRunWidget"
    editor="guidWidgetEditor" key1="VK_F5" mod1="Control"/>
</KeyBindings>
```

Consulte también

- [Elemento KeyBinding](#)

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento KeyBinding

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento KeyBinding especifica los métodos abreviados de teclado para los comandos.

Los comandos pueden tener enlaces de clave simple y doble asociados a ellos. Un ejemplo de un enlace de una sola tecla es **Ctrl+S** para el **comando Guardar**. Los enlaces de teclas dobles requieren dos combinaciones de teclas sucesivas para desencadenar un comando. Un ejemplo de un enlace de doble tecla es **Ctrl*+K, Ctrl+K**** para establecer un marcador.

Sintaxis

```
<Keybinding guid="MyGuid" id="MyId" Editor="MyEditor" key1="B" key2="x" mod1="Control" mod2="Alt" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario.
id	Necesario.
editor	Necesario. El GUID del editor indica el contexto de edición para el que estará activo este método abreviado de teclado. El valor de ámbito de enlace global es "guidVSSStd97".
key1	Necesario. Los valores válidos incluyen todos los alfanuméricos tipificables y también los valores hexadecimales de dos dígitos precedidos por 0x y VK_constants .
mod1	Opcional. Cualquier combinación de Ctrl , Alty Mayús separados por espacio.
key2	Opcional. Los valores válidos incluyen todos los alfanuméricos tipificables y también los valores hexadecimales de dos dígitos precedidos por 0x y VK_constants .
mod2	Opcional. Cualquier combinación de Ctrl , Alty Mayús separados por espacio.
emulator	Opcional.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Parent	
Anotación	

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento KeyBindings	Agrupar KeyBinding elementos y otras keyBindings agrupaciones.

Ejemplo

```
<KeyBindings>
  <KeyBinding guid="guidWidgetPackage" id="cmdidUpdateWidget"
    editor="guidWidgetEditor" key1="VK_F5"/>
  <KeyBinding guid="guidWidgetPackage" id="cmdidRunWidget"
    editor="guidWidgetEditor" key1="VK_F5" mod1="Control"/>
</KeyBindings>
```

Consulte también

- [Elemento KeyBindings](#)
- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

UsedCommands (Elemento)

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento UsedCommands agrupa elementos UsedCommand y otras agrupaciones UsedCommands.

El elemento UsedCommands es opcional. Si no llama a comandos definidos fuera del paquete, no tiene que incluir esta sección en el archivo .vsct.

Sintaxis

```
<UsedCommands condition="Defined(DEBUG)">
  <UsedCommand ... />
</UsedCommands>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
UsedCommand (Elemento)	El comando que implementa otro código.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
CommandTable (Elemento)	Define todos los elementos que representan comandos (por ejemplo, elementos de menú, menús, barras de herramientas y cuadros combinados) que un VSPackage proporciona al entorno de desarrollo integrado (IDE).

Ejemplo

```
<UsedCommands>
  <UsedCommand guid="guidVSStd97" id="cmdidCut"/>
  <UsedCommand guid="guidVSStd97" id="cmdidCopy"/>
  <UsedCommand guid="guidVSStd97" id="cmdidPaste"/>
</UsedCommands>
```

Consulte también

- [UsedCommand \(Elemento\)](#)

- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)

UsedCommand (Elemento)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Permite que un VSPackage tenga acceso a un comando que se define en otro archivo .vsct. Por ejemplo, si el VSPackage usa el comando Visual Studio **Copy** estándar, definido por el shell, puede agregar el comando a un menú o barra de herramientas sin volver a implementarlo.

Sintaxis

```
<UsedCommand guid="guidMyCommandGroup" id="MyCommand" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del par de identificadorGUID que identifica el comando.
id	Necesario. El identificador del par de identificadorGUID que identifica el comando.
Condición	Opcional. Consulte Atributos condicionales .

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
None	

Elementos primarios

ELEMENTO	DESCRIPCIÓN
UsedCommands (Elemento)	Agrupar los elementos UsedCommand y otras agrupaciones UsedCommands.

Observaciones

Al agregar un `<UsedCommands>` comando al elemento, un Visual Studio VSPackage informa al entorno que el VSPackage requiere el comando. Debe agregar `<UsedCommand>` un elemento para cualquier comando que requiera el paquete que podría no incluirse en todas las versiones y configuraciones de Visual Studio. Por ejemplo, si el paquete llama a un comando específico de Visual C++, el comando `<UsedCommand>` no estará disponible para los usuarios de Visual Web Developer a menos que incluya un elemento para el comando.

Ejemplo

```
<UsedCommands>
  <UsedCommand guid="guidVSStd97" id="cmdidCut"/>
  <UsedCommand guid="guidVSStd97" id="cmdidCopy"/>
  <UsedCommand guid="guidVSStd97" id="cmdidPaste"/>
</UsedCommands>
```

Consulte también

- [UsedCommands \(Elemento\)](#)
- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)

Elemento primario

06/04/2020 • 2 minutes to read • [Edit Online](#)

El elemento primario de un botón o cuadro combinado solo puede ser un grupo. El elemento primario de un menú o grupo puede ser cualquier otro menú o grupo. En un [elemento CommandPlacement](#), este elemento es necesario; en todos los demás casos es opcional. Si se omite este elemento, `Group_Undefined:0` se implicará el elemento primario de.

Sintaxis

```
<Parent guid="guidMyCommandSet" id="MyParentGroupOrMenu" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. GUID del identificador de comando GUID/ID.
id	Necesario. ID del identificador de comando GUID/ID.

Elementos secundarios

None

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento CommandTable	Define todos los elementos que representan comandos que un VSPackage proporciona al entorno de desarrollo integrado (IDE). Por ejemplo, elementos de menú, menús, barras de herramientas y cuadros combinados.
Elemento Buttons	Elementos de elemento Button de grupos.
Elemento Menús	Define todos los menús que implementa un VSPackage.
Elemento Grupos	Contiene entradas que definen los grupos de comandos de un VSPackage.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento Icono

06/04/2020 • 2 minutes to read • [Edit Online](#)

El atributo guid de la etiqueta Icon es el guid de un mapa de bits definido. El `id` atributo selecciona la ranura en la franja de mapa de bits. Este elemento es opcional. Si no se incluye este elemento, se implicará el valor de `guidOfficeIcon:msotcidNoIcon`.

Sintaxis

```
<Icon guid="guidImages" id="bmpPic1" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
guid	Necesario. El guid de un mapa de bits definido.
id	Necesario. Selecciona la ranura en la franja de mapa de bits.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Ninguno.	Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Buttons	

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Strings (Elemento)

06/04/2020 • 5 minutes to read • [Edit Online](#)

El elemento Strings debe contener al menos un elemento secundario **ButtonText**. Todos los demás elementos secundarios son opcionales. Los caracteres XML no válidos como '&' y '<' deben codificarse como entidades ('&' y '<' etc.).

Una y unana en la cadena de texto especifica el método abreviado de teclado para el comando.

Sintaxis

```
<Strings>
  <ButtonText>... </ButtonText>
  <CommandName>... </CommandName>
</Strings>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
language	Opcional. Idioma".

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
ButtonText	Este campo y los cinco campos de texto siguientes en una definición de comando le permiten especificar el texto que aparece en varios menús. De forma <code>ButtonText</code> predeterminada, el campo aparece en los controladores de menú. El <code>ButtonText</code> campo también se convierte en el valor predeterminado si los demás campos de texto están en blanco. El <code>ButtonText</code> campo no puede estar en blanco aunque se especifiquen los demás campos de texto.
ToolTipText	El <code>ToolTipText</code> campo especifica el texto que aparece en la información sobre herramientas de un elemento de menú. Si <code>ToolTipText</code> el campo está <code>ButtonText</code> en blanco, se utiliza el campo.

ELEMENTO	DESCRIPCIÓN
MenuText	El <code>MenuText</code> campo especifica el texto que se muestra para un comando si está en el menú principal, una barra de herramientas, en un menú contextual o en un submenú. Si <code>MenuText</code> el campo está en blanco, el entorno <code>ButtonText</code> de desarrollo integrado (IDE) utiliza el campo. El <code>MenuText</code> campo también se puede utilizar para la localización. Para los menús contextuales, el <code>MenuText</code> campo es el nombre que se muestra en la barra de herramientas Menús contextuales, que permite la personalización de los menús contextuales en el IDE. Por lo tanto, sea específico en lo que usted nombra el menú contextual; por ejemplo, utilice "Menú de acceso directo del paquete de widgets" en lugar de "Acceso directo". Si <code>MenuText</code> no se especifica el <code>ButtonText</code> campo, se utiliza el campo.
CommandName	El <code>CommandName</code> campo especifica el texto que aparece en la categoría de teclado en la ficha Comandos del cuadro de diálogo Personalizar (disponible haciendo clic en Personalizar en el menú Herramientas).
CanonicalName	El <code>CanonicalName</code> campo Inglés especifica el nombre del comando en texto en inglés que se puede introducir en la ventana Comando para ejecutar el elemento de menú. El IDE elimina los caracteres que no son letras, dígitos, guiones bajos o períodos incrustados. A continuación, este texto <code>ButtonText</code> se concatena al campo para definir el comando. Por ejemplo, Nuevo proyecto en el archivo menú se convierte en el comando, File.NewProject. Si no <code>CanonicalName</code> se especifica el campo Inglés, el IDE usa el <code>ButtonText</code> campo y elimina todas las letras, dígitos, guiones bajos y períodos incrustados. Por ejemplo, el texto del botón "&Definir comandos..." se convierte en DefineCommands, donde se quitan las y anquiersand, el espacio y los puntos suspensivos. Si <code>TextChanges</code> se especifica el indicador y se cambia el texto del comando, el comando correspondiente reconocido por la ventana Comando no cambia; sigue siendo la forma <code>ButtonText</code> canónica de los campos originales o ingleses. <code>CanonicalName</code>
LocCanonicalName	El <code>LocCanonicalName</code> campo se comporta de <code>CanonicalName</code> forma idéntica al campo Inglés, pero permite especificar el texto del comando localizado. Se pueden especificar ambos campos canónicos. Dado que el IDE solo analiza el texto introducido en la ventana Comando y lo asocia con un comando, el texto en inglés y el no inglés se pueden asociar al mismo comando.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Button (Elemento)	Define un elemento con el que el usuario puede interactuar.
Menu (Elemento)	Define un único elemento de menú.
Combo (Elemento)	Define los comandos que aparecen en un cuadro combinado.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)

Elemento ButtonText

06/04/2020 • 2 minutes to read • [Edit Online](#)

Este campo le permite especificar el texto que aparece en varios menús. De forma `ButtonText` predeterminada, el elemento aparece en los controladores de menú. El `ButtonText` elemento también se convierte en el valor predeterminado si los demás campos de texto están en blanco. El `ButtonText` elemento no puede estar en blanco aunque se especifiquen los demás campos de texto.

Sintaxis

```
<ButtonText>My Command</ButtonText>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Strings	Agrupar elementos de <code>ButtonText</code> <code>CommandName</code> texto, como y .

Valor de texto

El valor de `ButtonText` texto del elemento proporciona el texto que se muestra para los elementos de menú, combos y otros elementos de interfaz de usuario (UI) que tienen texto visible.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento CommandName

06/04/2020 • 2 minutes to read • [Edit Online](#)

El `CommandName` elemento especifica el texto que aparece en la categoría de teclado en el cuadro de diálogo **Opciones** y en la lista **Comandos** del cuadro de diálogo **Personalizar**.

Sintaxis

```
<CommandName>MyCommand</CommandName>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

Ninguno.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Strings	Agrupar elementos de <code>ButtonText</code> <code>CommandName</code> texto, como y .

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Bandera de mando Element

06/04/2020 • 10 minutes to read • [Edit Online](#)

Modifica su elemento primario.

Sintaxis

```
<CommandFlag>DynamicVisibility</CommandFlag>
```

Atributos y elementos

En la siguiente sección se describen los valores de elemento válidos.

Atributos

Ninguno.

Elementos secundarios

VALUE	DESCRIPCIÓN
AllowParams	Indica que los usuarios pueden introducir parámetros de comando en la ventana Comando cuando escriben el nombre canónico del comando. Válido para: <code>Button</code>
AlwaysCreate	El menú se crea incluso si no tiene grupos ni botones. Válido para: <code>Menu</code>
CaseSensitive	Las entradas de usuario distinguen mayúsculas de minúsculas. Válido para: <code>Combo</code>
CommandWellOnly	Aplique esta marca si el comando no aparece en el menú de nivel superior y desea que esté disponible para la personalización de shell adicional, por ejemplo, para enlazarlo a un método abreviado de teclado. Después de instalar el VSPackage, puede personalizar estos comandos abriendo el cuadro de diálogo Opciones y, a continuación, editando la ubicación del comando en la categoría Entorno de teclado. Esta marca no afecta a la colocación en menús contextuales, barras de herramientas, controladores de menú o submenús. Válido para: <code>Button</code>, <code>Combo</code>

VALUE	DESCRIPCIÓN
DefaultDisabled	De forma predeterminada, el comando está deshabilitado si el VSPackage que lo implementa no está cargado o no se ha llamado al <code>QueryStatus</code> método. Válido para: <code>Button</code> , <code>Combo</code>
DefaultDocked	Acoplado de forma predeterminada. Esta configuración ya no se aplica a las barras de herramientas porque siempre están acopladas.
DefaultInvisible	De forma predeterminada, el comando es invisible si el VSPackage que lo implementa no está cargado o no se ha llamado al <code>QueryStatus</code> método. Le recomendamos que combine <code>DynamicVisibility</code> esto con la bandera. Válido <code>Button</code> para: <code>Combo</code> , , <code>Menu</code>
DontCache	El entorno de desarrollo <code>QueryStatus</code> no almacena en caché los resultados del método para este comando. Para un menú, esto indica a un controlador de menú que no almacene en caché el texto de sus elementos de menú. Utilice esta marca cuando el menú contenga elementos dinámicos o elementos que tengan texto dinámico. Válido para: <code>Button</code> , <code>Menu</code>
DynamicItemStart	Indica el comienzo de una lista dinámica. Esto permite al entorno crear una lista <code>QueryStatus</code> llamando sucesivamente al método en los elementos de lista hasta que se devuelve la marca <code>OLECMDERR_E_UNsupported</code>. Esto funciona bien para elementos como las listas y listas de ventanas utilizadas más recientemente (MRU). Válido para: <code>Button</code>

VALUE	DESCRIPCIÓN
DinámicaVisibilidad	La visibilidad del comando se <code>QueryStatus</code> puede cambiar a través del <code>VisibilityConstraints</code> método o a través de un GUID de contexto que se incluye en la sección. Se aplica a los comandos que aparecen en los menús y las barras de herramientas de la ventana de herramientas, pero no en las barras de herramientas de nivel superior que aparecen en la ventana principal. Los elementos de la barra de herramientas de nivel superior <code>QueryStatus</code> se pueden deshabilitar pero no ocultar cuando se devuelve la marca <code>OLECMDF_INVISIBLE</code> desde el método. Los comandos de la barra de herramientas que aparecen en las barras de herramientas de la ventana de herramientas se pueden ocultar. En un menú, esta marca también indica que debe ocultarse automáticamente cuando todos sus miembros están ocultos. Esta marca se asigna normalmente a los submenús porque los menús de nivel superior ya tienen este comportamiento. Esta marca debe combinarse con la <code>DefaultInvisible</code> bandera. Válido <code>Button</code> para: <code>Combo</code> , , <code>Menu</code>
Filterkeys	Consulte el tema Claves de filtrado en Elemento combinado. Válido para: <code>Combo</code>
FixMenuController	Si este comando se coloca en un controlador de menú, el comando siempre es el predeterminado; es decir, el comando se selecciona cada vez que se selecciona el propio botón del controlador de menú. Si el controlador <code>TextIsAnchorCommand</code> de menú tiene el indicador establecido, el controlador <code>FixMenuController</code> de menú también toma su texto del comando que tiene la marca. Solo un comando en un <code>FixMenuController</code> controlador de menú debe tener la marca. Si más de un comando está marcado así, el último comando del menú se convierte en el comando predeterminado. Válido para: <code>Button</code>
IconAndText	Mostrar un icono y texto en el menú y la barra de herramientas. Válido <code>Button</code> para: <code>Combo</code> , , <code>Menu</code>
NoAutoComplete	La función de autocompletar está deshabilitada. Válido para: <code>Combo</code>

VALUE	DESCRIPCIÓN
NoButtonCustomize	No permita que el usuario personalice este botón. Válido para: Button , Combo
NoKeyCustomize	No habilite la personalización del teclado. Válido para: Button , Combo
NoShowOnMenuController	Si este comando se coloca en un controlador de menú, el comando no aparece en la lista desplegable. Válido para: Button
NotInTBList	No aparece en la lista de barras de herramientas disponibles. Esto solo es válido para los tipos de menú de la barra de herramientas. Válido para: Menu
NoToolbarClose	El usuario no puede cerrar la barra de herramientas. Esto solo es válido para los tipos de menú de la barra de herramientas. Válido para: Menu
Pict	Mostrar solo un icono en una barra de herramientas, pero solo texto en un menú. Si no se especifica ningún icono, muestra un espacio en blanco en el que se puede hacer clic en una barra de herramientas. Válido para: Button
PostExec	Hace que el comando no se bloquee. El entorno de desarrollo aplaza la ejecución hasta que se completan todas las consultas de preprocesamiento. Válido para: Button
RouteToDocs	El comando se enruta al documento activo. Válido para: Button
StretchHorizontally	Cuando se establece esta marca, el ancho se convierte en el ancho mínimo para el cuadro combinado y, si hay espacio en la barra de herramientas, el cuadro combinado se estira para rellenar el espacio disponible. Esto solo se produce si la barra de herramientas está acoplada horizontalmente y solo un cuadro combinado de la barra de herramientas puede utilizar la marca (la marca se omite en todos excepto en el primer cuadro combinado). Válido para: Combo

VALUE	DESCRIPCIÓN
TextMenuUseButton	Utilice <code>ButtonText</code> el campo para los menús. El campo <code>MenuText</code> predeterminado es si se especifica. Válido para: <code>Button</code>
TextChanges	El texto del comando o menú se puede <code>QueryStatus</code> cambiar en tiempo de ejecución, normalmente a través del método. Válido para: <code>Button</code> , <code>Menu</code>
TextChangesButton	Válido para: <code>Button</code>
TextIsAnchorCommand	Para un controlador de menú, el texto del menú se toma del comando predeterminado (ancla). Un comando de anclaje es el último comando seleccionado o bloqueado. Si no se establece esta marca, <code>MenuText</code> el controlador de menú utiliza su propio campo. Sin embargo, al hacer clic en el controlador de menú todavía se habilita el último comando seleccionado de ese controlador. Se recomienda combinar esta marca <code>TextChanges</code> con la marca. Esta marca solo se aplica a los menús de tipo <code>MenuController</code> o <code>MenuControllerLatched</code>. Válido para: <code>Menu</code>
TextMenuCtrlUseMenu	Utilice <code>MenuText</code> el campo de los controladores de menú. El campo <code>ButtonText</code> predeterminado es . Válido para: <code>Button</code>
TextMenuUseButton	Utilice <code>ButtonText</code> el campo para los menús. El campo <code>MenuText</code> predeterminado es si se especifica. Válido para: <code>Button</code>
TextOnly	Mostrar solo texto en una barra de herramientas o en un menú, pero sin icono, incluso si se especifica el icono. Válido para: <code>Button</code>

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Buttons	Proporciona un grupo para <code>Button</code> elementos de elemento .
Elemento Menús	Define todos los menús que implementa un <code>VSPackage</code> .

Vea también

- [Tabla de comandos de Visual Studio \(. Vsct\) Archivos](#)

Symbols (Elemento)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Define GUID e iD que utilizan otros elementos VSCT. Para el código no administrado, esta información normalmente procede de los archivos de encabezado especificados por [Extern Element](#). El código administrado utiliza los elementos secundarios del elemento Symbols para definir esta información.

Si crea un archivo .vsct a partir de un archivo .cto existente, los símbolos se generarán como elementos secundarios del elemento Symbols. Para obtener más información, consulte [Cómo: crear un archivo . Archivo Vsct de un archivo . Cto Archivo](#).

El elemento Symbols no debe confundirse con el [elemento Define](#), que define los pares nombre-valor para su uso por el preprocesador.

Sintaxis

```
<Symbols>
  <GuidSymbol>... </GuidSymbol>
  <GuidSymbol>... </GuidSymbol>
</Symbols>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
None	

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
GuidSymbol	Define un símbolo GUID. GuidSymbol tiene dos atributos necesarios: nombre y valor. El nombre es el nombre del símbolo y el valor es el valor del GUID como una cadena. Por ejemplo: <GuidSymbol name-"guidVsPackage1Pkg" value-"c5f54698-101a-4846-84d3-dc748f9cd848-" />
IDSymbol	Define un símbolo. IDSymbol tiene dos atributos necesarios: nombre y valor. El nombre es el nombre del símbolo y el valor es el valor del símbolo como una cadena. Por ejemplo: <IDSymbol name-"MyMenuGroup" value-"0x1020" />

Elementos primarios

ELEMENTO	DESCRIPCIÓN
CommandTable (Elemento)	El elemento raíz del archivo .vsct.

Ejemplo

```
<Symbols>
  <GuidSymbol name="guidVsPackage1Pkg" value="{c5f54698-101a-4846-84d3-dc748f9cd848}" />
  <GuidSymbol name="guidVsPackage1CmdSet" value="{cb9dfd7f-2fcc-4a3e-aae8-f7fe30b1cfac}">
    <IDSymbol name="MyMenuGroup" value="0x1020" />
    <IDSymbol name="cmdidMyCommand" value="0x0100" />
    <IDSymbol name="cmdidMyTool" value="0x0101" />
  </GuidSymbol>
</Symbols>
```

Consulte también

- [Archivos de tabla de comandos de Visual Studio \(.Vsct\)](#)

Elemento GuidSymbol

06/04/2020 • 2 minutes to read • [Edit Online](#)

El `GuidSymbol` elemento contiene el GUID del par GUID:ID que representa un menú, grupo o comando. El identificador procede `IDSymbol` de `GuidSymbol` un elemento del elemento. El `GuidSymbol` elemento `name` tiene un atributo que proporciona un nombre descriptivo `value` para el GUID, que se encuentra en el atributo.

Sintaxis

```
<GuidSymbol name="guidMyCommandSet" value="{xxxxxxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx}">
  <IDSymbol>... </IDSymbol>
  <IDSymbol>... </IDSymbol>
</GuidSymbol>
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
name	Necesario. Nombre del símbolo GUID.
value	Necesario. GUID del símbolo GUID.

Elementos secundarios

ELEMENTO	DESCRIPCIÓN
Elemento IDSymbol	Contiene el identificador del par GUID:ID que representa un menú, grupo o comando.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento Símbolos	Agrupar <code>GuidSymbol</code> los elementos de un archivo <code>.vsct</code> .

Observaciones

Normalmente, un archivo `.vsct` contiene tres `GuidSymbol` elementos en su `Symbols` sección, uno para el propio paquete, otro para el conjunto de comandos (la colección de menús, grupos y comandos que el paquete pone a disposición) y otro para los mapas de bits que proporcionan iconos para botones y otros componentes visuales. Cada `IDSymbol` elemento de `GuidSymbol` un elemento `value` determinado debe tener un único. Sin `IDSymbol` embargo, los elementos que tienen valores idénticos pueden existir en un paquete siempre que tengan elementos primarios diferentes.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

Elemento IDSymbol

06/04/2020 • 2 minutes to read • [Edit Online](#)

El `IDSymbol` elemento contiene el identificador del par GUID:ID que representa un menú, grupo o comando. El GUID procede `GuidSymbol` del elemento primario. El `IDSymbol` elemento `name` tiene un atributo que proporciona un nombre descriptivo `value` para el identificador, que se encuentra en el atributo.

Sintaxis

```
<IDSymbol name=ElementName value="0x0010" />
```

Atributos y elementos

En las siguientes secciones se describen los atributos, los elementos secundarios y los elementos primarios.

Atributos

ATRIBUTO	DESCRIPCIÓN
name	Necesario. Nombre del símbolo de ID.
value	Necesario. Valor de ID numérico del símbolo de ID.

Elementos secundarios

Ninguno.

Elementos primarios

ELEMENTO	DESCRIPCIÓN
Elemento GuidSymbol	Contiene el GUID del par GUID:ID que representa un menú, grupo o comando. Agrupa los elementos <code>IDSymbol</code> .

Observaciones

Cada `IDSymbol` elemento de `GuidSymbol` un elemento `value` determinado debe tener un único. Sin `IDSymbol` embargo, los elementos que tienen valores idénticos pueden existir en un paquete siempre que tengan elementos primarios diferentes.

Vea también

- [Archivos de tabla de comandos de Visual Studio \(.vsct\)](#)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los valores 1, '1' y 'true' se evalúan como true, y 0, '0' y 'false' se evalúan como false.

Ejemplos

```
<Menu Condition="Defined(DEBUG)" ...
</Menu>

<Menu Condition="% (SKU_MODE) = 'Demo'" ...
</Menu>

<Menus Condition="Defined(DEBUG)">
  <Menu ...
  </Menu>
</Menus>

<Menus Condition="Defined(DEMO_SKU)">
  <Menus Condition="!Defined(DEBUG)">
    <Menu ...
    </Menu>
  </Menus>

  <Menu ...
  </Menu>
</Menus>

<Menus Condition="(Defined(DEMO_SKU) or Defined(SAMPLE_SKU))
and !Defined(DEBUG)">
  <Menu ...
  </Menu>
</Menus>
```

Vea también

- [Tabla de comandos de Visual Studio \(. Vsct\)](#)

GUID y constantes en el SDK de Visual Studio

06/04/2020 • 2 minutes to read • [Edit Online](#)

La [VSConstants](#) clase proporciona acceso a una selección de GUID de entorno y constantes.

En esta sección

- [GUID de IDE](#) Enumera los GUID [VSConstants](#) proporcionados por .
- [Constantes IDE](#) Enumera las constantes [VSConstants](#) de SDK proporcionadas por .

Secciones relacionadas

- [Constantes COM en código administrado](#) Enumera las constantes COM estándar disponibles para el código administrado desde Managed Package Framework (MPF).
- [Comandos definidos por IDE para ampliar los sistemas](#) de proyectos Enumera los diferentes tipos de elementos de comando utilizados para ampliar los sistemas de proyecto.

GUID de IDE

06/04/2020 • 2 minutes to read • [Edit Online](#)

La [VSConstants](#) clase publica GUID de algunas partes del entorno de desarrollo integrado (IDE) como se muestra en la tabla siguiente.

Sistemas principales

CONSTANTE	GUID
HtmDocData_guid	62C81794-A9EC-11D0-8198-00A0C91BBEE3
HtmlEditorPackage_guid	1B437D20-F8FE-11D2-A6AE-00104BCC7269
HtmlLanguageService_guid	58E975A0-F8FE-11D2-A6AE-00104BCC7269
MiscellaneousFilesProject_guid	A2FE74E1-B743-11d0-AE1A-00A0C90FFFC3
SolutionItemsProject_guid	D1DCDB85-C5E8-11d2-BFCA-00C04F990235
VsEnvironmentPackage_guid	DA9FB551-C724-11d0-AE1F-00A0C90FFFC3
HtmlEditor_guid	C76D83F8-A489-11D0-8195-00A0C91BBEE3
TextEditor_guid	8B382828-6202-11d1-8870-0000F87579D2

Componentes ampliamente visibles

CONSTANTE	GUID
VsUIHierarchyWindow_guid	7D960B07-7AF8-11D0-8E5E-00A0C911005A
ExternalEditor_guid	8137C9E8-35FE-4AF2-87B0-DE3C45F395FD
Microsoft.VisualStudio.VSConstants.SID_SUIHostCommandDis patcher	e69cd190-1276-11d1-9f64-00a0c911004f
Microsoft.VisualStudio.VSConstants.SID_SVsGeneralOutputWi ndowPane	65482c72-defa-41b7-902c-11c091889c83

Archivos, carpetas virtuales y físicas y subProyectos

CONSTANTE	GUID
PhysicalFile_guid	6bb5f8ee-4483-11d3-8bcf-00c04f8ec28c
PhysicalFolder_guid	6bb5f8ef-4483-11d3-8bcf-00c04f8ec28c

CONSTANTE	GUID
SubProject_guid	EA6618E8-6E24-4528-94BE-6889FE16485C
VirtualFolder_guid	6bb5f8f0-4483-11d3-8bcf-00c04f8ec28c

Contextos de interfaz de usuario

CONSTANTE	GUID
CodeWindow_guid	8fe2df1d-e0da-4ebe-9d5c-415d40e487b5
Debugging_guid	adfc4e61-0397-11d1-9f4e-00a0c911004f
DesignMode_guid	adfc4e63-0397-11d1-9f4e-00a0c911004f
Dragging_guid	b706f393-2e5b-49e7-9e2e-b1825f639b63
EmptySolution_guid	adfc4e65-0397-11d1-9f4e-00a0c911004f
FullScreenMode_guid	adfc4e62-0397-11d1-9f4e-00a0c911004f
NoSolution_guid	adfc4e64-0397-11d1-9f4e-00a0c911004f
SolutionBuilding_guid	adfc4e60-0397-11d1-9f4e-00a0c911004f
SolutionExists_guid	f1536ef8-92ec-443c-9ed7-fdadf150da82
SolutionHasMultipleProjects_guid	93694fa0-0397-11d1-9f4e-00a0c911004f
SolutionHasSingleProject_guid	adfc4e66-0397-11d1-9f4e-00a0c911004f

Panel de resultados

CONSTANTE	GUID
BuildOutputPane_guid	1BD8A850-02D1-11D1-BEE7-00A0C913D1F8
DebugPane_guid	FC076020-078A-11D1-A7DF-00A0C9110051
GeneralPane_guid	3C24D581-5591-4884-A571-9FE89915CD64
SortedBuildOutputPane_guid	2032B126-7C8D-48AD-8026-0E0348004FC0
StoreValidationPane_guid	54065C74-1B11-4249-9EA7-5540D1A6D528
Microsoft.VisualStudio.VSConstants.SID_SVsGeneralOutputWindowPane	65482c72-defa-41b7-902c-11c091889c83

Conjuntos de comandos y propiedades

CONSTANTE	GUID
Microsoft.VisualStudio.VSConstants.GUID_VSStandardCommandSet97	5EFC7975-14BC-11CF-9B2B-00AA00573819
Microsoft.VisualStudio.VSConstants.GUID_VsUIHierarchyWindowCmds>	60481700-078b-11d1-aaf8-00a0c9055a90

IUnknown

CONSTANTE	GUID
IID_IUnknown	00000000-0000-0000-C000-000000000046

LISTAS de tareas GUID

CONSTANTE	GUID
All	1880202e-fc20-11d2-8bb1-00c04f8ec28c
CheckedTasks	18802036-fc20-11d2-8bb1-00c04f8ec28c
CommentTasks	18802034-fc20-11d2-8bb1-00c04f8ec28c
CompilerTasks	18802033-fc20-11d2-8bb1-00c04f8ec28c
CurrentFileTasks	18802035-fc20-11d2-8bb1-00c04f8ec28c
HTMLTasks	36ac1c0d-fe86-11d2-8bb1-00c04f8ec28c
ShortcutTasks	18802030-fc20-11d2-8bb1-00c04f8ec28c
UncheckedTasks	18802037-fc20-11d2-8bb1-00c04f8ec28c
UserTasks	1880202f-fc20-11d2-8bb1-00c04f8ec28c
VsTaskList_guid	BC5955D5-aa0d-11d0-a8c5-00a0c921a4d2
VsTaskListPackage_guid	4A9B7E50-aa16-11d0-a8c5-00a0c921a4d2

GUID de página del selector de componentes

CONSTANTES	GUID
Microsoft.VisualStudio.VSConstants.GUID_COMClassicPage	9A341D96-5A64-11d3-BFF9-00C04F990235
Microsoft.VisualStudio.VSConstants.GUID_COMPlusPage	9A341D95-5A64-11d3-BFF9-00C04F990235
Microsoft.VisualStudio.VSConstants.GUID_SolutionPage	9A341D97-5A64-11d3-BFF9-00C04F990235

GUID de vaciado varios

CONSTANTES	GUID
VsCfgProviderEventsHelper_guid	99913f1f-1ee3-11d1-8a6e-00c04f682e21
VsDocOutlinePackage_guid	21af45b0-ffa5-11d0-b63f-00a0c922e851
Microsoft.VisualStudio.VSConstants.SID_SVsToolboxActiveXDataProvider	35222106-bb44-11d0-8c46-00c04fc2aae2

Vea también

- [Constantes COM en código administrado](#)
- [Constantes IDE](#)
- [Comandos definidos por IDE para ampliar los systems del proyecto](#)

Constantes IDE

06/04/2020 • 7 minutes to read • [Edit Online](#)

La [VSConstants](#) clase proporciona constantes que son específicas del entorno de desarrollo integrado (IDE) y que se definieron anteriormente solo en archivos de encabezado.

Vistas lógicas y físicas

VALUE	DESCRIPCIÓN
Code_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores deben pasar OpenStandardEditor este valor al método para obtener el cuadro de diálogo Abrir con , en este caso en posibles vistas de código.
Debugging_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores pasan OpenStandardEditor este valor al método para obtener el cuadro Debugging_guid de diálogo Abrir con , en este caso rellenado con posibles vistas de depuración que se asignan a la misma vista que Code_guid .
Designer_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores pasan OpenStandardEditor este valor al método para obtener el cuadro de diálogo Abrir con , en este caso para ver vistas del diseñador de formularios.
Primary_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores pasan OpenStandardEditor este valor al método para obtener el cuadro de diálogo Abrir con , en este caso la vista predeterminada/primaria de la fábrica del editor.
TextView_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores pasan OpenStandardEditor este valor al método para obtener el cuadro de diálogo Abrir con , en este para una vista del editor de texto de datos o documentos.
UserChooseView_guid	GUID_VSStandardCommandSet97 <code>cmdidOpenWith</code> los controladores pasan OpenStandardEditor este valor al método que solicita al usuario que elija qué vista definida por el usuario que se va a usar.

Banderas de fábrica de editor

VALUE	DESCRIPCIÓN
Cef. CloneFile	Un indicador obsoleto combinado bit a CreateEditorInstance bit como el primer parámetro del método.
Cef. OpenAsNew	Combinado bit a bit como CreateEditorInstance el primer parámetro del método, esto indica que el generador del editor debe realizar las correcciones necesarias.

VALUE	DESCRIPCIÓN
Cef. OpenFile	Combinado bit a bit como CreateEditorInstance el primer parámetro del método, este indicador es mutuamente excluyente de CEF. CloneFile .
Cef. Silencioso	Combinado bit a bit como CreateEditorInstance el primer parámetro del método, esto indica que el generador del editor debe crear el editor sin mostrar una interfaz de usuario (UI).

Errores de Visual Studio

VALUE	DESCRIPCIÓN
VS_E_BUSY	Una constante devuelta por las interfaces al comportamiento asíncrono cuando el objeto en cuestión ya está ocupado
VS_E_INCOMPATIBLEDOCDATA	Un error HRESULT que es específico de Visual Studio para "Datos de documento incompatibles".
VS_E_PACKAGENOTLOADED	Un error HRESULT que es específico de Visual Studio y que indica "Paquete no cargado."
VS_E_PROJECTALREADYEXISTS	Un error HRESULT que es específico de Visual Studio y que indica que el "Proyecto ya existe."
VS_E_PROJECTMIGRATIONFAILED	Un error HRESULT que es específico de Visual Studio y que indica "Error de configuración del proyecto."
VS_E_PROJECTNOTLOADED	Un error HRESULT que es específico de Visual Studio y que indica "Proyecto no cargado."
VS_E_SOLUTIONALREADYOPEN	Un error HRESULT que es específico de Visual Studio y que indica "Solución ya abierta."
VS_E_SOLUTIONNOTOPEN	Un error HRESULT que es específico de Visual Studio y que indica "Solución no abierta."
VS_E_SPECIFYING_OUTPUT_UNSUPPORTED	Devuelta por las interfaces de compilación que IVsOutput tienen parámetros para especificar una matriz de la interfaz, pero la implementación solo puede aplicar el método a todas las salidas.
VS_E_UNSUPPORTEDFORMAT	El CreateEditorInstance método devuelve este valor si el documento tiene un formato que no se puede abrir en el editor.
VS_E_WIZARDBACKBUTTONPRESS	Un valor HRESULT que indica que el usuario pulsa el botón Atrás en un asistente de Visual Studio.

Constantes de Visual Studio

VALUE	DESCRIPCIÓN
VS_S_PROJECTFORWARDED	Un error HRESULT que es específico de Visual Studio y que indica "Proyecto reenviado."
VS_S_TBXMARKER	Constante específica de Visual Studio para un "marcador de cuadro de herramientas".
VSM_ENTERMODAL	Constante específica de Visual Studio para transmitir un OnBroadcastMessage mensaje de notificación a través del método que indica el principio de la modalidad.
VSM_EXITMODAL	Constante específica de Visual Studio para transmitir un OnBroadcastMessage mensaje de notificación a través del método que indica el final de la modalidad.
VSM_TOOLBARMETRICSCHANGE	Constante específica de Visual Studio para transmitir un OnBroadcastMessage mensaje de notificación a través del método que indica que las métricas de la barra de comandos han cambiado.
VSCOOKIE_NIL	Constante específica de Visual Studio que indica que no se ha establecido una cookie.
VSITEMID. Nula	Identificador de elemento de Visual Studio que representa la ausencia de un elemento de proyecto. Este valor se utiliza cuando no hay ninguna selección actual.
VSITEMID. Raíz	Identificador de elemento de Visual Studio que representa la raíz de una jerarquía de proyecto y se usa para identificar toda la jerarquía, a diferencia de un único elemento.
VSITEMID. Selección	Identificador de elemento de Visual Studio que representa el elemento o elementos seleccionados actualmente, que puede incluir la raíz de la jerarquía.

IVsSelectionEvents

Describe qué componente del IDE se acaba [OnElementValueChanged](#) de seleccionar, por ejemplo, en una llamada.

CONSTANTE	VALUE
SelectionElement.DocumentFrame	0x2
SelectionElement.PropertyBrowserSID	0x4
SelectionElement.StartupProject	0x3
SelectionElement.UndoManager	0x0
SelectionElement.UserContext	0x5
SelectionElement.WindowFrame	0x1

VSSELELEMID

Constantes utilizadas para indicar un nuevo estado de selección.

CONSTANTE	VALUE
VSConstants.VSSELELEMID	2
VSConstants.VSSELELEMID	7
VSConstants.VSSELELEMID	4
VSConstants.VSSELELEMID	6
VSConstants.VSSELELEMID	3
VSConstants.VSSELELEMID	0
VSConstants.VSSELELEMID	5
VSConstants.VSSELELEMID	1

Constantes de diálogo del selector de componentes

CONSTANTE	VALUE
CPDN_SELCHANGED	WM_USER + 1280
CPDN_SELDBLCLICK	WM_USER + 1281
CPPM_CLEARSELECTION	WM_USER + 1290
CPPM_GETSELECTION	WM_USER + 1287
CPPM_INITIALIZELIST	WM_USER + 1285
CPPM_INITIALIZETAB	WM_USER + 1288
CPPM_QUERYCANSELECT	WM_USER + 1286
CPPM_SETMULTISELECT	WM_USER + 1289

Vea también

- [Comandos definidos por IDE para ampliar los sistemas de proyectos](#)

Constantes COM en código administrado

06/04/2020 • 2 minutes to read • [Edit Online](#)

Managed Package Framework (MPF) pone algunas constantes COM estándar a disposición del código administrado. Estas constantes se enumeran a continuación para su referencia.

Valores HRESULT booleanos

S_FALSE

S_OK

Valores HRESULT

E_ABORT

E_ACCESSDENIED

E_FAIL

E_HANDLE

E_INVALIDARG

E_NOINTERFACE

E_NOTIMPL

E_OUTOFMEMORY

E_PENDING

E_POINTER

E_UNEXPECTED

UNDO_E_CLIENTABORT

Errores OLE

OLE_E_ADV

OLE_E_ADVISENOTSUPPORTED

OLE_E_BLANK

OLE_E_CANT_BINDTOSOURCE

OLE_E_CANT_GETMONIKER

OLE_E_CANTCONVERT

OLE_E_CLASSDIFF

OLE_E_ENUM_NOMORE

OLE_E_INVALIDHWND

OLE_E_INVALIDRECT

OLE_E_NOCACHE

OLE_E_NOCONNECTION

OLE_E_NOSTORAGE

OLE_E_NOT_INPLACEACTIVE

OLE_E_NOTRUNNING

OLE_E_OLEVERB

OLE_E_PROMPTSAVECANCELLED

OLE_E_STATIC

OLE_E_WRONGCOMPOBJ

Errores de envío OLE

DISP_E_ARRAYISLOCKED

DISP_E_BADCALLEE

DISP_E_BADINDEX

DISP_E_BADPARAMCOUNT

DISP_E_BADVARTYPE

DISP_E_BUFFERTOOSMALL

DISP_E_DIVBYZERO

DISP_E_EXCEPTION

DISP_E_MEMBERNOTFOUND

DISP_E_NONAMEDARGS

DISP_E_NOTACOLLECTION

DISP_E_OVERFLOW

DISP_E_PARAMNOTFOUND

DISP_E_PARAMNOTOPTIONAL

DISP_E_TYEMISMATCH

DISP_E_UNKNOWNINTERFACE

DISP_E_UNKNOWNLCID

DISP_E_UNKNOWNNAME

Vea también

- [Manejo de errores y valores devueltos](#)

Complementos de control de código fuente

06/04/2020 • 4 minutes to read • [Edit Online](#)

La sección de referencia del SDK de complemento de Control de código Visual Studio fuente contiene la especificación de interfaz completa que permite integrar los sistemas de control de código fuente con . Especifica la sintaxis y la semántica de las distintas funciones y tipos de Visual Studio datos que el complemento de control de código fuente debe implementar para interactuar con el entorno de desarrollo integrado (IDE).

En esta sección

- [Funciones de API de plug-in de control de código fuente](#) Enumera las funciones que debe implementar el complemento de control de código fuente para cumplir con la API de complemento de Control de código fuente.
- Funciones de devolución de [llamada implementadas por el IDE](#) Describe las funciones que el complemento de control de código fuente utiliza para devolver información al IDE mientras se ejecutan determinados comandos.
- [Enumeradores](#) Enumera los tipos de datos del enumerador en la API de complemento de Control de código fuente que el complemento de control de código fuente debe conocer.
- [Banderas de capacidad](#) Describe las `SCC_CAP_XXX` marcas, que indican las capacidades de un proveedor.
- [Bitflags utilizados por comandos específicos](#) Enumera los indicadores que son útiles en el contexto de comandos concretos.
- [Códigos de error](#) Enumera los valores de `SCCTRN` error comunes devueltos por las funciones como .
- [Cadenas utilizadas como claves para encontrar un complemento](#) de control de código fuente Describe las claves para tener acceso al registro para buscar el complemento de control de código fuente.
- [MSSCCPRJ. Archivo SCC](#) Describe un archivo del lado cliente que contiene información opaca para el IDE, pero que usa el complemento de control de código fuente para buscar la solución o el proyecto en el control de versiones.
- [Prácticas recomendadas para implementar un complemento de control de código](#) fuente Proporciona una colección de recordatorios técnicos importantes para recordar mientras implementa la API de complemento de Control de código fuente.
- [Restricciones en longitudes](#) de cuerda Describe las limitaciones de la API de complemento de Control de código fuente en las longitudes de cadenas utilizadas en varias funciones.
- [Glosario](#) Proporciona términos útiles y sus definiciones para leer la documentación del SDK de complemento de Control de código fuente.
- [Cómo: Desactivar advertencias](#) de compatibilidad para complementos de Control de código fuente Describe cómo deshabilitar las advertencias.

Secciones relacionadas

- [Ejemplo de plug-in de control de código fuente](#) Proporciona un ejemplo de funcionalidad de complemento de control de código fuente.

- [Guía de prueba para complementos de control de código fuente](#) Describe los procedimientos de prueba relacionados con un complemento de control de código fuente.
- [Creación de un complemento de control de código fuente](#) Describe cómo crear un complemento de control de código fuente que proporciona Visual Studio funcionalidad de control de código fuente mientras se usa la interfaz de usuario (UI) del control de código fuente.
- [Referencia del SDK de Visual Studio](#) Presenta una lista de temas de referencia.

Funciones de API de complemento de control de código fuente

06/04/2020 • 7 minutes to read • [Edit Online](#)

La API de complemento de Control de código fuente proporciona las siguientes funciones, que debe implementar el complemento de control de código fuente de acuerdo con esta API. Las firmas de cada función y la semántica asociada con los indicadores de bits y otros parámetros se describen en detalle en esta referencia.

Funciones de inicialización y limpieza

FUNCIÓN	DESCRIPCIÓN
SccCloseProject	Cierra un proyecto.
SccGetCommandOptions	Solicita al usuario opciones avanzadas para el comando dado.
SccGetVersion	Devuelve la versión del complemento de control de código fuente.
SccInitialize	Inicializa el complemento de control de código fuente. Se llama una vez para cada instancia del plug-in.
SccOpenProject	Abre un proyecto.
SccSetOption	Una función genérica utilizada para establecer una amplia variedad de opciones. Cada opción <code>SCC_OPT_XXX</code> comienza con y tiene su propio conjunto definido de valores.
SccUninitialize	Se llama una vez cuando es necesario desconectar un complemento de control de código fuente.

Funciones de control de código fuente principales

FUNCIÓN	DESCRIPCIÓN
SccAdd	Agrega una matriz de archivos especificados por nombres de ruta de acceso completos al sistema de control de código fuente.
SccAddFromScc	Permite al usuario buscar archivos que ya están en el sistema de control de código fuente y, a continuación, hacer que esos archivos formen parte del proyecto actual.
SccCheckin	Comprueba una matriz de archivos.
SccCheckout	Desprotege una matriz de archivos.

FUNCIÓN	DESCRIPCIÓN
SccDiff	Muestra las diferencias entre el archivo del usuario local especificado por un nombre de ruta de acceso completo y la versión bajo control de código fuente.
SccGet	Recupera una copia de solo lectura de un conjunto de archivos.
SccGetEvents	Comprueba el estado de los archivos que <code>SccQueryInfo</code> el autor de la llamada ha preguntado (a través de).
SccGetProjPath	Hace que el complemento de control de código fuente solicite al usuario una ruta de acceso del proyecto que sea significativa para el complemento.
SccHistory	Muestra el historial de una matriz de nombres de archivo locales completos.
SccPopulateList	Examina la lista de archivos para su estado actual. Además, <code>pfnPopulate</code> utiliza la función para notificar al autor <code>nCommand</code> de la llamada cuando un archivo no coincide con los criterios para el archivo .
SccProperties	Muestra las propiedades de un archivo completo.
SccQueryInfo	Examina una lista de archivos completos para su estado actual.
SccRemove	Quita la matriz de archivos completos del sistema de control de código fuente.
SccRename	Cambia el nombre del archivo dado a un nuevo nombre en el sistema de control de código fuente.
SccRunScc	Tiene acceso a toda la gama de características del sistema de control de código fuente.
SccUncheckout	Deshace una desprotección de una matriz de archivos.

Funciones que admiten capacidad adicional (versión 1.2 de la API de complemento de Control de código fuente)

Este grupo de funciones define la funcionalidad adicional incluida en la versión 1.2 de la API de complemento de Control de código fuente. Proporcionan acceso a funciones y características de control de código fuente más avanzadas.

FUNCIÓN	DESCRIPCIÓN
SccBeginBatch	Inicia una operación por lotes.
SccCreateSubProject	Crea un subproyecto con el nombre especificado en un proyecto primario existente.

FUNCIÓN	DESCRIPCIÓN
ScDirDiff	Muestra las diferencias entre el directorio del usuario local especificado por un nombre de ruta de acceso completo y la ubicación de la base de datos de control de código fuente.
ScDirQueryInfo	Examina una lista de directorios completos para su estado actual.
ScEndBatch	Finaliza una operación por lotes.
ScGetParentProjectPath	Devuelve la ruta de acceso primaria del proyecto dado (el proyecto debe existir).
ScIsMultiCheckoutEnabled	Comprueba si se permiten varias desprotecciones en un archivo.
ScWillCreateScFile	Comprueba si el plug-in creará MSSCCPRJ. SCC.

Funciones que admiten la capacidad avanzada (versión 1.3 de la API de complemento de Control de código fuente)

Este grupo de funciones define la funcionalidad adicional incluida en la versión 1.3 de la API de complemento de Control de código fuente. Proporcionan acceso a funciones y características de control de código fuente más avanzadas.

FUNCIÓN	DESCRIPCIÓN
ScAddFilesFromSCC	Agrega una lista de archivos del control de código fuente al proyecto actual.
ScBackgroundGet	Recupera una lista de archivos del control de código fuente sin una interfaz de usuario.
ScEnumChangedFiles	Recupera una lista de archivos en el control de código fuente que son diferentes de los archivos locales.
ScGetExtendedCapabilities	Recupera indicadores que especifican capacidades extendidas admitidas por el complemento de control de código fuente.
ScGetUserOption	Recupera opciones específicas del usuario.
ScPopulateDirList	Examina una lista de directorios y archivos de un proyecto o proyectos que están bajo control de código fuente. Cada directorio y nombre de archivo encontrados se pasa a una función de devolución de llamada.
ScQueryChanges	Examina los cambios de nombre realizados en una lista de archivos. Cada nombre de archivo se pasa a una función de devolución de llamada con su estado de cambio.

Requisitos

Encabezado: scc.h

(Suministrado en el SDK de entorno común incluye carpeta, de forma predeterminada *[unidad]. [drive]*)

Vea también

- [Complementos de control de código fuente](#)
- [Creación de un complemento de control de código fuente](#)

Función SccAdd

06/04/2020 • 4 minutes to read • [Edit Online](#)

Esta función agrega nuevos archivos al sistema de control de código fuente.

Sintaxis

```
SCCRTN SccAdd(  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG       nFiles,  
    LPCSTR*    lpFileNames,  
    LPCSTR     lpComment,  
    LONG*      pfOptions,  
    LPCMDOPTS  pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos seleccionados para agregarse al `lpFileNames` proyecto actual como se indica en la matriz.

lpFileNames

[en] Matriz de nombres locales completos de los archivos que se van a agregar.

lpComment

[en] El comentario que se aplicará a todos los archivos que se van a agregar.

pfOptions

[en] Matriz de indicadores de comando, proporcionados por archivo.

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La operación de adición se realizó correctamente.
SCC_E_FILEALREADYEXISTS	El archivo seleccionado ya está bajo control de código fuente.
SCC_E_TYENOTSUPPORTED	El tipo del archivo (por ejemplo, binario) no es compatible con el sistema de control de código fuente.
SCC_E_OPNOTSUPPORTED	El sistema de control de código fuente no admite esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_NONSPECIFICERROR	Error inespecífico; añadir no realizado.
SCC_I_OPERATIONCANCELED	La operación se canceló antes de la finalización.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.
SCC_E_FILENOTEXIST	No se encontró el archivo local.

Observaciones

Los `fOptions` habituales se reemplazan aquí `pfOptions` por `LONG` una matriz, con una especificación de opción por archivo. Esto se debe a que el tipo de archivo puede variar de un archivo a un archivo.

NOTE

No es válido `SCC_FILETYPE_TEXT` especificar `SCC_FILETYPE_BINARY` ambas opciones y opciones para el mismo archivo, pero no es válido especificar ninguna. Establecer ninguno es lo `SCC_FILETYPE_AUTO` mismo que establecer , en cuyo caso el complemento de control de código fuente detecta automáticamente el tipo de archivo.

A continuación se muestra la `pfOptions` lista de indicadores utilizados en la matriz:

OPCIÓN	VALUE	SIGNIFICADO
SCC_FILETYPE_AUTO	0x00	El complemento de control de código fuente debe detectar el tipo de archivo.
SCC_FILETYPE_TEXT	0x01	Indica un archivo de texto ASCII.
SCC_FILETYPE_BINARY	0x02	Indica un tipo de archivo distinto del texto ASCII.
SCC_ADD_STORELATEST	0x04	Almacena solo la copia más reciente del archivo, sin deltas.
SCC_FILETYPE_TEXT_ANSI	0x08	Trata el archivo como texto ANSI.

OPCIÓN	VALUE	SIGNIFICADO
SCC_FILETYPE_UTF8	0x10	Trata el archivo como texto Unicode en formato UTF8.
SCC_FILETYPE_UTF16LE	0x20	Trata el archivo como texto Unicode en formato UTF16 Little Endian.
SCC_FILETYPE_UTF16BE	0x40	Trata el archivo como texto Unicode en formato UTF16 Big Endian.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccAddFilesFromSCC

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función agrega una lista de archivos del control de código fuente al proyecto abierto actualmente.

Sintaxis

```
SCCRTN SccAddFilesFromSCC(  
    LPVOID    pContext,  
    HWND      hWnd,  
    LPSTR     lpUser,  
    LPSTR     lpAuxProjPath,  
    LONG      cFiles,  
    LPCSTR*   lpFilePaths,  
    LPCSTR    lpDestination,  
    LPCSTR    lpComment,  
    LPBOOL    pbResults  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpUser

[adentro, fuera] El nombre de usuario (hasta SCC_USER_SIZE, incluido el terminador nulo).

lpAuxProjPath

[adentro, fuera] Cadena auxiliar que identifica el `SCC_PRJPATH_` proyecto (hasta SIZE, incluido el terminador nulo).

cArchivos

[en] Número de archivos `lpFilePaths` dados por .

lpFilePaths

[adentro, fuera] Matriz de nombres de archivo para agregar al proyecto actual.

lpDestination

[en] La ruta de destino donde se van a escribir los archivos.

lpComment

[en] El comentario que se aplicará a cada uno de los archivos que se van a agregar.

pbResults

[adentro, fuera] Matriz de indicadores que se establecen para indicar el éxito (no cero o TRUE) o `cFiles` error (cero o FALSE) para cada archivo (el tamaño de la matriz debe ser al menos largo).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_E_PROJNOTOPEN	El proyecto no está abierto.
SCC_E_OPNOTPERFORMED	La conexión no es al mismo proyecto especificado por <code>lpAuxProjPath</code> .
SCC_E_NOTAUTHORIZED	El usuario no está autorizado a actualizar la base de datos.
SCC_E_NONSPECIFICERROR	Error desconocido.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccAddFromScc

06/04/2020 • 5 minutes to read • [Edit Online](#)

Esta función permite al usuario buscar archivos que ya están en el sistema de control de código fuente y, posteriormente, hacer que esos archivos formen parte del proyecto actual. Por ejemplo, esta función puede obtener un archivo de encabezado común en el proyecto actual sin copiar el archivo. La matriz return `lplpFileNames` de archivos, , contiene la lista de archivos que el usuario desea agregar al proyecto IDE.

Sintaxis

```
SCCRTN SccAddFromScc (
    LPVOID    pvContext,
    HWND      hWnd,
    LPLONG    lpnFiles,
    LPCSTR**  lplpFileNames
);
```

Parámetros

`pvContext`

[en] La estructura de contexto del complemento de control de código fuente.

`hWnd`

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

`lpnFiles`

[adentro, fuera] Un búfer para el número de archivos que se van a agregar. (Esto `NULL` es si se `lp lpFileNames` va a liberar la memoria señalada por. Consulte Comentarios para obtener más información.)

`lp lpFileNames`

[adentro, fuera] Matriz de punteros a todos los nombres de archivo sin rutas de directorio. Esta matriz se asigna y libera mediante el complemento de control de código fuente. Si `lpnFiles` es `lp lpFileNames` 1 `NULL` y no es , el `lp lpFileNames` primer nombre de la matriz a la que apunta contiene la carpeta de destino.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Los archivos se han localizado correctamente y se agregaron al proyecto.
SCC_I_OPERATIONCANCELED	La operación fue cancelada sin efecto.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.

Observaciones

El IDE llama a esta función. Si el complemento de control de código fuente admite la `lpnFiles` especificación de una carpeta de `lpFileNames` destino local, el IDE pasa el número 1 y pasa el nombre de la carpeta local a .

Cuando se devuelve `ScAddFromSc` la llamada a la función, el complemento ha asignado valores a `lpnFiles` y `lpFileNames` , asignando la `lpFileNames` memoria para la matriz de nombres de archivo según sea necesario (tenga en cuenta que esta asignación reemplaza el puntero en). El complemento de control de código fuente es responsable de colocar todos los archivos en el directorio del usuario o en la carpeta de designación especificada. A continuación, el IDE agrega los archivos al proyecto IDE.

Por último, el IDE llama a `NULL` esta `lpnFiles` función una segunda vez, pasando por . Esto se interpreta como una señal especial por el plug-in de control de código fuente para liberar la memoria asignada para la matriz de nombres de archivo en `lpFileNames`` .

`lpFileNames` es `char ***` un puntero. El complemento de control de código fuente coloca un puntero a una matriz de punteros a nombres de archivo, pasando así la lista de la manera estándar para esta API.

NOTE

Las versiones iniciales de la API de VSSCI no proporcionaban una manera de indicar el proyecto de destino para los archivos agregados. Para dar cabida a `lpFileNames` esto, se ha mejorado la semántica del parámetro para convertirlo en un parámetro de entrada/salida en lugar de un parámetro de salida. Si sólo se especifica un único archivo, es `lpnFiles` decir, el valor al `lpFileNames` que se señala el valor 1, a continuación, el primer elemento de contiene la carpeta de destino. Para usar estas nuevas semánticas, `ScSetOption` el `nooption` IDE `SCC_OPT_SHARESUBPROJ` llama a la función con el parámetro establecido en . Si un complemento de control de código fuente `SCC_E_OPTNOTSUPPORTED` no admite la semántica, devuelve . Al hacerlo, se deshabilita el uso de la característica Agregar desde El control de **código fuente**. Si un complemento admite la característica **Agregar desde Control** de código fuente (`SCC_CAP_ADDFROMSCC`), debe admitir la nueva semántica y devolver `SCC_I_SHARESUBPROJOK` .

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- `ScSetOption`

Función SccBackgroundGet

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función recupera del control de código fuente cada uno de los archivos especificados sin interacción del usuario.

Sintaxis

```
SCCRTN SccBackgroundGet(  
    LPVOID pContext,  
    LONG nFiles,  
    LPCSTR* lpFileNames,  
    LONG dwFlags,  
    LONG dwBackgroundOperationID  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz.

lpFileNames

[adentro, fuera] Matriz de nombres de archivos que se van a recuperar.

NOTE

Los nombres deben ser nombres de archivo locales completos.

dwFlags

[en] Indicadores `SCC_GET_ALL` de `SCC_GET_RECURSIVE` comando (,).

dwBackgroundOperationID

[en] Un valor único asociado a esta operación.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Operación completada correctamente.

VALUE	DESCRIPCIÓN
SCC_E_BACKGROUNDGETINPROGRESS	Una recuperación en segundo plano ya está en curso (el complemento de control de código fuente debe devolver esto solo si no admite operaciones por lotes simultáneas).
SCC_I_OPERATIONCANCELED	La operación fue cancelada antes de ser completada.

Observaciones

Esta función siempre se llama en un subproceso diferente del que cargó el complemento de control de código fuente. No se espera que esta función vuelva hasta que se realiza; sin embargo, se puede llamar varias veces con varias listas de archivos, todo al mismo tiempo.

El uso `dwFlags` del argumento es el mismo que [SccGet](#).

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccGet](#)

Función SccBeginBatch

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función inicia una secuencia por lotes de operaciones de control de código fuente. El [SccEndBatch](#) se llamará para finalizar el lote. Es posible que estos lotes no estén anidados.

Sintaxis

```
SCCRTN SccBeginBatch(void);
```

Parámetros

Ninguno.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Lote de operaciones comenzó con éxito.
SCC_E_UNKNOWNERROR	Fallo inespecífico.

Observaciones

Los lotes de control de código fuente se usan para ejecutar las mismas operaciones en varios proyectos o varios contextos. Los lotes se pueden usar para eliminar cuadros de diálogo redundantes por proyecto de la experiencia del usuario durante una operación por lotes. La `SccBeginBatch` función y [SccEndBatch](#) se utilizan como un par de funciones para indicar el principio y el final de una operación. No se pueden anidar. `SccBeginBatch` establece una marca que indica que una operación por lotes está en curso.

Mientras una operación por lotes está en vigor, el complemento de control de código fuente debe presentar como máximo un cuadro de diálogo para cualquier pregunta al usuario y aplicar la respuesta de ese cuadro de diálogo en todas las operaciones posteriores.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccEndBatch](#)

Función SccCheckin

06/04/2020 • 4 minutes to read • [Edit Online](#)

Esta función comprueba los archivos previamente desprotegidos en el sistema de control de código fuente, almacenando los cambios y creando una nueva versión. Esta función se llama con un recuento y una matriz de nombres de los archivos que se van a proteger.

Sintaxis

```
SCCRTN SccCheckin (  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG      nFiles,  
    LPSTR*     lpFileNames,  
    LPCSTR     lpComment,  
    LONG       fOptions,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento SCC puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos seleccionados para registrarse.

lpFileNames

[en] Matriz de nombres de ruta local completos de los archivos que se van a proteger.

lpComment

[en] Comentario que se aplicará a cada uno de los archivos seleccionados que se van a registrar. Este parámetro `NULL` es si el complemento de control de código fuente debe solicitar un comentario.

fOptions

[en] Marcas de comando, `SCC_KEEP_CHECKEDOUT` 0 o .

pvOptions

[en] Opciones específicas del complemento SCC.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El archivo se ha registrado correctamente.
SCC_E_FILENOTCONTROLLED	El archivo seleccionado no está bajo control de código fuente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Fallo inespecífico. El archivo no se registró.
SCC_E_NOTCHECKEDOUT	El usuario no ha traído el archivo, por lo que no puede registrarlo.
SCC_E_CHECKINCONFLICT	No se pudo realizar el registro porque: - Otro usuario se <code>bAutoReconcile</code> ha registrado por delante y era falso. - o bien - La fusión automática no se puede hacer (por ejemplo, cuando los archivos son binarios).
SCC_E_VERIFYMERGE	El archivo se ha fusionado automáticamente, pero no se ha registrado en pendiente de verificación del usuario.
SCC_E_FIXMERGE	El archivo se ha fusionado automáticamente, pero no se ha registrado debido a un conflicto de combinación que debe resolverse manualmente.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_I_OPERATIONCANCELED	La operación se canceló antes de la finalización.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.
SCC_E_FILENOTEXIST	No se encontró el archivo local.

Observaciones

El comentario se aplica a todos los archivos que se están protegiendo. El argumento `comment` `null` puede ser una cadena, en cuyo caso el complemento de control de código fuente puede solicitar al usuario una cadena de comentario para cada archivo.

Al `fOptions` argumento se le puede `SCC_KEEP_CHECKEDOUT` dar un valor de la marca para indicar la intención del usuario de proteger el archivo y volver a desprotegerlo.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccCheckout

06/04/2020 • 2 minutes to read • [Edit Online](#)

Dada una lista de nombres de archivo completos, esta función los desprotege en la unidad local. El comentario se aplica a todos los archivos que se están desprotegiendo. El argumento comment `null` puede ser una cadena.

Sintaxis

```
SCCRTN SccCheckout (  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG       nFiles,  
    LPCSTR*    lpFileNames,  
    LPCSTR     lpComment,  
    LONG       fOptions,  
    LPCMDOPTS  pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos seleccionados para ser desprotegidos.

lpFileNames

[en] Matriz de nombres de ruta local completos de los archivos que se van a desproteger.

lpComment

[en] Comentario que se aplicará a cada uno de los archivos seleccionados que se van a desproteger.

fOptions

[en] Marcas de comandos (consulte [Bitflags utilizados por comandos específicos](#)).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La salida tuvo éxito.
SCC_E_FILENOTCONTROLLED	El archivo seleccionado no está bajo control de código fuente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_NONSPECIFICERROR	Fallo inespecífico. El archivo no se ha traído.
SCC_E_ALREADYCHECKEDOUT	El usuario ya tiene el archivo desprotegido.
SCC_E_FILEISLOCKED	El archivo está bloqueado, lo que prohíbe la creación de nuevas versiones.
SCC_E_FILEOUTEXCLUSIVE	Otro usuario ha realizado una desprotección exclusiva en este archivo.
SCC_I_OPERATIONCANCELED	La operación se canceló antes de la finalización.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Bitflags utilizados por comandos específicos](#)

Función SccCloseProject

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función cierra un proyecto, marcando el final de una sesión determinada.

Sintaxis

```
SCCRTN SccCloseProject (  
    LPVOID pvContext  
);
```

Parámetros

pvContext La estructura de contexto del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El proyecto se cerró con éxito.
SCC_E_PROJNOTOPEN	Actualmente no hay ningún proyecto abierto.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

El [SccOpenProject](#) siempre se llama antes de esta función. A continuación, una llamada a esta función va seguida de una llamada a la `SccOpenProject` función o a [SccUninitialize](#), que finaliza completamente la conexión al sistema de control de código fuente.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccOpenProject](#)
- [SccInitialize](#)

Función SccCreateSubProject

06/04/2020 • 7 minutes to read • [Edit Online](#)

Esta función crea un subproyecto con el nombre especificado `lpParentProjPath` en un proyecto primario existente especificado por el argumento.

Sintaxis

```
SCCRTN SccCreateSubProject(  
    LPVOID pContext,  
    HWND hWnd,  
    LPSTR lpUser,  
    LPCSTR lpParentProjPath,  
    LPCSTR lpSubProjName,  
    LPSTR lpAuxProjPath,  
    LPSTR lpSubProjPath  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpUser

[adentro, fuera] El nombre de usuario (hasta SCC_USER_SIZE, incluido el terminador NULL).

lpParentProjPath

[en] Cadena que identifica la ruta de acceso del proyecto primario (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpSubProjName

[en] El nombre del subproyecto sugerido (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpAuxProjPath

[adentro, fuera] Cadena auxiliar que identifica el proyecto (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpSubProjPath

[adentro, fuera] Cadena de salida que identifica la ruta de acceso del subproyecto (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El subproyecto se creó correctamente.
SCC_E_INITIALIZEFAILED	No se pudo inicializar el proyecto primario.
SCC_E_INVALIDUSER	El usuario no pudo iniciar sesión en el sistema de control de código fuente.
SCC_E_COULDNOTCREATEPROJECT	No se puede crear un subproyecto.
SCC_E_PROJSYNTAXERR	Sintaxis de proyecto no válida.
SCC_E_UNKNOWNPROJECT	El proyecto primario es desconocido para el complemento de control de código fuente.
SCC_E_INVALIDFILEPATH	Ruta de acceso de archivo no válida o inutilizable.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_CONNECTIONFAILURE	Se ha producido un problema de conexión de complemento de control de código fuente.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.
SCC_E_UNKNOWNERROR	

Observaciones

Si ya existe un subproyecto con el nombre, la función puede cambiar el <nombre predeterminado para crear uno único, por ejemplo, añadiéndole "_ number>". El autor de la llamada `1pUser` `1pSubProjPath` debe `1pAuxProjPath` estar preparado para aceptar cambios en , , y . A `1pSubProjPath` `1pAuxProjPath` continuación, los argumentos y se usan en una llamada a [SccOpenProject](#). No deben ser modificados por el autor de la llamada al regresar. Estas cadenas proporcionan una manera para que el complemento de control de código fuente realice un seguimiento de la información que necesita asociar a un proyecto. El IDE del autor de la llamada no mostrará estos dos parámetros al devolverlos, porque el complemento puede usar una cadena con formato que podría no ser adecuada para su visualización. La función devuelve un código de éxito o `1pSubProjPath` error y, si se realiza correctamente, rellena la variable con la ruta de acceso completa del proyecto al nuevo proyecto.

Esta función es similar a [SccGetProjPath](#), excepto que crea de forma silenciosa un proyecto en lugar de solicitar al usuario que seleccione uno. Cuando `SccCreateSubProject` se llama `1pParentProjName` a `1pAuxProjPath` la función, y no estará vacía y corresponderá a un proyecto válido. Normalmente, el IDE recibe estas cadenas `SccGetProjPath` de una llamada anterior a la función o [sccGetParentProjectPath](#).

El `1pUser` argumento es el nombre de usuario. El IDE pasará el mismo nombre de usuario `SccGetProjPath` del que había recibido anteriormente , y el complemento de control de código fuente debe usar el nombre como valor predeterminado. Si el usuario ya tiene una conexión abierta con el complemento, el complemento debe intentar eliminar cualquier solicitud para asegurarse de que la función funciona en silencio. Sin embargo, si se produce un error en el inicio de sesión, el complemento debe `1pUser` solicitar al usuario un inicio de sesión y, cuando reciba un inicio de sesión válido, vuelva a devolver el nombre en . Dado que el complemento puede cambiar esta cadena,

el IDE siempre asignará un búfer de tamaño (SCC_USER_LEN+1 o SCC_USER_SIZE, que incluye espacio para el terminador nulo). Si se cambia la cadena, la nueva cadena debe ser un nombre de inicio de sesión válido (al menos tan válido como la cadena anterior).

Notas técnicas para SccCreateSubProject y SccGetParentProjectPath

Agregar soluciones y proyectos al control de código fuente se ha simplificado en Visual Studio para minimizar el número de veces que se solicita a un usuario que seleccione ubicaciones en el sistema de control de código fuente. Visual Studio activa estos cambios si un complemento de control `SccCreateSubProject` de `SccGetParentProjectPath` código fuente admite ambas funciones nuevas y . Sin embargo, la siguiente entrada del Registro se puede usar para deshabilitar estos cambios y revertir al comportamiento anterior de Visual Studio (Source Control Plug-in API Version 1.1):

**[HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8.0, SourceControl]
"DoNotCreateSolutionRootFolderInSourceControl"-dword:00000001**

Si esta entrada del Registro no existe o se establece en dword:00000000, Visual Studio intenta usar las nuevas funciones `SccCreateSubProject` y `SccGetParentProjectPath` .

Si la entrada del Registro se establece en dword:00000001, Visual Studio no intenta usar estas nuevas funciones y las operaciones de agregar al control de código fuente funcionan como lo hacían en versiones anteriores de Visual Studio.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccGetParentProjectPath](#)
- [SccGetProjPath](#)

Función SccDiff

06/04/2020 • 5 minutes to read • [Edit Online](#)

Esta función muestra (o, opcionalmente, solo comprueba) las diferencias entre el archivo actual (en el disco local) y su última versión protegida en el sistema de control de código fuente.

Sintaxis

```
SCCRTN SccDiff(  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LPCSTR    lpFileName,  
    LONG       fOptions,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpFileName

[en] Nombre de archivo para el que se solicita la diferencia.

fOptions

[en] Banderas de mando. Para obtener información detallada, vea la sección Comentarios de.

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La copia de trabajo y la versión del servidor son idénticas.
SCC_I_FILESDIFFERS	La copia de trabajo difiere de la versión bajo control de código fuente.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.
SCC_E_FILENOTCONTROLLED	El archivo no está bajo control de código fuente.

VALUE	DESCRIPCIÓN
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Error inespecífico; no se obtuvo la diferencia de archivo.
SCC_E_FILENOTEXIST	No se encontró el archivo local.

Observaciones

Esta función sirve para dos propósitos diferentes. De forma predeterminada, muestra una lista de cambios en un archivo. El complemento de control de código fuente abre su propia ventana, en el formato que elija, para mostrar las diferencias entre el archivo del usuario en el disco y la última versión del archivo bajo control de código fuente.

Como alternativa, el IDE puede simplemente necesitar determinar si un archivo ha cambiado. Por ejemplo, el IDE puede necesitar determinar si es seguro desproteger un archivo sin informar al usuario. En ese caso, el IDE `SCC_DIFF_CONTENTS` pasa en la bandera. El complemento de control de código fuente debe comprobar el archivo en el disco, byte a byte, con el archivo controlado por código fuente y devolver un valor que indica si los dos archivos son diferentes sin mostrar nada al usuario.

Como optimización del rendimiento, el complemento de control de código fuente puede utilizar una alternativa basada en `SCC_DIFF_CONTENTS` una suma de comprobación o una marca de tiempo en lugar de la comparación byte a byte que pide: estas formas de comparación son obviamente más rápidas pero menos confiables. No todos los sistemas de control de código fuente pueden admitir estos métodos de comparación alternativos, y el complemento puede tener que recurrir a una comparación de contenido. Todos los complementos de control de código fuente deben admitir, como mínimo, una comparación de contenido.

NOTE

Las banderas de diferencia rápida son mutuamente excluyentes. Es válido pasar ninguna marca, pero no es válido pasar simultáneamente más de uno. `SCC_DIFF_QUICK_DIFF`, que es una máscara que combina todas las marcas, se puede utilizar para probar, pero nunca se debe pasar como parámetro.

FOPTION	SIGNIFICADO
SCC_DIFF_IGNORECASE	Comparación sin distinción entre mayúsculas y minúsculas (se puede utilizar para diferencias visuales o rápidas).
SCC_DIFF_IGNORESPACE	Ignora el espacio en blanco (se puede utilizar para la diferencia rápida o visual).
SCC_DIFF_QD_CONTENTS	Compara silenciosamente el archivo, byte a byte.
SCC_DIFF_QD_CHECKSUM	Compara el archivo de forma silenciosa a través de una suma de comprobación cuando se admite. Si no se admite, vuelve a una comparación de contenido.

FOPTION	SIGNIFICADO
SCC_DIFF_QD_TIME	Compara silenciosamente el archivo a través de su marca de tiempo cuando se admite. Si no se admite, vuelve a una comparación de contenido.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccDirDiff

06/04/2020 • 4 minutes to read • [Edit Online](#)

Esta función muestra las diferencias entre el directorio local actual en el disco cliente y el proyecto correspondiente bajo control de código fuente.

Sintaxis

```
SCCRTN SccDirDiff(  
    LPVOID    pContext,  
    HWND      hWnd,  
    LPCSTR    lpDirName,  
    LONG      dwFlags,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpDirName

[en] Ruta de acceso completa al directorio local para el que se muestra una diferencia visual.

dwFlags

[en] Indicadores de comando (consulte la sección Comentarios).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El directorio del disco es el mismo que el proyecto en el control de código fuente.
SCC_I_FILESDIFFER	El directorio del disco es diferente del proyecto en el control de código fuente.
SCC_I_RELOADFILE	Es necesario volver a cargar un archivo o proyecto.

VALUE	DESCRIPCIÓN
SCC_E_FILENOTCONTROLLED	El directorio no está bajo control de código fuente.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR SCC_E_UNKNOWNERROR	Fallo inespecífico.
SCC_E_FILENOTEXIST	No se pudo encontrar el directorio local.

Observaciones

Esta función se utiliza para indicar al complemento de control de código fuente que muestre al usuario una lista de cambios en un directorio especificado. El complemento abre su propia ventana, en un formato de su elección, para mostrar las diferencias entre el directorio del usuario en el disco y el proyecto correspondiente bajo control de versión.

Si un complemento admite la comparación de directorios en absoluto, debe admitir la comparación de directorios en una base de nombre de archivo, incluso si no se admiten las opciones "quick-diff".

DWFLAGS	INTERPRETACIÓN
SCC_DIFF_IGNORECASE	Comparación sin distinción entre mayúsculas y minúsculas (se puede utilizar para diferencias rápidas o visuales).
SCC_DIFF_IGNORESPACE	Ignora el espacio en blanco (se puede utilizar para difusión rápida o visual).
SCC_DIFF_QD_CONTENTS	Si es compatible con el complemento de control de código fuente, compara silenciosamente el directorio, byte a byte.
SCC_DIFF_QD_CHECKSUM	Si es compatible con el complemento, compara de forma silenciosa el directorio a través de una suma de comprobación o, si no se admite, vuelve a SCC_DIFF_QD_CONTENTS.
SCC_DIFF_QD_TIME	Si es compatible con el complemento, compara de forma silenciosa el directorio a través de su marca de tiempo o, si no se admite, recae en SCC_DIFF_QD_CHECKSUM o SCC_DIFF_QD_CONTENTS.

NOTE

Esta función utiliza los mismos indicadores de comando que [SccDiff](#). Sin embargo, un complemento de control de código fuente puede optar por no admitir la operación "quick-diff" para directorios.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccDirQueryInfo

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función examina una lista de directorios completos para su estado actual.

Sintaxis

```
SCCRTN SccDirQueryInfo(  
LPVOID pContext,  
LONG nDirs,  
LPCSTR* lpDirNames,  
LPLONG lpStatus  
);
```

Parámetros

pContext

[en] La estructura de contexto del complemento de control de código fuente.

nDirs

[en] El número de directorios seleccionados para ser consultados.

lpDirNames

[en] Matriz de rutas de acceso completas de los directorios que se van a consultar.

lpStatus

[adentro, fuera] Una estructura de matriz para el complemento de control de código fuente para devolver los indicadores de estado (consulte Código de estado de [directorio](#) para obtener más información).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La consulta se realizó correctamente.
SCC_E_OPNOTSUPPORTED	El sistema de control de código fuente no admite esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.
SCC_E_UNKNOWNERROR	

Observaciones

La función rellena la matriz return con `SCC_DIRSTATUS` una máscara de bits de bits de la familia (consulte Código de estado de [directorio](#)), una entrada para cada directorio dado. El autor de la llamada asigna la matriz de estado.

El IDE usa esta función antes de cambiar el nombre de un directorio para comprobar si el directorio está bajo control de código fuente consultando si tiene un proyecto correspondiente. Si el directorio no está bajo control de código fuente, el IDE puede proporcionar la advertencia adecuada al usuario.

NOTE

Si un complemento de control de código fuente elige no implementar uno o varios de los valores de estado, los bits no implementados deben establecerse en cero.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Código de estado del directorio](#)

Función SccEndBatch

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función concluye un lote de operaciones de control de código fuente. Es posible que estos lotes no estén anidados.

Sintaxis

```
SCCRTN SccEndBatch(void);
```

Parámetros

Ninguno.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Lote de operaciones concluidas con éxito.
SCC_E_UNKNOWNERROR	Fallo inespecífico.

Observaciones

Los lotes de control de código fuente se usan para ejecutar las mismas operaciones de control de código fuente en varios proyectos o varios contextos. Los lotes se pueden usar para eliminar cuadros de diálogo redundantes de la experiencia del usuario durante una operación por lotes. El [SccBeginBatch](#) `SccEndBatch` y la función se utilizan como un par para indicar el principio y el final de una operación. No se pueden anidar.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccBeginBatch](#)

Función SccEnumChangedFiles

06/04/2020 • 2 minutes to read • [Edit Online](#)

Dada una lista de archivos locales, esta función determina qué archivos son diferentes de las versiones correspondientes en la base de datos de control de código fuente.

Sintaxis

```
SCCRTN SccEnumChangedFiles(  
    LPVOID pContext,  
    HWND hWnd,  
    LONG cFiles,  
    LPCSTR* lpFileNames,  
    LONG* plIsFileDifferent  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

cArchivos

[en] Número de nombres de `lpFileNames` archivo especificados en la matriz. También especifica el `plIsFileDifferent` tamaño de la matriz.

lpFileNames

[en] Matriz de nombres de archivo locales que se pueden comprobar.

plIsFileDifferent

[adentro, fuera] Matriz de valores que indica el estado de diferencia `cFiles` de cada archivo (la matriz debe tener al menos entradas). Distinto de cero significa que el archivo es diferente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Operación completada correctamente.
SCC_UNSPECIFIEDERROR	Error genérico.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)

Función SccGet

06/04/2020 • 7 minutes to read • [Edit Online](#)

Esta función recupera una copia de uno o más archivos para ver los archivos, pero no para editarlos. En la mayoría de los sistemas, los archivos se etiquetan como de solo lectura.

Sintaxis

```
SCCRTN SccGet(  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG       nFiles,  
    LPCSTR*    lpFileNames,  
    LONG       fOptions,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz.

lpFileNames

[en] Matriz de nombres completos de los archivos que se van a recuperar.

fOptions

[en] Indicadores `SCC_GET_ALL` de `SCC_GET_RECURSIVE` comando (,).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El éxito de conseguir la operación.
SCC_E_FILENOTCONTROLLED	El archivo no está bajo control de código fuente.

VALUE	DESCRIPCIÓN
SCC_E_OPNOTSUPPORTED	El sistema de control de código fuente no admite esta operación.
SCC_E_FILEISCHECKEDOUT	No se puede obtener el archivo que el usuario ha traído actualmente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NOSPECIFIEDVERSION	Se ha especificado una versión no válida o una fecha/hora.
SCC_E_NONSPECIFICERROR	Error inespecífico; archivo no se sincronizó.
SCC_I_OPERATIONCANCELED	Operación cancelada antes de la finalización.
SCC_E_NOTAUTHORIZED	El usuario no está autorizado para realizar esta operación.

Observaciones

Esta función se llama con un recuento y una matriz de nombres de los archivos que se van a recuperar. Si el IDE pasa `SCC_GET_ALL` la marca, `lpFileNames` esto significa que los elementos no son archivos, sino directorios, y que se deben recuperar todos los archivos bajo control de código fuente en los directorios especificados.

La `SCC_GET_ALL` marca se puede `SCC_GET_RECURSIVE` combinar con la marca para recuperar todos los archivos en los directorios dados y todos los subdirectorios también.

NOTE

`SCC_GET_RECURSIVE` nunca debe `SCC_GET_ALL` pasarse sin . Además, tenga en cuenta que si los directorios `C:-A` y `C:-A-B` se pasan en un get recursivo, `C:-A-B` y todos sus subdirectorios se recuperarán realmente dos veces. Es responsabilidad del IDE, y no del complemento de control de código fuente, asegurarse de que los duplicados como este se mantengan fuera de la matriz.

Por último, incluso si un complemento `SCC_CAP_GET_NOUI` de control de código fuente especificó la marca en la inicialización, lo que indica que no tiene una interfaz de usuario para un Get comando, esta función todavía puede ser llamada por el IDE para recuperar archivos. La marca simplemente significa que el IDE no muestra un get elemento de menú y que no se espera que el complemento proporcione ninguna interfaz de usuario.

Cambiar el nombre de los archivos y SccGet

Situación: un usuario desprotege un archivo, por ejemplo, *a.txt* y lo modifica. Antes de que se pueda proteger *a.txt* un segundo usuario cambia el nombre de *a.txt* a *b.txt* en la base de datos de control de código fuente, desprotege *b.txt*, realiza algunas modificaciones en el archivo y protege el archivo. El primer usuario desea los cambios realizados por el segundo usuario para que el primer usuario cambie el nombre de su versión local del archivo *a.txt* a *b.txt* y realice una obtención en el archivo. Sin embargo, la memoria caché local que realiza un seguimiento de los números de versión sigue pensando que la primera versión de *a.txt* se almacena localmente y, por lo tanto, el control de código fuente no puede resolver las diferencias.

Hay dos maneras de resolver esta situación en la que la memoria caché local de las versiones de control de código fuente no está sincronizada con la base de datos de control de código fuente:

1. No permita cambiar el nombre de un archivo en la base de datos de control de código fuente que está actualmente desprotegida.
2. Haga el equivalente de "eliminar antiguo" seguido de "añadir nuevo". El siguiente algoritmo es una manera de lograr esto.
 - a. Llame a la función [SccQueryChanges](#) para obtener información sobre el cambio de nombre de *a.txt* a *b.txt* en la base de datos de control de código fuente.
 - b. Cambie el nombre del archivo local *a.txt* a *b.txt*.
 - c. Llame `SccGet` a la función para *a.txt* y *b.txt*.
 - d. Dado que *a.txt* no existe en la base de datos de control de código fuente, la caché de la versión local se purga de la información de versión *a.txt* que falta.
 - e. El archivo *b.txt* que se desprotege se combina con el contenido del archivo *b.txt* local.
 - f. El archivo *b.txt* actualizado ahora se puede registrar.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Bitflags utilizados por comandos específicos](#)

Función SccGetCommandOptions

06/04/2020 • 6 minutes to read • [Edit Online](#)

Esta función solicita al usuario opciones avanzadas para un comando determinado.

Sintaxis

```
SCCRTN SccGetCommandOptions(  
    LPVOID pvContext,  
    HWND hWnd,  
    enum SCCCOMMAND iCommand,  
    LPCMDOPTS* ppvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

iCommand

[en] El comando para el que se solicitan opciones avanzadas (consulte [Código de comando](#) para ver los valores posibles).

ppvOptions

[en] La estructura de opciones `NULL` (también puede ser).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Correcto.
SCC_I_ADV_SUPPORT	El complemento de control de código fuente admite opciones avanzadas para el comando.
SCC_I_OPERATIONCANCELED	El usuario canceló el cuadro de diálogo Opciones del complemento de control de código fuente.
SCC_E_OPTNOTSUPPORTED	El complemento de control de código fuente no admite esta operación.
SCC_E_ISCHECKEDOUT	No se puede realizar esta operación en un archivo que está actualmente desprotegido.

VALUE	DESCRIPCIÓN
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

El IDE llama a esta `ppvOptions = NULL` función por primera vez con para determinar si el complemento de control de código fuente admite la característica de opciones avanzadas para el comando especificado. Si el complemento admite la característica para ese comando, el IDE llama a esta función de nuevo cuando el usuario solicita opciones avanzadas `ppvOptions` (normalmente `NULL` implementadas como un botón **Avanzado** en un cuadro de diálogo) y proporciona un puntero no NULL para que apunte a un puntero. El complemento almacena las opciones avanzadas especificadas por el usuario en una `ppvOptions` estructura privada y devuelve un puntero a esa estructura en . A continuación, esta estructura se pasa a todas las demás funciones de `ScGetCommandOptions` la API de complemento de Control de código fuente que necesitan conocerla, incluidas las llamadas posteriores a la función.

Un ejemplo puede ayudar a aclarar esta situación.

Un usuario elige el **Get** comando y el IDE muestra un **Obtener** cuadro de diálogo. El IDE `ScGetCommandOptions` llama `iCommand` a `SCC_COMMAND_GET` la `ppvOptions` función con establecido en y establecido en `NULL`. Esto es interpretado por el complemento de control de código fuente como la pregunta, "¿Tiene alguna opción avanzada para este comando?" Si el complemento `SCC_I_ADV_SUPPORT` devuelve , el IDE muestra un botón **Avanzado** en el cuadro de diálogo **Obtener**.

La primera vez `NULL` `ppvOptions` que el usuario hace clic en `ScGetCommandOptions` el avanzado botón, el IDE vuelve a llamar a la función, esta vez con un no que apunta a un `NULL` puntero. El complemento muestra su propio cuadro de diálogo **Obtener opciones**, solicita al usuario información, coloca esa información `ppvOptions` en su propia estructura y devuelve un puntero a esa estructura en .

Si el usuario vuelve a hacer clic en **Avanzado** `ScGetCommandOptions` en el `ppvOptions` mismo cuadro de diálogo, el IDE vuelve a llamar a la función sin cambiar, para que la estructura se devuelva al complemento. Esto permite que el complemento reinicie su cuadro de diálogo en los valores que el usuario había establecido previamente. El complemento modifica la estructura en su lugar antes de volver.

Por último, cuando el usuario hace clic en **Aceptar** en el Cuadro de diálogo **Obtener** del `ppvOptions` IDE, el IDE llama a `ScGet`, pasando la estructura devuelta que contiene las opciones avanzadas.

NOTE

El `SCC_COMMAND_OPTIONS` comando se utiliza cuando el IDE muestra un cuadro de diálogo **Opciones** que permite al usuario establecer preferencias que controlan cómo funciona la integración. Si el complemento de control de código fuente desea proporcionar su propio cuadro de diálogo de preferencias, puede mostrarlo desde un botón **Avanzado** en el cuadro de diálogo de preferencias del IDE. El plug-in es el único responsable de obtener y conservar esta información; el IDE no lo usa ni lo modifica.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Código de comando](#)

Función SccGetEvents

06/04/2020 • 3 minutes to read • [Edit Online](#)

Esta función recupera un evento de estado en cola.

Sintaxis

```
SCCRTN SccGetEvents (  
    LPVOID pvContext,  
    LPSTR lpFileName,  
    LPLONG lpStatus,  
    LPLONG pnEventsRemaining  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

lpFileName

[adentro, fuera] Zona de influencia donde el complemento de control de código fuente coloca el nombre de archivo devuelto (hasta _MAX_PATH caracteres).

lpStatus

[adentro, fuera] Devuelve el código de estado (consulte Código de [estado de archivo](#) para los valores posibles).

pnEventsRemaining

[adentro, fuera] Devuelve el número de entradas que quedan en la cola después de esta llamada. Si este número es grande, el autor de la llamada puede decidir llamar a [SccQueryInfo](#) para obtener toda la información a la vez.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Obtener eventos realizados correctamente.
SCC_E_OPNOTSUPPORTED	Esta función no se admite.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

Esta función se llama durante el procesamiento inactivo para ver si ha habido actualizaciones de estado para los archivos bajo control de código fuente. El complemento de control de código fuente mantiene el estado de todos los archivos que conoce y, siempre que el complemento anota un cambio de estado, el estado y el archivo asociado se almacenan en una cola. Cuando `SccGetEvents` se llama, se recupera y devuelve el elemento superior de la cola.

Esta función está restringida a devolver solo información almacenada previamente en caché y debe tener un cambio de velocidad muy rápido (es decir, sin leer el disco o pedir el estado del sistema de control de código fuente); de lo contrario, el rendimiento del IDE puede comenzar a degradarse.

Si no hay ninguna actualización de estado para informar, el complemento de `lpFileName` control de código fuente almacena una cadena vacía en el búfer al que apunta . De lo contrario, el complemento almacena el nombre completo de la ruta de acceso del archivo para el que ha cambiado la información de estado y devuelve el código de estado adecuado (uno de los valores detallados en Código de [estado de archivo](#)).

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Código de estado del archivo](#)

Función SccGetExtendedCapabilities

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función devuelve capacidades adicionales admitidas por el complemento de control de código fuente.

Sintaxis

```
SCCRTN SccGetExtendedCapabilities(  
    LPVOID pContext,  
    LONG lSccExCaps,  
    LPBOOL pbSupported  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

lSccExCaps

[en] Indicador que especifica una capacidad extendida para la que se va a probar (consulte la tabla Código de capacidad ampliada en [indicadores de capacidad](#) para los indicadores posibles).

pbSupported

[fuera] Devuelve distinto de `TRUE` cero () si se admite la capacidad especificada; de lo contrario, devuelve cero (`FALSE`).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La operación get capability se ha completado correctamente.
SCC_E_UNKNOWNERROR	Error desconocido o no especificado.
SCC_E_NONSPECIFICERROR	

Observaciones

Este método se llama a petición; es decir, cuando es necesario probar una capacidad, se llama a este método para determinar si se admite esa capacidad. Solo se especifica una marca a la vez.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [Códigos de error](#)
- [Banderas de capacidad](#)

Función SccGetParentProjectPath

06/04/2020 • 7 minutes to read • [Edit Online](#)

Esta función determina la ruta de acceso del proyecto primario de un proyecto especificado. Esta función se llama cuando el usuario está agregando un proyecto de Visual Studio al control de código fuente.

Sintaxis

```
SCCRTN SccGetParentProjectPath(  
    LPVOID pContext,  
    HWND hWnd,  
    LPSTR lpUser,  
    LPCSTR lpProjPath,  
    LPSTR lpAuxProjPath,  
    LPSTR lpParentProjPath  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpUser

[adentro, fuera] El nombre de usuario (hasta SCC_USER_SIZE, incluido el terminador NULL).

lpProjPath

[en] Cadena que identifica la ruta de acceso del proyecto (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpAuxProjPath

[adentro, fuera] Cadena auxiliar que identifica el proyecto (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpParentProjPath

[adentro, fuera] Cadena de salida que identifica la ruta de acceso del proyecto primario (hasta SCC_PRJPATH_SIZE, incluido el terminador NULL).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La ruta del proyecto principal se obtuvo correctamente.

VALUE	DESCRIPCIÓN
SCC_E_INITIALIZEFAILED	No se pudo inicializar el proyecto.
SCC_E_INVALIDUSER	El usuario no pudo iniciar sesión en el complemento de control de código fuente.
SCC_E_UNKNOWNPROJECT	Project es desconocido para el complemento de control de código fuente.
SCC_E_INVALIDFILEPATH	Ruta de acceso de archivo no válida o inutilizable.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_PROJSYNTAXERR	Sintaxis de proyecto no válida.
SCC_E_CONNECTIONFAILURE	Problema de conexión de almacén.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.
SCC_E_UNKNOWNERROR	

Observaciones

Esta función devuelve un código de éxito o `1pParentProjPath` error y, si se realiza correctamente, rellena la variable con la ruta de acceso completa del proyecto al proyecto especificado.

Esta función devuelve la ruta de acceso del proyecto primario de un proyecto existente. Para el proyecto raíz, la función devuelve la ruta de acceso del proyecto que se pasó (es decir, la misma ruta de acceso del proyecto raíz). Tenga en cuenta que una ruta de acceso del proyecto es una cadena que solo es significativa para el complemento de control de código fuente.

El IDE está preparado para `1pUser` `1pAuxProjPath` aceptar cambios en los parámetros y. El IDE conservará estas cadenas y las pasará a [SccOpenProject](#) cuando el usuario abra este proyecto en el futuro. Estas cadenas, por lo tanto, proporcionan una manera para que el complemento de control de código fuente realice un seguimiento de la información que necesita asociar a un proyecto.

Esta función es similar a [SccGetProjPath](#), excepto que no solicita al usuario que seleccione un proyecto. Tampoco crea un nuevo proyecto, pero solo funciona con un proyecto existente.

Cuando `SccGetParentProjectPath` se `1pProjPath` llama, y `1pAuxProjPath` no estará vacío y corresponderá a un proyecto válido. Normalmente, el IDE recibe estas cadenas `SccGetProjPath` de una llamada anterior a la función.

El `1pUser` argumento es el nombre de usuario. El IDE pasará el mismo nombre de usuario `SccGetProjPath` que había recibido previamente de la función y el complemento de control de código fuente debe usar el nombre como valor predeterminado. Si el usuario ya tiene una conexión abierta con el complemento, el complemento debe intentar eliminar cualquier solicitud para asegurarse de que la función funciona en silencio. Sin embargo, si se produce un error en el inicio de sesión, el complemento debe `1pUser` solicitar al usuario un inicio de sesión y, cuando reciba un inicio de sesión válido, vuelva a devolver el nombre en . Dado que el complemento puede cambiar esta cadena, el IDE `SCC_USER_LEN` siempre asignará un búfer de tamaño (+ 1). Si se cambia la cadena, la nueva cadena debe ser un nombre de inicio de sesión válido (al menos tan válido como la cadena anterior).

Notas técnicas para SccCreateSubProject y SccGetParentProjectPath

Agregar soluciones y proyectos al control de código fuente se ha simplificado en Visual Studio para minimizar el número de veces que se solicita a un usuario que seleccione ubicaciones en el sistema de control de código fuente. Visual Studio activa estos cambios si un complemento de control de código fuente admite

`SccGetParentProjectPath` las dos funciones nuevas, [SccCreateSubProject](#) y la función. Sin embargo, la siguiente entrada del Registro se puede usar para deshabilitar estos cambios y revertir al comportamiento anterior de Visual Studio (Source Control Plug-in API Version 1.1):

[HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8.0, SourceControl]

"DoNotCreateSolutionRootFolderInSourceControl"-dword:00000001

Si esta entrada del Registro no existe o se establece en `dword:00000000`, Visual Studio intenta usar las nuevas funciones `SccCreateSubProject` y `SccGetParentProjectPath`.

Si la entrada del Registro se establece en `dword:00000001`, Visual Studio no intenta usar estas nuevas funciones y las operaciones de agregar al control de código fuente funcionan como lo hacían en versiones anteriores de Visual Studio.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [SccCreateSubProject](#)
- [SccGetProjPath](#)

Función SccGetProjPath

06/04/2020 • 9 minutes to read • [Edit Online](#)

Esta función solicita al usuario una ruta de acceso del proyecto, que es una cadena que es significativa solo para el complemento de control de código fuente. Se llama cuando el usuario es:

- Creación de un nuevo proyecto
- Adición de un proyecto existente al control de versiones
- Intentar encontrar un proyecto de control de versiones existente

Sintaxis

```
SCCRTN SccGetProjPath (  
    LPVOID pvContext,  
    HWND   hWnd,  
    LPSTR  lpUser,  
    LPSTR  lpProjName,  
    LPSTR  lpLocalPath,  
    LPSTR  lpAuxProjPath,  
    BOOL   bAllowChangePath,  
    LPBOOL pbNew  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpUser

[adentro, fuera] El nombre de usuario (no debe superar SCC_USER_SIZE, incluido el terminador NULL)

lpProjName

[adentro, fuera] El nombre del proyecto IDE, el área de trabajo del proyecto o makefile (no debe superar SCC_PRJPATH_SIZE, incluido el terminador NULL).

lpLocalPath

[adentro, fuera] La ruta de trabajo del proyecto. Si `bAllowChangePath` es `TRUE`, el complemento de control de código fuente puede modificar esta cadena (no superar _MAX_PATH, incluido el terminador nulo).

lpAuxProjPath

[adentro, fuera] Un búfer para la ruta de acceso del proyecto devuelta (no debe superar SCC_PRJPATH_SIZE, incluido el terminador NULL).

bAllowChangePath

[en] Si se `TRUE` trata, el complemento de control `lpLocalPath` de código fuente puede solicitar y modificar la

cadena.

pbNew

[adentro, fuera] El valor que entra indica si se debe crear un nuevo proyecto. El valor devuelto indica el éxito de la creación de un proyecto:

ENTRANTE	INTERPRETACIÓN
TRUE	El usuario puede crear un nuevo proyecto.
FALSE	El usuario no puede crear un nuevo proyecto.

SALIENTE	INTERPRETACIÓN
TRUE	Se creó un nuevo proyecto.
FALSE	Se ha seleccionado un proyecto existente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El proyecto se creó o recuperó correctamente.
SCC_I_OPERATIONCANCELED	Se canceló la operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención.
SCC_E_CONNECTIONFAILURE	Se ha producido un problema al intentar conectarse al sistema de control de código fuente.
SCC_E_NONSPECIFICERROR	Se ha producido un error no especificado.

Observaciones

El propósito de esta función es que `1pProjName` `1pAuxProjPath` el IDE adquiera los parámetros y . Después de que el complemento de control de código fuente solicita al usuario esta información, pasa estas dos cadenas de nuevo al IDE. El IDE conserva estas cadenas en su archivo de solución y las pasa a [SccOpenProject](#) cada vez que el usuario abre este proyecto. Estas cadenas permiten al complemento realizar un seguimiento de la información asociada a un proyecto.

Cuando se llama por `1pAuxProjPath` primera vez a la función, se establece en una cadena vacía. `1pProjName` También puede estar vacío o puede contener el nombre del proyecto IDE, que el complemento de control de código fuente puede usar o ignorar. Cuando la función devuelve correctamente, el complemento devuelve las dos cadenas correspondientes. El IDE no hace ninguna suposición sobre estas cadenas, no las usará y no permitirá al usuario modificarlas. Si el usuario desea cambiar la configuración, `SccGetProjPath` el IDE llamará de nuevo, pasando los mismos valores que había recibido la vez anterior. Esto le da al plug-in un control completo sobre estas dos cadenas.

Para `lpUser`, el IDE puede pasar un nombre de usuario o simplemente puede pasar un puntero a una cadena vacía. Si hay un nombre de usuario, el complemento de control de código fuente debe usarlo como valor predeterminado. Sin embargo, si no se pasó ningún nombre o si el inicio de sesión ha fallado `lpUser` con el nombre especificado, el complemento debe solicitar al usuario un inicio de sesión y devolver el nombre cuando reciba un inicio de sesión válido. Dado que el complemento puede cambiar esta cadena, el IDE `SCC_USER_LEN` siempre asignará un búfer de tamaño (+1).

NOTE

La primera acción que realiza el IDE puede `ScOpenProject` ser `ScGetProjPath` una llamada a la función o a la función. Por lo tanto, ambos `lpUser` tienen un parámetro idéntico, lo que permite que el complemento de control de código fuente inicie sesión en el usuario en cualquier momento. Incluso si la devolución de la función indica un error, el complemento debe rellenar esta cadena con un nombre de inicio de sesión válido.

`lpLocalPath` es el directorio donde el usuario mantiene el proyecto. Puede ser una cadena vacía. Si no hay ningún directorio definido actualmente (como en el caso de un usuario `bAllowChangePath` `TRUE` que intenta descargar un proyecto desde el sistema de control de código fuente) y si es , el complemento de control de código fuente puede solicitar al usuario la entrada o utilizar algún otro método para colocar su propia cadena en . Si `bAllowChangePath` `FALSE` es , el complemento no debe cambiar la cadena, porque el usuario ya está trabajando en el directorio especificado.

Si el usuario crea un nuevo proyecto que se pondrá bajo control de código fuente, es `ScGetProjPath` posible que el complemento de control de código fuente no lo cree realmente en el sistema de control de código fuente en el momento. En su lugar, devuelve la cadena junto `pbNew` con un valor distinto de cero para , lo que indica que el proyecto se creará en el sistema de control de código fuente.

Por ejemplo, si un usuario del Asistente para **nuevo proyecto** en Visual Studio agrega su proyecto al control de código fuente, Visual Studio llama a esta función y el complemento determina si está bien crear un nuevo proyecto en el sistema de control de código fuente que contenga el proyecto de Visual Studio. Si el usuario hace clic en **Cancelar** antes de completar el asistente, el proyecto nunca se crea. Si el usuario **OK** hace clic `ScOpenProject` en Aceptar `SCC_OPT_CREATEIFNEW`, Visual Studio llama a , pasando y el proyecto controlado por código fuente se crea en ese momento.

Vea también

- [Funciones de API de plug-in de control de código fuente](#)
- [ScOpenProject](#)

ScGetUserOption (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función recupera una variedad de opciones específicas del usuario.

Sintaxis

```
SCCRTN ScGetUserOption(  
    LPVOID pContext,  
    LONG nOption,  
    LPLONG lpVal  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

nOpción

[en] Opción para recuperar (consulte Comentarios para ver las posibles opciones).

lpVal

[fuera] Valor asociado a la opción.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La opción se recuperó correctamente.
SCC_E_OPNOTSUPPORTED	No se admite la opción.
SCC_E_NONSPECIFICERROR	Se ha producido un error no especificado.

Observaciones

Este comando admite las siguientes opciones:

OPCIÓN DE USUARIO	DESCRIPCIÓN
<code>SCC_USEROPT_CHECKOUT_LOCALVER</code>	Determina si el usuario desea desproteger la versión local de los archivos. <code>lpVal</code> se <code>SCC_USEROPT_COLV_YES</code> asigna (el usuario desea desproteger los archivos locales) o <code>SCC_USEROPT_COLV_NO</code> .

Vea también

- Funciones de API de complemento de control de código fuente
- Códigos de error

SccGetVersion (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función obtiene el número de versión de la API de complemento de Control de código fuente compatible con el complemento de control de código fuente.

Sintaxis

```
LONG SccGetVersion(void);
```

Parámetros

Ninguno.

Valor devuelto

Tipo `LONG` de datos que contiene el número de versión de la API de complemento de Control de código fuente compatible:

WORD	DESCRIPCIÓN
HIWORD	Versión principal
LOWORD	Versión secundaria

Observaciones

Por ejemplo, si un complemento de control de código fuente admite la versión 1.3 de la API de complemento de Control de código fuente, esta función devolvería 0x0103.

Vea también

- [Funciones de API de complemento de control de código fuente](#)

ScchHistory (Función)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Esta función muestra el historial de los archivos especificados.

Sintaxis

```
SCCRTN ScchHistory(  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG      nFiles,  
    LPCSTR*   lpFileNames,  
    LONG      fOptions,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nFiles

[en] Número de archivos especificados en la lpFileName matriz.

lpFileName

[en] Matriz de nombres completos de archivos.

fOptions

[en] Marcas de comando (actualmente no se utilizan).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El historial de versiones se ha obtenido correctamente.
SCC_I_RELOADFILE	El sistema de control de código fuente modificó realmente el archivo en el disco mientras se recupera el historial (por ejemplo, obteniendo una versión antigua del mismo), por lo que el IDE debe volver a cargar este archivo.

VALUE	DESCRIPCIÓN
SCC_E_FILENOTCONTROLLED	El archivo no está bajo control de código fuente.
SCC_E_OPNOTSUPPORTED	El sistema de control de código fuente no admite esta operación.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_PROJNOTOPEN	El proyecto no se ha abierto.
SCC_E_NONSPECIFICERROR	Fallo inespecífico. No se pudo obtener el historial de archivos.

Observaciones

El complemento de control de código fuente puede mostrar su propio `hWnd` cuadro de diálogo para mostrar el historial de cada archivo, utilizando como la ventana primaria. Como alternativa, se puede utilizar la función de devolución de llamada de salida de texto opcional proporcionada a [SccOpenProject](#), si se admite.

Tenga en cuenta que, en determinadas circunstancias, el archivo que se está examinando puede cambiar durante la ejecución de esta llamada. Por ejemplo, Visual SourceSafe el comando history da al usuario la oportunidad de obtener una versión antigua del archivo. En tal caso, el complemento de `SCC_I_RELOAD` control de código fuente vuelve a advertir al IDE que necesita volver a cargar el archivo.

NOTE

Si el complemento de control de código fuente no admite esta función para una matriz de archivos, solo se puede mostrar el historial de archivos del primer archivo.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [SccOpenProject](#)

ScclInitialize (Función)

06/04/2020 • 5 minutes to read • [Edit Online](#)

Esta función inicializa el complemento de control de código fuente y proporciona capacidades y límites al entorno de desarrollo integrado (IDE).

Sintaxis

```
SCCRTN ScclInitialize (  
    LPVOID* ppvContext,  
    HWND    hWnd,  
    LPCSTR  lpCallerName,  
    LPSTR   lpScclName,  
    LPLONG  lpScclCaps,  
    LPSTR   lpAuxPathLabel,  
    LPLONG  pnCheckoutCommentLen,  
    LPLONG  pnCommentLen  
);
```

Parámetros

ppvContext

[en] El complemento de control de código fuente puede colocar un puntero a su estructura de contexto aquí.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpCallerName

[en] El nombre del programa que llama al complemento de control de código fuente.

lpScclName

[adentro, fuera] El búfer donde el complemento de control de código `SCC_NAME_LEN` fuente coloca su propio nombre (no debe superar).

lpScclCaps

[fuera] Devuelve los indicadores de capacidad del complemento de control de código fuente.

lpAuxPathLabel

[adentro, fuera] El búfer donde el complemento de control de código `lpAuxProjPath` fuente coloca una cadena que describe el parámetro devuelto `SCC_AUXLABEL_LEN` por [ScclOpenProject](#) y [ScclGetProjPath](#) (no superar).

pnCheckoutCommentLen

[fuera] Devuelve la longitud máxima permitida para un comentario de pago.

pnCommentLen

[fuera] Devuelve la longitud máxima permitida para otros comentarios.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La inicialización del control de código fuente se realizó correctamente.
SCC_E_INITIALIZEFAILED	No se pudo inicializar el sistema.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar la operación especificada.
SCC_E_NONSPECIFICERROR	Error inespecífico; sistema de control de código fuente no se inicializó.

Observaciones

El IDE llama a esta función cuando carga por primera vez el complemento de control de código fuente. Permite al IDE pasar cierta información, como el nombre del autor de la llamada, al complemento. El IDE también recupera cierta información, como la longitud máxima permitida para los comentarios y las capacidades del complemento.

Apunta `ppvContext` a `NULL` un puntero. El complemento de control de código fuente puede asignar una estructura `ppvContext` para su propio uso y almacenar un puntero a esa estructura en `ppvContext`. El IDE pasará este puntero a todas las demás funciones de la API de VSSCI, lo que permite que el complemento tenga información de contexto disponible sin recurrir al almacenamiento global y admitir varias instancias del complemento. Esta estructura debe desasignarse cuando se llama a [SccUninitialize](#).

Los `lpCallerName` `lpSccName` parámetros y permiten que el IDE y el complemento de control de código fuente intercambien nombres. Estos nombres se pueden utilizar simplemente para distinguir entre varias instancias, o pueden aparecer en menús o cuadros de diálogo.

El `lpAuxPathLabel` parámetro es una cadena que se utiliza como comentario para identificar la ruta de acceso del proyecto auxiliar que se almacena en el archivo de solución y se pasa al complemento de control de código fuente en una llamada a [SccOpenProject](#). Visual SourceSafe utiliza la cadena "SourceSafe Project."; otros complementos de control de código fuente deben abstenerse de usar esta cadena en particular.

El `lpSccCaps` parámetro proporciona al complemento de control de código fuente un lugar para almacenar bitflags que indican las capacidades del complemento. (Para obtener una lista completa de indicadores de bits de capacidad, consulte [Indicadores de capacidad](#)). Por ejemplo, si el complemento planea escribir resultados en una función de devolución de llamada proporcionada por el autor de la llamada, el complemento establecería el bit de capacidad `SCC_CAP_TEXTOUT`. Esto indicaría que el IDE crea una ventana para los resultados del control de versiones.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [SccUninitialize](#)
- [SccOpenProject](#)
- [Marcas de capacidad](#)

ScclsMultiCheckoutEnabled (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función comprueba si el complemento de control de código fuente permite varias desprotecciones en un archivo.

Sintaxis

```
SCCRTN ScclsMultiCheckoutEnabled(  
    LPVOID pContext,  
    LPBOOL pbMultiCheckout  
);
```

Parámetros

pContext

[en] La estructura de contexto del complemento de control de código fuente.

pbMultiCheckout

[fuera] Especifica si se habilitan varias desprotecciones para este proyecto (no cero significa que se admiten varias desprotecciones).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El cheque fue exitoso.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.
SCC_E_UNKNOWNERROR	

Observaciones

El IDE realiza dos comprobaciones para determinar si más de un usuario puede desproteger los archivos simultáneamente. En primer lugar, el sistema de control de código fuente debe admitir varias desprotecciones. El complemento de control de código fuente puede `SCC_CAP_MULTICHECKOUT` especificar esta capacidad durante la inicialización especificando el archivo . A partir de entonces, como segunda comprobación, el IDE llama a esta función para determinar si el proyecto actual admite o no varias desprotecciones. Si se admiten varias desprotecciones para el proyecto seleccionado, el complemento devuelve un código de éxito y se establece `pbMultiCheckout` en distinto de cero (`TRUE`) o `FALSE` .

Vea también

- [Funciones de API de complemento de control de código fuente](#)

SccOpenProject (Función)

06/04/2020 • 9 minutes to read • [Edit Online](#)

Esta función abre un proyecto de control de código fuente existente o crea uno nuevo.

Sintaxis

```
SCCRTN SccOpenProject (  
    LPVOID      pvContext,  
    HWND        hWnd,  
    LPSTR        lpUser,  
    LPCSTR       lpProjName,  
    LPCSTR       lpLocalProjPath,  
    LPSTR        lpAuxProjPath,  
    LPCSTR       lpComment,  
    LPTEXTOUTPROC lpTextOutProc,  
    LONG         dwFlags  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpUser

[adentro, fuera] El nombre del usuario (no debe superar SCC_USER_SIZE, incluido el terminador NULL).

lpProjName

[en] Cadena que identifica el nombre del proyecto.

lpLocalProjPath

[en] La ruta de acceso a la carpeta de trabajo para el proyecto.

lpAuxProjPath

[adentro, fuera] Una cadena auxiliar opcional que identifica el proyecto (no debe superar SCC_AUXPATH_SIZE, incluido el terminador NULL).

lpComment

[en] Comentar un nuevo proyecto que se está creando.

lpTextOutProc

[en] Una función de devolución de llamada opcional para mostrar la salida de texto desde el complemento de control de código fuente.

dwFlags

[en] Indica si es necesario crear un nuevo proyecto si el proyecto es desconocido para el complemento de

control de código fuente. El valor puede `SCC_OP_CREATEIFNEW` ser una combinación de y `SCC_OP_SILENTOPEN`.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El éxito en la apertura del proyecto.
SCC_E_INITIALIZEFAILED	No se pudo inicializar el proyecto.
SCC_E_INVALIDUSER	El usuario no pudo iniciar sesión en el sistema de control de código fuente.
SCC_E_COULDNOTCREATEPROJECT	El proyecto no existía antes de la llamada; se <code>SCC_OPT_CREATEIFNEW</code> estableció la bandera, pero no se pudo crear el proyecto.
SCC_E_PROJSYNTAXERR	Sintaxis de proyecto no válida.
SCC_E_UNKNOWNPROJECT	El proyecto es desconocido para el complemento <code>SCC_OPT_CREATEIFNEW</code> de control de código fuente y el indicador no se estableció.
SCC_E_INVALIDFILEPATH	Ruta de acceso de archivo no válida o inutilizable.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Un error inespecífico; el sistema de control de código fuente no se inicializó.

Observaciones

El IDE puede pasar un `lpUser` nombre de usuario (), o simplemente puede pasar un puntero a una cadena vacía. Si hay un nombre de usuario, el complemento de control de código fuente debe usarlo como valor predeterminado. Sin embargo, si no se pasó ningún nombre, o si el inicio de sesión falló con `lpUser` el nombre dado, `.` el complemento debe solicitar al usuario que inicie sesión y devolverá el nombre válido cuando reciba un inicio de sesión válido. Porque el complemento puede cambiar la cadena de nombre de usuario, el IDE siempre asignará un búfer de tamaño (`SCC_USER_LEN` + 1 o `SCC_USER_SIZE`, que incluye espacio para el terminador nulo).

NOTE

La primera acción que puede ser necesaria que el `SccOpenProject` IDE realice puede ser una llamada a la función o [sccGetProjPath](#). Por esta razón, ambos tienen `lpUser` un parámetro idéntico.

`lpAuxProjPath` y `lpProjName` se leen desde el archivo de solución, o se devuelven de una llamada a la

`SccGetProjPath` función. Estos parámetros contienen las cadenas que el complemento de control de código fuente asocia con el proyecto y son significativas solo para el complemento. Si no hay tales cadenas en el archivo de solución y no se `SccGetProjPath` ha pedido al usuario que examine `lpAuxProjPath` `lpProjName` (que devolvería una cadena a través de la función), el IDE pasa cadenas vacías para ambos y , y espera que el complemento actualice estos valores cuando se devuelva esta función.

`lpTextOutProc` es un puntero a una función de devolución de llamada proporcionada por el IDE al complemento de control de código fuente con el fin de mostrar la salida de resultado del comando. Esta función de devolución de llamada se describe en detalle en [LPTEXTOUTPROC](#).

NOTE

Si el complemento de control de código fuente tiene la `SCC_CAP_TEXTOUT` intención de aprovechar esto, debe haber establecido la marca en [el SccInitialize](#). Si no se ha establecido esa marca, o `lpTextOutProc` si `NULL` el IDE no admite esta característica, será .

El `dwFlags` parámetro controla el resultado en caso de que el proyecto que se está abriendo no exista actualmente. Consta de dos bitflags, `SCC_OP_CREATEIFNEW` y `SCC_OP_SILENTOPEN` . Si el proyecto que se está abriendo ya `SCC_OK` existe, la función simplemente abre el proyecto y devuelve . Si el proyecto no existe `SCC_OP_CREATEIFNEW` y la marca está activada, el complemento de control de código fuente `SCC_OK` puede crear el proyecto en el sistema de control de código fuente, abrirlo y devolver . Si el proyecto no existe, `SCC_OP_CREATEIFNEW` y si la marca está desactivada, `SCC_OP_SILENTOPEN` el complemento debe comprobar la marca. Si esa marca no está activada, el complemento puede solicitar al usuario un nombre de proyecto. Si esa marca está activada, el `SCC_E_UNKNOWNPROJECT` complemento simplemente debe devolver .

Orden de llamada

En el curso normal de los eventos, el [SccInitialize](#) se llamaría primero para abrir una sesión de control de código fuente. Una sesión puede consistir en una llamada a `SccOpenProject` , seguida de otras llamadas a la función de complemento de Control de código fuente y finalizará con una llamada a [SccCloseProject](#). Estas sesiones pueden repetirse varias veces antes de que se llame a [SccUninitialize](#).

Si el complemento de control `SCC_CAP_REENTRANT` de `SccInitialize` código fuente establece el bit en , la secuencia de sesión anterior puede repetirse muchas veces en paralelo. Diferentes `pvContext` estructuras realizan un seguimiento `pvContext` de las diferentes sesiones, en las que cada una está asociada a un proyecto abierto a la vez. Según `pvContext` el parámetro, el complemento puede determinar a qué proyecto se hace referencia en cualquier llamada determinada. Si no `SCC_CAP_REENTRANT` se establece el bit de capacidad, los complementos de control de código fuente no reentrantes están limitados en su capacidad para trabajar con varios proyectos.

NOTE

El `SCC_CAP_REENTRANT` bit se introdujo en la versión 1.1 de la API de complemento de Control de código fuente. No se establece o se omite en la versión 1.0, y se supone que todos los complementos de control de código fuente de la versión 1.0 no son reentrantes.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [SccCloseProject](#)
- [SccGetProjPath](#)
- [SccInitialize](#)

- [SccUninitialize](#)
- [Restricciones en las longitudes de cadena](#)
- [LPTEXTOUTPROC](#)

ScPopulateDirList (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función determina qué directorios y (opcionalmente) archivos se almacenan en el control de código fuente, dada una lista de directorios para examinar.

Sintaxis

```
SCCRTN ScPopulateDirList(  
    LPVOID      pContext,  
    LONG        nDirs,  
    LPCSTR*     lpDirPaths,  
    POPDIRLISTFUNCpfnPopulate,  
    LPVOID      pvCallerData,  
    LONG        fOptions  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

nDirs

[en] Número de rutas `lpDirPaths` de directorio en la matriz.

lpDirPaths

[en] Matriz de rutas de directorio que se va a examinar.

pfnPopulate

[en] Función de devolución de llamada para llamar `lpDirPaths` a cada ruta de acceso de directorio y (opcionalmente) nombre de archivo en (consulte [POPDIRLISTFUNC](#) para obtener más información).

pvCallerData

[en] Valor que se va a pasar sin cambios a la función de devolución de llamada.

fOptions

[en] Combinación de valores que controlan cómo se procesan los directorios (consulte la sección "PopulateDirList flags" de [Bitflags Used by Specific Commands](#) para los valores posibles).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La operación se completó correctamente.
SCC_E_UNKNOWNERROR	Se produjo un error.

Observaciones

Solo los directorios y (opcionalmente) nombres de archivo que están realmente en el repositorio de control de código fuente se pasan a la función de devolución de llamada.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [Marcadores de bits utilizados por comandos específicos](#)
- [POPDIRLISTFUNC](#)
- [Códigos de error](#)

SccPopulateList (Función)

06/04/2020 • 6 minutes to read • [Edit Online](#)

Esta función actualiza una lista de archivos para un comando de control de código fuente determinado y proporciona el estado de control de código fuente en todos los archivos determinados.

Sintaxis

```
SCCRTN SccPopulateList (  
    LPVOID          pvContext,  
    enum SCCCOMMAND nCommand,  
    LONG            nFiles,  
    LPCSTR*         lpFileNames,  
    POPLISTFUNC     pfnPopulate,  
    LPVOID          pvCallerData,  
    LPLONG          lpStatus,  
    LONG            fOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

nCommand

[en] El comando de control de código fuente `lpFileNames` que se aplicará a todos los archivos de la matriz (consulte Código de [comandos](#) para obtener una lista de posibles comandos).

nArchivos

[en] Número de archivos `lpFileNames` de la matriz.

lpFileNames

[en] Matriz de nombres de archivo conocidos por el IDE.

pfnPopulate

[en] La función de devolución de llamada IDE para llamar para agregar y quitar archivos (consulte [POPLISTFUNC](#) para obtener más información).

pvCallerData

[en] Valor que se va a pasar sin cambios a la función de devolución de llamada.

lpStatus

[adentro, fuera] Matriz para que el complemento de control de código fuente devuelva los indicadores de estado de cada archivo.

fOptions

[en] Indicadores de comando (consulte la sección "PopulateList flag" de [Bitflags Used by Specific Commands](#) para obtener más información).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Correcto.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

Esta función examina la lista de archivos para su estado actual. Utiliza la `pfnPopulate` función de devolución de llamada para notificar `nCommand` al autor de la llamada cuando un archivo no coincide con los criterios para el archivo. Por ejemplo, si `SCC_COMMAND_CHECKIN` el comando es y un archivo de la lista no está desprotegido, la devolución de llamada se utiliza para informar al autor de la llamada. Ocasionalmente, el complemento de control de código fuente puede encontrar otros archivos que podrían formar parte del comando y agregarlos. Esto permite, por ejemplo, que un usuario de Visual Basic desproteger un archivo .bmp que usa su proyecto pero no aparece en el archivo de proyecto de Visual Basic. Un usuario elige el comando **Get** en el IDE. El IDE mostrará una lista de todos los archivos que cree que el `ScpPopulateList` usuario puede obtener, pero antes de que se muestre la lista, se llama a la función para asegurarse de que la lista que se va a mostrar está actualizada.

Ejemplo

El IDE crea una lista de archivos que cree que el usuario puede obtener. Antes de que muestre esta `ScpPopulateList` lista, llama a la función, dando al complemento de control de código fuente la oportunidad de agregar y eliminar archivos de la lista. El complemento modifica la lista llamando a la función de devolución de llamada dada (consulte [POPLISTFUNC](#) para obtener más detalles).

El complemento continúa llamando a `pfnPopulate` la función, que agrega y elimina archivos, hasta `ScpPopulateList` que finaliza y, a continuación, vuelve de la función. A continuación, el IDE puede mostrar su lista. La `lpStatus` matriz representa todos los archivos de la lista original que pasa el IDE. El complemento rellena el estado de todos estos archivos además de hacer uso de la función de devolución de llamada.

NOTE

Un complemento de control de código fuente siempre tiene la opción de simplemente volver inmediatamente desde esta función, dejando la lista tal como está. Si un complemento implementa esta función, puede indicarlo estableciendo el `SCC_CAP_POPULATELIST` bitflag de capacidad en la primera llamada a `ScclInitialize`. De forma predeterminada, el complemento siempre debe suponer que todos los elementos que se pasan son archivos. Sin embargo, si `SCC_PL_DIR` el IDE establece la marca en el `fOptions` parámetro, todos los elementos que se pasan deben considerarse directorios. El complemento debe agregar todos los archivos que pertenecen a los directorios. El IDE nunca pasará una mezcla de archivos y directorios.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [ScclInitialize](#)
- [POPLISTFUNC](#)
- [Marcadores de bits utilizados por comandos específicos](#)
- [Código de comando](#)

SccProperties (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función muestra las propiedades de control de código fuente para un archivo o proyecto.

Sintaxis

```
SCCRTN SccProperties (  
    LPVOID pvContext,  
    HWND   hWnd,  
    LPCSTR lpFileName  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpFileName

[en] El nombre de ruta de acceso completo del archivo o proyecto.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Las propiedades se mostraron correctamente.
SCC_I_RELOADFILE	El sistema de control de versiones ha modificado las propiedades del archivo, por lo que el IDE debe volver a cargar este archivo.
SCC_E_PROJNOTOPEN	El proyecto especificado no se ha abierto en el control de código fuente.
SCC_E_NOTAUTHORIZED	El usuario no está autorizado para ver las propiedades de este archivo o proyecto.
SCC_E_FILENOTCONTROLLED	El archivo o proyecto especificado no está bajo control de código fuente.
SCC_E_NONSPECIFICERROR	Se ha producido un error desconocido o general.
SCC_E_UNKNOWNERROR	

Observaciones

El complemento de control de código fuente muestra las propiedades en su propio cuadro de diálogo.

Las propiedades se definen mediante el complemento de control de código fuente y pueden diferir de un complemento a un complemento. Si el complemento permite al usuario cambiar las propiedades de control

`SCC_I_RELOAD` de código fuente de un archivo, debe volver a indicar al IDE que este archivo o proyecto debe volver a cargarse.

Vea también

- [Funciones de API de complemento de control de código fuente](#)

SccQueryChanges (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función enumera una lista determinada de archivos, proporcionando información sobre los cambios de nombre para cada archivo a través de una función de devolución de llamada.

Sintaxis

```
SCCRTN SccQueryChanges(  
    LPVOID          pContext,  
    LONG            nFiles,  
    LPCSTR*         lpFileNames,  
    QUERYCHANGESFUNC pfnCallback,  
    LPVOID          pvCallerData  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

nArchivos

[en] Número de `lpFileNames` archivos en la matriz.

lpFileNames

[en] Matriz de nombres de archivo para obtener información sobre.

pfnCallback

[en] Función de devolución de llamada para llamar a cada nombre de archivo de la lista (consulte [QUERYCHANGESFUNC](#) para obtener más información).

pvCallerData

[en] Valor que se pasará sin cambios a la función de devolución de llamada.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	El proceso de consulta se completó correctamente.
SCC_E_PROJNOTOPEN	El proyecto no se ha abierto en el control de código fuente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención.
SCC_E_NONSPECIFICERROR	Se ha producido un error general o no especificado.

Observaciones

Los cambios que se consultan se realizan en el espacio de nombres: específicamente, cambiar el nombre, agregar y quitar un archivo.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [QUERYCHANGESFUNC](#)
- [Códigos de error](#)

ScqQueryInfo (Función)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Esta función obtiene información de estado para un conjunto de archivos seleccionados bajo control de código fuente.

Sintaxis

```
SCCRTN ScqQueryInfo(  
    LPVOID    pvContext,  
    LONG      nFiles,  
    LPCSTR*   lpFileNames,  
    LPLONG    lpStatus  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz y la longitud de la matriz.

lpFileNames

[en] Matriz de nombres de archivos que se van a consultar.

lpStatus

[adentro, fuera] Matriz en la que el complemento de control de código fuente devuelve los indicadores de estado de cada archivo. Para obtener más información, consulte Código de [estado de archivo](#).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La consulta se realizó correctamente.
SCC_E_ACCESSFAILURE	Se ha producido un problema con el acceso al sistema de control de código fuente, probablemente causado por problemas de red o contención. Se recomienda un reintento.
SCC_E_PROJNOTOPEN	El proyecto no está abierto bajo control de código fuente.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

Si `lpFileName` es una cadena vacía, actualmente no hay información de estado que actualizar. De lo contrario, es

el nombre de ruta de acceso completo del archivo para el que puede haber cambiado la información de estado.

La matriz return puede ser `SCC_STATUS_XXXX` una máscara de bits. Para obtener más información, consulte Código de [estado de archivo](#) . Es posible que un sistema de control de código fuente no admita todos los tipos de bits. Por ejemplo, `SCC_STATUS_OUTOFDATE` si no se ofrece, el bit simplemente no se establece.

Cuando utilice esta función para desproteger archivos, tenga en cuenta los siguientes `MSSCCI` requisitos de estado:

- `SCC_STATUS_OUTBYUSER` se establece cuando el usuario actual ha traído el archivo.
- `SCC_STATUS_CHECKEDOUT` no se `SCC_STATUS_OUTBYUSER` puede establecer a menos que se establezca.
- `SCC_STATUS_CHECKEDOUT` sólo se establece cuando el archivo se desprotege en el directorio de trabajo designado.
- Si el usuario actual desprotege el archivo en un `SCC_STATUS_OUTBYUSER` directorio distinto `SCC_STATUS_CHECKEDOUT` del directorio de trabajo, se establece, pero no.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [Código de estado de archivo](#)

SccRemove (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función elimina archivos del sistema de control de código fuente.

Sintaxis

```
SCCRTN SccRemove(  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG       nFiles,  
    LPCSTR*    lpFileNames,  
    LPCSTR     lpComment,  
    LONG       fOptions,  
    LPCMDOPTS  pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz.

lpFileNames

[en] Matriz de nombres de ruta local completos de los archivos que se van a quitar.

lpComment

[en] El comentario que se aplicará a cada archivo que se va a quitar.

fOptions

[en] Marcas de comando (sin usar).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La remoción fue exitosa.

VALUE	DESCRIPCIÓN
SCC_E_FILENOTCONTROLLED	El archivo seleccionado no está bajo control de código fuente.
SCC_E_OPNOTSUPPORTED	El sistema de control de código fuente no admite esta operación.
SCC_E_ISCHECKEDOUT	No se puede quitar un archivo porque un usuario lo tiene desprotegido.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_NONSPECIFICERROR	Error inespecífico; archivo no se eliminó.
SCC_I_OPERATIONCANCELED	La operación se canceló antes de la finalización.

Observaciones

Esta función elimina los archivos del sistema de control de código fuente, pero no los elimina del disco duro local del usuario.

Vea también

- [Funciones de API de complemento de control de código fuente](#)

SccRename (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función cambia el nombre de un archivo en el sistema de control de código fuente.

Sintaxis

```
SCCRTN SccRename(  
    LPVOID pvContext,  
    HWND   hWnd,  
    LPCSTR lpFileName,  
    LPCSTR lpNewName  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

lpFileName

[en] El nombre de archivo completo del archivo que se va a cambiar de nombre.

lpNewName

[in] Nuevo nombre completo. Si la ruta de acceso del directorio es diferente, el archivo se ha movido de un subdirectorio a otro.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La operación de cambio de nombre se completó correctamente.
SCC_E_PROJNOTOPEN	El proyecto no está abierto bajo control de código fuente.
SCC_E_FILENOTCONTROLLED	El archivo no está bajo control de código fuente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención.
SCC_E_NOTAUTHORIZED	El usuario no está autorizado para completar esta operación.

VALUE	DESCRIPCIÓN
SCC_E_COULDNOTCREATEPROJECT	No se pudo crear el proyecto como parte del proceso de cambio de nombre.
SCC_E_OPNOTPERFORMED	La operación no se realizó.
SCC_E_NONSPECIFICERROR	Se ha producido un error general o no especificado.

Observaciones

Esta función se puede utilizar para cambiar el nombre de un archivo o moverlo de una ubicación a otra en el sistema de control de código fuente. El complemento de control de código fuente no debe intentar acceder al archivo en el disco. Es responsabilidad del IDE cambiar el nombre del archivo local.

Vea también

- [Funciones de API de complemento de control de código fuente](#)

SccRunScc (Función)

06/04/2020 • 3 minutes to read • [Edit Online](#)

Esta función invoca la herramienta de administración del control de código fuente.

Sintaxis

```
SCCRTN SccRunScc(  
    LPVOID pvContext,  
    HWND hWnd,  
    LONG nFiles,  
    LPCSTR* lpFileNames  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz.

lpFileNames

[en] Matriz de nombres de archivo seleccionados.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La herramienta de administración del control de código fuente se invocó correctamente.
SCC_I_OPERATIONCANCELED	La operación fue cancelada.
SCC_E_INITIALIZEFAILED	No se pudo inicializar el sistema de control de código fuente.
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención.
SCC_E_CONNECTIONFAILURE	No se pudo conectar al sistema de control de código fuente.
SCC_E_FILENOTCONTROLLED	El archivo seleccionado no está bajo control de código fuente.

VALUE	DESCRIPCIÓN
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

Esta función permite al autor de la llamada acceder a toda la gama de características del sistema de control de código fuente a través de una herramienta de administración externa. Si el sistema de control de código fuente no tiene interfaz de usuario, el complemento de control de código fuente puede implementar una interfaz para realizar las funciones de administración necesarias.

Esta función se llama con un recuento y una matriz de nombres de archivo para los archivos seleccionados actualmente. Si la herramienta de administración lo admite, la lista de archivos se puede utilizar para preseleccionar archivos en la interfaz de administración; de lo contrario, la lista puede ser ignorada.

Esta función se invoca normalmente cuando el usuario selecciona la >**Iniciar** <**servidor** de Control de código fuente en el -> menú **Control de código fuente** de archivos. Esta opción del menú **Inicio** siempre se puede deshabilitar o incluso ocultar estableciendo una entrada del Registro. Consulte [Cómo: Instalar un complemento](#) de Control de código fuente para obtener más información. Esta función solo se llama `SCC_CAP_RUNSCC` si `ScclInitialize` devuelve el bit de capacidad (consulte [Indicadores](#) de capacidad para obtener más información sobre este y otros bits de capacidad).

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [Instalación de un complemento de control de código fuente](#)
- [Marcas de capacidad](#)
- [ScclInitialize](#)

SccSetOption (Función)

06/04/2020 • 9 minutes to read • [Edit Online](#)

Esta función establece opciones que controlan el comportamiento del complemento de control de código fuente.

Sintaxis

```
SCCRTN SccSetOption(  
    LPVOID pvContext,  
    LONG   nOption,  
    LONG   dwVal  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

nOpción

[en] La opción que se está configurando.

dwVal

[en] Configuración de la opción.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La opción se ha establecido correctamente.
SCC_I_SHARESUBPROJOK	Se <code>noption</code> devuelve <code>SCC_OPT_SHARESUBPROJ</code> si lo fue y el complemento de control de código fuente permite que el IDE establezca la carpeta de destino.
SCC_E_OPNOTSUPPORTED	La opción no se estableció y no se debe confiar en ella.

Observaciones

El IDE llama a esta función para controlar el comportamiento del complemento de control de código fuente. El primer `noption` parámetro, , indica el valor que se `dwVal` está establecer, mientras que el segundo, , indica qué hacer con ese valor. El complemento almacena esta información `pvContext``,` asociada a un por lo que el IDE debe llamar a esta función después de llamar a la [ScclInitialize](#) (pero no necesariamente después de cada llamada a la [SccOpenProject](#)).

Resumen de las opciones y sus valores:

NOPTION	DWVALUE	DESCRIPCIÓN
SCC_OPT_EVENTQUEUE	SCC_OPT_EQ_DISABLE SCC_OPT_EQ_ENABLE	Habilita/deshabilita la cola de eventos en segundo plano.
SCC_OPT_USERDATA	Valor arbitrario	Especifica un valor de usuario que se pasará a la función de devolución de llamada OPTNAMECHANGEPFN .
SCC_OPT_HASCANCELMODE	SCC_OPT_HCM_NO SCC_OPT_HCM_YES	Indica si el IDE admite actualmente la cancelación de una operación.
SCC_OPT_NAMECHANGEPFN	Puntero a la función de devolución de llamada OPTNAMECHANGEPFN	Establece un puntero a una función de devolución de llamada de cambio de nombre.
SCC_OPT_SCCCHECKOUTONLY	SCC_OPT_SCO_NO SCC_OPT_SCO_YES	Indica si el IDE permite la desproteger sus archivos manualmente (a través de la interfaz de usuario del control de código fuente) o si solo se deben desproteger a través del complemento de control de código fuente.
SCC_OPT_SHARESUBPROJ	N/D	Si el complemento de control de código fuente permite que el IDE especifique la carpeta del proyecto local, el complemento devuelve SCC_I_SHARESUBPROJOK .

SCC_OPT_EVENTQUEUE

Si `nOption` `SCC_OPT_EVENTQUEUE` es , el IDE está deshabilitando (o volviendo a habilitar) el procesamiento en segundo plano. Por ejemplo, durante una compilación, el IDE podría indicar al complemento de control de código fuente que detenga el procesamiento en reposo de cualquier tipo. Después de la compilación, volvería a habilitar el procesamiento en segundo plano para mantener actualizada la cola de eventos del complemento. Correspondiente al `SCC_OPT_EVENTQUEUE` valor `nOption` de , hay `dwVal` dos valores `SCC_OPT_EQ_ENABLE` posibles `SCC_OPT_EQ_DISABLE` para , a saber, y .

SCC_OPT_HASCANCELMODE

Si el `nOption` valor `SCC_OPT_HASCANCELMODE` de es , el IDE permite a los usuarios cancelar operaciones largas. Establecer `dwVal` `SCC_OPT_HCM_NO` en (el valor predeterminado) indica que el IDE no tiene modo de cancelación. El complemento de control de código fuente debe ofrecer su propio botón Cancelar si desea que el usuario pueda cancelar. `SCC_OPT_HCM_YES` indica que el IDE proporciona la capacidad de cancelar una operación, por lo que el complemento SCC no necesita mostrar su propio botón Cancelar. Si el IDE se `dwVal` establece en `SCC_MSG_STATUS` `DOCANCEL` `lpTextOutProc` `SCC_OPT_HCM_YES` , está preparado para responder y los mensajes enviados a la función de devolución de llamada (consulte [LPTEXTOUTPROC](#)). Si el IDE no establece esta variable, el complemento no debe enviar estos dos mensajes.

SCC_OPT_NAMECHANGEPFN

Si `nOption` está `SCC_OPT_NAMECHANGEPFN` establecido en , y tanto el complemento de control de código fuente como el IDE lo permiten, el complemento puede cambiar el nombre o mover un archivo durante una operación

de control de código fuente. Se `dwVal` establecerá en un puntero de función de tipo [OPTNAMECHANGEPFN](#). Durante una operación de control de código fuente, el complemento puede llamar a esta función, pasando tres parámetros. Estos son el nombre antiguo (con ruta de acceso completa) de un archivo, el nuevo nombre (con ruta de acceso completa) de ese archivo y un puntero a la información que tiene relevancia para el IDE. El IDE envía este último `ScsSetOption` `nOption` puntero `SCC_OPT_USERDATA` llamando `dwVal` a `set` a , con apuntar a los datos. La compatibilidad con esta función es opcional. Un complemento VSSCI que utiliza esta capacidad debe inicializar su `NULL` puntero de función y las variables de datos de usuario en , y no debe llamar a una función de cambio de nombre a menos que se le haya dado una. También debe estar preparado para mantener el valor que se le dio `ScsSetOption` o para cambiarlo en respuesta a una nueva llamada a . Esto no sucederá en medio de una operación de comando de control de código fuente, pero puede suceder entre comandos.

SCC_OPT_SCCCHECKOUTONLY

Si `nOption` se `SCC_OPT_SCCCHECKOUTONLY` establece en , el IDE indica que los archivos del proyecto actualmente abierto nunca se deben desproteger manualmente a través de la interfaz de usuario del sistema de control de código fuente. En su lugar, los archivos deben desprotegerse solo a través del complemento de control de código fuente bajo control IDE. Si `dwValue` se `SCC_OPT_SCO_NO` establece en , significa que los archivos deben ser tratados normalmente por el complemento y se pueden desproteger a través de la interfaz de usuario del control de código fuente. Si `dwValue` se `SCC_OPT_SCO_YES` establece en , solo se permite que el complemento desproteger archivos y no se debe invocar la interfaz de usuario del sistema de control de código fuente. Esto es para situaciones en las que el IDE puede tener "pseudoarchivos" que tienen sentido para desproteger solo a través del IDE.

SCC_OPT_SHARESUBPROJ

Si `nOption` se `SCC_OPT_SHARESUBPROJ` establece en , el IDE está probando si el complemento de control de código fuente puede usar una carpeta local especificada al agregar archivos desde el control de código fuente. El valor `dwVal` del parámetro no importa en este caso. Si el complemento permite que el IDE especifique la carpeta de destino local donde se agregarán los archivos desde el `SCC_I_SHARESUBPROJOK` control `ScsSetOption` de código fuente cuando se llama a [ScsAddFromScs](#), el complemento debe devolver cuando se llama a la función. A continuación, el `lpFileNames` IDE `ScsAddFromScs` utiliza el parámetro de la función para pasar en la carpeta de destino. El complemento utiliza esa carpeta de destino para colocar los archivos agregados desde el control de código fuente. Si el complemento no `SCC_I_SHARESUBPROJOK` vuelve `SCC_OPT_SHARESUBPROJ` cuando se establece la opción, el IDE asume que el complemento puede agregar archivos solo en la carpeta local actual.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [ScsInitialize](#)
- [ScsOpenProject](#)
- [ScsAddFromScs](#)
- [LPTEXTOUTPROC](#)
- [OPTNAMECHANGEPFN](#)

SccUncheckout (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función deshace una operación de desprotección anterior, restaurando así el contenido del archivo o archivos seleccionados al estado anterior a la desprotección. Se pierden todos los cambios realizados en el archivo desde la desprotección.

Sintaxis

```
SCCRTN SccUncheckout (  
    LPVOID    pvContext,  
    HWND      hWnd,  
    LONG      nFiles,  
    LPCSTR*   lpFileNames,  
    LONG      fOptions,  
    LPCMDOPTS pvOptions  
);
```

Parámetros

pvContext

[en] La estructura de contexto del complemento de control de código fuente.

hWnd

[en] Identificador de la ventana IDE que el complemento de control de código fuente puede usar como elemento primario para los cuadros de diálogo que proporciona.

nArchivos

[en] Número de archivos especificados en la `lpFileNames` matriz.

lpFileNames

[en] Matriz de nombres de ruta de acceso local completos de archivos para los que deshacer una desprotección.

fOptions

[en] Marcas de comando (no utilizadas).

pvOptions

[en] Opciones específicas del complemento de control de código fuente.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Deshacer checkout fue exitoso.
SCC_E_FILENOTCONTROLLED	El archivo seleccionado no está bajo control de código fuente.

VALUE	DESCRIPCIÓN
SCC_E_ACCESSFAILURE	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_NONSPECIFICERROR	Fallo inespecífico. Deshacer la salida no se realizó correctamente.
SCC_E_NOTCHECKEDOUT	El usuario no tiene el archivo desprotegido.
SCC_E_NOTAUTHORIZED	El usuario no puede realizar esta operación.
SCC_E_PROJNOTOPEN	El proyecto no se ha abierto desde el control de código fuente.
SCC_I_OPERATIONCANCELED	La operación se canceló antes de la finalización.

Observaciones

Después de `SCC_STATUS_CHECKEDOUT` esta `SCC_STATUS_MODIFIED` operación, los indicadores y se borrarán para los archivos en los que se realizó la desprotección de deshacer.

Vea también

- [Funciones de API de complemento de control de código fuente](#)

SccUninitialize (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función limpia las asignaciones o conexiones abiertas creadas por una llamada anterior a [ScclInitialize](#) como preparación para apagar el complemento de control de código fuente.

Sintaxis

```
SCCRTN SccUninitialize (  
    LPVOID pvContext  
);
```

Parámetros

pvContext

[en] Puntero a la estructura de contexto del complemento de control de código fuente creada en [ScclInitialize](#).

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	La limpieza se completó correctamente.

Observaciones

El complemento de control de código fuente es responsable de preparar la preparación para apagarse y liberar la memoria que el complemento ha asignado para la estructura de contexto. Se llama a la función una vez para cada instancia dada de un complemento. Una llamada a la [ScclInitialize](#) precede a esta llamada. No hay proyectos que aún estén `SccUninitialize` abiertos en el momento de la llamada a .

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [ScclInitialize](#)

SccWillCreateSccFile (Función)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Esta función determina si el complemento de control de código fuente admite la creación de MSSCCPRJ. SCC para cada uno de los archivos especificados.

Sintaxis

```
SCCRTN SccWillCreateSccFile(  
    LPVOID pContext,  
    LONG nFiles,  
    LPCSTR* lpFileNames,  
    LPBOOL pbSccFiles  
);
```

Parámetros

pContext

[en] El puntero de contexto del complemento de control de código fuente.

nArchivos

[en] El número de nombres `lpFileNames` de archivo incluidos en `pbSccFiles` la matriz, así como la longitud de la matriz.

lpFileNames

[en] Una matriz de nombres de archivo completos para comprobar (la matriz debe ser asignada por el autor de la llamada).

pbSccFiles

[adentro, fuera] Matriz en la que almacenar los resultados.

Valor devuelto

Se espera que la implementación del complemento de control de código fuente de esta función devuelva uno de los siguientes valores:

VALUE	DESCRIPCIÓN
SCC_OK	Correcto.
SCC_E_INVALIDFILEPATH	Una de las rutas de acceso de la matriz no es válida.
SCC_E_NONSPECIFICERROR	Fallo inespecífico.

Observaciones

Esta función se llama con una lista de archivos para determinar si el complemento de control de código fuente proporciona soporte en MSSCCPRJ. SCC para cada uno de los archivos dados (para más información sobre el MSSCCPRJ. SCC, consulte [MSSCCPRJ. SCC](#)). Los complementos de control de código fuente pueden declarar si

tienen la capacidad de crear MSSCCPRJ. SCC declarando `SCC_CAP_SCCFILE` durante la inicialización. El plug-in `TRUE` `FALSE` devuelve o `pbSccFiles` por archivo en la matriz para indicar cuál de los archivos dados tiene MSSCCPRJ. Soporte SCC. Si el complemento devuelve un código de éxito de la función, se respetan los valores de la matriz return. En caso de error, se omite la matriz.

Vea también

- [Funciones de API de complemento de control de código fuente](#)
- [Archivo MSSCCPRJ.SCC](#)

Funciones de devolución de llamada implementadas por el IDE

06/04/2020 • 3 minutes to read • [Edit Online](#)

Para que la integración con el entorno de desarrollo integrado (IDE) sea lo más fluida posible y para proporcionar una experiencia de usuario final unificada, el complemento de control de código fuente puede usar funciones de devolución de llamada implementadas por el IDE. El complemento puede llamar a estas funciones en momentos adecuados durante una operación de control de código fuente para pasar información al IDE; el IDE puede mostrar esta información como elementos incrustados en su interfaz de usuario nativa. El usuario tiene una experiencia menos fragmentada en este escenario que si el complemento empleó su propia interfaz de usuario.

El archivo de encabezado necesario es *scc.h*. La ubicación predeterminada es "Archivos de *programa*", *VSIP 8.0*, *EnvSDK*, *common*, *inc*\\. También se encuentra en la carpeta VSIP que tiene el ejemplo del complemento de control de código fuente en .\|

En esta sección

- [LPTEXTOUTPROC](#) Describe la función de devolución de llamada que usa [SccOpenProject](#) para mostrar mensajes desde el complemento de control de código fuente a través del IDE.
- [POPLISTFUNC](#) Describe la función de devolución de llamada que usa [SccPopulateList](#) cuando el IDE no tiene acceso completo a la información que solo está disponible para el complemento de control de código fuente, como una lista completa de archivos bajo control de versiones.
- [QUERYCHANGESFUNC](#) Describe la función de devolución de llamada que usa la operación [SccQueryChanges](#).
- [POPDIRLISTFUNC](#) Describe la función de devolución de llamada que utiliza la operación [SccPopulateDirList](#).
- [OPTNAMECHANGEPFN](#) Describe la función de devolución de llamada establecida por una llamada a la [SccSetOption](#) que permite que el complemento de control de código fuente para comunicar los cambios de nombre al IDE.

Secciones relacionadas

- [SccOpenProject](#) Abre un proyecto.
- [SccPopulateList](#) Examina la lista de archivos para su estado actual. Además, `pfnPopulate` utiliza la función para notificar al autor `nCommand` de la llamada cuando un archivo no coincide con los criterios para el archivo .
- [SccPopulateDirList](#) Examina una lista de directorios y archivos de un proyecto o proyectos que están bajo control de código fuente. Cada directorio y nombre de archivo encontrados se pasa a una función de devolución de llamada.
- [SccQueryChanges](#) Examina los cambios de nombre realizados en una lista de archivos. Cada nombre de archivo se pasa a una función de devolución de llamada junto con su estado de cambio.
- [SccSetOption](#) Establece una amplia variedad de opciones. Cada opción `SCC_OPT_XXX` comienza con y tiene su propio conjunto definido de valores.
- [Complementos de control de código fuente](#) Describe el contenido de la sección de referencia del SDK de

complemento de Control de código fuente.

LPTEXTOUTPROC

06/04/2020 • 8 minutes to read • [Edit Online](#)

Cuando el usuario ejecuta una operación de control de código fuente desde dentro del entorno de desarrollo integrado (IDE), es posible que el complemento de control de código fuente desee transmitir mensajes de error o de estado relacionados con la operación. El complemento puede mostrar sus propios cuadros de mensaje para este propósito. Sin embargo, para una integración más fluida, el complemento puede pasar cadenas al IDE, que, a continuación, las muestra en su forma nativa de mostrar información de estado. El mecanismo para `LPTEXTOUTPROC` esto es el puntero de función. El IDE implementa esta función (descrita con más detalle a continuación) para mostrar el error y el estado.

El IDE pasa al complemento de control de código fuente `lpTextOutProc` un puntero de función a esta función, como parámetro, al llamar a [sccOpenProject](#). Durante una operación SCC, por ejemplo, en medio de una llamada a la [SccGet](#) que implica muchos archivos, el complemento puede llamar a la `LPTEXTOUTPROC` función, pasando periódicamente cadenas para mostrar. El IDE puede mostrar estas cadenas en una barra de estado, en una ventana de salida o en un cuadro de mensaje independiente, según corresponda. Opcionalmente, el IDE puede mostrar ciertos mensajes con un botón **Cancelar**. Esto permite al usuario cancelar la operación y proporciona al IDE la capacidad de pasar esta información al complemento.

Signature

La función de salida del IDE tiene la siguiente firma:

```
typedef LONG (*LPTEXTOUTPROC) (  
    LPSTR display_string,  
    LONG msg_type  
);
```

Parámetros

`display_string`

Una cadena de texto para mostrar. Esta cadena no debe terminarse con un retorno de carro o un avance de línea.

`msg_type`

El tipo de mensaje. En la tabla siguiente se enumeran los valores admitidos para este parámetro.

VALUE	DESCRIPCIÓN
<code>SCC_MSG_INFO</code> , <code>SCC_MSG_WARNING</code> , <code>SCC_MSG_ERROR</code>	El mensaje se considera Información, Advertencia o Error.
<code>SCC_MSG_STATUS</code>	El mensaje muestra el estado y se puede mostrar en la barra de estado.
<code>SCC_MSG_DOCANCEL</code>	Enviado sin cadena de mensaje.
<code>SCC_MSG_STARTCANCEL</code>	Comienza a mostrar un botón Cancelar .
<code>SCC_MSG_STOPCANCEL</code>	Detiene la visualización de un botón Cancelar .

VALUE	DESCRIPCIÓN
<code>SCC_MSG_BACKGROUND_IS_CANCELLED</code>	Pregunta al IDE si se va a cancelar <code>SCC_MSG_RTN_CANCEL</code> la operación en segundo plano: IDE devuelve si se canceló la operación; de lo <code>SCC_MSG_RTN_OK</code> contrario, devuelve . El <code>display_string</code> parámetro se convierte como un ScMsgDataIsCancelled estructura, que se proporciona mediante el complemento de control de código fuente.
<code>SCC_MSG_BACKGROUND_ON_BEFORE_GET_FILE</code>	Indica al IDE acerca de un archivo antes de recuperarlo del control de versiones. El <code>display_string</code> parámetro se convierte como un ScMsgDataOnBeforeGetFile estructura, que se proporciona mediante el complemento de control de código fuente.
<code>SCC_MSG_BACKGROUND_ON_AFTER_GET_FILE</code>	Indica al IDE acerca de un archivo después de que se ha recuperado del control de versiones. El <code>display_string</code> parámetro se convierte como un ScMsgDataOnAfterGetFile estructura, que se proporciona mediante el complemento de control de código fuente.
<code>SCC_MSG_BACKGROUND_ON_MESSAGE</code>	Indica al IDE el estado actual de una operación en segundo plano. El <code>display_string</code> parámetro se convierte como un ScMsgDataOnMessage estructura, que se proporciona mediante el complemento de control de código fuente.

Valor devuelto

VALUE	DESCRIPCIÓN
<code>SCC_MSG_RTN_OK</code>	La cadena se mostró o la operación se completó correctamente.
<code>SCC_MSG_RTN_CANCEL</code>	El usuario desea cancelar la operación.

Ejemplo

Supongamos que el IDE llama a [La SccGet](#) con veinte nombres de archivo. El complemento de control de código fuente desea evitar la cancelación de la operación en medio de un archivo get. Después de obtener `lpTextOutProc` cada archivo, llama, pasándole la `SCC_MSG_DOCANCEL` información de estado de cada archivo y envía un mensaje si no tiene ningún estado para informar. Si en algún momento el complemento recibe `SCC_MSG_RTN_CANCEL` un valor devuelto del IDE, cancela la operación get inmediatamente, de modo que no se recuperan más archivos.

Estructuras

ScMsgDataIsCancelled

```
typedef struct {
    DWORD dwBackgroundOperationID;
} ScMsgDataIsCancelled;
```

Esta estructura se `SCC_MSG_BACKGROUND_IS_CANCELLED` envía con el mensaje. Se utiliza para comunicar el identificador de la operación en segundo plano que se canceló.

SccMsgDataOnBeforeGetFile

```
typedef struct {  
    DWORD dwBackgroundOperationID;  
    PCSTR szFile;  
} SccMsgDataOnBeforeGetFile;
```

Esta estructura se `SCC_MSG_BACKGROUND_ON_BEFORE_GET_FILE` envía con el mensaje. Se utiliza para comunicar el nombre del archivo a punto de recuperarse y el identificador de la operación en segundo plano que realiza la recuperación.

SccMsgDataOnAfterGetFile

```
typedef struct {  
    DWORD dwBackgroundOperationID;  
    PCSTR szFile;  
    SCRTN sResult;  
} SccMsgDataOnAfterGetFile;
```

Esta estructura se `SCC_MSG_BACKGROUND_ON_AFTER_GET_FILE` envía con el mensaje. Se utiliza para comunicar el resultado de la recuperación del archivo especificado, así como el identificador de la operación en segundo plano que realizó la recuperación. Vea los valores devueltos para el [SccGet](#) para lo que se puede dar como resultado.

SccMsgDataOnMessage

```
typedef struct {  
    DWORD dwBackgroundOperationID;  
    PCSTR szMessage;  
    BOOL bIsError;  
} SccMsgDataOnMessage;
```

Esta estructura se `SCC_MSG_BACKGROUND_ON_MESSAGE` envía con el mensaje. Se utiliza para comunicar el estado actual de una operación en segundo plano. El estado se expresa como una cadena que se `bIsError` mostrará en el IDE `TRUE` e indica la gravedad del mensaje (para un mensaje de error; `FALSE` para una advertencia o para un mensaje informativo). También se proporciona el identificador de la operación en segundo plano que envía el estado.

Ejemplo de código

Este es un breve `LPTTEXTOUTPROC` ejemplo `SCC_MSG_BACKGROUND_ON_MESSAGE` de llamada para enviar el mensaje, que muestra cómo convertir la estructura para la llamada.

```
LONG SendStatusMessage(  
    LPTEXTOUTPROC pTextOutProc,  
    DWORD         dwBackgroundID,  
    LPCTSTR       pStatusMsg,  
    BOOL          bIsError)  
{  
    SccMsgDataOnMessage msgData = { 0 };  
    LONG                result  = 0;  
  
    msgData.dwBackgroundOperationID = dwBackgroundID;  
    msgData.szMessage               = pStatusMsg;  
    msgData.bIsError                = bIsError;  
  
    result = pTextOutProc(reinterpret_cast<LPCTSTR>(&msgData), SCC_MSG_BACKGROUND_ON_MESSAGE);  
    return result;  
}
```

Vea también

- [Funciones de devolución de llamada implementadas por el IDE](#)
- [Complementos de control de código fuente](#)

POPLISTFUNC

06/04/2020 • 4 minutes to read • [Edit Online](#)

Esta devolución de llamada se proporciona a la [SccPopulateList](#) por el IDE y se utiliza el complemento `SccPopulateList` de control de código fuente para actualizar una lista de archivos o directorios (también se proporciona a la función).

Cuando un usuario elige el **get** comando en el IDE, el IDE muestra un cuadro de lista de todos los archivos que el usuario puede obtener. Desafortunadamente, el IDE no conoce la lista exacta de todos los archivos que el usuario puede obtener; solo el plug-in tiene esta lista. Si otros usuarios han agregado archivos al proyecto de control de código fuente, estos archivos deben aparecer en la lista, pero el IDE no conoce acerca de ellos. El IDE crea una lista de los archivos que cree que el usuario puede obtener. Antes de que muestre esta lista al usuario, llama a la [SccPopulateList](#), dando al complemento de control de código fuente la oportunidad de agregar y eliminar archivos de la lista.

Signature

El complemento de control de código fuente modifica la lista llamando a una función implementada por IDE con el siguiente prototipo:

```
typedef BOOL (*POPLISTFUNC) (  
    LPVOID pvCallerData,  
    BOOL fAddRemove,  
    LONG nStatus,  
    LPSTR lpFileName  
);
```

Parámetros

`pvCallerData` `pvCallerData` El parámetro pasado por el llamador (el IDE) a [SccPopulateList](#). El complemento de control de código fuente debe asumir nada sobre el contenido de este parámetro.

`fAddRemove` `TRUE` If `lpFileName`, es un archivo que se debe agregar a la lista de archivos. Si `FALSE` `lpFileName`, es un archivo que se debe eliminar de la lista de archivos.

`nEstado` Estado `lpFileName` de (una `SCC_STATUS` combinación de los bits; consulte Código de [estado](#) de archivo para obtener más información).

`lpFileName` Ruta de acceso de directorio completa del nombre de archivo que se va a agregar o eliminar de la lista.

Valor devuelto

VALUE	DESCRIPCIÓN
<code>TRUE</code>	El complemento puede seguir llamando a esta función.
<code>FALSE</code>	Ha habido un problema en el lado IDE (como una situación de memoria insuficiente). El plug-in debe detener el funcionamiento.

Observaciones

Para cada archivo que el complemento de control de código fuente desea agregar o eliminar `lpFileName` de la lista de archivos, llama a esta función, pasando el archivo . La `fAddRemove` marca indica un nuevo archivo para agregar a la lista o un archivo antiguo para eliminar. El `nStatus` parámetro proporciona el estado del archivo. Cuando el complemento SCC ha terminado de agregar y eliminar archivos, vuelve de la llamada [SccPopulateList](#).

NOTE

El `SCC_CAP_POPULATELIST` bit de funcionalidad es necesario para Visual Studio.

Vea también

- [Funciones de devolución de llamada implementadas por el IDE](#)
- [Complementos de control de código fuente](#)
- [SccPopulateList](#)
- [Código de estado del archivo](#)

QUERYCHANGESFUNC

06/04/2020 • 3 minutes to read • [Edit Online](#)

Se trata de una función de devolución de llamada utilizada por la operación [SccQueryChanges](#) para enumerar una colección de nombres de archivo y determinar el estado de cada archivo.

A [SccQueryChanges](#) la función se le da [QUERYCHANGESFUNC](#) una lista de archivos y un puntero a la devolución de llamada. El complemento de control de código fuente enumera la lista dada y proporciona el estado (a través de esta devolución de llamada) para cada archivo de la lista.

Signature

```
typedef BOOL (*QUERYCHANGESFUNC)(
    LPVOID pvCallerData,
    QUERYCHANGESDATA * pChangesData
);
```

Parámetros

pvCallerData

[en] Parámetro [pvCallerData](#) pasado por el llamador (el IDE) a [SccQueryChanges](#). El complemento de control de código fuente no debe hacer ninguna suposición sobre el contenido de este valor.

pChangesData

[en] Puntero a una estructura [QUERYCHANGESDATA](#) que describe los cambios en un archivo.

Valor devuelto

El IDE devuelve un código de error adecuado:

VALUE	DESCRIPCIÓN
SCC_OK	Continúe el procesamiento.
SCC_I_OPERATIONCANCELED	Detiene el procesamiento.
SCC_E_xxx	Cualquier error SCC apropiado debe detener el procesamiento.

Estructura QUERYCHANGESDATA

La estructura pasada para cada archivo tiene el siguiente aspecto:

```

struct QUERYCHANGESDATA_A
{
    DWORD    dwSize;
    LPCSTR   lpFileName;
    DWORD    dwChangeType;
    LPCSTR   lpLatestName;
};

typedef struct QUERYCHANGESDATA_A QUERYCHANGESDATA;

struct QUERYCHANGESDATA_W
{
    DWORD    dwSize;
    LPCWSTR  lpFileName;
    DWORD    dwChangeType;
    LPCWSTR  lpLatestName;
};

```

dwSize Tamaño de esta estructura (en bytes).

lpFileName El nombre de archivo original para este elemento.

dwChangeType Código que indica el estado del archivo:

CÓDIGO	DESCRIPCIÓN
SCC_CHANGE_UNKNOWN	No puedo decir lo que ha cambiado.
SCC_CHANGE_UNCHANGED	No hay cambios de nombre para este archivo.
SCC_CHANGE_DIFFERENT	Archivo con una identidad diferente, pero el mismo nombre existe en la base de datos.
SCC_CHANGE_NONEXISTENT	El archivo no existe ni en la base de datos ni localmente.
SCC_CHANGE_DATABASE_DELETED	Archivo eliminado en la base de datos.
SCC_CHANGE_LOCAL_DELETED	Archivo eliminado localmente, pero el archivo sigue existiendo en la base de datos. Si esto no se SCC_CHANGE_DATABASE_ADDED puede determinar, devuelva .
SCC_CHANGE_DATABASE_ADDED	Archivo agregado a la base de datos pero no existe localmente.
SCC_CHANGE_LOCAL_ADDED	El archivo no existe en la base de datos y es un nuevo archivo local.
SCC_CHANGE_RENAMED_TO	Archivo renombrado o movido en lpLatestName la base de datos como .
SCC_CHANGE_RENAMED_FROM	Archivo renombrado o movido en lpLatestName la base de datos desde ; si esto es demasiado caro de rastrear, SCC_CHANGE_DATABASE_ADDED devuelva una bandera diferente, como .

lpLatestName El nombre de archivo actual para este elemento.

Vea también

- [Funciones de devolución de llamada implementadas por el IDE](#)
- [SccQueryChanges](#)
- [Códigos de error](#)

POPDIRLISTFUNC

06/04/2020 • 2 minutes to read • [Edit Online](#)

Se trata de una función de devolución de llamada dada a la [función SccPopulateDirList](#) para actualizar una colección de directorios y (opcionalmente) nombres de archivo para averiguar cuáles están bajo control de código fuente.

La `POPDIRLISTFUNC` devolución de llamada debe llamarse solo para los directorios y nombres de archivo (en la lista dada a la `SccPopulateDirList` función) que realmente están bajo control de código fuente.

Signature

```
typedef BOOL (*POPDIRLISTFUNC)(
    LPVOID pvCallerData,
    BOOL bFolder,
    LPCSTR lpDirectoryOrFileName
);
```

Parámetros

`pvCallerData`

[en] Valor de usuario dado a [SccPopulateDirList](#).

`bCarpeta`

[en] `TRUE` si el `lpDirectoryOrFileName` nombre es un directorio; de lo contrario, el nombre es un nombre de archivo.

`lpDirectoryOrFileName`

[en] Ruta de acceso local completa a un directorio o nombre de archivo que está bajo control de código fuente.

Valor devuelto

El IDE devuelve un código de error adecuado:

VALUE	DESCRIPCIÓN
SCC_OK	Continúe el procesamiento.
SCC_I_OPERATIONCANCELED	Detiene el procesamiento.
SCC_E_xxx	Cualquier error de control de código fuente adecuado debe detener el procesamiento.

Observaciones

Si `fOptions` el parámetro `SccPopulateDirList` de `SCC_PDL_INCLUDEFILES` la función contiene la marca, la lista posiblemente contendrá nombres de archivo, así como nombres de directorio.

Vea también

- [Funciones de devolución de llamada implementadas por el IDE](#)
- [SccPopulateDirList](#)
- [Códigos de error](#)

OPTNAMECHANGEPFN

06/04/2020 • 2 minutes to read • [Edit Online](#)

Se trata de una función de devolución de llamada `SCC_OPT_NAMECHANGEPFN` especificada en una llamada a la [SccSetOption](#) (mediante la opción) y se utiliza para comunicar los cambios de nombre realizados por el complemento de control de código fuente al IDE.

Signature

```
typedef void (*OPTNAMECHANGEPFN)(  
    LPVOID pvCallerData,  
    LPCSTR pszOldName,  
    LPCSTR pszNewName  
);
```

Parámetros

`pvCallerData`

[en] Valor de usuario especificado en una llamada anterior a `SCC_OPT_USERDATA` [SccSetOption](#) (mediante la opción).

`pszOldName`

[en] El nombre original del archivo.

`pszNewName`

[en] El nombre al que se ha cambiado el nombre del archivo.

Valor devuelto

Ninguno.

Observaciones

Si se cambia el nombre de un archivo durante una operación de control de código fuente, el complemento de control de código fuente puede notificar al IDE sobre el cambio de nombre a través de esta devolución de llamada.

Si el IDE no admite esta devolución de llamada, no llamará a [la SccSetOption](#) para especificarlo. Si el complemento no admite esta devolución `SCC_E_OPNOTSUPPORTED` de `SccSetOption` llamada, se devolverá de la función cuando el IDE intente establecer la devolución de llamada.

Vea también

- [Funciones de devolución de llamada implementadas por el IDE](#)
- [SccSetOption](#)

Enumerators

06/04/2020 • 2 minutes to read • [Edit Online](#)

En esta sección se enumeran los tipos de datos del enumerador en la API de complemento de Control de código fuente que debe conocer el complemento de control de código fuente.

En esta sección

- [Código de comando](#) Enumera las opciones para el [SccGetCommandOptions](#) y [SccPopulateList](#) funciones.
- [Mensaje](#) Enumera los indicadores utilizados para la devolución de llamada de impresión, [LPTEXTOUTPROC](#).
- [Código de estado del archivo](#) Contiene valores constantes con nombre que especifican el estado de un archivo bajo control de código fuente.
- [Código de estado del directorio](#) Contiene valores constantes con nombre que especifican el estado de un directorio bajo control de código fuente.

Secciones relacionadas

- [Crear un complemento de control de código fuente](#) Define el SDK de complemento de Control de código fuente y describe los recursos incluidos.
- [SccGetCommandOptions](#) Solicita al usuario opciones avanzadas para el comando dado.
- [SccPopulateList](#) Examina la lista de archivos para su estado actual. Además, `pfnPopulate` utiliza la función para notificar al autor `nCommand` de la llamada cuando un archivo no coincide con los criterios para el archivo .
- [LPTEXTOUTPROC](#) Describe la función de devolución de llamada que usa [SccOpenProject](#) para mostrar mensajes desde el complemento de control de código fuente a través del IDE.
- [Complementos de control de código fuente](#) Proporciona una lista completa de todos los elementos de la API de complemento de Control de código fuente.

Enumerador de código de comando

06/04/2020 • 2 minutes to read • [Edit Online](#)

Este enumerador se utiliza en las opciones de [SccGetCommandOptions](#) y [SccPopulateList](#) para indicar el comando para el que se especifican las opciones.

Sintaxis

```
enum SCCCOMMAND {  
    SCC_COMMAND_GET,  
    SCC_COMMAND_CHECKOUT,  
    SCC_COMMAND_CHECKIN,  
    SCC_COMMAND_UNCHECKOUT,  
    SCC_COMMAND_ADD,  
    SCC_COMMAND_REMOVE,  
    SCC_COMMAND_DIFF,  
    SCC_COMMAND_HISTORY,  
    SCC_COMMAND_RENAME,  
    SCC_COMMAND_PROPERTIES,  
    SCC_COMMAND_OPTIONS  
};
```

Miembros

SCC_COMMAND_GET Corresponde a [SccGet](#).

SCC_COMMAND_CHECKOUT Corresponde al [SccCheckout](#).

SCC_COMMAND_CHECKIN Corresponde al [SccCheckin](#).

SCC_COMMAND_UNCHECKOUT Corresponde al [SccUncheckout](#).

SCC_COMMAND_ADD Corresponde al [SccAdd](#).

SCC_COMMAND_REMOVE Corresponde a la [SccRemove](#).

SCC_COMMAND_DIFF Corresponde al [SccDiff](#).

SCC_COMMAND_HISTORY Corresponde al [SccHistory](#).

SCC_COMMAND_RENAME Corresponde a [SccRename](#).

SCC_COMMAND_PROPERTIES Corresponde a [SccProperties](#).

SCC_COMMAND_OPTIONS Corresponde a [SccSetOption](#).

Vea también

- [Complementos de control de código fuente](#)
- [SccGetCommandOptions](#)
- [SccPopulateList](#)

Enumerador de mensajes

06/04/2020 • 2 minutes to read • [Edit Online](#)

Los siguientes indicadores `TEXTOUTPROC` se utilizan para la función, que es una función de devolución de llamada que el IDE proporciona cuando llama a la [SccOpenProject](#) (consulte [LPTEXTOUTPROC](#) para obtener más información sobre la función de devolución de llamada).

Si se pide al IDE que cancele el proceso, puede recibir uno de los mensajes de cancelación. En este caso, el complemento `SCC_MSG_STARTCANCEL` de control de código fuente se usa para pedir al IDE que muestre el botón **Cancelar**. Después de esto, se puede enviar cualquier conjunto de mensajes normales. Si cualquiera de `SCC_MSG_RTN_CANCEL` estas devuelve, el complemento cierra la operación y devuelve. El complemento también sondea `SCC_MSG_DOCANCEL` periódicamente para determinar si el usuario ha cancelado la operación. Cuando se realizan todas las operaciones, o si el `SCC_MSG_STOPCANCEL` usuario ha cancelado, el complemento envía. Los `SCC_MSG_INFO` tipos, `SCC_MSG_WARNING` y `SCC_MSG_ERROR` se usan para los mensajes que se muestran en la lista de mensajes de desplazamiento. `SCC_MSG_STATUS` es un tipo especial que indica que el texto debe aparecer en una barra de estado o un área de visualización temporal. No permanece permanentemente en la lista.

Sintaxis

```
enum {
    SCC_MSG_RTN_CANCEL = -1,
    SCC_MSG_RTN_OK = 0,
    SCC_MSG_INFO = 1
    SCC_MSG_WARNING,
    SCC_MSG_ERROR,
    SCC_MSG_STATUS,
    SCC_MSG_DOCANCEL,
    SCC_MSG_STARTCANCEL,
    SCC_MSG_STOPCANCEL
};
```

Miembros

`SCC_MSG_RTN_CANCEL` Devolver de devolución de llamada para indicar la cancelación.

`SCC_MSG_RTN_OK` Devolver de la devolución de llamada para continuar.

`SCC_MSG_INFO` mensaje es informativo.

`SCC_MSG_WARNING` mensaje es una advertencia.

`SCC_MSG_ERROR` mensaje es un error.

`SCC_MSG_STATUS` Mensaje está destinado a la barra de estado.

`SCC_MSG_DOCANCEL` Sin texto; IDE `SCC_MSG_RTN_OK` devuelve `SCC_MSG_RTN_CANCEL` o .

`SCC_MSG_STARTCANCEL` Inicia un bucle de cancelación.

`SCC_MSG_STOPCANCEL` Detiene el bucle de cancelación.

Vea también

- [Complementos de control de código fuente](#)

- [LPTEXTOUTPROC](#)

Enumerador de código de estado de archivo

06/04/2020 • 2 minutes to read • [Edit Online](#)

El `ScStatus` enumerador contiene valores constantes con nombre que especifican el estado de un archivo en el sistema de control de código fuente. Esta enumeración la usan `ScQueryInfo` y la `POPLISTFUNC` función de devolución de llamada (consulte `POPLISTFUNC` para obtener más información).

Sintaxis

```
enum ScStatus {
    SCC_STATUS_INVALID           = -1L,
    SCC_STATUS_NOTCONTROLLED     = 0x0000L,
    SCC_STATUS_CONTROLLED        = 0x0001L,
    SCC_STATUS_CHECKEDOUT        = 0x0002L,
    SCC_STATUS_OUTOTHER          = 0x0004L,
    SCC_STATUS_OUTEXCLUSIVE      = 0x0008L,
    SCC_STATUS_OUTMULTIPLE       = 0x0010L,
    SCC_STATUS_OUTOFDATE         = 0x0020L,
    SCC_STATUS_DELETED           = 0x0040L,
    SCC_STATUS_LOCKED            = 0x0080L,
    SCC_STATUS_MERGED            = 0x0100L,
    SCC_STATUS_SHARED            = 0x0200L,
    SCC_STATUS_PINNED            = 0x0400L,
    SCC_STATUS_MODIFIED          = 0x0800L,
    SCC_STATUS_OUTBYUSER         = 0x1000L,
    SCC_STATUS_NOMERGE           = 0x2000L,
    SCC_STATUS_RESERVED_1        = 0x4000L,
    SCC_STATUS_RESERVED_2        = 0x8000L
};
```

Miembros

`SCC_STATUS_INVALID` no se pudo obtener el estatus; no confíe en él.

`SCC_STATUS_NOTCONTROLLED` archivo no está bajo control de código fuente.

`SCC_STATUS_CONTROLLED` archivo está bajo control de código fuente.

`SCC_STATUS_CHECKEDOUT` Desprotegido por el usuario actual en el disco local.

`SCC_STATUS_OUTOTHER` archivo es desprotegido por otro usuario.

`SCC_STATUS_OUTEXCLUSIVE` archivo está desprotegido exclusivamente.

`SCC_STATUS_OUTMULTIPLE` archivo es desprotegido por más de un usuario.

`SCC_STATUS_OUTOFDATE` El archivo no es el más reciente.

`SCC_STATUS_DELETED` archivo se ha eliminado del proyecto.

`SCC_STATUS_LOCKED` archivo está bloqueado; no se permiten más versiones.

`SCC_STATUS_MERGED` archivo se ha fusionado pero aún no se ha corregido/verificado.

`SCC_STATUS_SHARED` archivo se comparte entre proyectos.

`SCC_STATUS_PINNED` archivo se comparte con una versión explícita.

SCC_STATUS_MODIFIED archivo se ha modificado/roto/violado.

SCC_STATUS_OUTBYUSER usuario actual desprotege SCC_STATUS_OUTBYUSER archivo.

SCC_STATUS_NOMERGE archivo nunca se puede combinar con y no es necesario guardar antes de un GET.

SCC_STATUS_RESERVED_1 Reservado para uso interno.

SCC_STATUS_RESERVED_2 Reservado para uso interno.

Vea también

- [Complementos de control de código fuente](#)
- [SccQueryInfo](#)
- [POPLISTFUNC](#)

Enumerador de código de estado de directorio

06/04/2020 • 2 minutes to read • [Edit Online](#)

El `SccDirStatus` enumerador contiene valores constantes con nombre que especifican el estado de un directorio en el sistema de control de código fuente. [SccDirQueryInfo](#) utiliza esta enumeración. Esto se introdujo en la versión 1.2 de la API de complemento de Control de código fuente.

Sintaxis

```
enum SccDirStatus {  
    SCC_DIRSTATUS_INVALID      = -1L,  
    SCC_DIRSTATUS_NOTCONTROLLED = 0x0000L,  
    SCC_DIRSTATUS_CONTROLLED   = 0x0001L,  
    SCC_DIRSTATUS_EMPTYPROJ    = 0x0002L  
};
```

Miembros

SCC_DIRSTATUS_INVALID no se pudo obtener el estatus; no confíe en él.

SCC_DIRSTATUS_NOTCONTROLLED Directory no está bajo control de código fuente.

directorio de SCC_DIRSTATUS_CONTROLLED está bajo control de código fuente.

SCC_DIRSTATUS_EMPTYPROJ proyecto correspondiente a este directorio está vacío.

Vea también

- [Complementos de control de código fuente](#)
- [SccDirQueryInfo](#)

Banderas de capacidad

06/04/2020 • 4 minutes to read • [Edit Online](#)

Las SCC_CAP_ marcas *xxx* son indicadores de bits que se utilizan para indicar las capacidades de un complemento de control de código fuente. Los SCC_EXCAP_ los indicadores *xxx* son indicadores incrementales que indican capacidades extendidas y se resuelven en valores enteros.

CÓDIGO DE CAPACIDAD	VALUE	DESCRIPCIÓN
SCC_CAP_REMOVE	0x00000001L	Admite el comando SccRemove y.
SCC_CAP_RENAME	0x00000002L	Admite el comando SccRename y el comando.
SCC_CAP_DIFF	0x00000004L	Admite el comando SccDiff y.
SCC_CAP_HISTORY	0x00000008L	Admite sccHistory y el comando.
SCC_CAP_PROPERTIES	0x00000010L	Admite SccProperties y el comando.
SCC_CAP_RUNSCC	0x00000020L	Admite el comando SccRunScc y.
SCC_CAP_GETCOMMANDOPTIONS	0x00000040L	Admite el SccGetCommandOptions y el comando.
SCC_CAP_QUERYINFO	0x00000080L	Admite el SccQueryInfo y el comando.
SCC_CAP_GETEVENTS	0x00000100L	Admite SccGetEvents y el comando.
SCC_CAP_GETPROJPATH	0x00000200L	Admite SccGetProjPath y el comando.
SCC_CAP_ADDFROMSCC	0x00000400L	Admite el comando SccAddFromScc y.
SCC_CAP_COMMENTCHECKOUT	0x00000800L	Admite un comentario sobre el pago.
SCC_CAP_COMMENTCHECKIN	0x00001000L	Admite un comentario sobre el check-in.
SCC_CAP_COMMENTADD	0x00002000L	Admite un comentario en Agregar.
SCC_CAP_COMMENTREMOVE	0x00004000L	Admite un comentario en Eliminar.
SCC_CAP_TEXTOUT	0x00008000L	Escribe texto en una función de salida proporcionada por IDE.
SCC_CAP_ADD_STORELATEST	0x00200000L	Admite el almacenamiento de archivos sin deltas.
SCC_CAP_HISTORY_MULTIFILE	0x00400000L	Admite el historial de varios archivos.

CÓDIGO DE CAPACIDAD	VALUE	DESCRIPCIÓN
<code>SCC_CAP_IGNORECASE</code>	0x00800000L	Admite la comparación de archivos que no distinguen mayúsculas de minúsculas.
<code>SCC_CAP_IGNORESPACE</code>	0x01000000L	Admite la comparación de archivos que omite el espacio en blanco.
<code>SCC_CAP_POPULATELIST</code>	0x02000000L	Admite la búsqueda de archivos adicionales.
<code>SCC_CAP_COMMENTPROJECT</code>	0x04000000L	Admite comentarios sobre la creación de proyectos.
<code>SCC_CAP_DIFFALWAYS</code>	0x10000000L	Admite diferencias en todos los estados si están bajo control.
<code>SCC_CAP_GET_NOUI</code>	0x200000000L	El complemento no admite una interfaz de usuario para Get, pero el IDE puede llamar a SccGet .
<code>SCC_CAP_REENTRANT</code>	0x40000000L	El plug-in es reentrante y seguro para subprocesos. En la versión 1.0, no se suponía que los complementos fueran reentrantes y seguros para subprocesos. Si un complemento 1.1 establece este bit, el host puede abrir varios proyectos en paralelo.

Bits de capacidad añadidos en la versión 1.2

CÓDIGO DE CAPACIDAD	VALUE	DESCRIPCIÓN
<code>SCC_CAP_CREATESUBPROJECT</code>	0x00010000L	Admite SccCreateSubProject .
<code>SCC_CAP_GETPARENTPROJECT</code>	0x00020000L	Admite SccGetParentProjectPath .
<code>SCC_CAP_BATCH</code>	0x00040000L	Admite SccBeginBatch y SccEndBatch .
<code>SCC_CAP_DIRECTORYSTATUS</code>	0x00080000L	Admite SccDirQueryInfo .
<code>SCC_CAP_DIRECTORYDIFF</code>	0x00100000L	Soporta el SccDirDiff .
<code>SCC_CAP_MULTICHECKOUT</code>	0x08000000L	Admite varias desprotecciones en un archivo y SccIsMultiCheckoutEnabled .
<code>SCC_CAP_SCCFILE</code>	0x800000000L	Admite el archivo <i>MSSCCPRJ.SCC</i> (sujeto a anulación de usuario/administrador) y SccWillCreateSccFile .

Bits de capacidad añadidos en la versión 1.3

Estos indicadores se pasan de uno en uno a la función [SccGetExtendedCapabilities](#) para determinar si se admite

la funcionalidad.

CÓDIGO DE CAPACIDAD AMPLIADA	VALUE	DESCRIPCIÓN
<code>SCC_EXCAP_CHECKOUT_LOCALVER</code>	1	Admite <code>SCC_CHECKOUT_LOCALVER</code> la opción de desprotecciones.
<code>SCC_EXCAP_BACKGROUND_GET</code>	2	Admite SccBackgroundGet .
<code>SCC_EXCAP_ENUM_CHANGED_FILES</code>	3	Admite SccEnumChangedFiles .
<code>SCC_EXCAP_POPULATELIST_DIR</code>	4	Admite la búsqueda de directorios adicionales.
<code>SCC_EXCAP_QUERYCHANGES</code>	5	Admite la enumeración de cambios de archivo.
<code>SCC_EXCAP_ADD_FILES_FROM_SCC</code>	6	Admite SccAddFilesFromSCC .
<code>SCC_EXCAP_GET_USER_OPTIONS</code>	7	Admite SccGetUserOption .
<code>SCC_EXCAP_THREADSAFE_QUERY_INFO</code>	8	Admite la llamada a <code>SccQueryInfo</code> en varios subprocesos.
<code>SCC_EXCAP_REMOVE_DIR</code>	9	Admite la función <code>SccRemoveDir</code> .
<code>SCC_EXCAP_DELETE_CHECKEDOUT</code>	10	Puede eliminar los archivos desprotegidos.
<code>SCC_EXCAP_RENAME_CHECKEDOUT</code>	11	Puede cambiar el nombre de los archivos desprotegidos.

Vea también

- [Complementos de control de código fuente](#)

Bitflags utilizados por comandos específicos

06/04/2020 • 7 minutes to read • [Edit Online](#)

El comportamiento de varias funciones de la API de complemento de Control de código fuente se puede modificar estableciendo uno o varios bits en un único valor. Estos valores se conocen como bitflags. Los distintos bitflags utilizados por la API de complemento de Control de código fuente se detallan aquí, agrupados por la función que los utiliza.

Bandera desprotegida

Este indicador se puede establecer para [SccAdd](#) o [SccCheckin](#).

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_KEEP_CHECKEDOUT</code>	0x1000	Mantenga el archivo desprotegido.

Añadir banderas

Estos indicadores son utilizados por el [SccAdd](#).

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_FILETYPE_AUTO</code>	0x00	Se espera que el complemento de control de código fuente detecte automáticamente si el archivo es de texto o binario.
<code>SCC_FILETYPE_TEXT</code>	0x01	El tipo de archivo es texto.
<code>SCC_FILETYPE_BINARY</code>	0x04	El tipo de archivo es binario. Nota: <code>SCC_FILETYPE_TEXT</code> y <code>SCC_FILETYPE_BINARY</code> son mutuamente excluyentes. Ajuste exactamente uno o ninguno.
<code>SCC_ADD_STORELATEST</code>	0x02	Almacene solo la versión más reciente (sin deltas).

Banderas Diff

El [SccDiff](#) utiliza estos indicadores para definir el ámbito de una operación diff. Las `SCC_DIFF_QD_XXX` banderas son mutuamente excluyentes. Si se especifica alguno de ellos, no se debe proporcionar ningún comentario visual. En un "diff rápido" (QD), el plug-in no determina cómo el archivo es diferente, sólo si es diferente. Si no se especifica ninguna de estas marcas, se realiza una "diferencia visual"; se calculan y muestran las diferencias de archivo detalladas. Si no se admite el QD solicitado, el complemento pasa al siguiente mejor. Por ejemplo, si el IDE solicita una suma de comprobación y el complemento no lo admite, el complemento realiza una comprobación de contenido completo (aún mucho más rápido que una pantalla visual).

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_DIFF_IGNORECASE</code>	0x0002	Ignore las diferencias de mayúsculas y minúsculas.
<code>SCC_DIFF_IGNORESPACE</code>	0x0004	Ignore las diferencias de espacio en blanco. Nota: Las <code>SCC_DIFF_IGNORECASE</code> <code>SCC_DIFF_IGNORESPACE</code> marcas y son marcas de bits opcionales.
<code>SCC_DIFF_QD_CONTENTS</code>	0x0010	QD comparando todo el contenido del archivo.
<code>SCC_DIFF_QD_CHECKSUM</code>	0x0020	QD por suma de comprobación.
<code>SCC_DIFF_QD_TIME</code>	0x0040	QD por marca de fecha/hora del archivo.
<code>SCC_DIFF_QUICK_DIFF</code>	0x0070	Esta es una máscara usada para marcar todos los bitflags QD. No debe pasarse a una función; los tres bitflags QD son mutuamente excluyentes. QD siempre significa que no se muestra ninguna interfaz de usuario.

Marca PopulateList

Este indicador es utilizado por el `fOptions` [SccPopulateList](#) en el parámetro.

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_PL_DIR</code>	0x00000001L	El IDE pasa directorios, no archivos.

Marcas PopulateDirList

Estos indicadores son utilizados por el `fOptions` [SccPopulateDirList](#) en el parámetro.

VALOR DE LA OPCIÓN	VALUE	DESCRIPCIÓN
<code>SCC_PDL_ONELEVEL</code>	0x0000	Examine solo un nivel de directorios para directorios (este es el valor predeterminado).
<code>SCC_PDL_RECURSIVE</code>	0x0001	Examine recursivamente todos los directorios de cada directorio dado.
<code>SCC_PDL_INCLUDEFILES</code>	0x0002	Incluya los nombres de archivo en el proceso de examen.

Banderas OpenProject

Estos indicadores son utilizados por el `dwFlags` [SccOpenProject](#) en el parámetro.

VALOR DE LA OPCIÓN	VALUE	DESCRIPCIÓN
SCC_OP_CREATEIFNEW	0x00000001L	Si project no existe en el control de código fuente, créelo. Si no se establece esta marca, solicite <code>SCC_OP_SILENTOPEN</code> al usuario que cree el proyecto (a menos que se especifique flag).
SCC_OP_SILENTOPEN	0x00000002L	No pida al usuario que cree un proyecto; sólo <code>SCC_E_UNKNOWNPROJECT</code> volver .

Obtener banderas

Estos indicadores los usan [SccGet](#) y [SccCheckout](#).

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_GET_ALL</code>	0x00000001L	El IDE pasa directorios, no archivos: obtener todos los archivos en estos directorios.
<code>SCC_GET_RECURSIVE</code>	0x00000002L	El IDE pasa directorios: obtenga estos directorios y todos sus subdirectorios.

nValores de la opción

Estos indicadores son utilizados por el `nOption` [SccSetOption](#) en el parámetro.

MARCA	VALUE	DESCRIPCIÓN
<code>SCC_OPT_EVENTQUEUE</code>	0x00000001L	Establezca el estado de la cola de eventos.
<code>SCC_OPT_USERDATA</code>	0x00000002L	Especifique los <code>SCC_OPT_NAMECHANGEPFN</code> datos de usuario para .
<code>SCC_OPT_HASCANCELMODE</code>	0x00000003L	El IDE puede controlar la cancelación.
<code>SCC_OPT_NAMECHANGEPFN</code>	0x00000004L	Establezca una devolución de llamada para los cambios de nombre.
<code>SCC_OPT_SCCCHECKOUTONLY</code>	0x00000005L	Deshabilite la desprotección de la interfaz de usuario del complemento de control de código fuente y no establezca el directorio de trabajo.
<code>SCC_OPT_SHARESUBPROJ</code>	0x00000006L	Agregar desde el sistema de control de código fuente para especificar un directorio de trabajo. Intente compartir en el proyecto asociado si es un descendiente directo.

dwVal bitflags

Estos indicadores son utilizados por el `dwVal` [SccSetOption](#) en el parámetro.

MARCA	VALUE	DESCRIPCIÓN	UTILIZADO NOPTION POR VALOR
<code>SCC_OPT_EQ_DISABLE</code>	0x00L	Suspende la actividad de la cola de eventos.	<code>SCC_OPT_EVENTQUEUE</code>
<code>SCC_OPT_EQ_ENABLE</code>	0x01L	Habilita el registro de la cola de eventos.	<code>SCC_OPT_EVENTQUEUE</code>
<code>SCC_OPT_HCM_NO</code>	0L	(Predeterminado) No tiene modo de cancelación; plug-in debe suministrarse si lo desea.	<code>SCC_OPT_HASCANCELMODE</code>
<code>SCC_OPT_HCM_YES</code>	1 L	IDE controla la cancelación.	<code>SCC_OPT_HASCANCELMODE</code>
<code>SCC_OPT_SCO_NO</code>	0L	(Predeterminado) Ok para desproteger desde la interfaz de usuario del complemento; directorio de trabajo.	<code>SCC_OPT_SCCCHECKOUTONLY</code>
<code>SCC_OPT_SCO_YES</code>	1 L	Sin desprotección de la interfaz de usuario del plug-in, sin directorio de trabajo.	<code>SCC_OPT_SCCCHECKOUTONLY</code>

Vea también

- [Complementos de control de código fuente](#)

Códigos de error

06/04/2020 • 6 minutes to read • [Edit Online](#)

Cuando una función de API de complemento de Control de código fuente devuelve un error, se espera que sea uno de los siguientes códigos de error. Todos los errores son negativos, las advertencias o los códigos de error informativos son positivos y el éxito es 0.

CÓDIGO DE ERROR	VALUE	DESCRIPCIÓN
<code>SCC_I_SHARESUBPROJOK</code>	7	El complemento admite la adición de archivos desde el control de código fuente en dos pasos. Para obtener más información, vea SccSetOption .
<code>SCC_I_FILEDIFFERS</code>	6	El archivo local es diferente del archivo de la base de datos de control de código fuente (por ejemplo, SccDiff puede devolver este valor).
<code>SCC_I_RELOADFILE</code>	5	El archivo local se cambió durante la operación de control de código fuente; el IDE debe volver a cargar el archivo si es posible.
<code>SCC_I_FILENOTAFFECTED</code>	4	El archivo no se ve afectado.
<code>SCC_I_PROJECTCREATED</code>	3	El proyecto se creó durante la operación de control de <code>SCC_OP_CREATEIFNEW</code> código fuente (por ejemplo, durante una llamada a SccOpenProject cuando se especifica flag).
<code>SCC_I_OPERATIONCANCELED</code>	2	La operación se canceló.
<code>SCC_I_ADV_SUPPORT</code>	1	El complemento admite opciones avanzadas para el comando especificado. Para obtener más información, vea SccGetCommandOptions .
<code>SCC_OK</code>	0	Correcto.
<code>SCC_E_INITIALIZEFAILED</code>	-1	Error: error de inicialización.
<code>SCC_E_UNKNOWNPROJECT</code>	-2	Error: proyecto es desconocido.
<code>SCC_E_COULDNOTCREATEPROJECT</code>	-3	Error: no se pudo crear el proyecto.
<code>SCC_E_NOTCHECKEDOUT</code>	-4	Error: el archivo no está desprotegido.

CÓDIGO DE ERROR	VALUE	DESCRIPCIÓN
SCC_E_ALREADYCHECKEDOUT	-5	Error: el archivo ya está desprotegido.
SCC_E_FILEISLOCKED	-6	Error: el archivo está bloqueado.
SCC_E_FILEOUTEXCLUSIVE	-7	Error: el archivo está desprotegido exclusivamente.
SCC_E_ACCESSFAILURE	-8	Se ha producido un problema al acceder al sistema de control de código fuente, probablemente debido a problemas de red o contención. Se recomienda un reintento.
SCC_E_CHECKINCONFLICT	-9	Error: hubo un conflicto durante el check-in.
SCC_E_FILEALREADYEXISTS	-10	Error: el archivo ya existe.
SCC_E_FILENOTCONTROLLED	-11	Error: el archivo no está bajo control de código fuente.
SCC_E_FILEISCHECKEDOUT	-12	Error: el archivo está desprotegido.
SCC_E_NOSPECIFIEDVERSION	-13	Error: no hay ninguna versión especificada.
SCC_E_OPNOTSUPPORTED	-14	Error: la operación no es compatible.
SCC_E_NONSPECIFICERROR	-15	Error inespecífico.
SCC_E_OPNOTPERFORMED	-16	Error, la operación no se realizó.
SCC_E_TYENOTSUPPORTED	-17	Error: el tipo del archivo, por ejemplo, binario, no es compatible con el sistema de control de código fuente.
SCC_E_VERIFYMERGE	-18	El archivo se ha fusionado automáticamente, pero no se ha comprobado porque está pendiente de verificación del usuario.
SCC_E_FIXMERGE	-19	El archivo se ha fusionado automáticamente, pero no se ha registrado debido a un conflicto de combinación que debe resolverse manualmente.
SCC_E_SHELLFAILURE	-20	Error debido a un error de shell.
SCC_E_INVALIDUSER	-21	Error: el usuario no es válido.
SCC_E_PROJECTALREADYOPEN	-22	Error: el proyecto ya está abierto.

CÓDIGO DE ERROR	VALUE	DESCRIPCIÓN
<code>SCC_E_PROJSYNTAXERR</code>	-23	Error de sintaxis del proyecto.
<code>SCC_E_INVALIDFILEPATH</code>	-24	Error: la ruta del archivo no es válida.
<code>SCC_E_PROJNOTOPEN</code>	-25	Error: el proyecto no está abierto.
<code>SCC_E_NOTAUTHORIZED</code>	-26	Error: el usuario no está autorizado para realizar esta operación.
<code>SCC_E_FILESYNTAXERR</code>	-27	Error de sintaxis de archivo.
<code>SCC_E_FILENOTEXIST</code>	-28	Error, el archivo local no existe.
<code>SCC_E_CONNECTIONFAILURE</code>	-29	Error: se ha producido un error de conexión.
<code>SCC_E_UNKNOWNERROR</code>	-30	Error desconocido.
<code>SCC_E_BACKGROUNDGETINPROGRESS</code>	-31	La operación de obtención en segundo plano está actualmente en curso.

Macros proporcionadas para la comprobación rápida

```
IS_SCC_ERROR(rtn) (((rtn) < 0) ? TRUE : FALSE)
IS_SCC_SUCCESS(rtn) (((rtn) == SCC_OK) ? TRUE : FALSE)
IS_SCC_WARNING(rtn) (((rtn) > 0) ? TRUE : FALSE)
```

Observaciones

Se espera que todas las funciones de API de complemento de Control de código fuente (excepto [SccAdd](#), [SccCheckin](#) y [SccDiff](#)) se realicen correctamente cuando los archivos locales que se pasan como argumentos no existan en la carpeta de trabajo. Por ejemplo, el IDE puede emitir una llamada a [SccCheckout](#) o [SccUncheckout](#) en un archivo que no existe en la carpeta de trabajo, pero que existe en el sistema de control de código fuente. Esta llamada tendría éxito. Solo cuando no hay ningún archivo en la carpeta de trabajo o en el sistema de control de código fuente se espera que la función falle.

Ciertas funciones, como `SccAdd` y `SccCheckin` `SCC_E_FILENOTEXIST`, deben devolverse específicamente cuando el archivo de la carpeta de trabajo no existe. Se espera que otras funciones se realicen correctamente cuando el archivo de trabajo no existe, si las funciones funcionan con un nombre de archivo válido en el sistema de control de código fuente.

El complemento de control de código fuente no debe realizar ninguna suposición con respecto a los privilegios de un archivo en la carpeta de trabajo, incluso si el complemento había marcado el archivo de solo lectura durante alguna operación. Un archivo de la carpeta de trabajo se puede mover, eliminar y cambiar fuera del control del complemento.

Vea también

- [Complementos de control de código fuente](#)

Cadenas usadas como claves para buscar un complemento de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Las siguientes cadenas son las claves para tener acceso al registro para encontrar información sobre el complemento de control de código fuente.

`STR_SCC_PROVIDER_REG_LOCATION`, `STR_PROVIDERREGKEY`, `STR_SCCPROVIDERPATH`, `STR_SCCPROVIDERNAME`, y son claves del Registro o valores que se usan para registrar un archivo DLL como un complemento de control de código fuente para Visual Studio.

`SCC_PROJECTNAME_KEY`, `SCC_PROJECTAUX_KEY`, `SCC_KEY`, `SCC_FILE_SIGNATURE`, `SCC_STATUS_FILE`, y se utilizan para describir el formato del MSSCCPRJ. SCC.

Claves y valores de cadena

CLAVE	VALUE
<code>STR_SCC_PROVIDER_REG_LOCATION</code>	Software-SourceCodeControlProvider
<code>STR_PROVIDERREGKEY</code>	ProviderRegKey
<code>STR_SCCPROVIDERPATH</code>	SCCServerPath
<code>STR_SCCPROVIDERNAME</code>	SCCServerName
<code>STR_SCC_INI_SECTION</code>	Control de código fuente
<code>STR_SCC_INI_KEY</code>	SourceCodeControlProvider
<code>SCC_PROJECTNAME_KEY</code>	SCC_Project_Name
<code>SCC_PROJECTAUX_KEY</code>	SCC_Aux_Path
<code>SCC_STATUS_FILE</code>	MSSCCPRJ. Scc
<code>SCC_KEY</code>	SCC
<code>SCC_FILE_SIGNATURE</code>	Un archivo de control de código fuente
<code>SCC_NSE</code>	Extensión del espacio de nombres
<code>SCC_NSE_PREFIX</code>	Prefijo protocol
<code>SCC_NSE_DisableOpenSCC</code>	DisableOpenFromSourceControl
<code>STR_SCCHELPCOLLECTION</code>	HelpCollection

CLAVE	VALUE
STR_UI_LANGUAGE	UILanguage
STR_SRCSAFE_ROOT_KEY	Software-Microsoft-SourceSafe

Vea también

- [Complementos de control de código fuente](#)
- [Instalación de un complemento de control de código fuente](#)
- [Archivo MSSCCPRJ.SCC](#)

MSSCCPRJ. Archivo SCC

06/04/2020 • 7 minutes to read • [Edit Online](#)

Al colocar una solución o proyecto de Visual Studio bajo control de código fuente mediante el IDE, el IDE recibe dos elementos clave de información. La información procede del complemento de control de código fuente en forma de cadenas. Estas cadenas, "AuxPath" y "ProjName", son opacas para el IDE, pero el complemento las usa para buscar la solución o el proyecto en el control de versiones. Normalmente, el IDE obtiene estas cadenas la primera vez llamando a [SccGetProjPath](#), a continuación, las guarda en el archivo de solución o proyecto para futuras llamadas a [SccOpenProject](#). Cuando se incrustan en los archivos de solución y proyecto, las cadenas "AuxPath" y "ProjName" no se actualizan automáticamente cuando un usuario bifurca, bifurca o copia archivos de solución y proyecto que están en control de versiones. Para asegurarse de que la solución y los archivos de proyecto apuntan a su ubicación correcta en el control de versiones, los usuarios deben actualizar manualmente las cadenas. Dado que las cadenas están diseñadas para ser opacas, es posible que no siempre esté claro cómo se deben actualizar.

El complemento de control de código fuente puede evitar este problema almacenando las cadenas "AuxPath" y "ProjName" en un archivo especial denominado archivo *MSSCCPRJ.SCC*. Es un archivo local del lado cliente que es propiedad y es mantenido por el plug-in. Este archivo nunca se coloca bajo control de código fuente, pero lo genera el complemento para cada directorio que contiene archivos controlados por código fuente. Para determinar qué archivos son archivos de solución y proyecto de Visual Studio, un complemento de control de código fuente puede comparar las extensiones de archivo con una lista estándar o proporcionada por el usuario. Una vez que el IDE detecta que un complemento es compatible con el archivo *MSSCCPRJ.SCC*, deja de incrustar las cadenas "AuxPath" y "ProjName" en los archivos de solución y proyecto, y lee esas cadenas del archivo *MSSCCPRJ.SCC* en su lugar.

Un complemento de control de código fuente que admita el archivo *MSSCCPRJ.SCC* debe cumplir las siguientes directrices:

- Solo puede haber un archivo *MSSCCPRJ.SCC* por directorio.
- Un archivo *MSSCCPRJ.SCC* puede contener "AuxPath" y "ProjName" para varios archivos que están bajo control de código fuente dentro de un directorio determinado.
- La cadena "AuxPath" no debe tener comillas dentro de ella. Se permite tener comillas a su alrededor como delimitadores (por ejemplo, se puede utilizar un par de comillas dobles para indicar una cadena vacía). El IDE quitará todas las comillas de la cadena "AuxPath" cuando se lea desde el archivo *MSSCCPRJ.SCC*.
- La cadena "ProjName" en *MSSCCPRJ. El archivo SCC* debe coincidir `SccGetProjPath` exactamente con la cadena devuelta por la función. Si la cadena devuelta por la función tiene comillas alrededor de ella, la cadena en el archivo *MSSCCPRJ.SCC* debe tener comillas alrededor de ella, y viceversa.
- Un archivo *MSSCCPRJ.SCC* se crea o actualiza cada vez que se coloca un archivo bajo control de código fuente.
- Si se elimina un archivo *MSSCCPRJ.SCC*, un proveedor debe regenerarlo la próxima vez que realice una operación de control de código fuente relacionada con ese directorio.
- Un archivo *MSSCCPRJ.SCC* debe seguir estrictamente el formato definido.

Una ilustración del MSSCCPRJ. Formato de archivo SCC

A continuación se muestra una muestra del formato de archivo *MSSCCPRJ.SCC* (los números de línea solo se proporcionan como guía y no deben incluirse en el cuerpo del archivo):

- [Línea 1] `SCC = This is a Source Code Control file`
- [Línea 2]
- [Línea 3] `[TestApp.sln]`
- [Línea 4] `SCC_Aux_Path = "\\server\vss\"`
- [Línea 5] `SCC_Project_Name = "$/TestApp"`
- [Línea 6]
- [Línea 7] `[TestApp.csproj]`
- [Línea 8] `SCC_Aux_Path = "\\server\vss\"`
- [Línea 9] `SCC_Project_Name = "$/TestApp"`

La primera línea indica el propósito del archivo y sirve como firma para todos los archivos de este tipo. Esta línea debe aparecer exactamente así en todos los archivos *MSSCCPRJ.SCC*:

```
SCC = This is a Source Code Control file
```

En la siguiente sección se detalla la configuración de cada archivo, marcada por el nombre de archivo entre corchetes. Esta sección se repite para cada archivo que se realiza el seguimiento. Esta línea es un ejemplo de un `[TestApp.csproj]` nombre de archivo, a saber, . El IDE espera las dos líneas siguientes. Sin embargo, no define el estilo de los valores definidos. Las variables `SCC_Aux_Path` `SCC_Project_Name` son y .

```
SCC_Aux_Path = "\\server\vss\"
```

```
SCC_Project_Name = "$/TestApp"
```

No hay delimitador final en esta sección. El nombre del archivo, así como todos los literales que aparecen en el archivo, se definen en el archivo de encabezado scc.h. Para obtener más información, consulte [Cadenas utilizadas como claves para buscar un complemento](#) de control de código fuente .

Vea también

- [Complementos de control de código fuente](#)
- [Cadenas utilizadas como claves para encontrar un complemento de control de código fuente](#)

Prácticas recomendadas para implementar un complemento de control de código fuente

06/04/2020 • 8 minutes to read • [Edit Online](#)

Los siguientes detalles técnicos pueden ayudarle a implementar Visual Studio de forma fiable un complemento de control de código fuente.

Problemas de administración de memoria

En la mayoría de los casos, el entorno de desarrollo integrado (IDE), que es el autor de la llamada, libera y asigna memoria. El complemento de control de código fuente devuelve cadenas y otros elementos en búferes asignados por el autor de la llamada. Las excepciones se indican en las descripciones de funciones específicas donde se producen.

Matrices de nombres de archivo

Cuando se pasa una matriz de archivos, no se pasa como una matriz contigua de nombres de archivo. Se pasa como una matriz de punteros a los nombres de archivo. Por ejemplo, en [SccGet](#), el `lpFileNames` parámetro pasa `lpFileNames` los nombres de `char **` archivo, donde en realidad es un puntero a un archivo. `lpFileNames[0]` es un puntero al `lpFileNames` primer nombre, `[1]` es un puntero al segundo nombre, y así sucesivamente.

Modelo grande

Todos los punteros son de 32 bits, incluso en sistemas operativos de 16 bits.

Rutas totalmente calificadas

Cuando los nombres de archivo o directorios se especifican como argumentos, deben ser rutas de acceso completas o rutas UNC, sin las barras diagonales inversas finales. Es responsabilidad del complemento de control de código fuente traducirlos a rutas de acceso relativas si se trata de un requisito del sistema de control de código fuente subyacente.

Especifique una ruta de acceso completa para el archivo DLL registrado

El IDE ya no carga archivos DLL desde rutas de acceso relativas (por ejemplo, `.-NewProvider.dll`). Se debe especificar una ruta de acceso completa del archivo DLL (por ejemplo, `C:-Providers-NewProvider.dll`). Este requisito refuerza la seguridad del IDE al impedir la carga de archivos DLL de control de código fuente no autorizados o suplantados.

Compruebe si hay un complemento VSSCI existente al instalar el complemento de control de código fuente

Un usuario que planea instalar el complemento de control de código fuente ya puede tener un complemento de control de código fuente existente instalado en el equipo. El programa de instalación (configuración) para el complemento que cree debe determinar si existen valores para las claves de registro relevantes. Si estas claves ya están establecidas, el programa de instalación debe preguntar al usuario si desea registrar el complemento como el complemento de control de código fuente predeterminado y reemplazar el que ya está instalado.

Códigos de resultado de error e informes

El `scc_ok` código de retorno de una función de control de código fuente indica que la operación se ha realizado correctamente para todos los archivos. Si se produce un error en la operación, se espera que devuelva el último código de error encontrado.

La regla para informar es que si se produce un error en el IDE, el IDE es responsable de notificarlo. Si se produce un error en el sistema de control de código fuente, el complemento de control de código fuente es responsable de notificarlo. Por ejemplo, el IDE no **notificaría ningún archivo actualmente**, mientras que **este archivo ya está desprotegido** sería notificado por el complemento.

La estructura del contexto

Durante la llamada a `SccInitialize`, `ppvContext` el llamador pasa el parámetro, que es un identificador no inicializado a un void. El complemento de control de código fuente puede omitir este parámetro o puede asignar una estructura de cualquier tipo y colocar un puntero a esa estructura en el puntero pasado. El IDE no entiende esta estructura, pero pasa un puntero a esta estructura en cada otra llamada en el complemento. Esto proporciona información valiosa de caché de contexto al complemento que puede usar para mantener la información de estado global que persiste en las llamadas de función sin usar variables globales. El plug-in es responsable de liberar la estructura en una llamada a la `SccUninitialize`.

Si el complemento establece `SCC_CAP_REENTRANT` el bit en `SccInitialize` (específicamente, en el `lpSccCaps` parámetro), se utilizan varias estructuras de contexto para realizar un seguimiento de todos los proyectos que están abiertos.

Bitflags y otras opciones de comando

Para cada comando, como `SccGet`, el IDE puede especificar muchas opciones que cambian el comportamiento del comando.

La API admite la configuración de determinadas `fOptions` opciones mediante el IDE a través del parámetro. Estas opciones se describen en [Bitflags utilizados por comandos específicos](#) junto con los comandos que afectan. En general, se trata de opciones para las que no se solicitaría al usuario.

La mayoría de las opciones de configuración configurables por el usuario no se definen de esta manera, ya que varían ampliamente entre los complementos de control de código fuente. Por lo tanto, el mecanismo recomendado es un botón **Avanzado**. Por ejemplo, en el cuadro de diálogo **Obtener**, el IDE muestra solo la información que comprende, pero también muestra un botón **Avanzado** si el complemento tiene opciones para este comando. Cuando el usuario hace clic en el avanzado botón, el IDE llama a la `SccGetCommandOptions` para habilitar el complemento de control de código fuente para solicitar al usuario información, como bitflags o una fecha y hora. El complemento devuelve esta información en una estructura `SccGet` que se devuelve durante el comando.

Vea también

- [Complementos de control de código fuente](#)
- [Crear un complemento de control de código fuente](#)

Restricciones en las longitudes de cadena

06/04/2020 • 2 minutes to read • [Edit Online](#)

La API de complemento de Control de código fuente limita las longitudes de las cadenas utilizadas en varias funciones.

Valores de longitud de cadena

CONSTANTE	VALUE
SCC_NAME_LEN	31
SCC_AUXLABEL_LEN	31
SCC_USER_LEN	31
SCC_PRJPATH_LEN	300

NOTE

La longitud no incluye `null` la terminación . Otras constantes con un sufijo "_SIZE" en lugar de `null` "_LEN" incluyen espacio para la terminación .

CONSTANTE	VALUE
SCC_NAME_SIZE	32
SCC_AUXLABEL_SIZE	32
SCC_USER_SIZE	32
SCC_PRJPATH_SIZE	301

Vea también

- [Complementos de control de código fuente](#)

Glosario del complemento de control de código fuente

06/04/2020 • 4 minutes to read • [Edit Online](#)

Los siguientes términos y definiciones útiles pertenecen a la documentación del SDK de complemento de Control de código fuente.

Definiciones

Protección Cuando un usuario realiza cambios en una copia de trabajo, un usuario debe enviar cambios desde la copia de trabajo al repositorio de control de código fuente central. Esto crea una nueva revisión del archivo que está disponible para otros usuarios. Este proceso se denomina check-in.

Desprotección El acto de solicitar una copia de trabajo del repositorio, informando al repositorio de su intención de modificarla. Una copia de trabajo refleja el estado del proyecto en el momento en que se desprotege.

Cliente Un programa que utiliza el sistema de control de código fuente. Para el propósito de esta documentación, es el IDE de Visual Studio.

Comentario Un mensaje que describe los cambios que un usuario puede adjuntar a una revisión cuando se realiza una operación de control de código fuente.

Conflicto Una situación cuando dos usuarios intentan proteger los cambios en la misma región del mismo archivo. Normalmente, se debe realizar una combinación.

Directorio Una carpeta local del lado cliente se conoce como un directorio. Esta es la copia en la que un usuario realmente realiza cambios. Puede haber muchas copias de trabajo de un proyecto determinado; generalmente cada desarrollador tiene su propia copia.

Obtener una operación get pone al día la copia de trabajo del usuario con la copia del repositorio. A diferencia de una desprotección, se realiza una obtención cuando el usuario simplemente necesita la copia más reciente, pero tiene la intención de no realizar ningún cambio.

Historial Normalmente es un resumen de todas las desprotecciones, comprobaciones, actualizaciones, etiquetas y versiones realizadas en el repositorio de control de código fuente.

IDE Generalmente hace referencia al entorno de desarrollo integrado de Visual Studio. Sin embargo, también podría hacer referencia a otros entornos de cliente que reconocen la API de complemento de Control de código fuente.

Combinar El proceso durante el cual se combinan dos o más archivos de código fuente para formar un nuevo archivo que incorpora todas las características de los archivos anteriores. Este concepto es vital en el control de versiones donde dos o más desarrolladores trabajan en archivos simultáneamente.

Proyecto Una carpeta de control de código fuente se conoce a menudo como un proyecto. Esto no tiene ninguna relación con proyectos o soluciones en Visual Studio.

Complemento un archivo DLL que proporciona funcionalidad de control de código fuente mediante la implementación de la API de complemento de Control de código fuente.

Repositorio La copia maestra donde un sistema de control de código fuente almacena el historial de revisiones completo de un proyecto. Cada proyecto tiene exactamente un repositorio.

Revisión Un cambio confirmado en el historial de un archivo o conjunto de archivos. Una revisión es una

instantánea en un proyecto que cambia continuamente.

Vea también

- [Complementos de control de código fuente](#)

Cómo: Desactivar las advertencias de compatibilidad para complementos de control de código fuente

06/04/2020 • 2 minutes to read • [Edit Online](#)

Un usuario puede ver varias advertencias de Visual Studio de compatibilidad al emplear el control de código fuente en . Las advertencias presentadas dependen de las capacidades del complemento de control de código fuente y se pueden deshabilitar como se detalla aquí.

Para deshabilitar la advertencia: "Para garantizar una integración óptima del control de código fuente con Visual Studio"

- Establezca la siguiente entrada del Registro (añadiendo el valor si es necesario):

```
HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8,0, SourceControl,  
DontDisplayCheckDotNETCompatible, dword:00000001
```

Esta advertencia se muestra para Visual SourceSafe todos los no plug-ins.

Para deshabilitar la advertencia: "El proveedor de control de código fuente instalado no admite todas las capacidades"

- Establezca los dos valores del Registro siguientes (añadiendo los valores si es necesario):

```
HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8,0, SourceControl,  
WarnedOldMSSCCIProvider, dword:00000000
```

```
HKEY_CURRENT_USER,Software, Microsoft, VisualStudio, 8,0, SourceControl, UseOldSCC,  
dword:00000001
```

Esta advertencia se muestra si el complemento de control de código fuente no admite explícitamente la reentrada para varios proyectos (es decir, si solo puede proteger un archivo y un proyecto a la vez).

Lo mejor es apoyar la `SCC_CAP_REentrant` reentrada (capacidad); al hacerlo, se eliminará esta advertencia. Sin embargo, si este soporte no es posible, se pueden establecer estas entradas del registro.

Vea también

- [Banderas de capacidad](#)

Conmutadores de línea de comandos (SDK de Visual Studio)

06/04/2020 • 2 minutes to read • [Edit Online](#)

Devenv.exe, el ejecutable Visual Studio que inicia el entorno de desarrollo integrado (IDE), admite modificadores de línea de comandos específicamente para el desarrollo de VSPackages. VSPackages también pueden hacer que sus propios modificadores estén disponibles en la línea de comandos de Devenv. En los temas siguientes se proporciona información acerca de cómo usar modificadores de línea de comandos en el IDE.

En esta sección

- [Conmutadores de línea de comandos de Devenv para VSPackage Development](#) Describe los modificadores de línea de comandos de Devenv que puede usar para simplificar el desarrollo, la depuración y la implementación de VSPackage.
- [Agregar modificadores de línea de comandos](#) Describe cómo implementar, buscar y leer modificadores de línea de comandos introducidos por el usuario para el paquete.

Secciones relacionadas

- [Conmutadores de línea de comandos Devenv](#) Explica cómo Devenv le permite establecer varias opciones para el IDE y cómo puede compilar, compilar y depurar proyectos desde la línea de comandos. Además, hay información sobre la sintaxis de los conmutadores Devenv y una tabla de conmutadores.

Modificadores de línea de comandos Devenv para el desarrollo de VSPackage

06/04/2020 • 4 minutes to read • [Edit Online](#)

Visual Studio permite a los desarrolladores automatizar `devenv.exe` tareas desde la línea de comandos al ejecutar , el archivo que inicia el IDE de Visual Studio.

Las tareas incluyen:

- Implementación de aplicaciones en configuraciones prediseñadas desde fuera del IDE.
- Creación automática de proyectos mediante configuraciones de compilación predefinidas o configuraciones de depuración.
- Cargar el IDE en configuraciones específicas, todo desde fuera del IDE. También puede personalizar el IDE al iniciarse.

Directrices para interruptores

La documentación de Visual Studio `devenv` describe los modificadores de línea de comandos de nivel de usuario. Para obtener más información, consulte [Conmutadores de línea de comandos de Devenv](#). La `devenv` herramienta también admite modificadores de línea de comandos adicionales que son útiles con el desarrollo, la implementación y la depuración de VSPackage.

INTERRUPTOR DE LÍNEA DE COMANDOS	DESCRIPCIÓN
<code>/ResetSkipPkgs</code>	Borra todas las opciones de carga de omisión que han agregado los usuarios que desean evitar cargar VSPackages problemáticos y, a continuación, inicia Visual Studio. La presencia de un SkipLoading etiqueta deshabilita la carga de un VSPackage. Borrar la etiqueta vuelve a habilitar la carga del VSPackage. Este modificador no toma ningún argumento.
<code>/RootSuffix</code>	Inicia Visual Studio mediante una ubicación alternativa. El siguiente comando se ejecuta mediante el acceso directo creado por el instalador del SDK de Visual Studio: <pre>devenv /RootSuffix exp</pre> En este <code>exp</code> caso, identifica una ubicación con un <code>10.0Exp</code> sufijo <code>10.0</code> determinado (por ejemplo, en lugar de). La instancia experimental permite depurar un VSPackage por separado de la instancia de Visual Studio que está utilizando para escribir código. Este modificador puede tomar cualquier cadena que identifique una ubicación que haya creado mediante VSRegEx.exe. Para obtener más información, consulte La instancia experimental.

INTERRUPTOR DE LÍNEA DE COMANDOS	DESCRIPCIÓN
<code>/SafeMode</code>	Inicia Visual Studio en modo seguro, cargando solo el IDE y los servicios predeterminados. El <code>/SafeMode</code> modificador impide que todos los VSPackages de terceros se carguen cuando se inicia Visual Studio, lo que garantiza una ejecución estable. Este modificador no toma ningún argumento.
<code>/Setup</code>	Obliga a Visual Studio a combinar metadatos de recursos que describen menús, barras de herramientas y grupos de comandos de todos los VSPackages disponibles. Solo puede ejecutar este comando como administrador. Este modificador no toma ningún argumento. El comando <code>devenv /Setup</code> suele ser el último paso del proceso de instalación. El uso <code>/Setup</code> del modificador no inicia el IDE.
<code>/Splash</code>	Muestra la pantalla de presentación de Visual Studio, como de costumbre y, a continuación, muestra un cuadro de mensaje antes de mostrar el IDE principal. El cuadro de mensaje le permite estudiar la pantalla de presentación (por ejemplo, para comprobar si hay un icono de producto de VSPackage). Este modificador no toma ningún argumento.

Vea también

- [Agregar modificadores de línea de comandos](#)
- [Modificadores de línea de comandos para Devenv](#)

Agregar modificadores de línea de comandos

06/04/2020 • 3 minutes to read • [Edit Online](#)

Puede agregar modificadores de línea de comandos que se aplican a su VSPackage cuando se ejecuta *devenv.exe*. Se [ProvideAppCommandLineAttribute](#) utiliza para declarar el nombre del modificador y sus propiedades. En este ejemplo, el modificador MySwitch se agrega para una subclase de VSPackage denominada **AddCommandSwitchPackage** sin argumentos y con el VSPackage cargado automáticamente.

```
[ProvideAppCommandLine("MySwitch", typeof(AddCommandSwitchPackage), Arguments = "", DemandLoad = 1)]
```

Los parámetros con nombre se muestran en las descripciones siguientes.

Parámetro	Descripción	
Argumentos	El número de argumentos para el switch. Puede ser "*", o una lista de argumentos.	
DemandLoad	Cargue el VSPackage automáticamente si se establece en 1, de lo contrario se establece en 0.	
HelpString	La cadena de ayuda o el identificador de recurso de la cadena que se va a mostrar con devenv /? .	
Nombre	El interruptor.	
PackageGuid	Identificador único global (GUID) del paquete.	

El primer valor de Arguments suele ser 0 o 1. Se puede utilizar un valor especial de '*' para indicar que el argumento es el resto de la línea de comandos. Esto puede ser útil para depurar escenarios donde un usuario debe pasar una cadena de comandos del depurador.

El demandLoad valor `true` es (1) o `false` (0) indica que el VSPackage debe cargarse automáticamente.

El valor HelpString es el identificador de recurso de la cadena que aparece en el **devenv /?** Pantalla de ayuda. Este valor debe tener la forma "#nnn" donde nnn es un entero. El valor de cadena en el archivo de recursos debe terminar en un nuevo carácter de línea.

El valor Name es el nombre del switch.

El valor PackageGuid es el GUID del paquete que implementa este modificador. El IDE usa este GUID para buscar el VSPackage en el registro al que se aplica el modificador de línea de comandos.

Recuperar conmutadores de línea de comandos

Cuando se carga el paquete, puede recuperar los modificadores de línea de comandos completando los pasos siguientes.

1. En la implementación [SetSite](#) de `QueryService` `VSPackage`, llame [SVsAppCommandLine](#) para obtener la [IVsAppCommandLine](#) interfaz.
2. Llame [GetOption](#) para recuperar los switches de línea de comandos que el usuario ingresó.

El código siguiente muestra cómo averiguar si el usuario ha introducido el modificador de línea de comandos `MySwitch`:

```
IVsAppCommandLine cmdline = (IVsAppCommandLine)GetService(typeof(SVsAppCommandLine));

int isPresent = 0;
string optionValue = "";

cmdline.GetOption("MySwitch", out isPresent, out optionValue);
```

Es su responsabilidad comprobar si los conmutadores de línea de comandos cambian cada vez que se carga el paquete.

Vea también

- [IVsAppCommandLine](#)
- [SetSite](#)
- [Modificadores de línea de comandos para Devenv](#)
- [Utilidad CreatePkgDef](#)
- [. Archivos Pkgdef](#)

Manejo de errores y valores devueltos

06/04/2020 • 6 minutes to read • [Edit Online](#)

VSPackages y COM usan la misma arquitectura para los errores. Las `SetErrorInfo` `GetErrorInfo` funciones y forman parte de la interfaz de programación de aplicaciones (API) de Win32. Cualquier VSPackage en el entorno de desarrollo integrado (IDE) puede llamar a estas API globales de Win32 para registrar información de error enriquecida al recibir una notificación de error. Proporciona Visual Studio SDK ensamblados de interoperabilidad para administrar la información de error.

Métodos de interoperabilidad

Como comodidad, el IDE proporciona `SetErrorInfo` un método, para usar en lugar de llamar a las API de Win32. En código administrado use `SetErrorInfo`. Cuando un `HRESULT` error llega al nivel donde se debe mostrar el mensaje `ICommandTarget` de error (a menudo es `ReportErrorInfo` el objeto que implementa un controlador de comandos), el IDE utiliza otro método, para mostrar el cuadro de mensaje adecuado. En código administrado, utilice el `ReportErrorInfo` método.

Como implementador de VSPackage, los `ISupportErrorInfo` objetos COM normalmente implementan . La `ISupportErrorInfo` interfaz garantiza que la información de error enriquecida puede moverse verticalmente hacia arriba de la cadena de llamadas. Los objetos que se pueden usar `ISupportErrorInfo` entre procesos o entre subprocesos deben admitir se deben admitir para asegurarse de que la información de error enriquecida se calcula correctamente al autor de la llamada.

Todos los objetos que están relacionados con VSPackages y que están implicados en la extensión del IDE, incluidos los generadores de editores, editores, jerarquías y servicios ofrecidos, deben admitir información de error enriquecida. Aunque el IDE no requiere estos `ISupportErrorInfo` VSPackage objetos para implementar , siempre se recomienda.

El IDE es responsable de informar de la Visual Studio información `HRESULT` de error y mostrarla a un usuario de cada vez que se propaga un al IDE. El IDE es también `ErrorInfo` el mecanismo para crear objetos.

Directrices generales

Puede usar `SetErrorInfo` los `ReportErrorInfo` métodos y para establecer y notificar errores que son internos de la implementación de VSPackage también. Sin embargo, como regla general, siga estas directrices para controlar los mensajes de error en el VSPackage:

- Implementar `ISupportErrorInfo` en los objetos COM de VSPackage.
- Cree un mecanismo de `SetErrorInfo` informes de `ICommandTarget` errores que llame al método en los objetos que implementan .
- Deje que el IDE muestre `ReportErrorInfo` errores a los usuarios a través del método.

Información de error en el IDE

Las reglas siguientes indican cómo controlar Visual Studio la información de error en el IDE:

- Como una estrategia defensiva para garantizar que la información de error `ReportErrorInfo` obsoleto no `SetErrorInfo` se notifica a los usuarios, las funciones que llaman al método primero deben llamar al método. Pase `null` para borrar los mensajes de error almacenados en caché antes de llamar a cualquier cosa que pueda establecer nueva información de error.

- Las funciones que no notifican directamente [SetErrorInfo](#) los mensajes de error `HRESULT` solo pueden llamar al método si devuelven un error. Es permisible borrar la `ErrorInfo` entrada a una función [S_OK](#) al devolver . La única excepción a esta regla `HRESULT` es cuando una llamada devuelve un error desde el que la parte receptora puede recuperarse o omitirse de forma segura.
- Cualquier parte que omita `HRESULT` explícitamente [SetErrorInfo](#) un [S_OK](#) debe llamar al método con . De lo `ErrorInfo` contrario, el objeto podría utilizarse accidentalmente `ErrorInfo` cuando otra parte genera un error sin proporcionar su propio archivo .
- Se recomienda a `HRESULT` todos los métodos [SetErrorInfo](#) que originan un error que llamen al método para proporcionar información de error enriquecida. Si el `HRESULT` devuelto `FACILITY_ITF` es un error especial, el `ErrorInfo` método es necesario para proporcionar un objeto adecuado. Si el error devuelto es un error [E_OUTOFMEMORY](#) [E_ABORT](#) del [E_INVALIDARG](#) [E_UNEXPECTED](#) sistema estándar (por ejemplo, , , , , etc.) es aceptable devolver el código de error sin llamar explícitamente al [SetErrorInfo](#) método. Como estrategia de codificación defensiva, `HRESULT` al originar un [SetErrorInfo](#) error (incluidos `ErrorInfo` los errores del sistema), `null` siempre llame al método, ya sea describiendo el error con mayor detalle, o .
- Todas las funciones que devuelven un error originado por otra llamada `HRESULT` deben `ErrorInfo` pasar la información que se recibió de la llamada con errores en el sin modificar el objeto.

Vea también

- [IOleCommandTarget](#)
- [SetErrorInfo](#) (automatización de componentes)
- [GetErrorInfo](#)
- [Interfaz ISupportErrorInfo](#)

Objetos

06/04/2020 • 2 minutes to read • [Edit Online](#)

La siguiente información de referencia explica los objetos que se utilizan para crear una instancia del editor de texto principal.

En esta sección

- [Objeto VSCodeWindow](#) Describe el objeto de ventana de código, una ventana de documento especializada que puede incluir una o varias vistas de texto.
- [Objeto VSCodeWindowManager](#) Describe el objeto de administrador de ventanas de código implementado por el servicio de lenguaje y es responsable de administrar los adornos.
- [Objeto VSTextBuffer](#) Describe el objeto de búfer de texto, que representa una secuencia de texto Unicode, los datos de un archivo.
- [VSTextView objeto](#) Describe la vista de texto, una ventana que permite a los usuarios ver y editar texto Unicode del búfer de texto.

Secciones relacionadas

- [Referencia del SDK de Visual Studio](#) Presenta una lista de temas de referencia.

Objeto VSCodeWindow

06/04/2020 • 2 minutes to read • [Edit Online](#)

Una ventana de código es una ventana de documento especializada [VsTextView](#) que puede incluir una o varias vistas de texto, normalmente el objeto.

Arquitectónicamente, la ventana de código es una ventana de documento que está dentro de un marco de ventana. Funcionalmente, la ventana de código es simplemente una ventana de documento con características adicionales. En el modo de interfaz de varios documentos (MDI), la ventana de código es el marco secundario MDI. Para obtener más información, consulte Personalización de ventanas de [código mediante la API heredada](#).

En la tabla siguiente se [VsCodeWindow](#) incluyen las interfaces del objeto.

MÉTODO	DESCRIPCIÓN
IServiceProvider	Proporciona un mecanismo de acceso genérico para buscar un servicio que identifica un identificador único global (GUID).
IVsCodeWindow	Representa un elemento secundario de interfaz de varios documentos (MDI) que contiene una o varias vistas de código.
IVsWindowPane	Rellena un marco de ventana.

Vea también

- [IServiceProvider](#)
- [Edición de figuras](#)

Objeto VSCodeWindowManager

06/04/2020 • 2 minutes to read • [Edit Online](#)

El servicio de lenguaje implementa el administrador de ventanas de código y es responsable de administrar los adornos (por ejemplo, la barra desplegable). Para obtener más información, consulte [Personalización](#) de Windows de código mediante la API heredada .

En la tabla siguiente se `VSCodeWindowManager` muestran las interfaces del objeto.

INTERFAZ	DESCRIPCIÓN
IVsCodeWindowManager	Permite agregar o quitar adornos (como barras desplegables) de una ventana de código.

Objeto VSTextBuffer

06/04/2020 • 2 minutes to read • [Edit Online](#)

El objeto de búfer de texto representa una secuencia de texto Unicode, que generalmente se asocia a un archivo. Un [VSTextBuffer](#) objeto se puede utilizar fuera del contexto del editor principal, como en un asistente.

En la tabla siguiente `VSTextBuffer` se muestran las interfaces de .

MÉTODO	DESCRIPCIÓN
IOleCommandTarget	Interfaz OLE estándar. Se utiliza para deshacer/rehacer el manejo en el búfer.
IPersistFile	Interfaz OLE estándar.
Ipersiststream	Interfaz OLE estándar.
IVsCompoundAction	Permite la creación de acciones compuestas (es decir, acciones que se agrupan en una sola unidad de deshacer/rehacer).
IVsPersistDocData	Permite la persistencia de los datos de documento administrados por el búfer de texto.
IVsTextBuffer	Proporciona servicios básicos; utilizado por muchos clientes.
IVsTextFind	Se utiliza para buscar un búfer.
IVsTextLines	Proporciona capacidades de lectura y escritura mediante coordenadas bidimensionales. Se hereda de <code>IVsTextBuffer</code> .
IVsTextStream	Proporciona capacidades de lectura y escritura mediante coordenadas unidimensionales. Se hereda de <code>IVsTextBuffer</code> .
IVsTextScanner	Proporciona acceso rápido, orientado a secuencias y secuencial al texto en el búfer.
IVsUserData	Proporciona acceso a una colección genérica de propiedades. La propiedad más importante es el nombre, o moniker, del búfer. Puede almacenar sus propios datos aleatorios en el búfer con esta interfaz creando un GUID y utilizándolo como clave.
IConnectionPointContainer	Admite puntos de conexión para eventos.

Observaciones

El `VSTextBuffer` se encuentra `QueryInterface` generalmente `IVsTextBuffer` por una llamada en . Para obtener más información, consulte [Zona de influenciade texto](#) .

Vea también

- [IVsTextBuffer](#)
- [VsTextView](#)
- [Edición de figuras](#)

VSTextView objeto

06/04/2020 • 2 minutes to read • [Edit Online](#)

La vista de texto es una ventana que permite a los usuarios ver y editar el texto Unicode del búfer de texto. Esencialmente, la vista es lo que la mayoría de los usuarios se refieren como el editor. Dado que la vista está separada del búfer por varias capas de texto (ajuste de palabras, texto de esquematización, etc.), no se garantiza que la vista sea una representación exacta del texto en el búfer. Para obtener más información acerca de la vista de texto, consulte [Acceso a la vista de texto mediante la API heredada](#).

En la tabla siguiente se [VsTextView](#) muestran las interfaces del objeto.

INTERFAZ	DESCRIPCIÓN
IDropSource	Interfaz OLE estándar.
IDropTarget	Interfaz OLE estándar.
IObjectWithSite	Interfaz OLE estándar.
IOleCommandTarget	Interfaz OLE estándar.
IVsCompoundAction	Permite la creación de acciones compuestas (es decir, acciones que se agrupan en una sola unidad de deshacer/rehacer).
IVsTextView	Proporciona los métodos básicos para administrar y tener acceso a la vista. <code>IVsTextView</code> no es seguro enhebrado.
IVsWindowPane	Crea y administra un panel de ventana.
IVsLayeredTextView	Interactúa con capas de texto.
IVsThreadSafeTextView	Realiza operaciones en la vista desde un subproceso diferente.

Vea también

- [Edición de figuras](#)
- [Objeto VSTextBuffer](#)

Glosario del SDK de Visual Studio

06/04/2020 • 20 minutes to read • [Edit Online](#)

Este glosario proporciona definiciones para Visual Studio SDK los términos que se utilizan en la documentación.

Términos

complemento Una aplicación de utilidad, controlador u otro software agregado a una aplicación principal. En el entorno de desarrollo integrado (IDE) de Visual Studio, un complemento es una aplicación basada en automatización que amplía las capacidades del IDE.

Modelo de automatización El modelo de automatización, conocido en versiones anteriores de Visual Studio como el modelo de extensibilidad, es una interfaz de programación que le da acceso a las rutinas subyacentes que controlan el IDE. Complementos, asistentes y macros usan objetos en el modelo de automatización para controlar o ampliar la funcionalidad del IDE.

contexto de interfaz de usuario de comandos Asociación de un GUID con la visibilidad de un comando o elemento de interfaz de usuario, como una barra de herramientas. El contexto de la interfaz de usuario de comandos es diferente del contexto de selección, ya que no está asociado a una ventana.

El contexto de la interfaz de usuario de comandos se puede utilizar para:

- Asigne un GUID a una barra de herramientas que aparece cuando se activa una ventana determinada.
- Asigne un GUID a la disponibilidad de un comando sin tener que cargar o ejecutar un VSPackage.
- Asigne un GUID para afectar al enlace de claves activo.
- Asigne un GUID para activar la grabación de macros.
- Asigne un GUID para activar el modo de depuración o para alternar entre el modo de diseño y de ejecución en un editor.

componente Pieza de software que se puede hacer parte de la funcionalidad de una aplicación sin que esa aplicación tenga ninguna información preexistente sobre la implementación del software. La comunicación entre un componente y una aplicación es únicamente a través de interfaces de estilo OLE.

administrador de componentes Un servicio, `S0leComponentManager`, que proporciona servicios de coordinación de interfaz de usuario para componentes de nivel superior. El `S0leComponentManager` servicio implementa `I0leComponentManager` la interfaz.

administrador de interfaz `S0leComponentUIManager` de usuario de componentes Un servicio, que proporciona servicios de coordinación de interfaz de usuario. El `S0leComponentUIManager` servicio implementa `I0leComponentUIManager` `I0leInPlaceComponentUIManager` las interfaces y.

bolsa de `IVsUserContext` **contexto** Un objeto (objeto COM) asociado a un componente de entorno. Este objeto contiene palabras clave de búsqueda, palabras clave F1 y atributos relacionados con el componente. Las bolsas de contexto apuntan además a cualquier bolsa de subcontexto que esté vinculada a ellas.

proveedor de contexto Un componente del IDE que tiene asociada una bolsa de contexto. Estos componentes incluyen una ventana de herramientas, un editor o una jerarquía de proyectos.

diseñador Una interfaz de programación que permite a los usuarios manipular elementos de la interfaz de usuario (formularios, botones y otros controles).

DocData Un objeto COM que encapsula los datos subyacentes de un documento en un mundo donde hay separación de documento/vista (por ejemplo, en el caso del editor de texto, este sería el búfer de texto subyacente a todas las vistas del editor de texto). Si EditorFactory no proporciona este objeto, el IDE fabrica uno en su nombre. La responsabilidad de este objeto es administrar la persistencia de datos **DocData** y la semántica de uso compartido para varias vistas sobre este mismo archivo. Si **DocData** el objeto **IOleCommandTarget** admite la interfaz, se incluye en el enrutamiento de comandos de UIShell.

Tecnología DocObject utilizada para hospedar la interfaz de usuario dentro de un marco proporcionado por el host. Más específicamente, este término se refiere **IOleDocument** a cualquier incrustación que soporte las interfaces relacionadas. Esta tecnología tiene muchas aplicaciones potenciales, como un detalle de implementación de documentos COM, ventanas de herramientas en Visual Basic 5.0, diseñadores ActiveX en Visual Basic 6.0, etc.

documento Utilizado para referirse genéricamente al documento **DocData** en **DocView** su conjunto, tanto el archivo. Por ejemplo, un DocumentFrame contiene un **DocView**, pero **DocData** también conserva una referencia a la para controlar la persistencia.

DocView El DocObject/Embedding/WindowPane con el que el usuario **DocData** interactúa para ver y manipular el archivo. Los usuarios no aprovechan la separación Documento/Vista que forma parte del diseño de la **DocObject** interfaz. Los usuarios utilizan un DocObject completo para actuar como una vista en lugar de **DocData** utilizar una noción más abstracta (y menos formalizada) de datos subyacentes conocidos como **DocView** los objetos siempre se incrustan con objetos de marco de documento (ventanas secundarias MDI) del IDE.

DTE **DTE** El objeto (Extensibilidad de herramientas de desarrollo) es el punto de acceso superior del modelo de automatización de Visual Studio, que permite automatizar y ampliar mediante programación el IDE.

Ventana Ayuda dinámica Ventana herramienta implementada por el IDE y muestra una lista de palabras clave de búsqueda o temas de Ayuda de F1.

editor Code (clase, CLSID) **DocView** que implementa el archivo. También se **DocData** implementa si se admite la vista y la separación de datos.

extensión Una característica que modifica, personaliza o agrega a un IDE. Crear extensiones mediante el modelo de automatización o VSPackages.

editor externo Un editor que no es específico del IDE, como Microsoft Word. Se ha registrado a través de sus propios mecanismos y se puede utilizar fuera del IDE. Si este editor se puede incrustar, aparece dentro de una ventana en el IDE. Si no se puede incrustar, se crea una ventana de nivel superior independiente.

jerarquía árbol de nodos, cada nodo asociado a un conjunto de propiedades.

componente independiente de nivel superior Componente A que utiliza una ventana de nivel superior no quede no quedado y puede funcionar eficazmente como una ventana de aplicación independiente, pero se implementa como un objeto en proceso. Por lo tanto, un componente de nivel superior independiente debe coordinar la modalidad y los servicios de bucle de mensajes con el IDE. Los objetos en proceso no tienen su propio bucle de mensajes.

proveedor de información El proveedor de información es un módulo que puede buscar **IVsUserContextItem** palabras clave y devolver una lista de temas, en forma de objetos. Para proporcionar F1 y elementos de palabra clave de búsqueda para el proveedor de información, registre el archivo de Ayuda compilado (. HxS) con el sistema. Los temas de Ayuda de estos archivos proporcionan la lista de temas que se muestran en la ventana Ayuda dinámica y muestran si un usuario presiona F1.

Componente in situ Un objeto VSPackage **IOleInPlaceComponent** que implementa la interfaz para administrar una ventana que se encuentra visualmente dentro de una ventana de documento propiedad del IDE. Los componentes in situ no participan en la combinación de menús OLE estándar; en su lugar, integran

sus elementos de interfaz de usuario en el IDE.

Hay dos tipos de componentes in situ: componentes in situ cableados y controles de componentes.

Los componentes in situ cableados tienen menús, barras de herramientas y comandos que se integran estrechamente en la interfaz de usuario del IDE, que aparecen como si estuvieran integrados directamente en el IDE.

Los controles de componentes no tienen ninguno de sus propios elementos de interfaz de usuario integrados en el IDE; en su lugar, utilizan los menús, comandos y barras de herramientas del IDE. Por ejemplo, el comando Negrita se podría utilizar para poner en negrita una palabra seleccionada dentro de un control de texto enriquecido incrustado en un formulario. Sin embargo, los controles de componentes pueden solicitar que se muestren los elementos de interfaz de usuario específicos del componente instalados dinámicamente.

Servicio de lenguaje Un conjunto de objetos que permite a los desarrolladores de VSPackage implementar características de editores de código de lenguaje de equipo, como el marcado de texto y colorear.

Proyecto de proyecto de archivos varios utilizado para albergar archivos abiertos que no están en ningún proyecto. La lista de elementos de este proyecto no se conserva.

Los proyectos de proyecto se componen de objetos de jerarquía u objetos COM que implementan la `IVsHierarchy` interfaz.

Diseñador o editor específico del proyecto Un diseñador que no se puede usar independientemente del tipo de proyecto. Todos los diseñadores específicos del proyecto deben introducir su información de Editor Factory en el registro. A continuación, el IDE puede crear instancias del diseñador cada vez que se abre un determinado tipo de archivo en un proyecto determinado.

ventana de tipo de proyecto Una ventana que realiza un seguimiento constante de la jerarquía de proyectos activa y el elemento desde el contexto de selección global. Las ventanas de `SVsTrackSelectionEx` tipo de proyecto usan el servicio para alertar al IDE de los cambios y mostrar comentarios al usuario. El Explorador de soluciones es un ejemplo de una ventana de tipo de proyecto.

Ventana Propiedades Anteriormente Navegador de propiedades.

proyectos basados en referencias Project que no requiere que los archivos del proyecto estén en el mismo directorio. En su lugar, las referencias a archivos de otros directorios no relacionados se almacenan y mantienen mediante el propio proyecto.

tabla de documentos en ejecución Estructura interna mediante la cual el IDE mantiene la lista de todos los documentos abiertos actualmente. La lista incluye todos los documentos abiertos en la memoria, independientemente de si los documentos se están editando actualmente. Un documento es cualquier elemento que se guarda, incluidos los procedimientos almacenados abiertos en un editor, los archivos de un proyecto o el archivo de proyecto principal (por ejemplo, el archivo *.vcproj).

contexto de selección Datos que forman parte del detalle de cada ventana del IDE y se utiliza para realizar un seguimiento de las selecciones activas. El contexto de selección consta de:

- Puntero a `IVsHierarchy` la interfaz de la jerarquía del proyecto
- Identificador de elemento del elemento de proyecto.
- Puntero a `ISelectionContainer` la interfaz que proporciona acceso a las propiedades de los objetos activos.
- Matriz de valores de elemento.

servicio Un contrato para un conjunto de interfaces COM que reside en un único objeto COM. Cuando se crea un servicio, que se identifica mediante un GUID, se define el conjunto de interfaces COM que lleva a

cabo el servicio. Los objetos COM utilizan servicios para comunicarse entre sí.

solución Grupo de proyectos relacionados con los que un usuario trabaja.

Diseñador estándar Un diseñador que se puede usar independientemente del tipo de proyecto. Todos los diseñadores estándar deben introducir su información de Editor Factory en el registro. A continuación, el IDE puede crear instancias del diseñador cada vez que se abre un archivo con una extensión específica. Los datos deben persistir en un archivo.

Editor estándar que se puede utilizar independientemente de cualquier tipo de proyecto en particular. Estos editores tienen EditorFactories registrados en el registro. Esto permite que el IDE localice e invoque el editor.

Editor de SO estándar Una incrustación que no es específica de Visual Studio. Se registra mediante las conocidas claves Win32 (por ejemplo, el Explorador de Win32 sabe cómo invocar). Si se puede incrustar un editor de este tipo, el editor sigue apareciendo en su lugar en el IDE. De lo contrario, se crea una ventana de nivel superior independiente para estos editores.

bolsa de `IVsUserContext` subcontexto Un objeto vinculado a una bolsa de contexto. El objeto contiene palabras clave de búsqueda, palabras clave F1 y atributos para una selección dentro de un componente IDE. Entre los ejemplos de subcontexto se incluyen un comando en una ventana de herramientas o una palabra clave en un editor.

Lista de tareas Ventana de herramientas implementada por el IDE y muestra una lista de tareas activas.

búfer de texto Nombre `VSTextBuffer` común para el objeto .

Vista de texto Nombre `VSTextView` común del objeto .

componente de nivel superior de herramienta Componente A que funciona como una ventana emergente no basada en modelos, coordinando estrechamente con la interfaz de usuario del IDE. Al igual que los componentes independientes de nivel superior, los componentes de nivel superior de herramientas también deben coordinar la modalidad y los servicios de bucle de mensajes con el IDE.

Componente de nivel superior Un objeto `VSPackage` que administra una ventana de nivel superior no esquemática en lugar del área de cliente de una ventana IDE. Los componentes de `IOleComponent` nivel superior implementan la interfaz para aprovechar los servicios de bucle de mensajes, como el acceso al tiempo de inactividad.

Interfaz de usuario activa Un objeto `VSPackage` que está visible y actualmente tiene el foco.

Jerarquía de interfaz de `IVsUIHierarchy` usuario Un objeto COM que implementa la interfaz para permitir la visualización de una jerarquía. La ventana de jerarquía `ISelectionContainer` de interfaz de usuario implementa la interfaz para actualizar la ventana Propiedades; otras ventanas de tipo de proyecto pueden usar esta implementación, si lo desea.

Tabla de comandos de Visual Studio de VSCT. El archivo .vsct contiene información sobre la ubicación y los comportamientos de menús, barras de herramientas y comandos en formato XML.

`VSPackage` Una pieza de software instalable que extiende el IDE de Visual Studio aportando uno o varios de los siguientes elementos: interfaz de usuario, servicios, tipos de proyecto o editor o diseñador. Un `VSPackage` consta de un objeto `IVsPackage` COM que implementa la interfaz y uno o varios otros objetos COM que implementan otras interfaces para admitir la selección y otras características. Además, un `VSPackage` tiene requisitos de registro específicos.

Crear un kit de desarrollo de software

06/04/2020 • 22 minutes to read • [Edit Online](#)

Un kit de desarrollo de software (SDK) es una colección de API a las que puede hacer referencia como un único elemento en Visual Studio. El cuadro de diálogo Administrador de **referencias** muestra todos los SDK que son relevantes para el proyecto. Al agregar un SDK a un proyecto, las API están disponibles en Visual Studio.

Hay dos tipos de SDK:

- Los SDK de plataforma son componentes obligatorios para desarrollar aplicaciones para una plataforma. Por ejemplo, Windows 8.1 el SDK Tienda Windows 8.x es necesario para desarrollar aplicaciones.
- Los SDK de extensión son componentes opcionales que extienden una plataforma pero no son obligatorios para desarrollar aplicaciones para esa plataforma.

En las secciones siguientes se describe la infraestructura general de los SDK y cómo crear un SDK de plataforma y un SDK de extensión.

SDK de la plataforma

Los SDK de plataforma son necesarios para desarrollar aplicaciones para una plataforma. Por ejemplo, Windows 8.1 el SDK es Windows 8.1 necesario para desarrollar aplicaciones para .

Instalación

Todos los SDK de la plataforma se instalarán en *HKLM, \Software, Microsoft, Microsoft, \ @InstallationFolder Microsoft, Sdk de Microsoft [TPI], v[TPV] y [raíz del SDK]*. En consecuencia, Windows 8.1 el SDK se instala en *HKLM- Software , Microsoft SDKs , Windows , v8.1*.

Diseño

Los SDK de plataforma tienen el siguiente diseño:

```
\[InstallationFolder root]
  SDKManifest.xml
  \References
    \[config]
      \[arch]
  \DesignTime
    \[config]
      \[arch]
```

NODO	DESCRIPCIÓN
<i>Carpeta Referencias</i>	Contiene archivos binarios que contienen API que se pueden codificar. Estos podrían incluir archivos o ensamblados de metadatos de Windows (WinMD).

NODO	DESCRIPCIÓN
<i>Carpeta DesignTime</i>	Contiene archivos que solo se necesitan en el momento de la ejecución previa/depuración. Estos podrían incluir documentos XML, bibliotecas, encabezados, binarios en tiempo de diseño de Toolbox, artefactos de MSBuild, etc. Los documentos XML, idealmente, se colocarían en la carpeta <i>.DesignTime</i>, pero los documentos XML para las referencias seguirán colocándose junto al archivo de referencia en Visual Studio. Por ejemplo, el documento XML para una referencia <i>[de referencia [config]\[arch]-sample.dll</i> será <i>.references\[config]\[arch]-sample.xml</i>, y la versión localizada de ese documento será <i>"References\[config]\[arch] [arch]\[locale]-sample.xml</i>.
<i>Carpeta de configuración</i>	Solo puede haber tres carpetas: <i>Depurar</i>, <i>Venta al público</i> y <i>CommonConfiguration</i>. Los autores del SDK pueden colocar sus archivos en <i>CommonConfiguration</i> si se debe consumir el mismo conjunto de archivos SDK, independientemente de la configuración a la que se destinará el consumidor del SDK.
<i>Carpeta de arquitectura</i>	Puede existir cualquier carpeta de <i>arquitectura</i> compatible. Visual Studio admite las siguientes arquitecturas: x86, x64, ARM y neutral. Nota: Win32 se asigna a x86 y AnyCPU se asigna a neutral. MSBuild solo busca en <i>"CommonConfiguration"</i> o <i>"neutral"</i> para los SDK de plataforma.
<i>SDKManifest.xml</i>	Este archivo describe cómo Visual Studio debe consumir el SDK. Consulte el manifiesto Windows 8.1 del SDK para: <pre><FileList DisplayName = "Windows" PlatformIdentity = "Windows, version=8.1" TargetFramework = ".NET for Windows Store apps, version=v4.5.1; .NET Framework, version=v4.5.1" MinVSVersion = "14.0"> <File Reference = "Windows.winmd"> <ToolboxItems VSCategory = "Toolbox.Default" /> </File> </FileList></pre> DisplayName: Valor que muestra el Examinador de objetos en la lista Examinar. PlatformIdentity: La existencia de este atributo indica a Visual Studio y MSBuild que el SDK es un SDK de plataforma y que las referencias agregadas desde él no se deben copiar localmente. TargetFramework: Visual Studio usa este atributo para asegurarse de que solo los proyectos que tienen como destino los mismos marcos de trabajo especificados en el valor de este atributo pueden consumir el SDK. MinVSVersion: Visual Studio usa este atributo para consumir solo los vales de servicio que se aplican a él. Referencia: Este atributo debe especificarse solo para aquellas referencias que contienen controles. Para obtener información sobre cómo especificar si una referencia contiene controles, consulte a continuación.

SDK de extensiones

En las secciones siguientes se describe lo que debe hacer para implementar un SDK de extensión.

Instalación

Los SDK de extensión se pueden instalar para un usuario específico o para todos los usuarios sin especificar una clave del Registro. Para instalar un SDK para todos los usuarios, utilice la siguiente ruta de acceso:

%Archivos de programa%-Microsoft<SDKs plataforma>de>destino.v<número de versión de la plataforma .

Para una instalación específica del usuario, utilice la ruta de acceso siguiente:

%USERPROFILE%-AppData-Local-Microsoft SDKs<>plataforma de destino>.v<número de versión de la plataforma .

Si desea utilizar una ubicación diferente, debe realizar una de estas dos acciones:

1. Especifique en una clave del Registro:

La<plataforma de destino de los SDK de HKLM-Software-Microsoft-Microsoft>-v<número>de versión de la plataforma>SDKName de ExtensionSDKs<><SDKVersion>

y agregue una subclave (predeterminada) `<path to SDK><SDKName><SDKVersion>` que tenga un valor de .

2. Agregue la propiedad `SDKReferenceDirectoryRoot` MSBuild al archivo de proyecto. El valor de esta propiedad es una lista delimitada por punto y coma de directorios en los que residen los SDK de extensión a los que desea hacer referencia.

Diseño de instalación

Los SDK de extensión tienen el siguiente diseño de instalación:

```
\<ExtensionSDKs root>
  \<SDKName>
    \<SDKVersion>
      SDKManifest.xml
      \References
        \<config>
          \<arch>
      \Redist
        \<config>
          \<arch>
      \DesignTime
        \<config>
          \<arch>
```

1. `\<SDKName> \<>SDKVersion` : el nombre y la versión del SDK de extensión se derivan de los nombres de carpeta correspondientes en la ruta de acceso a la raíz del SDK. MSBuild usa esta identidad para buscar el SDK en el disco y Visual Studio muestra esta identidad en la ventana **Propiedades** y el cuadro de diálogo Administrador de **referencias**.
2. *Carpeta de referencias*: los archivos binarios que contienen las API. Podrían ser archivos o ensamblados de metadatos de Windows (WinMD).
3. *Carpeta Redist*: los archivos necesarios para el tiempo de ejecución/depuración y deben empaquetarse como parte de la aplicación del usuario. Todos los binarios deben colocarse debajo de la `<del configuración> \<>arco`, y los nombres binarios deben tener el siguiente formato para garantizar la unicidad: `J<>` de la empresa. `<>` de productos. `<propósito>`. `<extensión>`. *Por ejemplo*, `*Microsoft.Cpp.Build.dll`. Todos los archivos con nombres que pueden colisionar con nombres de archivo de otros SDK (por ejemplo, archivos javascript, css, pri, xaml, png y jpg) deben colocarse debajo de los archivos

| de> | > | > * configuración de<de<configuración de los archivos de red (por ejemplo, javascript, css, pri, xaml, png y jpg) deben colocarse debajo de los archivos de<de configuración de los archivos de sdkname, excepto los archivos asociados con controles XAML. Estos archivos deben colocarse debajo> | de> | *- redist>|<config<arch<componentname.

4. *Carpeta DesignTime*: los archivos que se necesitan solo en tiempo previo a la ejecución/depuración y no deben empaquetarse como parte de la aplicación del usuario. Estos podrían ser documentos XML, bibliotecas, encabezados, binarios en tiempo de diseño de la caja de herramientas, artefactos de MSBuild, etc. Cualquier SDK destinado al consumo por un proyecto nativo debe tener un archivo *SDKName.props*. A continuación se muestra un ejemplo de este tipo de archivo.

```
<?xml version="1.0" encoding="utf-8"?>
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <ExecutablePath>C:\Temp\ExecutablePath;$(ExecutablePath)</ExecutablePath>

    <IncludePath>$(FrameworkSDKRoot)\..\v8.1\ExtensionSDKs\cppimagingssdk\1.0\DesignTime\CommonConfiguration\Neutral\include;$(IncludePath)</IncludePath>
    <AssemblyReferencePath>C:\Temp\AssemblyReferencePath;(AssemblyReferencePath)
  </AssemblyReferencePath>

  <LibraryPath>$(FrameworkSDKRoot)\..\v8.1\ExtensionSDKs\cppimagingssdk\1.0\DesignTime\Debug\ARM;$(LibraryPath)</LibraryPath>
  <SourcePath>C:\Temp\SourcePath\X64;$(SourcePath)</SourcePath>
  <ExcludePath>C:\Temp\ExcludePath\X64;$(ExcludePath)</ExcludePath>
  <_PropertySheetDisplayName>DevILSDK, 1.0</_PropertySheetDisplayName>
</PropertyGroup>
</Project>
```

Los documentos de referencia XML se colocan junto al archivo de referencia. Por ejemplo, el documento de referencia XML para el | > | > ensamblado de configuración de<de la<de<configuración de la configuración de la<de la configuración de la versión de archivo es de la<de la | > | > configuración de la configuración<archivo de la versión de ese documento y la versión localizada de ese documento es "Referencias| > | <configuración<configuración<configuración> | >regional" de la configuración regional de la versión de la<de la configuración regional de la dirección variable de la dirección base de la<de la configuración regional .

5. Carpeta de *configuración*: tres subcarpetas: *Depurar*, *Comercial* y *CommonConfiguration*. Los autores del SDK pueden colocar sus archivos en *CommonConfiguration* cuando se debe consumir el mismo conjunto de archivos SDK, independientemente de la configuración de destino del consumidor del SDK.
6. Carpeta de *arquitectura*: se admiten las siguientes arquitecturas: x86, x64, ARM, neutral. Win32 se asigna a x86 y AnyCPU se asigna a neutral.

SDKManifest.xml

El archivo *SDKManifest.xml* describe cómo Visual Studio debe consumir el SDK. A continuación se muestra un ejemplo:


```

<FileList>
  DisplayName = "My SDK"
  ProductFamilyName = "My SDKs"
  TargetFramework = ".NETCore, version=v4.5.1; .NETFramework, version=v4.5.1"
  MinVSVersion = "14.0"
  MaxPlatformVersion = "8.1"
  AppliesTo = "WindowsAppContainer + WindowsXAML"
  SupportPrefer32Bit = "True"
  SupportedArchitectures = "x86;x64;ARM"
  SupportsMultipleVersions = "Error"
  CopyRedistToSubDirectory = "."
  DependsOn = "SDKB, version=2.0"
  MoreInfo = "https://msdn.microsoft.com/MySDK">
  <File Reference = "MySDK.Sprint.winmd" Implementation = "XNASprintImpl.dll">
  <Registration Type = "Flipper" Implementation = "XNASprintFlipperImpl.dll" />
  <Registration Type = "Flexer" Implementation = "XNASprintFlexerImpl.dll" />
  <ToolboxItems VSCategory = "Toolbox.Default" />
</File>
</FileList>

```

La lista siguiente proporciona los elementos del archivo:

1. **DisplayName:** el valor que aparece en el Administrador de referencias, el Explorador de soluciones, el Examinador de objetos y otras ubicaciones de la interfaz de usuario de Visual Studio.
2. **ProductFamilyName:** el nombre general del producto SDK. Por ejemplo, Biblioteca de Windows para JavaScript (WinJS) el SDK se denomina "Microsoft.WinJS.1.0" y "Microsoft.WinJS.2.0", que pertenecen a la misma familia de productos SDK de la familia de productos, "Microsoft.WinJS". Este atributo permite que Visual Studio y MSBuild realicen esa conexión. Si este atributo no existe, el nombre del SDK se usa como nombre de familia del producto.
3. **FrameworkIdentity:** especifica una dependencia en una o varias bibliotecas de componentes de Windows. El valor de este atributo se coloca en el manifiesto de la aplicación consumidora. Este atributo solo es aplicable a las bibliotecas de componentes de Windows.
4. **TargetFramework:** especifica los ID que están disponibles en el Administrador de referencias y el cuadro de herramientas. Se trata de una lista delimitada por punto y coma de monikers de marco de trabajo de destino, por ejemplo ".NET Framework, versión v2.0; .NET Framework, versión .v4.5.1". Si se especifican varias versiones del mismo marco de destino, el Administrador de referencias utiliza la versión especificada más baja para fines de filtrado. Por ejemplo, si se especifica ".NET Framework, versión v2.0; .NET Framework, versión v4.5.1", el Administrador de referencias usará ".NET Framework, versión .v2.0". Si se especifica un perfil de marco de trabajo de destino específico, el Administrador de referencias solo usará ese perfil para fines de filtrado. Por ejemplo, cuando se especifica "Silverlight, version-v4.0, profile-WindowsPhone", el Administrador de referencias filtra solo el perfil de Windows Phone; un proyecto destinado a Silverlight 4.0 Framework completo no ve el SDK en el Administrador de referencias.
5. **MinVSVersion:** la versión mínima de Visual Studio.
6. **MaxPlatformVersion:** la versión máxima de la plataforma de destino debe usarse para especificar las versiones de plataforma en las que el SDK de extensión no funcionará. Por ejemplo, los proyectos de Windows 8 solo deben hacer referencia al paquete en tiempo de ejecución v11.0 de Microsoft Visual C++ . Por lo tanto, MaxPlatformVersion del proyecto de Windows 8 es 8.0. Esto significa que el Administrador de referencias filtra Microsoft Visual C++ Runtime Package para un proyecto Windows 8.1 de Windows 8.1 y MSBuild produce un error cuando un proyecto hace referencia a él. Nota: este elemento se Visual Studio 2013admite a partir de .
7. **AppliesTo:** especifica los IDD que están disponibles en el Administrador de referencias especificando los tipos de proyecto de Visual Studio aplicables. Se reconocen nueve valores: WindowsAppContainer, VisualC,

VB, CSharp, WindowsXAML, JavaScript, Managed y Native. El autor del SDK puede usar y ("+"), o ("|"), no ("!") operadores para especificar exactamente el ámbito de los tipos de proyecto que se aplican al SDK.

WindowsAppContainer identifica proyectos Tienda Windows 8.x para aplicaciones.

8. SupportPrefer32Bit: Los valores admitidos son "True" y "False". El valor predeterminado es "Verdadero". Si el valor se establece en "False", Tienda Windows 8.x MSBuild devuelve un error para los proyectos (o una advertencia para proyectos de escritorio) si el proyecto que hace referencia al SDK tiene Prefer32Bit habilitado. Para obtener más información acerca de Prefer32Bit, vea página [Compilar, Diseñador de proyectos \(C\)](#) o [Compilar página, Diseñador de proyectos \(Visual Basic\)](#).
9. SupportedArchitectures: una lista delimitada por punto y coma de las arquitecturas que admite el SDK. MSBuild muestra una advertencia si no se admite la arquitectura del SDK de destino en el proyecto de consumo. Si no se especifica este atributo, MSBuild nunca muestra este tipo de advertencia.
10. SupportsMultipleVersions: si este atributo se establece en **Error** o **Advertencia**, MSBuild indica que el mismo proyecto no puede hacer referencia a varias versiones de la misma familia de SDK. Si este atributo no existe o está establecido en **Permitir**, MSBuild no muestra este tipo de error o advertencia.
11. AppX: especifica la ruta de acceso a los paquetes de aplicación para la biblioteca de componentes de Windows en el disco. Este valor se pasa al componente de registro de la biblioteca de componentes de Windows durante la depuración local. La convención de nomenclatura para el nombre de archivo es * <Company>.<> del producto. <> de arquitectura. <> de configuración. <Versión>.appx*. La configuración y la arquitectura son opcionales en el nombre del atributo y el valor del atributo si no se aplican a la biblioteca de componentes de Windows. Este valor solo es aplicable a las bibliotecas de componentes de Windows.
12. CopyRedistToSubDirectory: especifica dónde se deben copiar los archivos de la carpeta *.redist* en relación con la raíz del paquete de la aplicación (es decir, la ubicación del **paquete** elegida en el asistente **Crear paquete** de aplicación) y la raíz de diseño en tiempo de ejecución. La ubicación predeterminada es la raíz del paquete de la aplicación y el diseño **F5**.
13. DependsOn: una lista de identidades de SDK que definen los SDK de los que depende este SDK. Este atributo aparece en el panel de detalles del Administrador de referencias.
14. MoreInfo: la dirección URL de la página web que proporciona ayuda y más información. Este valor se utiliza en el vínculo Más información en el panel derecho del Administrador de referencias.
15. Tipo de registro: especifica el registro de WinMD en el manifiesto de la aplicación y es necesario para WinMD nativo, que tiene un archivo DLL de implementación homólogo.
16. Referencia de archivo: se especifica solo para las referencias que contienen controles o son WinMD nativos. Para obtener información sobre cómo especificar si una referencia contiene controles, consulte [Especificar la ubicación de los elementos del cuadro de herramientas](#) a continuación.

Especifique la ubicación de los elementos de la caja de herramientas

El elemento **ToolBoxItems** del esquema *SDKManifest.xml* especifica la categoría y la ubicación de los elementos del cuadro de herramientas en los SDK de plataforma y extensión. En los ejemplos siguientes se muestra cómo especificar diferentes ubicaciones. Esto es aplicable a las referencias de WinMD o DLL.

1. Coloque los controles en la categoría predeterminada del cuadro de herramientas.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Toolbox.Default"/>
</File>
```

2. Coloque los controles bajo un nombre de categoría determinado.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory= "MyCategoryName"/>
</File>
```

3. Coloque los controles bajo nombres de categoría sin categorías particulares.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Graph">
  <ToolboxItems/>
  <ToolboxItems VSCategory = "Data">
  <ToolboxItems />
</File>
```

4. Coloque los controles bajo diferentes nombres de categoría en Blend y Visual Studio.

```
// Blend accepts a slightly different structure for the category name because it allows a path rather
// than a single category.
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Graph" BlendCategory = "Controls/sample/Graph">
  <ToolboxItems />
</File>
```

5. Enumerar controles específicos de forma diferente en Blend y Visual Studio.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Graph">
  <ToolboxItems/>
  <ToolboxItems BlendCategory = "Controls/sample/Graph">
  <ToolboxItems/>
</File>
```

6. Enumerar controles específicos y colocarlos en la ruta de acceso común de Visual Studio o solo en el grupo todos los controles.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Toolbox.Common">
  <ToolboxItems />
  <ToolboxItems VSCategory = "Toolbox.All">
  <ToolboxItems />
</File>
```

7. Enumerar controles específicos y mostrar solo un conjunto específico en ChooseItems sin que estén en el cuadro de herramientas.

```
<File Reference = "sample.winmd">
  <ToolboxItems VSCategory = "Toolbox.ChooseItemsOnly">
  <ToolboxItems />
</File>
```

Vea también

- [Tutorial: Crear un SDK con C++](#)
- [Tutorial: Crear un SDK con C o Visual Basic](#)

- [Administrar referencias en un proyecto](#)

NuGet frente a SDK como referencia de proyecto

06/04/2020 • 15 minutes to read • [Edit Online](#)

Este artículo está diseñado para ayudar a los desarrolladores a elegir si empaquetar su software como un paquete NuGet o como un kit de desarrollo de software (SDK). En concreto, se describen las diferencias entre los dos cuando se hace referencia a ellos en un proyecto de Visual Studio.

- **NuGet** es un sistema de administración de paquetes de código abierto que simplifica el proceso de incorporación de bibliotecas en un proyecto. Para .NET (incluido .NET Core), NuGet es el mecanismo compatible con Microsoft para compartir código. NuGet define cómo se crean, hospedan y consumen los paquetes para .NET, y proporciona las herramientas para cada uno de esos roles. En Visual Studio, se agregan paquetes NuGet a un proyecto mediante la interfaz de usuario [del Administrador de paquetes](#).
- Un **SDK** es una colección de archivos que Visual Studio trata como un único elemento de referencia. El cuadro de diálogo Administrador de referencias de Visual Studio enumera todos los SDK que son relevantes para el proyecto actual al elegir **Agregar referencia**. Al agregar un SDK a un proyecto, puede tener acceso a todo el contenido de ese SDK a través de IntelliSense, la ventana de Cuadro de herramientas, los diseñadores, el Examinador de objetos, MSBuild, la implementación, la depuración y el empaquetado.

¿Qué mecanismo debo usar?

La tabla siguiente le ayuda a comparar las características de referencia de un SDK con las características de referencia de NuGet.

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
El mecanismo hace referencia a una entidad y, después, todos los archivos y funcionalidades están disponibles.	Y	Para agregar un SDK, use el cuadro de diálogo Administrador de referencias . Todos los archivos y funcionalidades están disponibles durante el flujo de trabajo de desarrollo.	Y	
MSBuild consume automáticamente ensamblados y archivos de metadatos de Windows (.winmd).	Y	Las referencias del SDK se pasan automáticamente al compilador.	Y	
MSBuild consume automáticamente los archivos .h o .lib.	Y	El archivo <i>SDKName.props</i> indica a Visual Studio cómo configurar el directorio de Visual C++, etc., para el consumo automático de archivos .h o .lib.	N	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
MSBuild consume automáticamente los archivos <i>.js</i> o <i>.css</i> .	Y	En el Explorador de soluciones , puede expandir el nodo de referencia del SDK de JavaScript para mostrar archivos <i>.js</i> o <i>.css</i> individuales y, a continuación, generar <code><source include/></code> etiquetas arrastrando esos archivos a sus archivos de origen. El SDK es compatible con F5 y con la instalación de paquetes automática.	Y	
MSBuild agrega automáticamente el control en el Cuadro de herramientas .	Y	El Cuadro de herramientas puede consumir SDK y mostrar controles en las pestañas que especifique.	N	
El mecanismo admite el instalador de Visual Studio para extensiones (VSIX).	Y	VSIX tiene un manifiesto y lógica especiales para crear paquetes de SDK.	Y	VSIX se puede insertar en otro programa de instalación.
El Examinador de objetos enumera las referencias.	Y	El Examinador de objetos obtiene automáticamente la lista de referencias de los SDK y las enumera.	N	
Los archivos y los vínculos se agregan automáticamente al cuadro de diálogo Administrador de referencias (los vínculos de ayuda, etc., se rellenan automáticamente).	Y	En el cuadro de diálogo Administrador de referencias se enumeran automáticamente los SDK, junto con los vínculos de ayuda y la lista de dependencias de SDK.	N	NuGet proporciona su propio cuadro de diálogo Administrar paquetes NuGet .
El mecanismo admite varias arquitecturas.	Y	Los SDK pueden incluir varias configuraciones. MSBuild consume los archivos adecuados para cada configuración de proyecto.	N	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
El mecanismo admite varias configuraciones.	Y	Los SDK pueden incluir varias configuraciones. En función de la arquitectura del proyecto, MSBuild consume los archivos adecuados para cada arquitectura de proyecto.	N	
El mecanismo puede especificar "no para la copia".	Y	Dependiendo de si los archivos se colocan en la carpeta de tiempo de <i>diseño</i> o de <i>redist</i> , puede controlar qué archivos copiar en el paquete de la aplicación consumidora.	N	Debe declarar qué archivos se copian en el manifiesto del paquete.
El contenido aparece en archivos localizados.	Y	Los documentos XML localizados de los SDK se incluyen automáticamente para ofrecer una mejor experiencia en tiempo de diseño.	N	
MSBuild admite el consumo de varias versiones de un SDK simultáneamente.	Y	El SDK admite el consumo de varias versiones simultáneamente.	N	Esto no es equivalente a hacer referencia. No puede tener a la vez más de una versión de archivos NuGet en el proyecto.
El mecanismo admite la especificación de plataformas de destino aplicables, versiones de Visual Studio y tipos de proyectos.	Y	En el cuadro de diálogo Administrador de referencias y en el Cuadro de herramientas solo se muestran los SDK que se aplican a un proyecto, para que los usuarios puedan elegir más fácilmente los SDK adecuados.	Y (parcialmente)	Pivot es la plataforma de destino. No hay filtrado en la interfaz de usuario. Durante la instalación, podría devolver un error.
El mecanismo admite la especificación de información de registro para archivos WinMD nativos.	Y	Puede especificar la correlación entre el archivo .winmd y el archivo .dll en <i>SDKManifest.xml</i> .	N	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
El mecanismo admite la especificación de dependencias en otros SDK.	Y	El SDK solo notifica al usuario, y este debe instalarlos y hacer referencia a ellos de forma manual.	Y	NuGet los extrae automáticamente, sin que se notifique al usuario.
El mecanismo se integra con conceptos de la Microsoft Store, como el manifiesto de la aplicación y el identificador del marco de trabajo.	Y	El SDK debe pasar conceptos específicos de la Tienda para que el empaquetado y F5 funcionen correctamente con los SDK que están disponibles en la Tienda.	N	
El mecanismo se integra con la canalización de depuración de aplicaciones para aplicaciones de Tienda Windows 8.x.	Y	El SDK debe pasar conceptos específicos de la Tienda para que el empaquetado y F5 funcionen correctamente con los SDK que están disponibles en la Tienda.	Y	El contenido de NuGet pasa a formar parte del proyecto. No se necesita ninguna consideración especial de F5.
El mecanismo se integra con los manifiestos de la aplicación.	Y	El SDK debe pasar conceptos específicos de la Tienda para que el empaquetado y F5 funcionen correctamente con los SDK que están disponibles en la Tienda.	Y	El contenido de NuGet pasa a formar parte del proyecto. No se necesita ninguna consideración especial de F5.
El mecanismo implementa archivos que no son de referencia (por ejemplo, el marco de pruebas de implementación en el que se ejecutan pruebas de aplicaciones de Tienda Windows 8.x).	Y	Si coloca los archivos en la carpeta <i>.redist</i> , los archivos se implementan automáticamente.	Y	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
El mecanismo agrega automáticamente los SDK de plataforma en el IDE de Visual Studio.	Y	Si coloca el SDK de Windows 8 o el SDK de Windows Phone en una ubicación específica con un diseño específico, el SDK se integra automáticamente con todas las características de Visual Studio.	N	
El mecanismo admite una máquina de desarrollo limpia. (Es decir, no se requiere ninguna instalación, y funcionará una sencilla recuperación del control de código fuente).	N	Dado que hace referencia a un SDK, debe proteger la solución y el SDK por separado. Puede proteger el SDK desde las dos ubicaciones predeterminadas que no son del Registro desde las que MSBuild itera los SDK (para obtener más información, vea Crear un kit de desarrollo de software). Como alternativa, si una ubicación personalizada consta de los SDK, puede especificar el código siguiente en el archivo de proyecto: <pre><PropertyGroup> <SDKReferenceDirectoryRoot> C:\MySDKs </SDKReferenceDirectoryRoot> </PropertyGroup></pre> Después, proteja los SDK en dicha ubicación.	Y	Puede desproteger la solución y Visual Studio reconocerá inmediatamente los archivos y actuará sobre ellos.
Puede unirse a una gran comunidad existente de autores de paquetes.	N/D	La comunidad es nueva.	Y	
Puede unirse a una gran comunidad existente de consumidores de paquetes.	N/D	La comunidad es nueva.	Y	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
Puede unirse a un ecosistema de asociados (galerías personalizadas, repositorios, etc.).	N/D	Los repositorios disponibles incluyen Visual Studio Marketplace, el Centro de descarga de Microsoft y la Microsoft Store.	Y	
El mecanismo se integra con los servidores de compilación de integración continua para la creación y el consumo de paquetes.	Y	El SDK debe pasar la ubicación protegida (propiedad SDKReferenceDirector yRoot) en la línea de comandos a MSBuild.	Y	
El mecanismo admite las versiones de paquete preliminar y estable.	Y	El SDK admite la adición de referencias a varias versiones.	Y	
El mecanismo admite la actualización automática de los paquetes instalados.	Y	Si se incluye como VSIX o como parte de las actualizaciones automáticas de Visual Studio, el SDK proporciona notificaciones automáticas.	Y	
El mecanismo contiene un archivo .exe independiente para crear y consumir paquetes.	Y	El SDK contiene <i>MSBuild.exe</i> .	Y	
Los paquetes se pueden proteger en el control de versiones.	Y	No se puede proteger nada fuera del nodo Documentos, lo que significa que los SDK de extensión podrían no estar protegidos. El SDK de extensión podría tener un gran tamaño.	Y	
Puede usar una interfaz de PowerShell para crear y consumir paquetes.	Y (consumo), N (creación)	No hay herramientas para crear un SDK. El consumo ejecuta MSBuild en la línea de comandos.	Y	
Puede usar un paquete de símbolos para la compatibilidad con la depuración.	Y	Si coloca archivos .pdb en el SDK, los archivos se recogen automáticamente.	Y	

CARACTERÍSTICA	COMPATIBILIDAD CON EL SDK	NOTAS DEL SDK	COMPATIBILIDAD CON NUGET	NOTAS DE NUGET
El mecanismo admite las actualizaciones automáticas del administrador de paquetes.	N/D	El SDK se revisa con MSBuild.	Y	
El mecanismo admite un formato de manifiesto ligero.	Y	<i>SDKManifest.xml</i> admite muchos atributos, pero normalmente es necesario un pequeño subconjunto.	Y	
El mecanismo está disponible para todas las ediciones de Visual Studio.	Y	El SDK es compatible con todas las ediciones de Visual Studio.	Y	NuGet es compatible con todas las ediciones de Visual Studio.

Vea también

- [Administrar referencias en un proyecto](#)
- [Administrar referencias en un proyecto \(Visual Studio para Mac\)](#)

Tutorial: Crear un SDK con C o Visual Basic

06/04/2020 • 14 minutes to read • [Edit Online](#)

En este tutorial, aprenderá a crear un SDK de biblioteca de matemáticas simple mediante Visual C- y, a continuación, empaquetar el SDK como una extensión de Visual Studio (VSIX). Completará los siguientes procedimientos:

- [Para crear el componente SimpleMath Windows Runtime](#)
- [Para crear el proyecto de extensión SimpleMathVSIX](#)
- [Para crear una aplicación de ejemplo que use la biblioteca de clases](#)

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Para crear el componente SimpleMath Windows Runtime

1. En la barra de menús, elija **Archivo** > **nuevo** > **proyecto**.
2. En la lista de plantillas, expanda **Visual C** o **Visual Basic**, elija el nodo **Tienda Windows** y, a continuación, elija la plantilla **Componente de Windows en tiempo de ejecución**.
3. En el cuadro **Nombre**, especifique **SimpleMathy**, a continuación, elija el botón **Aceptar**.
4. En el **Explorador** de soluciones, abra el menú contextual del nodo de proyecto **SimpleMath** y, a continuación, elija **Propiedades**.
5. Cambie el nombre de **Class1.cs** para **Arithmetic.cs** y actualícelo para que coincida con el código siguiente:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WinRTMath
{
    public sealed class Arithmetic
    {
        public float add(float firstNumber, float secondNumber)
        {
            return (firstNumber + secondNumber);
        }

        public float subtract(float firstNumber, float secondNumber)
        {
            return (firstNumber - secondNumber);
        }

        public float multiply(float firstNumber, float secondNumber)
        {
            return (firstNumber * secondNumber);
        }

        public float divide(float firstNumber, float secondNumber)
        {
            return (firstNumber / secondNumber);
        }
    }
}

```

```

Public NotInheritable Class Arithmetic
    Public Function Add(ByVal firstNumber As Single, ByVal secondNumber As Single) As Single
        Return (firstNumber + secondNumber)
    End Function

    Public Function Subtract(ByVal firstNumber As Single, ByVal secondNumber As Single) As Single
        Return (firstNumber - secondNumber)
    End Function

    Public Function Multiply(ByVal firstNumber As Single, ByVal secondNumber As Single) As Single
        Return (firstNumber * secondNumber)
    End Function

    Public Function Divide(ByVal firstNumber As Single, ByVal secondNumber As Single) As Single
        Return (firstNumber / secondNumber)
    End Function
End Class

```

6. En el **Explorador** de soluciones , abra el menú contextual del nodo **Solución 'SimpleMath'** y, a continuación, elija **Configuration Manager**.

Se abre el cuadro de diálogo Administrador de configuración.

7. En la lista Configuración de la **solución activa**, elija **Liberar**.
8. En la columna **Configuration (Configuración)**, compruebe que la fila **SimpleMath** está establecida en **Release (Liberar)** y, a continuación, elija el botón **Close (Cerrar)** para aceptar el cambio.

IMPORTANT

El SDK del componente SimpleMath solo incluye una configuración. Esta configuración debe ser la compilación de la versión o Microsoft Store las aplicaciones que usan el componente no pasarán la certificación para el archivo .

9. En el **Explorador** de soluciones , abra el menú contextual del nodo de proyecto **SimpleMath** y, a continuación, elija **Compilar**.

Para crear el proyecto de extensión SimpleMathVSIX

1. En el menú contextual del nodo **Solución 'SimpleMath'**, elija **Agregar > nuevo proyecto**.
2. En la lista de plantillas, expanda **Visual C#** o **Visual Basic**, elija el nodo **Extensibilidad** y, a continuación, elija la plantilla **Proyecto VSIX**.
3. En el cuadro **Nombre**, especifique **SimpleMathVSIX**, a continuación, elija el botón **Aceptar**.
4. En el **Explorador** de soluciones , elija el elemento **source.extension.vsixmanifest**.
5. En la barra de menús, elija **Ver > código**.
6. Reemplace el XML existente por el siguiente XML:

```
<PackageManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011"
xmlns:d="http://schemas.microsoft.com/developer/vsx-schema-design/2011">
  <Metadata>
    <Identity Id="SimpleMath" Version="1.0" Language="en-US" Publisher="[YourName]" />
    <DisplayName>SimpleMath Library</DisplayName>
    <Description xml:space="preserve">Basic arithmetic operations in a WinRT-compatible library.
Implemented in C#.</Description>
  </Metadata>
  <Installation Scope="Global" AllUsers="true">
    <InstallationTarget Id="Microsoft.ExtensionSDK" TargetPlatformIdentifier="Windows"
TargetPlatformVersion="v8.0" SdkName="SimpleMath" SdkVersion="1.0" />
  </Installation>
  <Prerequisites>
    <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[14.0,16.0]" />
  </Prerequisites>
  <Dependencies>
    <Dependency Id="Microsoft.Framework.NDP" DisplayName="Microsoft .NET Framework" d:Source="Manual"
Version="4.5" />
  </Dependencies>
  <Assets>
    <Asset Type="Microsoft.ExtensionSDK" d:Source="File" Path="SDKManifest.xml" />
  </Assets>
</PackageManifest>
```

7. En el **Explorador** de soluciones , elija el proyecto **SimpleMathVSIX**.
8. En la barra de menús, elija **Proyecto > agregar nuevo elemento**.
9. En la lista **De elementos comunes**, expanda **Datos**, a continuación, elija **Archivo XML**.
10. En el cuadro **SDKManifest.xml** **Nombre**, especifique y, a continuación, elija el botón **Agregar**.
11. En el **Explorador** de soluciones **SDKManifest.xml** , abra el menú contextual para , elija **Propiedades**, a continuación, cambie el valor de la propiedad **Incluir en VSIX** a **True**.
12. Sustituye el contenido del archivo por el siguiente código XML:

C#

```
<FileList
  DisplayName="WinRT Math Library (CS)"
  MinVSVersion="11.0"
  TargetFramework=".NETCore,version=v4.5"
  AppliesTo="WindowsAppContainer"
  SupportsMultipleVersions="Error"
  MoreInfo="https://msdn.microsoft.com/">
</FileList>
```

Visual Basic

```
<FileList
  DisplayName="WinRT Math Library (VB)"
  MinVSVersion="11.0"
  TargetFramework=".NETCore,version=v4.5"
  AppliesTo="WindowsAppContainer"
  SupportsMultipleVersions="Error"
  MoreInfo="https://msdn.microsoft.com/">
</FileList>
```

13. En el **Explorador** de soluciones , abra el menú contextual del proyecto **SimpleMathVSIX**, elija **Agregar**, a continuación, elija **Nueva carpeta**.
14. Cambie el `references` nombre de la carpeta a .
15. Abra el menú contextual de la carpeta **Referencias**, elija **Agregar**, a continuación, elija **Nueva carpeta**.
16. Cambie el `commonconfiguration` nombre de la subcarpeta a , `neutral` cree una subcarpeta dentro de ella y asigne un nombre a la subcarpeta .
17. Repita los cuatro pasos anteriores, esta `redist` vez cambiando el nombre de la primera carpeta a .

El proyecto ahora contiene la siguiente estructura de carpetas:

```
references\commonconfiguration\netral
redist\commonconfiguration\netral
```

18. En el **Explorador** de soluciones , abra el menú contextual del proyecto **SimpleMath** y, a continuación, elija **Abrir carpeta en el Explorador** de archivos .
19. En el **Explorador** de archivos , desplácese hasta la carpeta *bin-Release*, abra el menú contextual del archivo **SimpleMath.winmd** y, a continuación, elija **Copy**.
20. En el **Explorador** de soluciones , pegue el archivo en la carpeta *references-commonconfiguration-neutral* del proyecto **SimpleMathVSIX**.
21. Repita el paso anterior, pegando el archivo **SimpleMath.pri** en la carpeta *redist-commonconfiguration-neutral* en el proyecto **SimpleMathVSIX**.
22. En el **Explorador** de soluciones , elija **SimpleMath.winmd**.
23. En la barra de menús, elija **Propiedades de vista** > (Teclado: elija la tecla **F4**).
24. En la ventana **Propiedades** , cambie la propiedad Acción de **compilación** a **Contenido**, a continuación, cambie la propiedad **Incluir en VSIX** a **True**.
25. En el **Explorador** de soluciones , repita este proceso para **SimpleMath.pri**.
26. En el **Explorador** de soluciones , elija el proyecto **SimpleMathVSIX**.

27. En la barra de menús, elija **Build** > **Build SimpleMathVSIX**.
28. En el **Explorador** de soluciones , abra el menú contextual del proyecto **SimpleMathVSIX** y, a continuación, elija **Abrir carpeta en el Explorador** de archivos .
29. En el **Explorador** de archivos , desplácese hasta la carpeta *.bin-Release* y, a continuación, ejecute *SimpleMathVSIX.vsix* para instalarlo.
30. Elija el botón **Instalar**, espere a que finalice la instalación y, a continuación, reinicie Visual Studio.

Para crear una aplicación de ejemplo que use la biblioteca de clases

1. En la barra de menús, elija **Archivo** > **nuevo** > **proyecto**.
2. En la lista de plantillas, expanda **Visual C** o **Visual Basic**y, a continuación, elija el nodo Tienda **Windows** .
3. Elija la plantilla **Aplicación en blanco**, asigne al proyecto el nombre **ArithmeticUI**, a continuación, elija el botón **Aceptar**.
4. En el **Explorador** de soluciones , abra el menú contextual del proyecto **ArithmeticUI** y, a continuación, elija **Agregar** > **referencia**.
5. En la lista de tipos de referencia, expanda **Windows**y, a continuación, elija **Extensiones**.
6. En el panel de detalles, elija la extensión **WinRT Math Library**.

Aparece información adicional sobre el SDK. Puede elegir el vínculo Más <https://msdn.microsoft.com/> **información** para abrir , como especificó en el archivo SDKManifest.xml anteriormente en este tutorial.

7. En el cuadro de diálogo Administrador de **referencias**, active la casilla Biblioteca de **matemáticas de WinRT** y, a continuación, elija el botón **Aceptar**.
8. En la barra de menús, elija **Ver** > **Navegador de objetos**.
9. En la lista **Examinar**, elija **Simple Math**.

Ahora puede explorar lo que hay en el SDK.

10. En el **Explorador** de soluciones , abra **MainPage.xaml** y reemplace su contenido por el siguiente XAML:

C#


```

<Page
  x:Class="ArithmeticUI.MainPage"
  IsTabStop="False"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:SimpleMath"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">

  <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <TextBox x:Name="_firstNumber" HorizontalAlignment="Left" Margin="414,370,0,0"
      TextWrapping="Wrap" Text="First Number" VerticalAlignment="Top" Height="32" Width="135"
      TextAlignment="Center"/>
    <TextBox x:Name="_secondNumber" HorizontalAlignment="Left" Margin="613,370,0,0"
      TextWrapping="Wrap" Text="Second Number" VerticalAlignment="Top" Height="32" Width="135"
      TextAlignment="Center"/>
    <Button Content="+" HorizontalAlignment="Left" Margin="557,301,0,0" VerticalAlignment="Top"
      Height="39" Width="49" Click="OnOperatorClick"/>
    <Button Content="-" HorizontalAlignment="Left" Margin="557,345,0,0" VerticalAlignment="Top"
      Height="39" Width="49" Click="OnOperatorClick"/>
    <Button Content="*" HorizontalAlignment="Left" Margin="557,389,0,0" VerticalAlignment="Top"
      Height="39" Width="49" Click="OnOperatorClick"/>
    <Button Content="/" HorizontalAlignment="Left" Margin="557,433,0,0" VerticalAlignment="Top"
      Height="39" Width="49" Click="OnOperatorClick"/>
    <Button Content="=" HorizontalAlignment="Left" Margin="755,367,0,0" VerticalAlignment="Top"
      Height="39" Width="49" Click="OnResultsClick"/>
    <TextBox x:Name="_result" HorizontalAlignment="Left" Margin="809,370,0,0" TextWrapping="Wrap"
      Text="Result" VerticalAlignment="Top" Height="32" Width="163" TextAlignment="Center" IsReadOnly="True"/>
  </Grid>
</Page>

```

Visual Basic

```

<Page
    x:Class="ArithmeticUI.MainPage"
    IsTabStop="False"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:SimpleMath"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <TextBox x:Name="_firstNumber" HorizontalAlignment="Left" Margin="414,370,0,0"
            TextWrapping="Wrap" Text="First Number" VerticalAlignment="Top" Height="32" Width="135"
            TextAlignment="Center"/>
        <TextBox x:Name="_secondNumber" HorizontalAlignment="Left" Margin="613,370,0,0"
            TextWrapping="Wrap" Text="Second Number" VerticalAlignment="Top" Height="32" Width="135"
            TextAlignment="Center"/>
        <Button Content="+" HorizontalAlignment="Left" Margin="557,301,0,0" VerticalAlignment="Top"
            Height="39" Width="49" Click="OnOperatorClick"/>
        <Button Content="-" HorizontalAlignment="Left" Margin="557,345,0,0" VerticalAlignment="Top"
            Height="39" Width="49" Click="OnOperatorClick"/>
        <Button Content="*" HorizontalAlignment="Left" Margin="557,389,0,0" VerticalAlignment="Top"
            Height="39" Width="49" Click="OnOperatorClick"/>
        <Button Content="/" HorizontalAlignment="Left" Margin="557,433,0,0" VerticalAlignment="Top"
            Height="39" Width="49" Click="OnOperatorClick"/>
        <Button Content="=" HorizontalAlignment="Left" Margin="755,367,0,0" VerticalAlignment="Top"
            Height="39" Width="49" Click="OnResultsClick"/>
        <TextBox x:Name="_result" HorizontalAlignment="Left" Margin="809,370,0,0" TextWrapping="Wrap"
            Text="Result" VerticalAlignment="Top" Height="32" Width="163" TextAlignment="Center" IsReadOnly="True"/>
    </Grid>
</Page>

```

11. Actualice **MainPage.xaml.cs** para que coincida con el código siguiente:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

namespace WinRTMathTestCS
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public static string operation = null;

        public MainPage()
        {
            this.InitializeComponent();
        }

        /// <summary>
        /// Invoked when this page is about to be displayed in a Frame.
    }
}

```

```

    /// </summary>
    /// <param name="e">Event data that describes how this page was reached. The Parameter
    /// property is typically used to configure the page.</param>
    protected override void OnNavigatedTo(NavigationEventArgs e)
    {
    }

    /// <summary>
    /// Sets the operator chosen by the user
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void OnOperatorClick(object sender, RoutedEventArgs e)
    {
        operation = (sender as Button).Content.ToString();
    }

    /// <summary>
    /// Calls the SimpleMath SDK to do simple arithmetic
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void OnResultsClick(object sender, RoutedEventArgs e)
    {
        try
        {
            float firstNumber = float.Parse(this._firstNumber.Text);
            float secondNumber = float.Parse(this._secondNumber.Text);

            WinRTMath.Arithmetic math = new WinRTMath.Arithmetic();

            switch (operation)
            {
                case "+":
                    this._result.Text = (math.add(firstNumber, secondNumber)).ToString();
                    break;
                case "-":
                    this._result.Text = (math.subtract(firstNumber, secondNumber)).ToString();
                    break;
                case "*":
                    this._result.Text = (math.multiply(firstNumber, secondNumber)).ToString();
                    break;
                case "/":
                    this._result.Text = (math.divide(firstNumber, secondNumber)).ToString();
                    break;
                default:
                    this._result.Text = "Choose operator";
                    break;
            }
        }
        catch
        {
            this._result.Text = "Enter valid #";
        }
    }
}

```

```

' The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

''' <summary>
''' An empty page that can be used on its own or navigated to within a Frame.
''' </summary>
Public NotInheritable Class MainPage
    Inherits Page

    ''' <summary>
    ''' Invoked when this page is about to be displayed in a Frame.
    ''' </summary>
    ''' <param name="e">Event data that describes how this page was reached. The Parameter
    ''' property is typically used to configure the page.</param>
    Protected Overrides Sub OnNavigatedTo(e As Navigation.NavigationEventArgs)

    End Sub

    Public Shared operation As String = Nothing

    ''' <summary>
    ''' Sets the operator chosen by the user
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub OnOperatorClick(ByVal sender As Object, ByVal e As RoutedEventArgs)
        operation = If(OfType sender Is Button, CType(sender, Button).Content.ToString()
    End Sub

    ''' <summary>
    ''' Calls the SimpleMath SDK to do simple arithmetic
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub OnResultsClick(ByVal sender As Object, ByVal e As RoutedEventArgs)

        Try

            Dim firstNumber As Single = Single.Parse(Me._firstNumber.Text)
            Dim secondNumber As Single = Single.Parse(Me._secondNumber.Text)

            Dim math As New WinRTMath.Arithmetic()

            Select Case (operation)

                Case "+"
                    Me._result.Text = (math.Add(firstNumber, secondNumber)).ToString()

                Case "-"
                    Me._result.Text = (math.Subtract(firstNumber, secondNumber)).ToString()
                Case "*"
                    Me._result.Text = (math.Multiply(firstNumber, secondNumber)).ToString()
                Case "/"
                    Me._result.Text = (math.Divide(firstNumber, secondNumber)).ToString()
                Case Else
                    Me._result.Text = "Choose operator"

            End Select

            Catch
                Me._result.Text = "Enter valid #"
            End Try
        End Sub
    End Class

```

12. Elija la tecla F5 para ejecutar la aplicación.

13. En la aplicación, escriba los dos números, = elija una operación y, a continuación, elija el botón.

Aparece el resultado correcto.

Ha creado y utilizado correctamente un SDK de extensión.

Vea también

- [Tutorial: Crear un SDK con C++](#)
- [Tutorial: Crear un SDK con JavaScript](#)
- [Creación de un kit de desarrollo de software](#)

Tutorial: Crear un SDK con C++

06/04/2020 • 12 minutes to read • [Edit Online](#)

En este tutorial se muestra cómo crear un SDK de biblioteca matemática nativo de C++, empaquetar el SDK como una extensión de Visual Studio (VSIX) y, a continuación, usarlo para crear una aplicación. El tutorial se divide en estos pasos:

- [Para crear las bibliotecas nativas y de Windows runtime](#)
- [Para crear el proyecto de extensión NativeMathVSIX](#)
- [Para crear una aplicación de ejemplo que use la biblioteca de clases](#)

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Para crear las bibliotecas nativas y de Windows runtime

1. En la barra de menús, elija **Archivo > nuevo > proyecto**.
2. En la lista de plantillas, expanda **Visual C++ > Windows Universal**, a continuación, seleccione la plantilla DLL (**aplicaciones universales** de Windows). En el cuadro **Nombre**, especifique y, a continuación, elija el botón **Aceptar**.
3. Actualice *NativeMath.h* para que coincida con el código siguiente.

```
#pragma once

class __declspec(dllexport) BasicMath
{
public:
    BasicMath();
    double add(double firstNumber, double secondNumber);
    double subtract(double firstNumber, double secondNumber);
};
```

4. Actualice *NativeMath.cpp* para que coincida con este código:

```
// NativeMath.cpp : Defines the exported functions for the DLL application.
//

#include "pch.h"
#include "NativeMath.h"

BasicMath::BasicMath()
{
}

double BasicMath::add(double firstNumber, double secondNumber)
{
    return firstNumber + secondNumber;
}

double BasicMath::subtract(double firstNumber, double secondNumber)
{
    return firstNumber - secondNumber;
}
```

5. En el **Explorador de soluciones**, abra el menú contextual **de la solución 'NativeMath'** y, a continuación, elija **Agregar > nuevo proyecto**.
6. En la lista de plantillas, expanda **Visual C++** y, a continuación, seleccione la plantilla **Componente de Windows en tiempo de ejecución**. En el cuadro NativeMathWRT **Nombre**, especifique y, a continuación, elija el botón **Aceptar**.
7. Actualice *Class1.h* para que coincida con este código:

```
#pragma once

namespace NativeMathWRT
{
    public ref class BasicMathWinMD sealed
    {
    public:
        BasicMathWinMD(){};
        double multiply(double firstNumber, double secondNumber);
        double divide(double firstNumber, double secondNumber);
    };
}
```

8. Actualice *Class1.cpp* para que coincida con este código:

```
// Class1.cpp
#include "pch.h"
#include "Class1.h"

using namespace NativeMathWRT;
using namespace Platform;

double BasicMathWinMD::multiply(double firstNumber, double secondNumber)
{
    return firstNumber * secondNumber;
}

double BasicMathWinMD::divide(double firstNumber, double secondNumber)
{
    if(0 == secondNumber)
        return -1;

    return firstNumber / secondNumber;
}
```

9. En la barra de menús, elija **Compilar > Compilar solución**.

Para crear el proyecto de extensión NativeMathVSIX

1. En el **Explorador de soluciones**, abra el menú contextual **de la solución 'NativeMath'** y, a continuación, elija **Agregar > nuevo proyecto**.
2. En la lista de plantillas, expanda **Extensibilidad de Visual C-** > y, a continuación, seleccione **Proyecto VSIX**. En el cuadro **Nombre**, especifique **NativeMathVSIX**, a continuación, elija el botón **Aceptar**.
3. En el **Explorador de soluciones**, abra el menú contextual **de source.extension.vsixmanifest**, a continuación, elija **Ver código**.
4. Utilice el siguiente XML para reemplazar el XML existente.

```
<PackageManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011"
xmlns:d="http://schemas.microsoft.com/developer/vsx-schema-design/2011">
  <Metadata>
    <Identity Id="NativeMathVSIX..c6b3cae1-e7e2-4e71-90f6-21017ea0dff7" Version="1.0" Language="en-US"
Publisher="MyName" />
    <DisplayName>Native Math SDK</DisplayName>
    <Description>Native Math Library w/ Windows Runtime Additions</Description>
  </Metadata>
  <Installation Scope="Global" AllUsers="true">
    <InstallationTarget Id="Microsoft.ExtensionSDK" TargetPlatformIdentifier="UAP"
TargetPlatformVersion="v0.8.0.0" SdkName="NativeMathSDK" SdkVersion="1.0" />
  </Installation>
  <Dependencies>
  </Dependencies>
  <Assets>
    <Asset Type="Microsoft.ExtensionSDK" d:Source="File" Path="SDKManifest.xml" />
  </Assets>
  <Prerequisites>
    <Prerequisite Id="Microsoft.VisualStudio.Component.CoreEditor" Version="[15.0,16.0)"
DisplayName="Visual Studio core editor" />
  </Prerequisites>
</PackageManifest>
```

5. En el **Explorador de soluciones**, abra el menú contextual del proyecto **NativeMathVSIX** y, a continuación, elija **Agregar > nuevo elemento**.
6. En la lista de elementos de **Visual C**, expanda **Datos**, a continuación, seleccione **Archivo XML**. En el

cuadro SDKManifest.xml Nombre, especifique y, a continuación, elija el botón **Aceptar**.

7. Utilice este XML para reemplazar el contenido del archivo:

```
<FileList
  Identity = "NativeMathSDK, Version=1.0"
  DisplayName = "Native Math SDK"
  MinVSVersion = "11.0"
  AppliesTo = "WindowsAppContainer + (CSharp | VB | VisualC)">
  <File Reference="NativeMathWRT.winmd" Implementation="NativeMathWRT.dll" />
</FileList>
```

8. En el **Explorador de soluciones**, en el proyecto **NativeMathVSIX**, cree esta estructura de carpetas:

```
\DesignTime
  \CommonConfiguration
    \Neutral
      \Include
        \Debug
          \x86
\Redist
  \Debug
    \x86
\References
  \CommonConfiguration
    \Neutral
```

9. En el **Explorador de soluciones**, abra el menú contextual de la solución '**NativeMath**' y, a continuación, elija **Abrir carpeta en el Explorador de archivos**.

10. En el **Explorador de archivos**, copie *\$(SolutionRoot)\NativeMath\NativeMath.hy*, a continuación, en el **Explorador de soluciones**, en el proyecto **NativeMathVSIX**, péguelo en la carpeta **\$(SolutionRoot)-NativeMathVSIX-DesignTime-CommonConfiguration-Neutral-Include.**

Copiar *\$(SolutionRoot)-Debug-NativeMath-NativeMath.lib*, a continuación, pegarlo en la carpeta **\$(SolutionRoot)-NativeMathVSIX-DesignTime-Debug-x86.**

Copiar *\$(SolutionRoot)-Debug-NativeMath-NativeMath.dll* y pegarlo en la carpeta **\$(SolutionRoot)-NativeMathVSIX-Redist-Debug-x86.**

Copie *\$(SolutionRoot)-Debug-NativeMathWRT-NativeMathWRT.dll* y péguelo en la carpeta *\$(SolutionRoot)-NativeMathVSIX-Redist-Debug-x86*. Copiar *\$(SolutionRoot)-Debug-NativeMathWRT-NativeMathWRT.winmd* y péguelo en la carpeta *\$(SolutionRoot)-NativeMathVSIX-References-CommonConfiguration-Neutral*.

Copiar *\$(SolutionRoot)-Debug-NativeMathWRT-NativeMathWRT.pri* y pegarlo en la carpeta *\$(SolutionRoot)-NativeMathVSIX-References-CommonConfiguration-Neutral*.

11. En la carpeta **\$(SolutionRoot)-NativeMathVSIX-DesignTime-Debug-x86.** cree un archivo de texto denominado *NativeMathSDK.props*, a continuación, pegue el siguiente contenido en él:

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <NativeMathSDKPath>$(FrameworkSDKRoot)\..\..\UAP\v0.8.0.0\ExtensionSDKs\NativeMathSDK\1.0\
  </NativeMathSDKPath>
    <IncludePath>$(NativeMathSDKPath)DesignTime\CommonConfiguration\Neutral\Include;$(IncludePath)
  </IncludePath>
    <LibraryPath>$(NativeMathSDKPath)DesignTime\Debug\x86;$(LibraryPath)</LibraryPath>
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <Link>
      <AdditionalDependencies>NativeMath.lib;%(AdditionalDependencies)</AdditionalDependencies>
    </Link>
  </ItemDefinitionGroup>
</Project>
```

12. En la barra de menús, elija **Verla > ventana Otras > propiedades** de Windows (Teclado: elija la tecla **F4**).
13. En el Explorador de **soluciones**, seleccione el archivo **NativeMathWRT.winmd**. En la ventana **Propiedades**, cambie la propiedad Acción de **compilación** a **Contenido**, a continuación, cambie la propiedad **Incluir en VSIX** a **True**.

Repita este proceso para el archivo **NativeMath.h**.

Repita este proceso para el archivo **NativeMathWRT.pri**.

Repita este proceso para el archivo **NativeMath.Lib**.

Repita este proceso para el archivo **NativeMathSDK.props**.
14. En el Explorador de **soluciones**, seleccione el archivo **NativeMath.h**. En la ventana **Propiedades**, cambie la propiedad **Incluir en VSIX** a **True**.

Repita este proceso para el archivo **NativeMath.dll**.

Repita este proceso para el archivo **NativeMathWRT.dll**.

Repita este proceso para el archivo **SDKManifest.xml**.
15. En la barra de menús, elija **Compilar > Compilar solución**.
16. En el **Explorador** de soluciones, abra el menú contextual del proyecto **NativeMathVSIX** y, a continuación, elija **Abrir carpeta en el Explorador** de archivos.
17. En el **Explorador** de archivos, desplácese hasta la carpeta *\$SolutionRoot\$-NativeMathVSIX-bin-Debug*, a continuación, ejecute *NativeMathVSIX.vsix* para iniciar la instalación.
18. Elija el botón **Instalar**, espere a que finalice la instalación y, a continuación, abra Visual Studio.

Para crear una aplicación de ejemplo que use la biblioteca de clases

1. En la barra de menús, elija **Archivo > nuevo > proyecto**.
2. En la lista de plantillas, expanda **Visual C++ > Windows Universal** y, a continuación, seleccione Aplicación en **blanco**. En el cuadro **Nombre**, especifique **NativeMathSDKSample**, a continuación, elija el botón **Aceptar**.
3. En el **Explorador** de soluciones, abra el menú contextual del proyecto **NativeMathSDKSample** y, a continuación, elija **Agregar > referencia**.
4. En el cuadro de diálogo **Agregar referencia**, en la lista de tipos de referencia, expanda **Windows universal**, a continuación, seleccione **Extensiones**. Por último, active la casilla de verificación **Native**

Math SDK y, a continuación, elija el botón **Aceptar**.

5. Mostrar las propiedades del proyecto para NativeMathSDKSample.

Las propiedades que definió en *NativeMathSDK.props* se aplicaron al agregar la referencia. Puede comprobar que las propiedades se aplicaron examinando la propiedad **Directorios VC++** de las Propiedades de **configuración** del proyecto .

6. En el **Explorador de soluciones** , abra **MainPage.xaml**, a continuación, use el siguiente XAML para reemplazar su contenido:

```
<Page
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:NativeMathSDKSample"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    x:Class="NativeMathSDKSample.MainPage"
    IsTabStop="false"
    mc:Ignorable="d">

    <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
        <TextBox x:Name="FirstNumber" HorizontalAlignment="Left" Margin="327,123,0,0"
            TextWrapping="Wrap" Text="1" VerticalAlignment="Top" Height="84" Width="259" FontSize="48"/>
        <TextBox x:Name="SecondNumber" HorizontalAlignment="Left" Margin="687,123,0,0"
            TextWrapping="Wrap" Text="1" VerticalAlignment="Top" Height="84" Width="271" FontSize="48"/>
        <Button x:Name="Execute" Content="=" HorizontalAlignment="Left" Margin="478,387,0,0"
            VerticalAlignment="Top" Height="63" Width="332" Click="Execute_Click"/>
        <RadioButton Name="add" Content="Add" HorizontalAlignment="Left" Margin="273,262,0,0"
            VerticalAlignment="Top" GroupName="Operation" IsChecked="True"/>
        <RadioButton Name="subtract" Content="Subtract" HorizontalAlignment="Left" Margin="453,262,0,0"
            VerticalAlignment="Top" GroupName="Operation" IsChecked="False"/>
        <RadioButton Name="multiplyWRT" Content="Multiply(WRT)" HorizontalAlignment="Left"
            Margin="615,262,0,0" VerticalAlignment="Top" GroupName="Operation" IsChecked="False"/>
        <RadioButton Name="divideWRT" Content="Divide(WRT)" HorizontalAlignment="Left"
            Margin="891,262,0,0" VerticalAlignment="Top" GroupName="Operation" IsChecked="False"/>
        <TextBlock Name="resultText" HorizontalAlignment="Left" Margin="478,525,0,0"
            TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" Width="332" Height="70" FontSize="48"/>
    </Grid>
</Page>
```

7. Actualice *Mainpage.xaml.h* para que coincida con este código:

```
//
// MainPage.xaml.h
// Declaration of the MainPage class.
//

#pragma once

#include "MainPage.g.h"

namespace NativeMathSDKSample
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public ref class MainPage sealed
    {
    public:
        MainPage();

    protected:
        virtual void OnNavigatedTo(Windows::UI::Xaml::Navigation::NavigationEventArgs^ e) override;
        virtual void Execute_Click(Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e);
    };
}
```

8. Actualice *MainPage.xaml.cpp* para que coincida con este código:

```
//
// MainPage.xaml.cpp
// Implementation of the MainPage class.
//

#include "pch.h"
#include "MainPage.xaml.h"

#include <sstream>

#include "NativeMath.h"

using namespace NativeMathSDKSample;

using namespace Platform;
using namespace Windows::Foundation;
using namespace Windows::Foundation::Collections;
using namespace Windows::UI::Xaml;
using namespace Windows::UI::Xaml::Controls;
using namespace Windows::UI::Xaml::Controls::Primitives;
using namespace Windows::UI::Xaml::Data;
using namespace Windows::UI::Xaml::Input;
using namespace Windows::UI::Xaml::Media;
using namespace Windows::UI::Xaml::Navigation;

using namespace NativeMathWRT;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

MainPage::MainPage()
{
    InitializeComponent();
}

/// <summary>
/// Invoked when this page is about to be displayed in a Frame.
/// </summary>
/// <param name="e">Event data that describes how this page was reached. The Parameter
/// property is typically used to configure the page.</param>
```

```

void MainPage::OnNavigatedTo(NavigationEventArgs^ e)
{
}

void NativeMathSDKSample::MainPage::Execute_Click(Platform::Object^ sender,
Windows::UI::Xaml::RoutedEventArgs^ e)
{
    double iFirstNumber = _wtof(FirstNumber->Text->Data());
    double iSecondNumber = _wtof(SecondNumber->Text->Data());
    double result = 0;

    BasicMathWinMD^ basicMathWinMD = ref new BasicMathWinMD();
    BasicMath basicMath;

    if(add->IsChecked->Value == true)
    {
        result = basicMath.add(iFirstNumber, iSecondNumber);
    }
    else if(subtract->IsChecked->Value == true)
    {
        result = basicMath.subtract(iFirstNumber, iSecondNumber);
    }
    else if(multiplyWRT->IsChecked->Value == true)
    {
        result = basicMathWinMD->multiply(iFirstNumber, iSecondNumber);
    }
    else if (divideWRT->IsChecked->Value == true)
    {
        result = basicMathWinMD->divide(iFirstNumber, iSecondNumber);
    }

    std::wstringstream s;
    s << result;

    resultText->Text = ref new String(s.str().c_str());
}

```

9. Elija la tecla F5 para ejecutar la aplicación.

10. En la aplicación, escriba dos números, seleccione = una operación y, a continuación, elija el botón.

Aparece el resultado correcto.

En este tutorial se muestra cómo crear y Windows en tiempo de ejecución usar unWindows en tiempo de ejecución SDK de extensión para llamar a una biblioteca y a una biblioteca que no es de biblioteca.

Pasos siguientes

Vea también

- [Tutorial: Crear un SDK con C o Visual Basic](#)
- [Crear un kit de desarrollo de software](#)

Tutorial: Crear un SDK con JavaScript

06/04/2020 • 9 minutes to read • [Edit Online](#)

En este tutorial se enseña cómo usar JavaScript para crear un SDK matemático simple como una extensión de Visual Studio (VSIX). El tutorial se divide en estas partes:

- [Para crear el proyecto SDK de extensión SimpleMathVSIX](#)
- [Para crear una aplicación de ejemplo que use el SDK](#)

Para JavaScript, no hay ningún tipo de proyecto de biblioteca de clases. En este tutorial, el archivo *arithmetic.js* de ejemplo se crea directamente en el proyecto VSIX. En la práctica, se recomienda compilar y probar primero los archivos JavaScript y CSS como una aplicación de la Tienda Windows (por ejemplo, mediante la plantilla Aplicación en blanco) antes de colocarlos en un proyecto VSIX.

Prerrequisitos

Para seguir este tutorial, debe instalar SDK de Visual Studio. Para obtener más información, vea SDK de [Visual Studio](#).

Para crear el proyecto SDK de extensión SimpleMathVSIX

1. En la barra de menús, elija **Archivo** > **nuevo** > **proyecto**.
2. En la lista de categorías de plantilla, en **Visual C**, seleccione **Extensibilidad**, a continuación, seleccione la plantilla **Proyecto VSIX**.
3. En el cuadro de `SimpleMathVSIX` texto **Nombre**, especifique y elija el botón **Aceptar**.
4. Si aparece el **Asistente para paquetes** de Visual Studio, elija el siguiente botón en el **bienvenida** página y, a continuación, en la página 1 de 7, elija el **finalizar** botón. **Next**

Aunque se abre el Diseñador de **manifiestos**, mantendremos este tutorial simple modificando el archivo de manifiesto directamente.

5. En el **Explorador** de soluciones, abra el menú contextual del archivo `source.extension.vsixmanifest` y, a continuación, elija **Ver código**. Utilice este código para reemplazar el contenido existente en el archivo.

```
<?xml version="1.0" encoding="utf-8"?>
<PackageManifest Version="2.0.0" xmlns="http://schemas.microsoft.com/developer/vsx-schema/2011"
xmlns:d="http://schemas.microsoft.com/developer/vsx-schema-design/2011">
  <Metadata>
    <Identity Id="SimpleMathVSIX" Version="1.0" Language="en-US" Publisher="myname" />
    <DisplayName>Simple Math</DisplayName>
    <Description>Does basic arithmetic calculations.</Description>
  </Metadata>
  <Installation Scope="Global" AllUsers="true">
    <InstallationTarget Id="Microsoft.ExtensionSDK" TargetPlatformIdentifier="Windows"
TargetPlatformVersion="v8.0" SdkName="SimpleMath" SdkVersion="1.0" />
  </Installation>
  <Dependencies>
    <Dependency Id="Microsoft.Framework.NDP" DisplayName="Microsoft .NET Framework" d:Source="Manual"
Version="4.5" />
  </Dependencies>
  <Assets>
    <Asset Type="Microsoft.ExtensionSDK" d:Source="File" Path="SDKManifest.xml" />
  </Assets>
</PackageManifest>
```

6. En el **Explorador** de soluciones , abra el menú contextual del proyecto **SimpleMathVSIX** y, a continuación, elija **Agregar > nuevo elemento**.
7. En la categoría **Datos**, seleccione Archivo **SDKManifest.xml** **XML**, asigne un nombre al archivo y elija el botón **Agregar**.
8. En el **Explorador** de soluciones , abra el menú contextual del archivo **SDKManifest.xml** y, a continuación, elija **Abrir** para mostrar el archivo en el **Editor XML**.
9. Agregue el código siguiente al archivo **SDKManifest.xml**.

```
<?xml version="1.0" encoding="utf-8" ?>
<FileList
  DisplayName="Simple Math"
  MinVSVersion="14.0"
  AppliesTo="JavaScript+WindowsAppContainer"
  SupportsMultipleVersions="Error"
  MoreInfo="https://msdn.microsoft.com/">

  <!-- JS -->
  <File Content="js\arithmetic.js" />
</FileList>
```

10. En el **Explorador** de soluciones , en el menú contextual del archivo **SDKManifest.xml**, elija **Propiedades**.
11. En la ventana **Propiedades**, establezca la propiedad **Incluir en VSIX** en **True**.
12. En el **Explorador** de soluciones , en el menú contextual del proyecto **SimpleMathVSIX** , elija **Agregar > nueva carpeta**, a continuación, asigne un nombre a la carpeta **Redist**.
13. Agregue subcarpetas en **Redist** para crear esta estructura de carpetas:
 - Redist\CommonConfiguration, Neutral, SimpleMath, js*
14. En el menú contextual de la carpeta ****js\ **** , elija **Agregar > nuevo elemento**.
15. En **Elementos de Visual C**, seleccione la categoría **Web** y, a continuación, seleccione el elemento **Archivo de JavaScript**. Asigne un **arithmetic.js** nombre al archivo y, a continuación, elija el botón **Agregar**.
16. Inserte el siguiente código en *arithmetic.js*:

```
(function (global) {
    "use strict";
    global.Arithmetic = {
        add: function (firstNumber, secondNumber) {
            return firstNumber + secondNumber;
        },

        subtract: function (firstNumber, secondNumber) {
            return firstNumber - secondNumber;
        },

        multiply: function (firstNumber, secondNumber) {
            return firstNumber * secondNumber;
        },

        divide: function (firstNumber, secondNumber) {
            return firstNumber / secondNumber;
        }
    };
})(this);
```

17. En el **Explorador** de soluciones , en el menú contextual del archivo **arithmetic.js**, elija **Propiedades**. Realice estos cambios de propiedad:
 - Establezca la propiedad **Incluir en VSIX en True**.
 - Establezca la propiedad **Copiar en directorio** de salida en **Copiar siempre**.
18. En el **Explorador** de soluciones , en el menú contextual del proyecto **SimpleMathVSIX**, elija **Compilar**.
19. Una vez completada correctamente la compilación, en el menú contextual del proyecto, elija **Abrir carpeta** en el Explorador de archivos . Navegue a la opción de depuración y ejecútela `SimpleMathVSIX.vsix` para instalarlo.
20. Elija el botón **Instalar** y deje que la instalación se complete.
21. Reinicie Visual Studio.

Para crear una aplicación de ejemplo que use el SDK

1. En la barra de menús, elija **Archivo > nuevo > proyecto**.
2. En la lista de categorías de plantilla, en **JavaScript**, seleccione **Tienda Windows** y, a continuación, seleccione la plantilla **Aplicación en blanco**.
3. En el cuadro `ArithmeticUI` **Nombre**, especifique . Elija el botón **Aceptar** .
4. En el **Explorador** de soluciones , abra el menú contextual del proyecto **ArithmeticUI** y, a continuación, elija **Agregar > referencia**.
5. En **Windows**, elija **Extensiones** y observe que se muestra **Simple Math**.
6. Active la casilla de verificación **Simple Math (Matemáticas simple)** y, a continuación, elija el botón **OK (Aceptar)**.
7. En el **Explorador** de soluciones , en **Referencias**, observe que se muestra la referencia **Simple Math**. Expándalo y observe que hay una carpeta `**js\**` que incluye **arithmetic.js**. Puede abrir **arithmetic.js** para confirmar que se ha instalado el código fuente.
8. Utilice el código siguiente para reemplazar el contenido de *default.htm*.


```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>ArithmeticUI</title>

  <!-- WinJS references -->
  <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
  <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
  <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

  <!-- ArithmeticUI references -->
  <link href="/css/default.css" rel="stylesheet" />
  <script src="/js/default.js"></script>
  <script src="/SimpleMath/js/arithmetic.js"></script>
</head>
<body>
  <form>
    <div id="calculator" class="ms-grid">
      <input name="firstNumber" id="firstNumber" type="number" step="any">
      <div id="operators">
        <button class="operator" type="button">+</button>
        <button class="operator" type="button">-</button>
        <button class="operator" type="button">*</button>
        <button class="operator" type="button">/</button>
      </div>
      <input id="secondNumber" type="number">
      <button class="calculate" type="button">=</button>
      <input id="result" type="number" name="result" disabled="" readonly="">
    </div>
  </form>
</body>
</html>

```

9. Utilice el código siguiente para reemplazar el contenido de *js.default.js*.

```

(function () {
    "use strict";

    var app = WinJS.Application;
    var operation = null;

    function calculateResult() {
        var firstNumber = parseFloat(document.querySelector("#firstNumber").value),
            secondNumber = parseFloat(document.querySelector("#secondNumber").value),
            result = 0;

        if (isNaN(firstNumber) || isNaN(secondNumber)) {
            result = 0;
        }
        else {
            switch (operation) {
                case "+":
                    result = Arithmetic.add(firstNumber, secondNumber);
                    break;
                case "-":
                    result = Arithmetic.subtract(firstNumber, secondNumber);
                    break;
                case "*":
                    result = Arithmetic.multiply(firstNumber, secondNumber);
                    break;
                case "/":
                    result = Arithmetic.divide(firstNumber, secondNumber);
                    break;
                default:
            }
        }
        document.querySelector("#result").value = result.toString();
    }

    app.onactivated = function (args) {
        document.querySelector("#calculator").addEventListener("click", function (event) {
            if (event.target.tagName.toLowerCase() === "button") {
                switch (event.target.className) {
                    case "operator":
                        operation = event.target.innerText;
                        break;
                    case "calculate":
                        calculateResult();
                        break;
                    default:
                        break;
                }
            }
        });
    };

    app.start();
})();

```

10. Sustituya el contenido de la dirección de *archivo .css-default.css* por este código:

```

form {
    display: -ms-grid;
    -ms-grid-rows: 1fr auto 1fr;
    -ms-grid-columns: 1fr auto 1fr;
    height: 100%;
    width: 100%;
}

button, input[type=number] {
    margin-right: 5px;
    -ms-grid-row-align: center;
}

#calculator {
    -ms-grid-column: 2;
    -ms-grid-row: 2;
    display: -ms-grid;
    -ms-grid-rows: 1fr;
    -ms-grid-columns: auto min-content auto auto auto;
}

.ms-grid input {
    width: 5em;
}

#firstNumber {
    -ms-grid-column: 1;
    -ms-grid-row-align: center;
}

#operators {
    -ms-grid-column: 2;
    -ms-grid-row-align: center;
}

#operators button.operator {
    margin-bottom: 5px;
    height: 40px;
}

#secondNumber {
    -ms-grid-column: 3;
}

button.calculate {
    -ms-grid-column: 4;
    -ms-grid-row-align: center;
    height: 40px;
}

#result {
    -ms-grid-column: 5;
}

```

11. Elija la tecla **F5** para compilar y ejecutar la aplicación.

12. En la interfaz de usuario de la aplicación, = escriba dos números, seleccione una operación y, a continuación, elija el botón. Aparece el resultado correcto.

Vea también

- [Creación de un kit de desarrollo de software](#)

Extensibilidad de ".NET" Compiler Platform (Roslyn)

06/04/2020 • 2 minutes to read • [Edit Online](#)

La misión principal de la plataforma de compiladores de .NET ("Roslyn") es abrir los compiladores de C y Visual Basic y permitir que las herramientas y los desarrolladores compartan la información enriquecida que los compiladores tienen sobre los programas. Las herramientas de análisis de código mejoran la calidad del código y los generadores de código ayudan en la construcción de aplicaciones. A medida que las herramientas se vuelven más inteligentes, necesitan acceso a más y más conocimientos de código profundos que solo poseen los compiladores. En lugar de ser traductores opacos (entrada de código fuente y código de objeto), los compiladores de Roslyn ofrecen API que puede usar para tareas relacionadas con el código en sus herramientas y aplicaciones.

La mejor parte es que los compiladores de Roslyn, sus API, ejemplos y tutoriales, y las herramientas reales creadas sobre estas API son todos de código abierto en github.com/dotnet/roslyn. Por favor, vaya al sitio de OSS para obtener más información y empezar a usar Roslyn. Encontrará vínculos para obtener las características más recientes de Visual Basic y de C. que puede usar como usuario final, así como vínculos para empezar a usar como generador de herramientas aprovechando las API de Roslyn.

Vea también

- [Comience con los analizadores Roslyn](#)

Comience con los analizadores Roslyn

06/04/2020 • 2 minutes to read • [Edit Online](#)

Con analizadores de código activos basados en proyectos en Visual Studio, los autores de API pueden enviar análisis de código específico según el dominio como parte de sus paquetes NuGet. Dado que estos analizadores funcionan con la plataforma de compilador de .NET (denominada en código "Roslyn"), pueden producir advertencias en el código a medida que escribe incluso antes de terminar la línea (no hay que esperar a compilar el código para detectar problemas). Los analizadores también pueden exponer una corrección automática de código a través del símbolo del sistema de bombilla de Visual Studio para permitirle limpiar el código inmediatamente.

Introducción

[Resumen de analizadores Roslyn](#)

[Tutorial: Escribir el primer analizador y la corrección del código](#)

[Agregar correcciones de código Tutorial: Proporcionar a los usuarios correcciones para problemas del analizador](#)

[Analizador Roslyn del mundo real](#) que también se puede ver como una [charla](#)

[Varios ejemplos en GitHub, agrupados en tres tipos de analizadores](#)

Vea también

- [Referencia de la versión del paquete de plataforma del compilador .NET](#)
- [Más documentos en el sitio de GitHub OSS](#)
- [Reglas de FxCop implementadas con analizadores Roslyn](#)

Analizadores Roslyn y biblioteca compatible con código para ImmutableArrays

06/04/2020 • 40 minutes to read • [Edit Online](#)

La plataforma del compilador de .NET ("Roslyn") le ayuda a crear bibliotecas compatibles con código. Una biblioteca compatible con código proporciona funcionalidad que puede usar y herramientas (analizadores Roslyn) para ayudarle a utilizar la biblioteca de la mejor manera o para evitar errores. En este tema se muestra cómo crear un analizador Roslyn del mundo real para detectar errores comunes al usar el paquete NuGet [System.Collections.Immutable](#). En el ejemplo también se muestra cómo proporcionar una corrección de código para un problema de código encontrado por el analizador. Los usuarios ven correcciones de código en la interfaz de usuario de bombilla de Visual Studio y pueden aplicar una corrección para el código automáticamente.

Introducción

Necesita lo siguiente para compilar este ejemplo:

- Visual Studio 2015 (no una edición Express) o una versión posterior. Puede usar [visual Studio Community Edition](#) gratuito
- [SDK de Visual Studio](#). También puede comprobar Visual Studio al instalar Visual Studio, comprobar **Visual Studio Extensibility Tools** en **Common Tools** para instalar el SDK al mismo tiempo. Si ya ha instalado Visual Studio, también puede instalar este SDK yendo al menú principal **Archivo** > **nuevo** > **proyecto**, eligiendo **C-** en el panel de navegación izquierdo y, a continuación, elija **Extensibilidad**. Cuando elige la plantilla de proyecto "Instalar las herramientas de extensibilidad de Visual Studio", se le pide que descargue e instale el SDK.
- SDK de [.NET Compiler Platform \("Roslyn"\)](#). También puede instalar este SDK yendo al menú principal **Archivo** > **nuevo** > **proyecto**, eligiendo **C-** en el panel de navegación izquierdo y, a continuación, eligiendo **Extensibilidad**. Cuando elige "Descargar el SDK de .NET Compiler Platform" plantilla de proyecto de ruta de navegación, le pide que descargue e instale el SDK. Este SDK incluye el visualizador de [sintaxis Roslyn](#). Esta útil herramienta le ayuda a averiguar qué tipos de modelo de código debe buscar en el analizador. La infraestructura del analizador llama al código para tipos de modelo de código específicos, por lo que el código solo se ejecuta cuando es necesario y solo puede centrarse en analizar el código relevante.

¿Cuál es el problema?

Imagine que proporciona una biblioteca con compatibilidad

[System.Collections.Immutable.ImmutableArray<T>](#) con `ImmutableArray` (por ejemplo, `int`). Los desarrolladores de CTM tienen mucha experiencia con las matrices de .NET. Sin embargo, debido a la naturaleza de `ImmutableArrays` y las técnicas de optimización utilizadas en la implementación, las intuiciones de desarrolladores de C- hacen que los usuarios de la biblioteca escriban código roto, como se explica a continuación. Además, los usuarios no ven sus errores hasta el tiempo de ejecución, que no es la experiencia de calidad a la que están acostumbrados en Visual Studio con .NET.

Los usuarios están familiarizados con escribir código como el siguiente:

```
var a1 = new int[0];
Console.WriteLine("a1.Length = { 0}", a1.Length);
var a2 = new int[] { 1, 2, 3, 4, 5 };
Console.WriteLine("a2.Length = { 0}", a2.Length);
```

La creación de matrices vacías para rellenar con líneas de código posteriores y el uso de la sintaxis del inicializador de colección son familiares para los desarrolladores de C. Sin embargo, al escribir el mismo código para un `ImmutableArray` se bloquea en tiempo de ejecución:

```
var b1 = new ImmutableArray<int>();
Console.WriteLine("b1.Length = { 0}", b1.Length);
var b2 = new ImmutableArray<int> { 1, 2, 3, 4, 5 };
Console.WriteLine("b2.Length = { 0}", b2.Length);
```

El primer error se debe a que la implementación de `ImmutableArray` usa una estructura para ajustar el almacenamiento de datos subyacente. Las estructuras deben tener constructores `default(T)` sin parámetros para que las expresiones puedan devolver structs con todos los miembros cero o nulo. Cuando el código `b1.Length` tiene acceso, hay un error de desreferencia nula en tiempo de ejecución porque no hay ninguna matriz de almacenamiento subyacente en la estructura `ImmutableArray`. La forma correcta de crear un `ImmutableArray<int>.Empty` `ImmutableArray` vacío es .

El error con inicializadores de `ImmutableArray.Add` colección se produce porque el método devuelve nuevas instancias cada vez que se llama. Dado que `ImmutableArrays` nunca cambia, cuando se agrega un nuevo elemento, se obtiene un nuevo objeto `ImmutableArray` (que puede compartir el almacenamiento por motivos de rendimiento con un `ImmutableArray` existente anteriormente). Dado `b2` que apunta a la primera `Add()` `ImmutableArray` antes de llamar cinco veces, `b2` es un valor predeterminado `ImmutableArray`. Llamar a `Length` en él también se bloquea con un error de desreferencia nula. La forma correcta de inicializar un `ImmutableArray` sin llamar manualmente a `Add` es usar `ImmutableArray.CreateRange(new int[] {1, 2, 3, 4, 5})` .

Buscar tipos de nodos de sintaxis relevantes para activar el analizador

Para comenzar a compilar el analizador, primero averiguar qué tipo de `SyntaxNode` necesita buscar. Inicie el visualizador de **sintaxis** desde el menú **Ver > otro > visualizador de sintaxis Roslyn** de Windows .

Coloque el intercalador del editor en `b1` la línea que declara . Verá que el visualizador de sintaxis `LocalDeclarationStatement` muestra que se encuentra en un nodo del árbol de sintaxis. Este nodo tiene `VariableDeclaration` un , que `VariableDeclarator` a su vez `EqualsValueClause` tiene un , `ObjectCreationExpression` que a su vez tiene un , y finalmente hay un . Al hacer clic en el árbol de nodos del visualizador de sintaxis, la sintaxis de la ventana del editor se resalta para mostrarle el código representado por ese nodo. Los nombres de los subtipos `SyntaxNode` coinciden con los nombres utilizados en la gramática de C.

Crear el proyecto de analizador

En el menú principal, elija **Archivo > nuevo > proyecto**. En el cuadro de diálogo **Nuevo proyecto**, en **Proyectos de C**, en la barra de navegación izquierda, elija **Extensibilidad**, en el panel derecho, elija la plantilla de proyecto **Analizador con corrección de código**. Introduzca un nombre y confirme el cuadro de diálogo.

La plantilla abre un archivo `DiagnosticAnalyzer.cs`. Elija esa pestaña del búfer del editor. Este archivo tiene una clase de analizador (formada a partir `DiagnosticAnalyzer` del nombre que dio al proyecto) que deriva de (un tipo de API Roslyn). La nueva clase `DiagnosticAnalyzerAttribute` tiene un `declaring` el analizador es relevante para el lenguaje de C- para que el compilador detecte y cargue el analizador.

```
[DiagnosticAnalyzer(LanguageNames.CSharp)]
public class ImmutableArrayAnalyzer : DiagnosticAnalyzer
{ }
```

Puede implementar un analizador mediante Visual Basic que tenga como destino el código de C- y viceversa. Es más importante en `DiagnosticAnalyzerAttribute` elegir si el analizador tiene como destino un idioma o ambos. Los analizadores más sofisticados que requieren un modelado detallado del lenguaje solo pueden dirigirse a un único

idioma. Si el analizador, por ejemplo, solo comprueba los nombres de tipo o los nombres de miembros públicos, es posible usar el modelo de lenguaje común que Roslyn ofrece en Visual Basic y C. Por ejemplo, FxCop advierte que `ISerializable` se implementa una clase `SerializableAttribute`, pero la clase no tiene el atributo es independiente del lenguaje y funciona para Visual Basic y el código de C.

Inicializar el analizador

Desplácese hacia abajo `DiagnosticAnalyzer` un poco `Initialize` en la clase para ver el método. El compilador llama a este método al activar un analizador. El método `AnalysisContext` toma un objeto que permite al analizador obtener información de contexto y registrar devoluciones de llamada para eventos para los tipos de código que desea analizar.

```
public override void Initialize(AnalysisContext context) {}
```

Abra una nueva línea en este método y escriba "context." para ver una lista de finalización de IntelliSense. Puede ver en la lista `Register...` de finalización hay muchos métodos para controlar varios tipos de eventos. Por ejemplo, el `RegisterCodeBlockAction` primero, , vuelve a llamar al código de un bloque, que suele ser código entre llaves. Registrarse para un bloque también vuelve a llamar al código para el inicializador de un campo, el valor dado a un atributo o el valor de un parámetro opcional.

Como otro `RegisterCompilationStartAction` ejemplo, , vuelve a llamar al código al inicio de una compilación, lo que resulta útil cuando necesita recopilar el estado en muchas ubicaciones. Puede crear una estructura de datos, por ejemplo, para recopilar todos los símbolos utilizados y cada vez que se llame al analizador para alguna sintaxis o símbolo, puede guardar información sobre cada ubicación en la estructura de datos. Cuando se le llama de nuevo debido al final de la compilación, puede analizar todas las `using` ubicaciones que guardó, por ejemplo, para informar de qué símbolos utiliza el código de cada instrucción.

Mediante el visualizador de **sintaxis**, aprendió que desea que se le llame cuando el compilador procesa un `ObjectCreationExpression`. Utilice este código para configurar la devolución de llamada:

```
context.RegisterSyntaxNodeAction(c => AnalyzeObjectCreation(c),  
    SyntaxKind.ObjectCreationExpression);
```

Se registra para un nodo de sintaxis y se filtra solo para los nodos de sintaxis de creación de objetos. Por convención, los autores del analizador usan una lambda al registrar acciones, lo que ayuda a mantener los analizadores sin estado. Puede usar la característica de Visual Studio `AnalyzeObjectCreation` Generar a partir del **uso** para crear el método. Esto también genera el tipo correcto de parámetro de contexto para usted.

Establecer propiedades para los usuarios del analizador

Para que el analizador aparezca en la interfaz de usuario de Visual Studio de forma adecuada, busque y modifique la siguiente línea de código para identificar el analizador:

```
internal const string Category = "Naming";
```

Cambio de `"Naming"` a `"API Guidance"`.

A continuación, busque y abra el archivo *Resources.resx* en el proyecto mediante el Explorador de **soluciones**. Puede poner una descripción para su analizador, título, etc. Puede cambiar el valor de

`"Don't use ImmutableArray<T> constructor"` todos estos por ahora. Puede colocar argumentos de formato {0} de {1} cadena en la cadena `Diagnostic.Create()` (, , `params` etc.) y, más adelante, cuando llame a , puede proporcionar una matriz de argumentos que se pasarán.

Analizar una expresión de creación de objetos

El `AnalyzeObjectCreation` método toma un tipo diferente de contexto proporcionado por el marco del analizador de código. El `Initialize` método `AnalysisContext` le permite registrar devoluciones de llamada de acción para configurar el analizador. El `SyntaxNodeAnalysisContext`, por ejemplo, tiene un `CancellationToken` que se puede pasar. Si un usuario comienza a escribir en el editor, Roslyn cancelará los analizadores en ejecución para guardar el trabajo y mejorar el rendimiento. Como otro ejemplo, este contexto tiene una propiedad `Node` que devuelve el nodo de sintaxis de creación de objetos.

Obtenga el nodo, que puede suponer que es el tipo para el que filtró la acción del nodo de sintaxis:

```
var objectCreation = (ObjectCreationExpressionSyntax)context.Node;
```

Inicie Visual Studio con el analizador la primera vez

Inicie Visual Studio creando y ejecutando el analizador (presione **F5**). Dado que el proyecto de inicio en el **Explorador** de soluciones es el proyecto VSIX, al ejecutar el código se compila el código y un VSIX y, a continuación, se inicia Visual Studio con ese VSIX instalado. Al iniciar Visual Studio de esta manera, se inicia con un subárbol del Registro distinto para que el uso principal de Visual Studio no se vea afectado por las instancias de prueba al compilar analizadores. La primera vez que se inicia de esta manera, Visual Studio realiza varias inicializaciones similares a la primera vez que inició Visual Studio después de instalarlo.

Cree un proyecto de consola y, a continuación, escriba el código de matriz en las aplicaciones de consola Método principal:

```
var b1 = new ImmutableArray<int>();
Console.WriteLine("b1.Length = {0}", b1.Length);
var b2 = new ImmutableArray<int> { 1, 2, 3, 4, 5 };
Console.WriteLine("b2.Length = {0}", b2.Length);
```

Las líneas de `ImmutableArray` código con tienen ondulaciones porque necesita obtener el `using` paquete NuGet `immutable` y agregar una instrucción al código. Presione el botón del puntero derecho en el nodo del proyecto en el **Explorador** de soluciones y elija **Administrar paquetes NuGet**. En el Administrador de NuGet, escriba "Immutable" en el cuadro de búsqueda y elija el elemento **System.Collections.Immutable** (no elija **Microsoft.Bcl.Immutable**) en el panel izquierdo y presione el botón **Instalar** en el panel derecho. La instalación del paquete agrega una referencia a las referencias del proyecto.

Todavía verá ondulaciones `ImmutableArray` rojas debajo, así que coloque el intercalador en ese identificador y presione **Ctrl + .** (período) para abrir el menú de arreglos `using` sugerido y elegir agregar la instrucción apropiada.

Guardar todo y cerrar la segunda instancia de Visual Studio por ahora para poner en un estado limpio para continuar.

Finalice el analizador utilizando editar y continuar

En la primera instancia de Visual Studio, establezca `AnalyzeObjectCreation` un punto de interrupción al principio del método presionando **F9** con el intercalador en la primera línea.

Vuelva a iniciar el analizador con **F5** y, en la segunda instancia de Visual Studio, vuelva a abrir la aplicación de consola que creó la última vez.

Vuelve a la primera instancia de Visual Studio en el punto de interrupción porque el compilador de Roslyn vio una expresión de creación de objetos y llamó al analizador.

Obtener el nodo de creación de objetos. Pase por encima `objectCreation` de la línea que establece la variable presionando **F10** y, en la **ventana Inmediato**, evalúe la expresión. `objectCreation.ToString()` Verá que el nodo

de sintaxis al `"new ImmutableArray<int>()"` que apunta la variable es el código, justo lo que está buscando.

Obtener `ImmutableArray<T>` objeto. Debe comprobar si el tipo que se está creando es `ImmutableArray`. En primer lugar, obtendrá el objeto que representa este tipo. Compruebe los tipos mediante el modelo semántico para asegurarse de que `ToString()` tiene exactamente el tipo correcto y no se compara la cadena de . Introduzca la siguiente línea de código al final de la función:

```
var immutableArrayOfTType =  
    context.SemanticModel  
        .Compilation  
        .GetTypeByMetadataName("System.Collections.Immutable.ImmutableArray`1");
```

Los tipos genéricos se designan en metadatos con backticks (') y el número de parámetros genéricos. Es por eso que no ves "... `ImmutableArray<T>`" en el nombre de los metadatos.

El modelo semántico tiene muchas cosas útiles que le permiten hacer preguntas sobre símbolos, flujo de datos, duración variable, etc. Roslyn separa los nodos de sintaxis del modelo semántico por varias razones de ingeniería (rendimiento, modelado de código erróneo, etc.). Desea que el modelo de compilación busque la información contenida en las referencias para una comparación precisa.

Puede arrastrar el puntero de ejecución amarillo en el lado izquierdo de la ventana del editor. Arrástrelo hasta la `objectCreation` línea que establece la variable y pase por encima de la nueva línea de código con **F10**. Si coloca el puntero del `immutableArrayOfTType` ratón sobre la variable , verá que hemos encontrado el tipo exacto en el modelo semántico.

Obtenga el tipo de la expresión de creación de objetos. "Tipo" se utiliza de varias maneras en este artículo, pero esto significa que si tiene sesión "nueva Foo", necesita obtener un modelo de Foo. Debe obtener el tipo de la expresión de creación de objetos para ver si es el tipo de `ImmutableArray<T>`. Utilice el modelo semántico de nuevo para obtener información de símbolo para el símbolo de tipo (`ImmutableArray`) en la expresión de creación de objetos. Introduzca la siguiente línea de código al final de la función:

```
var symbolInfo = context.SemanticModel.GetSymbolInfo(objectCreation.Type).Symbol as INamedTypeSymbol;
```

Dado que el analizador debe controlar código incompleto o incorrecto en `using` los búferes del `symbolInfo` `null` editor (por ejemplo, falta una instrucción), debe comprobar si es . Debe obtener un tipo con nombre (`INamedTypeSymbol`) del objeto de información de símbolo para finalizar el análisis.

Compare los tipos. Dado que hay un tipo genérico abierto de `T` que estamos buscando y el tipo en el código es un tipo genérico concreto, consulta la `immutableArrayOfTType` información del símbolo para lo que se construye el tipo (un tipo genérico abierto) y se compara ese resultado con . Introduzca lo siguiente al final del método:

```
if (symbolInfo != null &&  
    symbolInfo.ConstructedFrom.Equals(immutableArrayOfTType))  
{}
```

Informe del diagnóstico. Informar del diagnóstico es bastante fácil. Utilice la regla creada para usted en la plantilla de proyecto, que se define antes de la `Initialize` método. Dado que esta situación en el código es un error, `DiagnosticSeverity.Warning` puede cambiar la línea `DiagnosticSeverity.Error` que inicializó `Rule` para reemplazar (green squiggle) con (red squiggle). El resto de la regla se inicializa a partir de los recursos que editó cerca del principio del tutorial. También debe informar de la ubicación del squiggle, que es la ubicación de la especificación de tipo de la expresión de creación de objetos. Introduzca este código `if` en el bloque:

```
context.ReportDiagnostic(Diagnostic.Create(Rule, objectCreation.Type.GetLocation()));
```

La función debería tener este aspecto (quizás con un formato diferente):

```
private void AnalyzeObjectCreation(SyntaxNodeAnalysisContext context)
{
    var objectCreation = (ObjectCreationExpressionSyntax)context.Node;
    var immutableArrayOfTType =
        context.SemanticModel
            .Compilation
            .GetTypeByMetadataName(
                "System.Collections.Immutable.ImmutableArray`1");
    var symbolInfo = context.SemanticModel.GetSymbolInfo(objectCreation.Type).Symbol as
        INamedTypeSymbol;
    if (symbolInfo != null &&
        symbolInfo.ConstructedFrom.Equals(immutableArrayOfTType))
    {
        context.ReportDiagnostic(
            Diagnostic.Create(Rule, objectCreation.Type.GetLocation()));
    }
}
```

Quite el punto de interrupción para que pueda ver el analizador funcionando (y dejar de volver a la primera instancia de Visual Studio). Arrastre el puntero de ejecución al principio del método y presione **F5** para continuar la ejecución. Cuando vuelva a la segunda instancia de Visual Studio, el compilador comenzará a examinar el código de nuevo y llamará al analizador. Se puede ver un ondulación debajo `ImmutableType<int>` de .

Adición de una "corrección de código" para el problema de código

Antes de comenzar, cierre la segunda instancia de Visual Studio y detenga la depuración en la primera instancia de Visual Studio (donde está desarrollando el analizador).

Agregue una nueva clase. Use el menú contextual (botón de puntero derecho) en el nodo del proyecto en el **Explorador** de soluciones y elija agregar un nuevo elemento. Agregue una `BuildCodeFixProvider` clase denominada . Esta clase debe `CodeFixProvider` derivar de , y deberá usar **Ctrl+.** (período) para invocar la corrección `using` de código que agrega la instrucción correcta. Esta clase también debe anotarse con `ExportCodeFixProvider` attribute y `using` deberá agregar `LanguageNames` una instrucción para resolver la enumeración. Debe tener un archivo de clase con el siguiente código:

```
using Microsoft.CodeAnalysis;
using Microsoft.CodeAnalysis.CodeFixes;

namespace ImmutableArrayAnalyzer
{
    [ExportCodeFixProvider(LanguageNames.CSharp)]
    class BuildCodeFixProvider : CodeFixProvider
    {}
}
```

Stub out miembros derivados. Ahora, coloque el intercalador del `CodeFixProvider` editor en el identificador y presione **Ctrl+.** (período) para eliminar la implementación de esta clase base abstracta. Esto genera una propiedad y un método para usted.

Implemente la propiedad. Rellene el `FixableDiagnosticIds` cuerpo `get` de la propiedad con el siguiente código:

```
return ImmutableArray.Create(ImmutableArrayAnalyzerAnalyzer.DiagnosticId);
```

Roslyn reúne diagnósticos y correcciones haciendo coincidir estos identificadores, que son solo cadenas. La plantilla de proyecto generó un identificador de diagnóstico para usted y puede cambiarlo. El código de la

propiedad solo devuelve el identificador de la clase de analizador.

El **registerCodeFixAsync** método toma un contexto. El contexto es importante porque una corrección de código se puede aplicar a varios diagnósticos o puede haber más de un problema en una línea de código. Si escribe "contexto." en el cuerpo del método, la lista de finalización de IntelliSense le mostrará algunos miembros útiles. Hay un miembro `CancellationToken` que puede comprobar para ver si algo desea cancelar la corrección. Hay un miembro `Document` que tiene muchos miembros útiles y le permite llegar a los objetos de modelo de proyecto y solución. Hay un miembro `Span` que es el inicio y el final de la ubicación del código especificado cuando notificó el diagnóstico.

Haga que el método sea asíncrono. Lo primero que debe hacer es corregir la declaración de método generada para que sea un `async` método. La corrección de código para stubbing out la `async` implementación de una `Task` clase abstracta no incluye la palabra clave aunque el método devuelve un archivo .

Obtenga la raíz del árbol de sintaxis. Para modificar el código, debe generar un nuevo árbol de sintaxis con los cambios que realiza la corrección de código. Necesita el `Document` desde el `GetSyntaxRootAsync` contexto para llamar a `GetSyntaxRootAsync`. Este es un método asíncrono porque hay trabajo desconocido para obtener el árbol de sintaxis, posiblemente incluyendo obtener el archivo del disco, analizarlo y compilar el modelo de código Roslyn para él. La interfaz de usuario de Visual `async` Studio debe responder durante este tiempo, lo que permite. Reemplace la línea de código del método por la siguiente:

```
var root = await context.Document
    .GetSyntaxRootAsync(context.CancellationToken);
```

Busque el nodo con el problema. Se pasa en el intervalo del contexto, pero el nodo que encuentra puede no ser el código que tiene que cambiar. El diagnóstico notificado solo proporcionaba el intervalo para el identificador de tipo (donde pertenecía el squiggle), pero debe reemplazar toda la expresión de creación de objetos, incluida la `new` palabra clave al principio y los paréntesis al final. Agregue el código siguiente al método (y use `Ctrl+.` para añadir `using` una `ObjectCreationExpressionSyntax` declaración para):

```
var objectCreation = root.FindNode(context.Span)
    .FirstAncestorOrSelf<ObjectCreationExpressionSyntax>();
```

Registre la corrección de código para la interfaz de usuario de la bombilla. Al registrar la corrección de código, Roslyn se conecta automáticamente a la interfaz de usuario de bombilla de Visual Studio. Los usuarios finales verán que pueden usar `Ctrl+.` (período) cuando el analizador se `ImmutableArray<T>` ondea un mal uso del constructor. Dado que el proveedor de corrección de código solo se ejecuta cuando hay un problema, puede suponer que tiene la expresión de creación de objetos que estaba buscando. Desde el parámetro `context`, puede registrar la nueva corrección de `RegisterCodeFixAsync` código agregando el siguiente código al final del método:

```
context.RegisterCodeFix(
    CodeAction.Create("Use ImmutableArray<T>.Empty",
        c => ChangeToImmutableArrayEmpty(objectCreation,
            context.Document,
            c)),
    context.Diagnostics[0]);
```

Debe colocar el intercalador del editor `CodeAction` en el identificador y, a continuación, usar `Ctrl+.` (período) para agregar `using` la declaración adecuada para este tipo.

A `ChangeToImmutableArrayEmpty` continuación, coloque el intercalador del editor en el identificador y utilice `Ctrl+.` de nuevo para generar este código auxiliar de método para usted.

Este último fragmento de código que agregó `CodeAction` registra la corrección de código pasando un identificador

de diagnóstico y el identificador de diagnóstico para el tipo de problema encontrado. En este ejemplo, solo hay un identificador de diagnóstico para el que este código proporciona correcciones, por lo que puede pasar el primer elemento de la matriz de identificadores de diagnóstico. Al crear `CodeAction` el, se pasa el texto que la interfaz de usuario de bombilla debe usar como descripción de la corrección de código. También se pasa una función que toma un `CancellationToken` y devuelve un nuevo documento. El nuevo documento tiene un nuevo árbol de `ImmutableArray.Empty` sintaxis que incluye el código parchado que llama a `.`. Este fragmento de código utiliza una expresión lambda para que pueda cerrarse sobre el nodo `objectCreation` y el documento del contexto.

Construya el nuevo árbol de sintaxis. En `ChangeToImmutableArrayEmpty` el método cuyo código auxiliar generó anteriormente, `ImmutableArray<int>.Empty;` escriba la línea de código: `.` Si vuelve a ver la ventana de herramientas Visualizador de **sintaxis**, puede ver que esta sintaxis es un nodo `SimpleMemberAccessExpression`. Eso es lo que este método necesita para construir y devolver en un nuevo documento.

El primer `ChangeToImmutableArrayEmpty` cambio es `async Task<Document>` agregar antes porque los generadores de código no pueden asumir que el método debe ser asíncrono.

Rellene el cuerpo con el código siguiente para que el método tenga un aspecto similar al siguiente:

```
private async Task<Document> ChangeToImmutableArrayEmpty(
    ObjectCreationExpressionSyntax objectCreation, Document document,
    CancellationToken c)
{
    var generator = SyntaxGenerator.GetGenerator(document);
    var memberAccess =
        generator.MemberAccessExpression(objectCreation.Type, "Empty");
    var oldRoot = await document.GetSyntaxRootAsync(c);
    var newRoot = oldRoot.ReplaceNode(objectCreation, memberAccess);
    return document.WithSyntaxRoot(newRoot);
}
```

Tendrá que poner el intercalador del `SyntaxGenerator` editor en el identificador y usar **Ctrl+.** (período) para agregar `using` la declaración adecuada para este tipo.

Este código `SyntaxGenerator` usa `.`, que es un tipo útil para construir código nuevo. Después de obtener un generador para `ChangeToImmutableArrayEmpty` el `MemberAccessExpression` documento que tiene el problema de código, llama `.`, pasando el tipo que tiene el miembro al que queremos tener acceso y pasando el nombre del miembro como una cadena.

A continuación, el método recupera la raíz del documento y, dado que esto puede implicar un trabajo arbitrario en el caso general, el código espera esta llamada y pasa el token de cancelación. Los modelos de código Roslyn son inmutables, como trabajar con una cadena .NET; cuando se actualiza la cadena, se obtiene un nuevo objeto de cadena a cambio. Cuando se `ReplaceNode` llama, se obtiene un nuevo nodo raíz. La mayor parte del árbol de sintaxis se `objectCreation` comparte (porque es `memberAccess` inmutable), pero el nodo se reemplaza por el nodo, así como todos los nodos primarios hasta la raíz del árbol de sintaxis.

Pruebe la corrección de código

Ahora puede presionar **F5** para ejecutar el analizador en una segunda instancia de Visual Studio. Abra el proyecto de consola que utilizó anteriormente. Ahora debería ver que la bombilla aparece donde `ImmutableArray<int>` está la nueva expresión de creación de objetos. Si pulsa **Ctrl+.** (período), verá la corrección de código y verá una vista previa de la diferencia de código generada automáticamente en la interfaz de usuario de la bombilla. Roslyn crea esto para ti.

Consejo Profesional: Si inicia la segunda instancia de Visual Studio y no ve la bombilla con la corrección de código, es posible que deba borrar la caché de componentes de Visual Studio. Borrar la memoria caché obliga a Visual Studio a volver a examinar los componentes, por lo que Visual Studio debe recoger el componente más

reciente. En primer lugar, cierre la segunda instancia de Visual Studio. A continuación, en el Explorador de **Windows**, desplácese hasta `%LOCALAPPDATA%-Microsoft-VisualStudio-16.0Roslyn\`. (El "16.0" cambia de versión a versión con Visual Studio.) Elimine el subdirectorio `ComponentModelCache`.

Hablar de vídeo y terminar el proyecto de código

Puede ver este ejemplo desarrollado y discutido más adelante en [esta charla](#). La charla demuestra el analizador de trabajo y le guía a través de la construcción de la misma.

Puede ver todo el código terminado [aquí](#). Las subcarpetas `DoNotUseImmutableArrayCollectionInitializer` y `DoNotUseImmutableArrayCtor` tienen cada una un archivo de C- para buscar problemas y un archivo de C- que implementa las correcciones de código que aparecen en la interfaz de usuario de bombilla de Visual Studio. Tenga en cuenta que el código terminado tiene un poco más de abstracción para evitar la obtención de la `immutableArray T>` objeto de tipo una y otra vez. Utiliza acciones registradas anidadas para guardar el objeto de tipo en un contexto que está disponible siempre que se ejecuten las subacciones (analizar la creación de objetos y analizar las inicializaciones de la colección).

Vea también

- [\Charla de la construcción de 2015](#)
- [Código completado en GitHub](#)
- [Varios ejemplos en GitHub, agrupados en tres tipos de analizadores](#)
- [Otros documentos en el sitio de GitHub OSS](#)
- [Reglas de FxCop implementadas con analizadores Roslyn en GitHub](#)

Referencia de la versión del paquete de plataforma del compilador .NET

06/04/2020 • 2 minutes to read • [Edit Online](#)

En la tabla siguiente se muestra qué versiones del paquete de la [plataforma del compilador de .NET \(Roslyn\)](#) se admiten para diferentes versiones de Visual Studio.

Por ejemplo, para asegurarse de que el analizador personalizado funciona en todas las versiones de Visual Studio 2017, debe tener como destino la versión 2.0 de Microsoft.Net.Compilers.

VERSIÓN DEL PAQUETE ROSLYN	VERSIÓN MÍNIMA COMPATIBLE CON VISUAL STUDIO
3.x	Visual Studio 2019
2.10.0	Visual Studio 2017, versión 15.9
2.9.0	Visual Studio 2017, versión 15.8
2.8.2	Visual Studio 2017 versión 15.7
2.7.0	Visual Studio 2017, versión 15.6
2.6.1	Versión 15.5 de Visual Studio 2017
2.4.0	Visual Studio 2017, versión 15.4
2.3.2	Visual Studio 2017 versión 15.3
2.2.0	Visual Studio 2017, versión 15.2
2.1.0	Visual Studio 2017, versión 15.1
2.0.0	Visual Studio 2017 RTM
1.3.2	Actualización de Visual Studio 2015 3
1.2.2	Actualización de Visual Studio 2015 2
1.1.1	Actualización de Visual Studio 2015 1
1.0.1	Visual Studio 2015 RTM

TIP

Para los paquetes Roslyn donde la versión mínima compatible con Visual Studio es una versión de Visual Studio 2017, todas las versiones de Visual Studio 2019 también se admiten porque vinieron más tarde.

Vea también

- [SDK de la plataforma del compilador de .NET](#)
- [Comience con los analizadores Roslyn](#)