

Curso de Patrones y Buenas Prácticas .Net

Entorno Integrado de Desarrollo (IDE)

Plataforma: Visual Studio .Net

Framework: Microsoft .Net Framework

Documento de Referencia y Capacitación
(Este documento está sujeto a cambios)

Fecha de Creación: 03-Jun-2010

Versión: 2.0.0.0

Autor: Julio Cesar Robles Uribe

Tabla de Contenido

Introducción al entorno IDE (Visual C#)	6
Herramientas de Visual C#.....	7
Cómo expone las herramientas el IDE.....	8
Ventanas del Editor y del Diseñador de Windows Forms.....	8
Explorador de soluciones y Diseñador de proyectos	8
Ventanas Compilador, Depurador y Lista de errores	9
Personalizar el IDE	10
Crear un proyecto (Visual C#)	11
Crear un proyecto nuevo	11
¿Qué contiene el proyecto?	12
Propiedades.....	12
Referencias.....	13
Recursos.....	13
Formularios	13
Otro archivos de código fuente	14
Modificar las propiedades de un proyecto (Visual C#)	14
Diseñar una interfaz de usuario (Visual C#).....	16
Agregar controles.....	17
Establecer propiedades.....	18
Controlar eventos.....	18
Tutorial: Crear un formulario Windows Forms sencillo	18
Para crear un formulario Windows Forms	19
Para escribir el código de la aplicación	19
Para probar la aplicación.....	19
Editar código (Visual C#).....	20
IntelliSense	20
Listas de finalización	20
Información rápida.....	21
Lista de miembros	21
Información de parámetros.....	21
Agregar using.....	22
Refactorización	22
Fragmentos de código.....	22
Subrayado con líneas onduladas.....	22
Ayudas de legibilidad	23

Esquematización.....	23
Colorización	23
Desplazamientos y búsquedas (Visual C#).....	24
Vista de clases	24
Exploración CTRL-TAB.....	25
Barras de exploración.....	26
Buscar en archivos	26
Generar y depurar (Visual C#).....	27
Configuración de generación	27
Errores al generar.....	27
Configuraciones de lanzamiento y de depuración	28
Depuración	29
Para practicar la depuración de una aplicación.....	31
Agregar y editar recursos (Visual C#)	34
Agregar recursos a proyectos	34
Editar recursos	34
Compilar recursos en ensamblados	35
Obtener acceso a los recursos en tiempo de ejecución	35
Recursos en ensamblados satélite.....	35
Obtener Ayuda (Visual C#).....	36
Ayuda en línea y local.....	36
Búsqueda con F1	37
Buscar	37
Índice	38
Tabla de contenido	38
Cómo	38
Implementar aplicaciones de C#.....	39
Implementación ClickOnce	39
Windows Installer.....	39
Características del editor de código de Visual C#	40
Refactorización.....	41
Refactorización de varios proyectos	41
Obtener vista previa de cambios (Cuadro de diálogo).....	41
Refactorización tolerante a errores	41
Fragmentos de código (C#)	42
Utilizar fragmentos de código	42
Crear fragmentos de código	42

Color del código.....	43
Símbolos (token)	43
Palabras clave contextuales	43
Colores de coincidencia de llaves.....	44
Color en negrita.....	44
Color de resaltado	45
Ejemplo.....	45
Configuración de color.....	45
Metadatos como origen	46
Ver metadatos como origen en el Editor de código	46
Ver metadatos como origen en la ventana Definición de código.....	47
Valores de configuración del IDE de Visual C#	48
Ventanas y vistas	48
Teclado	49

Introducción al entorno IDE (Visual C#)

El entorno de desarrollo integrado (IDE) de Visual C# es un conjunto de herramientas de desarrollo expuestas a través de una interfaz de usuario común. Algunas de las herramientas se comparten con otros lenguajes de Visual Studio, y otras, como el compilador de C#, son únicas de Visual C#. La documentación de esta sección describe de forma general cómo utilizar las herramientas más importantes de Visual C# mientras se trabaja en el IDE en distintas fases del proceso de desarrollo.

Para mayor referencia y documentación del Entorno integrado de desarrollo y sobre el lenguaje de programación C#, puede consultar

IDE C# Express

<http://msdn.microsoft.com/es-es/library/sbht894x.aspx>

[http://msdn.microsoft.com/es-es/library/ms173063\(v=VS.80\).aspx](http://msdn.microsoft.com/es-es/library/ms173063(v=VS.80).aspx)

Manual de C#

<http://msdn.microsoft.com/es-es/library/zkxk2fwf.aspx>

Guía del Programador C#

<http://msdn.microsoft.com/es-es/library/67ef8sbd.aspx>

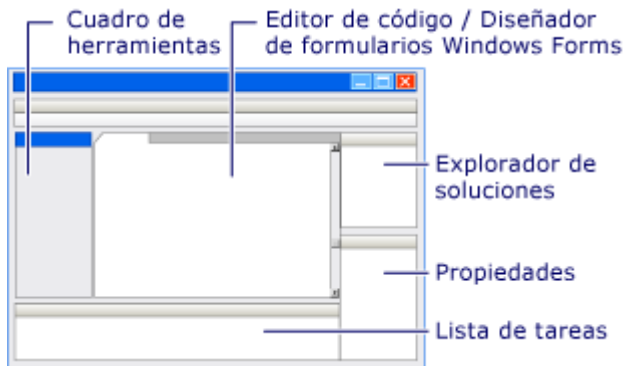
Herramientas de Visual C#

A continuación se detallan las herramientas y ventanas más importantes de Visual C#. Las ventanas de la mayoría de estas herramientas se pueden abrir desde el menú **Ver**.

- El Editor de código, para escribir código fuente.
- El compilador de C#, para convertir el código fuente de C# en un programa ejecutable.
- El depurador de Visual Studio, para probar el programa.
- El **Cuadro de herramientas** y el **Diseñador**, para desarrollar rápidamente interfaces de usuario con el mouse.
- El **Explorador de soluciones**, para ver y administrar archivos de proyecto y configuraciones.
- El **Diseñador de proyectos**, para configurar opciones del compilador, rutas de implementación, recursos, etc.
- La **Vista de clases**, para desplazarse por el código fuente según los tipos, no los archivos.
- La **Ventana Propiedades**, para configurar propiedades y eventos en los controles de la interfaz de usuario.
- El Examinador de objetos, para ver los métodos y clases disponibles en las bibliotecas de vínculos dinámicos, incluidos los ensamblados de .NET Framework y los objetos COM.
- Document Explorer, para explorar y buscar la documentación del producto en su equipo local y en Internet.

Cómo expone las herramientas el IDE

Puede interactuar con las herramientas a través de ventanas, menús, páginas de propiedades y asistentes en el IDE. El IDE básico tiene un aspecto similar al siguiente:



Puede tener acceso rápidamente a las ventanas de herramientas o archivos abiertos presionando CTRL + TAB.

Ventanas del Editor y del Diseñador de Windows Forms

El Editor de código y el Diseñador de Windows Forms utilizan la ventana principal grande. Para alternar entre la vista de código y la vista Diseño, puede presionar F7 o hacer clic en **Código** o **Diseñador** en el menú **Ver**. En la vista Diseño, puede arrastrar controles a la ventana desde el **Cuadro de herramientas**, que se puede hacer visible haciendo clic en la ficha **Cuadro de herramientas** del margen izquierdo.

La ventana **Propiedades** de la parte inferior derecha sólo contiene información en vista Diseño. Permite establecer las propiedades y enlazar los eventos para los controles de la interfaz de usuario, como botones, cuadros de texto, etc. Cuando esta ventana se establece como **Ocultar automáticamente**, se contrae en el margen derecho cada vez que se cambia a **Vista Código**.

Explorador de soluciones y Diseñador de proyectos

La ventana de la parte superior derecha es el **Explorador de soluciones**, que muestra todos los archivos del proyecto en una vista de árbol jerárquica. Cuando se utiliza el menú **Proyecto** para agregar nuevos archivos al proyecto, se verán reflejados en el **Explorador de soluciones**. Además de los archivos, el **Explorador de soluciones** también muestra la configuración del proyecto y las referencias a las bibliotecas externas que necesita la aplicación.

Para obtener acceso a las páginas de propiedades del **Diseñador de proyectos**, haga clic con el botón secundario del mouse en el nodo **Propiedades** del **Explorador de soluciones** y, a continuación, haga clic en **Abrir**. Utilice estas páginas para modificar opciones de generación, requisitos de seguridad, detalles de implementación y muchas otras propiedades del proyecto.

Ventanas Compilador, Depurador y Lista de errores

El compilador de C# no tiene ninguna ventana porque no es una herramienta interactiva, pero puede establecer sus opciones en el **Diseñador de proyectos**. Cuando se hace clic en **Generar** en el menú **Generar**, el IDE invoca el compilador de C#. Si la generación se realiza correctamente, el panel de estado muestra un mensaje Generación satisfactoria. Si se producen errores de generación, aparece la ventana **Lista de errores** aparece debajo de la ventana del editor/diseñador con una lista de errores. Haga doble clic en un error para ir a la línea de código fuente que presenta el problema. Presione F1 para consultar la documentación de Ayuda correspondiente al error resaltado.

El depurador tiene distintas ventanas que muestran valores de variables e información de tipos a medida que se ejecuta la aplicación. Puede utilizar la ventana Editor de código mientras depura el programa para especificar una línea en la que desee hacer una pausa durante la ejecución en el depurador y para recorrer el código línea a línea.

Personalizar el IDE

En Visual C#, todas las ventanas pueden ser acoplables o flotantes, estar ocultas o visibles o cambiarse de ubicación. Para cambiar el comportamiento de una ventana, haga clic en el icono de flecha abajo o de pin de la barra de título y seleccione entre las opciones disponibles. Para mover una ventana acoplada a una nueva ubicación, arrastre la barra de título hasta que aparezcan los iconos de colocación de la ventana. Mantenga presionado el botón primario del mouse y mueva el puntero del mouse sobre el icono en la nueva ubicación. Coloque el puntero sobre los iconos izquierdo, derecho, superior o inferior para acoplar la ventana en el lado especificado. Coloque el puntero sobre el icono del medio para transformar la ventana en una ventana con fichas. Mientras coloca el puntero, aparece un rectángulo azul semitransparente que indica dónde se acoplará la ventana en la nueva ubicación.



Puede personalizar muchos otros aspectos del IDE haciendo clic en **Opciones** en el menú **Herramientas**.

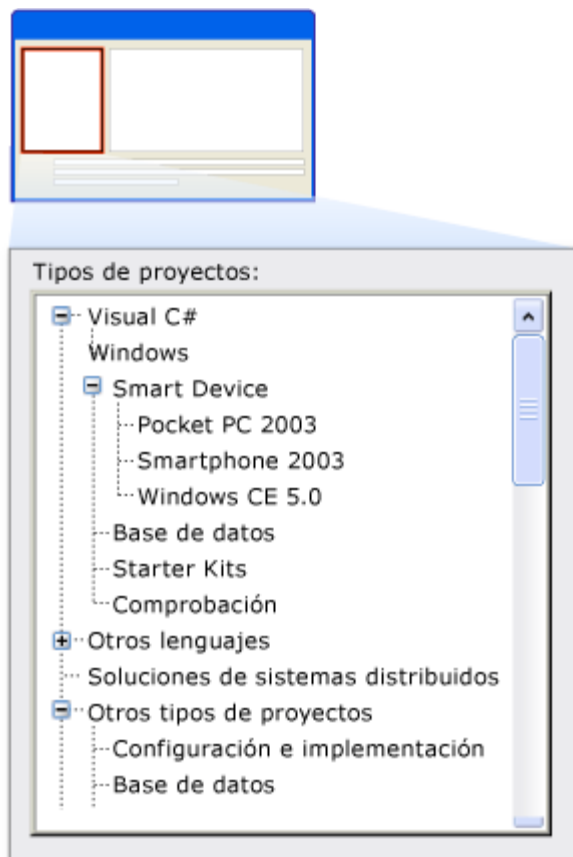
Crear un proyecto (Visual C#)

Cuando esté listo para empezar a programar, el primer paso es preparar un proyecto. El proyecto contiene todo el material necesario para la aplicación. Además de los archivos de código fuente, incluye los archivos de recursos, como iconos, las referencias a archivos externos de los que depende la aplicación y los datos de configuración, como los valores del compilador. Cuando se genera un proyecto, Visual C# invoca al compilador de C# y otras herramientas internas para crear un ensamblado ejecutable con los archivos del proyecto.

Crear un proyecto nuevo

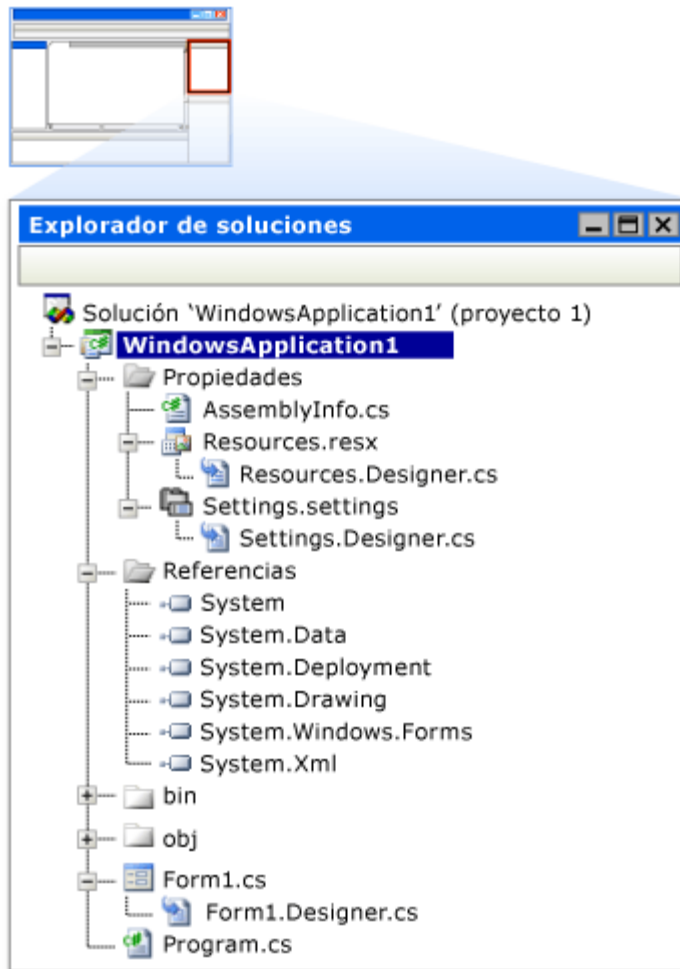
Para crear un nuevo proyecto, haga clic en el menú **Archivo**, elija **Nuevo** y haga clic en **Proyecto**.

La ilustración siguiente muestra el cuadro de diálogo **Nuevo proyecto**. Como puede ver, **Visual C#** está seleccionado de forma predeterminada en la ventana de la izquierda, y en la derecha, tiene la opción de elegir entre seis o más plantillas de proyecto. Si expande el nodo **Smart Device** u **Otros tipos de proyectos** de la izquierda, aparecerán distintos tipos de proyectos en el lado derecho.



Los Starter kits son otro tipo de plantilla de proyecto. Si instala un starter kit, verá que aparece en el cuadro de diálogo **Nuevo proyecto**.

Después de seleccionar una plantilla de proyecto y hacer clic en **Aceptar**, Visual Studio crea el proyecto y todo está listo para comenzar la codificación. Los archivos de proyecto, referencias, configuración y recursos son visibles en la ventana **Explorador de soluciones**, situada a la derecha.



¿Qué contiene el proyecto?

Propiedades

El nodo **Propiedades** representa opciones de configuración que se aplican a la totalidad del proyecto y se almacenan en el archivo .csproj de su carpeta de soluciones. Esta configuración incluye opciones de compilación, seguridad e implementación, entre muchas otras. Para modificar el proyecto, se utiliza el **Diseñador de proyectos**, que es un conjunto de **Páginas de propiedades** a las que se tiene acceso haciendo clic con el botón secundario del mouse en **Propiedades**, y seleccionando **Abrir**.

Referencias

En el contexto de un proyecto, una referencia identifica simplemente un archivo binario que la aplicación necesita para poder ejecutarse. Normalmente, una referencia identifica un archivo DLL, como uno de los archivos de la biblioteca de clases de .NET Framework. También puede hacer referencia a un ensamblado .NET (denominado shim) que permite que la aplicación llame a los métodos de un objeto COM o de un archivo DLL nativo de Win32. Si su programa crea una instancia de una clase definida en otro ensamblado, debe agregar una referencia al archivo correspondiente en el proyecto antes de compilarlo. Para agregar una referencia, haga clic en **Agregar referencia** en el menú **Proyecto**. Todos los proyectos de C# incluyen de forma predeterminada una referencia a mscorlib.dll, que contiene las clases básicas de .NET Framework. Puede agregar las referencias a archivos DLL de .NET Framework adicionales y a otros archivos haciendo clic en el Menú **Proyecto** y seleccionando **Agregar referencia**.

Nota

No confunda el concepto de una referencia de proyecto con el concepto de tipos de referencia en C# u otros lenguajes de programación. El primero se refiere a un archivo y a su ubicación en disco. El segundo se refiere a los tipos de C#, que se declaran con la palabra clave **class**.

Recursos

Los recursos son datos que se incluyen con la aplicación pero que se pueden almacenar de forma que pueden modificarse con independencia del resto del código fuente. Por ejemplo, puede almacenar todas las cadenas como recursos en lugar de codificarlas directamente en el código fuente. De esta forma, podrá traducir las cadenas a distintos idiomas más adelante y agregarlas a la carpeta de la aplicación que distribuye a los clientes sin necesidad de volver a compilar el ensamblado. Los cinco tipos de recursos definidos por Visual C# son: cadenas, imágenes, iconos, audio y archivos. Para agregar, quitar o modificar recursos, utilice el **Diseñador de recursos**, al que se tiene acceso en la ficha **Recursos** del **Diseñador de proyectos**.

Formularios

Cuando se crea un proyecto de formularios Windows Forms, Visual C# agrega de forma predeterminada un formulario al proyecto y lo llama Form1. Los dos archivos que representan el formulario se llaman Form1.cs y Form1.designer.cs. El código se escribe en Form1.cs. El Diseñador de Windows Forms escribe en el archivo designer.cs el código que implementa todas las acciones que se realizaron al arrastrar y colocar objetos desde el Cuadro de herramientas.

Puede agregar un nuevo formulario haciendo clic en el elemento de menú **Proyecto** y seleccionando **Agregar Windows Forms**. Cada formulario tiene dos archivos asociados. Form1.cs, o como haya decidido llamarlo, contiene el código fuente que se escribe para configurar el formulario y sus controles, como cuadros de lista y cuadros de texto, y responde a eventos tales como los clics en botones y las pulsaciones de teclas. En los proyectos de formularios Windows Forms simples, la mayoría o incluso la totalidad de la codificación se hace en este archivo.

El archivo Designer.cs contiene el código fuente que escribe el **Diseñador de formularios** cuando se arrastran controles y se colocan en el formulario, cuando se establecen las propiedades en la ventana **Propiedades**, etc. Normalmente, no debería modificar manualmente este archivo.

Nota

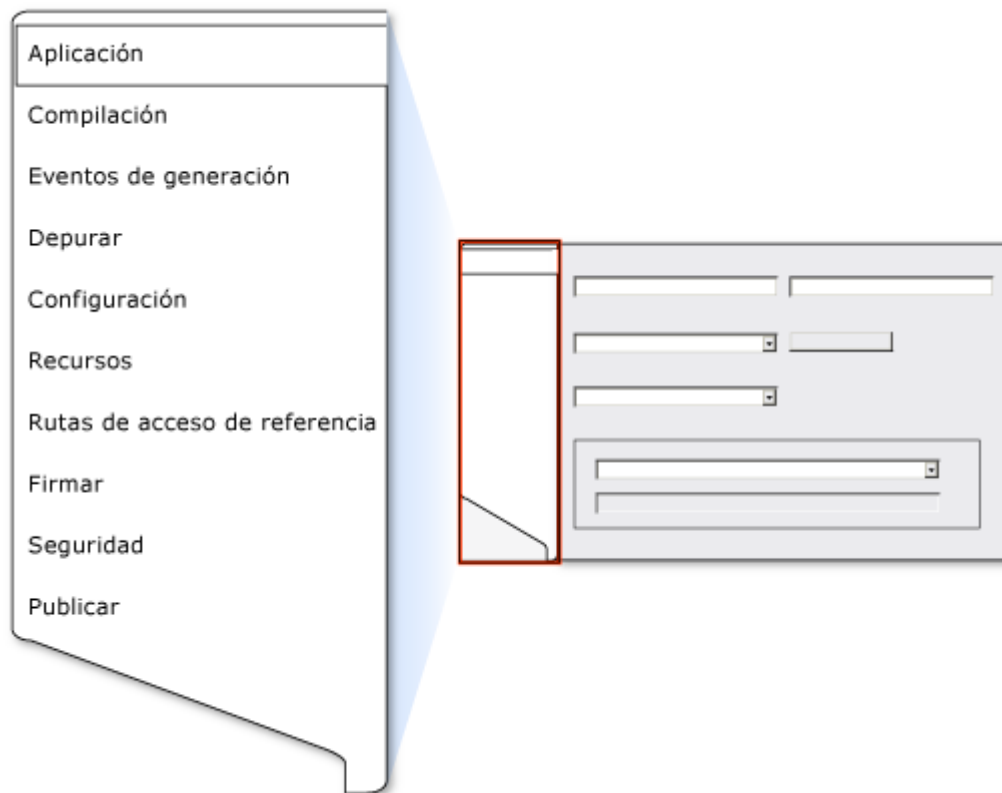
Naturalmente, si crea un proyecto de aplicación de consola, no contendrá archivos de código fuente para formularios Windows Forms.

Otro archivos de código fuente

Un proyecto puede incluir cualquier número de archivos .cs adicionales asociados o no a un formulario Windows Forms concreto. En la ilustración anterior del **Explorador de soluciones**, program.cs contiene el punto de entrada para la aplicación. Un solo archivo .cs puede contener cualquier número de definiciones de clases y estructuras. Puede agregar archivos o clases nuevos o existentes al proyecto haciendo clic en **Agregar nuevo elemento** o **Agregar elemento existente** en el menú **Proyecto**.

Modificar las propiedades de un proyecto (Visual C#)

Después de crear un proyecto, puede utilizar el **Diseñador de proyectos** para realizar tareas tales como cambiar el nombre del archivo ejecutable, personalizar el proceso de generación, agregar una referencia a un archivo DLL o reforzar la configuración de seguridad. Puede obtener acceso al **Diseñador de proyectos** en el Menú **Proyecto** haciendo clic en **Propiedades** o haciendo clic con el botón secundario en el elemento **Propiedades** en el **Explorador de soluciones**. El **Diseñador de proyectos** aparecerá en la ventana del editor/diseñador tal como se muestra en la ilustración siguiente:



Las propiedades del proyecto se agrupan en 10 páginas en el **Diseñador de proyectos**. Las páginas de propiedades del **Diseñador de proyectos** están ubicadas en el mismo panel central que utiliza el **Diseñador de Windows Forms** y el editor de código.

Nota

Visual Studio Team System incluye una página de propiedades adicional para análisis de código.

En la ilustración anterior, se muestra la página de propiedades **Aplicación**. Al hacer clic en las etiquetas en la ficha izquierda (**Generar**, **Generar eventos**, **Depurar**, etc.) puede tener acceso a la página de propiedades correspondiente. La información específica del proyecto que se escribe aquí se almacena en un archivo .csproj que no se puede ver en el **Explorador de soluciones** y que está ubicado en la carpeta del proyecto de la unidad. Para obtener ayuda para cualquiera de las páginas de propiedades mientras trabaja en Visual C#, coloque el cursor del mouse en la página y presione **F1**.

La tabla siguiente ofrece una breve descripción de cada página del **Diseñador de proyectos**:

Página de propiedades	Descripción
Aplicación	Cambiar el nombre del ensamblado, el tipo de proyecto, la información del ensamblado, incluido el número de versión, y otras opciones de recursos.
Generar	Cambiar la ubicación donde se almacena el ensamblado compilado, las opciones de compilación condicional, la forma en que se controlan los errores y advertencias, y otros valores de configuración
Eventos de generación	Crear y modificar pasos de generación personalizada
Depuración	Especificar los argumentos de la línea de comandos cuando se ejecuta en el depurador y otros valores de configuración.
Recursos	Agregar al proyecto cadenas, iconos, imágenes u otros tipos de archivos como recursos.
Configuración	Almacenar valores de configuración tales como las cadenas de conexión para una base de datos o la combinación de colores que desea un usuario concreto. Estas configuraciones se pueden recuperar dinámicamente en tiempo de ejecución.
Rutas de acceso de referencia	Especificar la ruta de acceso en la que están ubicados los ensamblados a los que hace referencia el proyecto
Firma	Especificar las opciones de certificado de ClickOnce y proporcionar un nombre seguro para su ensamblado.
Seguridad	Especificar la configuración de seguridad que necesita la aplicación para funcionar
Publicación	Especificar las opciones para distribuir su aplicación a un sitio Web, servidor ftp o ubicación de archivo.
Análisis de código (sólo Visual Studio Team System)	Opciones para las herramientas que analizan el código fuente para detectar problemas potenciales de seguridad, comprobar el cumplimiento de las instrucciones de diseño de .NET Framework, etc.

Diseñar una interfaz de usuario (Visual C#)

En Visual C#, la forma más rápida y cómoda de crear la interfaz del usuario (UI) es hacerlo visualmente, con el **Diseñador de Windows Forms** y el **Cuadro de herramientas**. Hay tres pasos básicos para crear todas las interfaces de usuario:

- Agregar los controles a la superficie de diseño.
- Establecer las propiedades iniciales de los controles.
- Escribir los controladores para los eventos especificados.

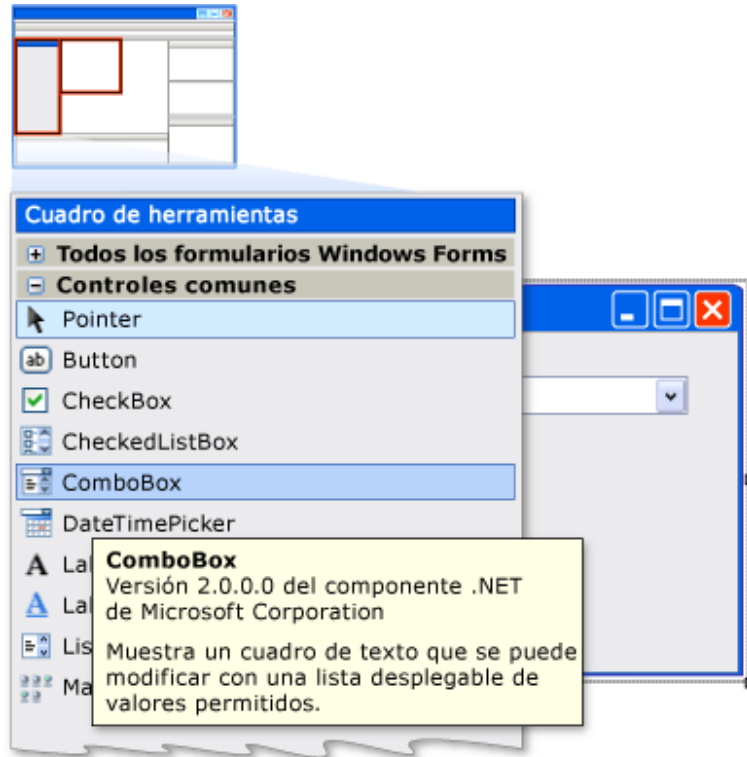
Aunque también puede escribir su propio código para crear la UI, los diseñadores permiten hacer este trabajo mucho más rápidamente de lo que es posible mediante codificación manual.

Nota

También puede utilizar Visual C# para crear aplicaciones de consola que tengan una interfaz de usuario basada en texto simple.

Agregar controles

En el diseñador, se utiliza el mouse para arrastrar controles, como botones y cuadros de texto, hasta una superficie de diseño que representa el formulario. La siguiente ilustración muestra un cuadro combinado que se ha arrastrado desde la ventana **Cuadro de herramientas** a un formulario en el **Diseñador de Windows Forms**.



A medida que se trabaja visualmente, el diseñador traduce las acciones en código fuente de C# y las escribe en un archivo de proyecto llamado <nombre>.designer.cs, donde <nombre> es el nombre asignado al formulario. Cuando se ejecuta la aplicación, el código fuente ajusta la posición y el tamaño de los elementos de la interfaz de usuario para que aparezcan como en la superficie de diseño.

Establecer propiedades

Después de agregar un control al formulario, puede utilizar la ventana **Propiedades** para establecer sus propiedades, como el color de fondo y el texto predeterminado. Los valores que especifique en la ventana **Propiedades** sólo son los valores iniciales que se asignarán a la propiedad cuando se cree el control en tiempo de ejecución. En muchos casos, se puede tener acceso a estos valores o modificarlos mediante programación en tiempo de ejecución. Para ello, basta con obtener o establecer la propiedad en la instancia de la clase del control en la aplicación. La ventana **Propiedades** resulta útil en tiempo de diseño porque permite examinar todas las propiedades, eventos y métodos que admite un control.

Controlar eventos

Los programas con interfaces de usuario gráficas son principalmente controlados por eventos. Esperan hasta que un usuario haga algo, como escribir texto en un cuadro de texto, hacer clic en un botón o cambiar la selección de una lista. Cuando esto sucede, el control, que es simplemente una instancia de una clase de .NET Framework, envía un evento a la aplicación. Tiene la opción de controlar un evento escribiendo un método especial en la aplicación al que se llamará cuando se reciba el evento.

Puede utilizar la ventana **Propiedades** para especificar los eventos que desea controlar en su código; seleccione un control en el diseñador y haga clic en el botón **Eventos**, con el icono de rayo, en la barra de herramientas de la ventana **Propiedades** para ver sus eventos. En el diagrama siguiente se muestra el botón de eventos.



Cuando agregue un controlador de eventos a través de la ventana **Propiedades**, el diseñador escribirá automáticamente el cuerpo del método vacío, y deberá escribir el código para que el método lleve a cabo la acción deseada. La mayoría de los controles generan un gran número de eventos, pero en la mayoría de los casos, las aplicaciones sólo necesitan controlar unos pocos, si no sólo uno. Por ejemplo, puede que necesite controlar el evento **Click** de un botón, pero no su evento **Paint** a menos que desee personalizar su aspecto de forma avanzada.

Tutorial: Crear un formulario Windows Forms sencillo

En este tutorial generará y ejecutará un formulario Windows Forms sencillo.

Nota

Los cuadros de diálogo y comandos de menú que verá en Visual Studio pueden variar con respecto a los descritos en la Ayuda en función de su edición o configuración activa. Para cambiar su configuración, elija **Importar y exportar configuraciones** en el menú **Herramientas**.

Para crear un formulario Windows Forms

1. Inicie Visual Studio.
2. Cree una aplicación para Windows denominada **Hola a todos**.
3. En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.
4. Haga clic en el botón para seleccionarlo. En la ventana Propiedades, establezca la propiedad **Text** en **Hola**.

Para escribir el código de la aplicación

1. Haga doble clic en el botón para agregar un controlador de eventos para el evento **Click**. Se abrirá el Editor de código con el punto de inserción situado dentro del controlador de eventos.
2. Inserte el código siguiente:

C#

```
MessageBox.Show("!Hola a Todos!");
```

Para probar la aplicación

1. Presione F5 para ejecutar la aplicación.
2. Cuando se esté ejecutando la aplicación, haga clic en el botón y compruebe que aparece "¡Hola a todos!".
3. Cierre el formulario Windows Forms para volver a Visual Studio.

Editar código (Visual C#)

El Editor de código de Visual C# es un procesador de textos para escribir el código fuente. Al igual que Microsoft Word ofrece un apoyo exhaustivo para frases, párrafos y gramática, el Editor de código de C# lo hace para la sintaxis de C# y para .NET Framework. Este apoyo se puede agrupar en cinco categorías principales:

- IntelliSense: documentación básica continuamente actualizada sobre las clases y los métodos de .NET Framework a medida que se escriben en el editor, así como generación de código automática.
- Refactorización: reestructuración inteligente de la base de código a medida que éste evoluciona durante el transcurso de un proyecto de desarrollo.
- Fragmentos de código: una biblioteca que se puede examinar y que contiene los patrones de código que se repiten con frecuencia.
- Subrayado con líneas onduladas: notificaciones visuales de palabras incorrectas, errores de sintaxis y situaciones de advertencia a medida que se escribe.
- Ayudas de legibilidad: Esquematización y color.

IntelliSense

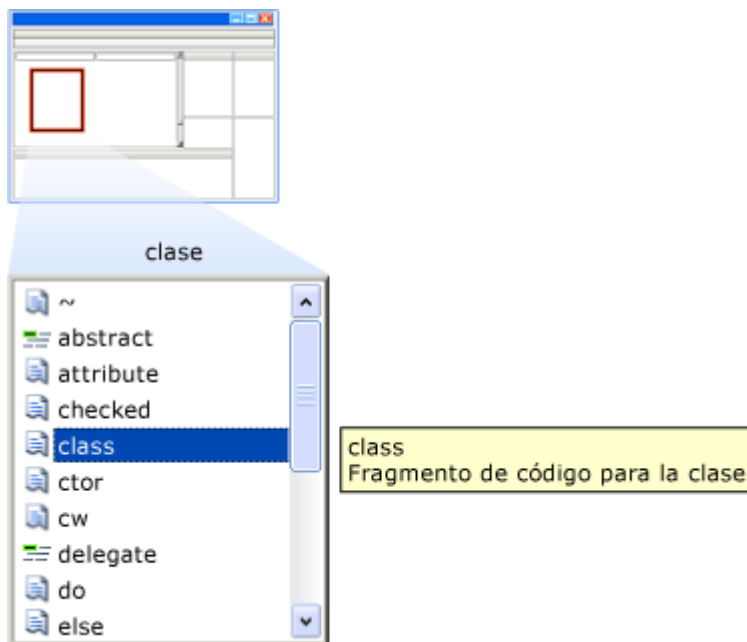
IntelliSense es el nombre de un conjunto de funciones relacionadas que han sido diseñadas para minimizar el tiempo necesario para buscar ayuda y para ayudarlo a escribir código de manera más precisa y eficiente. Todas ellas proporcionan información básica sobre las palabras claves del lenguaje, los tipos de .NET Framework y las firmas de métodos a medida que se escriben en el editor. La información se muestra mediante información sobre herramientas, cuadros de lista y etiquetas inteligentes.

Nota

Muchas de las funciones de IntelliSense se comparten con otros lenguajes de Visual Studio y se documentan mediante ilustraciones en el nodo Ayudas de codificación de MSDN library. Las secciones siguientes proporcionan una breve descripción de IntelliSense e incluyen vínculos a documentación más completa.

Listas de finalización

A medida que se escribe código fuente en el editor, IntelliSense muestra un cuadro de lista que contiene todas las palabras clave de C# y las clases de .NET Framework. Si encuentra una coincidencia en el cuadro de lista para el nombre que se está escribiendo, selecciona el elemento. Si el elemento seleccionado es el deseado, se puede presionar la tecla TAB para que IntelliSense termine de escribir el nombre o la palabra clave.



Información rápida

Cuando el cursor se sitúa sobre un tipo de .NET Framework, IntelliSense muestra información rápida sobre herramientas que contiene documentación básica sobre el tipo.

Lista de miembros

Cuando se introduce un tipo de .NET Framework en el Editor de código y después se escribe el operador punto (.), IntelliSense muestra un cuadro de lista que contiene los miembros del tipo. Cuando se realiza una selección y se presiona TAB, IntelliSense escribe el nombre del miembro.

Información de parámetros

Cuando se escribe un nombre de método en el Editor de código seguido de un paréntesis de apertura, IntelliSense muestra información rápida sobre herramientas de parámetros que indica el orden y los tipos de los parámetros del método. Si el método está sobrecargado, puede el método, se puede desplazarse hacia abajo a través de todas las firmas sobrecargadas

```
Console.Write( |
```

▲ 1 de 18 ▼ void Console.Write (bool value)
value: Valor que se va a escribir.

Agregar using

En ocasiones, puede intentar crear una instancia de una clase de .NET Framework sin un nombre suficientemente completo. En este caso, IntelliSense muestra una etiqueta inteligente después del identificador sin resolver. Al hacer clic en la etiqueta inteligente, IntelliSense muestra una lista de directivas **using** que permitirán resolver el identificador. Cuando seleccione una directiva de la lista, IntelliSense la agregará al principio del archivo de código fuente y podrá seguir escribiendo código en la posición actual.

Refactorización

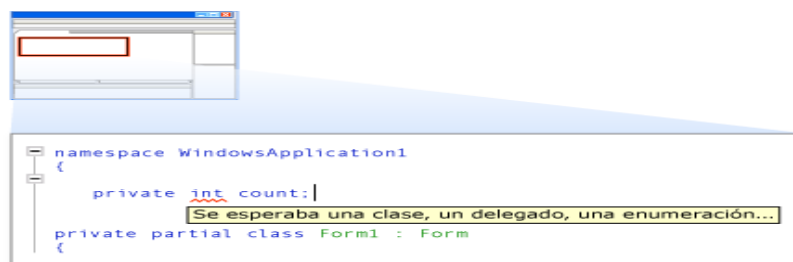
A medida que una base de código crece y evoluciona en el transcurso de un proyecto de desarrollo, puede ser deseable hacer cambios para que sea más legible para los usuarios o más portable. Por ejemplo, puede desear dividir algunos métodos en métodos de menor tamaño, cambiar los parámetros de un método o cambia el nombre de los identificadores. La función de refactorización, a la que se tiene acceso mediante un clic con el botón secundario del mouse en el Editor de código, lleva a cabo estas tareas de forma mucho más cómoda, inteligente y rigurosa que las herramientas tradicionales, como buscar y reemplazar.

Fragmentos de código

Los fragmentos de código son unidades pequeñas de código fuente de C# de uso frecuente que se puede escribir con precisión y rapidez mediante un par de pulsaciones de tecla. Para obtener acceso al menú de fragmentos de código, es preciso hacer clic con el botón secundario del mouse en el Editor de código. Puede examinar los numerosos fragmentos de código que se incluyen en Visual C# y crear los suyos propios.

Subrayado con líneas onduladas

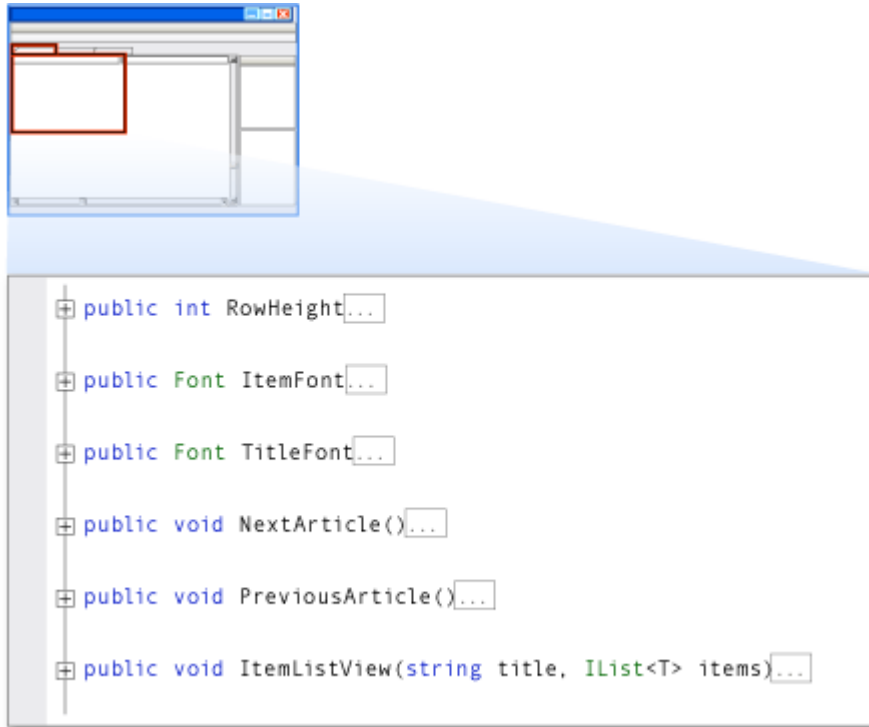
El subrayado con líneas onduladas ofrece información instantánea sobre los errores del código a medida que se escribe. Un subrayado con líneas onduladas rojas identifica un error de sintaxis, como la falta de un punto y coma o que una llave de cierre no tiene su correspondiente llave de apertura. Un subrayado con líneas onduladas verdes identifica una advertencia potencial del compilador, y con líneas azules, identifica un problema de Editar y continuar. La ilustración siguiente muestra un subrayado con líneas onduladas rojas:



Ayudas de legibilidad

Esquematzación

El Editor de código trata automáticamente los espacios de nombres, las clases y los métodos como regiones que se pueden contraer para facilitar la localización y la lectura de otras partes del archivo de código fuente. También puede crear sus propias regiones contraíbles delimitando el código con las directivas **#region** y **#endregion**.



Colorización

El editor asigna distintos colores a las distintas categorías de identificadores de un archivo de código fuente de C#.

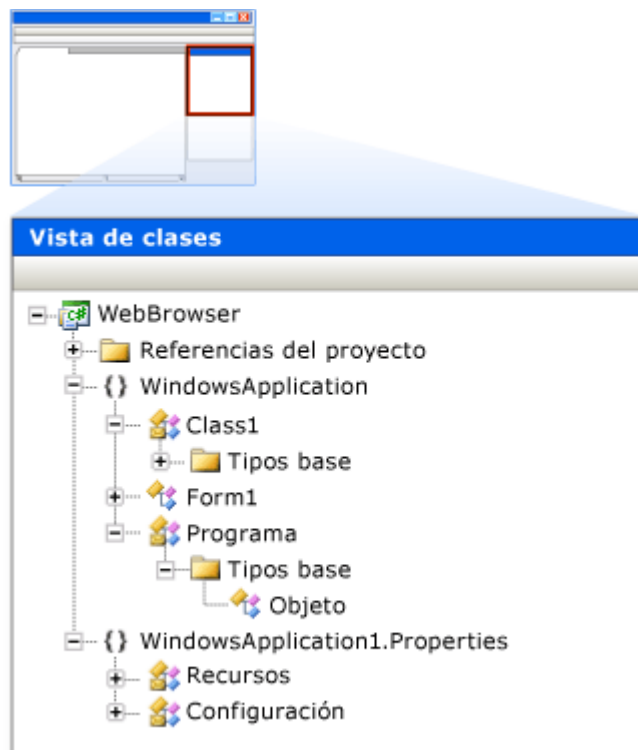
Desplazamientos y búsquedas (Visual C#)

Visual C# proporciona las herramientas siguientes para ayudarle a desplazarse y buscar en el código fuente, los archivos de proyecto y las ventanas abiertas.

- Vista de clases
- Barras de exploración
- Exploración CTRL-TAB
- Buscar en archivos

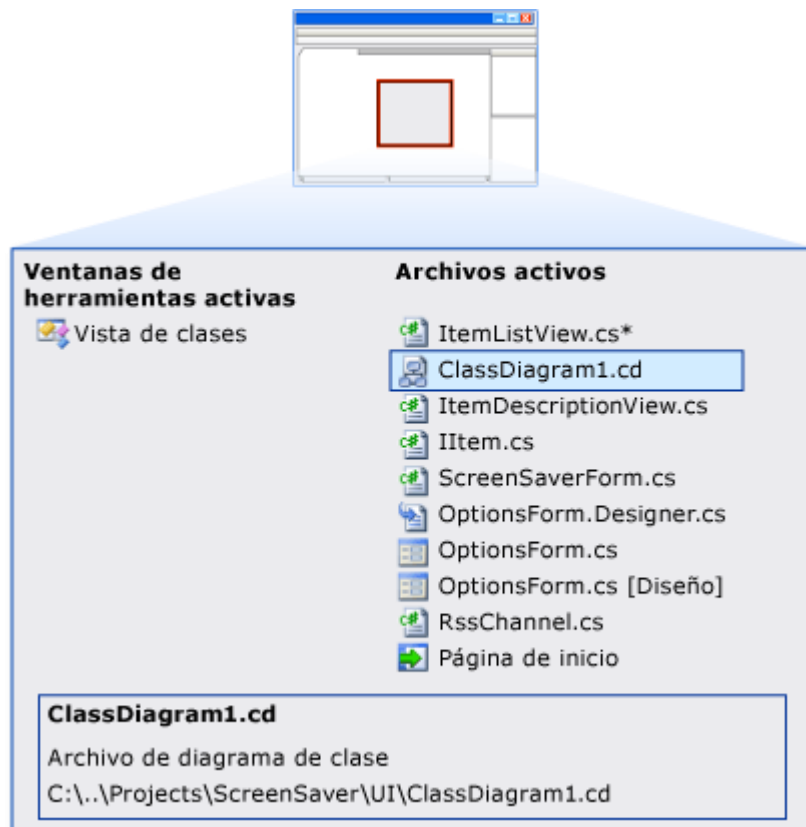
Vista de clases

La ventana **Vista de clases** proporciona una vista del proyecto basada en clases en lugar de archivos, como en el **Explorador de soluciones**. Puede utilizar la **Vista de clases** para desplazarse rápidamente a cualquier clase o miembro de clase del proyecto. Para obtener acceso a la **Vista de clases**, haga clic en **Vista de clases** en el menú **Ver**.



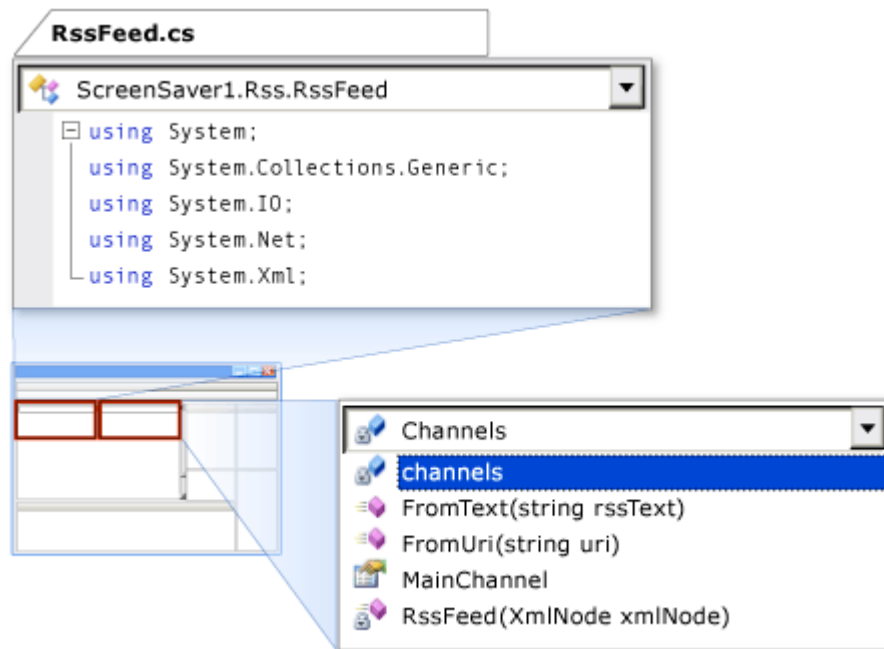
Exploración CTRL-TAB

En cualquier momento, puede tener varias ventanas activas en un proyecto de Visual C#. Para desplazarse rápidamente a una ventana, presione CTRL+TAB para abrir una ventana que muestra todas las ventanas activas de herramientas y código fuente. Mueva las teclas de dirección mientras mantiene presionada la tecla CTRL para seleccionar la ventana que desea mostrar.



Barras de exploración

La parte superior de cada ventana del editor de código contiene la barra de exploración, que se compone de dos cuadros de lista. El de la izquierda contiene todas las clases definidas en el archivo actual. El de la derecha, contiene todos los miembros de la clase seleccionada en el cuadro de lista de la izquierda. Puede ir directamente a un método seleccionándolo en el cuadro de lista derecho.



Buscar en archivos

Si presiona CTRL+MAYÚS+F, se abrirá el cuadro de diálogo **Buscar en archivos**, que permite realizar operaciones de buscar y reemplazar en la totalidad de un proyecto.

Nota

Para cambiar el nombre de métodos o tipos, o para cambiar parámetros de método, utilice la función Refactorización, que es más profunda e inteligente que Buscar y reemplazar.

Generar y depurar (Visual C#)

En Visual C#, una aplicación ejecutable se genera haciendo clic en **Generar** en el menú **Generar** (o presionando CTRL+MAYÚS+B). Puede generar e iniciar la aplicación de un solo paso presionando F5 o haciendo clic en **Ejecutar** en el menú **Depurar**.

La generación requiere la entrada de los archivos del proyecto en el compilador de C#, que convierte el código fuente en lenguaje intermedio de Microsoft (MSIL) y después une el MSIL a los metadatos, recursos, manifiestos y otros módulos, si los hubiere, para crear un ensamblado. Un ensamblado es un archivo ejecutable que normalmente tiene una extensión .exe o .dll. A medida que desarrolle la aplicación, puede que desee generar una versión de depuración para probarla y ver cómo funciona. Por último, cuando todo sea correcto, creará una versión de lanzamiento que distribuirá a los clientes.

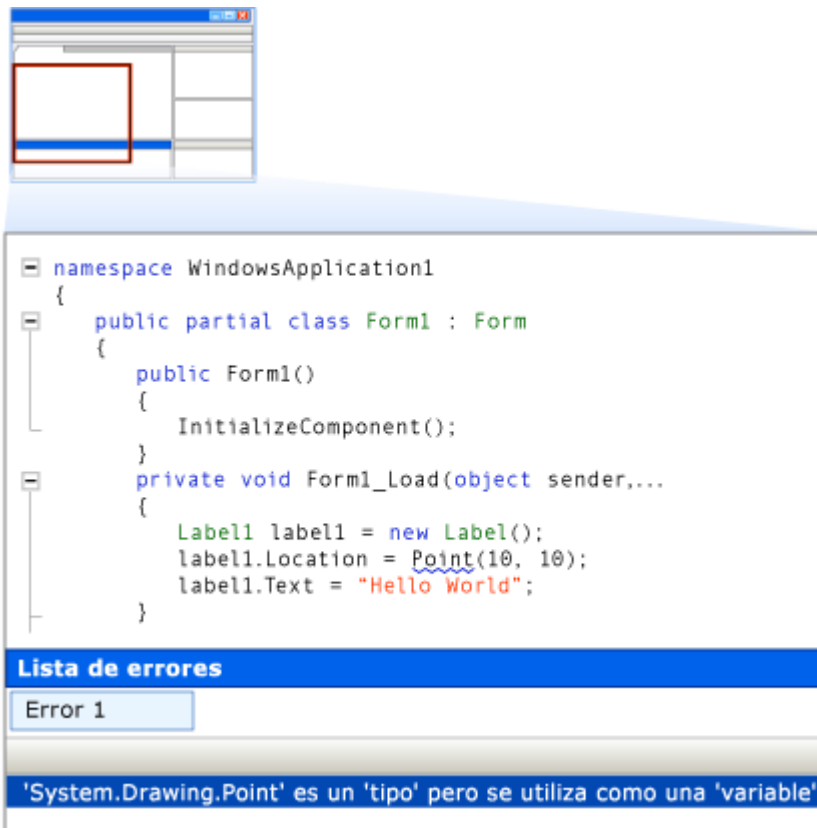
Configuración de generación

Para especificar los distintos valores de configuración de generación, haga clic con el botón secundario en el elemento de proyecto en el **Explorador de soluciones** y, a continuación, seleccione el panel **Generar** en el **Diseñador de proyectos**.

Visual Studio utiliza la herramienta MSBuild para crear ensamblados. MSBuild también se puede ejecutar desde la línea de comandos y se puede personalizar de muchas maneras

Errores al generar

Si hay errores de sintaxis de C#, o si no es posible resolver identificadores en un tipo o miembro conocido, la generación no finalizará correctamente y verá una lista de errores en la Lista de errores (Ventana), que aparece de forma predeterminada justo debajo del editor de código. Puede hacer doble clic en un mensaje de error para ir a la línea de código en la que se produjo el error.



Los mensajes de error del compilador de C# suelen ser muy claros y descriptivos, pero si no consigue deducir el problema, puede ir a la página de Ayuda del mensaje presionando F1 con el mensaje de error seleccionado en la lista de errores. La página de Ayuda contiene información útil adicional. Si todavía no puede resolver el problema, el paso siguiente es plantear su pregunta en uno de los foros o grupos de noticias de C#. Para obtener acceso a los foros, haga clic en **Formular una pregunta** en el menú **Comunidad**.

Nota

Si la página de Ayuda del error del compilador no es de utilidad para resolver el error, puede enviar una descripción del problema a Microsoft para ayudar a mejorar la documentación. Para enviar el correo electrónico, haga clic en el vínculo situado en la parte inferior de la página de Ayuda que contiene el error.

Configuraciones de lanzamiento y de depuración

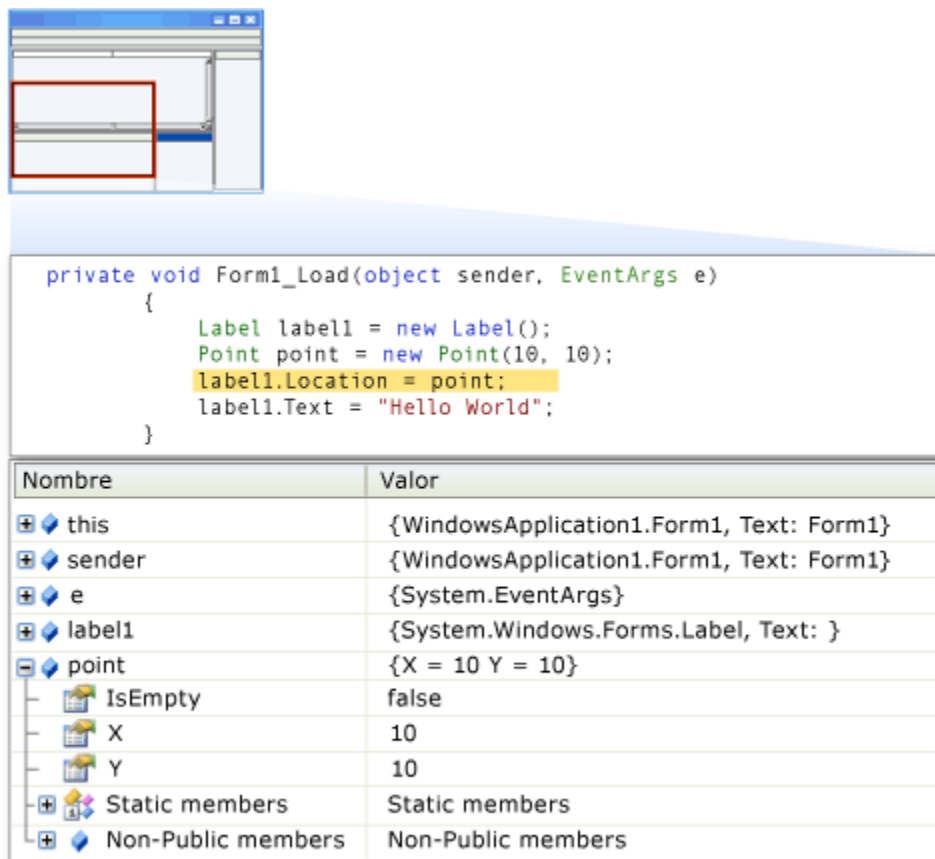
Mientras siga trabajando activamente en el proyecto, es muy probable que desee generar la aplicación con la configuración de depuración, que le permite ver el valor de las variables y controlar la ejecución en el depurador. También puede crear y probar generaciones en la configuración de lanzamiento para comprobar que no se ha introducido ningún error que

sólo se manifieste en uno u otro tipo de generación. En programación de .NET Framework, estos errores son muy raros, pero pueden producirse.

Cuando esté preparado para distribuir la aplicación a los usuarios finales, cree una generación de lanzamiento, que tendrá un tamaño muy inferior y un rendimiento muy superior al de la configuración de depuración correspondiente. Puede establecer la configuración de generación en el panel **Generar** del **Diseñador de proyectos** o en la barra de herramientas **Generar**.

Depuración

Cuando trabaje con el editor de código, puede establecer en cualquier momento un punto de interrupción en una línea de código presionando **F9**. Cuando presione **F5** para ejecutar la aplicación en el depurador de Visual Studio, la aplicación se detendrá en la línea y podrá examinar el valor de las variables, observar cómo o cuándo la ejecución sale de un bucle, recorrer el código línea a línea mediante la tecla **F10** o establecer puntos de interrupción adicionales.



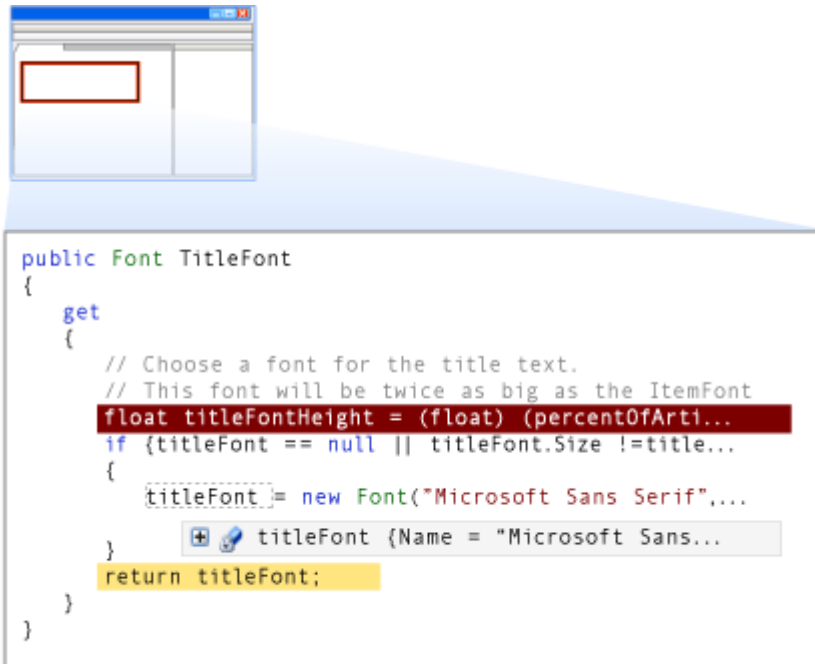
The image shows a Visual Studio window with a code editor and the Watch window. The code editor displays the `Form1_Load` method, and the Watch window shows the values of the variables in scope.

```
private void Form1_Load(object sender, EventArgs e)
{
    Label label1 = new Label();
    Point point = new Point(10, 10);
    label1.Location = point;
    label1.Text = "Hello World";
}
```

Nombre	Valor
this	{WindowsApplication1.Form1, Text: Form1}
sender	{WindowsApplication1.Form1, Text: Form1}
e	{System.EventArgs}
label1	{System.Windows.Forms.Label, Text: }
point	{X = 10 Y = 10}
IsEmpty	false
X	10
Y	10
Static members	Static members
Non-Public members	Non-Public members

También puede establecer puntos de interrupción condicionales, que sólo detendrán la ejecución si se cumple una condición determinada. Los puntos de seguimiento son similares a los puntos de interrupción, pero en lugar de detener la interrupción, se limitan a escribir el valor de una variable determinada en la ventana de resultados.

Cuando la ejecución se detiene en un punto de interrupción, puede situar el cursor sobre cualquier variable del ámbito para ver información sobre ella. La ilustración siguiente muestra una sugerencia de datos en el depurador:



Puede recorrer el código línea a línea presionando F10 después de que el depurador se haya detenido en un punto de interrupción. Incluso puede corregir ciertos tipos de errores en el código y proseguir con la depuración sin necesidad de detener y volver a compilar la aplicación.

El depurador de Visual Studio es una eficaz herramienta, y merece la pena invertir tiempo en leer la documentación para comprender distintos conceptos como Editar y continuar, Ver datos en el depurador, Visualizadores y Depuración Just-In-Time.

Para practicar la depuración de una aplicación

1. Inicie Visual C# Express.
2. En el menú Archivo, haga clic en **Nuevoproyecto**.

Aparecerá el cuadro de diálogo Nuevo proyecto.

3. En el cuadro de diálogo Nuevo proyecto, haga clic en Aplicación de Windows Forms
4. y luego en Aceptar.

Se abre un nuevo proyecto de formularios Windows Forms que muestra un formulario nuevo en el Diseñador de Windows Forms.

5. Desde el Cuadro de herramientas, arrastre un control TextBox al formulario.
6. Arrastre un control **Button** desde el Cuadro de herramientas al formulario y colóquelo junto a **TextBox**.
7. Haga doble clic en el botón para crear el controlador de eventos **Click** predeterminado y mostrar el editor de código.
8. Agregue el código siguiente al controlador de eventos **button1_Click**.

```
textBox1.Text = "Button was clicked!";
```

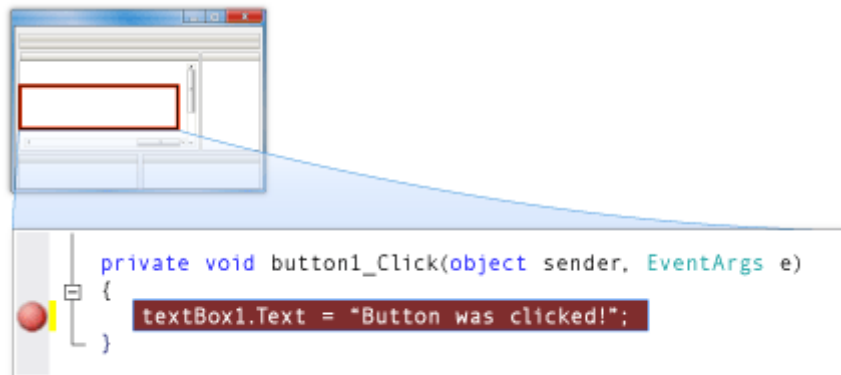
9. En el menú Generar, haga clic en Generar solución.

El proyecto se genera sin errores.

10. En el editor de código, haga clic en el margen izquierdo en la misma línea que el texto que agregó.

Aparece un punto rojo en el margen y se resalta la línea de código. Esto se conoce como adición de un punto de interrupción. El punto de interrupción detendrá temporalmente la ejecución del programa justo antes de ejecutarse esa línea de código. La ilustración siguiente muestra un punto de interrupción en el IDE.

Establecer un punto de interrupción

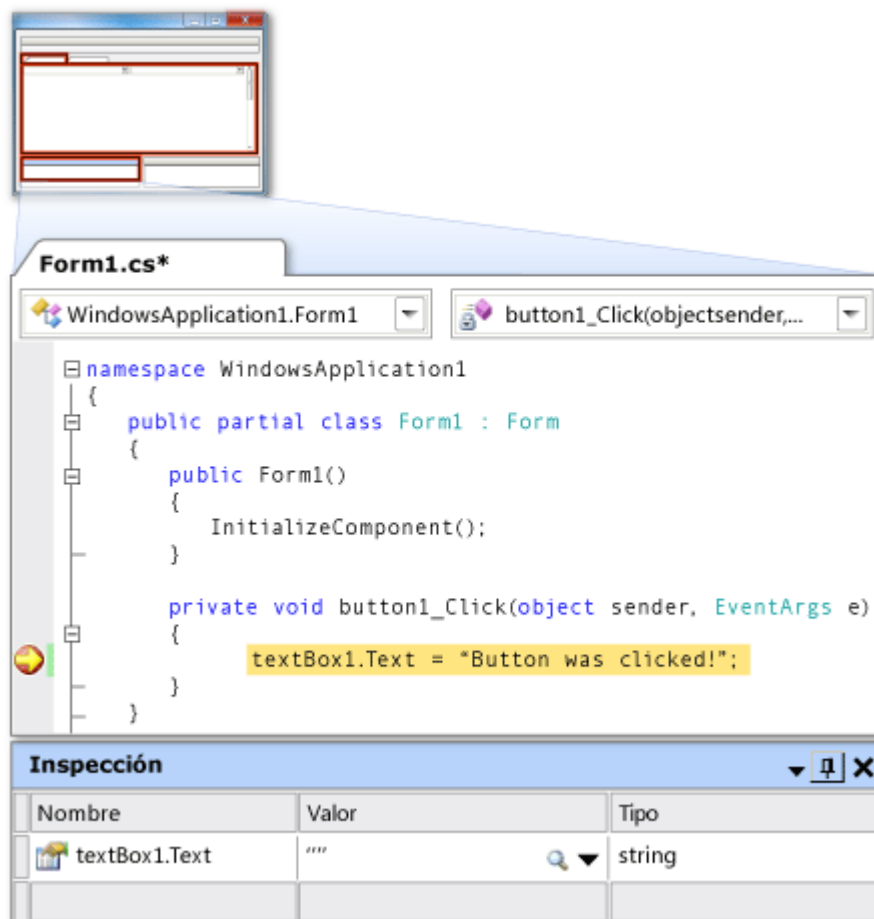


11. En el menú Depurar, haga clic en Iniciar depuración.

El formulario Windows Forms comienza a ejecutarse.

12. Haga clic en el botón y confirme que la ejecución del código se ha detenido en la línea de código donde agregó el punto de interrupción y que el código aparece resaltado en amarillo.
13. En el menú Depurar, elija Ventanas y, a continuación, haga clic en Inspección.
14. En la ventana Inspección, haga clic en la primera fila bajo el encabezado Nombre, escriba `textBox1.Text` y, a continuación, presione ENTRAR.
La ventana Inspección muestra el valor de esta variable entre comillas, como se muestra en la ilustración siguiente:

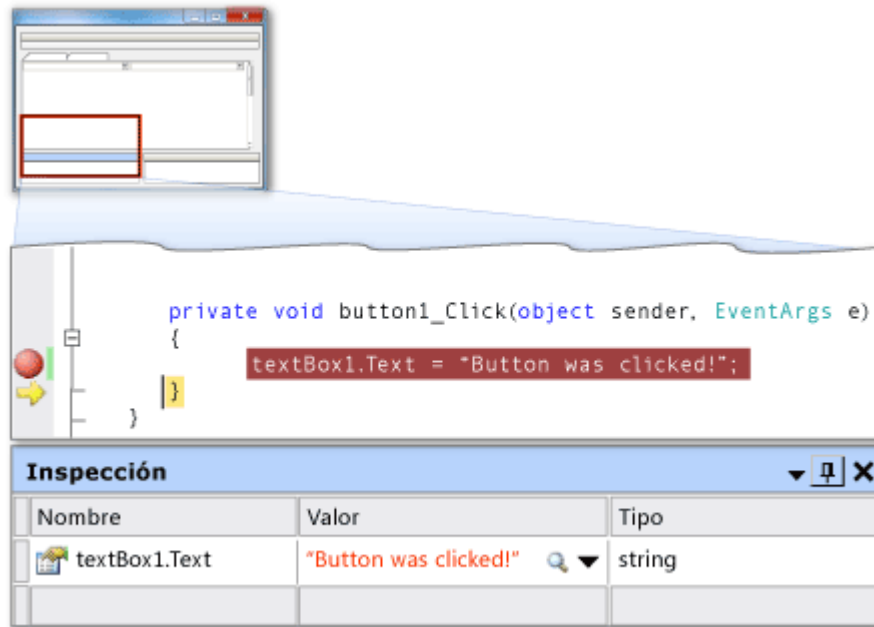
Ventana Inspección en el punto de interrupción



15. En el menú Depurar, haga clic en Paso a paso por instrucciones.

El valor de `textBox1.Text` en la ventana Inspección cambia a `"Button was clicked!"`, como se muestra en la ilustración siguiente:

Ventana Inspección



Nota:

Paso a paso por instrucciones permite desplazarse por el código línea a línea. El comando se llama Paso a paso por instrucciones porque si la siguiente instrucción es una llamada a un método, el depurador ejecuta la primera línea del método llamado en lugar de la siguiente línea del método actual.

16. En el menú Depurar, haga clic en Continuar.

La aplicación se ejecuta y muestra el texto en el cuadro de texto. Hacer clic en Continuar durante una sesión de depuración equivale a hacer clic en Iniciar para empezar una sesión de depuración: el código se ejecuta ininterrumpidamente hasta llegar a un punto de interrupción.

17. La aplicación debería detener la ejecución por sí misma. Si no lo hace, en el menú Depurar, haga clic en Interrumpir todos o presione **Ctrl+Alt+Inter** y, a continuación, haga clic en Detener depuración.

Agregar y editar recursos (Visual C#)

Generalmente, las aplicaciones de Visual C# incluyen datos que no son código fuente. Tales datos se denominan *recursos de proyectos* y pueden ser datos binarios, archivos de texto, archivos de audio o vídeo, tablas de cadenas, iconos, imágenes, archivos XML u otro tipo de datos necesarios para la aplicación. Los datos de recursos del proyecto se almacenan en formato XML en el archivo .resx (denominado Resources.resx de forma predeterminada), que se puede abrir en el **Explorador de soluciones**.

Agregar recursos a proyectos

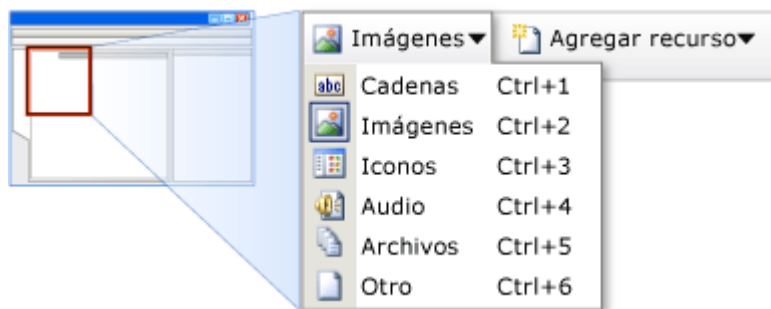
Puede agregar recursos a un proyecto haciendo clic con el botón secundario del mouse en el nodo **Propiedades** de su proyecto en **Explorador de soluciones**, haciendo clic en **Abrir** y luego en el botón **Agregar recurso** en la página **Recursos** de **Diseñador de proyectos**.

Puede agregar recursos a su proyecto en forma de recursos vinculados, que son archivos externos, o de recursos incrustados, que se incrustan directamente en el archivo .resx.

- Cuando se agrega un recurso vinculado, el archivo .resx que almacena la información de recursos del proyecto sólo incluye una ruta de acceso relativa al archivo de recursos en el disco. Si agrega imágenes, vídeos u otros archivos complejos como recursos vinculados, puede modificarlos mediante un editor predeterminado que asocie a ese tipo de archivo en el Diseñador de recursos.
- Cuando agrega un recurso incrustado, los datos se almacenan directamente en el archivo de recursos del proyecto (.resx). Las cadenas sólo se pueden almacenar como recursos incrustados.

Editar recursos

El Diseñador de recursos permite a agregar y modificar recursos de proyectos durante el desarrollo mediante la asociación de una aplicación predeterminada para modificar cada recurso. Para obtener acceso al Diseñador de recursos, haga clic con el botón secundario del mouse en **Propiedades** en el **Explorador de soluciones**, haga clic en **Abrir** y, después, en la ficha **Recursos** del Diseñador de proyectos. La ilustración siguiente muestra las opciones del menú Diseñador de recursos:



Para editar los recursos incrustados, debe trabajar directamente en el archivo .resx para manipular los caracteres o bytes individuales. Por eso es más conveniente almacenar los tipos de

archivo complejos como recursos vinculados durante el desarrollo. Puede utilizar el Editor binario para modificar archivos de recursos, incluido el archivo .resx, en el nivel binario en formato ASCII o hexadecimal. Puede utilizar el Editor de imágenes para modificar iconos y cursores así como archivos .jpeg y GIF almacenados como recursos vinculados. También puede elegir otras aplicaciones como editores para estos tipos de archivo.

Compilar recursos en ensamblados

Cuando genera su aplicación, Visual Studio invoca la herramienta resgen.exe para convertir los recursos de la aplicación en una clase interna llamada Resources. Esta clase está contenida en el archivo Resources.Designer.cs, que se anida en el archivo Resources.resx en el **Explorador de soluciones**. La clase Resources encapsula todos los recursos del proyecto en propiedades **get** estáticas de sólo lectura para proporcionar recursos con establecimiento inflexible de tipos en tiempo de ejecución. Cuando genera la aplicación a través del IDE de Visual C#, todos los datos de recursos encapsulados, incluidos los archivos vinculados y los recursos que se incrustaron en el archivo .resx, se compilan directamente en el ensamblado de la aplicación (el archivo .exe o .dll). Es decir, el IDE de Visual C# siempre utiliza la opción del compilador /resource. Si genera la aplicación desde la línea de comandos, puede especificar la opción del compilador /linkresource, que le permitirá implementar recursos en un archivo independiente del ensamblado principal de la aplicación. Éste es un escenario avanzado y sólo es necesario en ciertas situaciones raras. Un escenario más común para implementar los recursos independientemente del ensamblado principal de la aplicación es utilizar ensamblados satélite como se describe a continuación.

Obtener acceso a los recursos en tiempo de ejecución

Para obtener acceso a un recurso en tiempo de ejecución, haga referencia a él como lo haría con cualquier otro miembro de una clase. El ejemplo siguiente muestra cómo recuperar un recurso de mapa de bits denominado Image01. Tenga en cuenta que la clase Resources está en un espacio de nombres denominado <projectName>.Properties, por lo que debe usar el nombre completo de cada recurso o agregar la directiva using correspondiente en el archivo de código fuente desde el que obtiene acceso a la clase Resources.

```
System.Drawing.Bitmap bitmap1 = myProject.Properties.Resources.Image01;
```

Internamente, la propiedad get utiliza la clase ResourceManager para crear una nueva instancia del objeto.

Recursos en ensamblados satélite

Si crea aplicaciones que se localizarán (traducirán) en varios idiomas, puede almacenar cada conjunto de cadenas de referencia cultural como un recurso en su propio ensamblado satélite. Cuando distribuya la aplicación, incluya el ensamblado de la aplicación principal junto con los ensamblados satélite necesarios. A continuación, puede agregar ensamblados satélite adicionales o modificar los existentes sin volver a compilar el ensamblado de la aplicación principal.

Obtener Ayuda (Visual C#)

Los archivos de Ayuda incluidos con Visual C# Express son un subconjunto de MSDN Library para Visual Studio Express Editions. Los archivos se pueden instalar en un equipo local desde el CD de Visual C# Express o se pueden descargar desde Internet. MSDN Library para Visual Studio Express Editions es un subconjunto de la biblioteca de MSDN completa. Puede ver la documentación de MSDN, tanto de forma local (Express) como de forma total en línea, utilizando el explorador de Ayuda denominado Microsoft Document Explorer. Sin embargo, si ve el contenido de MSDN en línea, debe tener en cuenta que algunos de los temas podrían no ser aplicables a Visual C# Express.

Nota:

Si instala tanto MSDN Library para Visual Studio Express Editions como MSDN Library completa, puede ver contenido duplicado en el Tabla de Contenido. Si desea instalar MSDN Library, debería desinstalar primero MSDN Library para Visual Studio Express Editions.

Hay cinco maneras de obtener acceso a la Ayuda:

- Búsqueda con F1
- Buscar
- Índice
- Tabla de contenido
- Cómo

Ayuda en línea y local

Para encontrar la página Opciones de la Ayuda, en el menú Herramientas, haga clic en Opciones, expanda Entorno, expanda Ayuda y, a continuación, haga clic en En línea. Puede que tenga que hacer clic en Mostrar todas las configuraciones para ver las opciones. En la página de propiedades Opciones de la Ayuda, puede especificar las opciones siguientes para definir el comportamiento de la búsqueda y de la búsqueda con F1:

- Buscar primero en MSDN Library en línea y, después, en la documentación local.
- Probar primero en MSDN Library local para Visual Studio Express Editions y, después, en la documentación en línea.
- Probar sólo en la biblioteca local MSDN Library para Visual Studio Express Editions.

Estas opciones también aparecerán la primera vez que inicie una búsqueda. La documentación de MSDN en línea puede contener información más reciente que la documentación local. Por tanto, si dispone de una conexión a Internet mientras trabaja en Visual C#, se recomienda elegir la opción de buscar primero en MSDN Library en línea.

Puede instalar la biblioteca completa MSDN Library o MSDN Library para Visual Studio Express descargándolas de [MSDN](#).

Búsqueda con F1

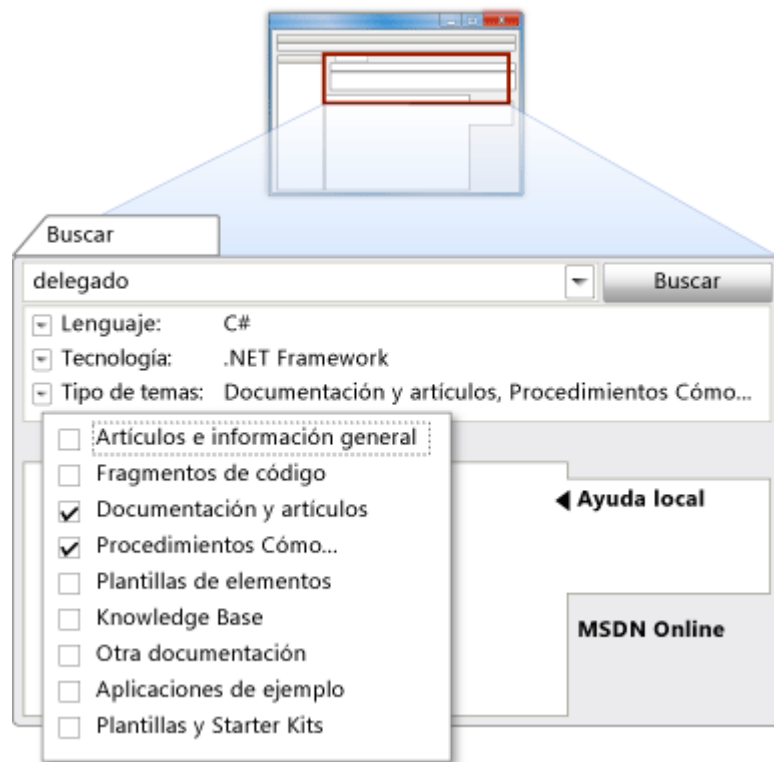
F1 proporciona capacidad de búsqueda contextual. En el editor de código, puede obtener acceso a la documentación de Ayuda de las palabras clave de C# y las clases de .NET Framework colocando el cursor encima o inmediatamente después de la palabra clave o del miembro de clase y presionando F1. Cuando un cuadro de diálogo o cualquier otra ventana tenga el foco, puede presionar F1 para recibir Ayuda para esa ventana.

Una búsqueda con F1 no muestra más que una página. Si no se encuentra ninguna coincidencia, se muestra una página informativa que proporciona algunas sugerencias para solucionar problemas.

Buscar

Utilice la interfaz de Buscar para la devolución de todos los documentos que coincidan con cualquier término o conjunto de términos especificado.

La ilustración siguiente representa la interfaz de Buscar:



También puede utilizar la página Opciones de la Ayuda para especificar si desea buscar en los sitios web de Codezone y también en MSDN Library. Los sitios de Codezone son una iniciativa de los socios de Microsoft y ofrecen información útil sobre C# y .NET Framework. El contenido de Codezone sólo está disponible en línea.

Las mismas opciones de búsqueda en línea y local se aplican a las funciones Buscar y Búsqueda con F1.

En la interfaz de Buscar puede restringir o ampliar la búsqueda especificando los tipos de documentos que desea incluir. Hay tres opciones: Lenguaje, Tecnología y Tipo de contenido. Generalmente, obtendrá los mejores resultados activando sólo las opciones que se apliquen a su escenario de desarrollo.

Índice

El índice constituye un medio rápido para buscar documentos en la biblioteca local MSDN Library para Visual Studio Express Editions. No es una búsqueda de texto completo; se limita a buscar las palabras clave de índice asignadas a cada documento. Por lo general, una búsqueda en el índice es más rápida y relevante que una búsqueda de texto completo. Si existen varios documentos que contienen la palabra clave de índice especificada en el cuadro de búsqueda, aparecerá una ventana en la que se puede elegir entre las opciones disponibles.

De forma predeterminada, la ventana Índice está ubicada en el lado izquierdo de Document Explorer. Puede tener acceso a ella desde el menú Ayuda.

Tabla de contenido

La tabla de contenido de MSDN Express Library muestra todos los temas de la biblioteca en una estructura de vista de árbol jerárquica. Es una herramienta muy útil para examinar la documentación a fin de obtener una idea de lo que contiene la biblioteca y de explorar documentos que podrían no encontrarse a través del índice o de una búsqueda. A menudo, al buscar un documento a través de F1, índice o búsqueda, podría desear conocer su ubicación en la tabla de contenido de modo que pueda consultar otra documentación relacionada. Haga clic en el botón Sincronizar con tabla de contenido de la barra de herramientas de Document Explorer para ver en MSDN Express Library la ubicación del tema que se muestra actualmente.

Cómo

La función Cómo se manifiesta como una vista con filtro de MSDN Express Library. Principalmente, incluye documentos denominados "Cómo..." o Tutoriales que muestran la forma de llevar a cabo una tarea concreta. Puede tener acceso a la Ayuda de la función Cómo desde la barra de herramientas de Document Explorer, el menú Ayuda o la página Inicio.

Implementar aplicaciones de C#

La implementación es el proceso mediante el cual se distribuye una aplicación o componente finalizado para su instalación en otros equipos. Para las aplicaciones de consola o las aplicaciones para clientes inteligentes basadas en formularios Windows Forms, hay dos opciones de implementación: ClickOnce y Windows Installer.

Implementación ClickOnce

La implementación ClickOnce permite publicar aplicaciones para Windows en un servidor Web o en recurso compartido de archivos de red con el fin de simplificar la instalación. En la mayoría de los escenarios, se recomienda la opción de implementación ClickOnce porque permite implementar aplicaciones de actualización automática basadas en Windows, que se pueden instalar y ejecutar con una interacción mínima del usuario.

Para configurar las propiedades para la implementación ClickOnce, puede utilizar el Asistente para publicación (accesible desde el menú **Generar**) o la página **Publicar** del **Diseñador de proyectos**.

Windows Installer

La implementación de Windows Installer permite crear unos paquetes del instalador que se distribuyen entre los usuarios; el usuario ejecuta el archivo y los pasos de instalación mediante un asistente para instalar la aplicación. Esto se hace agregando un proyecto de instalación a su solución; una vez generado, crea un archivo de instalación que se distribuye entre los usuarios; el usuario ejecuta el archivo y los pasos de instalación mediante un asistente para instalar la aplicación.

Características del editor de código de Visual C#

Visual C# proporciona herramientas que ayudan a editar el código y a desplazarse por él como son:

- **Refactorización:** Muestra operaciones de refactorización que ayudan a modificar el código sin cambiar el comportamiento de la aplicación.
- **Fragmentos de código (C#):** Proporciona información general sobre el uso de fragmentos de código en Visual C# para agregar automáticamente construcciones de código comunes a la aplicación.
- **Color del código:** Describe cómo el Editor de código colorea las construcciones de código.
- **Metadatos como origen:** Describe cómo el IDE permite ver los metadatos como código fuente.

Refactorización

La refactorización es el proceso que consiste en mejorar el código una vez escrito cambiando su estructura interna sin modificar su comportamiento externo.

Visual C# proporciona los siguientes comandos de refactorización en el menú Refactorización:

- Extraer método
- Cambiar nombre
- Encapsular campo
- Extraer interfaz
- Promocionar una variable local a parámetro
- Quitar parámetros
- Reordenar parámetros

Refactorización de varios proyectos

Visual Studio admite la refactorización de varios proyectos. Todas las operaciones de refactorización que corrigen referencias entre archivos corrigen dichas referencias entre todos los proyectos del mismo lenguaje. Esto sólo funciona en referencias entre proyectos. Por ejemplo, si tiene una aplicación de consola que haga referencia a una biblioteca de clase, al cambiar el nombre a un tipo de biblioteca de clase (mediante la operación de refactorización **Rename**), también se actualizarán las referencias al tipo de biblioteca de clase en la aplicación de consola.

Obtener vista previa de cambios (Cuadro de diálogo)

Muchas operaciones de refactorización proporcionan la posibilidad de revisar todos los cambios de referencias que realiza una operación de refactorización en el código, antes de aplicar dichos cambios. Para estas operaciones de refactorización, aparecerá una opción **Vista previa de los cambios de referencia** en el cuadro de diálogo de refactorización. Después de seleccionar dicha opción y aceptar la operación de refactorización, aparecerá el Obtener vista previa de cambios (Cuadro de diálogo). Observe que el cuadro de diálogo **Obtener vista previa de cambios** tiene dos vistas. La vista inferior mostrará el código con todas las actualizaciones de referencias debido a la operación de refactorización. Si presiona **Cancelar** en el cuadro de diálogo **Obtener vista previa de cambios**, se detendrá la operación de refactorización y el código no sufrirá ningún cambio.

Refactorización tolerante a errores

La refactorización tolera errores. En otros términos, se puede realizar una refactorización en un proyecto que no se puede generar. Sin embargo, en estos casos el proceso de refactorización podría no actualizar correctamente las referencias ambiguas.

Fragmentos de código (C#)

Visual Studio ofrece una nueva característica denominada fragmentos de código. Los fragmentos de código pueden utilizarse para escribir un alias corto que luego pueda expandirse dentro de un constructor común de programación. Así, el fragmento de código **for** crea un bucle **for** vacío. Algunos fragmentos de código son fragmentos de código con incorporación de entorno, lo que permite seleccionar líneas de código e incluirlas en el proceso del fragmento de código. Por ejemplo, al seleccionar líneas de código y activar después el fragmento de código **for**, se creará un bucle **for** que incluirá dichas líneas en su bloque. De este modo, los fragmentos de código hacen de la escritura de código de programación un proceso más rápido, sencillo y fiable.

Utilizar fragmentos de código

Los fragmentos de código suelen utilizarse en el editor de código; para ello, escriba un nombre corto para el alias (acceso directo al fragmento de código) y presione la tecla TAB. Además, el menú IntelliSense ofrece el comando **Insertar fragmento de código** que proporciona una lista de fragmentos de código disponibles para su inserción en el editor de código. Para activar la lista de fragmentos de código, escriba CTRL+K y, a continuación, X.

Una vez seleccionado un fragmento de código, su texto queda insertado automáticamente en la posición del cursor. En este punto, todos los campos modificables del fragmento de código se resaltan en amarillo y se selecciona automáticamente el primer campo modificable. El campo seleccionado está rodeado por un rectángulo rojo. Por ejemplo, en el fragmento de código **for**, los campos modificables son la variable del inicializador (i de forma predeterminada) y la expresión de longitud (length de forma predeterminada).

Una vez seleccionado un campo, puede escribirse un nuevo valor. Presione la tecla TAB para recorrer de manera cíclica los campos modificables del fragmento de código; utilice MAYÚS+TAB para hacerlo en el orden inverso. Al hacer clic en un campo, se coloca el cursor en él; si hace doble clic, se selecciona. Cuando se resalta un campo aparece su información sobre herramientas, que lo describe.

Sólo se puede editar la primera instancia de un campo; cuando ese campo esté resaltado, las restantes instancias aparecerán con contorno. Cuando se cambia el valor de un campo modificable, éste cambia en todas las posiciones que ocupa dentro del fragmento de código.

Presione ENTRAR o ESC para cancelar la edición de campos y que el editor de código vuelva a su estado normal.

Los colores predeterminados de los campos modificables del fragmento de código se pueden modificar; para ello, cambie el valor del campo **Fragmento de código** en el panel **Fuentes y colores** del cuadro de diálogo **Opciones**.

Crear fragmentos de código

Aparte de los que ya se incluyen con Visual Studio de forma predeterminada, se puede crear y utilizar fragmentos de código personalizados.

Nota

Para los fragmentos de código en C#, los caracteres válidos para especificar el campo [<Acceso directo>](#) son: caracteres alfanuméricos, el signo de sostenido (#), el carácter de tilde (~), el carácter de subrayado (_) y el carácter de guión corto (-).

Color del código

El editor de código analiza los símbolos (token) y las construcciones de código, para que resulten reconocibles y discernibles con facilidad de los demás contenidos del código fuente de dicho editor. Después de analizar el código, el editor de código colorea las construcciones de la forma apropiada.

Símbolos (token)

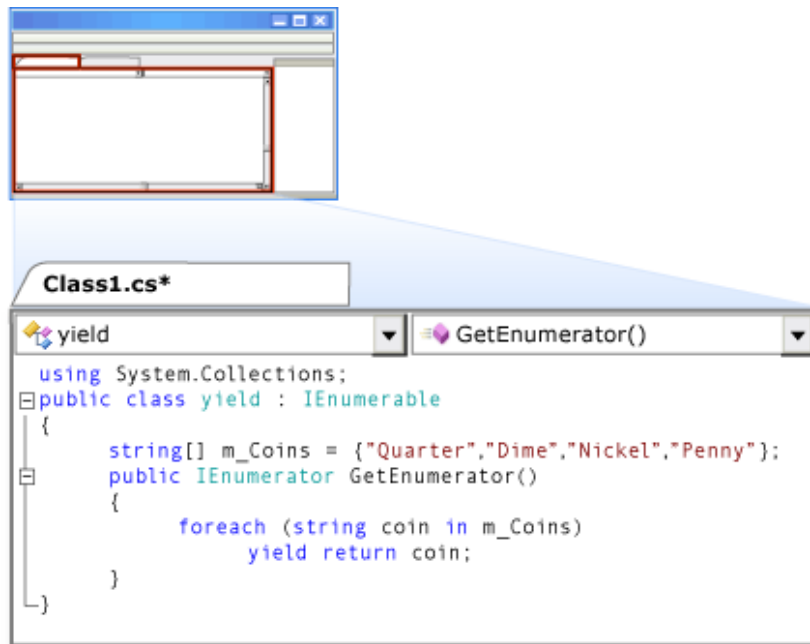
El editor de código colorea los tipos de símbolos (token) siguientes.

- Comentario
- Código excluido
- Identificador
- Palabra clave
- Número
- Operador
- Palabra clave del preprocesador
- Cadena
- Cadena (especificador literal @ de C#)
- Tipos de usuario
- Tipos de usuario (tipos de valor)
- Tipos de usuario (enumeraciones)
- Tipos de usuario (delegados)
- Sección CData XML
- Atributo de documento XML
- Comentario de documento XML
- Etiqueta de documento XML

Puede modificar la configuración del colores predeterminada mediante el Fuentes y colores, Entorno, Opciones (Cuadro de diálogo).

Palabras clave contextuales

El editor de código colorea de forma correcta las palabras clave contextuales. En el ejemplo siguiente, el tipo **yield** se colorea en verde azulado, mientras que la palabra clave **yield** se colorea en azul.



Colores de coincidencia de llaves

El editor de código facilita color en negrita o de resaltado para la coincidencia de llaves.

Color en negrita

Al editar cualquiera de los pares de construcciones de código siguientes, los pares de construcciones de código o de cadenas aparecen brevemente en negrita, para indicar que existe una asociación entre ellos:

" "	Una cadena
@ " "	Una cadena textual
#if, #endif	Directivas de preprocesador para secciones condicionales
#region, #endregion	Directivas de preprocesador para secciones condicionales
case, break	Palabras clave de instrucciones de control
default, break	Palabras clave de instrucciones de control
for, break	Palabras clave de expresiones de evaluación
for, continue	Palabras clave de expresiones de evaluación
foreach, break	Palabras clave de expresiones de evaluación
foreach, continue	Palabras clave de expresiones de evaluación
while, break	Palabras clave de expresiones de evaluación
while, continue	Palabras clave de expresiones de evaluación

Puede deshabilitar esta característica; para ello, desactive la propiedad **Resaltar con el delimitador automático** en el General, Editor de texto, Opciones (Cuadro de diálogo).

Color de resaltado

Cuando el cursor está situado inmediatamente antes de un delimitador inicial, o justo después del final, aparecen rectángulos grises que resaltan los dos delimitadores e indican que existe una asociación entre ellos. Esta característica está disponible para los pares siguientes:

{ }	llaves
[]	corchetes
()	paréntesis

Ejemplo

Para ilustrar los colores de coincidencia de llaves, escriba (no copie y pegue) el código siguiente en el editor de código.

other

```
class A
{
    public A()
    {
        if(true)
            int x =0;
        else
            int x =1;
    }
}
```

Configuración de color

La configuración de color se conserva en Valores de configuración de Visual Studio.

Metadatos como origen

Metadatos como origen permite ver metadatos que aparecen como código fuente de C# en un búfer de sólo lectura. Esto habilita una vista de declaraciones de tipos y miembros (sin implementaciones). Puede ver los metadatos como origen ejecutando el comando **Ir a definición** para los tipos de miembro cuyo código fuente no esté disponible desde el proyecto o solución.

Nota

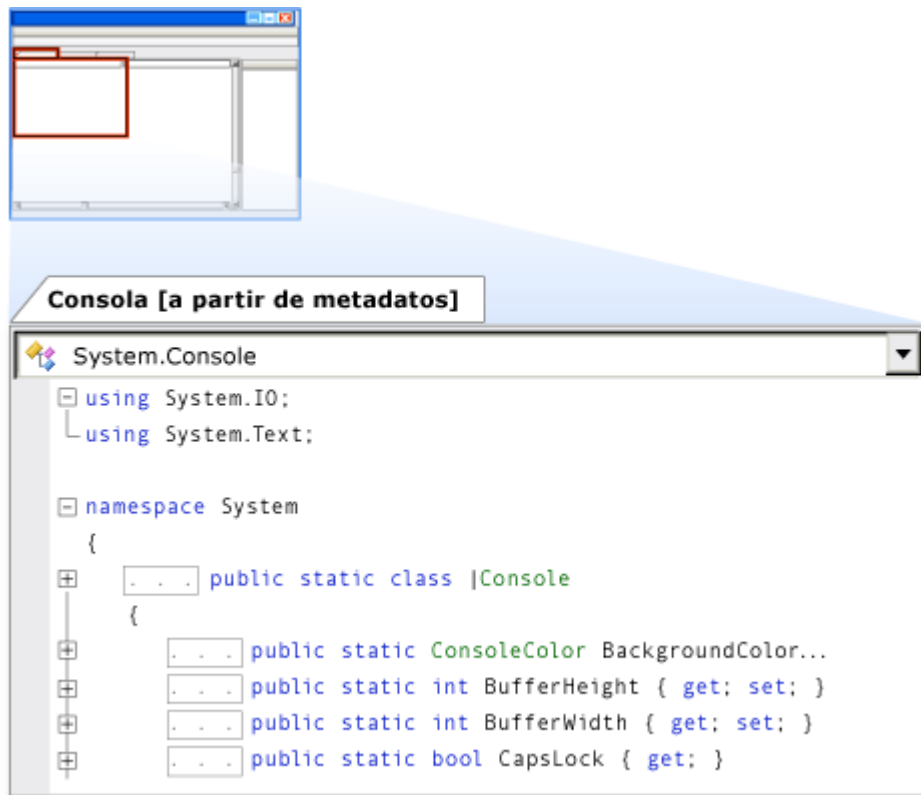
Cuando intente ejecutar el comando **Ir a definición** para tipos o miembros marcados como internos, el entorno de desarrollo integrado (IDE) no mostrará sus metadatos como origen, independientemente de que el ensamblado de referencia sea de confianza o no.

Puede ver los metadatos como origen en el Editor de código o en la ventana **Definición de código**.

Ver metadatos como origen en el Editor de código

Cuando ejecute el comando **Ir a definición** para un elemento cuyo código fuente no esté disponible, en el Editor de código aparecerá un documento con fichas que contiene una vista de los metadatos del elemento, mostrados en forma de origen. En la ficha del documento aparece el nombre del tipo, seguido por **[desde metadatos]**.

Por ejemplo, si ejecuta el comando **Ir a definición** para Console, en el Editor de código aparecerán metadatos para **Console** en forma de código fuente C# de aspecto similar a su declaración, pero sin implementación.



Ver metadatos como origen en la ventana Definición de código

Cuando la ventana **Definición de código** está activa o visible, el IDE ejecuta automáticamente el comando **Ir a definición** para los elementos que se encuentran bajo el cursor del Editor de código y para los elementos seleccionados en la **Vista de clases** o en el **Examinador de objetos**. Si el código fuente no está disponible para ese elemento, el IDE muestra los metadatos del elemento como origen en la ventana **Definición de código**.

Por ejemplo, si coloca el cursor dentro de la palabra **Console** en el Editor de código, en la ventana **Definición de código** aparecerán metadatos para **Console** como origen. El origen se parece a la declaración **Console**, pero sin implementación.

Si desea ver la declaración de un elemento que aparece en la ventana **Definición de código**, debe utilizar explícitamente el comando **Ir a definición**, porque la ventana **Definición de código** sólo tiene un nivel de profundidad.

Valores de configuración del IDE de Visual C#

La configuración de Visual C# es una configuración predefinida de ventanas de herramientas, menús y métodos abreviados de teclado. Esta configuración forma parte de la característica Valores de configuración de Visual Studio, que puede personalizar para adaptarla a sus hábitos de trabajo.

Ventanas y vistas

Característica	¿Se muestra de manera predeterminada?	Notas
Vista de clases	No	<ul style="list-style-type: none">• Vista de clases está disponible en el menú Ver.• Hay un filtro aplicado.
Ventana de comandos	No	
Ayuda dinámica (Ventana)	No	Al presionar la tecla F1, no se muestra la ventana de Ayuda dinámica.
Examinador de objetos	No	No aparecen de manera predeterminada los miembros heredados.
Resultados (Ventana)	No	
Explorador de soluciones	Sí	El Explorador de soluciones aparece acoplado en el lado derecho del IDE.
Página de inicio	Sí, al iniciar el IDE	La Página de inicio muestra artículos de información sobre MSDN RS para Visual C#.
Lista de tareas (Visual Studio)	No	
Cuadro de herramientas	Sí, cuando se crea una aplicación de Windows Forms	El Cuadro de herramientas aparece como una ventana contraída que se acopla en el lado izquierdo del IDE.

Teclado

Visual C# Express ofrece numerosos métodos abreviados de teclado: cuantos más conozca, más rápidamente podrá trabajar.

Por supuesto, los métodos abreviados de teclado más importantes son F5, que genera e inicia la aplicación, y F1, que inicia la Ayuda en línea. Puede descargar un cartel que contiene todos los métodos abreviados de teclado predeterminados (agrupados por tareas) para Visual C# desde el Centro de descargas de Microsoft.

Presione...

Presione	Para abrir
F1	Menú Ayuda, F1Help
F3	Menú Edición, FindNext
F4	Menú Ver, PropertiesWindow
F5	Menú Depurar, Iniciar
F6	Menú Ventana, NextSplitPane
F7	Menú Ver, ViewCode
F8	Menú Edición, GoToNextLocation
F9	Menú Depurar, ToggleBreakpoint
F10	Menú Depurar, StepOver
F11	Menú Depurar, StepInto
F12	Menú Edición, GoToDefinition
INSERT	Menú Edición, OvertypingMode
RETROCESO	Menú Edición, DeleteBackwards
ENTRAR	Menú Edición, BreakLine
ESC	Menú Ventana, ActivateDocumentWindow Menú Edición, CancelSelection
TAB	Menú Edición, InsertTab
FIN	Menú Edición LineEnd
INICIO	Menú Edición, LineStart

Presione Ctrl y...

Presione CTRL+	Para abrir
A	Menú Edición, SelectAll
B	Menú Depurar, NewBreakpoint
C	Menú Edición, Copiar
D	Menú Edición, GoToFindCombo
E	Menú Base de datos, Run
F	Editar, Find
G	Menú Edición, GoTo
H	Menú Edición, Replace
I	Menú Edición, IncrementalSearch
J	Menú Edición, ListMembers
L	Menú Edición, LineCut
N	Menú Archivo, NewFile
O	Menú Archivo, OpenFile
P	Menú Archivo, Print
Q	Menú Base de datos, RunSelection
S	Menú Archivo, SaveAll
T	Menú Edición, CharTranspose
U	Menú Edición, MakeLowercase
V	Menú Edición, Paste
W	Menú Edición, SelectCurrentWord
X	Menú Edición, Cut
Y	Edite el menú, Redo
Z	Menú Edición, Undo
Presione CTRL+	Para abrir
F1	Menú Ayuda, DynamicHelp
F2	Menú Ventana, MovetoDropDownBar

F3	Menú Edición, FindNextSelected
F4	Menú Ventana, CloseDocumentWindow
F5	Menú Depurar, StartWithoutDebugging
F6	Menú Ventana, NextDocumentWindow
F7	Menú Generar, Compile
F9	Menú Depurar, EnableBreakpoint
F10	Menú Depurar, RunToCursor
F11	Menú Depurar, ToggleDisassembly
F12	Menú Edición, GoToDeclaration
Presione CTRL+	Para abrir
Interrumpir	Menú Generar, Cancelar
Barra espaciadora	Menú Edición, CompleteWord
Retroceso	Menú Edición, WordDeleteToStart
Enter	Menú Edición, LineOpenAbove
Esc	No aplicable
Tabulador	Menú Ventana, NextDocumentWindow
Fin	Menú Edición, DocumentEnd
Inicio	Menú Edición, DocumentStart
Supr	Menú Edición, WordDeleteToEnd
]	Menú Edición, GotoBrace
.	Menú Herramientas, GoToCommandLine
-	Menú Ver, NavigateBackward
=	Menú Edición, SelectToLastGoBack
Flecha izquierda	Menú Edición, WordPrevious
Flecha derecha	Menú Edición, WordNext
Flecha Arriba	Menú Edición, ScrollLineUp
Flecha Abajo	Menú Edición, ScrollLineDown
Re Pág	Menú Ventana, Previous Tab Menú Edición, ViewTop

Av Pág	Menú Ventana, NextTab Menú Edición, ViewBottom
--------	------------------------------------------------

Presione Ctrl-Mayús y...

Presione CTRL+MAYÚS+	Para abrir
A	Menú Archivo, AddNewItem
B	Menú Generar, BuildSolution
C	Menú Ver, ClassView
E	Menú Ver, ResourceView
F	Menú Edición, FindinFiles
G	Menú Edición, OpenFile
H	Menú Edición, ReplaceinFiles
I	Menú Edición, ReverseIncrementalSearch
L	Menú Edición, LineDelete
M	Menú Herramientas, CommandWindowMarkMode
N	Menú Archivo, NewProject
O	Menú Archivo, OpenProject
P	Menú Herramientas, RunTemporaryMacro
R	Menú Herramientas, RecordTemporaryMacro
S	Menú Archivo, SaveSelectedItems
T	Menú Edición, WordTranspose
U	Menú Edición, MakeUppercase
V	Menú Edición, CycleClipboardRing
Presione CTRL+MAYÚS+	Para abrir
F3	Menú Edición, FindPreviousSelected
F5	Menú Depurar, Restart
F6	Menú Ventana, PreviousDocumentWindow
F9	Menú Depurar, ClearAllBreakpoints
F10	Menú Depurar, SetNextStatement
F11	No aplicable

F12	Menú Ver, NextTask
Presione CTRL+MAYÚS+	Para abrir
Espacio	Menú Edición, ParameterInfo
Enter	Menú Edición, LineOpenBelow
Tabulador	Menú Ventana, PreviousDocumentWindow
Fin	Menú Edición, DocumentEndExtend
Inicio	Menú Edición, DocumentStartExtend
]	Menú Edición, GotoBraceExtend
-	Menú Ver, NavigateForward
Flecha izquierda	Menú Edición, WordPreviousExtend
Flecha derecha	Menú Edición, WordNextExtend
Re Pág	Menú Edición, ViewTopExtend
Av Pág	Menú Edición, ViewBottomExtend
1	Menú Ver, BrowseNext
2	Menú Ver, BrowsePrevious
8	Menú Ver, PopBrowseContext

Presione Alt y...

Presione ALT+	Para abrir
F5	Menú Base de datos, StepInto
F6	Menú Ventana, NextPane
F8	Menú Ver, MacroExplorer
F10	Menú Base de datos, StepInto
F11	Menú Herramientas, MacrosIDE
F12	Menú Edición, FindSymbol
Interrumpir	Menú Depurar, BreakAll
Retroceso	Menú Edición, Undo
-	Menú Ver, ObjectBrowserBack
Flecha izquierda	Menú Ver, WebNavigateBack

Flecha derecha	Menú Ver, WebNavigateForward Menú Edición, CompleteWord
----------------	---------------------------------------------------------

Presione Mayúsculas y...

Presione Mayúsculas y...	Para abrir
F1	Menú Ayuda, WindowHelp
F3	Menú Edición, FindPrevious
F4	Menú Ver, PropertyPages
F5	Menú Depurar, StopDebugging
F6	Menú Ventana, PreviousSplitPane
F7	Menú Ver, ViewDesigner
F8	Menú Edición, GoToPrevLocation
F9	Menú Depurar, QuickWatch
F11	Menú Depurar, StepOut
F12	Menú Edición, GoToReference
Retroceso	Menú Edición, DeleteBackwards
Esc	Menú Ventana, CloseToolWindow
Tabulador	Menú Edición, TabLeft
Fin	Menú Edición, LineEndExtend
Inicio	Menú Edición, LineStartExtend

Presione MAYÚS+ALT+...

Presione MAYÚS+ALT+	Para abrir
A	Menú Archivo, AddExistingItem
T	Menú Edición, LineTranspose
F2	Menú Ayuda, Indexresults
F3	Menú Ayuda, Searchresults
F6	Menú Ventana, PreviousPane
F12	Menú Edición, QuickFindSymbol
Enter	Menú Ver, FullScreen
Fin	Menú Edición, LineEndExtendColumn

Inicio	Menú Edición, LineStartExtendColumn
-	Menú Ver, ObjectBrowserForward
Flecha izquierda	Menú Edición, CharLeftExtendColumn
Flecha derecha	Menú Edición, CharRightExtendColumn
Flecha Arriba	Menú Edición, LineUpExtendColumn
Flecha Abajo	Menú Edición, LineDownExtendColumn

Presione CTRL+ALT+...

Presione CTRL+ALT+	Para abrir
A	Menú Ver, CommandWindow
B	Menú Depurar, Autos
C	Menú Depurar, CallStack
D	Menú Depurar, Disassembly
E	Menú Depurar, Exceptions
F	Menú Ver, Favorites
G	Menú Depurar, Registers
H	Menú Depurar, Threads
I	Menú Depurar, Immediate
J	Menú Ver, ObjectBrowser
K	Menú Ver, TaskList
L	Menú Ver, SolutionExplorer
M	Menú Depurar, Memory
N	Menú Depurar, RunningDocuments
O	Menú Ver, Output
P	Menú Herramientas, DebugProcesses
Q	Menú Depurar, QuickWatch
R	Menú Ver, ShowWebBrowser
S	Menú Ver, ServerExplorer
T	Menú Ver, DocumentOutline

U	Menú Depurar, Módulos
V	Menú Depurar, Locals
W	Menú Depurar, Watch
X	Menú Ver, Toolbox
F1	Menú Ayuda, Contents
F2	Menú Ayuda, Index
F3	Menú Ayuda, Search
F12	Menú Ver, FindSymbolResults
Interrumpir	Menú Depurar, Autos
Ins	Menú Proyecto, Override

Presione CTRL+MAYÚS+ALT+...

Presione CTRL+MAYÚS+ALT+	Para abrir
Flecha izquierda	Menú Edición, WordPreviousExtendColumn
Flecha derecha	Menú Edición, WordNextExtendColumn