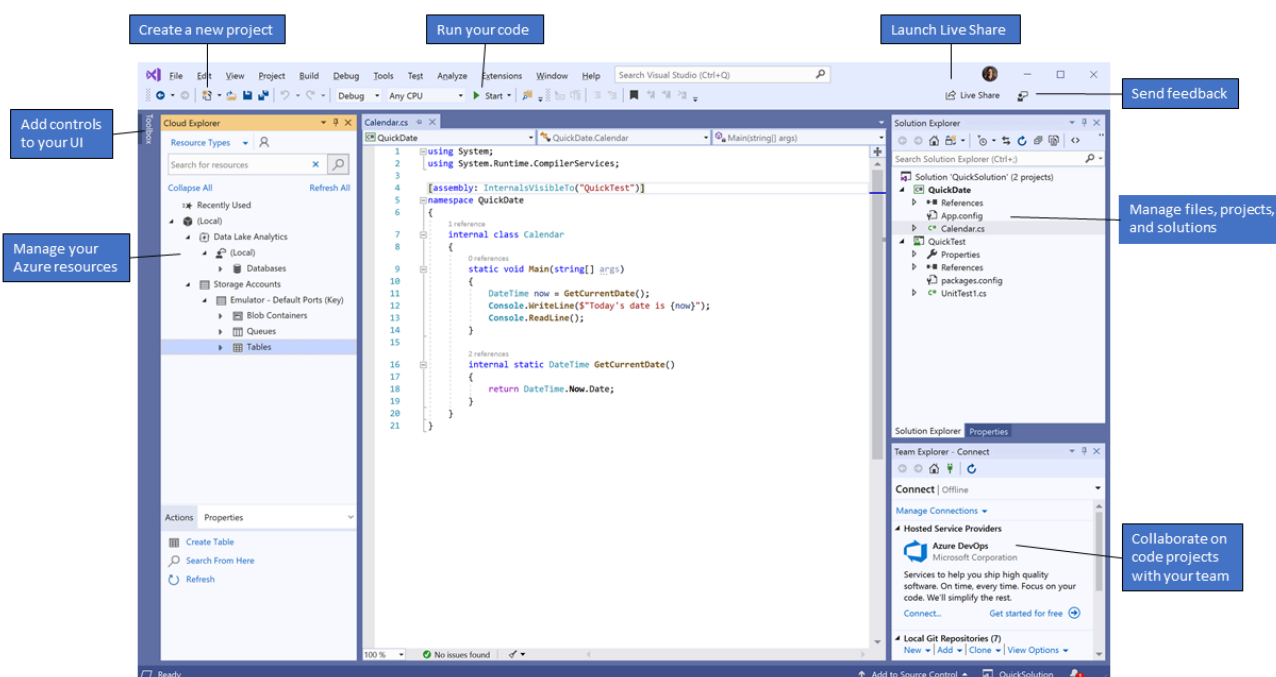


Visual Studio - Primer Contacto

El *entorno de desarrollo integrado* de Visual Studio es un panel de inicio creativo que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Un entorno de desarrollo integrado (IDE) es un programa con numerosas características que se pueden usar para muchos aspectos del desarrollo de software. Más allá del editor estándar y el depurador que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.



En esta imagen se muestra Visual Studio con un proyecto abierto y varias ventanas de herramientas clave que probablemente usará:

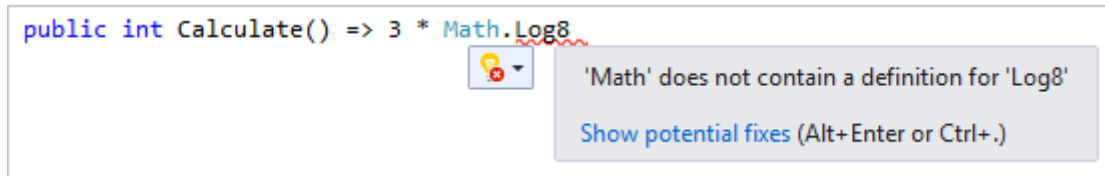
- El **Explorador de soluciones** (parte superior derecha) permite ver, navegar y administrar los archivos de código. El **Explorador de soluciones** puede ayudar a organizar el código al agrupar los archivos en **soluciones y proyectos**.
- La **ventana del editor** (centro), donde es probable que pase la mayor parte del tiempo, muestra el contenido del archivo. Es donde puede editar código o diseñar una interfaz de usuario, como una ventana con botones y cuadros de texto.
- **Team Explorer** (parte inferior derecha) permite realizar el seguimiento de los elementos de trabajo y compartir código con otros usuarios mediante tecnologías de control de versiones como **Git** y **Control de versiones de Team Foundation (TFVC)**.

Características de productividad populares

Algunas de las características populares de Visual Studio que ayudan a ser más productivos durante el desarrollo de software incluyen:

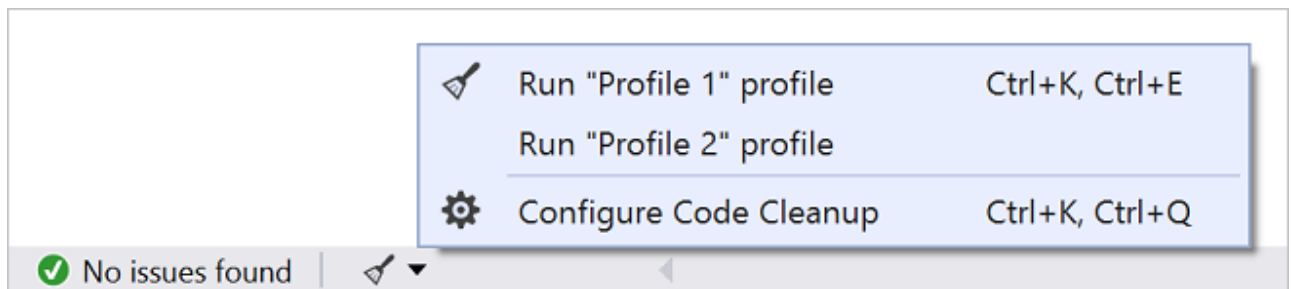
- Subrayados ondulados y [Acciones rápidas](#)

Los subrayados ondulados son rayas con formas de onda debajo de las palabras que alertan de errores o posibles problemas en el código a medida que se escribe. Estas pistas visuales permiten corregir problemas inmediatamente sin esperar a que el error se detecte durante la compilación o cuando se ejecute el programa. Si mantiene el cursor sobre un subrayado ondulado, se ve información adicional sobre el error. También puede aparecer una bombilla en el margen izquierdo con acciones, conocidas como Acciones rápidas, para corregir el error.



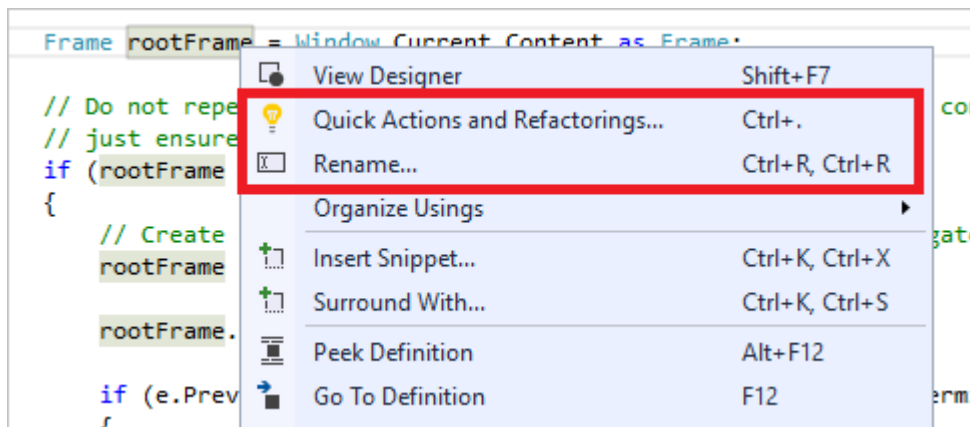
- Limpieza de código

Con solo un clic en un botón, puede dar formato al código y aplicar cualquier corrección de código sugerida por la [configuración del estilo del código](#), las [convenciones de .editorconfig](#) y los [analizadores de Roslyn](#). **Limpieza de código** ayuda a solucionar problemas con el código antes de la revisión del código. (Actualmente solo está disponible para el código de C#).



- [Refactorización](#)

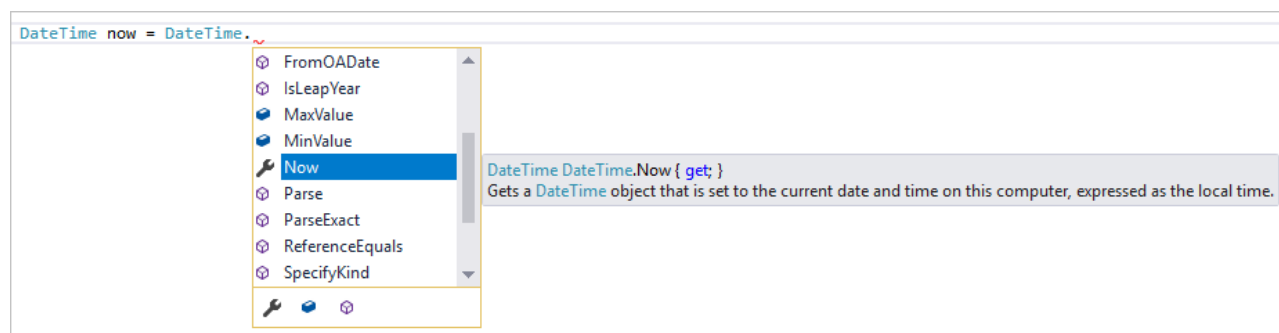
La refactorización incluye operaciones como el cambio de nombre inteligente de variables, la extracción de una o más líneas de código en un nuevo método, el cambio del orden de los parámetros de método, etc.



- [IntelliSense](#)

IntelliSense es un término que define un conjunto de características que muestran información sobre el código directamente en el editor y, en algunos casos, escriben pequeños fragmentos de código

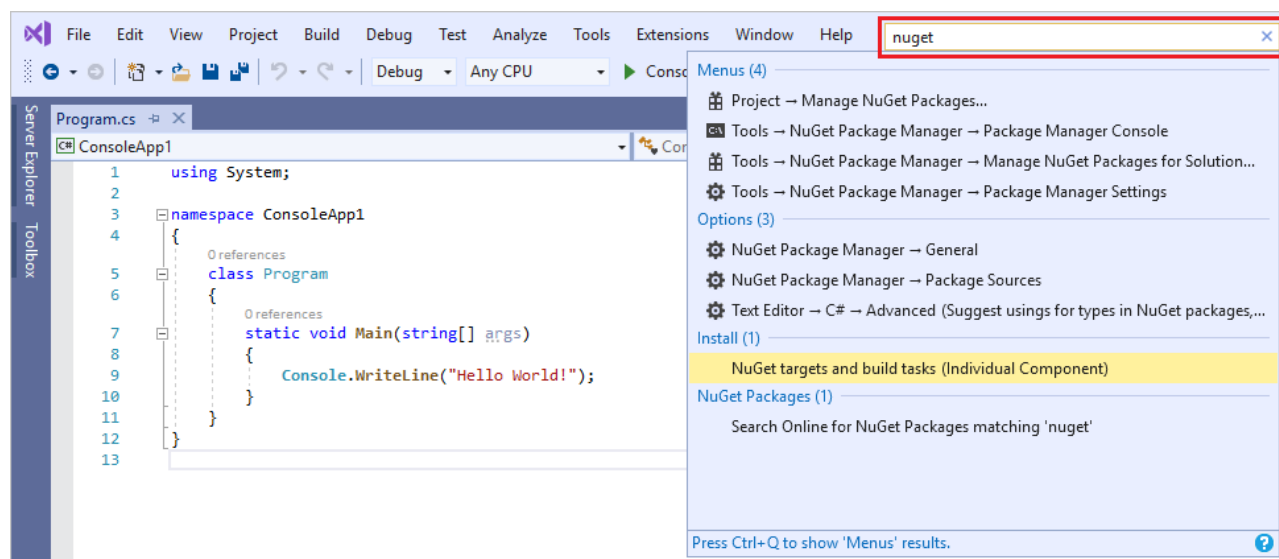
automáticamente. Es como tener documentación básica insertada en el editor, lo que evita tener que buscar información escrita en cualquier otro lugar. Las características de IntelliSense varían según el lenguaje. Para más información, vea [IntelliSense para C#](#), [IntelliSense para Visual C++](#), [IntelliSense para JavaScript](#) e [IntelliSense de Visual Basic](#). La siguiente ilustración muestra cómo IntelliSense muestra una lista de miembros de un tipo:



- Cuadro de búsqueda

Visual Studio puede parecer abrumador a veces con tantas propiedades, opciones y menús. El cuadro de búsqueda supone una excelente manera de encontrar rápidamente lo que necesita en Visual Studio. Al empezar a escribir el nombre de lo que está buscando, Visual Studio muestra resultados que llevan exactamente a donde necesita ir. Si necesita agregar funcionalidad a Visual Studio para, por ejemplo, agregar compatibilidad con otro lenguaje de programación, el cuadro de búsqueda proporciona resultados que abren el Instalador de Visual Studio para instalar un componente individual o una carga de trabajo.

Presione **Ctrl+Q** como acceso directo al cuadro de búsqueda.

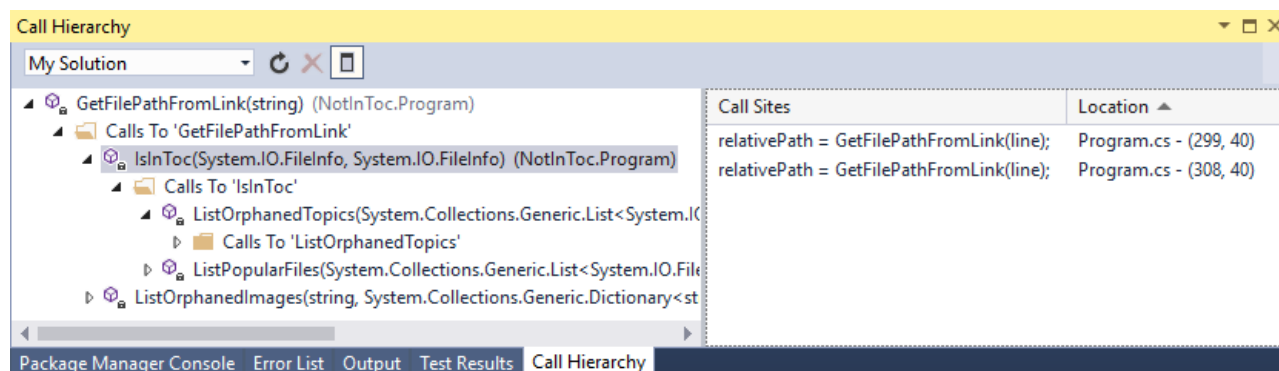


- Live Share

Edita y depura colaborativamente con otros usuarios en tiempo real, sin importar el tipo de aplicación o el lenguaje de programación. Puede compartir al instante y de manera segura su proyecto y, según sea necesario, las sesiones de depuración, las instancias de terminal, las aplicaciones web de localhost, las llamadas de voz, etc.

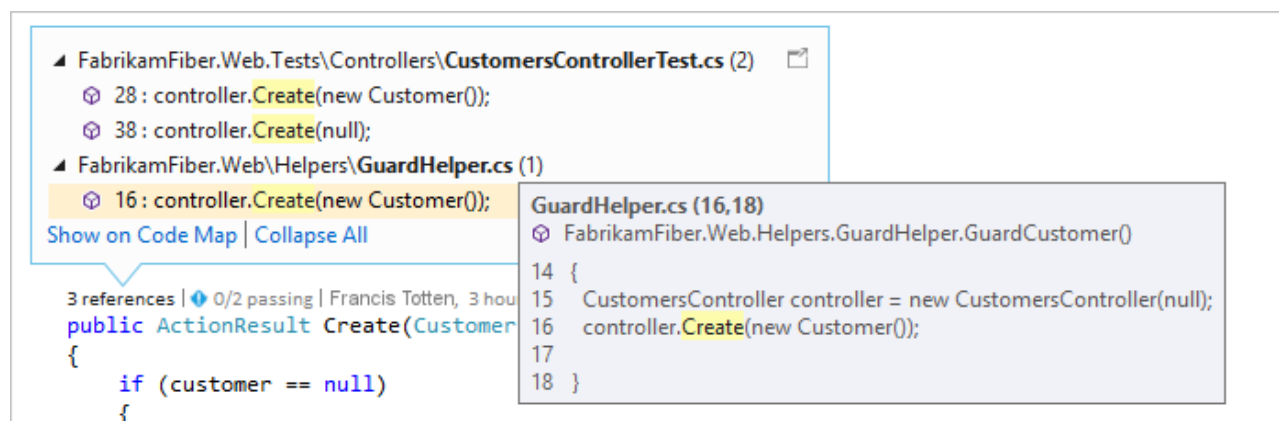
- Jerarquía de llamadas

En la ventana **Jerarquía de llamadas** se muestran los métodos que llaman a un método seleccionado. Puede ser información útil si está pensando en cambiar o quitar el método, o si está intentando realizar un seguimiento de un error.



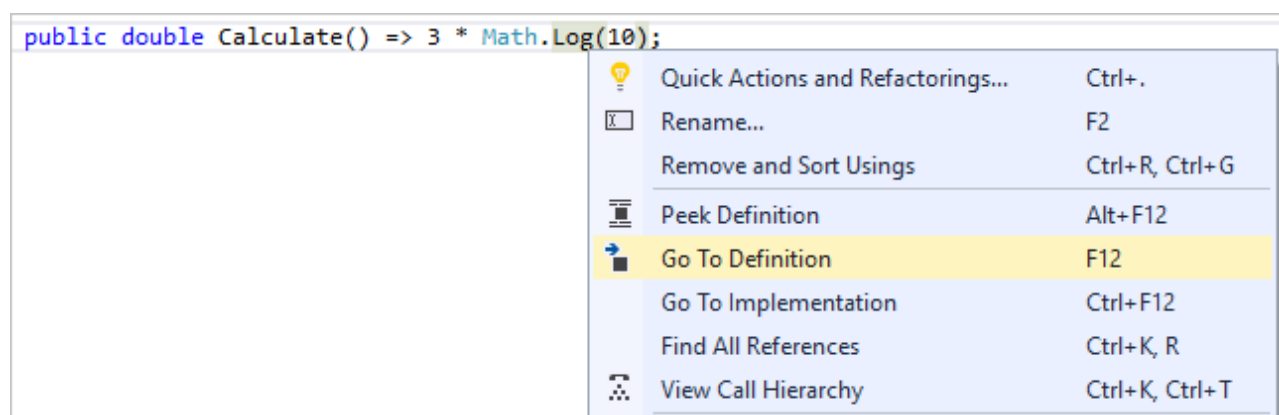
- CodeLens

CodeLens ayuda a buscar referencias al código, cambios en él, errores vinculados, elementos de trabajo, revisiones de código y pruebas unitarias, todo sin salir del editor.



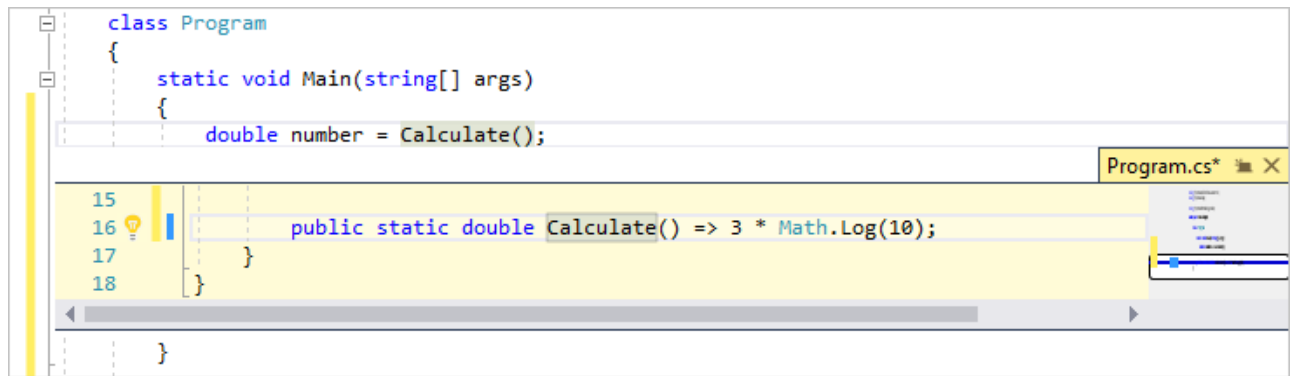
- Ir a definición

La característica Ir a definición lleva directamente a la ubicación donde se define una función o un tipo.



- Ver la definición

En la ventana **Ojea la definición** se muestra la definición de un tipo o método sin abrir en realidad un archivo independiente.



Usar IntelliSense y la refactorización

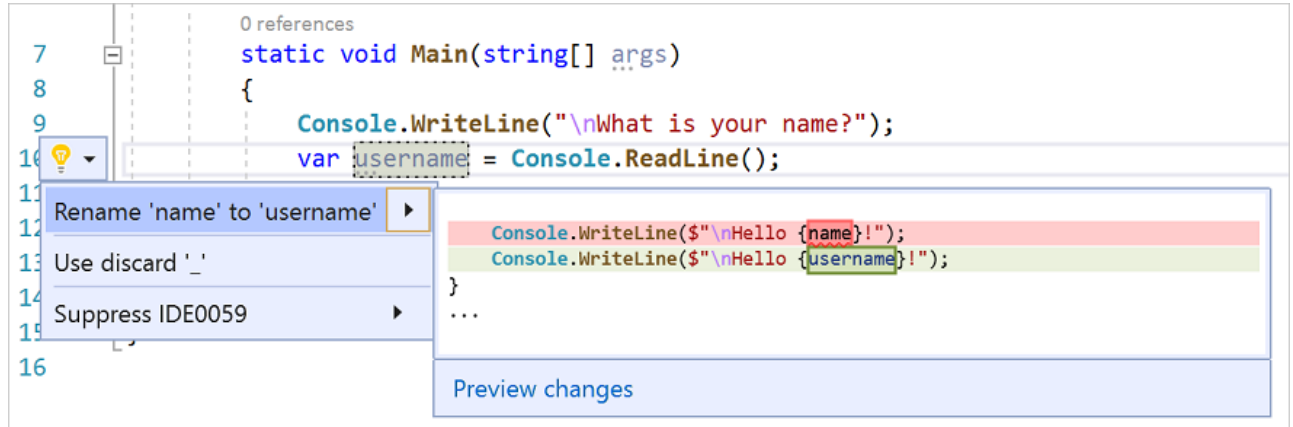
Veamos un par de formas en las que la [refactorización](#) e [IntelliSense](#) pueden ayudar a crear código de forma más eficaz.

En primer lugar, vamos a cambiar el nombre de la variable `name`:

1. Haga doble clic en la variable `name` para seleccionarla.
2. Escriba el nombre nuevo de la variable, `username`.

Observe que aparece un cuadro gris alrededor de la variable y una bombilla en el margen.

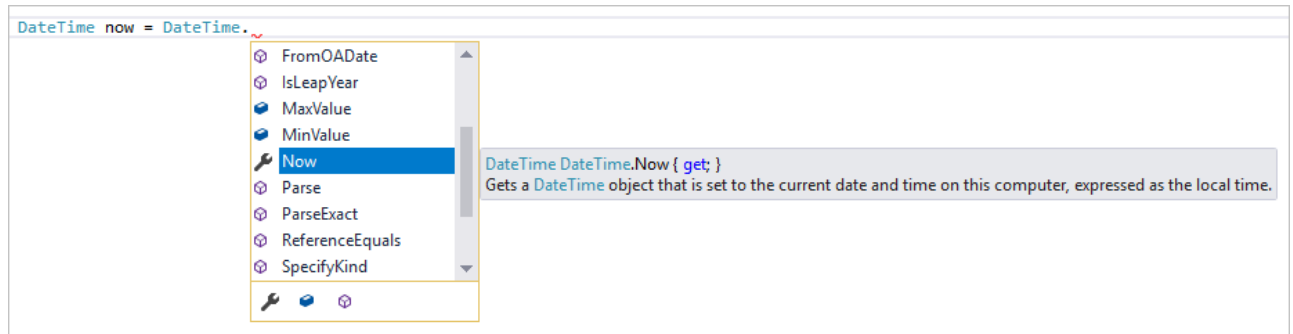
3. Haga clic en el icono de bombilla para mostrar las [Acciones rápidas](#) disponibles. Seleccione **Rename 'name' to 'username'** (Cambiar "name" a "username").



Se cambia el nombre de la variable en el proyecto, que en nuestro caso es solo en dos lugares.

4. Ahora echemos un vistazo a IntelliSense. Por debajo de la línea que dice `Console.WriteLine($"\\nHello {username}!");`, escriba `DateTime now = DateTime..`

Los miembros de la clase `DateTime` se muestran en un cuadro. Además, la descripción del miembro seleccionado actualmente se muestra en un cuadro independiente.



5. Seleccione el miembro denominado **Now**, que es una propiedad de la clase, haciendo doble clic en él o presionando la tecla **Tab**. Agregue un punto y coma al final de la línea de código para completarla.
6. Por debajo, escriba o pegue las líneas de código siguientes:

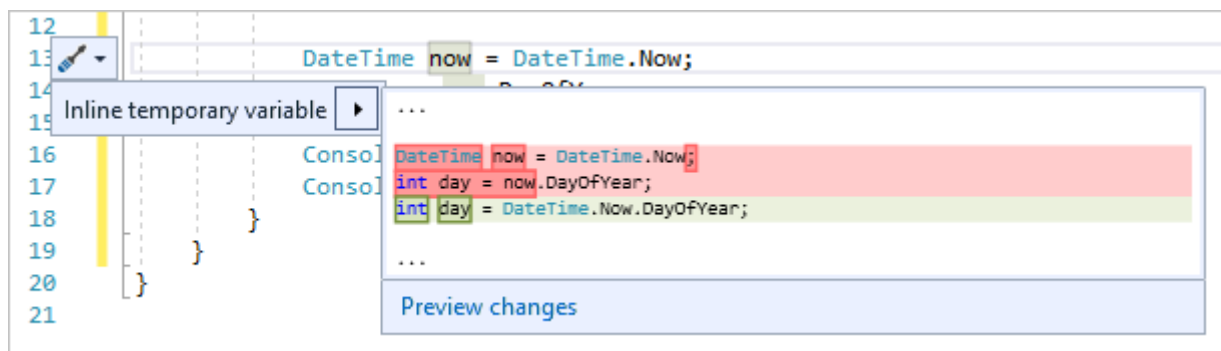
```
int dayOfYear = now.DayOfYear;

Console.WriteLine("Day of year: ");
Console.WriteLine(dayOfYear);
```

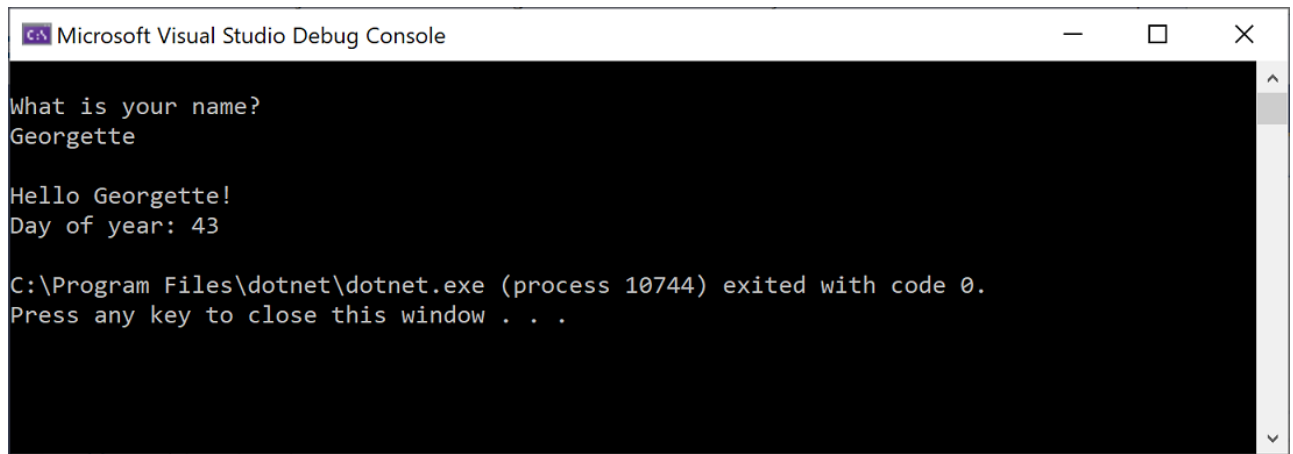
7. A continuación, vamos a volver a usar la refactorización para que hacer que el código sea más conciso. Haga clic en la variable `now` en la línea `DateTime now = DateTime.Now;`.

Observe que aparece un pequeño icono de destornillador en el margen de esa línea.

8. Haga clic en el icono de destornillador para ver las sugerencias disponibles en Visual Studio. En este caso, se muestra la refactorización **Variable temporal en línea** para quitar una línea de código sin cambiar el comportamiento general del código:



9. Haga clic en **Variable temporal en línea** para refactorizar el código.
10. Vuelva a ejecutar el programa presionando **Ctrl+F5**. La salida tendrá un aspecto similar a este:



```
Microsoft Visual Studio Debug Console

What is your name?
Georgette

Hello Georgette!
Day of year: 43

C:\Program Files\dotnet\dotnet.exe (process 10744) exited with code 0.
Press any key to close this window . . .
```

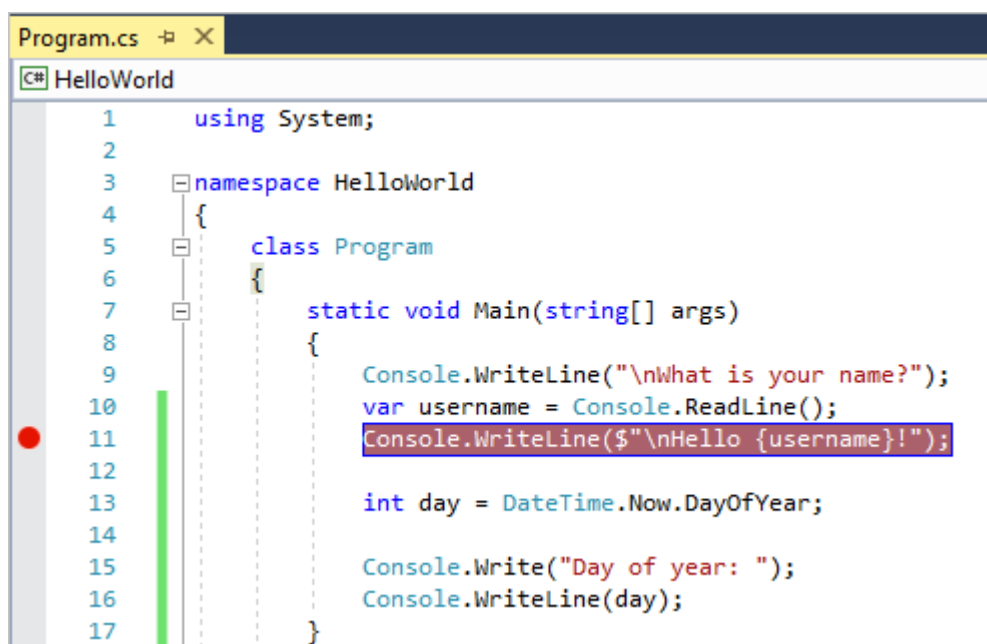
Depurar código

Cuando se escribe código, debe ejecutarlo y probarlo para ver si hay errores. El sistema de depuración de Visual Studio permite examinar el código por cada instrucción e inspeccionar las variables a medida que se avanza. Puede establecer *puntos de interrupción* que detengan la ejecución del código en una línea determinada. Puede observar cómo cambia el valor de una variable a medida que el código se ejecuta, etc.

Vamos a establecer un punto de interrupción para ver el valor de la variable `username` mientras el programa se encuentra "en marcha".

1. Busque la línea de código en la que se indica `Console.WriteLine($"\\nHello {username}!");`. Para establecer un punto de interrupción en esta línea de código, es decir, para que el programa detenga la ejecución en esta línea, haga clic en el margen izquierdo del editor. También puede hacer clic en cualquier lugar de la línea de código y, después, presionar **F9**.

Aparece un círculo de color rojo en el margen izquierdo y el código se resalta en color rojo.



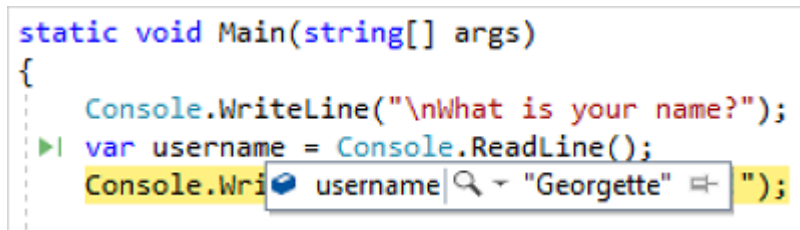
```
Program.cs
C# HelloWorld

1  using System;
2
3  namespace HelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("\\nWhat is your name?");
10             var username = Console.ReadLine();
11             Console.WriteLine($"\\nHello {username}!");
12
13             int day = DateTime.Now.DayOfYear;
14
15             Console.Write("Day of year: ");
16             Console.WriteLine(day);
17         }
18     }
19 }
```

2. Para iniciar la depuración, seleccione **Depurar > Iniciar depuración** o presione **F5**.
3. Cuando aparezca la ventana de consola y se le solicite su nombre, escríbalo y presione **Entrar**.

El foco se devuelve al editor de código de Visual Studio y la línea de código con el punto de interrupción se resalta en color amarillo. Esto significa que es la siguiente línea de código que el programa va a ejecutar.

- Mantenga el ratón sobre la variable `username` para ver su valor. Como alternativa, puede hacer clic con el botón derecho en `username` y seleccionar **Agregar inspección** para agregar la variable a la ventana **Inspección**, donde también puede ver su valor.



```
static void Main(string[] args)
{
    Console.WriteLine("\nWhat is your name?");
    var username = Console.ReadLine();
    Console.WriteLine(username + "Georgette");
}
```

- Para permitir que el programa se ejecute hasta completarse, vuelva a presionar **F5**.

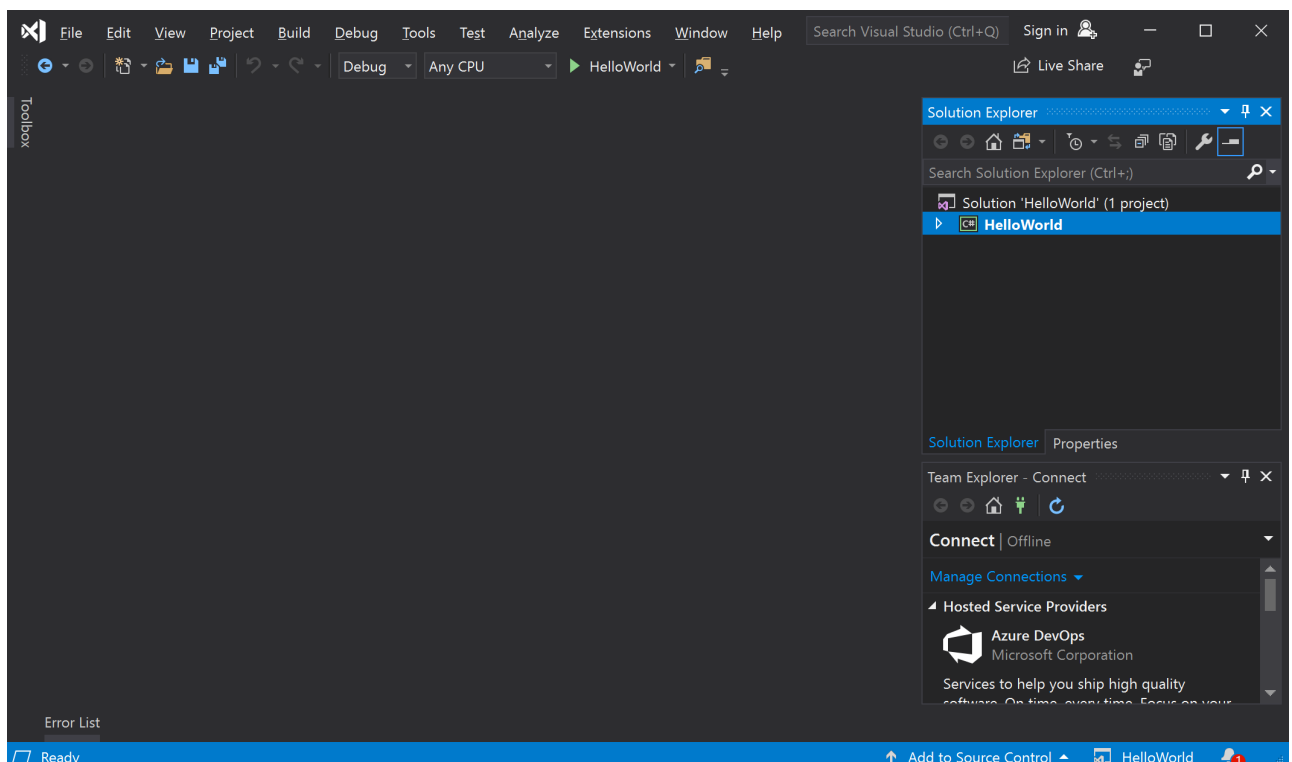
Para obtener más detalles sobre el proceso de depuración de Visual Studio, consulte [Paseo por las características del depurador](#).

Personalizar Visual Studio

Puede personalizar la interfaz de usuario de Visual Studio, incluso cambiar el tema de color predeterminado. Para cambiar al tema **Oscuro**:

- En la barra de menús, seleccione **Herramientas > Opciones** para abrir el cuadro de diálogo **Opciones**.
- En la página de opciones **Entorno > General**, cambie la selección de **Tema de color** a **Oscuro** y, después, elija **Aceptar**.

El tema de color para todo el IDE se cambia a **Oscuro**.



Para obtener información sobre otras maneras de personalizar el IDE, vea [Personalizar Visual Studio](#).

Selección de la configuración del entorno

Vamos a configurar Visual Studio para que use la configuración de entorno adaptada a los desarrolladores de C#.

1. En la barra de menús, elija **Herramientas > Importar y exportar configuraciones**.
2. En el **Asistente para importar y exportar configuraciones**, seleccione **Restablecer todas las configuraciones** en la primera página y, luego, seleccione **Siguiente**.
3. En la página **Guardar configuración actual**, seleccione una opción para guardar o no la configuración actual y, luego, elija **Siguiente**. (Si no personalizó la configuración, seleccione **No, just reset settings, overwriting my current settings** [No, solo restablecer la configuración y sobrescribir la configuración actual]).
4. En la página **Elija una colección de configuraciones predeterminadas**, elija **Visual C#** y, luego, **Finalizar**.
5. En la página **Restablecimiento completado**, elija **Cerrar**.