This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

Visual Studio 2010 - Visual C#

**What's New in Visual C# 2010**

This page lists the new and enhanced features available in Visual C# 2010. The new features work together to improve Microsoft Office programmability, increase your ability to explore code, provide support for test-driven development, and more. For more information, see [Visual Studio 2010 Product Highlights](http://msdn.microsoft.com/en-us/library/dd547188.aspx).

Click the links in the following sections to read more about a feature.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifC# 4.0 Language and Compiler

**Dynamic Support**

Visual C# 2010 provides support for late binding to dynamic types by introducing a new type, **dynamic**. This addition enables many new scenarios, including simplified access to COM APIs such as the Office Automation APIs, to dynamic APIs such as IronPython libraries, and to the HTML Document Object Model (DOM). For more information, see [Using Type dynamic (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/dd264736.aspx) and [dynamic (C# Reference)](http://msdn.microsoft.com/en-us/library/dd264741.aspx).

**Office Programmability**

Access to COM interfaces, including the Office Automation APIs, is greatly enhanced by the addition of named and optional arguments, the **dynamic** type, indexed properties, and optional **ref** modifiers.

For more information, see [Named and Optional Arguments (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/dd264739.aspx), [Using Type dynamic (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/dd264736.aspx), [How to: Use Indexed Properties in COM Interop Programming (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/ee310208.aspx), and [How to: Access Office Interop Objects by Using Visual C# 2010 Features (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/dd264733.aspx).

**Type Equivalence Support**

You can now deploy an application that has embedded type information instead of type information that is imported from a Primary Interop Assembly (PIA). With embedded type information, your application can use types from an assembly without requiring a reference to the runtime assembly or Primary Interop Assembly (PIA). If various versions of the runtime assembly are published, the application that contains the embedded type information can work with the different versions without having to be recompiled. For more information, see [/link (C# Compiler Options)](http://msdn.microsoft.com/en-us/library/dd264728.aspx). For an example, see [Walkthrough: Embedding Types from Managed Assemblies (C# and Visual Basic)](http://msdn.microsoft.com/en-us/library/dd409610.aspx).

**Covariance and Contravariance**

*Covariance* enables you to use a more derived type than that specified by the generic parameter, whereas *contravariance* enables you to use a less derived type. This allows for implicit conversion of classes that implement variant interfaces and provides more flexibility for matching method signatures with variant delegate types. Variant interfaces and delegates can be created by using the new [in](http://msdn.microsoft.com/en-us/library/dd469484.aspx) and [out](http://msdn.microsoft.com/en-us/library/dd469487.aspx) language keywords. The .NET Framework also introduces variance support for several existing generic interfaces and delegates, including the [IEnumerable<(Of <(T>)>)](http://msdn.microsoft.com/en-us/library/9eekhta0.aspx) interface and the [Func<(Of <(TResult>)>)](http://msdn.microsoft.com/en-us/library/bb534960.aspx) and [Action<(Of <(T>)>)](http://msdn.microsoft.com/en-us/library/018hxwa8.aspx) delegates. For more information, see [Covariance and Contravariance (C# and Visual Basic)](http://msdn.microsoft.com/en-us/library/ee207183.aspx).

**New Command-Line Options**

The [/langversion](http://msdn.microsoft.com/en-us/library/f4ckecs0.aspx) command-line option causes the compiler to accept only syntax that is valid in the specified version of C#.

The [/appconfig](http://msdn.microsoft.com/en-us/library/ee523958.aspx) compiler option enables a C# application to specify the location of an assembly's application configuration file to the compiler.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifVisual C# Integrated Development Environment

The following sections describe enhancements to the Visual Studio integrated development environment (IDE).

**Call Hierarchy**

*Call Hierarchy* enables you to navigate through your code by displaying the following:

* All calls to and from a selected method, property, or constructor
* All implementations of an interface member
* All overrides of a virtual or abstract member

This enables you to better understand how code flows and to evaluate the effects of changes to code. For more information, see [Call Hierarchy](http://msdn.microsoft.com/en-us/library/dd409859.aspx).

**Navigate To**

You can use the **Navigate To** feature to search for a symbol or file in source code. You can search for keywords that are contained in a symbol by using Camel Casing and underscore characters to divide the symbol into keywords.

For more information, see [How to: Search for Objects, Definitions, and References (Symbols)](http://msdn.microsoft.com/en-us/library/4sadchd3.aspx).

**Highlighting References**

When you click a symbol in source code, all instances of that symbol are highlighted in the document. To move to the next or previous highlighted symbol, you can use CTRL+SHIFT+DOWN ARROW or CTRL+SHIFT+UP ARROW. For more information, see [How to: Use Reference Highlighting](http://msdn.microsoft.com/en-us/library/ee349251.aspx).

**Generate From Usage**

The Generate From Usage feature enables you to use classes and members before you define them. Without leaving your current location in code, you can generate a stub for a class, constructor, method, property, field, or enum that you want to use but have not yet defined. This minimizes interruption to your workflow.

Generate From Usage supports programming styles such as test-first development. For more information, see [Generate From Usage](http://msdn.microsoft.com/en-us/library/dd409796.aspx) and [Walkthrough: Test-First Support with the Generate From Usage Feature](http://msdn.microsoft.com/en-us/library/dd998313.aspx).

**IntelliSense Suggestion Mode**

IntelliSense now provides two alternatives for IntelliSense statement completion: *completion mode* and *suggestion mode*. Suggestion mode is used when classes and members are used before they are defined. For more information, see [List Members](http://msdn.microsoft.com/en-us/library/exbffbc2.aspx).

**Live Semantic Errors**

The Live Semantic Errors feature has been enhanced in Visual C# 2010. The use of wavy underlines to signal errors and warnings as you type has been extended to include constructs that are outside of method bodies, such as return types, parameter types, and default values in method declarations.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifSample Applications

Visual C# 2010 includes sample applications that demonstrate the use of new C# features to simplify Office programming, provide access to IronPython libraries, create dynamic objects, develop and use covariant and contravariant generic delegates, and more. For information about Visual C# language samples and how to access them, see [Visual C# Sample Applications](http://msdn.microsoft.com/en-us/library/z9hsy596.aspx).