Visual Studio 2010 - Visual C#

**How to: Use Indexed Properties in COM Interop Programming (C# Programming Guide)**

*Indexed properties* improve the way in which COM properties that have parameters are consumed in C# programming. Indexed properties work together with other features introduced in Visual C# 2010, such as [named and optional arguments](http://msdn.microsoft.com/en-us/library/dd264739.aspx), a new type ([dynamic](http://msdn.microsoft.com/en-us/library/dd264741.aspx)), and [embedded type information](http://msdn.microsoft.com/en-us/library/dd409610.aspx), to enhance Microsoft Office programming.

In earlier versions of C#, methods are accessible as properties only if the **get** method has no parameters and the **set** method has one and only one value parameter. However, not all COM properties meet those restrictions. For example, the Excel [Range](http://go.microsoft.com/fwlink/?LinkId=166053) property has a **get** accessor that requires a parameter for the name of the range. In the past, because you could not access the **Range** property directly, you had to use the **get\_Range** method instead, as shown in the following example.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl05_code');" \o "Copy Code)

// Visual C# 2008 and earlier.

var excelApp = new Excel.Application();

// . . .

Excel.Range targetRange = excelApp.get\_Range("A1", Type.Missing);

Indexed properties enable you to write the following instead:

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl06_code');" \o "Copy Code)

// Visual C# 2010.

var excelApp = new Excel.Application();

// . . .

Excel.Range targetRange = excelApp.Range["A1"];

|  |
| --- |
| **Description: NoteNote** |
| The previous example also uses the [optional arguments](http://msdn.microsoft.com/en-us/library/dd264739.aspx) feature, introduced in Visual C# 2010, which enables you to omit **Type.Missing**. |

Similarly, to set the value of the **Value** property of a [Range](http://go.microsoft.com/fwlink/?LinkId=179211) object in Visual C# 2008 and earlier, two arguments are required. One supplies an argument for an optional parameter that specifies the type of the range value. The other supplies the value for the **Value** property. Before Visual C# 2010, C# allowed only one argument. Therefore, instead of using a regular set method, you had to either use the **set\_Value** method or a different property, [Value2](http://go.microsoft.com/fwlink/?LinkId=166050). The following examples illustrate these techniques. Both set the value of the A1 cell to Name.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

// Visual C# 2008.

targetRange.set\_Value(Type.Missing, "Name");

// Or

targetRange.Value2 = "Name";

Indexed properties enable you to write the following code instead.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

// Visual C# 2010.

targetRange.Value = "Name";

You cannot create indexed properties of your own. The feature only supports consumption of existing indexed properties.

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

The following code shows a complete example. For more information about how to set up a project that accesses the Office API, see [How to: Access Office Interop Objects by Using Visual C# 2010 Features (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/dd264733.aspx).

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl01_code');" \o "Copy Code)

// You must add a reference to Microsoft.Office.Interop.Excel to run

// this example.

using System;

using Excel = Microsoft.Office.Interop.Excel;

namespace IndexedProperties

{

class Program

{

static void Main(string[] args)

{

CSharp2010();

//CSharp2008();

}

static void CSharp2010()

{

var excelApp = new Excel.Application();

excelApp.Workbooks.Add();

excelApp.Visible = true;

Excel.Range targetRange = excelApp.Range["A1"];

targetRange.Value = "Name";

}

static void CSharp2008()

{

var excelApp = new Excel.Application();

excelApp.Workbooks.Add(Type.Missing);

excelApp.Visible = true;

Excel.Range targetRange = excelApp.get\_Range("A1", Type.Missing);

targetRange.set\_Value(Type.Missing, "Name");

// Or

//targetRange.Value2 = "Name";

}

}

}