Visual Studio 2010

**Walkthrough: Creating and Using Dynamic Objects (C# and Visual Basic)**

Dynamic objects expose members such as properties and methods at run time, instead of in at compile time. This enables you to create objects to work with structures that do not match a static type or format. For example, you can use a dynamic object to reference the HTML Document Object Model (DOM), which can contain any combination of valid HTML markup elements and attributes. Because each HTML document is unique, the members for a particular HTML document are determined at run time. A common method to reference an attribute of an HTML element is to pass the name of the attribute to the **GetProperty** method of the element. To reference the **id** attribute of the HTML element <div id="Div1">, you first obtain a reference to the <div> element, and then use divElement.GetProperty("id"). If you use a dynamic object, you can reference the **id** attribute as divElement.id.

Dynamic objects also provide convenient access to dynamic languages such as IronPython and IronRuby. You can use a dynamic object to refer to a dynamic script that is interpreted at run time.

You reference a dynamic object by using late binding. In C#, you specify the type of a late-bound object as **dynamic**. In Visual Basic, you specify the type of a late-bound object as **Object**. For more information, see [dynamic (C# Reference)](http://msdn.microsoft.com/en-us/library/dd264741.aspx) and [Early and Late Binding (Visual Basic)](http://msdn.microsoft.com/en-us/library/0tcf61s1.aspx).

You can create custom dynamic objects by using the classes in the [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) namespace. For example, you can create an [ExpandoObject](http://msdn.microsoft.com/en-us/library/system.dynamic.expandoobject.aspx) and specify the members of that object at run time. You can also create your own type that inherits the [DynamicObject](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.aspx) class. You can then override the members of the [DynamicObject](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.aspx) class to provide run-time dynamic functionality.

In this walkthrough you will perform the following tasks:

* Create a custom object that dynamically exposes the contents of a text file as properties of an object.
* Create a project that uses an **IronPython** library.

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

You need IronPython 2.6.1 for .NET 4.0 to complete this walkthrough. You can download IronPython 2.6.1 for .NET 4.0 from [CodePlex](http://go.microsoft.com/fwlink/?LinkId=187223).

|  |
| --- |
| **Description: NoteNote** |
| Your computer might show different names or locations for some of the Visual Studio user interface elements in the following instructions. The Visual Studio edition that you have and the settings that you use determine these elements. For more information, see [Visual Studio Settings](http://msdn.microsoft.com/en-us/library/2ewd52wf.aspx). |

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifCreating a Custom Dynamic Object

The first project that you create in this walkthrough defines a custom dynamic object that searches the contents of a text file. Text to search for is specified by the name of a dynamic property. For example, if calling code specifies dynamicFile.Sample, the dynamic class returns a generic list of strings that contains all of the lines from the file that begin with "Sample". The search is case-insensitive. The dynamic class also supports two optional arguments. The first argument is a search option enum value that specifies that the dynamic class should search for matches at the start of the line, the end of the line, or anywhere in the line. The second argument specifies that the dynamic class should trim leading and trailing spaces from each line before searching. For example, if calling code specifies dynamicFile.Sample(StringSearchOption.Contains), the dynamic class searches for "Sample" anywhere in a line. If calling code specifies dynamicFile.Sample(StringSearchOption.StartsWith, false), the dynamic class searches for "Sample" at the start of each line, and does not remove leading and trailing spaces. The default behavior of the dynamic class is to search for a match at the start of each line and to remove leading and trailing spaces.

**To create a custom dynamic class**

1. Start Visual Studio.
2. On the **File** menu, point to **New** and then click **Project**.
3. In the **New Project** dialog box, in the **Project Types** pane, make sure that **Windows** is selected. Select **Console Application** in the **Templates** pane. In the **Name** box, type **DynamicSample**, and then click **OK**. The new project is created.
4. Right-click the DynamicSample project and point to **Add**, and then click **Class**. In the **Name** box, type **ReadOnlyFile**, and then click **OK**. A new file is added that contains the ReadOnlyFile class.
5. At the top of the ReadOnlyFile.cs or ReadOnlyFile.vb file, add the following code to import the [System.IO](http://msdn.microsoft.com/en-us/library/system.io.aspx) and [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) namespaces.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl02_code');" \o "Copy Code)

Imports System.IO

Imports System.Dynamic

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl03_code');" \o "Copy Code)

using System.IO;

using System.Dynamic;

1. The custom dynamic object uses an enum to determine the search criteria. Before the class statement, add the following enum definition.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl04_code');" \o "Copy Code)

Public Enum StringSearchOption

StartsWith

Contains

EndsWith

End Enum

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl05_code');" \o "Copy Code)

public enum StringSearchOption

{

StartsWith,

Contains,

EndsWith

}

1. Update the class statement to inherit the **DynamicObject** class, as shown in the following code example.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl06_code');" \o "Copy Code)

Public Class ReadOnlyFile

Inherits DynamicObject

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl07_code');" \o "Copy Code)

class ReadOnlyFile : DynamicObject

1. Add the following code to the **ReadOnlyFile** class to define a private field for the file path and a constructor for the **ReadOnlyFile** class.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl08_code');" \o "Copy Code)

' Store the path to the file and the initial line count value.

Private p\_filePath As String

' Public constructor. Verify that file exists and store the path in

' the private variable.

Public Sub New(ByVal filePath As String)

If Not File.Exists(filePath) Then

Throw New Exception("File path does not exist.")

End If

p\_filePath = filePath

End Sub

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl09_code');" \o "Copy Code)

// Store the path to the file and the initial line count value.

private string p\_filePath;

// Public constructor. Verify that file exists and store the path in

// the private variable.

public ReadOnlyFile(string filePath)

{

if (!File.Exists(filePath))

{

throw new Exception("File path does not exist.");

}

p\_filePath = filePath;

}

1. Add the following **GetPropertyValue** method to the **ReadOnlyFile** class. The **GetPropertyValue** method takes, as input, search criteria and returns the lines from a text file that match that search criteria. The dynamic methods provided by the **ReadOnlyFile** class call the **GetPropertyValue** method to retrieve their respective results.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl10_code');" \o "Copy Code)

Public Function GetPropertyValue(ByVal propertyName As String,

Optional ByVal StringSearchOption As StringSearchOption = StringSearchOption.StartsWith,

Optional ByVal trimSpaces As Boolean = True) As List(Of String)

Dim sr As StreamReader = Nothing

Dim results As New List(Of String)

Dim line = ""

Dim testLine = ""

Try

sr = New StreamReader(p\_filePath)

While Not sr.EndOfStream

line = sr.ReadLine()

' Perform a case-insensitive search by using the specified search options.

testLine = UCase(line)

If trimSpaces Then testLine = Trim(testLine)

Select Case StringSearchOption

Case StringSearchOption.StartsWith

If testLine.StartsWith(UCase(propertyName)) Then results.Add(line)

Case StringSearchOption.Contains

If testLine.Contains(UCase(propertyName)) Then results.Add(line)

Case StringSearchOption.EndsWith

If testLine.EndsWith(UCase(propertyName)) Then results.Add(line)

End Select

End While

Catch

' Trap any exception that occurs in reading the file and return Nothing.

results = Nothing

Finally

If sr IsNot Nothing Then sr.Close()

End Try

Return results

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl11_code');" \o "Copy Code)

public List<string> GetPropertyValue(string propertyName,

StringSearchOption StringSearchOption = StringSearchOption.StartsWith,

bool trimSpaces = true)

{

StreamReader sr = null;

List<string> results = new List<string>();

string line = "";

string testLine = "";

try

{

sr = new StreamReader(p\_filePath);

while (!sr.EndOfStream)

{

line = sr.ReadLine();

// Perform a case-insensitive search by using the specified search options.

testLine = line.ToUpper();

if (trimSpaces) { testLine = testLine.Trim(); }

switch (StringSearchOption)

{

case StringSearchOption.StartsWith:

if (testLine.StartsWith(propertyName.ToUpper())) { results.Add(line); }

break;

case StringSearchOption.Contains:

if (testLine.Contains(propertyName.ToUpper())) { results.Add(line); }

break;

case StringSearchOption.EndsWith:

if (testLine.EndsWith(propertyName.ToUpper())) { results.Add(line); }

break;

}

}

}

catch

{

// Trap any exception that occurs in reading the file and return null.

results = null;

}

finally

{

if (sr != null) {sr.Close();}

}

return results;

}

1. After the **GetPropertyValue** method, add the following code to override the [TryGetMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.trygetmember.aspx) method of the [DynamicObject](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.aspx) class. The [TryGetMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.trygetmember.aspx) method is called when a member of a dynamic class is requested and no arguments are specified. The **binder** argument contains information about the referenced member, and the **result** argument references the result returned for the specified member. The [TryGetMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.trygetmember.aspx) method returns a Boolean value that returns **true** if the requested member exists; otherwise it returns **false**.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl16_code');" \o "Copy Code)

' Implement the TryGetMember method of the DynamicObject class for dynamic member calls.

Public Overrides Function TryGetMember(ByVal binder As GetMemberBinder,

ByRef result As Object) As Boolean

result = GetPropertyValue(binder.Name)

Return If(result Is Nothing, False, True)

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl17_code');" \o "Copy Code)

// Implement the TryGetMember method of the DynamicObject class for dynamic member calls.

public override bool TryGetMember(GetMemberBinder binder,

out object result)

{

result = GetPropertyValue(binder.Name);

return result == null ? false : true;

}

1. After the **TryGetMember** method, add the following code to override the [TryInvokeMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.tryinvokemember.aspx) method of the [DynamicObject](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.aspx) class. The [TryInvokeMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.tryinvokemember.aspx) method is called when a member of a dynamic class is requested with arguments. The **binder** argument contains information about the referenced member, and the **result** argument references the result returned for the specified member. The **args** argument contains an array of the arguments that are passed to the member. The [TryInvokeMember](http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicobject.tryinvokemember.aspx) method returns a Boolean value that returns **true** if the requested member exists; otherwise it returns **false**.

The custom version of the **TryInvokeMember** method expects the first argument to be a value from the **StringSearchOption** enum that you defined in a previous step. The **TryInvokeMember** method expects the second argument to be a Boolean value. If one or both arguments are valid values, they are passed to the **GetPropertyValue** method to retrieve the results.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl22_code');" \o "Copy Code)

' Implement the TryInvokeMember method of the DynamicObject class for

' dynamic member calls that have arguments.

Public Overrides Function TryInvokeMember(ByVal binder As InvokeMemberBinder,

ByVal args() As Object,

ByRef result As Object) As Boolean

Dim StringSearchOption As StringSearchOption = StringSearchOption.StartsWith

Dim trimSpaces = True

Try

If args.Length > 0 Then StringSearchOption = CType(args(0), StringSearchOption)

Catch

Throw New ArgumentException("StringSearchOption argument must be a StringSearchOption enum value.")

End Try

Try

If args.Length > 1 Then trimSpaces = CType(args(1), Boolean)

Catch

Throw New ArgumentException("trimSpaces argument must be a Boolean value.")

End Try

result = GetPropertyValue(binder.Name, StringSearchOption, trimSpaces)

Return If(result Is Nothing, False, True)

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl23_code');" \o "Copy Code)

// Implement the TryInvokeMember method of the DynamicObject class for

// dynamic member calls that have arguments.

public override bool TryInvokeMember(InvokeMemberBinder binder,

object[] args,

out object result)

{

StringSearchOption StringSearchOption = StringSearchOption.StartsWith;

bool trimSpaces = true;

try

{

if (args.Length > 0) { StringSearchOption = (StringSearchOption)args[0]; }

}

catch

{

throw new ArgumentException("StringSearchOption argument must be a StringSearchOption enum value.");

}

try

{

if (args.Length > 1) { trimSpaces = (bool)args[1]; }

}

catch

{

throw new ArgumentException("trimSpaces argument must be a Boolean value.");

}

result = GetPropertyValue(binder.Name, StringSearchOption, trimSpaces);

return result == null ? false : true;

}

1. Save and close the file.

**To create a sample text file**

1. Right-click the DynamicSample project and point to **Add**, and then click **New Item**. In the **Installed Templates** pane, select **General**, and then select the **Text File** template. Leave the default name of TextFile1.txt in the **Name** box, and then click **Add**. A new text file is added to the project.
2. Copy the following text to the TextFile1.txt file.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl24_code');" \o "Copy Code)

List of customers and suppliers

Supplier: Lucerne Publishing (http://www.lucernepublishing.com/)

Customer: Preston, Chris

Customer: Hines, Patrick

Customer: Cameron, Maria

Supplier: Graphic Design Institute (http://www.graphicdesigninstitute.com/)

Supplier: Fabrikam, Inc. (http://www.fabrikam.com/)

Customer: Seubert, Roxanne

Supplier: Proseware, Inc. (http://www.proseware.com/)

Customer: Adolphi, Stephan

Customer: Koch, Paul

1. Save and close the file.

**To create a sample application that uses the custom dynamic object**

1. In **Solution Explorer**, double-click the Module1.vb file if you are using Visual Basic or the Program.cs file if you are using Visual C#.
2. Add the following code to the Main procedure to create an instance of the **ReadOnlyFile** class for the TextFile1.txt file. The code uses late binding to call dynamic members and retrieve lines of text that contain the string "Customer".

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl25_code');" \o "Copy Code)

Dim rFile As Object = New ReadOnlyFile("..\..\TextFile1.txt")

For Each line In rFile.Customer

Console.WriteLine(line)

Next

Console.WriteLine("----------------------------")

For Each line In rFile.Customer(StringSearchOption.Contains, True)

Console.WriteLine(line)

Next

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl62_ctl00_ctl26_code');" \o "Copy Code)

dynamic rFile = new ReadOnlyFile(@"..\..\TextFile1.txt");

foreach (string line in rFile.Customer)

{

Console.WriteLine(line);

}

Console.WriteLine("----------------------------");

foreach (string line in rFile.Customer(StringSearchOption.Contains, true))

{

Console.WriteLine(line);

}

1. Save the file and press CTRL+F5 to build and run the application.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifCalling a Dynamic Language Library

The next project that you create in this walkthrough accesses a library that is written in the dynamic language IronPython. Before you create this project, you must have IronPython 2.6.1 for .NET 4.0 installed. You can download IronPython 2.6.1 for .NET 4.0 from [CodePlex](http://go.microsoft.com/fwlink/?LinkId=187223).

**To create a custom dynamic class**

1. In Visual Studio, on the **File** menu, point to **New** and then click **Project**.
2. In the **New Project** dialog box, in the **Project Types** pane, make sure that **Windows** is selected. Select **Console Application** in the **Templates** pane. In the **Name** box, type **DynamicIronPythonSample**, and then click **OK**. The new project is created.
3. If you are using Visual Basic, right-click the DynamicIronPythonSample project and then click **Properties**. Click the **References** tab. Click the **Add** button. If you are using Visual C#, in **Solution Explorer**, right-click the **References** folder and then click **Add Reference**.
4. On the **Browse** tab, browse to the folder where the IronPython libraries are installed. For example, C:\Program Files\IronPython 2.6 for .NET 4.0. Select the **IronPython.dll**, **IronPython.Modules.dll**, **Microsoft.Scripting.dll**, and **Microsoft.Dynamic.dll** libraries. Click **OK**.
5. If you are using Visual Basic, edit the Module1.vb file. If you are using Visual C#, edit the Program.cs file.
6. At the top of the file, add the following code to import the **Microsoft.Scripting.Hosting** and **IronPython.Hosting** namespaces from the IronPython libraries.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl01_code');" \o "Copy Code)

Imports Microsoft.Scripting.Hosting

Imports IronPython.Hosting

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl02_code');" \o "Copy Code)

using Microsoft.Scripting.Hosting;

using IronPython.Hosting;

1. In the Main method, add the following code to create a new **Microsoft.Scripting.Hosting.ScriptRuntime** object to host the IronPython libraries. The **ScriptRuntime** object loads the IronPython library module random.py.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl03_code');" \o "Copy Code)

' Set the current directory to the IronPython libraries.

My.Computer.FileSystem.CurrentDirectory =

My.Computer.FileSystem.SpecialDirectories.ProgramFiles &

"\IronPython 2.6 for .NET 4.0\Lib"

' Create an instance of the random.py IronPython library.

Console.WriteLine("Loading random.py")

Dim py = Python.CreateRuntime()

Dim random As Object = py.UseFile("random.py")

Console.WriteLine("random.py loaded.")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl04_code');" \o "Copy Code)

// Set the current directory to the IronPython libraries.

System.IO.Directory.SetCurrentDirectory(

Environment.GetFolderPath(Environment.SpecialFolder.ProgramFiles) +

@"\IronPython 2.6 for .NET 4.0\Lib");

// Create an instance of the random.py IronPython library.

Console.WriteLine("Loading random.py");

ScriptRuntime py = Python.CreateRuntime();

dynamic random = py.UseFile("random.py");

Console.WriteLine("random.py loaded.");

1. After the code to load the random.py module, add the following code to create an array of integers. The array is passed to the **shuffle** method of the random.py module, which randomly sorts the values in the array.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl05_code');" \o "Copy Code)

' Initialize an enumerable set of integers.

Dim items = Enumerable.Range(1, 7).ToArray()

' Randomly shuffle the array of integers by using IronPython.

For i = 0 To 4

random.shuffle(items)

For Each item In items

Console.WriteLine(item)

Next

Console.WriteLine("-------------------")

Next

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl63_ctl00_ctl06_code');" \o "Copy Code)

// Initialize an enumerable set of integers.

int[] items = Enumerable.Range(1, 7).ToArray();

// Randomly shuffle the array of integers by using IronPython.

for (int i = 0; i < 5; i++)

{

random.shuffle(items);

foreach (int item in items)

{

Console.WriteLine(item);

}

Console.WriteLine("-------------------");

}

1. Save the file and press CTRL+F5 to build and run the application.