Important	This document may not represent best practices for current development, links to downloads and other resources may no longer be valid. Current recommended version can be found here.

Walkthrough: Creating and Using an ASP.NET Web Service in Visual Web Developer

Visual Studio 2008

Note:
This topic pertains to a legacy technology. XML Web services and XML Web service clients should now be created using Windows Communication Foundation.

In addition to letting you create Web pages, Microsoft Visual Studio also lets you create Web services that use ASP.NET XML. Creating a Web service in Visual Studio is similar to creating a Web page. You can also use the Microsoft Visual Web Developer Web development tool to reference and use Web services that are in a Visual Web Developer solution, on your local computer or in a local or external UDDI directory. In this walkthrough, you will create the Web service in one solution and use it in another.

Tasks illustrated in this walkthrough include:

- Creating a simple XML Web service in Visual Web Developer.
- Creating a separate Web site that uses the Web service.

Prerequisites

In order to complete this walkthrough, you will need:

- Microsoft Internet Information Services (IIS) installed locally on your computer.

Creating a Web Service Under the IIS Root

Create a new Web service and page by following these steps.

Note:
You must use an IIS Web site for this walkthrough.

To create a Web service

1. Open Visual Web Developer.
2. On the **File** menu, click **New Web Site**.

   The **New Web Site** dialog box appears.
3. Under **Visual Studio installed templates**, click **ASP.NET Web Service**.
4. Click **Browse**.
5. Click **Local IIS**.
6. Click **Default Web Site**.
7. Click **Create New Web Application**.

   Visual Web Developer creates a new IIS Web application.
8. Type the name **TemperatureWebService**.
9. Click **Open**.

   The **New Web Site** dialog box appears, with the name of the new Web site in the rightmost **Location** list. The location includes the protocol (**http://**) and location (**localhost**). This indicates that you are working with a local IIS Web site.
10. In the **Language** list, click the programming language that you prefer to work in.

The programming language that you choose will be the default for the Web site. However, you can use more than one language in the same Web application by creating pages and components in different programming languages. For more information about how to create components using different languages, see Shared Code Folders in ASP.NET Web Sites.

11. Click **OK**.

    Visual Web Developer creates the new Web service and opens a new class named **Service**, which is the default Web service. However, in the following procedure you will create a new Web service with a specified name and you will not use the **Service** class.

12. Close the Service class.

Creating the Web Service

You will create a Web service that converts temperature from Fahrenheit to Celsius and vice versa.

To create the Web service

1. In Solution Explorer, right-click the Web site name (http://localhost/TemperatureWebService), and then click **Add New Item**.
2. Under **Visual Studio installed templates**, click **Web Service**, and then in the **Name** box, type **Convert**.
3. Make sure that the **Place code in separate file** check box is selected, and then click **Add**.

   Visual Web Developer creates a new Web service that is made up of two files. The Convert.asmx file is the file that can be invoked to call Web service methods, and it points to the code for the Web service. The code itself is in a class file (Convert.vb, Convert.cs, or Convert.jsl, depending on the programming language) in the App_Code folder. The code file contains a template for a Web service. The code file includes some code for a Web service method.

You will create two methods in the Web service. The first method converts Fahrenheit temperatures to Celsius, and the second method converts Celsius temperatures to Fahrenheit.

To create the conversion methods

1. Add the following code inside the class, after the **HelloWorld** method:

   C#

```csharp
[System.Web.Services.WebMethod()]
public double FahrenheitToCelsius(double Fahrenheit)
{
    return ((Fahrenheit - 32) * 5) / 9;
}

[System.Web.Services.WebMethod()]
public double CelsiusToFahrenheit(double Celsius)
{
    return ((Celsius * 9) / 5) + 32;
}
```

   Notice that the function names are preceded with an attribute (`[System.Web.Services.WebMethod()]` or `<System.Web.Services.WebMethod()>`) as part of the function declaration.

2. After you have entered the functions, save the file.

Now, you can test the Web service in Visual Web Developer.

To test the Web service

1. In Solution Explorer, click Convert.asmx, and then press CTRL+F5.

   The Web service is invoked and a page appears in the browser that shows the methods that are exposed by the Web service.

2. Click **CelsiusToFahrenheit**, which invokes that method.

   A page appears that prompts you for parameter values for the **CelsiusToFahrenheit** method.

3. In the **Celsius** box, type **100**, and then click **Invoke**.

   A new window appears that displays the XML that the Web service returns when the **CelsiusToFahrenheit** method is invoked. The value **212** appears in the XML.

4. Close the browser that contains the method results.
5. In the original browser, click **Back** to return to the list of methods.
6. Click **FahrenheitToCelsius** and test to make sure that the method is returning the results that you expect.

   If you type **212**, the **FahrenheitToCelsius** method will return **100**.

7. Close the browser.

You have finished creating the Web service; the next step is to use it.

Using the Web Service

Now that you have a Web service, you will create a Web site where you will reference and use the Web service that you created. For the walkthrough, you will create a separate Web site that has a page where you start the Web service methods that you just created.

To create a Web site to use the Web service

1. On the **File** menu, click **New Web Site**.
2. Under **Visual Studio installed templates**, click **ASP.NET Web Site**.
3. Click **Browse**.
4. Click **Local IIS**.
5. Click **Default Web Site**.
6. Click **Create New Web Application**.

   Visual Web Developer creates a new IIS Web application.
7. Type the name **TemperatureWeb**.
8. Click **Open**.
9. In the **Language** list, click the programming language that you prefer to work in.
10. Click **OK**.

    Visual Web Developer creates a new local IIS Web site and a new page named Default.aspx.

Adding the Web Service as a Component

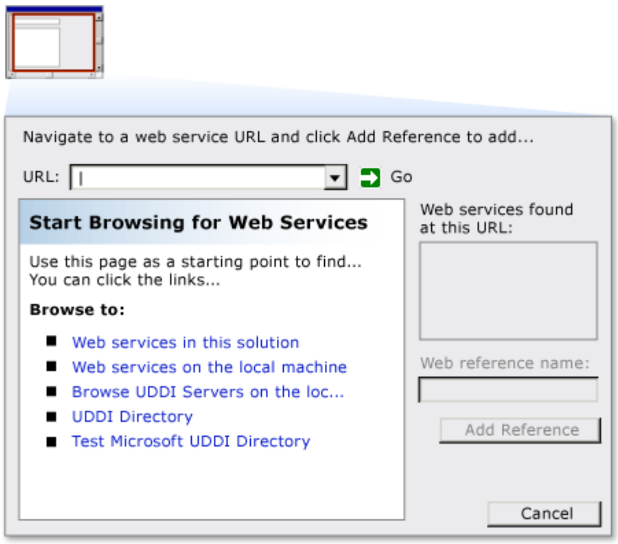
The Web service is a component that you can reference in your application. Therefore, you must create a reference to it.

To create a reference to the Web service

1. On the **WebSite** menu, click **Add Web Reference**.

   The **Add Web Reference** dialog box appears, as shown in the following screen shot.

   Add Web Reference dialog box


Navigate to a web service URL and click Add Reference to add...
URL:
Go
Start Browsing for Web Services
Use this page as a starting point to find...
You can click the links...
Browse to:
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the loc...
UDDI Directory
Test Microsoft UDDI Directory
Web services found at this URL:
Web reference name:
Add Reference
Cancel

2. In the **URL** list, enter the following URL for the Web service, and then click **Go**:

   http://localhost/TemperatureWebService/Convert.asmx

   When Visual Web Developer finds the Web service, information about the Web service appears in the **Add Web References** dialog box.

Note:

If you cannot add a reference to a Web service, it might be that the proxy server is not configured correctly. In Microsoft Internet Explorer, on the **Tools** menu, click **Internet Options**, click **Connections**, and then click **LAN Settings**. Select the **Bypass proxy server for local addresses** check box. Additionally, set the proxy server address to the exact name of the proxy server instead of allowing Internet Explorer to detect the proxy server. For more information, contact the network administrator.

3. Click one of the method links.

   The test page for the method appears.

4. Click **Add Reference**.

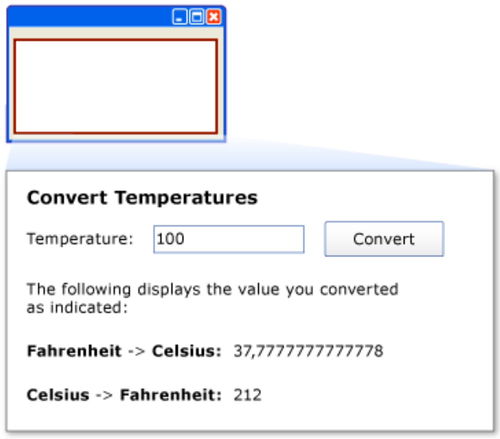
   Visual Web Developer creates an App_WebReferences folder and adds a folder to it for the new Web reference. By default, Web references are assigned a namespace corresponding to their server name (in this case, **localhost**). Make a note of the name for the Web reference namespace. In the folder, Visual Web Developer adds a .wsdl file that references the Web service. It also adds supporting files, such as discovery (.disco and .discomap) files, that include information about where the Web service is located.

Note:

If the server name for the Web service contains characters that cannot be used for a class name, such as a hyphen (-), Visual Web Developer converts those characters to an underscore character (_). Therefore, the namespace in Visual Web Developer for the Web service might not match the server name exactly.

You can now use the Web service. In this walkthrough, you will add controls to Default.aspx, and then program the controls to convert a specified temperature to both Fahrenheit and Celsius. When the page is running, it will look something like the following illustration.

Temperature conversion page


Convert Temperatures
Temperature:
100
Convert
The following displays the value you converted as indicated:
Fahrenheit -> Celsius: 37,7777777777778
Celsius -> Fahrenheit: 212

To call the Web service methods

1. Open the Default.aspx page and switch to Design view.
2. From the **Standard** group in the Toolbox, drag the following controls onto the page and set their properties as indicated:

Control	Properties
Textbox	**ID**: **TemperatureTextbox** **Text**: (empty)
Button	**ID**: **ConvertButton** **Text**: **Convert**
Label	**ID**: **FahrenheitLabel** **Text**: (empty)
Label	**ID**: **CelsiusLabel** **Text**: (empty)

3. Optionally, add text to the page for captions.

   For this walkthrough, the layout of the page is not important.

4. Double-click **ConvertButton** to create an event handler for its **Click** event.
5. Make sure your event handler code matches the code in the following example.

C#

```
protected void ConvertButton_Click(object sender, EventArgs e)
{
    localhost.Convert wsConvert = new localhost.Convert();
    double temperature =
        System.Convert.ToDouble(TemperatureTextbox.Text);
    FahrenheitLabel.Text = "Fahrenheit To Celsius = " +
        wsConvert.FahrenheitToCelsius(temperature).ToString();
    CelsiusLabel.Text = "Celsius To Fahrenheit = " +
        wsConvert.CelsiusToFahrenheit(temperature).ToString();
}
```

6. Press CTRL+F5 to run the page.
7. In the text box, type a value, such as **100,** and then click **Convert**.

   The page displays the result of converting the temperature value into both Fahrenheit and Celsius.

Debugging the Web Service

You can debug a Web service in the same way that you debug Web pages.

Note:

Visual Web Developer Express edition and Visual Studio Standard edition do not support stepping into a Web service from a page that references it. If you are using Visual Web Developer Express edition or Visual Studio Standard edition, skip this section and the ones following. For more information about how to debug Web sites, see Walkthrough: Debugging Web Pages in Visual Web Developer.

To start, you must configure the Web site that contains the Web service to enable debugging.

To enable debugging in the Web services Web site

1. On the **File** menu, click **Open Web Site**.
2. Click **Local IIS**.
3. Click **TemperatureWebService,** and then click **Open**.
4. On the **Website** menu, click **ASP.NET Configuration** to open the Web Site Administration Tool.

   Note:

   If this is the first time that you have run the Web Site Administration Tool, there might be some delay before it appears.

5. Click **Application**, and then click **Application Configuration**.
6. Under **Debugging and Tracing**, click **Configure debugging and tracing**.
7. Select the **Enable debugging check box**.

   The Web Site Administration Tool creates a Web.config file for the Web site and sets a configuration option to enable debugging.

   Note:

   To see the Web.config file in Solution Explorer, click the Web site name and then, on the Solution Explorer toolbar, click **Refresh**.

8. Close the Web Site Administration Tool.

You must now enable debugging for the Web site that uses the Web service.

To enable debugging in the Web site

1. Open the TemperatureWeb site.
2. On the **Website** menu, click **ASP.NET Configuration** to open the Web Site Administration Tool.
3. Click **Application**, click **Application Configuration**, under **Debugging and Tracing**, click **Configure debugging and tracing**, and then select the **Enable debugging check box**.

4. Close the Web Site Administration Tool.

> **Note:**
>
> To see the Web.config file in Solution Explorer, select the Web site name and then, on the Solution Explorer toolbar, click **Refresh**.

5. In Solution Explorer, right-click Default.aspx, and then click **View Code**.

   Visual Web Developer opens the code file for the page.

6. Position the pointer in the following line:

   C#

```
double temperature =
    System.Convert.ToDouble(TemperatureTextbox.Text);
```

7. Press F9 to set a breakpoint on the line.

Testing Debugging

Both the Web site and the Web service are configured for debugging, so that you can now try debugging. You will start in the Default.aspx page and step through the code until the code invokes the Web service. The debugger will switch to the Web service and continue stepping through the code.

To debug the page and Web service

1. Press F5 to run the Default.aspx page with debugging.

   The page appears in the browser.

2. In the box, type a value, such as **100**, and then click **Convert**.

   Visual Web Developer starts running the code for the page, but stops and highlights the line with the breakpoint on it.

3. Press F11 to step to the next line.

4. Press F11 again.

   Because the next line invokes the Web service, the debugger steps into the Web service, stopping on the first line of the **FahrenheitToCelsius** method.

5. Continue pressing F11.

   The debugger steps through the rest of the method, and then returns to the calling page. If you continue stepping, the debugger will step back into the Web service and into the **CelsiusToFahrenheit** method.

6. Close the browser, which also closes the debugger.

Next Steps

This walkthrough has illustrated the basic principles of creating a very simple Web service and using it in an ASP.NET application. You might want to experiment with additional, more complex Web service features. Suggestions for additional exploration include the following:

- Understand the processing that occurs when you make an XML Web service call.

  For more information, see Anatomy of an XML Web Service Lifetime.

- Understand how to use XML Web services.

  For more information, see XML Web Service Scenarios.

- Create XML Web services that offer more sophisticated functionality than the simple Web service from this walkthrough. You can create and chain asynchronous Web service method calls, use transactions, secure the Web services, configure Web services for Microsoft Windows authentication, and so on.

  For more information, see XML Web Services Using ASP.NET.

- Learn about language-specific and technology-specific features provided by the XML Web services.

  For more information, see XML Web Services in Visual FoxPro, Introduction to Programming Web Services in Managed Code, Web Services (How Do I in Visual Basic), XML Web Services Created with ATL Server, and Creating and Accessing Web Services Walkthroughs.

See Also

Tasks

Walkthrough: Debugging Web Pages in Visual Web Developer

Reference

@ WebService Directive in XML Web Services

Other Resources

Guided Tour of Creating Web Sites in Visual Web Developer
<webServices> Element
Design Guidelines for XML Web Services Created Using ASP.NET
Securing XML Web Services Created Using ASP.NET
XML Web Services Using ASP.NET