**ASP.NET XML Web Service Basics**

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/a7xexaft(d=printer,v=vs.80).aspx)

**This topic is specific to a legacy technology. XML Web services and XML Web service clients should now be created using** [Windows Communication Foundation](http://go.microsoft.com/fwlink/?LinkID=127777) .

Since ASP.NET provides the infrastructure for the inner workings of a Web service, developers can focus on implementing the functionality of their specific Web service. Enabling a Web service using ASP.NET entails creating a file with an .asmx file name extension, declaring a Web service in that file and possibly another file, and defining Web service methods. The procedures are listed in [Walkthrough: Building a Basic XML Web Service Using ASP.NET](http://msdn.microsoft.com/en-us/library/7hs6sw69(v=vs.100).aspx) and are elaborated upon here.

**Declaration of Web Services**

When you create a Web service in ASP.NET, you place the required [@ WebService](http://msdn.microsoft.com/en-us/library/yt16sxzt(v=vs.100).aspx) directive at the top of a text file with an .asmx file name extension. The presence of the .asmx file and the **@ WebService** directive correlate the URL address of the Web service with its implementation. You also implement the Web service class that defines the methods and data types visible by Web service clients.

The Web service class you define can be included directly in the .asmx file, or in a separate file. If you use a separate file, it must be compiled into an assembly. Optionally, you can apply a **WebService** attribute to the class that implements the Web service. The class that implements the Web service can derive from the **WebService** class.

By applying the optional [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webserviceattribute(v=vs.100).aspx) attribute to a class that implements a Web service, you can set the default XML namespace for the Web service along with a string to describe the Web service. It is highly recommended that this default namespace, which originally is http://tempuri.org, be changed before the Web service is made publicly consumable. This is important because the Web service must be distinguished from other Web services that might inadvertently use the namespace as the default (<http://tempuri.org/>).

Classes that implement a Web service created using ASP.NET can optionally derive from the [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webservice(v=vs.100).aspx) class to gain access to the common ASP.NET objects, such as **Application**, **Session**, **User**, and **Context**. The **Application** and **Session** properties provide access to storing and receiving state across the lifetime of the Web application or a particular session. For more information on state management, see [How to: Manage State in Web Services Created Using ASP.NET](http://msdn.microsoft.com/en-us/library/hk34sw2t(v=vs.100).aspx). The **User** property contains the identity of the caller, if authentication is enabled, for the Web service. With the identity, a Web service can determine whether the request is authorized. For more information on authentication, see [Securing XML Web Services](http://msdn.microsoft.com/en-us/library/w67h0dw7(v=vs.100).aspx). The **Context** property provides access to all HTTP-specific information about the Web service client's request. For more information on the **Context** property, see [WebService.Context Property](http://msdn.microsoft.com/en-us/library/system.web.services.webservice.context(v=vs.100).aspx).

**Definition of Web Service Methods**

Methods of a class that implement a Web service do not automatically have the ability to receive Web service requests and send back responses, but with Web services created using ASP.NET, it is very simple to add that capability. Apply a [WebMethod](http://msdn.microsoft.com/en-us/library/system.web.services.webmethodattribute(v=vs.100).aspx) attribute to public methods. Methods of a Web service class that can be communicated with over the Web are called Web service methods.

Web service methods are a key part of the messaging infrastructure employed by Web services. That is, a client and a Web service communicate using messages, specifically SOAP messages, by default. Clients send a SOAP request to a Web service and a Web service method typically returns a SOAP response. Web services define the type of messages they accept using operations, as defined by Web Services Description Language (WSDL). These operations correlate to each of the Web service methods within a Web service. Even though each of these Web service methods are defined in ASP.NET using a method of a class, it is important to realize that the data that is eventually communicated over the network must be serialized into XML. As such, it is important to remember that Web services are not a replacement for DCOM, but rather a messaging infrastructure for communicating across platforms using industry standards.

# Walkthrough: Building a Basic XML Web Service Using ASP.NET

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [.NET Framework 1.1](http://msdn.microsoft.com/en-us/library/7hs6sw69(d=printer,v=vs.71).aspx)
* [.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/7hs6sw69(d=printer,v=vs.80).aspx)

**This topic is specific to a legacy technology. XML Web services and XML Web service clients should now be created using** [Windows Communication Foundation](http://go.microsoft.com/fwlink/?LinkID=127777) .

Developing an XML Web service using ASP.NET starts with the following steps:

1. Create a file with an .asmx file name extension and declare a Web service in it using an @WebService directive
2. Create a class that implements the Web service. The class can optionally derive from the [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webservice(v=vs.100).aspx) class.
3. Optionally, apply the [WebServiceAttribute](http://msdn.microsoft.com/en-us/library/system.web.services.webserviceattribute(v=vs.100).aspx) attribute to the class implementing the Web service.
4. Define the Web service methods that compose the functionality of the Web service.

[Declaring a Web Service](javascript:void(0))

When you create an XML Web service in ASP.NET, you place the required [@ WebService](http://msdn.microsoft.com/en-us/library/yt16sxzt(v=vs.100).aspx) directive at the top of a text file with an .asmx file name extension. The presence of the .asmx file and the **@ WebService** directive correlate the URL address of the XML Web service with its implementation. Next, you implement the XML Web service class that defines the methods and data types visible by XML Web service clients. Finally, you add your XML Web service logic to those methods in order to process XML Web service requests and send back responses. The XML Web service class you define can be included directly in the .asmx file, or in a separate file. If you use a separate file, it must be compiled into an assembly. Optionally, you can apply a **WebService** attribute to the class implementing the XML Web service. The class that implements the XML Web service can derive from the **WebService** class.

#### To declare a Web service whose implementation resides in the same file

1. Add an [@ WebService](http://msdn.microsoft.com/en-us/library/yt16sxzt(v=vs.100).aspx) directive to the top of a file with an .asmx file name extension, specifying the class that implements the Web service and the programming language that is used in the implementation.

The **Class** attribute can be set to a class that resides in the same assembly as the **@ WebService** directive or to a class within a separate assembly. If the class resides in a separate assembly, it must be placed in the \Bin directory under the Web application where the Web service resides. The **Language** attribute can be set to **C#**, **VB**, and **JS**, which refer to C#, Visual Basic .NET, and JScript .NET, respectively.

The following code example sets the **Language** attribute of the **@ WebService** directive, and sets the **Class** attribute to Util, which resides in the same file.

C#

<%@ WebService Language="C#" Class="Util" %>

VB

<%@ WebService Language="VB" Class="Util" %>

#### To declare a Web service whose implementation resides in an assembly

1. Add an [@ WebService](http://msdn.microsoft.com/en-us/library/yt16sxzt(v=vs.100).aspx) directive to the top of a file with an .asmx extension, specifying the class that implements the Web service, the assembly that contains the implementation, and the programming language that is used in the implementation. If you use a separate file, it must be compiled into an assembly.

The following **@ WebService** directive is the only line in a file with an .asmx extension, specifying that the MyName.MyWebService class resides in the MyAssembly assembly within the \Bin directory of the Web application that is hosting the Web service.

C#

<%@ WebService Language="C#" Class="MyName.MyWebService,MyAssembly" %>

VB

<%@ WebService Language="VB" Class="MyName.MyWebService,MyAssembly" %>

|  |
| --- |
| **7hs6sw69.note(en-us,VS.100).gifNote:** |
| If you do not specify an assembly within the **@ WebService** directive, then ASP.NET searches through the list of assemblies in the \Bin directory of the Web application that is hosting the Web service the first time the Web service is accessed. Therefore, you will improve performance on the first access by providing the assembly name. |

[Deriving from the WebService Class](javascript:void(0))

Classes that implement a Web service that was created using ASP.NET can optionally derive from the [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webservice(v=vs.100).aspx) class to gain access to the common ASP.NET objects, such as [Application](http://msdn.microsoft.com/en-us/library/system.web.services.webservice.application(v=vs.100).aspx), [Session](http://msdn.microsoft.com/en-us/library/system.web.services.webservice.session(v=vs.100).aspx), [User](http://msdn.microsoft.com/en-us/library/system.web.services.webservice.user(v=vs.100).aspx), and [Context](http://msdn.microsoft.com/en-us/library/system.web.services.webservice.context(v=vs.100).aspx).

#### To derive from the WebService class and access common ASP.NET objects

C#

[VB](http://msdn.microsoft.com/en-us/library/7hs6sw69(d=printer,v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

<%@ WebService Language="C#" Class="Util" %>

using System;

using System.Web.Services;

public class Util: WebService

[Applying the WebService Attribute](javascript:void(0))

Apply the optional [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webserviceattribute(v=vs.100).aspx) attribute to a class that implements a Web service to set the default XML namespace for the Web service, which originally is http://tempuri.org, along with a string to describe the Web service.

It is highly recommended that this default namespace, which is http://tempuri.org, be changed before the XML Web service is made publicly consumable. This is important because the XML Web service must be distinguished from other XML Web services that might inadvertently use the namespace as the default (<http://tempuri.org/>).

#### To set the XML namespace of which a Web service is a member

1. Apply a [WebService](http://msdn.microsoft.com/en-us/library/system.web.services.webserviceattribute(v=vs.100).aspx) attribute to the class that is implementing the Web service, setting the **Namespace** property.

The following code example sets the XML namespace to http://www.contoso.com/.

C#

[VB](http://msdn.microsoft.com/en-us/library/7hs6sw69(d=printer,v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

<%@ WebService Language="C#" Class="Util" Debug=true%>

using System.Web.Services;

using System;

[WebService(Namespace="http://www.contoso.com/")]

public class Util: WebService

[Defining Web Service Methods](javascript:void(0))

Methods of a class that implements a Web service do not automatically have the ability to be communicated with over the Web, but with Web services created using ASP.NET, it is very simple to add that capability. To add this capability, apply a [WebMethod](http://msdn.microsoft.com/en-us/library/system.web.services.webmethodattribute(v=vs.100).aspx) attribute to public methods. Methods of a Web service that can be communicated with over the Web are called Web service methods.

#### To declare a Web service method

1. Add public methods to the class that is implementing the Web service.
2. Apply the **WebMethod** attribute to the public methods that you want to be mapped to Web service operations.

The following code example has two public methods, one of which is a Web service method. The Multiply method is a Web service method, because it has a **WebMethod** attribute applied to it.

C#

[VB](http://msdn.microsoft.com/en-us/library/7hs6sw69(d=printer,v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

<%@ WebService Language="C#" Class="Util" %>

using System.Web.Services;

using System;

[WebService(Namespace="http://www.contoso.com/")]

public class Util: WebService

{

[ WebMethod]

public long Multiply(int a, int b)

{

return a \* b;

}

}