**Using Web Services in ASP.NET AJAX**

**Visual Studio 2008**

This topic describes how to access Web services from client script in AJAX-enabled ASP.NET Web pages. The services can be either custom services that you create, or built-in application services. The application services are provided as part of ASP.NET AJAX, and include authentication, roles, and profile services.

Custom Web services can be in the form of .ASP.NET Web services (.asmx services), or Windows Communication Foundation (WCF) services (.svc services).

This topic contains the following information:

* [Scenarios](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#Scenarios)
* [Background](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#Background)
* [Examples](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#Examples)
* [Additional Resources](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#AdditionalResources)
* [Class Reference](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#ClassReference)
* [What's New](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#WhatsNew)

[Scenarios](javascript:void(0))

You use WCF and ASP.NET in the following cases:

* If you have already created WCF services, you can add endpoints to enable script in AJAX-enabled Web pages to access the services. For more information, see [Exposing WCF Services to Client Script](http://msdn.microsoft.com/en-us/library/bb514961(v=vs.90).aspx).
* If you have already created ASP.NET Web (.asmx) services, you can modify them to enable script in AJAX-enabled Web pages to access the same service. For more information, see [Exposing Web Services to Client Script](http://msdn.microsoft.com/en-us/library/bb398998(v=vs.90).aspx).
* If you want to create a custom Web service that will be accessed from ASP.NET AJAX Web pages, you can implement it as a WCF service or as an ASP.NET Web service (.asmx file).
* You can use the built-in ASP.NET application services to access users' authentication, roles, and profile information from client script that runs in an AJAX-enabled Web page. For more information, see [Using Forms Authentication with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398896(v=vs.90).aspx).

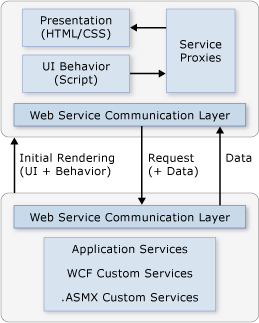
[Background](javascript:void(0))

ASP.NET enables you to create Web services can be accessed from client script in Web pages. The pages communicate with the server through a Web service communication layer that uses AJAX technology to make Web service calls. Data is exchanged asynchronously between client and server, typically in JSON format.

**Client-Server Communication for AJAX Clients**

In AJAX-enabled Web pages, the browser makes an initial request to the server for the page, and then makes subsequent asynchronous requests to Web services for data. The client communication elements are in the form of downloaded proxy classes from the server and the core client-script library. The server communication elements are handlers and custom services. The following illustration shows the elements that are involved in the communication between the client and the server.

Client Server Communication



**AJAX Client Architecture**

Browsers call Web service methods by using proxy classes. A proxy class is a script that is automatically generated by the server and downloaded to the browser at page-load time. The proxy class provides a client object that represents the exposed methods of a Web service.

To call a Web service method, client script calls the corresponding methods of the proxy class. The calls are made asynchronously, through the [XMLHTTP](http://go.microsoft.com/fwlink/?LinkId=75121) object.

The Web service communication layer contains the library script types that enable the proxy classes to make service calls. For more information, see the classes that are contained in the [Sys.Net](http://msdn.microsoft.com/en-us/library/bb310860(v=vs.90).aspx) namespace.

The code in the proxy classes and in the core Web service communication layer hides the complexity of [XMLHTTP](http://go.microsoft.com/fwlink/?LinkId=75121) and the differences between browsers. This simplifies the client script that is required to call the Web service.

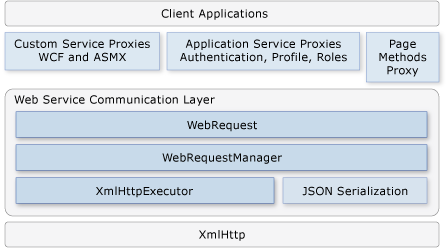
There are two approaches to making a Web service request:

* Call Web services by using the HTTP POST verb. A POST request has a body that contains the data that the browser sends to the server. It does not have a size limitation. Therefore, you can use a POST request when the size of the data exceeds the intrinsic size limitation for a GET request. The client serializes the request into JSON format and sends it as POST data to the server. The server deserializes the JSON data into .NET Framework types and makes the actual Web service call. During the response, the server serializes the return values and passes them back to the client, which deserializes them into JavaScript objects for processing.
* Call Web services by using the HTTP GET verb. This resembles the functionality of a POST request, with the following differences:
  + The client uses a query string to send the parameters to the server.
  + A GET request can call only a Web service method that is configured by using the [ScriptMethodAttribute](http://msdn.microsoft.com/en-us/library/system.web.script.services.scriptmethodattribute(v=vs.90).aspx) attribute.
  + Data size is limited to the URL length allowed by the browser.

|  |
| --- |
| **NoteNote:** |
| GET requests should be avoided for method calls that modify data on the server or that expose critical information. In GET requests, the message is encoded by the browser into the URL and is therefore an easier target for tampering. For both GET and POST requests, you should follow security guidelines to protect sensitive data. |

The following illustration shows the ASP.NET AJAX client architecture.

AJAX Client Architecture



Elements of the client architecture include the Web service communication layer in the core library and the downloaded proxy classes for the services that are used on the page. Individual elements shown in the figure are as follows:

* Custom Service Proxy Classes. These consist of client script that is automatically generated by the server and downloaded to the browser. The proxy classes provide an object for each WCF or ASMX service that is used in the page. (That is, they provide an object for each item in the ServiceReferences element of the [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager(v=vs.90).aspx) control in the page.) Calling a proxy method in client script creates an asynchronous request to the corresponding Web service method on the server.
* Authentication Proxy Class. The [AuthenticationService](http://msdn.microsoft.com/en-us/library/bb310861(v=vs.90).aspx) proxy class is generated by the server authentication application service. It enables the user to log on or log out through JavaScript in the browser without making a round trip to the server.
* Role Proxy Class. The [RoleService](http://msdn.microsoft.com/en-us/library/bb513880(v=vs.90).aspx) proxy class is generated by the server roles application service. It enables you to group users and treat each group as a unit through JavaScript without making round trips to the server. This can be useful to enable or deny access to resources on the server.
* Profile Proxy Class. The [ProfileService](http://msdn.microsoft.com/en-us/library/bb383800(v=vs.90).aspx) class is generated by the server profile application service. It makes the current user's profile information available to the client through JavaScript without making round trips to the server.
* Page Methods Proxy Class. This provides the scripting infrastructure for client script to call static methods in an ASP.NET page as if they were Web service methods. For more information, see [Calling Web Services from Client Script](http://msdn.microsoft.com/en-us/library/bb398995(v=vs.90).aspx).
* Web Service Communication Layer. This is the library that contains the client script types. These types enable the browser (client) to communicate with the services on the server. They also shield client applications from the complexities of establishing and maintaining the asynchronous communication between client and server. They encapsulate the browser **XMLHTTP** object that provides the asynchronous capability, and enable client applications to be browser independent. The following are the main elements of the Web service communication layer:
  + WebRequest. This provides client script functionality to make a Web request. For more information, see the [WebRequest](http://msdn.microsoft.com/en-us/library/bb310979(v=vs.90).aspx) class.
  + WebRequestManager. This manages the flow of the Web requests issued by the [WebRequest](http://msdn.microsoft.com/en-us/library/bb310979(v=vs.90).aspx) object to the associated executor object. For more information, see the [WebRequestManager](http://msdn.microsoft.com/en-us/library/bb397435(v=vs.90).aspx) class.
  + XmlHttpExecutor. This makes asynchronous network requests by using the browser's **XMLHTTP** support. For more information, see the [XmlHttpExecutor](http://msdn.microsoft.com/en-us/library/bb397459(v=vs.90).aspx) class.
  + JSON Serialization. This serializes JavaScript objects into JSON format. Deserialization is available by using the JavaScript eval function. For more information, see the [JavaScriptSerializer](http://msdn.microsoft.com/en-us/library/bb310857(v=vs.90).aspx) class.

The default serialization format is JSON, but individual methods in Web services and in ASP.NET Web pages can return alternative formats such as XML. The serialization format of a method can be specified with attributes. For example, for an ASMX service, you can set the [ScriptMethodAttribute](http://msdn.microsoft.com/en-us/library/system.web.script.services.scriptmethodattribute(v=vs.90).aspx) attribute to cause a Web service method to return XML data, as shown in the following example:

C#

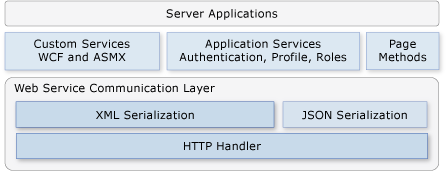
[VB](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

[ScriptMethod(ResponseFormat.Xml)]

**AJAX Server Architecture**

The following illustration shows the AJAX server architecture, which includes the elements that enable communication with client applications.

AJAX Server Architecture



Elements of the server architecture include the Web service communication layer with a HTTP handler and serialization classes, custom services, page methods, and application services. Individual elements shown in the figure are as follows:

* Custom Web Services. These provide the service functionality that you implement and return the appropriate response to the client. Custom Web services can be ASP.NET or WCF services. The Web service communication layer automatically generates client-script proxy classes that can be called asynchronously from client script.
* Page Methods. This component enables a method in an ASP.NET page to be called as if it were a Web service method. Page methods must be defined in the page that is performing the page-method call.
* Authentication Service. The authentication service generates an authentication proxy class that enables the user to log on or log out through client JavaScript. This application service is always available and you do not have to instantiate it. For more information, see [Using Forms Authentication with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398896(v=vs.90).aspx).
* Role Service. The role service generates a role proxy class that enables client JavaScript to access roles information for the currently authenticated user. This application service is always available and you do not have to instantiate it. For more information, see [Using Roles Information with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb514828(v=vs.90).aspx).
* Profile Service. The profile service generates a profile proxy class that enables client JavaScript to get and set profile properties for the user that is associated with the current request. This application service is always available and you do not have to instantiate it. For more information, see [Using Profile Information with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398816(v=vs.90).aspx).
* JSON Serialization. The server JSON serialization component enables customizable serialization and deserialization of common .NET Framework types to and from JSON format. For more information, see [JavaScriptSerializer](http://msdn.microsoft.com/en-us/library/system.web.script.serialization.javascriptserializer(v=vs.90).aspx).
* XML Serialization. The Web service communication layer supports XML serialization for SOAP requests to Web services and for returning XML types from a JSON request to a Web service.

[Examples](javascript:void(0))

The following examples show how to call ASP.NET and WCF services from client script. Examples of how to call application services from client script are provided in other sections of the documentation. The related links are provided later in this topic.

**Calling Web Service Methods in AJAX**

The .NET Framework enables you to call ASP.NET Web services (.asmx) methods from the browser asynchronously by using client script. The page can call server-based methods without a postback and without refreshing the whole page, because only data is transferred between the browser and the server.

This following example shows how to expose a Web service method in an ASP.NET Web page.

C#

[VB](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">

<style type="text/css">

body { font: 11pt Trebuchet MS;

font-color: #000000;

padding-top: 72px;

text-align: center }

.text { font: 8pt Trebuchet MS }

</style>

<title>Simple Web Service</title>

<script type="text/javascript">

// This function calls the Web Service method.

function GetServerTime()

{

Samples.AspNet.ServerTime.GetServerTime(OnSucceeded);

}

// This is the callback function that

// processes the Web Service return value.

function OnSucceeded(result)

{

var RsltElem = document.getElementById("Results");

RsltElem.innerHTML = result;

}

</script>

</head>

<body>

<form id="Form1" runat="server">

<asp:ScriptManager runat="server" ID="scriptManager">

<Services>

<asp:ServiceReference path="ServerTime.asmx" />

</Services>

</asp:ScriptManager>

<div>

<h2>Server Time</h2>

<p>Calling a service that returns the current server time.</p>

<input id="EchoButton" type="button"

value="GetTime" onclick="GetServerTime()" />

</div>

</form>

<hr/>

<div>

<span id="Results"></span>

</div>

</body>

</html>

The following example shows a Web page and the related Web service called by the page script.

C#

[VB](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

<%@ WebService Language="C#" Class="Samples.AspNet.ServerTime" %>

using System;

using System.Web;

using System.Web.Services;

using System.Xml;

using System.Web.Services.Protocols;

using System.Web.Script.Services;

namespace Samples.AspNet

{

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[ScriptService]

public class ServerTime : System.Web.Services.WebService

{

[WebMethod]

public string GetServerTime()

{

return String.Format("The server time is {0}.",

DateTime.Now);

}

}

}

**Making HTTP Requests from an AJAX Client**

The previous example shows how to call Web services from client script by calling the automatically generated proxy classes for the Web service. You can also make lower-level calls to Web services from client script. You might do this if you have to manage the communication layer or examine the data that is being sent to or from the server. To call Web services in this manner, you use the [WebRequest](http://msdn.microsoft.com/en-us/library/bb310979(v=vs.90).aspx) class.

The following example shows how to use a **WebRequest** object to implement GET and POST Web requests that connect to the specified URLs (HTTP end points).

**Calling WCF Service Operations in AJAX**

You can call Windows Communication Foundation (WCF) services (.svc) asynchronously from client script basically like you call .asmx-based services. This following example shows how to expose and call WCF service operations in an ASP.NET Web page.

C#

[VB](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

<%@ Page Language="C#" AutoEventWireup="true"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<style type="text/css">

body { font: 11pt Trebuchet MS;

font-color: #000000;

padding-top: 72px;

text-align: center }

.text { font: 8pt Trebuchet MS }

</style>

<title>Simple WCF Service Page</title>

</head>

<body>

<form id="form1" runat="server">

<asp:ScriptManager ID="ScriptManager1" runat="server">

<Services>

<asp:ServiceReference

Path="SimpleService.svc/ws"/>

</Services>

<Scripts>

<asp:ScriptReference Path="service.js" />

</Scripts>

</asp:ScriptManager>

<div>

<h2>Simple WCF Service</h2>

<input type='button' name="clickme"  value="Greetings"

onclick="javascript:OnClick()" /> &nbsp; &nbsp;

<input type='button' name="clickme2"  value="Greetings2"

onclick="javascript:OnClick2()" />

<hr/>

<div>

<span id="Results"></span>

</div>

</div>

</form>

</body>

</html>

C#

[VB](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

using System;

using System.Web;

using System.Collections;

using System.Collections.Generic;

using System.Threading;

using System.Xml;

using System.Xml.Serialization;

using System.Text;

using System.IO;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Description;

using System.ServiceModel.Dispatcher;

using System.ServiceModel.Channels;

using System.ServiceModel.Activation;

// This a WCF service which consists of a contract,

// defined below as ISimpleService, and DataContractType,

// a class which implements that interface, see SimpleService,

// and configuration entries that specify behaviors associated with

// that implementation (see <system.serviceModel> in web.config)

namespace Aspnet.Samples

{

[ServiceContract()]

public interface ISimpleService

{

[OperationContract]

string HelloWorld1(string value1);

[OperationContract]

string HelloWorld2(DataContractType dataContractValue1);

}

[ServiceBehavior(IncludeExceptionDetailInFaults = true)]

[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]

public class SimpleService : ISimpleService

{

public SimpleService()

{ }

public string HelloWorld1(string value1)

{

return "Hello " + value1;

}

public string HelloWorld2(DataContractType dataContractValue1)

{

return "Hello " + dataContractValue1.FirstName +

" " + dataContractValue1.LastName;

}

}

[DataContract]

public class DataContractType

{

string firstName;

string lastName;

[DataMember]

public string FirstName

{

get { return firstName; }

set { firstName = value; }

}

[DataMember]

public string LastName

{

get { return lastName; }

set { lastName = value; }

}

}

}

**Additional Examples**

* [Calling Web Services from Client Script](http://msdn.microsoft.com/en-us/library/bb398995(v=vs.90).aspx)
* [Walkthrough: Creating and Using AJAX-Enabled Web Service](http://msdn.microsoft.com/en-us/library/bb532367(v=vs.90).aspx)
* [Using Forms Authentication with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398896(v=vs.90).aspx)
* [Using Profile Information with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb398816(v=vs.90).aspx)
* [Using Roles Information with ASP.NET AJAX](http://msdn.microsoft.com/en-us/library/bb514828(v=vs.90).aspx)

[Back to top](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#Introduction)

[Class Reference](javascript:void(0))

The following tables list the main classes that are associated with Web services that can be called from client script.

**Client Namespaces**

|  |  |
| --- | --- |
| **Name** | **Description** |
| [Sys.Net Namespace](http://msdn.microsoft.com/en-us/library/bb310860(v=vs.90).aspx) | Contains classes that manage communication between ASP.NET AJAX client applications and Web services on the server. The **Sys.Net** namespace is part of the Microsoft AJAX Library. |
| [Sys.Serialization Namespace](http://msdn.microsoft.com/en-us/library/bb310840(v=vs.90).aspx) | Contains classes that are related to data serialization for ASP.NET AJAX client applications. |
| [Sys.Services Namespace](http://msdn.microsoft.com/en-us/library/bb311017(v=vs.90).aspx) | Contains types that provide script access in ASP.NET AJAX client applications to the ASP.NET authentication service, profile service, and other application services. The **Sys.Services** namespace is part of the Microsoft AJAX Library. |

**Server Namespaces**

|  |  |
| --- | --- |
| **Name** | **Description** |
| [System.Web.Script.Serialization](http://msdn.microsoft.com/en-us/library/system.web.script.serialization(v=vs.90).aspx) | Contains classes that provide JavaScript Object Notation (JSON) serialization and deserialization for managed types. It also provides extensibility features to customize serialization behavior. |

[Back to top](http://msdn.microsoft.com/en-us/library/bb515101(d=printer,v=vs.90).aspx#Introduction)

[Additional Resources](javascript:void(0))

[About Native XMLHTTP](http://go.microsoft.com/fwlink/?LinkId=75121)

[http://www.json.org](http://go.microsoft.com/fwlink/?LinkId=77297)

[What is the Windows Communication Foundation?](http://go.microsoft.com/fwlink/?LinkId=91610)

[XML Web Services Infrastructure](http://go.microsoft.com/fwlink/?LinkId=91613)

[WCF Services and ASP.NET](http://go.microsoft.com/fwlink/?LinkId=91971)

[Understanding Service-Oriented Architecture](http://go.microsoft.com/fwlink/?LinkId=91611)

[What's New](javascript:void(0))

The following are new features in ASP.NET version 3.0:

* Application services in Windows Communication Foundation (WCF).
* Application services in AJAX called from a .NET Framework client application.

**Walkthrough: Creating and Using AJAX-Enabled Web Service**

**Visual Studio 2008**

Visual Studio 2008 and Microsoft Visual Web Developer Express let you create ASP.NET custom Web services (.asmx files) that you can access from client script. In this walkthrough, you will create an AJAX-enabled Web service and call its methods by using client script.

When you create ASP.NET Web services and expose them to client script, ASP.NET automatically creates JavaScript proxy classes for the Web services. You can call a Web service method by calling the corresponding method of the JavaScript proxy class.

This walkthrough shows:

* How to create an AJAX-enabled Web service in Visual Studio 2008 or Microsoft Visual Web Developer Express.

[Prerequisites](javascript:void(0))

In order to complete this walkthrough, you need:

* Visual Studio 2008 or Microsoft Visual Web Developer Express.
* Microsoft Internet Information Services (IIS) installed locally on your computer. For more information, see [Installing IIS 7.0 on Windows Vista](http://go.microsoft.com/fwlink/?LinkId=125895) on the IIS Web site.

[Creating an AJAX-enabled Web Service](javascript:void(0))

This section describes how to create an AJAX-enabled Web service.

|  |
| --- |
| **NoteNote:** |
| You must use an IIS Web site for this walkthrough. |

**To create an AJAX-enabled Web service**

1. Open Visual Studio 2008. or Microsoft Visual Web Developer Express.
2. On the File menu, click New Web Site.

The New Web Site dialog box appears.

1. Under Visual Studio installed templates, click ASP.NET Web Site.
2. Click Browse.
3. Click Local IIS.
4. Click Default Web Site.
5. In the upper-right corner, click the Create New Web Application icon.

Visual Studio creates a new IIS Web application.

1. Specify the name HelloWorldService.

|  |
| --- |
| **NoteNote:** |
| The name of the Web site that you are creating is not important. |

1. Click Open.

The New Web Site dialog box is displayed, 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.

1. In the Language list, select the programming language that you prefer to work in.
2. Click OK.

[Writing the Custom Web Service](javascript:void(0))

This section describes how to write an AJAX-enabled Web service that provides a method that returns the string "Hello World" and the current server time.

**To write the custom Web service**

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

Visual Studio 2008 creates a new Web service that consists of two files. The HelloWorld.asmx file is the file that can be invoked to call Web service methods. It points to the Web service code. The code itself is in a class file (HelloWorld.vb or HelloWorld.cs, depending on the programming language) in the App\_Code folder. The code file contains a template for a Web service.

1. Delete the existing code in the class and replace it with the following code:

C#

[VB](http://msdn.microsoft.com/en-us/library/bb532367(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

using System;

using System.Web;

using System.Collections;

using System.Web.Services;

using System.Web.Services.Protocols;

namespace Samples.Aspnet

{

[WebService(Namespace="http://mycompany.org")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

// The following attribute allows the service to

// be called from script using ASP.NET AJAX.

[System.Web.Script.Services.ScriptService]

public class HelloWorld : System.Web.Services.WebService

{

public HelloWorld()

{

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string Greetings()

{

string serverTime =

String.Format("Current date and time: {0}.", DateTime.Now);

string greetings = "Hello World! <br/>" + serverTime;

return greetings;

}

}

}

Notice that the function names are preceded with the [WebMethodAttribute](http://msdn.microsoft.com/en-us/library/system.web.services.webmethodattribute(v=vs.90).aspx) attribute as part of the function declaration. In addition, the HelloWorld class is qualified with the [ScriptServiceAttribute](http://msdn.microsoft.com/en-us/library/system.web.script.services.scriptserviceattribute(v=vs.90).aspx) attribute.

These attributes enable the Web service to be called from script in AJAX-enabled ASP.NET Web pages.

1. Save the file and close it.
2. Open the HelloWorld.asmx file and add the following directive:

C#

[VB](http://msdn.microsoft.com/en-us/library/bb532367(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

<%@ WebService Language="C#" CodeBehind="~/App\_Code/HelloWorld.cs" Class="Samples.Aspnet.HelloWorld" %>

1. Save the file and close it.

Now you can test the Web service in your browser. This test does not use script to call the Web service methods. This is only to verify that the Web service is working correctly.

**To test the Web service**

1. Open the browser and enter the following URL: http://localhost/HelloWorldService/HelloWorld.asmx

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

1. Click Greetings. A page appears with the Invoke button.
2. Click the Invoke button to call the method and see the returned value in XML format.
3. Close the browser.

You have finished creating the AJAX-enabled Web service.

[Using the Web Service](javascript:void(0))

The next step is to call the Web service from client script.

**To call the Web service from client script**

1. In Solution Explorer, right-click the Web site name (http://localhost/HelloWorldService), and then click Add New Item.
2. Under Visual Studio installed templates, click JScript File, and then in the Name box, type HelloWorld.js.
3. Click OK.
4. Add the following code to the script file:

PH

var helloWorldProxy;

// Initializes global and proxy default variables.

function pageLoad()

{

// Instantiate the service proxy.

helloWorldProxy = new Samples.Aspnet.HelloWorld();

// Set the default call back functions.

helloWorldProxy.set\_defaultSucceededCallback(SucceededCallback);

helloWorldProxy.set\_defaultFailedCallback(FailedCallback);

}

// Processes the button click and calls

// the service Greetings method.

function OnClickGreetings()

{

var greetings = helloWorldProxy.Greetings();

}

// Callback function that

// processes the service return value.

function SucceededCallback(result)

{

var RsltElem = document.getElementById("Results");

RsltElem.innerHTML = result;

}

// Callback function invoked when a call to

// the service methods fails.

function FailedCallback(error, userContext, methodName)

{

if(error !== null)

{

var RsltElem = document.getElementById("Results");

RsltElem.innerHTML = "An error occurred: " +

error.get\_message();

}

}

if (typeof(Sys) !== "undefined") Sys.Application.notifyScriptLoaded();

This client script will be used by the Default.aspx page to call the HelloWorld service methods.

1. In Visual Studio 2008, open the Default.aspx page.
2. Delete the existing markup in the page and replace it with the following code:

C#

[VB](http://msdn.microsoft.com/en-us/library/bb532367(d=printer,v=vs.90).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">

<style type="text/css">

body { font: 11pt Trebuchet MS;

color: #000000;

padding-top: 72px;

text-align: center }

.text { font: 8pt Trebuchet MS }

</style>

<title>Using AJAX Enabled ASP.NET Service</title>

</head>

<body>

<form id="Form1" runat="server">

<asp:ScriptManager runat="server" ID="scriptManager">

<Services>

<asp:ServiceReference path="~/HelloWorld.asmx" />

</Services>

<Scripts>

<asp:ScriptReference Path="~/HelloWorld.js" />

</Scripts>

</asp:ScriptManager>

<div>

<h3>Using AJAX Enabled ASP.NET Service</h3>

<table>

<tr align="left">

<td>Click to call the Hello World service:</td>

<td>

<button id="Button1"

onclick="OnClickGreetings(); return false;">Greetings</button>

</td>

</tr>

</table>

</div>

</form>

<hr/>

<div>

<span id="Results"></span>

</div>

</body>

</html>

The page contains a [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager(v=vs.90).aspx) control. The path attribute in the ServiceReference element of the Services section references the HelloWorld service that you built previously. The path attribute in the ServiceReference element of the Script section references the HelloWorld.js script.

1. Open the browser and enter the following URL:

http://<localhost>/HelloWorldService/Default.aspx

1. In the displayed page, click the Greetings button.

This generates a call to the Web service that returns a greeting message and the current server date and time.

[Next Steps](javascript:void(0))

This walkthrough has illustrated the basic principles of creating a Web service and then calling it in a Web page from client script. You might want to experiment with additional, more complex Web service features. Suggestions for additional exploration include the following:

* Understand how clients access Web services. For more information, see [Accessing Web Services in Managed Code](http://msdn.microsoft.com/en-us/library/7sxekx3f(v=vs.90).aspx).
* Understand the fundamentals of Web services. For more information, see [Programming the Web with Web Services](http://msdn.microsoft.com/en-us/library/9t8zkaxa(v=vs.90).aspx).
* Understand how to access Web services from a browser. For more information, see [How to: Access XML Web Services from a Browser](http://msdn.microsoft.com/en-us/library/45fez2a8(v=vs.90).aspx).