**Microsoft Office SharePoint Server (MOSS) for Developers**

**(Part 1)**

**Topic 8: Web Services**





[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

The Windows SharePoint Services Web services provided by the [Microsoft.SharePoint.SoapServer](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.soapserver.aspx) namespace include methods for accessing content on a Web site—such as for working with lists or site data—as well as methods for customizing meetings, imaging, document workspaces, or search.

SOAP interfaces used in these services provide Microsoft .NET Framework developers with object models for creating solutions that work with Windows SharePoint Services remotely from a client or custom application. The interfaces are defined through the server-side object model of the [Microsoft.SharePoint](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.aspx) assembly, and their design is optimized to reduce the number of roundtrips transacted between client computer and server.

Most Web services provide their functionality through the /\_vti\_bin virtual directory, which maps to the \\Program Files\Common Files\Microsoft Shared\web Server extensions\12\ISAPI physical directory in the file system. The Administration Web service uses the /\_vti\_adm virtual directory, which maps to \12\ADMISAPI.

You will need to install the [**Windows SharePoint Services 3.0 SDK**](http://msdn.microsoft.com/en-us/library/ms441339.aspx)

**Windows SharePoint Services Web Services**

Windows SharePoint Services Web services provide methods that you can use to work remotely with a deployment of Windows SharePoint Services. The following table lists and describes the available Web services.

|  |  |
| --- | --- |
| **Service**  | **Description**  |
| [Administration Web Service](http://msdn.microsoft.com/en-us/library/administration.aspx) | Provides methods for managing a deployment of Windows SharePoint Services, such as for creating or deleting sites. |
| [Alerts Web Service](http://msdn.microsoft.com/en-us/library/alerts.aspx) | Provides methods for working with alerts for list items in a SharePoint site. |
| [Authentication Web Service](http://msdn.microsoft.com/en-us/library/websvcauthentication.aspx) | Provides classes for logging on to a SharePoint site that is using forms-based authentication. |
| [Copy Web Service](http://msdn.microsoft.com/en-us/library/copy.aspx) | Provides methods for copying items between locations in Windows SharePoint Services. |
| [DocumentWorkspace Web Service](http://msdn.microsoft.com/en-us/library/ms774834.aspx) | Provides methods for managing Document Workspace sites and the data they contain. |
| [Forms Web Service](http://msdn.microsoft.com/en-us/library/forms.aspx) | Provides methods for returning forms used in the user interface when working with the contents of a list. |
| [Imaging Web Service](http://msdn.microsoft.com/en-us/library/imaging.aspx) | Provides methods that enable you to create and manage picture libraries. |
| [ListDataRetrieval Web Service](http://msdn.microsoft.com/en-us/library/ms774413.aspx) | Provides a method for performing queries against lists in Windows SharePoint Services. |
| [Lists Web Service](http://msdn.microsoft.com/en-us/library/lists.aspx) | Provides methods for working with lists and list data. |
| [Meetings Web Service](http://msdn.microsoft.com/en-us/library/ms774629.aspx) | Provides methods that enable you to create and manage Meeting Workspace sites. |
| [People Web Service](http://msdn.microsoft.com/en-us/library/people.aspx) | Provides methods for working with security groups. |
| [Permissions Web Service](http://msdn.microsoft.com/en-us/library/permissions.aspx) | Provides methods for working with the permissions for a site or list. |
| [SharepointDirectoryManagementService Web Service](http://msdn.microsoft.com/en-us/library/websvcsharepointdirectorymanagementservice.aspx) | Provides methods for remotely managing distribution groups. |
| [SiteData Web Service](http://msdn.microsoft.com/en-us/library/ms774821.aspx) | Provides methods that return metadata or list data from sites or lists in Windows SharePoint Services.  |
| [Sites Web Service](http://msdn.microsoft.com/en-us/library/sites.aspx) | Provides a method for returning information about the site templates for a site collection. |
| [SPSearch Web Service](http://msdn.microsoft.com/en-us/library/search.aspx) | Provides methods for remotely performing searches within a Windows SharePoint Services deployment. |
| [UsersGroups Web Service](http://msdn.microsoft.com/en-us/library/ms772647.aspx) | Provides methods for working with users, site groups, and cross-site groups. |
| [Versions Web Service](http://msdn.microsoft.com/en-us/library/ms772545.aspx) | Provides methods for working with file versions. |
| [Views Web Service](http://msdn.microsoft.com/en-us/library/views.aspx) | Provides methods for working with views of lists. |
| [WebPartPages Web Service](http://msdn.microsoft.com/en-us/library/ms774569.aspx) | Provides methods to send and retrieve Web Part information to and from Web services. |
| [Webs Web Service](http://msdn.microsoft.com/en-us/library/webs.aspx) | Provides methods for working with sites and subsites. |

**Walkthrough: Creating a Custom Web Service**

This programming task provides an overview of how to create a custom Web service that operates within the context of Windows SharePoint Services 3.0. It steps through the process of creating a simple "Hello World" Web service and then shows how to modify the service so that it implements the Windows SharePoint Services 3.0 server-side object model to return site and list data.

 Basic Steps for Creating a Web Service

* Create an ASP.NET Web service in Microsoft Visual Studio 2005.
* Create a class library within the Web service that defines the programming logic for the Web service.
* Generate and edit a static discovery file and a Web Services Description Language (WSDL) file.
* Deploy the Web service files to the \_vti\_bin directory.
* Create a client application to consume the Web service.

 Creating an ASP.NET Web Service in Visual Studio

The first step is to create an ASP.NET Web service Web site in Visual Studio 2005.

### To create an ASP.NET Web service

1. In Visual Studio, click **File**, point to **New**, and then select **Web Site**.
2. In the **Templates** box of the **New Web Site** dialog box, select **ASP.NET Web Service**, select **File System** in the **Location** box, select a programming language and location for the project, and then click **OK**.
3. Within the new Web service solution, create a separate class library project to contain the Web service logic. To create the project, click **File**, point to **New**, and then select **Project**.
4. In the **New Project** dialog box, select a language in the **Project types** box, select **Class Library** in the **Templates** box, provide a name and location for the project, select **Add to Solution** in the **Solution** box, and then click **OK**.
5. Add a reference to the **System.Web.Services** namespace in the class library project. Right-click the project in **Solution Explorer**, click **Add Reference**, select **System.Web.Services** in the **Add Reference** dialog box, and then click **OK**.
6. Replace the default class file in the class library project with the default service class file that Visual Studio provides in the **App\_Code** folder of the Web service.

**To replace the class file with the service class file**

* 1. In **Solution Explorer**, drag Service.cs or Service.vb to the top node in the class library project.
	2. Delete the Class1.cs or Class1.vb file, and also delete the Service.cs or Service.vb file that remains in the **App\_Code** folder.
1. Create a strong name for the class library:
	1. In **Solution Explorer**, right-click the class library project, and then click **Properties**.
	2. In the **Properties** dialog box, click **Signing**, select **Sign the assembly**, and then select **<New>** in the **Choose a strong name key file** list.
	3. In the **Create Strong Name Key** dialog box, provide a file name for the key, clear the **Protect my key file with a password** check box, and then click **OK**.
2. To build only the class library project, right-click the project in **Solution Explorer**, and then click **Build**.
3. To add your assembly to the global assembly cache (GAC), you can either drag the assembly into the %windows%\assembly directory using 2 instances of Windows Explorer, or use the command line utility gacutil.exe that is installed with the Microsoft .NET Framework SDK 2.0.

**To use gacutil.exe to copy the class library DLL into the GAC**

* 1. To open the Visual Studio command prompt, click **Start**, point to **All Programs**, point to **Microsoft Visual Studio 2008**, point to **Visual Studio Tools**, and click **Visual Studio 2008 Command Prompt**.
	2. At the command prompt type a command in the following form, and press ENTER:

gacutil.exe -if "<Full file system path to DLL>".

1. Now you are ready to modify the assembly information in the default Service.asmx file of the Web service with information for the DLL from the GAC. To get information from the GAC, open the %windows%\assembly directory in Windows Explorer, right-click your assembly, and click **Properties**.
2. To open Service.asmx in **Solution Explorer**, right-click the file and click **Open**.
3. Remove the **CodeBehind** attribute from the page directive in Service.asmx, and modify the contents of the **Class** attribute so that the directive matches the following format, where the assembly name "MyServiceAssembly" and the public key token are values specified in the **Properties** dialog box that you opened in step 10:

|  |
| --- |
|  |
| <%@ WebService Language="C#" Class="Service, MyServiceAssembly, Version=1.0.0.0, Culture=neutral, PublicKeyToken=8f2dca3c0f2d0131" %> |

1. In Visual Basic include the namespace to identify the class, for example, Class="MyServiceNamespace.Service, MyServiceAssembly, Version=1.0.0.0, Culture=neutral, PublicKeyToken=8f2dca3c0f2d0131".
2. Rename your .asmx file appropriately, and then save your changes.

 Generating and Modifying Static Discovery and WSDL Files

To provide discovery and description for your custom Web service, you must create a .disco file and a .wsdl file. Because Windows SharePoint Services virtualizes its URLs (for example, http://MyServer/MySite/MySubsite becomes http://MyServer), you cannot use the autogenerated .disco and .wsdl files generated by ASP.NET. Instead, you must create a .disco page and a .wsdl ASPX page that provide the necessary redirection and maintain virtualization.

You can use ASP.NET to generate the .disco and .wsdl files by temporarily hosting your Web service in a virtual directory, such as /\_layouts, and then using the .NET Framework Web Service Discovery tool (Disco.exe) to obtain the generated files.

### To generate the static discovery and WSDL files

1. In Windows Explorer, copy the .asmx file of your Web service to \\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\LAYOUTS.
2. Run Disco.exe at the command prompt from the LAYOUTS directory to generate .disco and .wsdl files. Run a command in the following format to generate the files in \LAYOUTS:

disco http://MyServer/\_layouts/MyCustomWebService.asmx

1. To register namespaces of the Windows SharePoint Services object model, open both the .disco and .wsdl files and replace the opening XML processing instruction -- <?xml version="1.0" encoding="utf-8"?> -- with instructions such as the following:

|  |
| --- |
|  |
| <%@ Page Language="C#" Inherits="System.Web.UI.Page" %> <%@ Assembly Name="Microsoft.SharePoint, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %> <%@ Import Namespace="Microsoft.SharePoint.Utilities" %> <%@ Import Namespace="Microsoft.SharePoint" %><% Response.ContentType = "text/xml"; %> |

1. In the .disco file, modify the contract reference and SOAP address tags to be like the following example, which replaces literal paths with code generated paths through use of the [Microsoft.SharePoint.Utilities.SPHttpUtility](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.utilities.sphttputility.aspx) class, and which replaces the method name that is specified in the **binding** attribute:

|  |
| --- |
|  |
| <contractRef ref=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(SPWeb.OriginalBaseUrl(Request) + "?wsdl"),Response.Output); %> docRef=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(SPWeb.OriginalBaseUrl(Request)),Response.Output); %> xmlns="http://schemas.xmlsoap.org/disco/scl/" /><soap address=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(SPWeb.OriginalBaseUrl(Request)),Response.Output); %> xmlns:q1="http://tempuri.org/" binding="q1:HelloWorld" xmlns="http://schemas.xmlsoap.org/disco/soap/" /><soap address=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(SPWeb.OriginalBaseUrl(Request)),Response.Output); %> xmlns:q2="http://tempuri.org/" binding="q2:ServiceSoap12" xmlns="http://schemas.xmlsoap.org/disco/soap/" /> |

1. In the .wsdl file, make the following similar substitution for the SOAP address that is specified:

|  |
| --- |
|  |
| <soap:address location=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(SPWeb.OriginalBaseUrl(Request)),Response.Output); %> /> |

1. Rename both files in the respective formats MyCustomWebServicedisco.aspx and MyCustomWebServicewsdl.aspx so that your service is discoverable through Windows SharePoint Services.

 Copying the Web Service Files to the \_vti\_bin Directory

The \_vti\_bin virtual directory maps physically to the *Local\_Drive*:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\ISAPI directory, which contains the default Web service files used in Windows SharePoint Services. Copy the new MyCustomWebServicewsdl.aspx and MyCustomWebServicedisco.aspx files, and also the MyCustomWebService.asmx file, to the ISAPI folder.

From the \_vti\_bin directory, a Web service offers its functionality to the site that is specified when adding a Web reference for the service.

To verify that your custom Web service is discoverable, navigate to http://MyServer/\_vti\_bin/MyCustomWebService.asmx.

 List the New Web Service in spdisco.aspx

To make your Web service discoverable in Visual Studio as a Web service alongside the default Windows SharePoint Services Web services, open the spdisco.aspx file located in \Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\ISAPI and add the following code, specifying the .asmx file for your Web service.

|  |
| --- |
|  |
| <contractRef ref=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(spWeb.Url + "/\_vti\_bin/MyCustomWebService.asmx?wsdl"), Response.Output); %> docRef=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(spWeb.Url + "/\_vti\_bin/MyCustomWebService.asmx"), Response.Output); %> xmlns=" http://schemas.xmlsoap.org/disco/scl/ " /><discoveryRef ref=<% SPHttpUtility.AddQuote(SPHttpUtility.HtmlEncode(spWeb.Url + "/\_vti\_bin/MyCustomWebService.asmx?disco"),Response.Output); %> xmlns="http://schemas.xmlsoap.org/disco/" /> |

 Creating a Windows Application to Consume the Web Service

After you copy the Web services files to the \_vti\_bin directory, the next step is to create a Windows Application to consume the Web service.

### To create a Windows Application that consumes the Web service

1. Open Visual Studio 2005, and on the **File** menu, point to **New**, and then click **Project**.
2. In the **New Project** dialog box, select **Visual C#** or **Visual Basic**, and then select the **Windows Application** template.
3. Type a name for the application in the **Name** box, specify a location for the project files in the **Location** box, and then click **OK**.
4. In **Solution Explorer**, right-click the project, and then click **Add Web Reference**.
5. In the address bar of the **Add Web Reference** browser, type the URL for the site to apply the service to, as follows, and then press ENTER:

**http://Server\_Name/[sites/][Site\_Name/]\_vti\_bin/MyCustomWebService.asmx**

1. Click **Add Reference** to download the service contract for the Web service.
2. Open **Form1** in Design view, display the **Toolbox**, and then drag a button onto the form.
3. Double-click the **Button1** control on **Form1** to display the code-behind file in the code editor, and add the following code that calls your custom method.

|  |
| --- |
| Visual Basic |
| Dim MyCustomService As New *Web\_Reference\_Folder*.MyServiceClass()MyCustomService.UseDefaultCredentials = TrueMessageBox.Show(MyCustomService.HelloWorld()) |
| C# |
| *Web\_Reference\_Folder*.MyServiceClass MyCustomService = new *Web\_Reference\_Folder*.MyServiceClass();MyCustomService.UseDefaultCredentials = true;MessageBox.Show(MyCustomService.HelloWorld()); |

1. Press F5 to compile and run the project and see a message box that displays "Hello World".

# **Additional resources and references**

* [Working](http://msdn.microsoft.com/en-us/library/bb153523.aspx) with SharePoint Web Services
* [Introduction to SharePoint Products and Technologies for the Professional .NET Developer](http://msdn.microsoft.com/en-us/library/cc537498.aspx)
* [Windows SharePoint Services Developer Center](http://msdn2.microsoft.com/en-us/SharePoint)
* [Windows SharePoint Services TechCenter](http://technet.microsoft.com/en-us/office/sharepointserver/default.aspx)
* [SharePoint Server Developer Center](http://msdn2.microsoft.com/en-us/office/aa905503.aspx)
* [SharePoint Server TechCenter](http://technet.microsoft.com/en-us/windowsserver/sharepoint/default.aspx)