CodePlex Archive Open Source Project Archive

SPWCFWebServiceDemo

SharePoint 2010 WCF WebService sample

download archive

This demo shows how to implement a simple WCF webservice for SharePoint 2010

home issues discussions

In this “Walkthrough” I’d like to show you how to create a simple WCF Web Service for SharePoint 2010.

This Web Service will provide two methods:

- Ping => Just as test.
- GetUsedEmailAliasses => This method reports all aliasses used for “E-Mail enabled lists” in SharePoint.

Here is the corresponding blog post:

https://ikarstein.wordpress.com/2012/01/21/walkthrough-create-simple-wcf-web-service-for-sharepoint-2010-with-visual-studio-2010/

Let’s start.

1. Create a new SharePoint 2010 project in Visual Studio 2010. Use the “Empty Project” template. Name the project “ik.SharePoint2010.SPWCFWebServiceDemo”.

image_thumb

2. In the project creation wizard specify the Central Administration address as deployment location. You need to deploy as “Farm Solution”:

image_thumb1

3. Now add this Assembly References:

- Microsoft.SharePoint.Client
- Microsoft.SharePoint.Client.Runtime
- Microsoft.SharePoint.Client.ServerRuntime
  (You may need to locate this assembly in the file system: "C:\Windows\assembly\GAC_MSIL\Microsoft.SharePoint.Client.ServerRuntime\14.0.0.0__71e9bce111e9429c\Microsoft.SharePoint.Client.Serve
- System.Configuration
- System.ServiceModel
- System.ServiceModel.Web

  Now my Assembly Reference tree looks like this:

  …

4. Now map the SharePoint hive folder “Layouts” to your project:

a) Right click your project in the Solution Explorer Pane

b) In the context menu open sub menu “Add” and click “SharePoint Mapped Folder”

c) Select this:

image_thumb2

d) Click “OK”

5. Create a sub folder named “ik.SharePoint2010.SPWCFWebServiceDemo” in folder “Layouts”

6. Add a new project item of type “Interface” to your project. Name it “ISPWCFSvc.cs”.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Runtime.InteropServices;
using System.ServiceModel;
using System.ServiceModel.Web;

namespace ik.SharePoint2010.SPWCFWebServiceDemo
{
    [ServiceContract(Namespace="http://schemas.microsoft.com/sharepoint/soap/"),
     Guid("f01e2ff6-c291-4b8b-a154-cd7059ed4900")]
    public interface ISPWCFSvc
    {
        [OperationContract, WebInvoke(Method = "GET")]
        List<string> GetUsedEmailAliasses();

        [OperationContract, WebInvoke(Method = "GET")]
        string Ping();
    }
}
```

7. Add a new project item of type “Class” and name it “SPWCFSvc.cs”

```
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Security.Permissions;
using System.ServiceModel;
using System.ServiceModel.Activation;
using System.Web;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Administration;
using Microsoft.SharePoint.Client;
using Microsoft.SharePoint.Security;


namespace ik.SharePoint2010.SPWCFWebServiceDemo
{
    [Guid("ae428eb5-02d1-4e50-8cee-eb3806f16ffd"),
    ServiceBehavior(Namespace = "http://schemas.microsoft.com/sharepoint/soap/"),
    AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Required),
    ClientRequestServiceBehavior,
    SharePointPermission(SecurityAction.Demand, ObjectModel = true),
    AspNetHostingPermission(SecurityAction.Demand, Level = AspNetHostingPermissionLevel.Minimal)]
    public class SPWCFSvc : ISPWCFSvc
    {
        [SharePointPermission(SecurityAction.Demand, ObjectModel = true)]
        public List<string> GetUsedEmailAliasses()
        {
            List<string> l = new List<string>();

            SPProcessIdentity pi = SPContext.Current.Site.WebApplication.Farm.TimerService.ProcessIdentity;
            string userName = pi.Username;
            object configDB = pi.GetType().GetProperty("ConfigurationDatabase", BindingFlags.Instance | BindingFlags.NonPublic).GetValue(pi
            SPServer server = (SPServer)configDB.GetType().GetProperty("Server").GetValue(configDB, null);
            string dbName = (string)configDB.GetType().GetProperty("Name").GetValue(configDB, null);
            string serverName = server.Name;

            SPSecurity.RunWithElevatedPrivileges(() =>
            {
                System.Data.SqlClient.SqlConnection s = new System.Data.SqlClient.SqlConnection(new System.Data.SqlClient.SqlConnectionStri
                {
                    DataSource = serverName,
                    InitialCatalog = dbName,
                    IntegratedSecurity = true
                }.ConnectionString);


                SPConfigDBDataContext dc = new SPConfigDBDataContext(s);

                foreach( var item in dc.EmailEnabledLists )
                {
                    l.Add(item.Alias);
                }
            });

            return l;
        }

        [SharePointPermission(SecurityAction.Demand, ObjectModel = true)]
        public string Ping()
        {
            return "Pong";
        }
    }
}
```

8. Now add a new “Text File” project item and name it “spwcfsvc.svc”

```
<% @ServiceHost Service="$SharePoint.Type.ae428eb5-02d1-4e50-8cee-eb3806f16ffd.AssemblyQualifiedName$"
Factory="Microsoft.SharePoint.Client.Services.MultipleBaseAddressWebServiceHostFactory,
Microsoft.SharePoint.Client.ServerRuntime, Version=14.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c"%>
```

Here we use an existing service factory of SharePoint that will handle the service instanciation for us.

In this file we use the token replacement functionality of Visual Studio 2010 for SharePoint 2010 development. We like to replace the token “$SharePoint.Type.ae428eb5-02d1-4e50-8cee-eb3806f16ffd.AssemblyQualifiedName$” through the “real” full qualified name during package creation.

By default *.svc files will not be processed by the token replacement engine. We need to specify *.svc files for that build step. Therefore we edit the C# project file in notepad. Insert this line as shown in the screenshot:

<TokenReplacementFileExtensions>svc</TokenReplacementFileExtensions>

image_thumb[4]

After deploying the solution you may have a look into the SharePoint hive and look into "spwcfsvc.svc"

9. Now Copy & Paste this file into the you folder in the mapped Layouts folder:

C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\ISAPI\web.config

10. You need to edit this file. Remove the complete "<system.web>" tag and the (four) "location" tags at the beginning of the copied file! – The configuration settings in the copied & modified web.config file will manage the diffrent authentication methods for us. This works by using the class attribute "ClientRequestServiceBehavior" in the code above.

11. Now we add a "Linq to SQL" connection to the SharePoint Configuration database. **Be sure only to read this DB!!!**

12. Add a new project item of type "Linq to SQL Classes" to the project. Name it "SPConfigDB.dbml"

image_thumb[5]

13. In the Server Explorer of Visual Studio 2010 create a connection to the SharePoint Configuration DB of your development environment. – After that, open the connection, open the "Table" node and select "EmailEnabledLists".

image_thumb[6]

14. Drag the "EmailEnabledLists" node to the "SPConfigDB.dbml" canvas. – It looks like this:

image_thumb[7]

15. In the Properties Pane set the "Context Namespace" property to "ik.SharePoint2010.SPWCFWebServiceDemo". (If you do not see this properties like in the screenshot below you need to right click the "SPConfigDB.dbml" project item node in the Solution explorer and click "Properties" in the context menu.

image_thumb[8]

16. Now you should be able to compile and deploy your project.

17. Edit the startup settings. In the "Debug" select "Start browser with URL" and enter there the URL to your webservice: "http://<central-administration-url-and-port>/_layouts/ik.sharepoint2010.spwcfwebservicedemo/spwcf.svc/GetUsedEmailAliasses"

SNAGHTML4b070d4_thumb

18. Now start the project. You get something like this:

SNAGHTML4acbe57_thumb

I've only one e-mail enabled list or library in my current dev system.