

Wrapping a User Control in a Web Part

This article demonstrates how to wrap a custom User Control in a Web Part and deploy it as a feature using WSPBuilder (<http://www.codeplex.com/wspbuilder>).

This article assumes you have some basic knowledge about WSPBuilder and how to deploy SharePoint assemblies as solution packages (wsp).

Solution Overview

The solution architecture consists of the following two projects:

- A WSPBuilder Project using the Visual Studio project template installed during the setup of WSPBuilder. This project will allow us to create the wsp file to be loaded in the SharePoint solutions store.
- A classic ASP.Net Web Application Project to create our user control. We will create custom post-build scripts to copy the user control to the **CONTROLTEMPLATES** sub-folder created in the WSPBuilder project.

Step-by-Step Instructions

1. Download and setup WSPBuilder from the Codeplex site

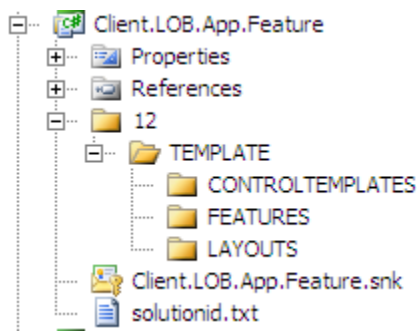
(<http://www.codeplex.com/wspbuilder>)

2. Create the WSPBuilder feature project

1. In Visual Studio, point to **New** on the **File** menu, and then click **Project**.
2. In the **Project types** pane, click **WSPBuilder**. In the Templates pane, click **WSPBuilder Project**. In the **Name** box, type **Client.LOB.App.Feature**, and then click **OK**.

Please select a good name for your project. I usually like to use the following namespace convention: ClientName.LOB.Application.Feature.

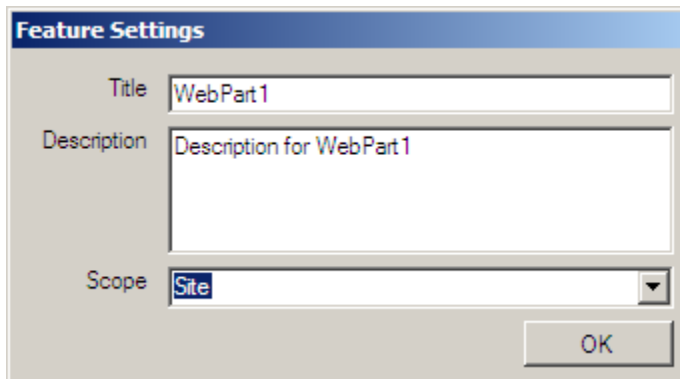
3. Right-click on the **Client.LOB.App.Feature** project name, point to **Properties**, and click on **Application**.
4. Update the Application framework to the latest version installed on your server (I will use .Net Framework 3.5 in this article).
5. Save the project and solution files.
6. In the solution explorer windows, point to the **12** folder of the **Client.LOB.App.Feature** project.
7. Add a **TEMPLATE** folder under the **12** folder.
8. Add a **FEATURES** folder under the **TEMPLATE**.
9. Add a **LAYOUTS** folder under the **TEMPLATE** folder.
10. Add a **CONTROLTEMPLATES** folder under the **TEMPLATE** folder.



11. Right-click on the project name **Client.LOB.App.Feature**, point to **Add** and click **New Item....**
12. In the Categories pane, click **WSPBuilder**. In the Templates pane, click **Web Part Feature**. In the **Name** box, type the name of your Web Part, in our case **WebPart1**, and then click **OK**.

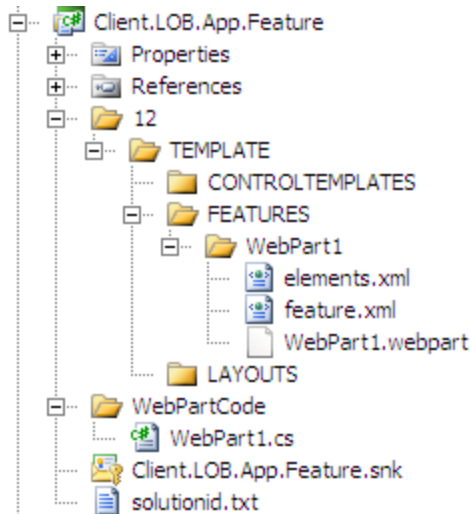
Since a feature project could include multiple web parts, give your Web Parts a good name.

13. A popup will come up with Title, Description and Scope. Because the Web Part will be deployed in the Web Part gallery of our site collection, you must select **Site** for the scope.

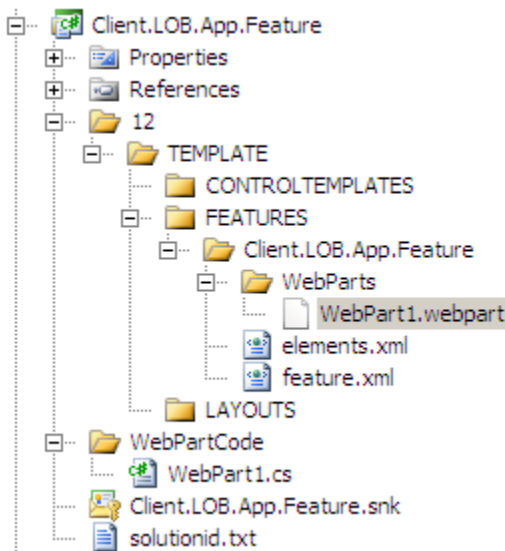


Notice that WSPBuilder did two things for you:

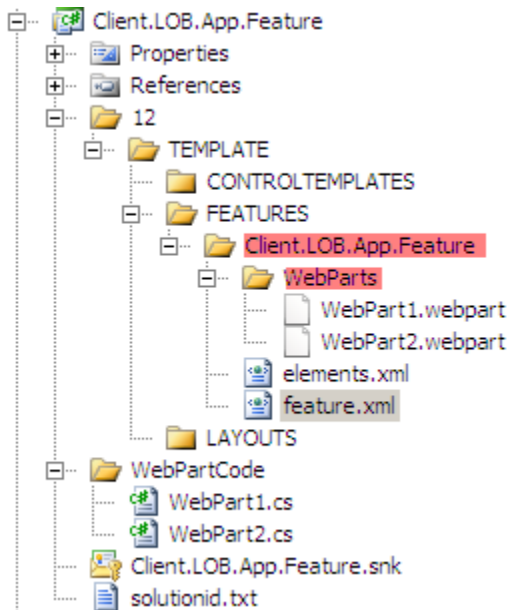
- It created the feature in the features folder
- It created the Web Part code in a folder called WebPartCode



14. Since I personally like to deploy in a single feature all user controls related to the same application, I usually re-arrange my folder structure in the following way:



This structure allow me to easily add additional web parts using a unique elements.xml and feature.xml as shown in the figure below:



15. Open the **elements.xml** file and make the following modifications:

```

1 |<?xml version="1.0" encoding="utf-8" ?>
2 |<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
3 |   <Module Name="WebPartPopulation" Path="WebParts" Url="_catalogs/wp" RootWebOnly="TRUE">
4 |
5 |     <File Url="WebPart1.webpart" Type="GhostableInLibrary">
6 |       <Property Name="Group" Value="Enter custom group name here..."></Property>
7 |       <Property Name="QuickAddGroups" Value="Enter same custom group name here..." />
8 |     </File>
9 |
10 |    <File Url="WebPart2.webpart" Type="GhostableInLibrary">
11 |      <Property Name="Group" Value="Enter same custom group name here..."></Property>
12 |      <Property Name="QuickAddGroups" Value="Enter same custom group name here..." />
13 |    </File>
14 |
15 |   </Module>
16 | </Elements>
17 |

```

16. Open the **feature.xml** file and make the following modifications:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <Feature Id="fd72a338-2d4a-42a2-bf9d-c80a922d095d"
3      Title="Feature Title goes here..."
4      Description="Feature description goes here..."
5      Version="1.0.0.0"
6      Hidden="FALSE"
7      Scope="Site"
8      DefaultResourceFile="core"
9      xmlns="http://schemas.microsoft.com/sharepoint/">
10 <ElementManifests>
11   <ElementManifest Location="elements.xml"/>
12   <ElementFile Location="webparts\WebPart1.webpart" />
13   <ElementFile Location="webparts\WebPart2.webpart" />
14 </ElementManifests>
15 </Feature>

```

17. Open the **WebPart1.cs** file in the **WebPartCode** folder
18. Remove the MyProperty property and attribute and replace with custom properties of your web part if any. For demonstration purpose, I will just remove the MyProperty property.
19. Scroll down to the **CreateChildControls** method
20. Find the comment line that says *“// Your code here...”*
21. Replace the line

```

this.Controls.Add(new LiteralControl(this.MyProperty));

```

With

```

Client.LOB.App.Web.UserControl1 ctrl =
    (Client.LOB.App.Web.UserControl1)Page.LoadControl("~/_controltemplates/Client.LOB.App.Feature/UserControl1.ascx");

```

Your **CreateChildControls()** method should look like this:

```

protected override void CreateChildControls()
{
    if (!_error)
    {
        try
        {
            base.CreateChildControls();
        }
    }
}

```

```
// Your code here...
Client.LOB.App.Web.UserControl1 ctrl =
    (Client.LOB.App.Web.UserControl1)Page.LoadControl("~/_controltemplates/Client.LOB.App.Feature/UserControl1.ascx");

// Adds it to the controls collection of the Web Part
this.Controls.Add(ctrl);
}
catch (Exception ex)
{
    HandleException(ex);
}
}
```

You could repeat the same steps with the WebPart2.cs file. Just make sure you are actually loading another user control.

3. Create the ASP.Net Web Application project

The next step is to create the Web Application project where the user control will be created:

1. In Visual Studio, right-click on the solution file, point to **Add**, and then click **New Project**.
2. In the Project Types pane, click **Web**.
3. In the Templates pane, click **ASP.NET Web Application**.
4. Name the project **Client.LOB.App.Web**, and then click **OK**.

Select a good name for your project. I usually like to use the following namespace convention: ClientName.LOB.Application.Web.

5. Sign your project with the same assembly key used in the **Client.LOB.App.Feature** project
6. Delete the **Default.aspx** file.
7. Right-click the **Client.LOB.App.Web** project, point to **Add**, and then click **New Item**.
8. In the Categories pane, click **Web**.
9. In the Templates pane, click **Web User Control**. Name the control **UserControl1.ascx**, and then click **Add**.

Select a good name for your user control. It is usually a good idea to keep the name synchronized with the name of your Web part.

10. Delete the **CodeBehind** attribute in the **UserControl1.ascx** file and replace with the following code:

```
<%@ Control Language="C#" AutoEventWireup="true"
    CodeBehind="UserControl1.ascx.cs"
    Inherits="Client.LOB.App.Web.UserControl1, Client.LOB.App.Web,
    Version=1.0.0.0, Culture=neutral, PublicKeyToken=ab332afbc40bde2b"
%>
```

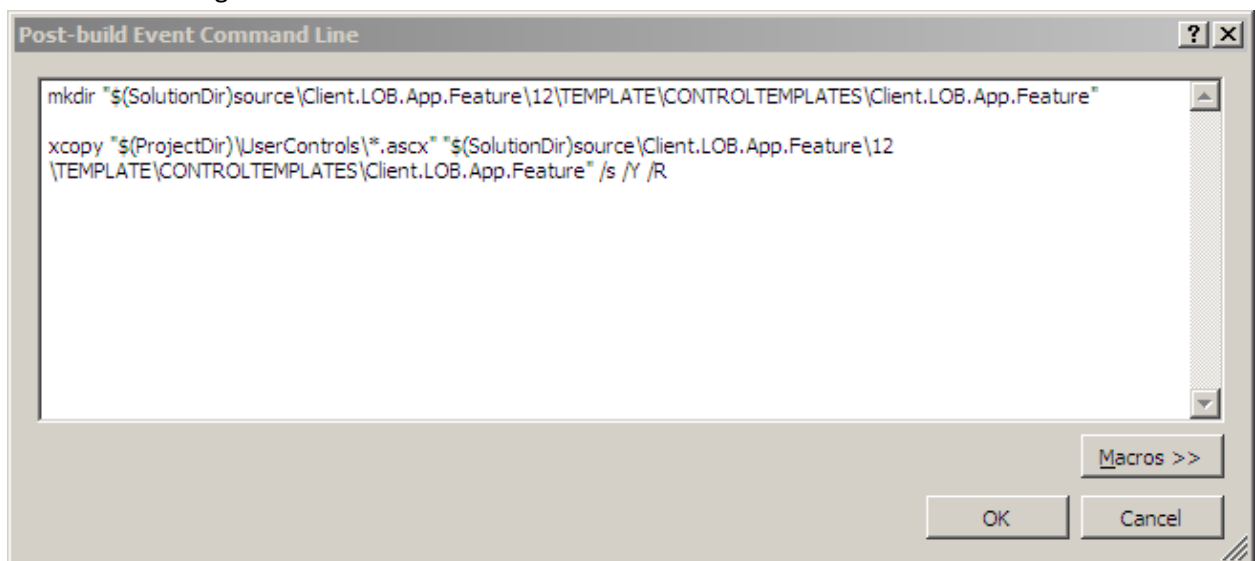
Replace the PublicKeyToken with the token created for the Client.LOB.App.Feature project. The token value can be found using the reflector tool and loading the assembly ClientLOB.App.Feature.dll assembly.

11. Right-click the **Client.LOB.App.Web** project, point to **Properties**, and then click **Build Events**.
12. Click the **Edit Post-built** button and enter the following lines:

```
mkdir
    "$(SolutionDir)source\Client.LOB.App.Feature\12\TEMPLATE\CONTROLTEMPLATES\Client.LOB.App.Feature"

xcopy "$(ProjectDir)\UserControls\*.ascx"
    "$(SolutionDir)source\Client.LOB.App.Feature\12\TEMPLATE\CONTROLTEMPLATES\Client.LOB.App.Feature" /s /Y /R
```

As shown in the figure below:



13. Save and build the **Client.LOB.App.Web** project

Since the path structure of your solution might be different than the one I used on my server, you might get an error with exit code 2. Please adjust the Post-built event with the correct path structure.

14. Right-click the **Client.LOB.App.Feature** and refresh the project
15. Navigate to the **12\CONTROLTEMPLATES** folder and include the hidden **Client.LOB.App.Feature** folder and **UserControl1.ascx** file to the project

*By building the solution, the UserControl1.ascx was copied to the **12\CONTROLTEMPLATES\Client.LOB.App.Feature** folder in the **Client.LOB.App.Feature** project.*

4. Create WSP file

1. Right-click the **Client.LOB.App.Feature** project, point to **Add Reference**, and click on the Projects tab
2. Select the **Client.LOB.App.Web** project and click **Ok**
3. Open the WebPart1.cs file and add the following reference:

```
using Client.LOB.App.Web;
```

4. Repeat the same step as above for all the web parts classes you have created (if any)
5. Save and build the **Client.LOB.App.Feature** project
6. Right-click the **Client.LOB.App.Feature** project, point to **WSP Builder**, and click **WSP Build**

We are now ready to test our solution!

5. Deploy the Solution

Now that we are finished setting everything up, we can deploy the solution. Since I usually like to control the deployment process, I have created the following script files that you can run from the file folder (PS: You could also create a PowerShell script to deploy your solution).

Install and Deploy Solution.cmd

```
echo off
```

```
@SET STSADM="c:\program files\common files\microsoft shared\web server
extensions\12\bin\stsadm.exe"
@SET SOLUTION=Client.LOB.App.Feature.wsp
SET URL=http://localhost
cls
ECHO OFF

ECHO -----
ECHO Adding the solution...
ECHO -----
%STSADM% -o addsolution -filename %SOLUTION%

IF %ERRORLEVEL% NEQ 0 goto removeError
%STSADM% -o execadmsvcjobs

ECHO -----
ECHO Deploying the solution...
ECHO -----
%STSADM% -o deploysolution -name %SOLUTION% -immediate -allowgacdeployment -
url %URL% -force

IF %ERRORLEVEL% NEQ 0 goto removeError

%STSADM% -o execadmsvcjobs
goto removeSuccess

:removeError
ECHO *** FAILURE ***
ECHO -----
pause > nul
goto :deployDone

:removeSuccess
ECHO *** SUCCESS ***
IISRESET

:deployDone
```

Retract and Delete Solution.cmd

```
echo off
```

```
@SET STSADM="c:\program files\common files\microsoft shared\web server
extensions\12\bin\stsadm.exe"
@SET SOLUTION=Client.LOB.App.Feature.wsp
SET URL=http://localhost

cls
ECHO OFF

ECHO .
ECHO -----
ECHO Retract solution
ECHO -----
ECHO .
echo -----
%STSADM% -o retractsolution -name %SOLUTION% -immediate -allcontenturls

IF %ERRORLEVEL% NEQ 0 goto removeError

%STSADM% -o execadmsvcjobs
ECHO OFF

ECHO .
ECHO -----
ECHO Delete the solution
ECHO -----
ECHO .
%STSADM% -o deletesolution -name %SOLUTION% -override

IF %ERRORLEVEL% NEQ 0 goto removeError

%STSADM% -o execadmsvcjobs
goto removeSuccess

:removeError
ECHO *** FAILURE ***
ECHO -----
pause > nul
goto :deployDone

:removeSuccess
ECHO *** SUCCESS ***
IISRESET

:deployDone
```

Note: Please make sure you are logged in to the SharePoint server with Admin privileges, otherwise you will not be able to add and deploy a solution file.

After the solution is deployed, just connect to your site collection, activate the feature and load the Web Part on a page. At this point your Web Part is empty because we did not actually add any html or code to the user control. However, the solution structure is in place to allow you to focus on pure .Net development. Just go back to your **Client.LOB.App.Web** project and start coding. When you are ready to re-deploy the web part, build the solution and rebuild the WSP file (PS: you might need to de-activate and re-activate the feature to see the latest code).

About the Author

Gilles Uréna is an independent SharePoint Consultant based in Richmond, Va. Gilles is the owner of SharePoint Global Services, an organization built from the ground up to provide consulting services on the SharePoint platform and surrounding technologies. You can contact Gilles at gurena@sharepointglobalservices.com.