Lab 08: Leveraging Publishing & Custom Web Parts

**Lab Time:** 60 minutes

**Lab Overview:** In this lab you will practice working with the additional Web Parts provided in Publishing sites, specifically the Content Query Web Part. At the end of the lab, you will practice creating a custom Web Part when the provided Web Parts don't satisfy your business requirements.

Exercise 1: Adding dummy content to a Publishing site

In this exercise you will use a provided WSS solution package that will create new site columns, content types, page layout as well as a new section within the top-level web in the Publishing site collection with dummy content for use within the remainder of this lab. You need some content in order to effectively work with the Content Query Web Part.

1. A sample application, packaged as a WSS solution package, is provided along with the source code that will create a new site within the Publishing site collection's top-level site named **Widgets** and fill this site with 10 content pages. The sample application is deployed using a Feature and Feature receiver. Upon activation, the Feature will create all the necessary site infrastructure assets, the Widgets site and content pages. Upon deactivation, everything is removed from the site.
2. The solution, and associated Visual Studio project, can be found in the following directory:

c:\Student\Resources\Sample Data\WidgetContentBuilder

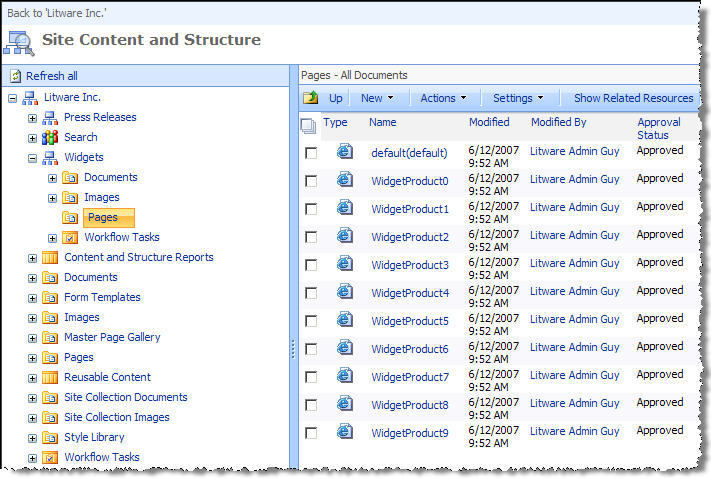
1. Within that directory you will find a file **wsp-install.bat**. Executing this file should (A) add the solution to the SharePoint farm's solution store, (B) deploy the solution and (C) activates the **WidgetContentBuilder** Feature for the Publishing site created in a previous lab. The WSP file can be found in the following directory:

c:\Student\Resources\Sample Data\WidgetContentBuilder\wsp\Debug

*Note: The Feature is deployed as a hidden Feature. Activation/deactivation must be done using STSADM from the command line.*

*Note 2: If you created your Publishing site using a different URL other than* ***http://wcm.litwareinc.com****, you will need to edit the* ***wsp-install.bat*** *file before executing the batch file. Another option is to let the activation line in the batch file fail, then manually activate the Feature from the command line.*

1. After activating the **WidgetContentBuilder** Feature (either using the batch file or from the command line), you should see a new **Widgets** site within the **http://wcm.litwareinc.com** site collection with 10 approved content pages as the following image shows:

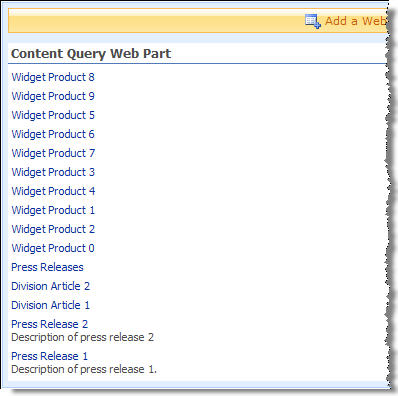


At this point, you have sample content that can be used for testing the Content Query Web Part.

Exercise 2: Implementing and customizing the Content Query Web Part

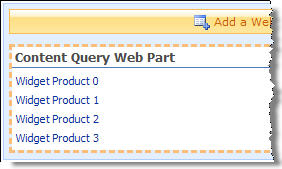
In this exercise you will add the Content Query Web Part to a page and perform some simple and advanced customization tasks on it. In the end, you would like to see a sorted list (by name) of the first 4 products from the North America division with the first bit of the page content listed as a teaser with a link to get the full post.

1. To start, add a Content Query Web Part to the Widget site's default page. Browse to the following page: **http://wcm.litwareinc.com/Widgets/Pages/default.aspx**.
2. Switch into edit mode by selecting **Site Actions » Edit Page**.
3. Scroll to the bottom of the page and click the **Add a Web Part** orange colored bar in the **Top** Web Part zone. Under the **All Web Parts / Default** section, check the box next to **Content Query Web Part** and click **Add**.
4. By default, the Content Query Web Part will display all content across the entire site collection... not what you want (as shown in the following image):



The first thing to do is to filter the scope and type of content.

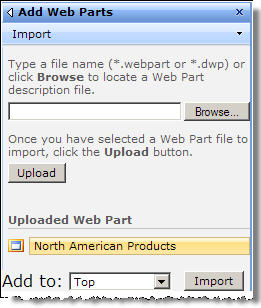
1. Select the **edit** menu in the upper-right corner of the **Content Query Web Part** and select **Modify Shared Web Part**. Expand the **Query** section and use the following information to select the desired settings:
   * **Source:** Show items from the following sites and all subsites: Widgets
   * **List Type:** Show items from this list type: Pages Library
   * Content Type:
     1. **Show items of this content type group:** Widget Content Builder
     2. **Show items of this content type:** Widget Product page
     3. **Include child content types:** unchecked
2. Expand the **Additional Filters** section and set a single filter using the following information:
   * **Show items when:** [Division] [is equal to] [North America] (*don't include the brackets... they are used simply for grouping the instructions*)
3. Expand the **Grouping and Sorting** section and set a single filter using the following information:
   * **Group items by:** None
   * **Sort items by:** Name
   * **Show items in ascending order**
   * **Limit the number of items to display:** checked
   * **Item limit:** 4
4. Once all the settings that can be set using the Content Query Web Part ToolPane are set, click **OK** at the bottom of the **ToolPane**. The resulting contents should display four pages of the Widget product pages 0-3 as shown in the following image:



1. This is as far as you can get without rolling up your sleeves and getting dirty with the code. To edit some of the more power properties of the Content Query Web Part, you need to edit the XML by hand. To do this, export the Web Part by selecting **edit** and then **Export...**. Save the **\*.webpart** file anywhere you like... such as the desktop.
2. Open the saved **\*.webpart** file in Visual Studio. First, change the name of the Web Part. Search for the property **Title** (~ line 27) and change the contents of the XML element to **North American Products**.
3. Next, you need to add a few extra fields to the XML created by the Content Query Web part... search for the property **CommonViewFields** (~ line 68) and change the contents of the XML element to **ProductDescription,RichHTML;**. This tells the Content Query Web Part to pull an additional field (along with the data type) from the content and add to the resulting XML output: ProductDescription.
4. The Content Query Web Part uses XSL to transform the XML it generates to resulting HTML for rendering in the browser. The Publishing Portal site template adds quite a few style sheets to the Style Library list when the site is created. While you could customize one of these, it's better if you create your own XSL file as you may want to use one of the provided styles at a later point in your project. To use your own style sheet, you need to tell the Content Query Web Part to import the XSL style sheet you will create.

Search for the property **ItemXslLink** (~ line 84) and change the contents of the XML element to **/Style Library/XSL Style Sheets/WCM401.xsl**.

1. Create a new XSL file and add it to the Style Library list. To do this, open **SharePoint Designer** and open the site **http://wcm.litwareinc.com**. It is easier to start by copying the existing ItemStyle.xsl file... select the file **Style Library\XSL Style Sheets\ItemStyle.xsl** by **right-clicking** it and selecting **Copy**, then **right-click** it again and select **Paste**. The file was pasted with the name of **ItemStyle\_copy(1).xsl**... **rename** this file to **WCM401.xsl**.
2. Check out the **WCM401.xsl** file you just created by right-clicking it and selecting **Check Out** then open the file.
3. The opening **<xsl:stylesheet>** node contains a few schema declarations. The remainder of the file contains a few <xsl:template> sections. The first template, **Default**, is the one you will customize. Delete all other templates from the **WCM401.xsl** file. The file should now contain only ~47 lines of XML. Change the name attribute in the **<xsl:template>** node from **Default** to **WidgetProductList**. Also change the match attribute from **\*** to **Row[@Style='WidgetProductList']**.
4. Save your changes to the XSL file.
5. Now, make sure the \*.webpart file is valid. If you aren't already on the Widget's site default page, browse to it (**http://wcm.litwareinc.com/Widgets/Pages/default.aspx**). Now you need to import the \*.webpart file. From the Page Editing Toolbar, select **Page » Add Web Parts » Import.** In the **ToolPane**, click **Browse** and select the **\*.webpart** file you have been working with. Once selected, click **Upload** in the ToolPane. After the page issues a postback, you should see your Web Part listed below the Upload button in the Task Pane, as shown in the following image:

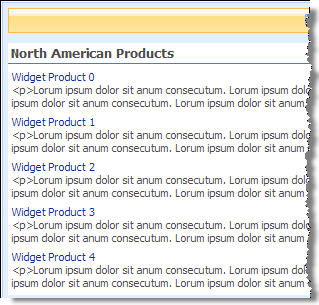


1. Add the Web Part to the page by either dragging it into a Web Part zone, or by selecting the zone from the ToolPane and clicking **Import**. You should see an instance of the new Content Query Web Part with a title of **North American Products** that contains no rendered results. That's because you need to set the item style to use. Do this by selecting **edit » Modify Shared Web Part** from the **North American Products** Content Query Web Part. Expand the **Presentation** section and scroll down to the **Styles** subsection. Notice the **Item style** is set to **WidgetProductList**... that's the only style in our XSL style sheet. You just need to apply it by clicking **OK** or **Apply** in the ToolPane.

The **North American Products** Web Part should now contain the same rendered results as the original Content Query Web Part. This is because you have simply copied the default rendering from the ItemStyle.xsl style sheet.

Now you are set to customize the XSL style sheet.

1. Go back to **SharePoint Designer** and open **/Style Library/XSL Style Sheets/WCM401.xsl** if it is not already open. The XSL template is broken into two sections. The first section (containing four <xsl:variable> elements) is used for validating XSL variables. The second section (starting with <div>) is used for rendering the output.
2. Change the description to show the contents of the article. Change the **<xsl:value-of select="@Description" />** node to **<xsl:value-of select="@ProductDescription" />** and save your changes.
3. Go back to the browser and reload (do not refresh) the page by clicking the **Widgets** link in the navigation. You will now see your modified style being used as shown in the following image:



Now we are showing the contents of the ProductDescription field... but the rendering is not ideal. Next you'll format the contents to not show the markup and show only the first 150 characters.

1. Go back to **SharePoint Designer** and open **/Style Library/XSL Style Sheets/WCM401.xsl** if it is not already open. You need to create an XSL function to parse a string, removing any markup. Add the following XSL just before the closing **</xsl:stylesheet>** node (you don't need to include the HTML comments, they are provided for extra documentation and readability):

<!-- function to remove any HTML markup from a string -->

<xsl:template name="PurgeMarkup">

<!-- single parameter of the function -->

<xsl:param name="contentToProcess" />

<xsl:choose>

<!-- when a '<' is encountered... -->

<xsl:when test="contains($contentToProcess, '&lt;')">

<!-- assign a new variable the string from just after the opening '<' to

just after the closing '>' and recursively call this function again -->

<xsl:variable name="nextContentToProcess">

<xsl:call-template name="PurgeMarkup">

<xsl:with-param name="contentToProcess" select="substring-after($contentToProcess, '&gt;')" />

</xsl:call-template>

</xsl:variable>

<!-- have the function return a string = concatenation of the

string before and after processing -->

<xsl:value-of select="concat(substring-before($contentToProcess, '&lt;'), $nextContentToProcess)" />

</xsl:when>

<!-- otherwise the content must not have an opening '<' so just kick it back -->

<xsl:otherwise>

<xsl:value-of select="$contentToProcess" />

</xsl:otherwise>

</xsl:choose>

</xsl:template>

1. Then you need to add a new XSL variable that will contain the text that has been processed by the function you just created. Add the following XSL code after the other **<xsl:variable>** nodes in the top of your **WidgetProductList** XSL template (just before the opening **<div>** tag):

<!-- assign the variable 'ProductDescription' the processed value

of the data field 'ProductDescription' passed by the CQWP

that now contains no HTML markup -->

<xsl:variable name="ProductDescription">

<xsl:call-template name="PurgeMarkup">

<xsl:with-param name="contentToProcess" select="@ProductDescription" />

</xsl:call-template>

</xsl:variable>

1. Finally, change the part of your rendering markup to use the variable instead of the value coming from the Content Query Web Part by changing the **@** symbol to a **$** so the XSL node that displays the description looks like the following markup:

<xsl:value-of select="$ProductDescription" />

1. Save you changes, go back to the browser and select the **Widgets** site... as the following images shows, you should no longer see the HTML markup:



Almost done... now you need to truncate the text down to a small teaser.

1. Go back to SharePoint Designer and change the part of the **WidgetProductList** that displays the product description text to the following markup:

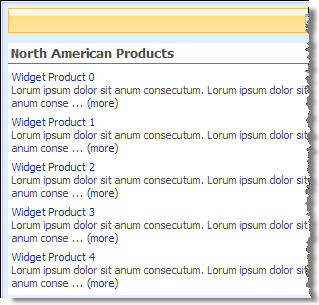
<div class="description">

<xsl:value-of select="substring($ProductDescription, 1, 150)" />

... (<a href="{$SafeLinkUrl}" target="{$LinkTarget}">more</a>)

</div>

1. Save your changes, go back to the browser and select the Widgets site... as the following image shows, all your work for this Content Query Web Part is complete!



In this exercise you learned how to configure the Content Query Web Part as well as create a custom style sheet for use within a specific Content Query Web Part.

Exercise 3: Creating a custom Web Part

In this exercise you will create a custom Web Part because inevitably, you will reach a point where the provided Web Parts don't meet your business requirements.

Similar to previous labs, since you have already created Visual Studio projects using the automated process of creating WSS solution packages, we have given you a project to get started with.

1. In Visual Studio, open the **Lab8** solution located in the following directory:

c:\Student\Labs\08\_WebParts\Lab\Lab8.sln

1. The first thing you need to do is create a new class that will be the Web Part. Create a new class named **CustomWebPart.cs** in the root of the project, replacing the default code with the following code:

using System;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

namespace Lab8

{

public class CustomWebPart: WebPart

{

}

}

1. Next, override the **CreateChildControls()** method to add two controls to the Web Part: a label displaying the current date & time and a button that when clicked, triggers a postback with a server side handler. Add the following code to the **CustomWebPart** class:

protected override void CreateChildControls()

{

base.CreateChildControls();

// create a new label containing the date & time and add

// to to the controls collection

Label lblDateTime = new Label();

lblDateTime.Text = DateTime.Now.ToString();

Controls.Add(lblDateTime);

// create a new button that, when clicked, changes the web part

// title to the current date & time

Button btnDateTime = new Button();

btnDateTime.Text = "Set Title to Current Date/Time";

btnDateTime.Click += new EventHandler(OnDateTime\_Click);

Controls.Add(btnDateTime);

}

1. Finally, add the following code to implement the server side handler when the button is clicked:

protected void OnDateTime\_Click(object sender, EventArgs e)

{

this.Title = DateTime.Now.ToString();

}

1. The Web Part code is now complete. The only thing missing for the assembly is to flag it to the .NET Framework that it can be called/executed by assemblies that are not fully trusted. This is a requirement for Web Parts as the SharePoint assemblies are not running in full trust... rather a more restricted level of trust. Add the following line of code to the end of the **AssemblyInfo.cs** file to tell the .NET Framework that assemblies that aren't fully trusted can call & execute the Web Part assembly:

[assembly: System.Security.AllowPartiallyTrustedCallers]

1. Now you need to address the deployment part of the project. You will use a Feature to deploy this Web Part to a specific site. Create a new XML file named **feature.xml** in the **Lab8** Feature folder in the project, filling it with the following markup:

<?xml version="1.0" encoding="utf-8" ?>

<Feature xmlns="http://schemas.microsoft.com/sharepoint/"

Id="5A461434-720A-463c-BB26-CC523A9902F9"

Title="Lab 8 - Custom Web Parts"

Hidden="FALSE"

Scope="Site"

Version="1.0.0.0">

<ElementManifests>

<ElementManifest Location="elements.xml" />

<ElementFile Location="customWebPart.webpart" />

</ElementManifests>

</Feature>

1. Next, create the element manifest listed in the Feature definition file. Create an XML file in the **Lab8** folder named **elements.xml** and fill it with the following XML markup:

<?xml version="1.0" encoding="utf-8" ?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<Module Url="\_catalogs/wp"

RootWebOnly="TRUE">

<File Url="CustomWebPart.webpart"

Type="GhostableInLibrary">

<Property Name="Group" Value="WCM401" />

<Property Name="Title" Value="Lab 7 - Custom Web Part" />

</File>

</Module>

</Elements>

1. Finally, create another XML file named **customWebPart.webpart** in the **Lab8** folder and fill it with the following XML markup:

<?xml version="1.0" encoding="utf-8" ?>

<webParts>

<webPart xmlns="http://schemas.microsoft.com/WebPart/v3">

<metaData>

<type name="Lab8.CustomWebPart, Lab8, Version=1.0.0.0, Culture=neutral, PublicKeyToken=d4e5777b16a5749f" />

<importErrorMessage>Error importing the Web Part.</importErrorMessage>

</metaData>

<data>

<properties>

<property name="Title" type="string">Custom Web Part</property>

</properties>

</data>

</webPart>

</webParts>

1. Now it is time to package the Feature and Web Part assembly up for deployment. Add a new XML file named **manifest.xml** to the **DeploymentFiles** folder in the project and add the following XML code to the file:

<?xml version="1.0" encoding="utf-8" ?>

<Solution xmlns="http://schemas.microsoft.com/sharepoint/"

SolutionId="A305AC7A-4F65-4684-AC3B-62031FDBFE82"

DeploymentServerType="WebFrontEnd"

ResetWebServer="FALSE">

<FeatureManifests>

<FeatureManifest Location="Lab8\feature.xml" />

</FeatureManifests>

<Assemblies>

<Assembly DeploymentTarget="WebApplication" Location="Lab8.dll">

<SafeControls>

<SafeControl Namespace="Lab8" Safe="True" TypeName="\*" />

</SafeControls>

</Assembly>

</Assemblies>

</Solution>

1. Finally, open the **BuildSharePointPackage.ddf** file in the **DeploymentFiles** folder and add the following lines between the comments:

DeploymentFiles\manifest.xml

bin\debug\Lab8.dll

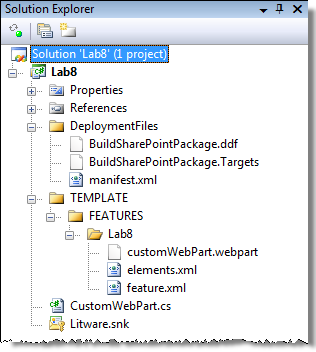
.Set DestinationDir=Lab8

TEMPLATE\FEATURES\Lab8\feature.xml

TEMPLATE\FEATURES\Lab8\elements.xml

TEMPLATE\FEATURES\Lab8\customWebPart.webpart

1. When you save everything, your project should look like the following image:



1. Now it's time to deploy the WSS solution package.
2. First the WSS solution package must be deployed. Open a command prompt and navigate to the following directory:

c:\Program Files\Common Files\Microsoft Shared\web server extensions\12\BIN

1. Enter the following command into the command line window and hit **Enter**:

stsadm -o addsolution -filename c:\Student\Labs\08\_WebParts\Lab\wsp\Debug\Lab8.wsp

1. Launch Central Administration by selecting **Start » All Programs » Microsoft Office Server » SharePoint 3.0 Central Administration**.
2. From the **Central Administration** site, select the **Operations** tab and then select **Solution management** under the **Global Configuration** section.
3. On the **Solution Management** page, click the link on **lab8.wsp**.
4. On the **Solution Properties** page, select **Deploy Solution**.
5. On the **Deploy Solution** page, specify **Now** in the **Deploy When?** section and click **OK**.
6. Test the Feature by browsing to the **http://wcm.litwareinc.com/** site and select **Site Actions » Site Settings » Modify All Site Settings**.
7. On the **Site Settings** page, select **Site collection features** under the **Site Collection Administration** section.
8. On the **Site Collection Features** page, click **Activate** on the **Lab 8 - Custom Web Parts Feature**.
9. With the Feature activated, go back to the homepage of the site and add the Web Part to any Web Part zone. Notice when you click the button, the title of the Web Part changes.

In this exercise you created a custom Web Part that was deployed with a WSS Feature.