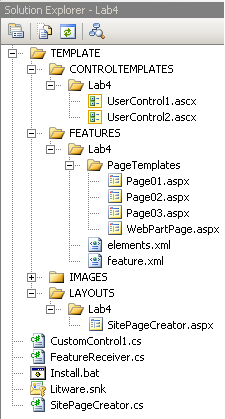
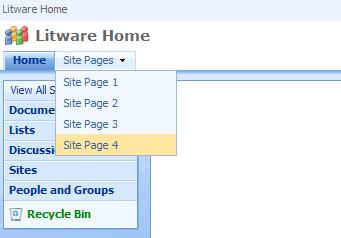
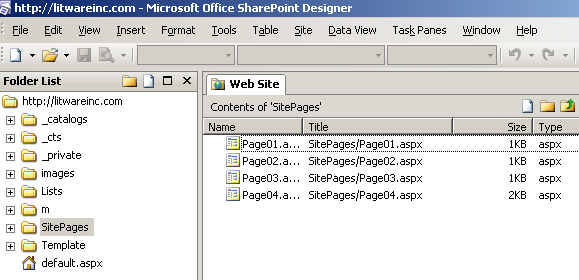
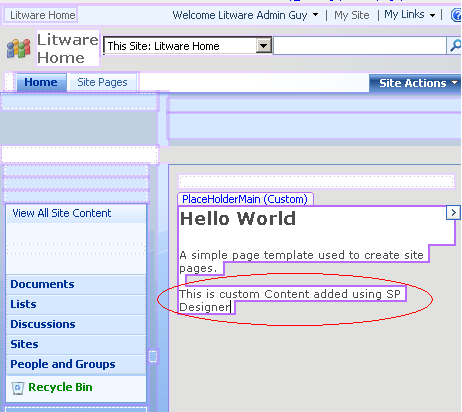
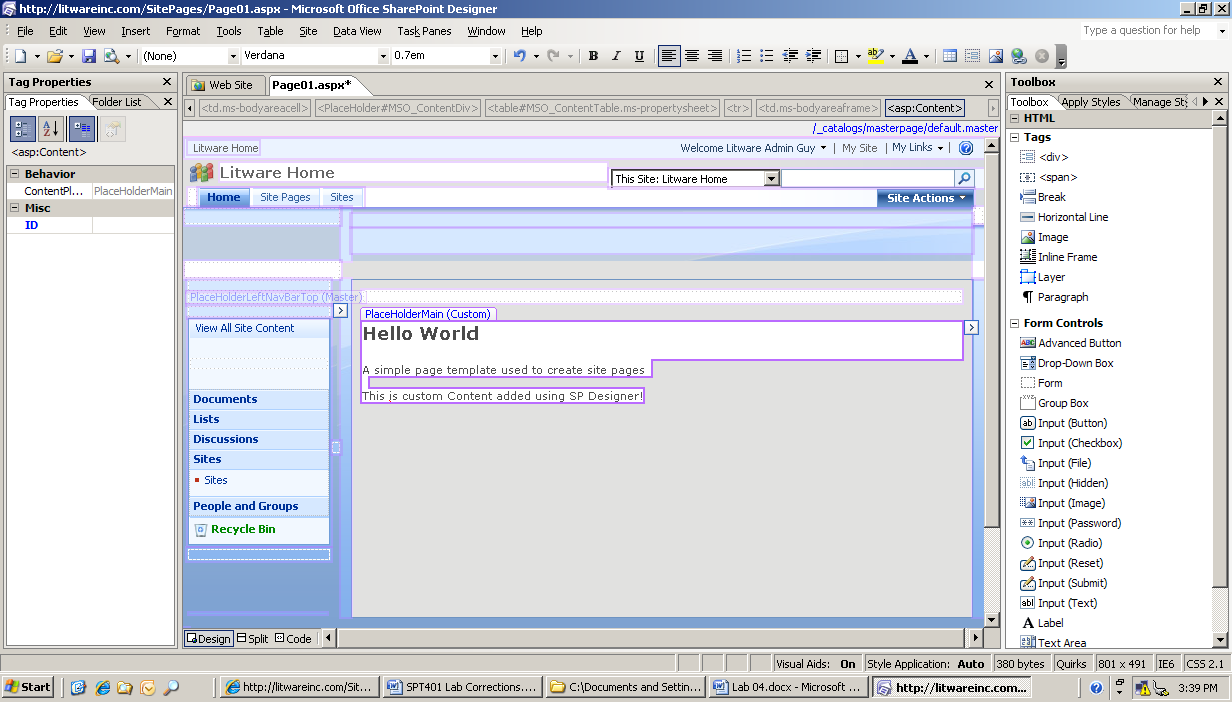
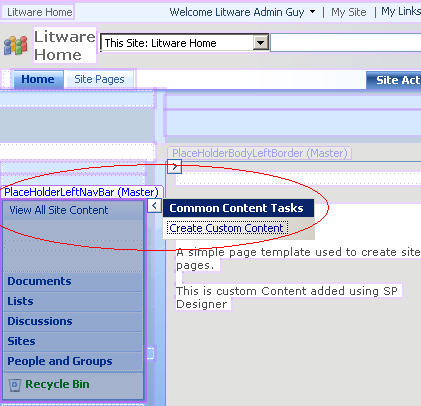
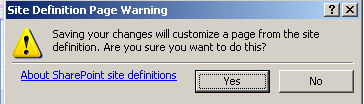
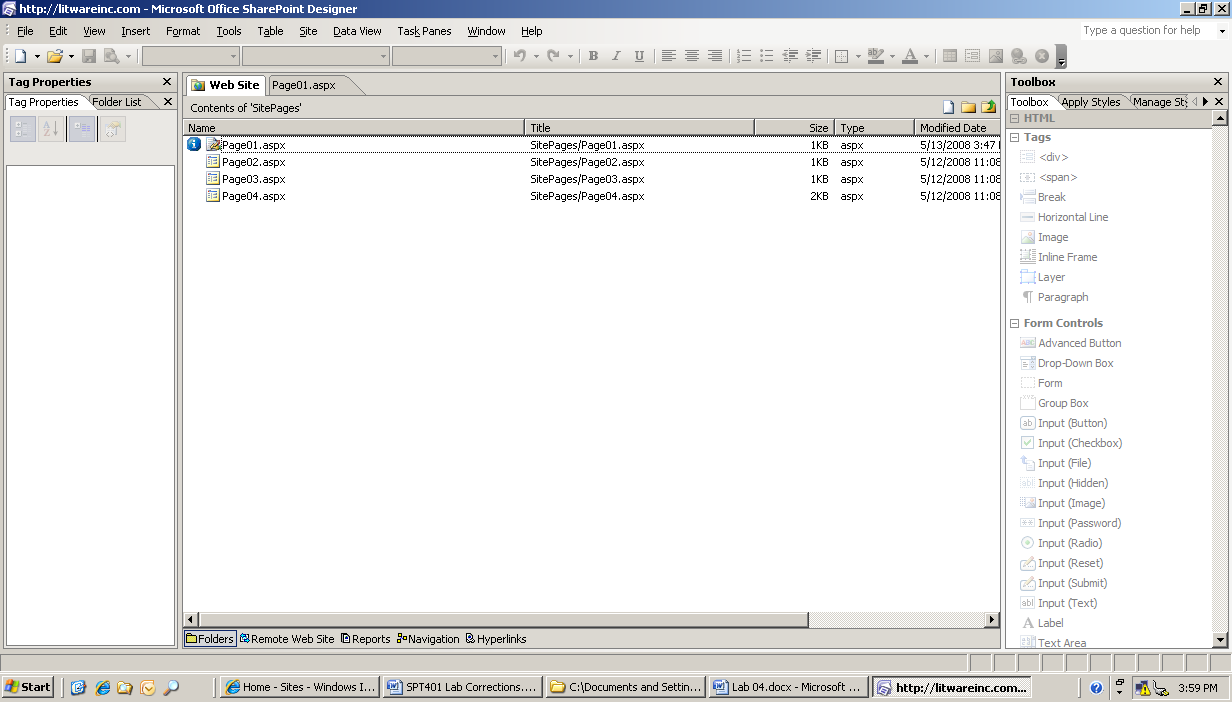
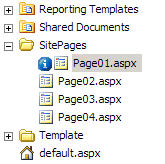
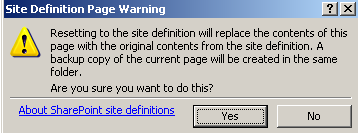
Lab 04: Developing Custom Site Pages

**Lab Time**: 45 Minutes

**Lab Directory**: C:/Student/Labs/04\_SitePages

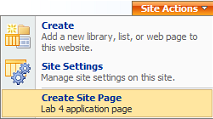
**Lab Overview**: In this lab, you will work with a Visual Studio project to write, deploy and test custom site pages that exist within the virtual file system of a WSS site. This will teach you how to create and instantiate page templates using a Feature. It will also show you how to create site pages on-the-fly using custom code.

# Exercise 1: Build and Activate a Feature which provisions site pages

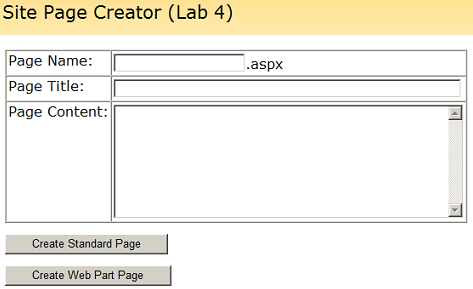
1. Launch Visual Studio. Open the project named **Lab4.sln** located inside the **\04\_SitePages\Lab** directory.
   1. This is a project that contains a Feature named **Lab4** that defines several page templates that are used to provision site page instances upon activation.
   2. Use the **Solution Explorer** to get an idea of the project's directory structure as well as what source files are included as part of this project. 
2. Look at the code inside the page template definitions for **Page01.aspx**, **Page02.aspx**, **Page03.aspx** and **WebPartPage.aspx** that are located in the **PageTemplates** directory within the **Features\Lab4** directory. This should give you a sense of what goes inside a simple page template.
   1. Note that all three Page0x.aspx files (note: x = the page #) utilize the **default.master** master page that most all site pages utilize which gives all of your custom pages a similar look and feel to the built in site pages (i.e. same navigational elements and images across the top and down the left side of the screen)
   2. Note that the location of the **MasterPageFile** "~masterurl/default.master" maps to the "C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\GLOBAL" directory.  This master page is used widely across many different Site Pages in WSS.
   3. Note that all three Page0x files have a **Content** area defined in them (asp:Content...>).  This area contains the unique items in these particular pages (outside of those defined in the master page)
   4. Note that **Page03.aspx** utilizes two Custom Controls (**UserControl1.ascx** and **UserControl2.ascx**) that are defined in your /TEMPLATE?CONTROLTEMPLATE/Lab4/ directory.
3. Open **elements.xml** and see how **File** elements are nested inside the **Module** element to automatically provision instances from these page templates upon Feature activation.
4. Open the **feature.xml** file and note that the **Lab4** Feature has been configured with a Feature receiver class named **Lab4.FeatureReceiver** that has event handlers that fire during activation and deactivation.
   1. Look at this event handler code inside **FeatureReceiver.cs** or **FeatureReceiver.vb**.
   2. You can see there is code in the **FeatureActivated** event handler that creates a new drop down menu to the current site's top-link bar that includes menu items to navigate to the site page instances that have been provisioned in **elements.xml**.
   3. There is also code inside the **FeatureDeactivated** event handler to remove these menu items and also to delete the directory named **SitePages** in which all the site pages have been provisioned.
5. Build the project. This will recompile the project's output assembly **Lab4.dll** and then it will run **Install.bat** which contains commands to install **Lab4.dll** in the GAC, to copy the Feature files into the WSS **FEATURES** directory and to install (or reinstall) the Feature itself. At this point, the Feature should be installed and ready for activation in any site within the current farm.
6. To test this feature you should either create a new top-level site or go to an existing site such as the one at **http://litwareinc.com** and activate the **Lab4** Feature.
   1. **Site Actions > Site Settings > Site Features** (Under Site Administration column)
   2. Locate the **Lab4 - Working with Site Pages** feature and click the **Activate** button
7. After the Feature has been activated, you should see a new drop down menu appear in the top-link bar of the current site:  
     
   
8. Use this menu to navigate to each of the pages.
   1. When you get to **Page04.aspx**, you should be able to see it is a Web Part Page. (Note: do not wait for the spinning green SharePoint loading symbol to finish loading the web part... go directly to step B instead).
   2. If you go into **Edit Mode**, you should be able to move the Web Part instance that is there between zones.  (notice that this web part is a Watch My Gears Run web part that is custom built to perpetually display the green spinning SharePoint Symbol...).
   3. You should also be able to add new Web Part instances just like with any other Web Part Page.
9. Launch the **SharePoint Designer** and open the site in which you just activated the **Lab4** Feature.
   1. **Start > SharePoint Designer 2007** (pinned to the start menu near the top)
   2. **File menu > Open Site...**
   3. In the **Site name:** text box type **http://litwareinc.com/**
   4. Click the **Open** button  
10. You should be able to verify that **SharePoint Designer** can recognize and open each of the site pages that have been provisioned by the **Lab4** Feature.
    1. In **SharePoint Designer** under the **Folder List**  click on **SitePages** to see the 4 pages that our feature provided.
    2. Open **Page01.aspx** in **SharePoint Designer** and make a simple custom change.
       1. In the **Contents of** '**SitePages**' area double click on **Page01.aspx** this will open Page01 for editing with SP Designer
       2. Notice how the Master Page is loaded (Loading Master Page...)  along with our unique content provided by Page01.  SharePoint Designer is about the only tool that can pull all of the disparate information together to allow us to edit the page as if these files were all stored in one central location (i.e. when you open a page in SP Designer it will not only traverse the physical directory structure on the Web server to find the default.master page but also pull Page01.aspx out of the SQL Content Database where it is stored and put these two back together so that we may edit them using a graphical view or code view. Compare this to our experience in Visual Studio where at best we only see the code view of our page fragment (i.e. just the code for the Page01.aspx file).
11. Edit your **PlaceholderMain (Custom)** by just clicking into this area and put your insertion point after the default text.  Add a line by pressing enter and type the text you see in the Oval. 
12. You may even edit bits of the master page area (left side and top of screen) by clicking in a master page area and selecting the arrow icon and selecting **Create Custom Content** .  However doing this will Un-ghost the Master Page area for this page and new changes to the master page will no longer be picked up on this page so the recommendation is to avoid making changes to the Master page area of a page using SharePoint Designer (**Note**: if you selected **Create Custom Content** then you should select the arrow icon **>** again and click on **Default to Master's content** and click **yes** to the next question about reverting the region to the Master page content.) 
13. Save your change.
    1. In SharePoint designer Click on the **File menu** and click **Save**
    2. Answer yes to the question about customizing a page from the site definition  
    3. Return to the browser and refresh **Page01.aspx**.
    4. Now return to **SharePoint Designer.**
    5. Under the **Folder List** (i.e. Click on the **Web Site** tab ) expand the **SitePages** Folder if necessary). You should be able to see the customization changes made by SharePoint Designer (you may have to expand the **View** menu and click **Refresh** or press the **F5** key on your keyboard to refresh this screen).
       1. Notice the blue icon next to the file in the **Folder List** tool window in SharePoint Designer indicating the page has been customized (e.g. unghosted).  This means that this version of Page01.aspx on this site is now independent of any other site collections feature activated Page01.aspx.  Therefore any change to the base Page01.aspx will NOT be seen on this Site Collections Page01 in the future...  
14. Now we will re-ghost this page.
    1. Right-click **Page01.aspx** and select **Reset to Site Definition**
    2. Click Y**es** to the Site Definition Page Warning (note that SharePoint will return the page to its uncustomized (e.g. ghosted) state. 
    3. Note that when we reset a site using SharePoint designer a backup copy of the changed page is created as a safety net.
    4. Note: Should you not want to keep this version you should delete it as it will be accessible via the newly created site link otherwise. In our case we will leave this alone as we may need to utilize it later.

# Exercise 2: Create a Site Page using Code

1. In **Internet Explorer** let's examine the Custom application page added by our feature: **SitePageCreator.aspx**.
   1. From the site you activated the feature on, drop down the **Site Actions** menu for the site, you should notice that there is a menu item to navigate to a custom application page.



* 1. Select the **Create Site Page** menu item from the **Site Actions** menu to navigate to the page. You can see it supplies three textbox controls which allow the user to enter a new page name, a page title and some content for the page body.



1. Your job is now to write code inside this page’s event handlers to take the user input from **SitePageCreator.aspx** to create a new site page on the fly.  When you are done your code should add the new site page to the **\SitePages** directory within the virtual file system of the current site. It should also update the drop down menu on the top link bar to add a new menu item which allows users to navigate to this new page.
   1. In Visual Studio open **C:\Student\Labs\04\_SitePages\Lab\C#\Lab4.sln,** and then open the file named **SitePageCreator.cs** or **SitePageCreator.vb.**
   2. **Next** locate the event handler method named **btnCreateStandardPage\_Click**.
   3. You need to accomplish several things to create a page on the fly here (note: you may use these detailed directions below or look at an existing page (like Page01.aspx to get a sense of what we need to write out here).
      1. In order to write things on the fly you need to utilize a stream writer and a memoryStream to write your html into

C#

MemoryStream stream = new MemoryStream();

StreamWriter writer = new StreamWriter(stream);

VB.Net

Dim stream As New MemoryStream()

Dim writer As New StreamWriter(stream)

* + 1. Next we need to utilize the writer to write out our @Page directive   
       (**Note**: although this code is broken up across multiple lines it **MUST BE ENTERED** on one physical line in order to run as written)

C#

writer.WriteLine(

"<%@ Page MasterPageFile='~masterurl/default.master'

meta:progid='SharePoint.WebPartPage.Document' %>");

VB.Net

writer.WriteLine(

"<%@ Page MasterPageFile='~masterurl/default.master'

meta:progid='SharePoint.WebPartPage.Document' %>")

* + 1. Next we need to utilize the writer to write out our asp: Content tag and then populate this tag with information taken from the SitePageCreator.aspx page (namely the Title field and the Page Content field)  
        (**Note**: this Content tag matches up to a **ContentPlaceHolder** tag defined in the default.master page file)

C#

writer.WriteLine("<asp:Content runat='server' ContentPlaceHolderID='PlaceHolderMain'>");

writer.WriteLine("<h3>" + txtPageTitle.Text + "</h3>");

writer.WriteLine(txtPageContent.Text);

writer.WriteLine("</asp:Content>");

VB.Net

writer.WriteLine("<asp:Content runat='server' ContentPlaceHolderID='PlaceHolderMain'>")

writer.WriteLine(("<h3>" + txtPageTitle.Text + "</h3>"))

writer.WriteLine(txtPageContent.Text)

writer.WriteLine("</asp:Content>")

* + 1. Now we need to flush out the writer to persist this information to the stream object

C#

writer.Flush();

VB.Net

writer.Flush()

* + 1. Next we need to take the site name provided by the user on the **SitePageCreator.aspx** page and create our physical page name adding this new page to our current Site’s web pages.

C#

string NewPageUrl = @"SitePages/" + txtPageName.Text + ".aspx";

this.Web.Files.Add(NewPageUrl, stream);

VB.Net

Dim NewPageUrl As String = "SitePages/" + txtPageName.Text + ".aspx"

Me.Web.Files.Add(NewPageUrl, stream)

* + 1. Now we should add navigational elements to our main page to allow easy access to this newly created page.

C#

// grab the existing site TopNavigationBar for our use

SPNavigationNodeCollection topNav = this.Web.Navigation.TopNavigationBar;

// create dropdown menu for custom site pages

foreach (SPNavigationNode node in topNav[0].Children) {

if(node.Title.Equals("Site Pages")) {

SPNavigationNode newNode = new SPNavigationNode(txtPageName.Text,NewPageUrl);

node.Children.AddAsLast(newNode);

}

}

VB.Net

' grab the existing site topNavigationBar for our use

Dim topNav As SPNavigationNodeCollection = Me.Web.Navigation.TopNavigationBar

' create dropdown menu for custom site pages

Dim node As SPNavigationNode

For Each node In topNav(0).Children

If node.Title.Equals("Site Pages") Then

Dim newNode As New SPNavigationNode(txtPageName.Text, NewPageUrl)

node.Children.AddAsLast(newNode)

End If

Next node

* + 1. Lastly we will utilize the **SPUtility** class to navigate directly to our newly created page

C#

SPUtility.Redirect(Web.Url + @"/" + NewPageUrl, SPRedirectFlags.Default, this.Context);

VB.Net

SPUtility.Redirect((Web.Url + "/" + NewPageUrl, SPRedirectFlags.Default, \_ Me.Context)

1. When you are done, rebuild the project. That will also run Install.bat to redeploy our changes and test your work. You should be able to use the application page named **SitePageCreator.aspx** to dynamically add new pages to the current site along with an associated navigation menu item.
   1. Navigate to your drop down menu **Site Actions > Create Site Page**
   2. Using this page make up some information: Name, Title, Content and enter it where appropriate
   3. Click the **Create Standard Page** Button
   4. You should now be on the newly created page.

# Exercise 3: Create a Web Part Page using Code

1. Within the source file named **SitePageCreator.cs** or **SitePageCreator.vb**, locate the event handler method named **btnCreateWebPartPage\_Click**. Your job in this exercise is to write code inside this event handler to take the user input from **SitePageCreator.aspx** and to create a new Web Part Page on the fly just as you did in the previous exercise. However, instead of creating a new site page, you should clone the Web Part Page instance named **WebPartPage.aspx**

C#

// clone Web Part Page template to create new Web Part Page

string NewPageUrl = @"SitePages/" + txtPageName.Text + ".aspx";

SPFile template = Web.GetFile(@"Template\WebPartPage.aspx");

template.CopyTo(NewPageUrl);

VB.Net

' clone Web Part Page template to create new Web Part Page

Dim NewPageUrl as string = "SitePages/" + txtPageName.Text + ".aspx"

Dim template as SPFile = Web.GetFile("Template\WebPartPage.aspx")

template.CopyTo(NewPageUrl)

1. After creating the page, use the **SPLimitedWebPartManager** class to add a new **ContentEditor** Web Part to the page that has the title and body content that the user entered on the page.

C#

// add Content Editor Web Part

SPFile NewPage = Web.GetFile(NewPageUrl);

SPLimitedWebPartManager mgr;

mgr = NewPage.GetLimitedWebPartManager(PersonalizationScope.Shared);

ContentEditorWebPart wp1 = new ContentEditorWebPart();

wp1.Title = txtPageTitle.Text;

wp1.ChromeType = PartChromeType.TitleOnly;

wp1.AllowClose = false;

XmlDocument doc = new XmlDocument();

string ns1 = "http://schemas.microsoft.com/WebPart/v2/ContentEditor";

XmlElement elm = doc.CreateElement("Content", ns1);

elm.InnerText = txtPageContent.Text;

wp1.Content = elm;

VB.Net

' add Content Editor Web Part

Dim NewPage As SPFile = Web.GetFile(NewPageUrl)

Dim mgr As SPLimitedWebPartManager

mgr = NewPage.GetLimitedWebPartManager(PersonalizationScope.Shared)

Dim wp1 As New ContentEditorWebPart()

wp1.Title = txtPageTitle.Text

wp1.ChromeType = PartChromeType.TitleOnly

wp1.AllowClose = False

Dim doc As New XmlDocument()

Dim ns1 As String = "http://schemas.microsoft.com/WebPart/v2/ContentEditor"

Dim elm As XmlElement = doc.CreateElement("Content", ns1)

elm.InnerText = txtPageContent.Text

wp1.Content = elm

1. Now you will add this newly created web part to the Left Zone of the existing page using the **LimitedWebPartManager** (mgr) that we created earlier. Also, you will specify that this item should be displayed first in that zone.

C#

// add Web Part to Left Zone

mgr.AddWebPart(wp1, "Left", 0);

VB.Net

' add Web Part to Left Zone

mgr.AddWebPart(wp1, "Left", 0)

1. Next you will add this page to the site's **TopNavigationBar** in a node entitled "**Site Pages**". Make certain that this new page is appended to the end of the existing page list.

C#

// add navigation node

SPNavigationNodeCollection topNav = this.Web.Navigation.TopNavigationBar;

foreach (SPNavigationNode node in topNav[0].Children) {

if (node.Title.Equals("Site Pages")) {

SPNavigationNode newNode = new SPNavigationNode(txtPageName.Text, NewPageUrl);

node.Children.AddAsLast(newNode);

}

}

VB.Net

' add navigation node

Dim topNav As SPNavigationNodeCollection = Me.Web.Navigation.TopNavigationBar

Dim node As SPNavigationNode

For Each node In topNav(0).Children

If node.Title.Equals("Site Pages") Then

Dim newNode As New SPNavigationNode(txtPageName.Text, NewPageUrl)

node.Children.AddAsLast(newNode)

End If

Next node

1. Now, just like earlier, we will utilize the **SPUtility** class to navigate directly to our newly created page.

C#

// navigate directly to the newly created Web Part Page

SPUtility.Redirect(Web.Url + @"/" + NewPageUrl, SPRedirectFlags.Default, this.Context);

VB.Net

' navigate directly to the newly created Web Part Page

SPUtility.Redirect(Web.Url + "/" + NewPageUrl, SPRedirectFlags.Default, Me.Context)

1. When you are done, rebuild the project, which will also run Install.bat to re-deploy our changes and test your work. You should be able to use the application page named **SitePageCreator.aspx** to dynamically add new Web Part pages to the current site along with an associated navigation menu item.
   1. Navigate to your drop down menu **Site Actions > Create Site Page**
   2. Using this page make up some information: Name, Title, Content and enter it where appropriate
   3. Click the **Create Web Part** **Page**
   4. You should now be on the newly created Web Part page.