## Pages and Navigation

**Lab Time**: 45 minutes

**Lab Folder**: C:\Student\Labs\Pages

**Lab Overview**: In the following lab, you create two SharePoint solutions. The first solution will be named **WingtipFarmPages** and will demonstrate techniques for creating pages in a SharePoint site that can only be used when deploying the solution as a farm solution. The second solution named **WingtipSandboxedPages** will demonstrate techniques that work within the sandbox resulting in a solution that can be deployed as either a farm solution or as a sandboxed solution.

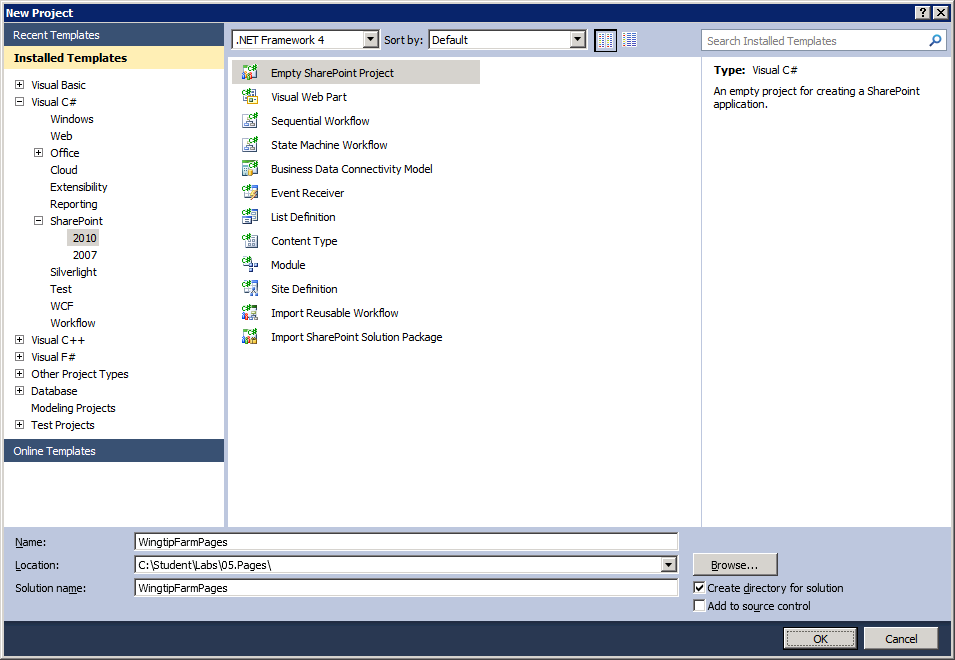
Lab Setup Requirements

* Before you begin this lab, you must run the batch file named **SetupLab.bat**. This batch file creates a new blank site collection at the location **http://intranet.wingtip.com/sites/Pages**. This is the site you will use to test and debug the code you are going to write with the Visual Studio 2010 SharePoint Tools.

### Exercise 1: Creating Pages in a Farm Solution

In this exercise you will create a few components that work in a farm solution but not in a sandboxed solution. These types of components include Application pages and User Controls. You will also add CustomAction items to provide the user with means to navigate to your new application pages.

1. Open **Visual Studio 2010** and choose to create a new project. Complete the dialog that appears using the following information.
   1. Pick the **Empty SharePoint Project** template that you can find in the **Visual C# » SharePoint » 2010** template group.
   2. Name the project **WingtipFarmPages**
   3. Complete the dialog using the **OK** button.

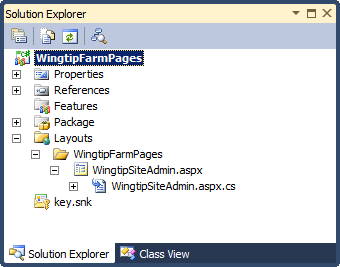


1. Complete the SharePoint Customization wizard that appears using the following information.

**Debugging site:** http://intranet.wingtip.com/sites/Pages

**Deploy as a farm solution**: selected

1. Add a new Application Page to the project.
2. Inside the Solution Explorer, right click the **WingtipFarmPages** project node and select the menu item **Add » New Item**.
3. Create a new item named **WingtipSiteAdmin** based on the **Application Page** project item template.
4. Once the new application page named WingtipSiteAdmin.aspx has been created, take note of where it was created inside the project structure. It has been created inside SharePoint Root folder nested in a child folder named **WingtipFarmPages**. You should also notice that the application page has a code-beside file named WingtipSiteAdmin.aspx.cs.



* 1. The WingtipSiteAdmin.aspx page should already be open inside of a code window. If the page is not open, double-click on it in the Solution Explorer to open it.
  2. At the top of the page, you should see a Page directive. Above this, you will see several other directives named Assembly, Import and Register. You are going to leave all these directives at the top of the page exactly how they are. However, you will delete all four of the existing Content tags and replace them with the code below. Note you can copy and paste this code from [[Lab Files]]\StarterFiles\WingtipSiteAdmin\_Starter.txt.

<asp:Content ID="Title" ContentPlaceHolderID="PlaceHolderPageTitle" runat="server">

Wingtip Site Administration

</asp:Content>

<asp:Content ID="TitleInTitleArea" ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea" runat="server" >

Wingtip Site Administration

</asp:Content>

<asp:Content ID="PageHead" ContentPlaceHolderID="PlaceHolderAdditionalPageHead"   
 runat="server">

<style type="text/css">

#MSO\_ContentTable{ padding-left:10px; }

</style>

</asp:Content>

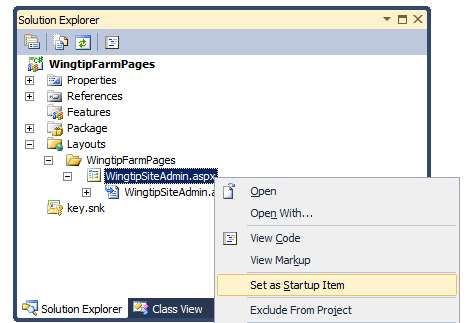
<asp:Content ID="Main" ContentPlaceHolderID="PlaceHolderMain" runat="server">

<h1>Wingtip Site Configuration</h1>

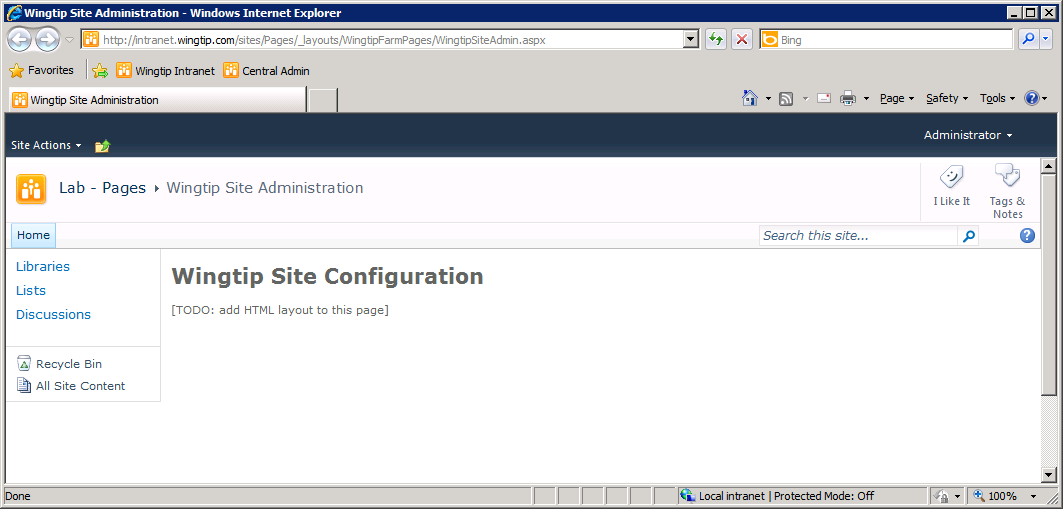
<p>[TODO: add HTML layout to this page]</p>

</asp:Content>

1. Test your new application page.
2. Right-click on **WingtipSiteAdmin.aspx** in the Solution Explorer and select the **Set as Startup Item** menu command.



* 1. Test the project by selecting the **Debug » Start Debugging** menu command. Note that pressing the **[F5]** keystroke is the shortcut key for running the **Debug » Start Debugging**. After a few second, you should see your application page running inside the browser.



* 1. Close the browser window to stop the debugger. Return to Visual Studio 2010 and navigate back to the code window for WingtipSiteAdmin.aspx.
  2. Now you will modify the HTML inside the Content tag for PlaceHolderMain. Modify the content of PlaceHolderMain with the following code which you can copy and paste from [[Lab Files]]\StartFiles\PlaceHolderMain\_Starter.txt.

<asp:Content ID="Main" ContentPlaceHolderID="PlaceHolderMain" runat="server">

<h1>Wingtip Site Configuration</h1>

<asp:RadioButtonList ID="lstSiteIcon" runat="server">

<asp:ListItem Text="Default Site Icon" />

<asp:ListItem Text="Wingtip Site Icon" />

</asp:RadioButtonList>

<div style="margin: 10px;" >

<p><asp:Button ID="cmdUpdate" runat="server"

Text="Update" CssClass="ms-ButtonHeightWidth" /></p>

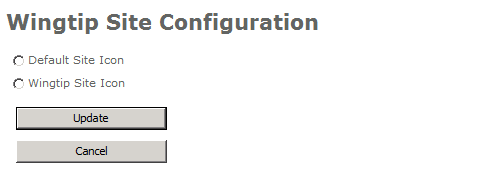
<p><asp:Button ID="cmdCancel" runat="server"

Text="Cancel" CssClass="ms-ButtonHeightWidth" /></p>

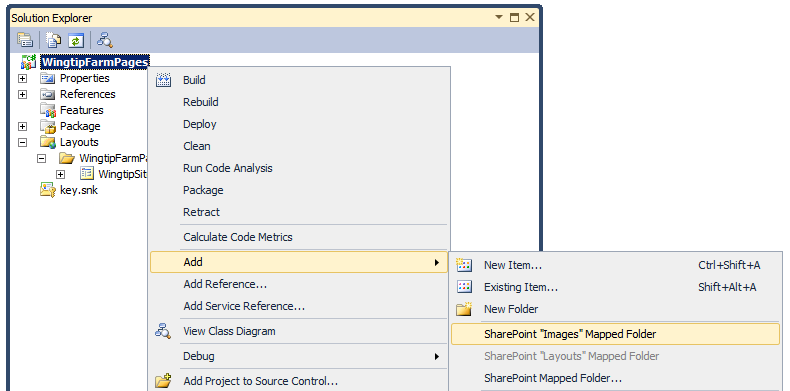
</div>

</asp:Content>

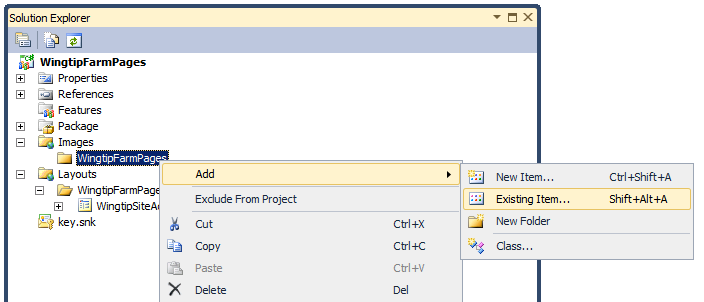
* 1. Run WingtipSiteAdmin.aspx again in the debugger. Your main section of your application page should look like this.



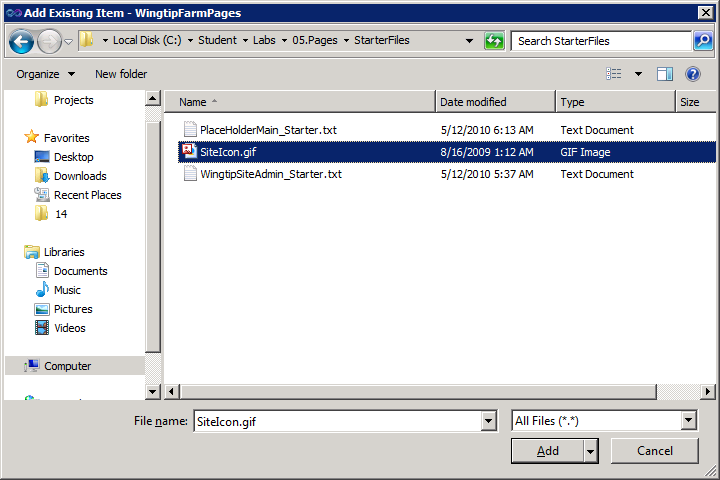
1. Add a new \*.GIF file to the **Images** folder.
2. Right-click on WingtipFarmPages in the Solution Explorer and select the **Add » SharePoint Images Mapped Folder** command.



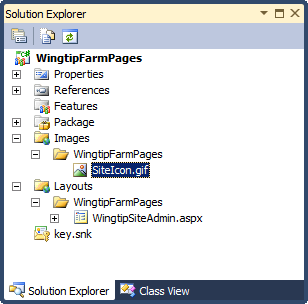
* 1. You should notice that the **Images** directory has been created with an inner directory named **WingtipFarmPages**. Right click on the **WingtipFarmPages** folder and select the **Add » Existing Item** menu command.



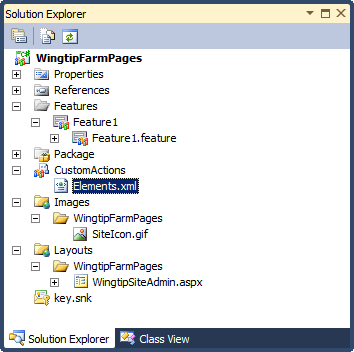
* 1. When you run the **Add » Existing Item** menu command, you will be prompted with the Add Existing Item dialog. Navigate to [[LAB FILES]]\StarterFiles folder and select the image file named SiteIcon.gif.



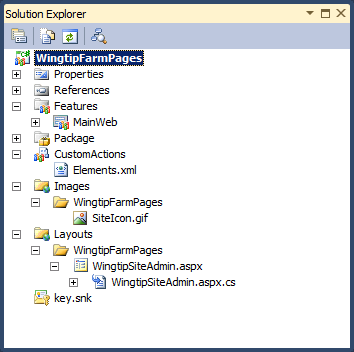
* 1. The **WingtipFarmPages** project should now have the image file configured to be deployed inside the **Images** folder in a child folder named **WingtipFarmPages**.



1. Create a **CustomAction** element which will allow the user to navigate to the application page.
2. Right-click on the **WingtipFarmPages** project in the Solution Explorer and select the **Add >> New Item** menu command.
3. Create a new SharePoint Project Item named **CustomActions** from the **Empty Element** project template. Note that the addition of this SharePoint Item will trigger the SharePoint Developer Tools to add a feature which is named **Feature1** by default.



* 1. Right-click on the **Feature1** node in the **Solution Explorer** and select the **Rename** command. Rename the feature to **MainWeb** to indicate that this is the main feature for this project and that it will activate at site level as opposed to site collection level.



* 1. Open the **MainWeb** feature in the Feature Designer. Change the **Title** of the feature to **Wingtip Farm Pages**. Close the Feature Designer.
  2. Open the Elements.xml file inside the **CustomActions** project item and add the CAML shown below this step. Note that you can copy and paste from [[Lab Files]]\StartingFiles\CustomAction\_Starter.txt.

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

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

<CustomAction

Id="WingtipSiteAdmin"

GroupId="SiteAdministration"

Location="Microsoft.SharePoint.SiteSettings"

Rights="ManageWeb"

Sequence="1"

Title="Wingtip Site Admin"

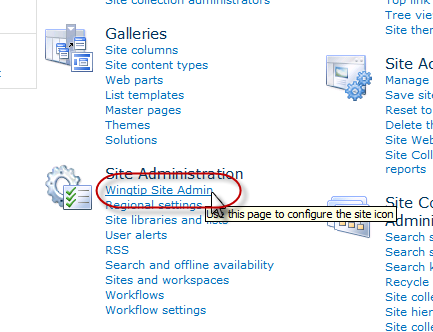
Description="Use this page to configure the site icon" >

<UrlAction Url="~site/\_layouts/WingtipFarmPages/WingtipSiteAdmin.aspx" />

</CustomAction>

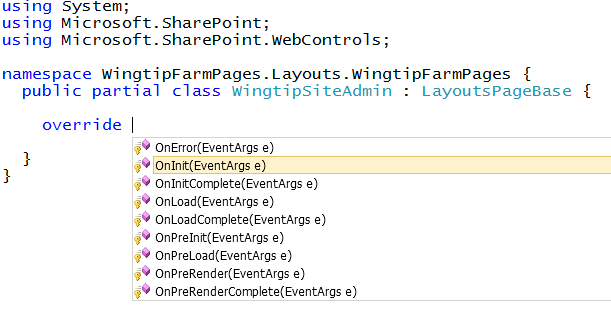
</Elements>

* 1. Start the project in the debugger and navigate to the **Site Settings** page. You should be able to see the link created by the **CustomAction** element as the first link in the **Site Administration** section.

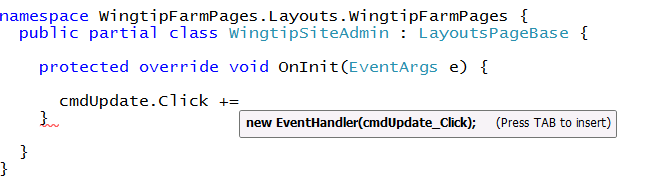


* 1. Click the **Wingtip Site Admin** link to test it and to make sure it works properly. Clicking this link should navigate the browser to WingtipSiteAdmin.aspx.

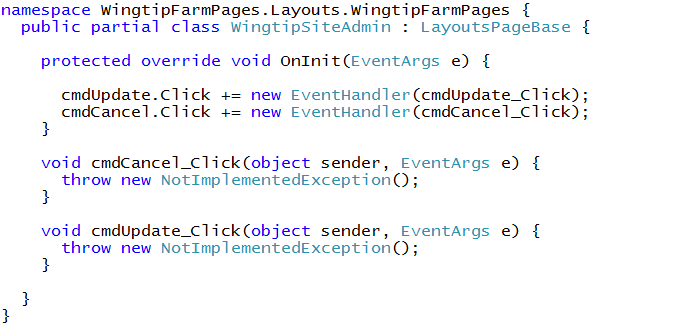
1. Write the code behind WingtipSiteAdmin.aspx.
2. Open the source file named WingtipSiteAdmin.aspx.cs. You should see there is a class defined inside named WingtipSiteAdmin which inherits from LayoutsPageBase.
3. The WingtipSiteAdmin class already contains a method named Page\_Load. Delete this method so the class definition is empty.
4. Add an override implementation of the OnInit() method. The easiest way to accomplish this when using C# is to type override inside the class and then to allow IntelliSense to provide you with methods available for overriding. Select the OnInit() method as shown below and press **[TAB]** key to accept.



* 1. Modify the OnInit() method to add event handlers for the two command buttons named cmdUpdate and cmdCancel. The easiest way to accomplish this in C# is by typing the control name and the event name followed by a +=. At this point, Visual Studio will recognize that you want to add an event handler if you press the **[TAB]** key twice.



* 1. After this step, you should have created and wired up event handlers for the two method.



* 1. Add the following using statement to the top of WingtipSiteAdmin.aspx.cs.

using Microsoft.SharePoint.Utilities;

* 1. Update the command handler for the cmdCancel button with the code shown below. This code redirects the user back to the Site Settings page.

SPUtility.Redirect("settings.aspx",

SPRedirectFlags.RelativeToLayoutsPage,

this.Context);

* 1. Update the command handler for the cmdUpdatebutton with the code shown below. This code updates the site icon and then redirects the user back to the Site Settings page.

void cmdUpdate\_Click(object sender, EventArgs e) {

if (lstSiteIcon.Text == "Wingtip Site Icon") {

this.Web.SiteLogoUrl = "/\_layouts/images/WingtipFarmPages/SiteIcon.gif";

this.Web.Update();

}

else {

this.Web.SiteLogoUrl = "";

this.Web.Update();

}

SPUtility.Redirect("settings.aspx",   
 SPRedirectFlags.RelativeToLayoutsPage,   
 this.Context);

}

* 1. As a final step, you must add the code to initialize the control named lstSiteIcon with the current value setting. Create an override for the OnPreRender() method and add the following code.

protected override void OnPreRender(EventArgs e) {

if(string.IsNullOrEmpty(this.Web.SiteLogoUrl)){

lstSiteIcon.Items.FindByText("Default Site Icon").Selected = true;

}

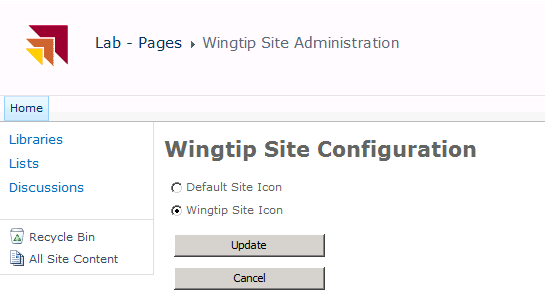
else{

lstSiteIcon.Items.FindByText("Wingtip Site Icon").Selected = true;

}

}

1. Test your work:
2. You should now be able to navigate to WingtipSiteAdmin.aspx page from the Site Settings page and change the site icon back and forth between the standard site icon and the custom Wingtip icon.



In this exercise you created an application page that contained a few buttons and a code behind file.

### Exercise 2: Creating Pages for a Sandboxed Solution

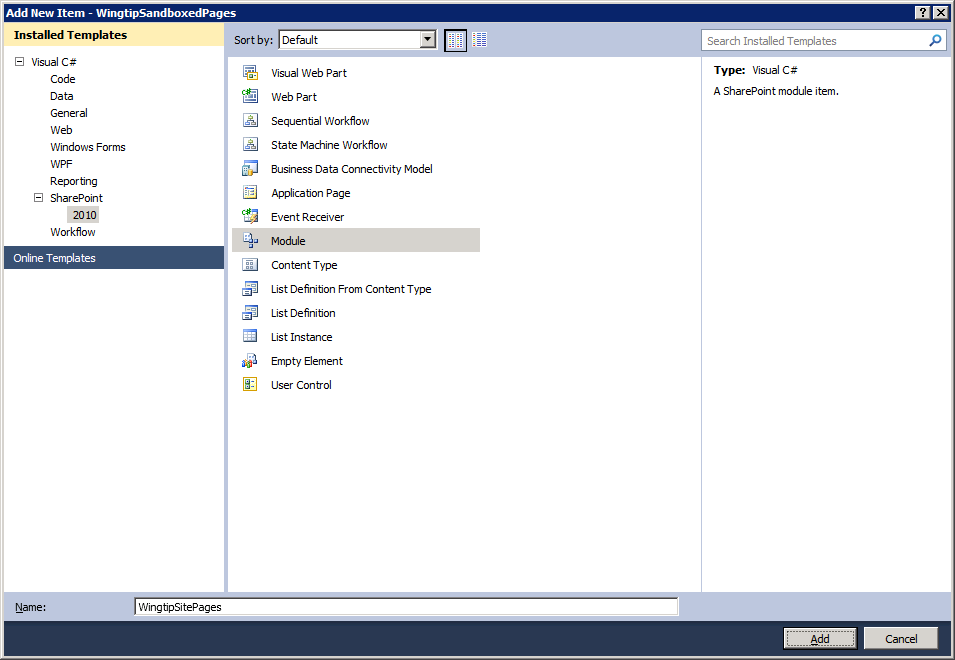
In this exercise you will create new site pages and add navigation elements to the Top Navigation bar.

1. Open **Visual Studio 2010** and choose to create a new project. Complete the dialog that appears using the following information.
2. Pick the **Empty SharePoint Project** template that you can find in the **Visual C# » SharePoint » 2010** template group.
   1. Name the project **WingtipSandboxedPages**
   2. Complete the dialog using the **OK** button.
3. Complete the SharePoint Customization wizard that appears using the following information:

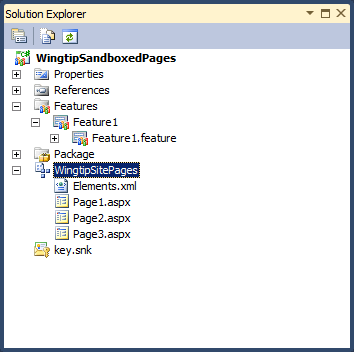
**Debugging site:** http://intranet.wingtip.com/sites/Pages

**Deploy as a sandboxed solution**: selected

1. In this exercise, you will use site page templates to provision page instances during feature activation. Start by creating a new **Module** named **WingtipSitePages**. Note that the addition of this SharePoint Item will trigger the SharePoint Developer Tools to add a feature which is named **Feature1** by default.



1. After the **WingtipSitePages** module is created, its folder will initially contain the elements.xml file and a text file named Sample.txt. Delete Sample.txt.
2. Inside the [[LAB FILES]]\StarterFiles folder for this lab you will find three page templates named Page1.aspx, Page2.aspx and Page3.aspx. Add all three of these page templates into the **WingtipSitePages** module.



1. Open each of these page templates and take a moment to inspect the code in each one.
2. Rename **Feature1** to **MainSite**.
3. Open the **MainSite** feature in the Feature Designer and change its scope from **Web** to **Site**. One of the primary motivations for changing the feature scope to the site collection is that it will cause the feature to be automatically activated when the containing solution is activated as a sandboxed solution.
4. Add a feature receiver to add a node to the TopNav bar.
5. Right-click on the node for the **MainSite** feature and click **Add Event Receiver**.
6. Add methods for **FeatureActivated** and **FeatureDeactivating**.
7. Add a using statement for the **Microsoft.SharePoint.Navigation** namespace.
8. Add an implementation for **FeatureActivated** to add TopNav links for the three pages.

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

SPSite siteCollection = (SPSite)properties.Feature.Parent;

if (siteCollection != null) {

SPWeb site = siteCollection.RootWeb;

// create dropdown menu for custom site pages

SPNavigationNodeCollection topNav = site.Navigation.TopNavigationBar;

topNav.AddAsLast(new SPNavigationNode("Page 1", "WingtipSitePages/Page1.aspx"));

topNav.AddAsLast(new SPNavigationNode("Page 2", "WingtipSitePages/Page2.aspx"));

topNav.AddAsLast(new SPNavigationNode("Page 3", "WingtipSitePages/Page3.aspx"));

}

}

* 1. Add an implementation of FeatureDeactivating to delete the folder into which the site pages were created and to delete the custom links on the Top Navigation.

public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {

SPSite siteCollection = (SPSite)properties.Feature.Parent;

SPWeb site = siteCollection.RootWeb;

try {

// delete folder of site pages provisioned during activation

SPFolder sitePagesFolder = site.GetFolder("WingtipSitePages");

sitePagesFolder.Delete();

}

catch { }

SPNavigationNodeCollection topNav = site.Navigation.TopNavigationBar;

for (int i = topNav.Count - 1; i >= 0; i--) {

if (topNav[i].Url.Contains("WingtipSitePages")) {

// delete node

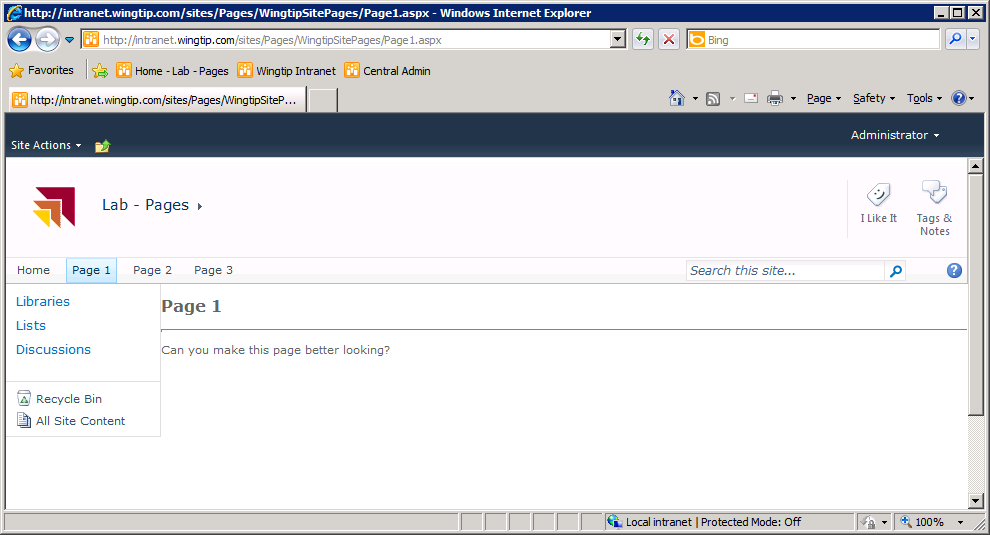
topNav[i].Delete();

}

}

}

1. Test your work:
2. Press the **[F5]** to start the debugger which will take you to the home page of the test site. You should see links that allow you to navigate to any of the three new site pages.



* 1. Stop the debugger and return to Visual Studio.

1. Now you will add some CSS style rules to Page1.aspx to make it look better.
2. Open the source file for the page template named Page1.aspx.
3. Add a new content tag for PlaceHolderAdditionalPageHead.
4. Add some CSS styles to make Page1.aspx look better. You can copy and paste the text for these styles from Page1\_Styles\_Starter.txt in [[LAB FILES]]\StarterFiles folder.

<asp:Content ContentPlaceHolderId="PlaceHolderAdditionalPageHead" runat="server">

<style type="text/css">

#MSO\_ContentTable {padding-top: 2px;padding-left:8px;}

.s4-specialNavLinkList a.s4-rcycl {display:none;}

.s4-specialNavLinkList LI {min-height:0px;}

h2 {color:Blue;}

p {color:Green;}

</style>

</asp:Content>

* 1. Start the debugger and examine Page1.aspx in the test site test your work. You should see that the styles you have added effect what the page looks like. Feel free to experiment by adding other CSS style rules to this page.

1. Prepopulate Page2.aspx with Web Parts.
2. Examine the page template named Page2.aspx. This is a simple starting point for a Web Part page. Currently, however, the page instance for Page2.aspx is created with no Web Parts on it.
   1. Open the elements.xml file for the **WingtipSitePages** Module. Locate the File element used to provision to page instance named Page2.aspx. Now open the File element so that you can add AllUserWebPart elements inside in the next step.

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

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

<Module Name="WingtipSitePages">

<File Path="WingtipSitePages\Page1.aspx" Url="WingtipSitePages/Page1.aspx" />

<File Path="WingtipSitePages\Page2.aspx" Url="WingtipSitePages/Page2.aspx" >

<!-- TODO: Add AllUsersWebPart elements here -->

</File>

<File Path="WingtipSitePages\Page3.aspx" Url="WingtipSitePages/Page3.aspx" />

</Module>

</Elements>

* 1. Locate the starter file named AllUsersWebPart\_Starter.txt in [[LAB FILES]]\StarterFiles folder. Open the file up and inspect the two AllUserWebPart elements inside.
  2. Copy and paste the contents of AllUsersWebPart\_Starter.txt into the File element for Page2.aspx.

<File Path="WingtipSitePages\Page2.aspx" Url="WingtipSitePages/Page2.aspx" >

<!-- Add a Web Part to left zone -->

<AllUsersWebPart WebPartZoneID="Left" WebPartOrder="0">

<![CDATA[

<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2"

xmlns:cewp="http://schemas.microsoft.com/WebPart/v2/ContentEditor">

<!— serialized web part content omited from this listing for readability -->

</WebPart>

]]>

</AllUsersWebPart>

<!-- Add a Web Part to right zone -->

<AllUsersWebPart WebPartZoneID="Right" WebPartOrder="0">

<![CDATA[

<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2"

xmlns:iwp="http://schemas.microsoft.com/WebPart/v2/Image">

<!-- serialized web part content omited from this listing for readability -->

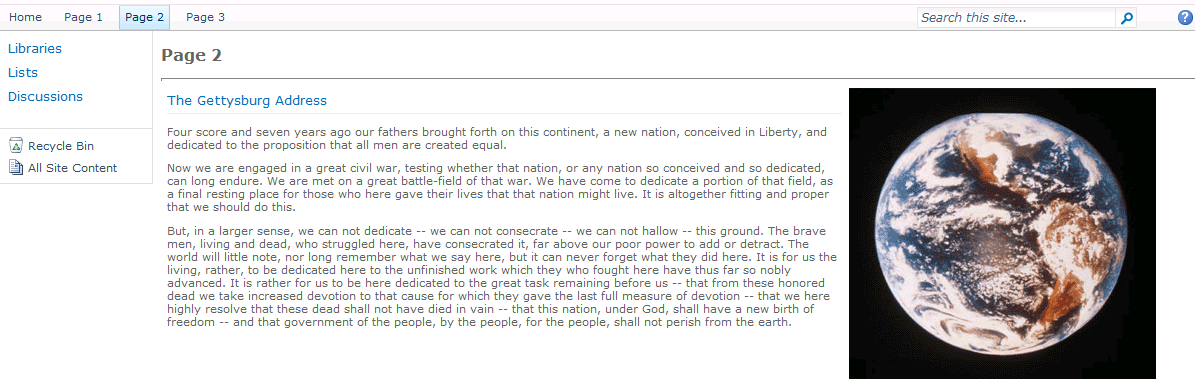
</WebPart>

]]>

</AllUsersWebPart>

</File>

* 1. Now, test your work. Start the debugger and navigate to Page2.aspx. You should now see two Web Parts on the page.



* 1. Stop the debugger and return to Visual Studio.

1. Work with the ECMAScript used in Page3.aspx.
2. Open the page template for Page3.aspx. Examine the page to see how it defines JavaScript methods and wire up client-side event handlers to HTML controls defined by the DOM.
   1. Start the debugger and test Page3.aspx. When you click the two command buttons on the page, you should see their code execute.
3. If you still have time, experiment by modifying the JavaScript in the two methods named MyHandler and MyOtherHandler.

In this exercise you created custom site pages that were deployed with a sandbox solution.