## Developing a Custom Branding Solution

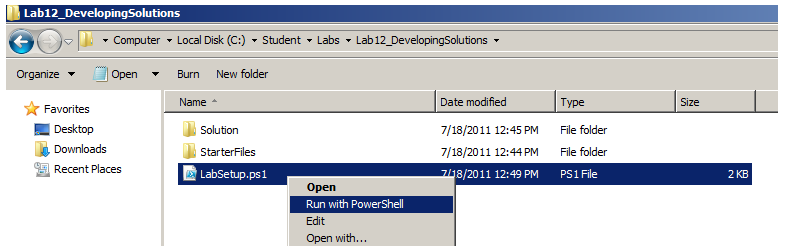
**Lab Time**: 60 minutes

### Exercise 1: Creating a new Site Collection for Testing Purposes

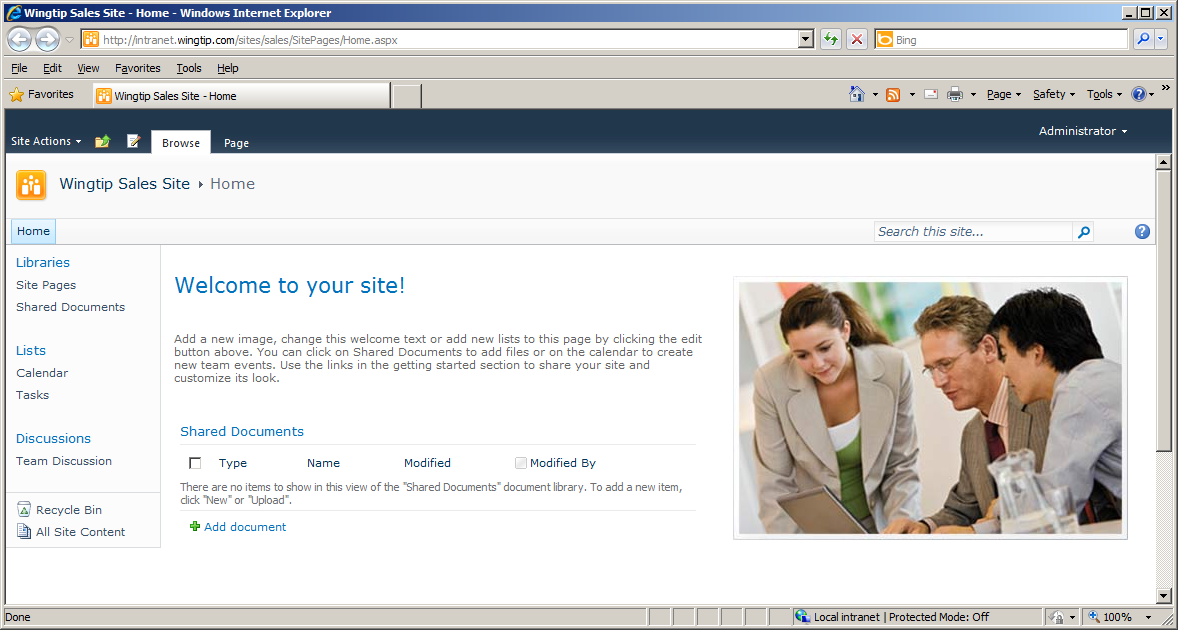
In this exercise you will run a Windows PowerShell script to create a new site collection for testing your custom branding solution. More specifically, you will run a Windows PowerShell script to create a new site collection which has a team Site as its top-level site. Next, you will created two child sites so that you will have a site collection with a site hierarchy to test your custom branding solution while you are developing it.

The steps in this lab assume that you are working on a SharePoint development workstation that is configured as a single-server farm and that this farm already has a Web Application that has been created at the root URL of **http://intranet.wingtip.com**. If you are working on a local SharePoint farm which has a Web Application with a different root URL, you must open the PowerShell script mentioned below and edit the URL before running it.

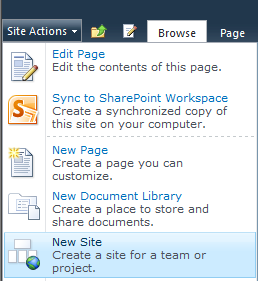
1. Using the Windows Explorer, look inside the lab directory which is located at a path of **C:\Student\Labs\Lab12\_DevelopingSolutions**. Inside this directory, you should be able to see a batch file named **LabSetup.ps1**. Right-click the file **LabSetup.ps1** and click the command **Run with PowerShell** to run the script that will create a new site collection that you will use to test your custom branding solution.



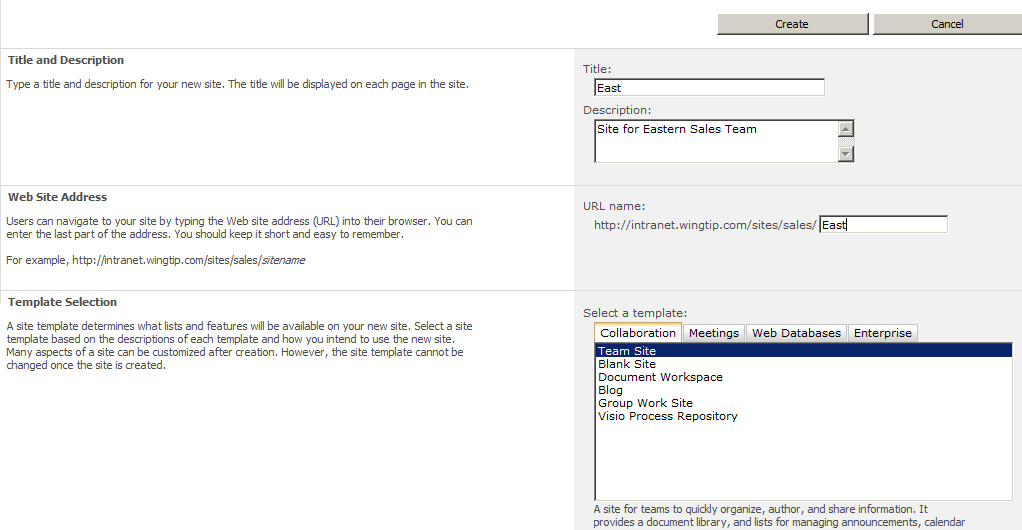
1. After the PowerShell script has created the new test site, it should then launch a new session of the Internet Explorer and navigate to the new Team Site at the root which has a title of **Wingtip Sales Site**. The site should be located at an URL of **http://intranet.wingtip.com/sites/sales**.



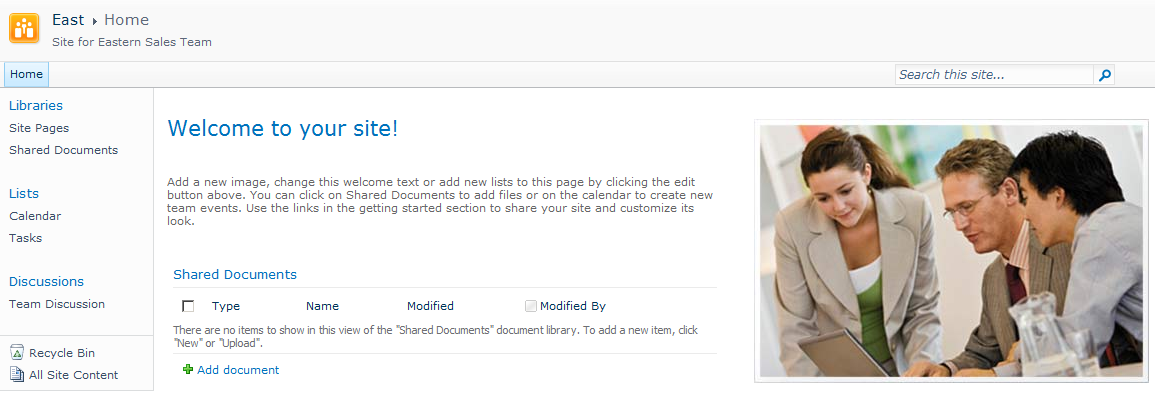
1. Now you will use the browser to create a new child site. Begin by dropping down the **Site Actions** menu and clicking the **New Site** command. When you do this, you will be redirected to the **New SharePoint Site** page.



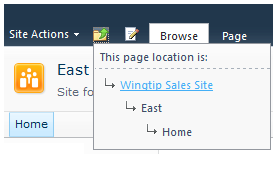
1. You must now fill in information in the **New SharePoint Site** page to give your new site a title and an URL. Give the site a **Title** of **East** and an **URL** of **East** which is relative to the URL of the parent site. Select **Team Site** as the site template as shown in the screenshot below. You can also enter a **Description** for the site if you would like. You can leave all other input controls on the page with their default values. Click the **Create** button to create a new child site.



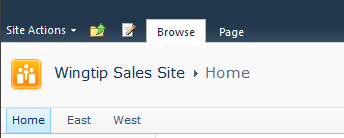
1. At this point you should have created a new child site with a **Title** of **East**.



1. Now you will create a second child site. However, before you do you should first navigate back to the top-level site using the breadcrumb control. This will ensure the second child site you create will be a sibling site to the child site you just created.



1. Now go through the exact same set of steps to create a second child site. The only difference is that the second child site should be given a **Title** and an **URL** of **West** whereas you gave the first child site a **Title** and an **URL** or **East**.
2. When you are done creating the **West** child site, navigate to the home page of the top-level site. You should see links in the Top Nav bar that allow users to navigate to the two new child sites.

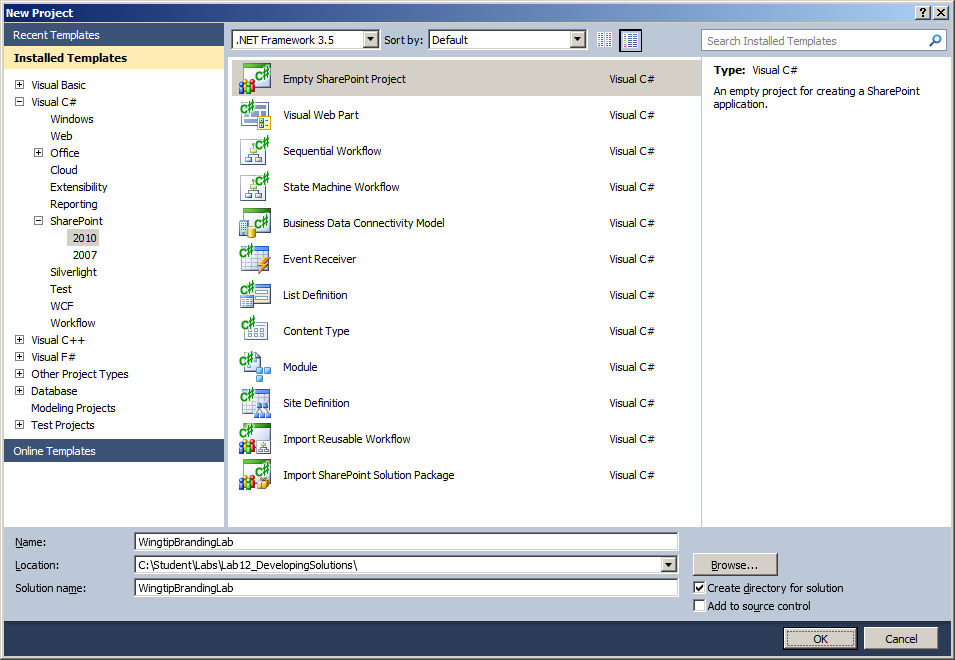


1. At this point, you have finished creating the site collection with a site hierarchy for testing purposes. You can now move on to the next exercise where you will begin to develop a custom branding solution with Visual Studio 2010.

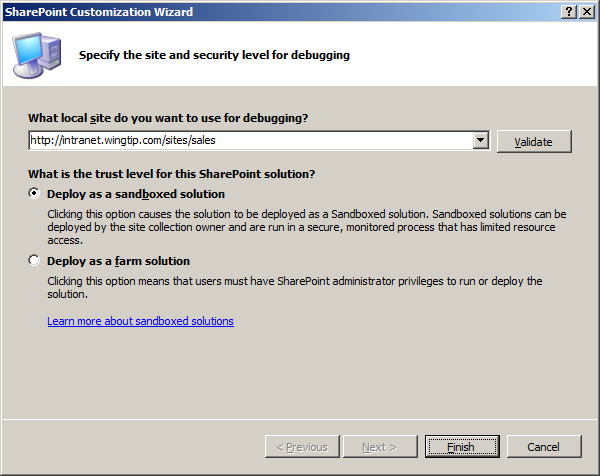
### Exercise 2: Creating a New SharePoint Project with a Feature

In this exercise you will move through the first steps of creating and testing a new SharePoint project in Visual Studio 2010. By the end of this exercise you will have created a feature which activates at the level of the site collection and you will have tested it by deploying the project and seeing that the feature has been activated in the site collection you have created for your testing.

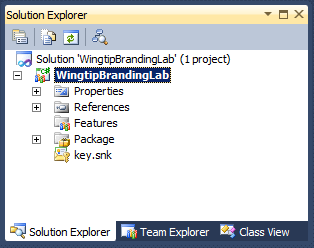
1. Launch Visual Studio 2010.
2. Create a new Visual Studio project based on the project template named **Empty SharePoint Project**. Give the new project a name of **WingtipBrandingLab**. Create the new project in the directory for this lab which is **C:\Student\Labs\Lab12\_DevelopingSolutions**. Click the **OK** button to move to the next step.



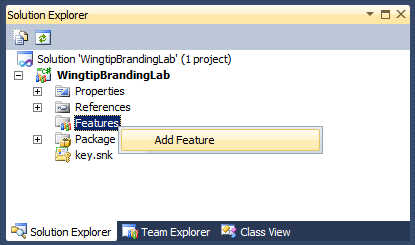
1. When you click the OK button on the New Project dialog, the **SharePoint Customization Wizard** appears and prompts you to provide an URL to a local SharePoint test site and to select either **Deploy as a sandboxed solution** or **Deploy as a farm solution** for testing.
   1. For the local test site, enter an URL of **http://intranet.wingtip.com/sites/sales**.
   2. Select **Deploy as a sandboxed solution**, as shown in the following screenshot.
   3. When you are done, click the **Finish** button to create the new project.



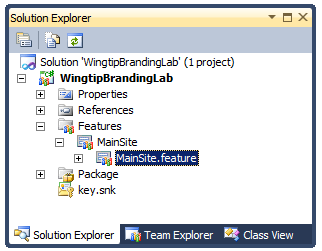
1. Examine the SharePoint project that has been created. At this point the **Features** node should be empty.



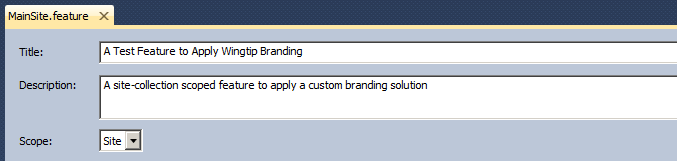
1. Add a feature to the project by right-clicking the **Features** node and clicking the **Add Feature** command.



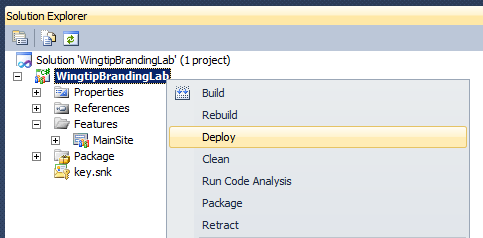
1. The new feature that has been created has an initial name of Feature1. Right-click on the node for this feature and click the **Rename** command. Give the new feature a name of **MainSite**. This name has been selected to convey the feature will have scope equal to Site which means it will activate at the level of the site collection.



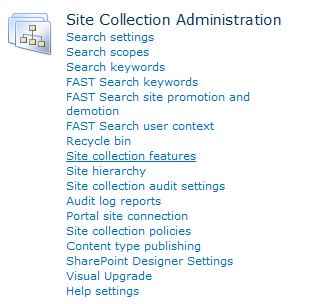
1. Double click on the feature named **MainSite** to open it in the Feature Designer if it is not already open. In the Feature Designer, modify the **Title** to **A Test Feature To Apply Wingtip Branding**. Add a Description such as the one shown in the following screenshot. Make sure you use the drop down box to modify the value of the **Scope** property from the default value of **Web** to a new value of **Site**. This change will ensure the feature activates at the level of the site collection.



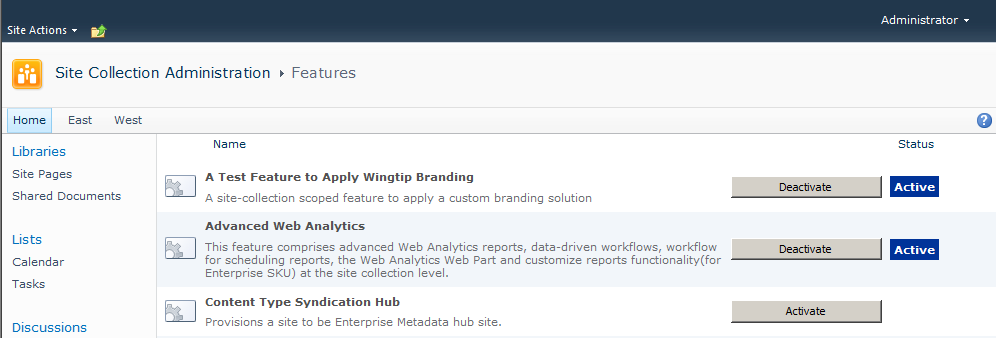
1. While this new feature doesn’t actually do anything yet, you can still test it to make sure that Visual Studio 2010 is able to activate it in your test site. Right-click on the project node named **WingtipBrandingLab** and click the **Deploy** command. When you run this command, Visual Studio builds the **MainSite** feature into a solution package named **WingtipBrandingLab.wsp** and then uploads the solution and activates it within the testing site collection.



1. After running the **Deploy** command, return to the test site in the browser. This is the site located at **http://intranet.wingtip.com/sites/sales**. Make sure you are at the top-level site. Drop down the **Site Actions** menu and click the **Site Setting** command to navigate to the Site Setting page. Find the group of links with the caption **Site Collection Administration**. Within the group, locate and click the link with the caption of Site collection features to navigate to the Site Collection Administration page for Features.



1. On the Site Collection Administration page for features you should be able to verify that your feature is showing and that it is in an activated state. Note that this page does not show the names of features such as **MainSite**. Instead, it displays a features **Title** property and **Description** property.



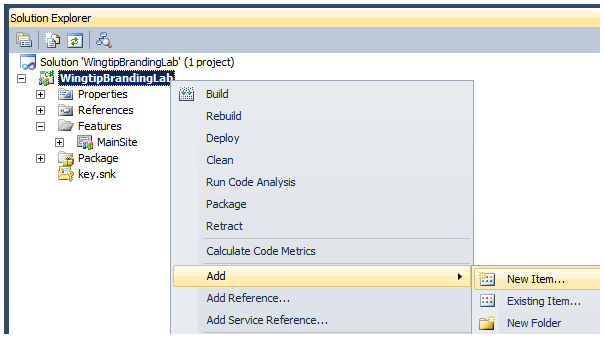
1. You are now at a point where you can add new elements to your feature and test them by simply running the **Deploy** command. Each time you run the **Deploy** command, Visual Studio 2010 will deactivate and remove the previous version and then uploaded and activate the updated version. This makes it fast and easy to add and edit branding files such as master pages and CSS files because you can quickly test your work at any point.

### Exercise 3: Adding a Module to Deploy a Custom Master Page

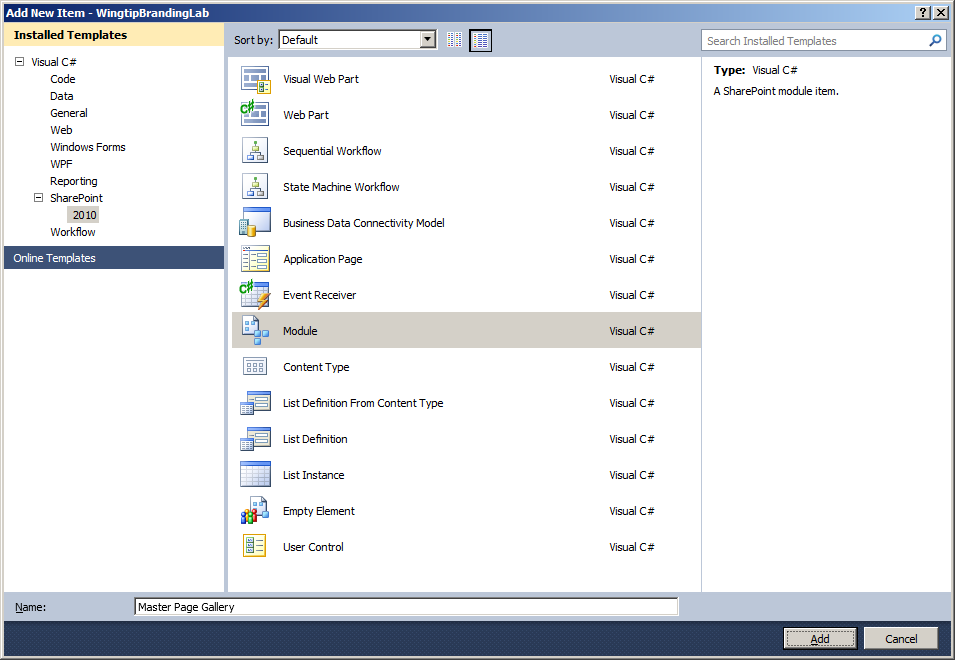
The next step in creating a custom branding solution is to add a custom master page. You will do this by adding a **Module** project item and configuring it to deploy the custom master page into the master page gallery.

The purpose of this lab exercise is for you to learn how to properly deploy a master page. Therefore, you will not be required to actually design or edit a master page. Instead, the lab provides you with a file for a custom master page that has already been completed.

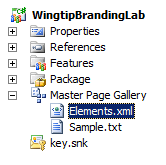
1. Return to Visual Studio 2010 and make sure the project named **WingtipBrandingLab is open.**
2. In **Solution Explorer**, right-click the **WingtipBrandingLab** project node. Select the command **Add** > **New Item**. Selecting this command will display the **Add New Item** dialog.



1. In the **Add New Item** dialog box, select the project item template named **Module**. Give the new Module item a name of **Master Page Gallery** as shown in the following screenshot. Click the **Add** button to add the new Module item to your project.



1. After you create the new Module, it contains an element manifest named **Elements.xml** and a sample element file named **Sample.txt**.



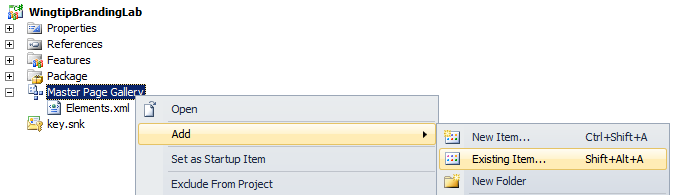
1. Delete the file named **Sample.txt** file by right-clicking it and select the **Delete** command.

Over the next few steps, you will add a custom master page to the Module folder.

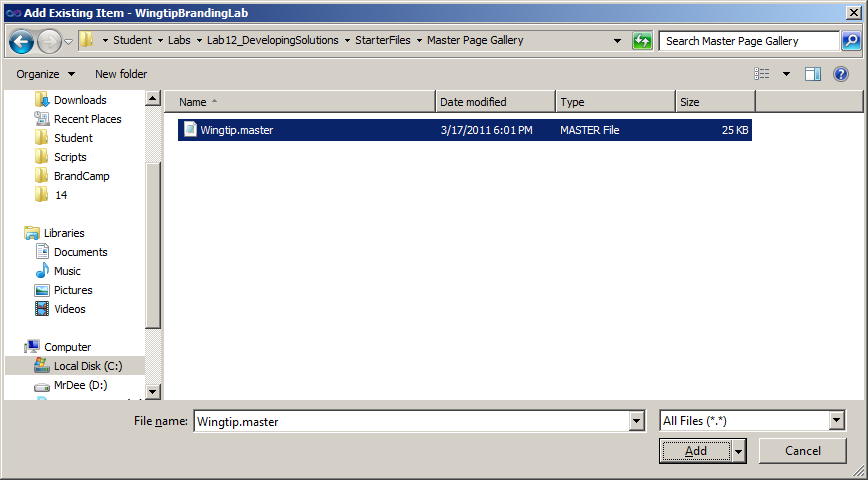
1. Using the Windows Explorer, locate the completed master page file named Wingtip.master. This master page file is located in the following directory inside the Student folder.

C:\Student\Labs\Lab12\_DevelopingSolutions\StarterFiles\MasterPageGallery

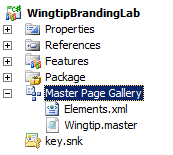
1. Return to Visual Studio 2010. Right-click on the node for the **Master Page Gallery** Module and select the **Add > Existing** command. When you select this command, Visual Studio displays the **Add Existing Item** dialog.



1. Use the **Add Existing Item** dialog to navigate to the custom master page named **wingtip.master**.



1. At this point, you should be able to verify that you can see the master page file named **wingtip.master** inside the **Master Page Gallery** Module.



Now that you have added the custom master page, the next step is to manually edit the **elements.xml** file inside the **Master Page Gallery** module to ensure that this master page is correctly deployed to the master page gallery during Feature activation.

1. Open the **elements.xml file** and inspect the XML that is currently inside. You should be able to see the **Module** element with an inner **File** element. The following code snippet shows what the **elements.xml** file should look like initially. Note this listing contains a few indents and line breaks to make the XML contents more readable.

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

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

<Module Name="Master Page Gallery">

<File Path="Master Page Gallery\Wingtip.master"

Url="Master Page Gallery/Wingtip.master" />

</Module>

</Elements>

1. Modify the Module element by adding two new attributes named **Path** and **Url**. Assign the **Path** attribute a value of **Master Page Gallery**. Assign the **Url** property a value of **\_catlogs/masterpage**.

<Module Name="Master Page Gallery"

Path="Master Page Gallery"

Url="\_catalogs/masterpage" >

<File Path="Master Page Gallery\Wingtip.master"

Url="Master Page Gallery/Wingtip.master" />

</Module>

The **Path** attribute is used specific where the template files can be found within the feature folder. The **Url** attribute is used to specify the site-relative path to where the files will be created. In this case the **Url** attribute is given the relative path from the root of a site to the master page gallery which is **\_catalogs/masterpage**.

1. Now modify the **Path** attribute and the **Url** attribute of the File so they both have a shortened value of **Wingtip.master**. Also add a Type attribute and give it a value of **GhostableInLibrary**.

<Module Name="Master Page Gallery"

Path="Master Page Gallery"

Url="\_catalogs/masterpage" >

<File Path="Wingtip.master" Url="Wingtip.master" Type="GhostableInLibrary" />

</Module>

1. The final step is to add two **Property** elements into the **File** element. Begin by modifying the **File** element so it has an opening tag and a closing tag. Next Add a **Property** element with a Name of **ContentTypeId** and a value of **0x010105**. Add a second **Property** element with a **Name** of **UIVersion** and a value of **4**.

<Module Name="Master Page Gallery"

Path="Master Page Gallery"

Url="\_catalogs/masterpage" >

<File Path="Wingtip.master" Url="Wingtip.master" Type="GhostableInLibrary" >

<Property Name="ContentTypeId" Value="0x010105" />

<Property Name="UIVersion" Value="4" />

</File>

</Module>

The **ContentTypeId** property is required to add your master page into the master page gallery of a publishing site. If you do not add this **Property** element, **wingtip.master** will not be recognized as a master page and will not work correctly in a publishing site. The **UIVersion** property is configure to a value of **4** to indicate this master page has been designed to work with the new SharePoint 2010 user interface and should not be configured for use in sites that are running under the older SharePoint 2007 using the Visual Upgrade feature.

1. As a final check, ensure the elements.xml file looks like the code

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

<Module Name="Master Page Gallery"

Path="Master Page Gallery"

Url="\_catalogs/masterpage" >

<File Path="Wingtip.master" Url="Wingtip.master" Type="GhostableInLibrary" >

<Property Name="ContentTypeId" Value="0x010105" />

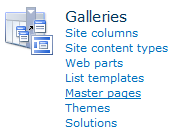
<Property Name="UIVersion" Value="4" />

</File>

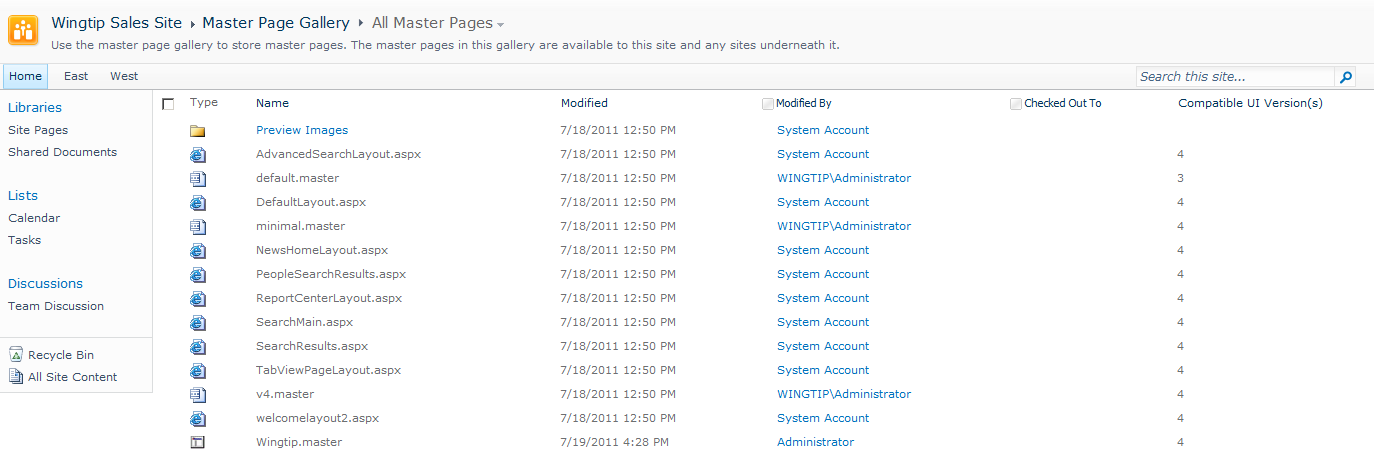
</Module>

</Elements>

1. Now it's time to test your work. Run the **Deploy** command on the project to build and deploy the project **WingtipBrandingLab**. Your project should deploy without any errors.
2. After running the **Deploy** command, switch over to the browser and return to the test site located at **http://intranet.wingtip.com/sites/sales**. Make sure you are at the top-level site. Drop down the **Site Actions** menu and click the **Site Setting** command to navigate to the Site Setting page. Find the group of links with the caption **Galleries**. Within the group, locate and click the link with the caption of **Master page** to navigate to the Master Page Gallery of the site.



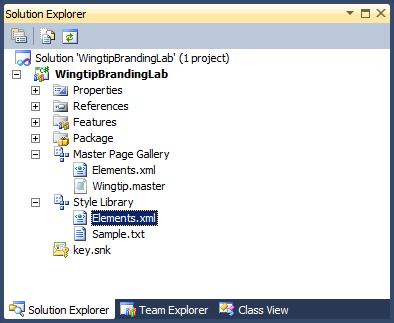
1. Once you have navigated to the Master Page Gallery, you should be able to see the custom master page named **wingtip.master** has been added to the site.



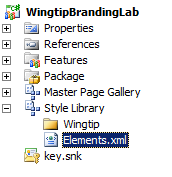
### Exercise 4: Adding a Module to Deploy Files to the Style Library

In this exercise you work through the steps to deploy a custom CSS file named WingtipMaster.css along with a set of images files . You will use the best practice of deploying custom CSS files in the Style Library which works with both sandboxed solutions and farm solutions.

1. Return to Visual Studio 2010 and make sure the project named **WingtipBrandingLab is open.**
2. In **Solution Explorer**, right-click the **WingtipBrandingLab** project node. Select the command **Add** > **New Item**. Selecting this command will display the **Add New Item** dialog.
3. In the **Add New Item** dialog box, select the project item template named **Module**. Give the new Module item a name of **Style Library**. Click the **Add** button to add the new Module item to your project.
4. After you create the new Module, it contains an element manifest named **Elements.xml** and a sample element file named **Sample.txt**.



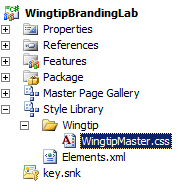
1. Delete the file named **Sample.txt** file by right-clicking it and select the **Delete** command.
2. Create a new folder named **Wingtip** inside the **Style Library** module.



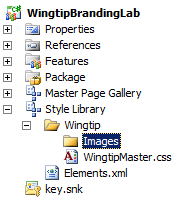
1. Using the Windows Explorer, locate the completed CSS file named file named **WingtipMaster.css**. This CSS file is located in the following directory inside the Student folder.

C:\Student\Labs\Lab12\_DevelopingSolutions\StarterFiles\StyleLibrary\Wingtip

1. Next, add the CSS file named **WingtipMaster.css** into the Wingtip folder of the **Style Library** module.



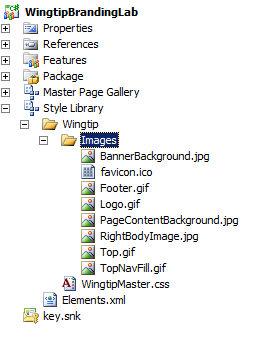
1. Create a new folder named **Images** inside the **Wingtip** folder.



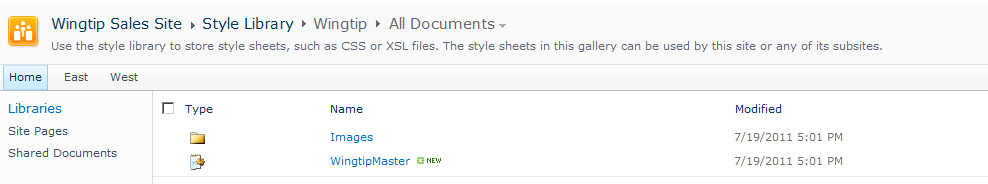
1. Using the Windows Explorer, locate the set of image files in the following folder.

C:\Student\Labs\Lab12\_DevelopingSolutions\StarterFiles\StyleLibrary\Wingtip\Images

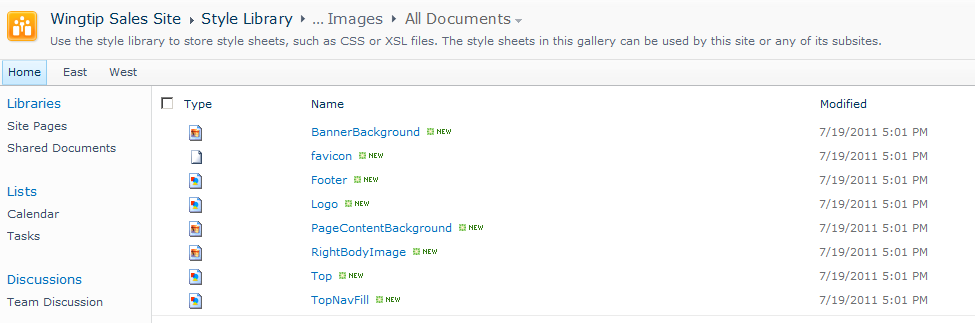
1. Next, add these image files into the **Images** folder of the **Style Library** module.



1. Now it's time to test your work. Run the **Deploy** command on the project to build and deploy the project **WingtipBrandingLab**. Your project should deploy without any errors.
2. After running the **Deploy** command, switch over to the browser and return to the test site located at **http://intranet.wingtip.com/sites/sales**. Make sure you are at the top-level site. Drop down the **Site Actions** menu and click the **View All Site Content** command to navigate to the Site Setting page. Click on the link to navigate to the Style Library.
3. You should be able to see a folder named Wingtip. If you navoiogate into this folder



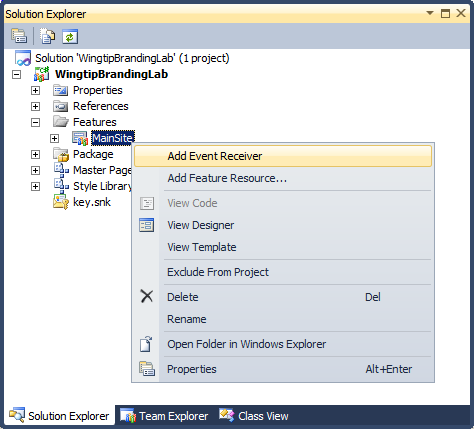
1. If you navigate further into the Images folder, you should be able to see all the image files.



### Exercise 5: Adding a Feature Receiver to Apply Branding Attributes

In this exercise you will modify the MainSite feature by creating a feature receiver and adding the C# code required to configure all the sites within the current site collection to use Wingtip.master as their master page.

1. Return to Visual Studio 2010. Open the project named **WingtipBrandingLab** if it isn't open.
2. In Solution Explorer, right-click the node for the **MainSite** feature and then click the command with the caption of **Add Event Receiver** to add a Feature receiver class.



1. When you select the **Add Event Receiver** command, Visual Studio creates a new C# source file for the event receiver class and adds a large number of comments. Clean up the C# file by removing all the comments to create an empty class definition which looks like the following code listing.

using System;

using System.Runtime.InteropServices;

using System.Security.Permissions;

using Microsoft.SharePoint;

using Microsoft.SharePoint.Security;

namespace WingtipBrandingLab.Features.MainSite {

[Guid("fbbeeddc-0492-4c76-bf3f-e8b815bbca11")]

public class MainSiteEventReceiver : SPFeatureReceiver {

}

}

When you modify the code inside a receiver class, it is important that you never remove the **Guid** attribute or modify the Guid value inside it. That Guid value is used by the SharePoint Tools in Visual Studio 2010 to wire up this event handler. Also note that the Guid value in your code will be different from that the Guid value in the code listing here.

1. Place your cursor inside the class definition and add two string constants named **v4MasterUrl** and **WingtipMasterUrl**. These string constants will be used to track the site-relative paths to standard master page named **v4.master** and the custom master page named **Wingtip.master**. Assign literal values to these string constants using the values shown in the following code listing.

public class MainSiteEventReceiver : SPFeatureReceiver {

const string v4MasterUrl = "\_catalogs/masterpage/v4.master";

const string WingtipMasterUrl = "\_catalogs/masterpage/Wingtip.master";

}

1. Place your cursor below the two string constants and add two methods named **FeatureActivated** and **FeatureDeactivating**. You will need to define these methods with the **override** keyword since you are overriding these methods from the base class **SPFeatureReceiver**.

public class MainSiteEventReceiver : SPFeatureReceiver {

const string v4MasterUrl = "\_catalogs/masterpage/v4.master";

const string WingtipMasterUrl = "\_catalogs/masterpage/Wingtip.master";

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

}

public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {

}

}

1. Place your cursor inside the empty implementation of **FeatureActivated** method. Create a new **SPSite** variable named **siteCollection** to track a reference to the current site collection. Due to the fact that this feature has been scoped at site collection level, you can access the **SPSite**  object for the current site collection using **properties.Feature.Parent**. Note that you will need to perform a type conversion so structure your code to look like this.

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

SPSite siteCollection = properties.Feature.Parent as SPSite;

}

1. Create an **if** statement to ensure that the **siteCollection** variable is not null and that you were able to successfully acquire the **SPSite** reference to the current site collection.

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

SPSite siteCollection = properties.Feature.Parent as SPSite;

if (siteCollection != null) {

}

}

1. Place your cursor inside the body of the if statement. Add a new **SPWeb** variable named **topLevelSite** and initialize it with the **RootWeb** property of the **siteCollection** variable. Next, create a string variable named WebAppRelativePath which will be used to track the relative path from the root of the site collection to the root of the current Web application. Write your code like the code shown in the following listing.

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

SPSite siteCollection = properties.Feature.Parent as SPSite;

if (siteCollection != null) {

SPWeb topLevelSite = siteCollection.RootWeb;

// calculate relative path of site from Web Application root

string WebAppRelativePath = topLevelSite.ServerRelativeUrl;

if (!WebAppRelativePath.EndsWith(@"/")) {

WebAppRelativePath += @"/";

}

}

}

1. Complete the implementation of the **FeatureActivated** method by adding a **foreach** loop which enumerates through each site in the current site collection and updates the **MasterUrl** property and the **CustomMasterUrl** property to use the master page named Wingtip.master that your feature will create in the master page gallery of the top-level site. You can use the following code to accomplish this goal.

public override void FeatureActivated(SPFeatureReceiverProperties properties) {

SPSite siteCollection = properties.Feature.Parent as SPSite;

if (siteCollection != null) {

SPWeb topLevelSite = siteCollection.RootWeb;

// calculate relative path of site from Web Application root

string WebAppRelativePath = topLevelSite.ServerRelativeUrl;

if (!WebAppRelativePath.EndsWith(@"/")) {

WebAppRelativePath += @"/";

}

foreach (SPWeb site in siteCollection.AllWebs) {

site.MasterUrl = WebAppRelativePath + WingtipMasterUrl;

site.CustomMasterUrl = WebAppRelativePath + WingtipMasterUrl;

site.Update();

}

}

}

This example updates the SPWeb property named CustomMasterUrl in addition to the MasterUrl property. Remember that updating the CustomMasterUrl property is important only in publishing sites that contain publishing pages that have been created inside the Pages document library. However, it is safe to do this in any scenario. Assigning a new value to the CustomMasterUrl property in non-publishing site or even a SharePoint Foundation farm has no effect nor will it cause any problems.

1. Now implement the **FeatureDeactivating** method to redirect the pages of all site from Wingtip.master back to v4.master. You can simply copy and paste all the code from FeatureActvited and then replace the constant value of **WingtipMasterUrl** with **v4MasterUrl**.

public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {

SPSite siteCollection = properties.Feature.Parent as SPSite;

if (siteCollection != null) {

SPWeb topLevelSite = siteCollection.RootWeb;

// calculate relative path of site from Web Application root

string WebAppRelativePath = topLevelSite.ServerRelativeUrl;

if (!WebAppRelativePath.EndsWith(@"/")) {

WebAppRelativePath += @"/";

}

foreach (SPWeb site in siteCollection.AllWebs) {

site.MasterUrl = WebAppRelativePath + v4MasterUrl;

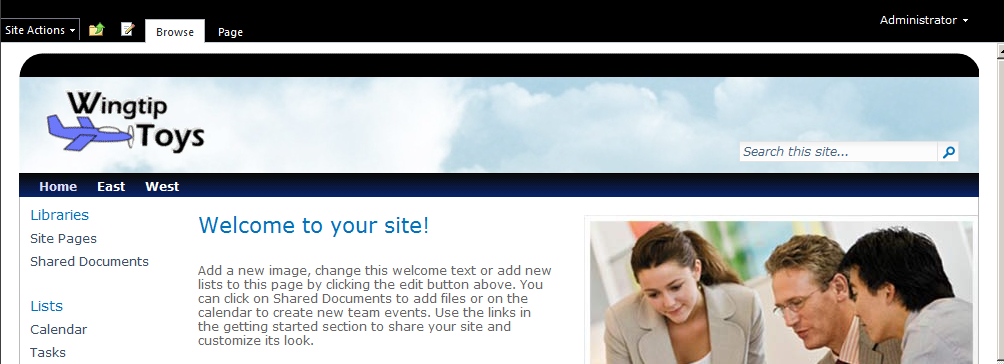
site.CustomMasterUrl = WebAppRelativePath + v4MasterUrl;

}

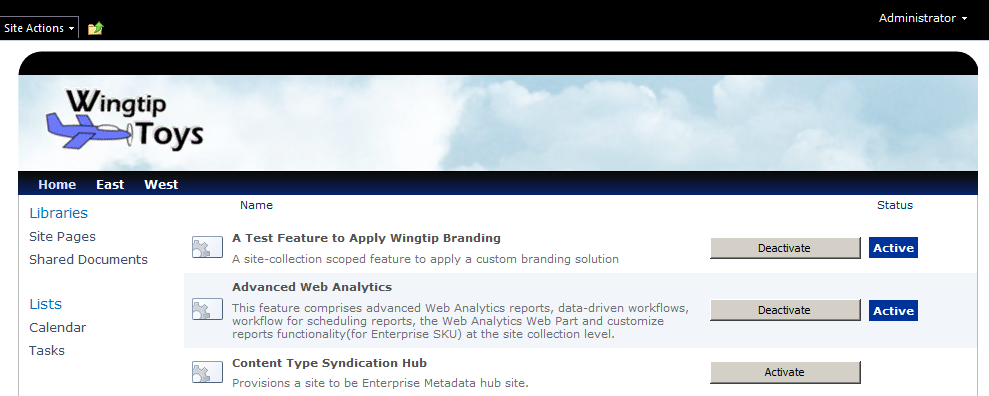
}

}

1. Now it's time to test your work. Run the **Deploy** command on the project to build and deploy the project **WingtipBrandingLab**. Your project should deploy without any errors.
2. After running the **Deploy** command, switch over to the browser and return to the test site located at **http://intranet.wingtip.com/sites/sales**. Make sure you are at the top-level site. Refresh the home page and you should be able to see the site is using the new master page named Wingtip.master instead of the standard master page v4.master.



1. Click on the link of a child site such as East or West. You should be able to verify that they are using the custom master page as well.
2. Drop down the **Site Actions** menu and click the **Site Setting** command to navigate to the Site Setting page. Find the group of links with the caption **Site Collection Administration**. Within the group, locate and click the link with the caption of **Site collection features** to navigate to the **Site Collection Administration page** for Features. You should locate the feature that is applying your branding because it has a Title of **A Test Feature to Apply Wingtip Branding**.

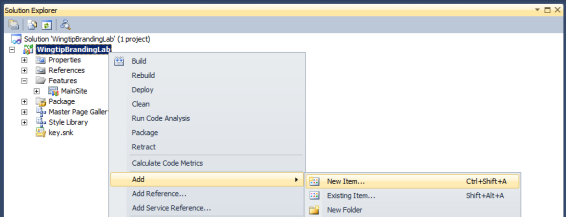


1. See what happens when you deactivate and then reactivate the feature. You should see that the custom master page is removed when you deactivate the feature and that the custom master page is reapplied when you reactivate it. When you are done with this testing, make sure you finish by activating the feature so that site collection is using the custom master page.
2. Return to the home page of the top-level site.
3. Now you will perform a test to see what happens when you create a new child site. Now you will use the browser to create a new child site. Begin by dropping down the **Site Actions** menu and clicking the **New Site** command. When you do this, you will be redirected to the **New SharePoint Site** page.
4. Enter the required information in the **New SharePoint Site** page to give your new site a **Title** of **North** and an **URL** of **North** which is relative to the URL of the parent site. Select **Team Site** as the site template. You can also enter a **Description** for the site if you would like. You can leave all other input controls on the page with their default values. Click the **Create** button to create a new child site.
5. When the new site has been created, you should be able to see that it does not have the custom master page. Instead, it is using the standard master page named v4.master. This presents a problem in that your branding solution does not automatically apply its branding properties to new sites as they are created. This will be address in the final exercise of this lab.

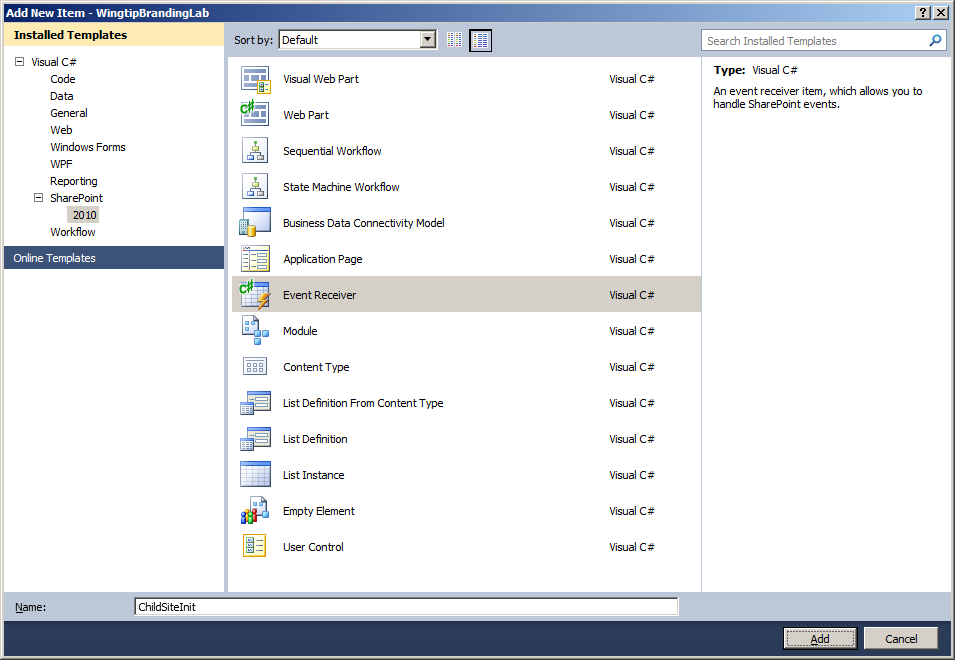
### Exercise 6: Adding an Event Receiver to Brand Child Sites

You must now add support to your branding solution to automatically apply your custom master page to child sites as they are created inside a site collection that has activated your branding feature. You will accomplish this by adding a new event receiver that executes custom code whenever a new child site is created.

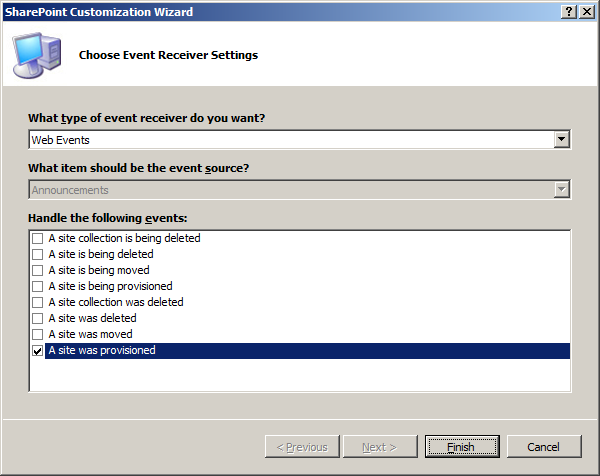
1. Return to Visual Studio 2010. Open the project named **WingtipBrandingLab** if it isn't open.
2. In Solution Explorer, right-click the **WingtipBranding** project node, and on the Project menu, select **Add** and then **New Item.**



1. In the **Add New Item** dialog box, select the **Event Receiver** project item template. In the textbox at the bottom, add a name for the item of **ChildSiteInit**. When you are done, click the **Add** button which will launch the **SharePoint Customization Wizard**.



1. Whenever you create a new **Event Receiver** project item, the **SharePoint Customization Wizard** prompts you to select the type of event receive that you would like to create. Select an Event Receiver of type **Web Events**. At the bottom of the dialog box, there is a list of event types with a caption of **Handle the following events**. Check the very last event with a caption of **A site was provisioned**. Click the **Finish** button to create the new Event Receiver item.



1. When you click **Finish**, the SharePoint Tools in Visual Studio 2010 create a new C# source file and add a new event handler method named **WebProvisioned** along with an **Elements.xml** file, which contains the **Receivers** element that is required to register this event handler.

The purpose of adding this event receiver is that **WebProvisioned** event handler will execute each time a new child site is created. This makes it fairly easy to write the code that copies the relevant SPWeb properties from the top-level site to the new child site.

1. Open the C# source file and locate the event receiver class named **ChildSiteInit**. Remove all the comments so your code look like the following listing.

using System;

using System.Security.Permissions;

using Microsoft.SharePoint;

using Microsoft.SharePoint.Security;

using Microsoft.SharePoint.Utilities;

using Microsoft.SharePoint.Workflow;

namespace WingtipBrandingLab.ChildSiteInit {

public class ChildSiteInit : SPWebEventReceiver {

public override void WebProvisioned(SPWebEventProperties properties) {

}

}

}

1. Place your cursor inside the WebProvisioned method and create two **SPWeb** variables named **childSite** and **topSite**. Initialize the **childSite** variable to reference the new site that has just been created by accessing the **Web** property of the **properties** argument. Initialize the **topSite** variable to with the top-level site in the current site collection using **childSite.Site.RootWeb**.

public override void WebProvisioned(SPWebEventProperties properties) {

// obtain referecnes to new child site and top-level site

SPWeb childSite = properties.Web;

SPWeb topSite = childSite.Site.RootWeb;

}

1. Copy the value from the MasterUrl property and the CustomMasterUrl property from the top-level site to the new child site. Call the Update method on the childSite variable to save your changes to the content database.

public override void WebProvisioned(SPWebEventProperties properties) {

// obtain referecnes to new child site and top-level site

SPWeb childSite = properties.Web;

SPWeb topSite = childSite.Site.RootWeb;

// copy master page settings from top-level site

childSite.MasterUrl = topSite.MasterUrl;

childSite.CustomMasterUrl = topSite.CustomMasterUrl;

childSite.Update();

}

1. Now it's time to test your work. Run the **Deploy** command on the project to build and deploy the project **WingtipBrandingLab**. Your project should deploy without any errors.
2. After running the **Deploy** command, switch over to the browser and return to the test site located at **http://intranet.wingtip.com/sites/sales**. Make sure you are at the top-level site. Refresh the home page and you should be able to see the site is using the new master page named Wingtip.master instead of the standard master page v4.master.
3. Now you will perform a test to see what happens when you create a new child site. Drop down the **Site Actions** menu and click the **New Site** command. When you do this, you will be redirected to the **New SharePoint Site** page.
4. Enter the required information in the **New SharePoint Site** page to give your new site a **Title** of **South** and an **URL** of **South** which is relative to the URL of the parent site. Select **Team Site** as the site template. You can also enter a **Description** for the site if you would like. You can leave all other input controls on the page with their default values. Click the **Create** button to create a new child site.
5. When the new site has been created, you should be able to verify that it has the same custom master page as the other sites. Now you have a branding solution that automatically applies the custom master page to new sites as they are created inside a branded site collection.