* 1. 

Hands-On Lab

Visual Studio 2010 SharePoint Tools

* 1. Lab version: 1.1.0
  2. Last updated: 1/4/2011
  3. 

Contents

[Overview 3](#_Toc281415902)

[Exercise 1: Creating a SharePoint 2010 Project 6](#_Toc281415903)

[Task 1 – Navigating the new Site Collection 6](#_Toc281415904)

[Task 2 – Creating a New SharePoint Project 7](#_Toc281415905)

[Task 3 – Adding a new Feature 11](#_Toc281415906)

[Task 4 – Adding Images to Your Project 14](#_Toc281415907)

[Exercise 2: Feature Event Receiver 18](#_Toc281415908)

[Task 1 – Adding a Feature Event Receiver 19](#_Toc281415909)

[Task 2 – Adding Functionality to the Event Receiver 20](#_Toc281415910)

[Exercise 3: Deploying and Debugging SharePoint Projects 23](#_Toc281415911)

[Task 1 – Configuring and Deploying a SharePoint Project 24](#_Toc281415912)

[Task 2 – Debugging SharePoint Feature Using Visual Studio 2010 30](#_Toc281415913)

[Exercise 4: Creating, Deploying and Testing Web Parts 33](#_Toc281415914)

[Task 1 – Adding a new Web Part 33](#_Toc281415915)

[Exercise 4: Verification 39](#_Toc281415916)

[Exercise 5: Creating a Visual Web Part with AJAX Behavior 42](#_Toc281415917)

[Task 1 – Adding a Visual Web Part 42](#_Toc281415918)

[Task 2 – Adding Markup to the Web Part 46](#_Toc281415919)

[Task 3 – Implementing the Web Part 50](#_Toc281415920)

[Exercise 5: Verification 53](#_Toc281415921)

[Summary 56](#_Toc281415922)

Overview

In this lab, you will start using the Visual Studio 2010 SharePoint Developers Tools. This will give you a chance to become familiar with the standard project structure used by SharePoint Developers Tools. You will go through the steps of creating and testing a project that contains a Feature, a Feature Receiver and a Web Part. Along the way you will learn to configure the SharePoint project’s deployment options as well debug a SharePoint project using Visual Studio 2010 by single-stepping through the code in your solution.

# Objectives

* 1. In this Hands-On Lab, you will learn how to:
  + Create a brand new SharePoint project using the templates included in the SharePoint Developers Tools.
  + Create and code a Feature Event Receiver in order to react to SharePoint feature activation and deactivation.
  + Configure, deploy, test and debug a SharePoint project using Visual Studio 2010 SharePoint developers Tools.
  + Create, modify and test SharePoint Web Parts using Visual Studio 2010.
  + Create, modify and test SharePoint Ajax-enabled Visual Web Parts using Visual Studio 2010.

# Exercises

* 1. This Hands-On Lab is comprised by the following exercises:
  + Exercise 1: Creating a SharePoint 2010 Project
  + Exercise 2: Feature Event Receiver
  + Exercise 3: Deploying and Debugging SharePoint Projects
  + Exercise 4: Creating, Deploying and Testing Web Parts
  + Exercise 5: Creating a Visual Web Part with AJAX Behavior
  1. Estimated time to complete this lab: **45 minutes**.

# System Requirements

* 1. You must have the following items to complete this lab:
  + Microsoft Visual Studio 2010
  + Internet Information Server 7.0
  + Microsoft SharePoint Server 2010
  + Microsoft SharePoint Server 2010 Developer Tools
    1. **Note:** You can follow the [Set up the Development Environment for SharePoint Server](http://msdn.microsoft.com/en-us/library/ee554869%28office.14%29.aspx) article in order to get instruction to install SharePoint Server appropiatelly.

# Setup

* 1. All the requisites for this lab are verified using the **Configuration Wizard**. To make sure that everything is correctly configured, follow these steps.
  2. Run the **Configuration Wizard** for the Training Kit if you have not done it previously. To do this, browse to **Source\Setup** folder of this lab, and double-click the **Dependencies.dep** file. Install any pre-requisites that are missing (rescanning if necessary) and complete the wizard.
     1. **Note:** The Configuration Wizard is used for checking dependencies and setting up the environment. If the Configuration Wizard is not installed on your machine, you must install it running the DependencyChecker.msi file located on the %VS2010TKInstallationFolder%\Assets\DependencyChecker folder (e.g. by default the Training Kit is installed under C:\VS2010TrainingKit).
     2. The Configuration Wizard will also setup a new blank site collection at the location **http://localhost/sites/SharePointToolsLab**. 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. If you want to change the site collection location, modify the **SetupSiteCollection.ps1** in the **Setup\scripts\tasks** folder of this lab.

# Starting Materials

* 1. This Hands-On Lab includes the following starting materials.
  + Each exercise is accompanied by a starting solution you can find in the **Begin** folder of each exercise. Some code sections are missing from these solutions, which will be completed through each exercise. Therefore the starting solutions will not work if you run them directly.
  + Inside each exercise you will also find an **End** folder containing the resulting solution you should obtain after completing the exercises. You can use this solution as a guide if you need additional help working through the exercises.
  + Each exercise contains a **Visual Basic** and a **C#** version; Inside the End/Begin solution folder you will find two folders: VB, containing the Visual Basic version of the exercise, and C#, containing the C# version of it.

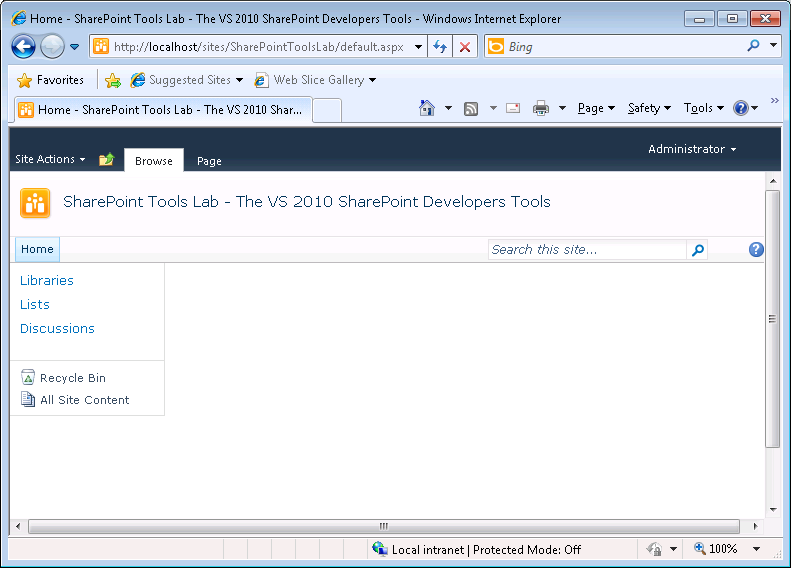
# Next Step

Exercise 1: Using HTML Code Snippets in Visual Studio 2010

Exercise 1: Creating a SharePoint 2010 Project

* 1. In the first exercise, you will create an empty project and focus on the aspects of the Visual Studio 2010 SharePoint Tools that are common across all projects created with this toolset. Most of your work will involve changing the properties of the project and properties of the project's main feature.

Task 1 – Navigating the new Site Collection

* 1. In this task, you will verify that an empty SharePoint Site Collection was created using the Configuration Wizard application.
  2. Launch the Internet Explorer and navigate to the top-level site at **http://localhost/sites/SharePointToolsLab**. You should observe that the newly-created site is a Blank site. This is the site you will use to test and debug the project you will develop throughout this entire lab exercise.
     1. 
     2. Figure 1
     3. SharePoint Blank Site Collection
     4. **Note:** If you haven’t already done so, run the **Configuration Wizard** for the Lab to create the new site collection that will be used to test and debug the code you will be writing in this lab. Remember to change the site URL if you modified it in the setup scripts.

Task 2 – Creating a New SharePoint Project

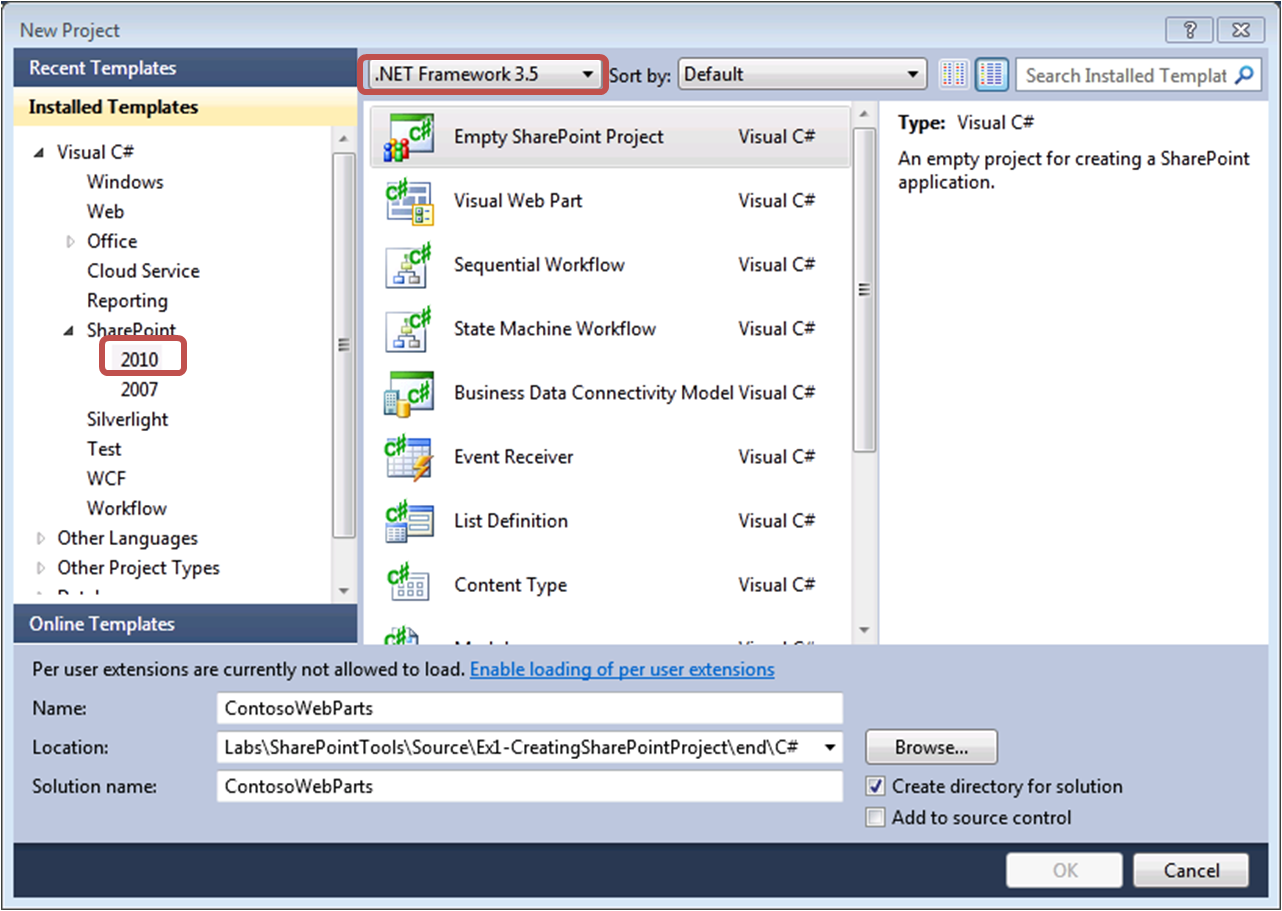
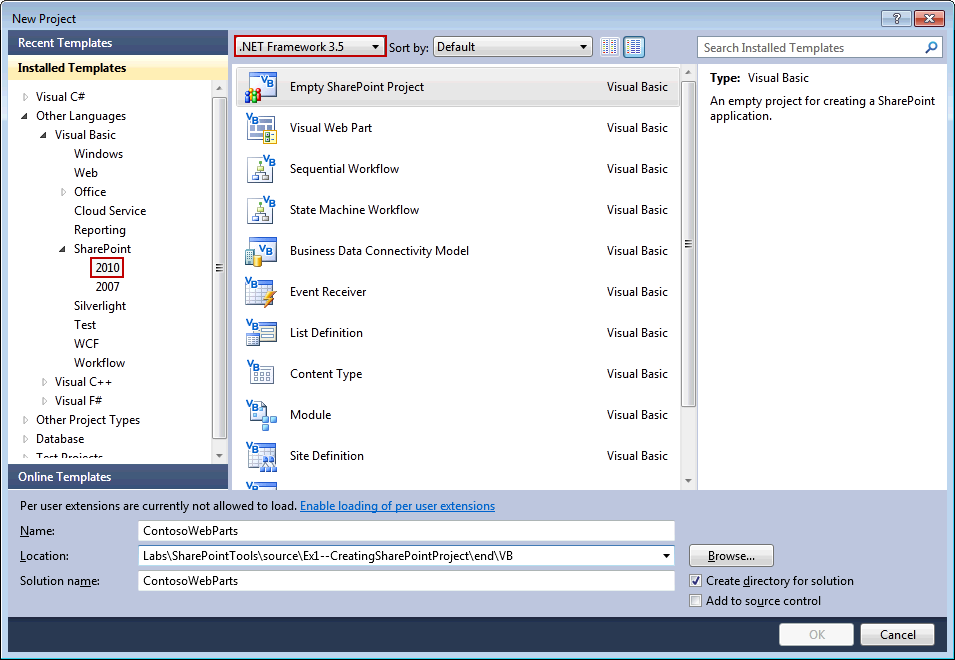
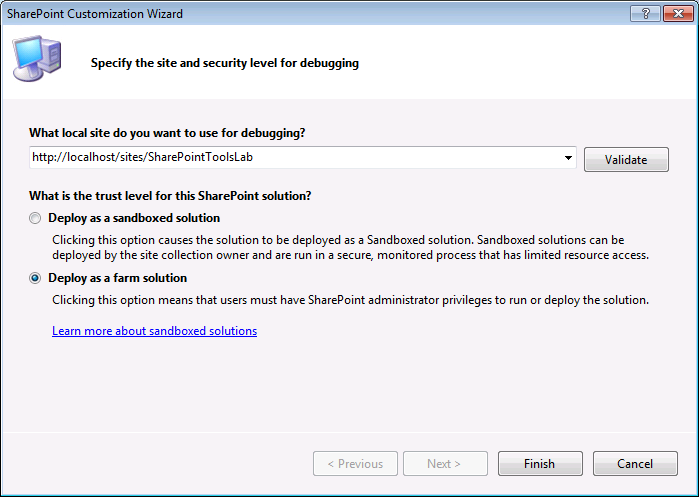
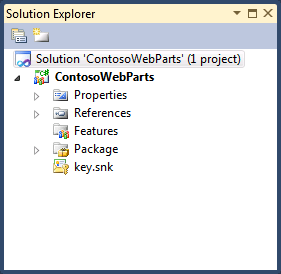
* 1. In this task, you will use the templates included in Visual Studio 2010 to create an Empty SharePoint project.
  2. Open Microsoft Visual Studio 2010 from **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Microsoft Visual Studio 2010**.
  3. On the **File** menu, point to **New**, and click **Project**.
  4. In the **New Project** dialog box make sure that **.NET Framework 3.5** is selected and select either **Visual C#** or **Visual Basic,** then select **SharePoint** – **2010** in the Installed Templates list and the **Empty SharePoint Project** type. You may set the location to the **Source\Ex1-CreatingSharePointProject\begin** (choosing the folder that matches the language of your preference) which is the provided folder for this lab.
     1. **Note:** .NET Framework 4.0 is currently not supported by SharePoint 2010 and the SharePoint Project template will set the target framework attribute to .NET Framework 3.5.
  5. Change the **Name** to **ContosoWebParts** and click **OK**.
     1. 
     2. Figure 2
     3. New Empty SharePoint Project (C#)
     4. 
     5. Figure 3
     6. New Empty SharePoint Project (VB)
  6. When the **SharePoint Customization Wizard** prompts you for a local debugging site, enter **http://localhost/sites/SharePointToolsLab** and change the trust level to **Deploy as a farm solution** and then click **Finish**.
     1. 
     2. Figure 4
     3. SharePoint Customization Wizard
     4. **Note:** Sanboxed solution has certain restrictions, such as being unable to access filesystem, even during deployment. The deployment of ASCX files is not allowed in a sandbox solution, since your project will contain a user control, you need to create a farm solution. For more information on Sandboxed solutions please visit: <http://msdn.microsoft.com/en-us/magazine/ee335711.aspx>
  7. Once the new project has been created, you should be able to observe it has four nodes: **Properties**, **References**, **Features** and **Package**. While the first two nodes are standard on all Visual Studio projects, the Features node and the Package node are unique to projects create with the Visual Studio 2010 SharePoint Tools.
     1. 

Figure 5

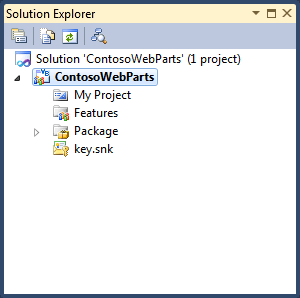
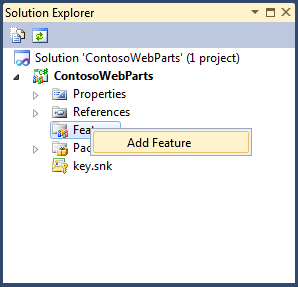
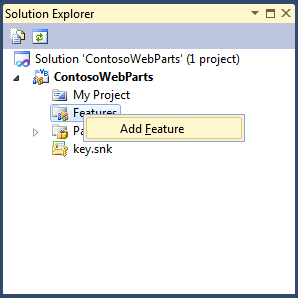
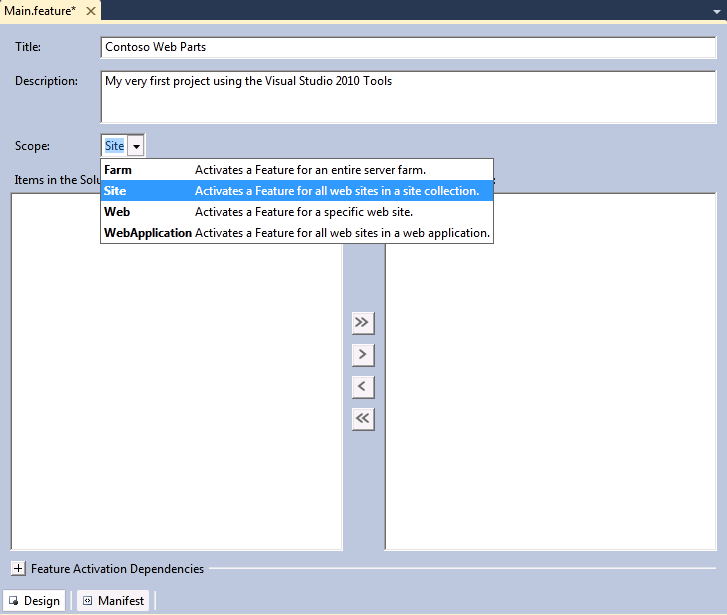
* + 1. SharePoint Project solution structure (C#)
    2. 

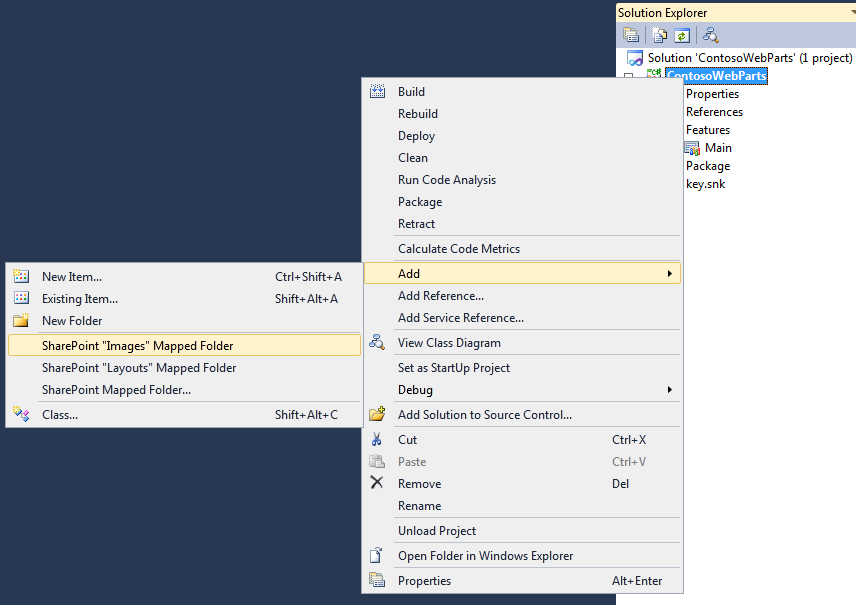
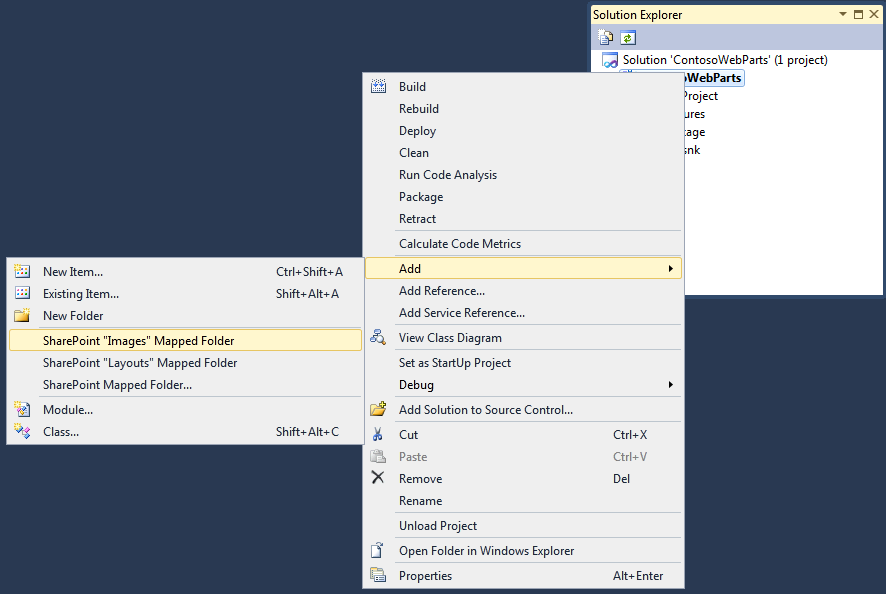
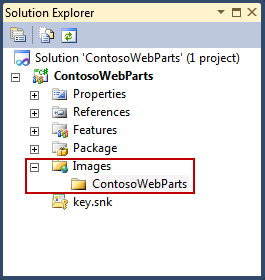
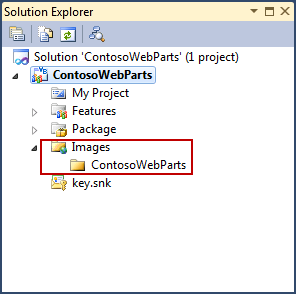
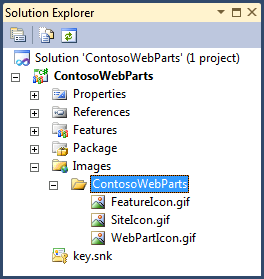
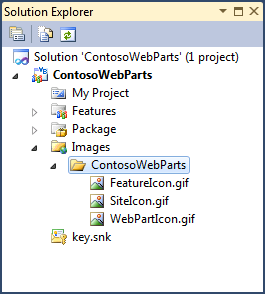
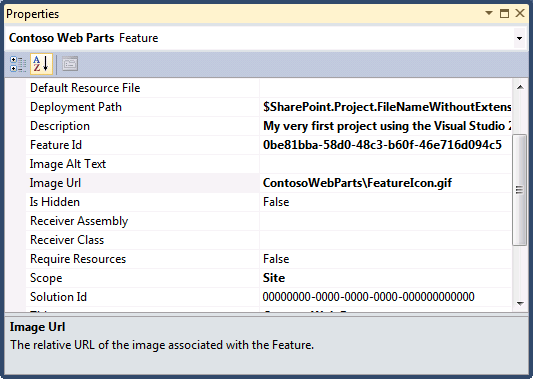
Figure 6

* + 1. SharePoint Project solution structure (VB)

Task 3 – Adding a new Feature

* 1. In this task, you will add a new Feature to your project which is a core element to all SharePoint project.
  2. Add a new feature to the project. To do this, right-click on the **Features** node and select the **Add Feature** option.
     1. 
     2. Figure 7
     3. Add New Feature (C#)
     4. 
     5. Figure 8
     6. Add New Feature (VB)
  3. Rename the recently created feature to Main. To do this, right-click on the node for the default feature added named **Feature1** and click **Rename**. Rename this feature to **Main**.
     1. **Note:** If you are following the lab using the C# version, there is currently a bug that won’t allow you to rename the feature if you have a “#“ symbol in the path. (for example **End\C#\)**.
     2. As a workaround, copy your solution to another folder and you will then be able to rename the feature.
  4. Open the **Main** feature in the feature designer. Use the feature designer to change the feature’s **Title**, **Description** and **Scope** using the following values:
     + Title: **Contoso Web Parts**
     + Description: **My very first project using the Visual Studio 2010 Tools**
     + Scope: **Site**
     1. 
     2. Figure 9
     3. Feature Designer
     4. **Note:** A Scope setting of Web results in site-level activation of the feature while a Scope setting of Site results in activation at the site collection level. The change of the feature's Scope setting to Site is required due to the fact that Web Parts deployment requires a feature which activates at site collection scope in order to properly deploy Web Part description files.

Task 4 – Adding Images to Your Project

* 1. In this task, you will add a few images into your project so they are deployed inside the SharePoint Images folder. To do this, proceed as follows:
  2. In the Solution Explorer, right-click the **ContosoWebParts** project, point to **Add** and choose the **SharePoint "Images" Mapped Folder** command to add an Images Mapped Folder to your project.
     1. 
     2. Figure 10
     3. Add SharePoint “Images” Mapped Folder (C#)
     4. 
     5. Figure 11
     6. Add SharePoint “Images” Mapped Folder (VB)
  3. You will now see that there is an **Images** directory inside your project. Also, you will find a child folder inside it, with the same name as the project **ContosoWebParts**.
     1. 
     2. Figure 12
     3. SharePoint Images Folder (C#)
     4. 
     5. Figure 13
     6. SharePoint Images Folder (VB)
     7. **Note:** SharePoint Tools helps to ensure best practices with SharePoint development. When adding image files to your project, you should not add them directly inside the Images folder. Instead, add them to the inner directory in this case named **ContosoWebParts** to avoid file name conflicts with the image deployed by SharePoint in the Images directory.
  4. Next, you will add images to the site’s images folder. To do this, right-click the **ContosoWebParts** folder under the **Images** folder, point to **Add** and click **Existing Item**. In the Add Existing Item dialog, browse to **Source\Assets** folder and select all three **GIF** files in the folder.
     + FeatureIcon.gif
     + SiteIcon.gif
     + WebPartIcon.gif.
     1. 
     2. Figure 14
     3. Images added to the SharePoint “Images” Folder (C#)
     4. 
     5. Figure 15
     6. Images added to the SharePoint “Images” Folder (VB)
  5. The first GIF file that you will put to use is the **FeatureIcon.gif** by modifying the Image Url property of the **Main** feature. To do this:
     1. Open the **Main** feature in the feature designer and make sure it is the active window.
     2. Press **F4** to openthe Properties Window.
     3. Update the **Image Url** property with an URL pointing to the **FeatureIcon.gif** image.
        1. 
        2. Figure 16
        3. Feature Image Url
        4. **Note:** The Image URL is relative to the **Images** mapped folder.

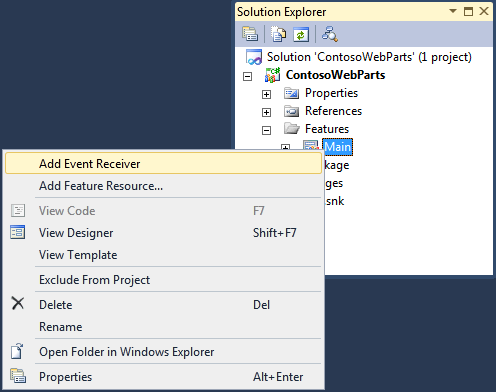
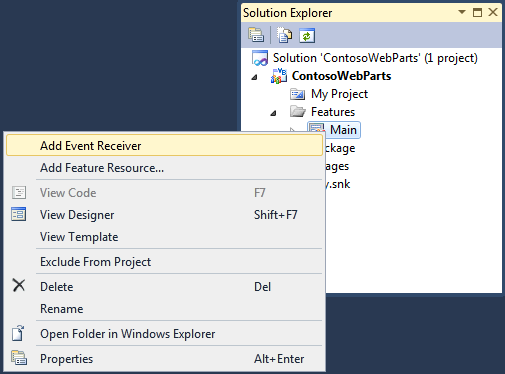
# Next Step

* 1. Exercise 2: Feature Event Receiver

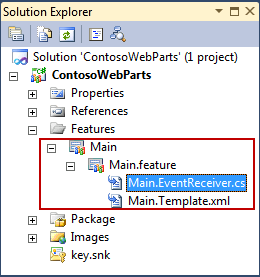
Exercise 2: Feature Event Receiver

* 1. In the previous exercise, you learned how to modify the project properties and features properties using visual designers and property sheets.
  2. In this exercise, you will add some code. Over the next few steps, you are going to add a feature receiver so you can write code against the SharePoint object model that will be automatically executed during feature activation and feature deactivation.

Task 1 – Adding a Feature Event Receiver

* 1. In this task, you will add an event receiver class to your feature in order to react to the event generated with configuring your SharePoint site collection features.
  2. Open Microsoft Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
  3. Open the solution file **ContosoWebParts.sln** located under the **Source\Ex2-EventReceiver\begin\** (choosing the folder that matches the language of your preference.) Alternatively, you may continue working with the solution obtained after completing the previous exercise.
  4. In the Solution Explorer, right-click on the Main feature node, and choose **Add Event Receiver**, toadd an event receiver to the selected feature.
     1. 
     2. Figure 17
     3. Add Event Receiver (C#)
     4. 
     5. Figure 18
     6. Add Event Receiver (VB)

Task 2 – Adding Functionality to the Event Receiver

* 1. In this task, you will implement the **FeatureActivated** and **FeatureDeactivating** methods to change the Title and Image URL of the top-level site.
  2. Implement the **FeatureActivated** method, to do this, proceed as follows:
     1. **Note:** Before changing the **Title** property, your code should track its original value so that it can be restored upon feature deactivation.
     2. In the Solution Explorer, double-click on the **Main.EventReceiver.cs** (C#) class file under the **Main** Feature node. In VB, click on the **Main** feature and press F7 to open the **Code View**.
        1. 
        2. Figure 19
        3. Main.EventReceiver.cs file (C#)
     3. Locate the definition of the **FeatureActivated** method.
        1. **Note:** The definition is the first comment inside the **MainEventReceiver** class.
     4. Replace the commented definition with the following code:
        1. *(Code Snippet – SharePoint Tools Lab – FeatureActivated Properties CSharp)*
        2. C#
        3. **public override void FeatureActivated(SPFeatureReceiverProperties properties)**
        4. **{**
        5. **SPSite siteCollection = properties.Feature.Parent as SPSite;**
        6. **if (siteCollection != null)**
        7. **{**
        8. **// save top site's original Title and SiteLogoUrl**
        9. **SPWeb site = siteCollection.RootWeb;**
        10. **site.Properties["OriginalTitle"] = site.Title;**
        11. **site.Properties.Update();**
        12. **// update the Title and SiteIconUrl**
        13. **site.Title = "VS 2010 SPT Rocks";**
        14. **site.SiteLogoUrl = "\_layouts/images/ContosoWebParts/SiteIcon.gif";**
        15. **site.Update();**
        16. **}**
        17. **}**
        18. *(Code Snippet – SharePoint Tools Lab – FeatureActivated Properties VB)*
        19. Visual Basic
        20. **Public Overrides Sub FeatureActivated(ByVal properties As SPFeatureReceiverProperties)**
        21. **Dim siteCollection = TryCast(properties.Feature.Parent, SPSite)**
        22. **If siteCollection IsNot Nothing Then**
        23. **' save top site's original Title and SiteLogoUrl**
        24. **Dim site = siteCollection.RootWeb**
        25. **site.Properties("OriginalTitle") = site.Title**
        26. **site.Properties.Update()**
        27. **' update the Title and SiteIconUrl**
        28. **site.Title = "VS 2010 SPT Rocks"**
        29. **site.SiteLogoUrl = "\_layouts/images/ContosoWebParts/SiteIcon.gif"**
        30. **site.Update()**
        31. **End If**
        32. **End Sub**
     5. **Note:** The code above first obtains **SPWeb** reference to the top-level from the **SPSite** to get the **Title’s** property current value. It stores the value as a name/value pair inside the SPWeb.Properties collection and then modifies it to the new value “**VS 2010 SPT Rocks**”.
     6. Finally, it changes the **SiteLogoUrl** property to point to **SiteIcon.gif** inside the Images folder.
     7. **Note:** You should also be able to see that there are several method stubs inside the class definition that are commented out. There is also a GUID attribute that has been applied to the receiver class to give a unique identifier. Do not remove the GUID from the class because it will be used behind the scenes by the SPT during the packaging process.
  3. Implement **FeatureDeactivating** so that your code restores the original **Title** property value and changes the **SiteLogoUrl** property value back to an empty string.
     1. *(Code Snippet – SharePoint Tools Lab – FeatureDeactivating Properties CSharp)*
     2. C#
     3. **public override void FeatureDeactivating(SPFeatureReceiverProperties properties)**
     4. **{**
     5. **SPSite siteCollection = properties.Feature.Parent as SPSite;**
     6. **if (siteCollection != null)**
     7. **{**
     8. **// restore top site's original Title and SiteLogoUrl**
     9. **SPWeb site = siteCollection.RootWeb;**
     10. **site.Title = site.Properties["OriginalTitle"];**
     11. **site.SiteLogoUrl = string.Empty;**
     12. **site.Update();**
     13. **}**
     14. **}**
     15. *(Code Snippet – SharePoint Tools Lab – FeatureDeactivating Properties VB)*
     16. Visual Basic
     17. **Public Overrides Sub FeatureDeactivating(ByVal properties As SPFeatureReceiverProperties)**
     18. **Dim siteCollection = TryCast(properties.Feature.Parent, SPSite)**
     19. **If siteCollection IsNot Nothing Then**
     20. **' restore top site's original Title and SiteLogoUrl**
     21. **Dim site = siteCollection.RootWeb**
     22. **site.Title = site.Properties("OriginalTitle")**
     23. **site.SiteLogoUrl = String.Empty**
     24. **site.Update()**
     25. **End If**
     26. **End Sub**

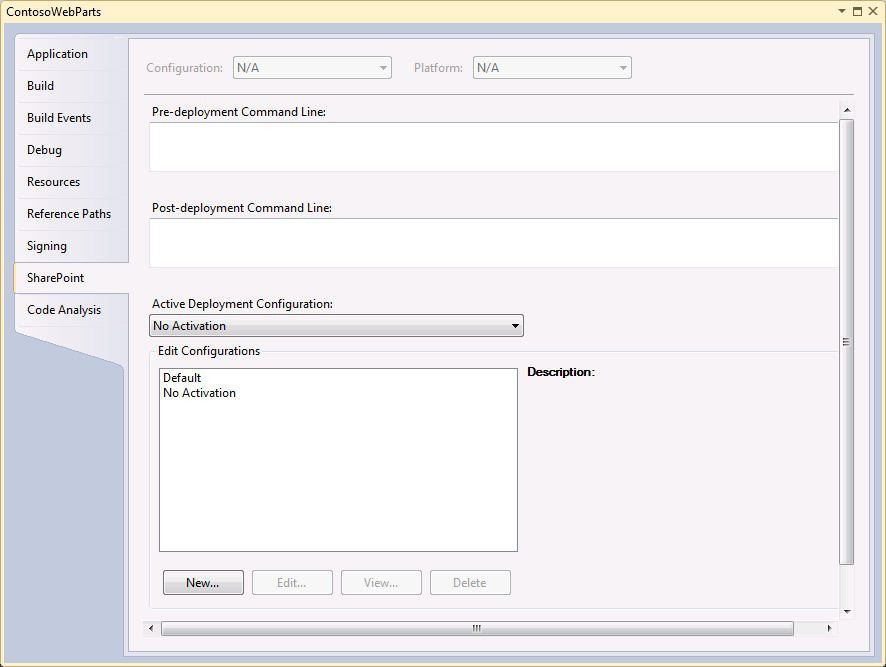
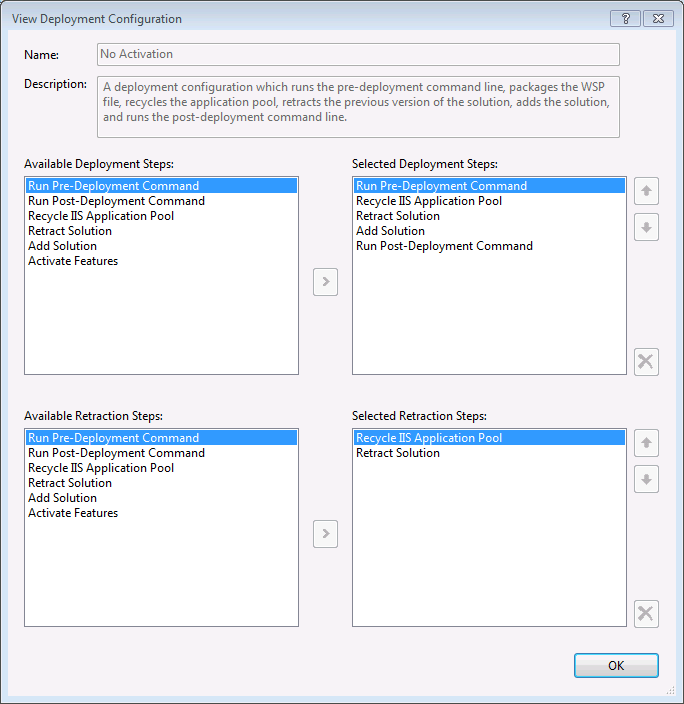
# Next Step

* 1. Exercise 3: Deploying and Debugging SharePoint Projects

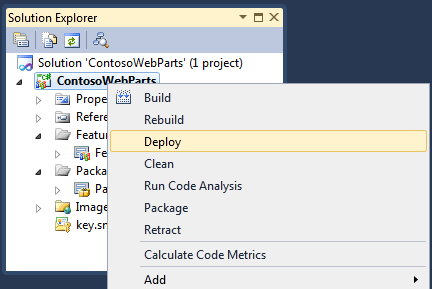
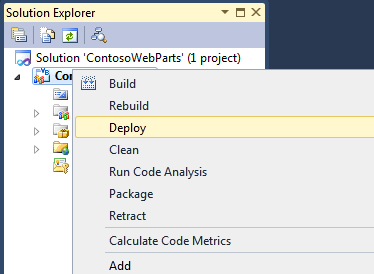
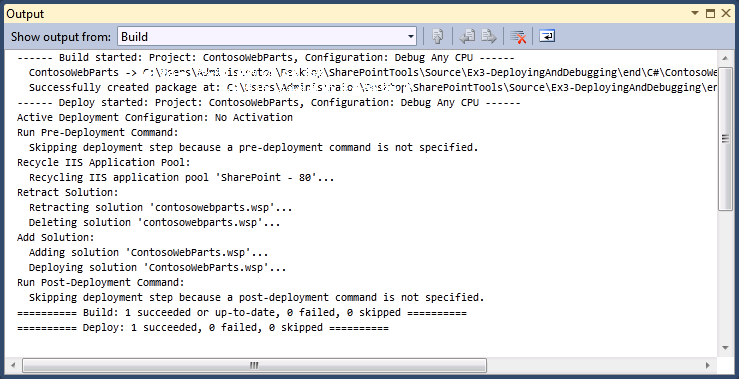
Exercise 3: Deploying and Debugging SharePoint Projects

* 1. In this exercise, you will test the work done so far and see how your project behaves when tested inside a SharePoint site. Also, you will debug your code using Visual Studio 2010 debugger.

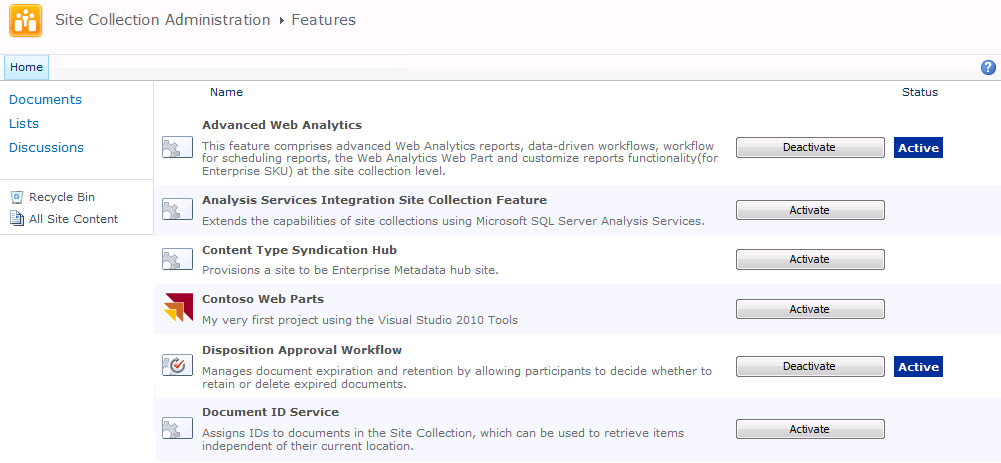
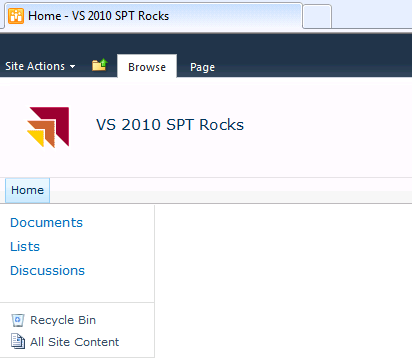
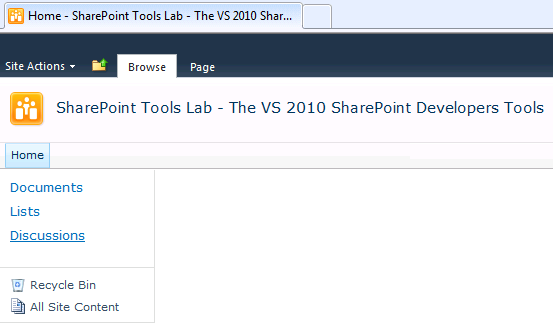
Task 1 – Configuring and Deploying a SharePoint Project

* 1. In this task, you will configure the project properties previous to deploy the SharePoint project to the SharePoint Server.
  2. Open Microsoft Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
  3. Open the solution file **ContosoWebParts.sln** located under the **Source\Ex3-DeployingAndDebugging\begin\** (choosing the folder that matches the language of your preference.) Alternatively, you may continue working with the solution obtained after completing the previous exercise.
  4. Right-click on the ContosoWebParts project node in the Solution Explorer and click Properties.
  5. Navigate to the **SharePoint** tab to see the project deployment options.
  6. Note that there are two textboxes that allow you to add command-line instructions which will execute either just before and directly after the SPT deploy steps are processed. You are not going to add anything to either of the two top textboxes.
  7. Find the combo box control with the caption Active Deployment Configuration and change the selected item from Default to **No Activation**.
     1. 
     2. Figure 20
     3. SharePoint Project Properties
  8. Inside the **Edit Configurations** listbox, select the **No Activation** configuration and click the **View…** button to see its deployment configuration steps.
  9. Click **OK** to close the View Deployment Configuration dialog. Save all your work and close the page which shows the project properties.
     1. 
     2. Figure 21

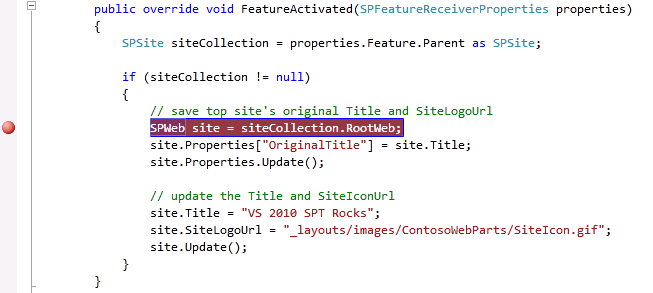
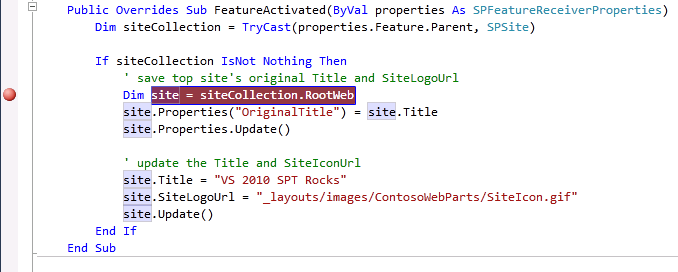
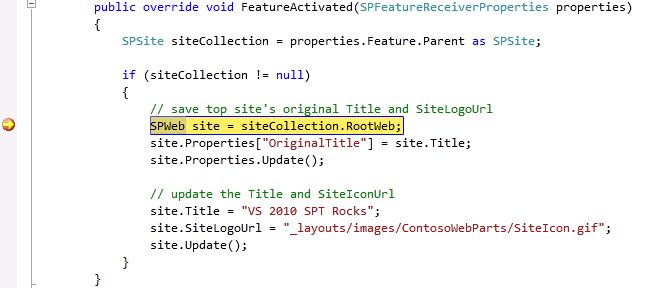
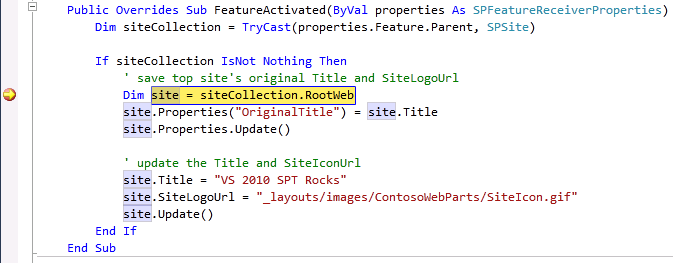
No Activation Deployment Configuration

* 1. Now it's time to run the Deploy command. First, make sure the Output Window is visible so you can see how the things progress during the execution of the Deploy command. Then right-click on the ContosoWebParts project in Solution Explorer and run the project **Deploy** command.
     1. 
     2. Figure 22
     3. Deploy SharePoint Project (C#)
     4. 
     5. Figure 23
     6. Deploy SharePoint Project (C#)
  2. Examine the Output Window after the Deploy command has completed. Verify the steps that provided output to the Output windows.
     1. 
     2. Figure 24

Successful Deploy Output window

* 1. At this point, the solution package for the **ContosoWebParts** project has been deployed on the local SharePoint. Let's test out your work by trying to activate the feature you defined inside the ContosoWebParts project.
     1. Open Internet Explorer, and browse to the test site at **http://localhost/sites/SharePointToolsLab**.
     2. Point to **Site Actions** and click **Site Settings** to navigate to the Site Settings page.
     3. Inside the **Site Collection Administration** section of the **Site Settings** page, click on the **Site collection features** link to navigate to the Site Collection Administration > Features page.
     4. Locate the feature you have been working on with a title of **Contoso Web Parts**. You should also be able to see the feature's custom feature icon.
        1. 
        2. Figure 25
        3. SharePoint Site Collection Administration
  2. Activate the **Contoso Web Parts** feature. If you return to the site's home page you should be able to verify that the code inside the **FeatureActivated** event handler executed and changed the site's title and its site icon.
     1. 
     2. Figure 26
     3. SharePoint Feature Activated
  3. Navigate back to the **Site Collection Administration** > **Features** page, and deactivate the **ContosoWebParts** feature so that you can test the code in the **FeatureDeactivating** event handler. Return to the site's home page and verify that the site title has been restored to its original value and that the site icon has been changed back to the default site icon for SharePoint sites.
     1. 
     2. Figure 27
     3. SharePoint Feature Deactivated

Task 2 – Debugging SharePoint Feature Using Visual Studio 2010

* 1. In this task, you will use Visual Studio 2010 to enter in debug mode so that you can single step through the code while it's executing inside the context of the test site.
  2. Open the source **Main Event Receiver** in **Code View** by pressing **F7** with the **Main** Feature selected, and add a breakpoint to **FeatureActivated**.
     1. 
     2. Figure 28
     3. FeatureActivated Breakpoint (C#)
     4. 
     5. Figure 29
     6. FeatureActivated Breakpoint (VB)
  3. Start the project in debug mode by running **Debug \ Start Debugging** menu command or pressing the equivalent shortcut key which is **F5**.
  4. After running the Start Debugging command wait until the Visual Studio debugger has launched the Internet Explorer and navigated to the test site.
  5. Within the test site, navigate to the **Site Collection Administration** > **Features** page and activate the **ContosoWebParts** feature.
     1. **Note:** At this point, Visual Studio should come into focus, and break at the line where you set the breakpoint. If this does not occur, try deactivating and reactivating the feature again.
     2. 
     3. Figure 30
     4. Debugging FeatureActivated (C#)
     5. 
     6. Figure 31
     7. Debugging FeatureActivated (VB)
  6. Press the **F11** key to single-step through the remaining lines of code. You should be able to hit **F11** repeatedly until the Visual Studio debugger relinquishes control back to the Internet Explorer.
  7. Return to Visual Studio and stop the debugger by running the **Debug \ Stop Debugging** menu command or by pressing the equivalent shortcut key which is **SHIFT+F5**.

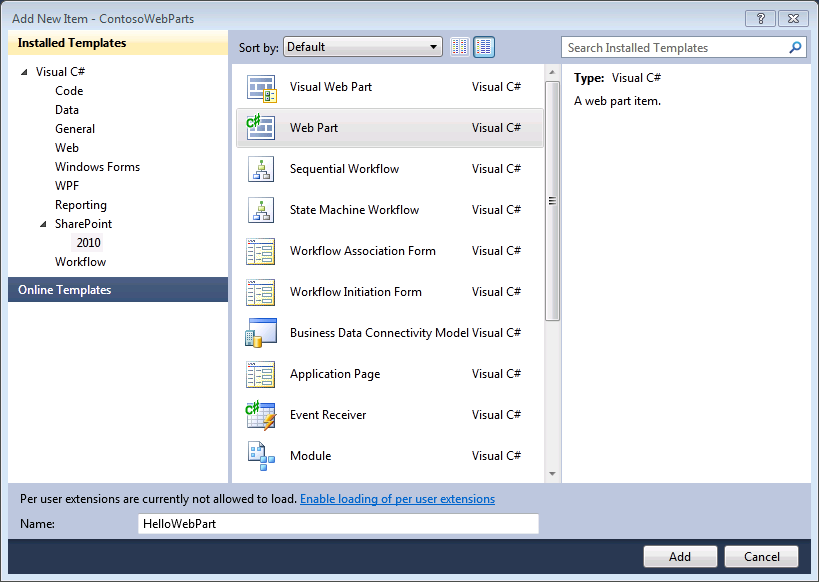
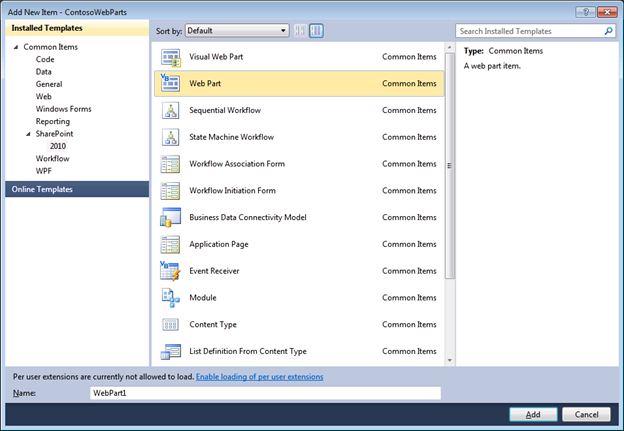
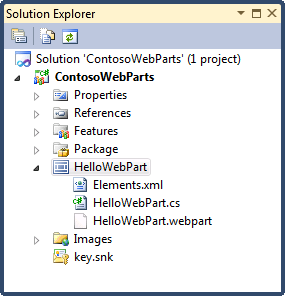
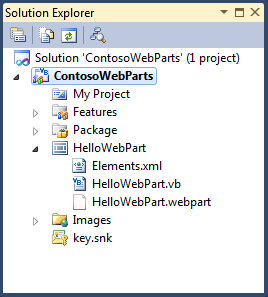
# Next Step

* 1. Exercise 4: Creating, Deploying and Testing Web Parts

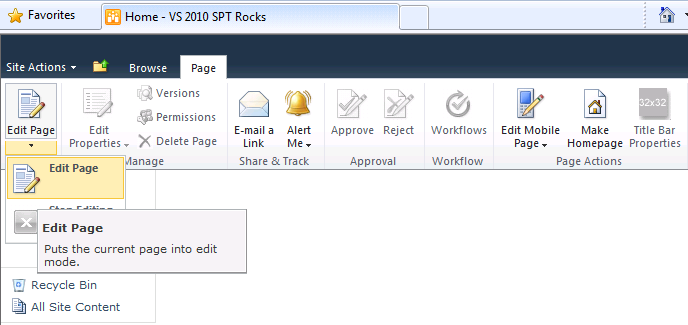
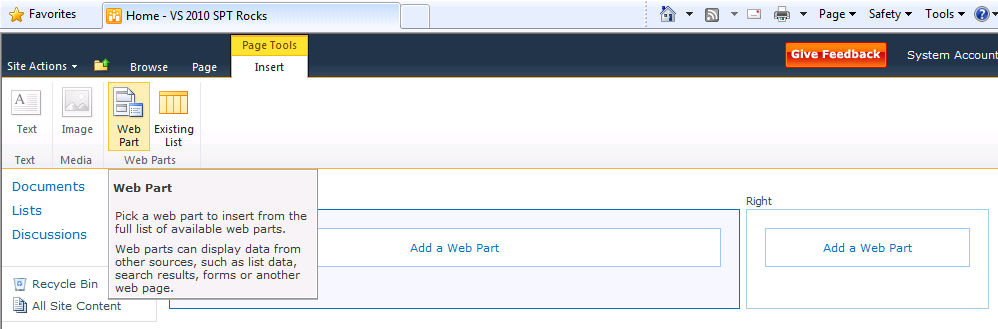
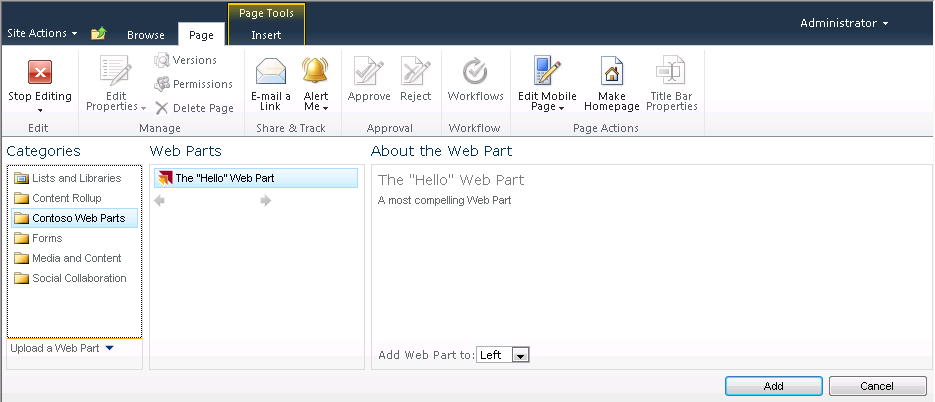
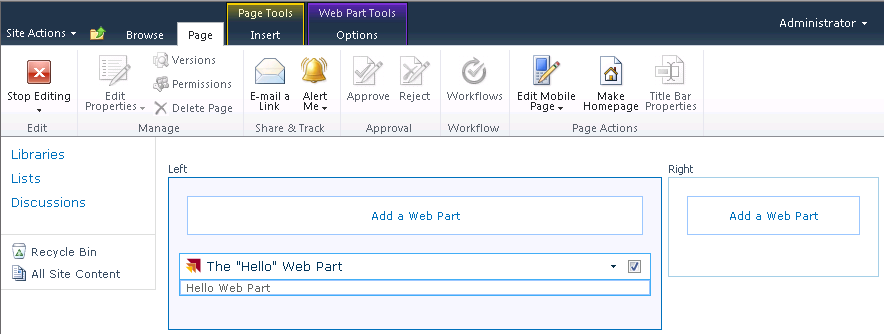
Exercise 4: Creating, Deploying and Testing Web Parts

* 1. In this exercise, you will add a custom Web Part to your project.

Task 1 – Adding a new Web Part

* 1. In this task, you will add a Web Part to your project.
  2. Open Microsoft Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
  3. Open the solution file **ContosoWebParts.sln** located under the **Source\Ex4-WebParts\begin\** (choosing the folder that matches the language of your preference.) Alternatively, you may continue working with the solution obtained after completing the previous exercise.
  4. In the Solution Explorer, right-click on the **ContosoWebParts** project, point to **Add** and select **New Item**.
  5. Select the **Web Part** project item template and give it a name of **HelloWebPart**.
     1. 
     2. Figure 32
     3. New Web Part item (C#)
     4. 
     5. Figure 33
     6. New Web Part item (VB)
  6. Inspect the SharePoint Project Item node for the Web Part named **HelloWebPart**. You should be able to verify that it contains three SharePoint Project Item files named **Elements.xml**, **HelloWebPart.cs** / **HelloWebPart.vb** and **HelloWebPart.webpart**.
     1. 
     2. Figure 34
     3. Inspecting HelloWebPart element (C#)
     4. 
     5. Figure 35
     6. Inspecting HelloWebPart element (VB)
     7. **Note:**
     8. - **Elements.xml** contains the definition of the elements that form the web part.
     9. - **HelloWebPart.webpart** is an xml file containing metadata that describes the web part.
     10. - **HelloWebPart.cs** or **HelloWebPart.vb** contains the code required to implement the web part’s functionality.
  7. In the Solution Explorer, double-click on the **HelloWebPart.webpart** file to open it, and perform the following changes:
     1. Find the XML element for the **Title** property and change its value to: **The "Hello" Web Part**.
     2. Find the XML element for the **Description** property and change its value to: **A most compelling Web Part**.
     3. Add the following properties after the recently modified Description property.
        1. *(Code Snippet – SharePoint Tools Lab – HelloWebPart Properties)*
        2. XML
        3. **<property name="ChromeType" type="chrometype">TitleAndBorder</property>**
        4. **<property name="CatalogIconImageUrl" type="string">\_layouts/images/ContosoWebParts/WebPartIcon.gif</property>**
        5. **<property name="TitleIconImageUrl" type="string">\_layouts/images/ContosoWebParts/WebPartIcon.gif</property>**
  8. In the Solution Explorer, double-click on the **Elements.xml** file to open it.
  9. Modify the **URL** attribute of the **File** XML element, to ensure that the web part name is unique. To do this, add the text value of "**ContosoWebPart\_**" to the beginning of the URL property value.
     1. XML
     2. <?xml version="1.0" encoding="utf-8"?>
     3. <Elements xmlns="http://schemas.microsoft.com/sharepoint/" >
     4. <Module Name="HelloWebPart" List="113" Url="\_catalogs/wp">
     5. <File Path="HelloWebPart\HelloWebPart.webpart" **Url="ContosoWebParts\_HelloWebPart.webpart"** Type="GhostableInLibrary">
     6. <Property Name="Group" Value="Custom" />
     7. </File>
  10. Change the value for the **Group** property to "**Contoso Web Parts**".
      1. XML
      2. <?xml version="1.0" encoding="utf-8"?>
      3. <Elements xmlns="http://schemas.microsoft.com/sharepoint/" >
      4. <Module Name="HelloWebPart" List="113" Url="\_catalogs/wp">
      5. <File Path="HelloWebPart\HelloWebPart.webpart" Url="ContosoWebParts\_HelloWebPart.webpart" Type="GhostableInLibrary">
      6. **<Property Name="Group" Value="Contoso Web Parts" />**
      7. </File>
      8. Finally, open the source file named **HelloWebPart.cs** (C#) or **HelloWebPart.vb** (VB), and replace the class definition with the following code.
      9. *(Code Snippet – SharePoint Tools Lab – HelloWebPart Class CSharp)*
      10. C#
      11. **[ToolboxItemAttribute(false)]**
      12. **public class HelloWebPart : WebPart**
      13. **{**
      14. **public HelloWebPart()**
      15. **{**
      16. **}**
      17. **protected Label outputLabel;**
      18. **protected override void CreateChildControls()**
      19. **{**
      20. **outputLabel = new Label();**
      21. **outputLabel.Text = "Hello Web Part";**
      22. **Controls.Add(outputLabel);**
      23. **}**
      24. **protected override void RenderContents(HtmlTextWriter writer)**
      25. **{**
      26. **base.RenderContents(writer);**
      27. **}**
      28. **}**
      29. *(Code Snippet – SharePoint Tools Lab – HelloWebPart Class VB)*
      30. Visual Basic
      31. **<ToolboxItemAttribute(False)> \_**
      32. **Public Class HelloWebPart**
      33. **Inherits WebPart**
      34. **Public Sub New()**
      35. **End Sub**
      36. **Protected outputLabel As Label**
      37. **Protected Overrides Sub CreateChildControls()**
      38. **outputLabel = New Label()**
      39. **outputLabel.Text = "Hello Web Part"**
      40. **Controls.Add(outputLabel)**
      41. **End Sub**
      42. **Protected Overrides Sub RenderContents(ByVal writer As System.Web.UI.HtmlTextWriter)**
      43. **MyBase.RenderContents(writer)**
      44. **End Sub**
      45. **End Class**
  11. Press **CTRL+SHIFT+B** to build the solution.

Exercise 4: Verification

* 1. Run the **Deploy** command which will retract the previously deployed solution and will deploy the updated solution package.
  2. Open Internet Explorer, and browse to **http://localhost/sites/SharePointToolsLab**. Follow these steps to go through the activation process.
     1. Point to **Site Actions** and select **Site Settings** to navigate to the Site Settings page.
     2. Inside the **Site Collection Administration** section of the Site Settings page, click on the **Site collection features** link to navigate to the Site Collection Administration > Features page.
     3. Locate the **Contoso Web Parts** and **activate** it.
        1. **Note:** If it was already active, then deactivate first and then activate it. The key point here is that feature activation is what provisions the .webpart file into the Web Part Gallery which will allow you to test your work.
  3. Add a test instance of the Web Part you have just created to a Web Part Page. To do this:
     1. Navigate to the home page **http://localhost/sites/SharePointToolsLab**. The ribbon at the top of the page should have a tab titled **Page**.
     2. Click on the **Page** tab and then select the command from the **Edit Page** button that takes you into the mode for adding, modifying and deleting Web Parts from the page.
        1. 
        2. Figure 36
        3. The Edit Page button
     3. Click on the **Add a Web Part** link or make sure the Left Web Part zone is selected (indicated by a different blue color in the left zone) and use the contextual **Page Tools** ribbon tab.
     4. Click on the **Insert** tab on the ribbon and click on the **Web Part** button.
        1. 
        2. Figure 37
        3. Insert a new Web Part
        4. **Note:** At this point you should see the new SharePoint 2010 UI for adding new Web Parts to a page. You should be able to browse through the standard Web Part Categories such as Lists and Libraries, Authoring and Content Rollup. You should also be able to see a new custom category named Contoso Web Parts which you created when you edited the Group property value inside the **elements.xml** file.
     5. Select the **Contoso Web Parts** category in the left-hand section. In the right-hand section, click on your web part, named **The "Hello" Web Part**.
     6. Make sure the dropdown box with the **Add Web Part to** name,has the **Left** value, and click the **Add** button to add the Web Part instance to the page.
        1. 
        2. Figure 38
        3. Select a new Web Part to add
  4. At this point, you have gone through all the required steps to create, deploy and test a Web Part. The test instance of your Web Part should look like the following screenshot.
     1. 
     2. Figure 39

Web Part inserted in Home page

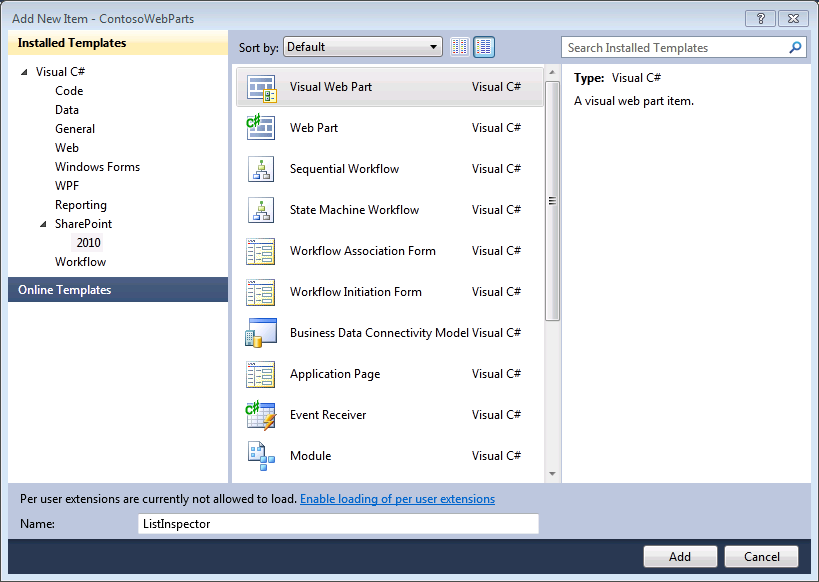
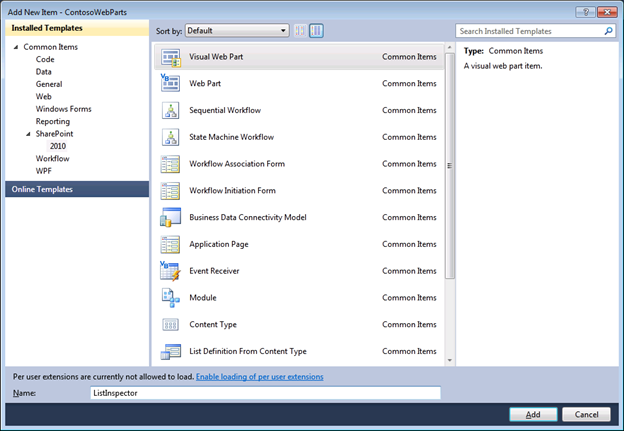
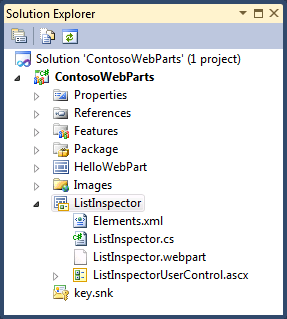
# Next Step

* 1. Exercise 5: Creating a Visual Web Part with AJAX Behavior

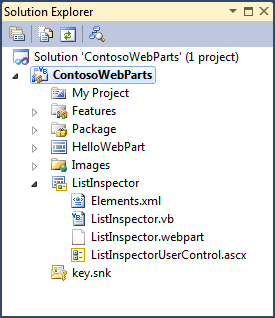
Exercise 5: Creating a Visual Web Part with AJAX Behavior

* 1. In this exercise, you will add a second Web Part using the Visual Web Part template. This template, allows you to create the UI for a Web Part using an ASP.NET User Control and the Visual Studio User Control Designer. You will also use the UpdatePanel control from ASP.NET AJAX to give your Web Part a Web 2.0 user experience eliminating postbacks.

Task 1 – Adding a Visual Web Part

* 1. In this task, you will use the Visual Web Part template to create a new SharePoint Project Item in the project.
  2. Open Microsoft Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
  3. Open the solution file **ContosoWebParts.sln** located under the **Source\Ex5-AjaxVisualWebPart\begin\** (choosing the folder that matches the language of your preference.) Alternatively, you may continue working with the solution obtained after completing the previous exercise.
  4. Add a new Web Part to the **ContosoWebParts** project named **ListInspector**. To do this:
     1. Right-click on the **ContosoWebParts** project in Solution Explorer, point to **Add** and select **New Item**.
     2. Select the Visual Web Part project item template, give it a name of **ListInspector** and click **Add**.
        1. 
        2. Figure 40
        3. Adding a new Visual Web Part (C#)
        4. 
        5. Figure 41
        6. Adding a new Visual Web Part (VB)
  5. Inspect the SharePoint Project Item node for the Web Part named ListInspector. Verify that it contains the SharePoint Project Item files named **Elements.xml**, **ListInspector.cs** / **ListInspector.vb**, **ListInspector.webpart**, **ListInspectorUserControl.ascx** and **ListInspectorUserControl.ascx.cs** / **ListInspectorUserControl.ascx.vb**.
     1. 
     2. Figure 42

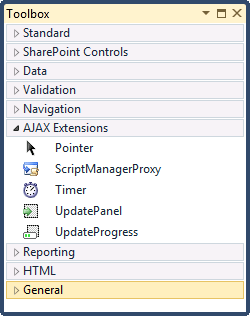
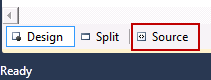
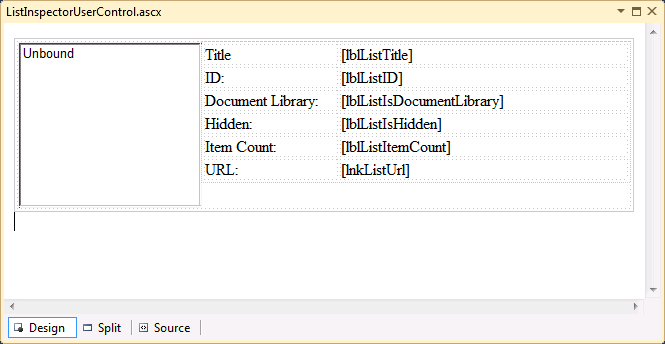
Visual Web Part node structure (C#)

* + 1. 
    2. Figure 43

Visual Web Part node structure (VB)

* + 1. **Note:** The structure of the Visual Web Part is similar to the Web Part. Additionally, it includes a user control (**ListInspectorUserContro.ascx**) which allows the developer to create a rich UI.
  1. Open the Web Part Description file, named **ListInspector.webPart,** and examine the XML content inside. You should see the XML already contains two property elements for the Title property and the Description property.
  2. Replace the **<data>** element of the Web Part Description with the following content.
     1. *(Code Snippet – SharePoint Tools Lab – ListInspector Description)*
     2. XML
     3. <?xml version="1.0" encoding="utf-8"?>
     4. <webParts>
     5. <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
     6. <metaData>
     7. <type name="ContosoWebParts.ListInspector.ListInspector, $SharePoint.Project.AssemblyFullName$" />
     8. <importErrorMessage>$Resources:core,ImportErrorMessage;</importErrorMessage>
     9. </metaData>
     10. **<data>**
     11. **<properties>**
     12. **<property name="Title" type="string">ListInspector</property>**
     13. **<property name="Description" type="string">Web Part which shows all the lists in the current site and allows you to get several of its property values</property>**
     14. **<property name="ChromeType" type="chrometype">TitleAndBorder</property>**
     15. **<property name="CatalogIconImageUrl" type="string">\_layouts/images/ContosoWebParts/WebPartIcon.gif</property>**
     16. **<property name="TitleIconImageUrl" type="string">\_layouts/images/ContosoWebParts/WebPartIcon.gif</property>**
     17. **</properties>**
     18. **</data>**
     19. </webPart>
     20. </webParts>
  3. Open the **Elements.xml** and modify the URL attribute of File element to ensure that it is unique. To do this, add "**ContosoWebPart\_**" to the beginning of the URL property value.
  4. Change the value for the Group property to "**Contoso Web Parts**". Elements.xml should look as follows:
     1. XML
     2. <?xml version="1.0" encoding="utf-8"?>
     3. <Elements xmlns="http://schemas.microsoft.com/sharepoint/" >
     4. <Module Name="ListInspector" List="113" Url="\_catalogs/wp">
     5. **<File Path="ListInspector\ListInspector.webpart" Url="ContosoWebParts\_ListInspector.webpart" Type="GhostableInLibrary" >**
     6. **<Property Name="Group" Value="Contoso Web Parts" />**
     7. **</File>**
     8. </Module>
     9. </Elements>

Task 2 – Adding Markup to the Web Part

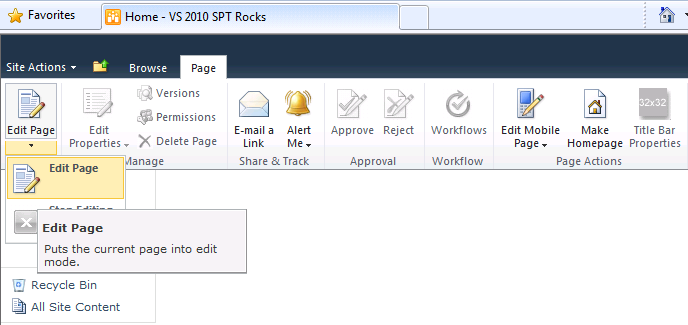
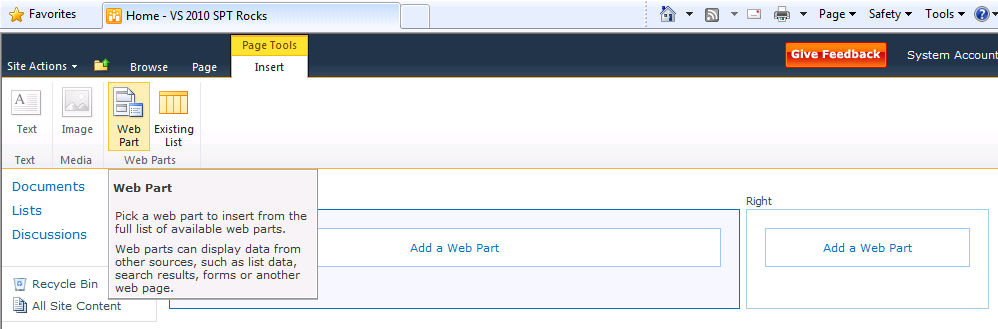
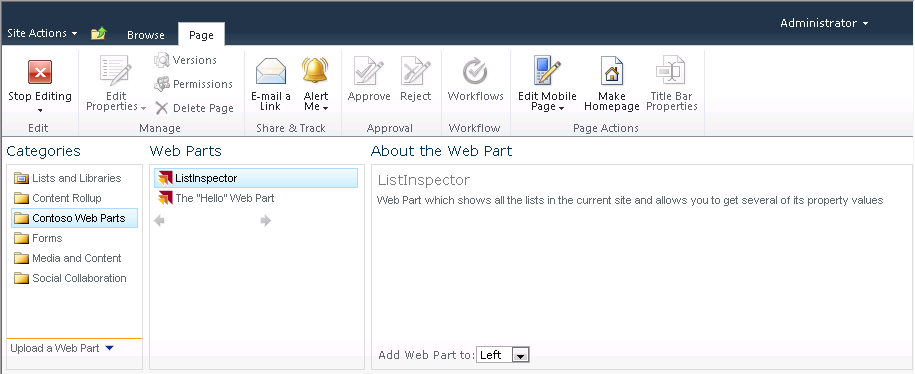
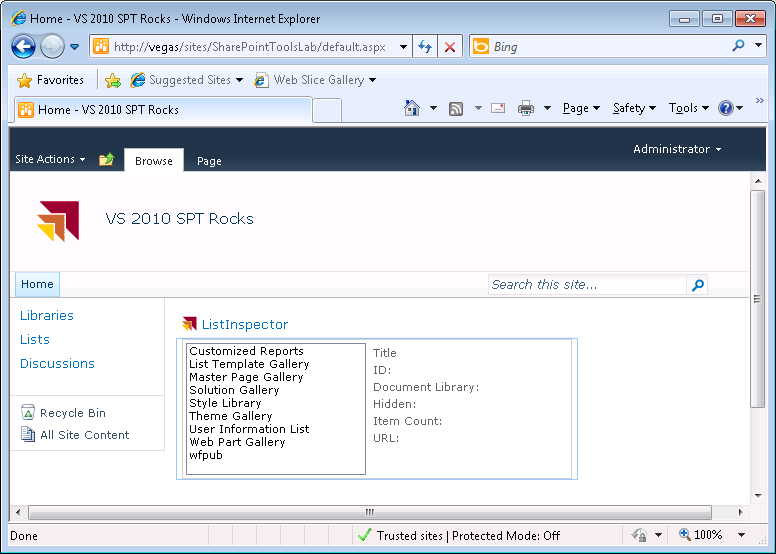
* 1. In this task you will add the markup necessary to create the elements of the WebPart.
  2. Add reference to **System.Web.Extensions**. To do this:
     1. In the Solution Explorer, right-click on the **ContosoWebParts** project, and choose Add Reference.
     2. In the **.NET** tab, search for the **System.Web.Extensions** component and click **OK**
        1. **Note:** This reference is required because developing the new Visual Web Part will involve the UpdatePanel control from ASP.NET AJAX.
  3. In the Solution Explorer, right-click on the **ListInspectorUserControl.ascx** file and choose View Designer to open the User Control in Design mode.
     1. **Note:** The designer should be empty at first as no controls or HTML text has been added.
  4. From the **AJAX Extensions** section of the Toolbox, drag and drop the **UpdatePanel** control onto the user control designer.
     1. **Note:** You can enable the Toolbox from the **View\Toolbox** menu
     2. 
     3. Figure 44
     4. Ajax Extensions Toolbox
  5. In the Bottom Left of the Designer, select **Source**, to switch the User Control back to code view.
     1. 
     2. Figure 45
     3. Switching to Source View
     4. **Note:** Note that the User Control designer now contains a reference to the **System.Web.Extensions**. This is because you added the **UpdatePanel** control.
     5. ASP.NET
     6. ...
     7. Inherits="ContosoWebParts.ListInspector.ListInspectorUserControl" %>
     8. **<asp:UpdatePanel ID="UpdatePanel1" runat="server">**
     9. **</asp:UpdatePanel>**
  6. Inside the **UpdatePanel** element, add a **ContentTemplate** as shown in the following code block.
     1. *(Code Snippet – SharePoint Tools Lab – ListInspector WebPart ContentTemplate)*
     2. ASP.NET
     3. <asp:UpdatePanel ID="UpdatePanel1" runat="server">
     4. **<ContentTemplate>**
     5. **<table style="width: 100%; border-style: solid; border-color: #CCCCCC; border-width: 1px;">**
     6. **<tr>**
     7. **<td style="width: 180px; vertical-align: top;">**
     8. **<asp:ListBox ID="lstLists" runat="server" Width="100%" Rows="10" AutoPostBack="true"**
     9. **OnSelectedIndexChanged="lslLists\_SelectedIndexChanged"></asp:ListBox>**
     10. **</td>**
     11. **<td style="width: auto; vertical-align: top;">**
     12. **<table style="width: 100%;">**
     13. **<tr>**
     14. **<td style="width: 132px;">**
     15. **Title**
     16. **</td>**
     17. **<td style="width: auto;">**
     18. **<asp:Label ID="lblListTitle" runat="server"></asp:Label>**
     19. **</td>**
     20. **</tr>**
     21. **<tr>**
     22. **<td>**
     23. **ID:**
     24. **</td>**
     25. **<td>**
     26. **<asp:Label ID="lblListID" runat="server"></asp:Label>**
     27. **</td>**
     28. **</tr>**
     29. **<tr>**
     30. **<td>**
     31. **Document Library:**
     32. **</td>**
     33. **<td>**
     34. **<asp:Label ID="lblListIsDocumentLibrary" runat="server"></asp:Label>**
     35. **</td>**
     36. **</tr>**
     37. **<tr>**
     38. **<td>**
     39. **Hidden:**
     40. **</td>**
     41. **<td>**
     42. **<asp:Label ID="lblListIsHidden" runat="server"></asp:Label>**
     43. **</td>**
     44. **</tr>**
     45. **<tr>**
     46. **<td>**
     47. **Item Count:**
     48. **</td>**
     49. **<td style="margin-left: 40px">**
     50. **<asp:Label ID="lblListItemCount" runat="server"></asp:Label>**
     51. **</td>**
     52. **</tr>**
     53. **<tr>**
     54. **<td>**
     55. **URL:**
     56. **</td>**
     57. **<td style="margin-left: 40px">**
     58. **<asp:HyperLink ID="lnkListUrl" runat="server" Target="\_blank" />**
     59. **</td>**
     60. **</tr>**
     61. **</table>**
     62. **</td>**
     63. **</tr>**
     64. **</table>**
     65. **</ContentTemplate>**
     66. </asp:UpdatePanel>
  7. Switch the User Control back into Design view and verify that you can see the table layout.
     1. 
     2. Figure 46

ListInspector Visual Web Part table layout

Task 3 – Implementing the Web Part

* 1. In this task, you will add the code needed to implement the web part’s functionality.
  2. In the Solution Explorer, right-click on **ListInspectorUserControl.ascx** andchoose View Code (**F7**) to open its code behind.
  3. Add a the following namespace directive:
     1. C#
     2. **using Microsoft.SharePoint;**
     3. Visual Basic
     4. **Imports Microsoft.SharePoint**
  4. Replace the default class definition with the following:
     1. *(Code Snippet - SharePoint Tools Lab – ListInspectorUserControl Class Definition CSharp)*
     2. C#
     3. **public partial class ListInspectorUserControl : UserControl**
     4. **{**
     5. **protected Guid selectedListId = Guid.Empty;**
     6. **protected bool updateListProperties = false;**
     7. **}**
     8. *(Code Snippet – SharePoint Tools Lab – ListInspectorUserControl Class Definition VB)*
     9. Visual Basic
     10. **Partial Public Class ListInspectorUserControl**
     11. **Inherits UserControl**
     12. **Protected selectedListId As Guid = Guid.Empty**
     13. **Protected updateListProperties As Boolean = False**
     14. **End Class**
  5. Add a method named **lslLists\_SelectedIndexChanged** to the ListInspectorUserControl class.
     1. *(Code Snippet – SharePoint Tools Lab – SelectedIndexChanged Method CSharp)*
     2. C#
     3. **protected void lslLists\_SelectedIndexChanged(object sender, EventArgs e)**
     4. **{**
     5. **selectedListId = new Guid(lstLists.SelectedValue);**
     6. **updateListProperties = true;**
     7. **}**
     8. *(Code Snippet – SharePoint Tools Lab – SelectedIndexChanged Method VB)*
     9. Visual Basic
     10. **Protected Sub lslLists\_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles lstLists.SelectedIndexChanged**
     11. **selectedListId = New Guid(lstLists.SelectedValue)**
     12. **updateListProperties = True**
     13. **End Sub**
     14. **Note:** The ListInspectorWeb Part will not work until you add this method implementation. This is because the pre-provided table layout text you pasted, contains a listbox named **lstLists** containing an attribute which attaches it to an event handler named **lslLists\_SelectedIndexChanged**.
  6. Add an overridden implementation of the **OnPreRender** method
     1. *(Code Snippet – SharePoint Tools Lab – OnPreRender Method CSharp)*
     2. C#
     3. **protected override void OnPreRender(EventArgs e)**
     4. **{**
     5. **if ((lstLists.SelectedIndex > -1) && (!updateListProperties))**
     6. **{**
     7. **selectedListId = new Guid(lstLists.SelectedValue);**
     8. **}**
     9. **lstLists.Items.Clear();**
     10. **SPWeb site = SPContext.Current.Web;**
     11. **foreach (SPList list in site.Lists)**
     12. **{**
     13. **ListItem listItem = new ListItem(list.Title, list.ID.ToString());**
     14. **lstLists.Items.Add(listItem);**
     15. **}**
     16. **if (selectedListId != Guid.Empty)**
     17. **{**
     18. **lstLists.Items.FindByValue(selectedListId.ToString()).Selected = true;**
     19. **}**
     20. **if (updateListProperties)**
     21. **{**
     22. **SPList list = SPContext.Current.Web.Lists[selectedListId];**
     23. **lblListTitle.Text = list.Title;**
     24. **lblListID.Text = list.ID.ToString().ToUpper();**
     25. **lblListIsDocumentLibrary.Text = (list is SPDocumentLibrary).ToString();**
     26. **lblListIsHidden.Text = list.Hidden.ToString();**
     27. **lblListItemCount.Text = list.ItemCount.ToString();**
     28. **lnkListUrl.Text = list.DefaultViewUrl;**
     29. **lnkListUrl.NavigateUrl = list.DefaultViewUrl;**
     30. **}**
     31. **}**
     32. *(Code Snippet – SharePoint Tools Lab – OnPreRender Method VB)*
     33. Visual Basic
     34. **Protected Overrides Sub OnPreRender(ByVal e As System.EventArgs)**
     35. **If (lstLists.SelectedIndex > -1) AndAlso Not updateListProperties Then**
     36. **selectedListId = New Guid(lstLists.SelectedValue)**
     37. **End If**
     38. **lstLists.Items.Clear()**
     39. **Dim site = SPContext.Current.Web**
     40. **For Each list As SPList In site.Lists**
     41. **Dim listItem = New ListItem(list.Title, list.ID.ToString())**
     42. **lstLists.Items.Add(listItem)**
     43. **Next**
     44. **If Not selectedListId = Guid.Empty Then**
     45. **lstLists.Items.FindByValue(selectedListId.ToString()).Selected = True**
     46. **End If**
     47. **If updateListProperties Then**
     48. **Dim list = SPContext.Current.Web.Lists(selectedListId)**
     49. **lblListTitle.Text = list.Title**
     50. **lblListID.Text = list.ID.ToString().ToUpper()**
     51. **lblListIsDocumentLibrary.Text = (TypeOf list Is SPDocumentLibrary).ToString()**
     52. **lblListIsHidden.Text = list.Hidden.ToString()**
     53. **lblListItemCount.Text = list.ItemCount.ToString()**
     54. **lnkListUrl.Text = list.DefaultViewUrl**
     55. **lnkListUrl.NavigateUrl = list.DefaultViewUrl**
     56. **End If**
     57. **End Sub**
     58. **Note:** The code above clears the items from the **lstLists** control and then add a new list item for each **SPList** item in the current site. When adding ListItem instances to the ListBox control, it uses the list title as the ListItem Text and the list's identifying GUID as the ListItem Value.
     59. If the **UpdateListProperties** field contains a value of true, then it will update all the Label controls on the right-hand side of the table layout to display properties of the selected list.
  7. Press **CTRL+SHIFT+B** to build the solution.

Exercise 5: Verification

* 1. Run the **Deploy** command which will retract the previously deployed solution and will deploy the updated solution package.
  2. Open Internet Explorer, and browse to **http://localhost/sites/SharePointToolsLab**. Follow these steps to go through the activation process.
     1. Point to **Site Actions** and select **Site Settings** to navigate to the Site Settings page.
     2. Inside the **Site Collection Administration** section of the Site Settings page, click on the **Site collection features** link to navigate to the Site Collection Administration > Features page.
     3. Locate the **Contoso Web Parts** and **activate** it.
        1. **Note:** If it was already active, then deactivate first and then activate it. The key point here is that feature activation is what provisions the .webpart file into the Web Part Gallery which will allow you to test your work.
  3. Add a test instance of the Web Part you have just created to a Web Part Page. To do this:
     1. Navigate to the home page **http://localhost/sites/SharePointToolsLab**. The ribbon at the top of the page should have a tab titled **Page**.
     2. Click on the **Page** tab and then select the command from the **Edit Page** button that takes you into the mode for adding, modifying and deleting Web Parts from the page.
        1. 
        2. Figure 47
        3. The Edit Page button
     3. Click on the **Add a Web Part** link or make sure the Left Web Part zone is selected (indicated by a different blue color in the left zone) and use the contextual **Page Tools** ribbon tab.
     4. Click on the **Insert** tab on the ribbon and click on the **Web Part** button.
        1. 
        2. Figure 48
        3. Insert a new Web Part
     5. Select the **Contoso Web Parts** category in the left-hand section. In the right-hand section, click on your web part, named **ListInspector**.
     6. Make sure the dropdown list with the **Add Web Part to** name, has the **Left** value, and click the **Add** button to add the Web Part instance to the page.
        1. 
        2. Figure 49
        3. Select a new Web Part to add
  4. At this point, you have gone through all the required steps to create, deploy and test a Web Part. The test instance of your Web Part should look like the following screenshot.
     1. 
     2. Figure 50

Web Part inserted in Home page

# Next Step

* 1. Summary

Summary

* 1. In this lab, you created your first SharePoint project using Visual Studio 2010 and the SharePoint Developers Tools. You have added your first Feature and added a SharePoint mapped folder to the project.
  2. Among other tasks you implemented a Feature Event Receiver, a SharePoint Web Part and a SharePoint Visual Web Part which included basic Ajax capabilities.
  3. You also learned how to configure a SharePoint project to deploy it to a SharePoint Server installation, and went through the necessary steps to test and debug your developedproject.