Lab 02: Developing Your First Feature

**Lab Time**: 45 Minutes

**Lab Directory**: C:/Student/Labs/02\_Features

**Lab Overview**: You are going to start this lab by creating a new Site Collection using the Windows SharePoint Services (WSS) Central Administration Web site. If you are new to using WSS, you should feel free to spend a little time getting familiar with how users create new lists and document libraries. However, the primary goal of this lab is for you to create and test your first WSS Feature.

1. You will create a new Site Collection for Lab02 and optionally add some task lists / document libraries to familiarize ourselves with basic processes in WSS administration.
2. Next you will create a Visual Studio project from scratch and add all the key ingredients to create the classic 'Hello World' Feature (i.e. you will add a menu item on the Site Actions Menu called Hello World that will hyperlink to our Litware Corporate Portal site).
3. Then you will go through the required steps to deploy and test it.
4. Next you will add an event handler that fires and executes code against the WSS object model each time the Feature is activated within a site (i.e. we will add an event handler to additionally alter the site title to Hello World when activated and then return the site title back to the original setting when deactivated).
5. Along the way, you will go through the required steps to make our assembly strongly named and also to deploy it into the Global Assembly Cache (GAC).

# Setup: enabling intellisense support for SharePoint CAML code in Visual Studio

1. In order to enable Intellisense support for SharePoint CAML documents in Visual Studio you need to add the sharepoint\_catalog.xml file to the Visual Studio Directory "C:\Program Files\Microsoft Visual Studio 9.0\Xml\Schemas\".
   1. Navigate to your **Student\Labs\02\_Features\CAML\_IntelliSense**\ folder
   2. Double click on the **install.bat** file that you find there.
   3. Run or Close and re-Open Visual Studio.
   4. You should now have IntelliSense when you program SharePoint CAML files in Visual Studio.

**For Example:** After you add an xml file named “elements.xml” to your application in Visual Studio:

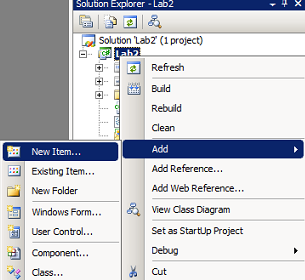
* **Open** the file by double clicking on it in the Solution Explorer
* In the **XML Document Properties** window set the **Schema** (by clicking on the ellipsis “…”) and select the wss.xsd schema and click **OK**.
* You should now see IntelliSense when you click back into the open document and type a “<” character. You should see **Elements, Feature,** and **List**, among other choices.

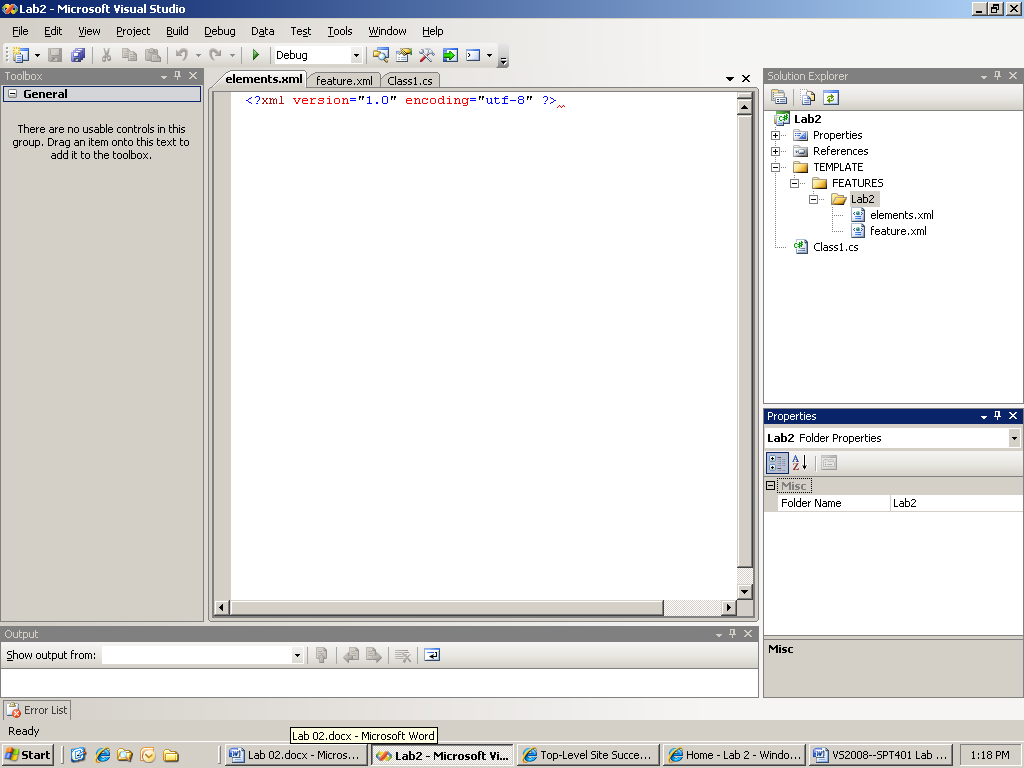
# Exercise 1: Provisioning a new Site Collection

1. Make sure you are logged on as **LITWAREINC\Administrator**. Launch the WSS Central Administration Web site by using the command in the Windows Start menu. You can find the command at **Start >> Administrative Tools >> SharePoint 3.0 Central Administration**.
2. Once the home page of the WSS Central Administration Web site appears, click on the **Application Management** tab at the top of the page to get to the Application Management page. Under the **SharePoint Site Management** section, click the link with the caption **Create site collection**. This will take you to the page where you can create a new site collection and a new top-level site.  On the **Create Site Collection** page, fill in the required information (see below for instructions) to create a new site collection.
   1. Make sure the Web Application used for site creation is the one named Public Internet Site that is accessible through the URL **http://litwareinc.com**.
   2. Enter **Lab 2** as the title.
   3. Create the new site collection so that its URL is **http://litwareinc.com/sites/Lab2**.
   4. Under the **Template Selection** section, look at all the sites templates that are available. Choose **Blank Site** from the Collaboration tab as the site template for the new top-level site that is automatically created.
   5. Assign the primary site collection owner as **LITWAREINC\Administrator**.  Be sure to verify this by using **Ctrl+k** or by clicking the check person icon checkIcon (your entry should become LitwareInc Administrator )
   6. Leave the **Quota Template** with the default setting of **No Quota**
   7. Click **OK** to create the new site collection and top-level site. Once you see the page that confirms everything has been created, navigate to the top-level site using the browser by clicking the link [http://litwareinc.com/sites/Lab2](http://litwareinc.com/sites/Lab1) .
3. Now take the time to familiarize yourself with the site you just created and make a few administrative changes through links off the **Site Settings** page. The title you entered of Lab 2 is just a little too boring.
   1. Choose the **Site Settings** command under the **Site Actions** menu.
   2. Once you are at the **Site Settings** page, click the link with the caption **Title, description and icon** under the **Look and Feel** section.
   3. When you reach the Title, Description, and Icon page, enter a different title that is a variation such as **Lab 2 Test Site**. Click **OK** and confirm that the home page now reflects the new site title.
4. If you are new to SharePoint and haven't used the product much as a user, take a few minutes and create a few new lists. (If you are already comfortable with creating and working with lists, you should move on to the next exercise).
   1. In particular, navigate to the **Create** page (available from the **Site Actions** menu) and create a new list from the **Contacts** list type and then add one or two contacts using whatever test data you'd like.
   2. After creating the **Contacts** list, create a new document library. Click on the **New** button and create and save a new Word document back to the document library.

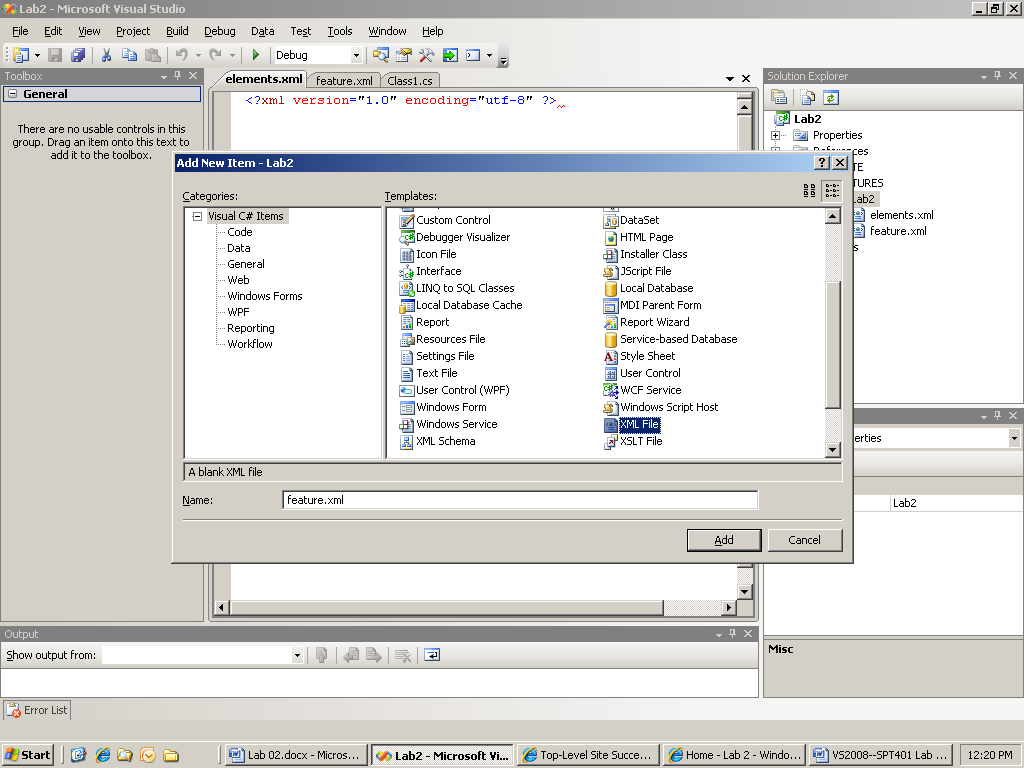
# Exercise 2: Creating the 'Hello World' Feature

1. In this exercise you will create a new Visual Studio project to develop a very simple Feature. Start by opening Visual Studio 2008 and creating a new Class Library DLL project named Lab2 by going to **File > New > Project**, then choosing your preferred language in the left column of the box that opens up, and then selecting “Class Library” in the adjacent window. The screen shots for this lab were designed using C#, You may use either C# or Visual Basic .Net as there are solution files for the lab in both languages. While you can create this new project any place you would like, you might prefer to create it at the following path so the project files will reside inside the same directory structure as all the other lab exercises. **C:\Student\Labs\02\_Features\Lab**
2. Within your new Visual Studio project, add a folder to the root directory of the current project named **TEMPLATE** by right clicking on “Lab 2” in the **Solution Explorer** window, hovering your mouse over **Add**, and selecting **New Folder**.



* 1. Once you have created the **TEMPLATE** directory, create another directory inside that, named **FEATURES**.
  2. Finally, create another directory inside the **Features** directory using the same name as the name of the Feature project which in this case is **Lab2**.
  3. When you are done creating these folders, your project should have a folder structure like the picture below: 

Note this picture shows what your project should look like after finishing steps 3 and 4 and if you chose the C# language.

1. Create an XML file inside the **/TEMPLATES/FEATURES/Lab2** directory by right clicking on the **Lab2** folder under the **Features** folder, hovering your mouse over **Add**, and then selecting **New Item**. Select **XML File** in the dialog window that opens, and name it **feature.xml**. Click the **Add** button. 

Inside the new file is where you will add the XML-based information that defines the high-level attributes of the Feature itself.

* 1. **Open** the file by double clicking on it in the **Solution Explorer**.
  2. In the **XML Document Properties** window set the **Schema** (by clicking on the ellipsis “…”) and select the wss.xsd schema and click **OK**.
  3. You should now see IntelliSense when you click back into the open document and type a “<” character.
  4. Add the following XML content to the feature.xml file to define a top-level Feature element along with attributes that define the Feature itself.

<Feature Id="B2CB42E2-4F0A-4380-AABA-1EF9CD526F20"

Title="Lab 2 - Getting Started with WSS Development"

Description="This is my very first custom feature"

Scope="Web"

Hidden="FALSE"

ImageUrl="menuprofile.gif"

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

<ElementManifests>

<ElementManifest Location="elements.xml" />

</ElementManifests>

</Feature>

1. Next, create the **elements.xml** file in the same directory.
   1. **Open** the file by double clicking on it in the Solution Explorer
   2. In the **XML Document Properties** window set the **Schema** (by clicking on the ellipsis “…”) and select the wss.xsd schema and click **OK**.
   3. You should now see IntelliSense when you click back into the open document and type a “<” character.
   4. Add the following XML which defines a **CustomAction** element which will allow the user to navigate to another site that has already been defined within the current VPC image:

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

<CustomAction

Id="SiteActionsToolbar"

GroupId="SiteActions"

Location="Microsoft.SharePoint.StandardMenu"

Sequence="100"

Title="Hello World"

Description="A custom menu item to navigate to another site"

ImageUrl="\_layouts/images/crtsite.gif" >

<UrlAction Url="http://litwareinc.com/sites/IntranetPortal"/>

</CustomAction>

</Elements>

1. Now that you have implemented the Lab2 Feature by creating the **feature.xml** file and the **elements.xml**, it's time to deploy it. You will do this by creating a new batch file and configuring the current Visual Studio project to run this batch file each time you rebuild the current project.
   1. Start by creating a new batch file in the root directory of your project named **Install.bat**.
      1. You can create a batch file by navigating to your **Solution Explorer** in Visual Studio and right click on your project **Lab2** (i.e. the top item in the solution explorer picture from step 2)
      2. Click on **Add** > **New Item...**
      3. Choose **Text File,** name the file **Install.bat** and then click **Add.**
   2. Add the following instructions to **Install.bat**.

@SET TEMPLATEDIR="c:\program files\common files\Microsoft shared\web server extensions\12\Template"

@SET STSADM="c:\program files\common files\Microsoft shared\web server extensions\12\bin\stsadm"

@SET GACUTIL="c:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\gacutil.exe"

Echo Installing Lab2.dll in GAC

REM %GACUTIL% -if bin\debug\Lab2.dll

Echo Copying files to TEMPLATE directory

xcopy /e /y TEMPLATE\\* %TEMPLATEDIR%

Echo Installing Feature

%STSADM% -o installfeature -filename Lab2\feature.xml -force

IISRESET

Note that the call to %GACUTIL% is commented out with an REM statement. You will uncomment this instruction in the next exercise after you have configured the Lab2 project to build its output assembly with a strong name.

1. Now that you have created the **Install.bat** file, it's time to configure the Visual Studio project to run it each time you rebuild.
   1. Go to the **Project Properties** for the Lab2 project by right clicking the project in **Solution Explorer** and choosing **Properties** from the menu. If using VB, follow the instructions in step B.  If using C# go to step C.
   2. **Visual Basic .Net directions**:
      1. Navigate to the **Compile** tab
      2. Click on the **Build Events...** button (scroll down if necessary to see this button)
      3. Add the following Post-build event command line instructions.

cd $(ProjectDir)

Install.bat

Note that the first line with **cd $(ProjectDir)** is required to change the current directory to that of the project directory. The second line runs the batch file **Install.bat** to copy the Feature files to the correct location and install the Feature with the **InstallFeature** operation of the command-line **stsadm.exe** utility.

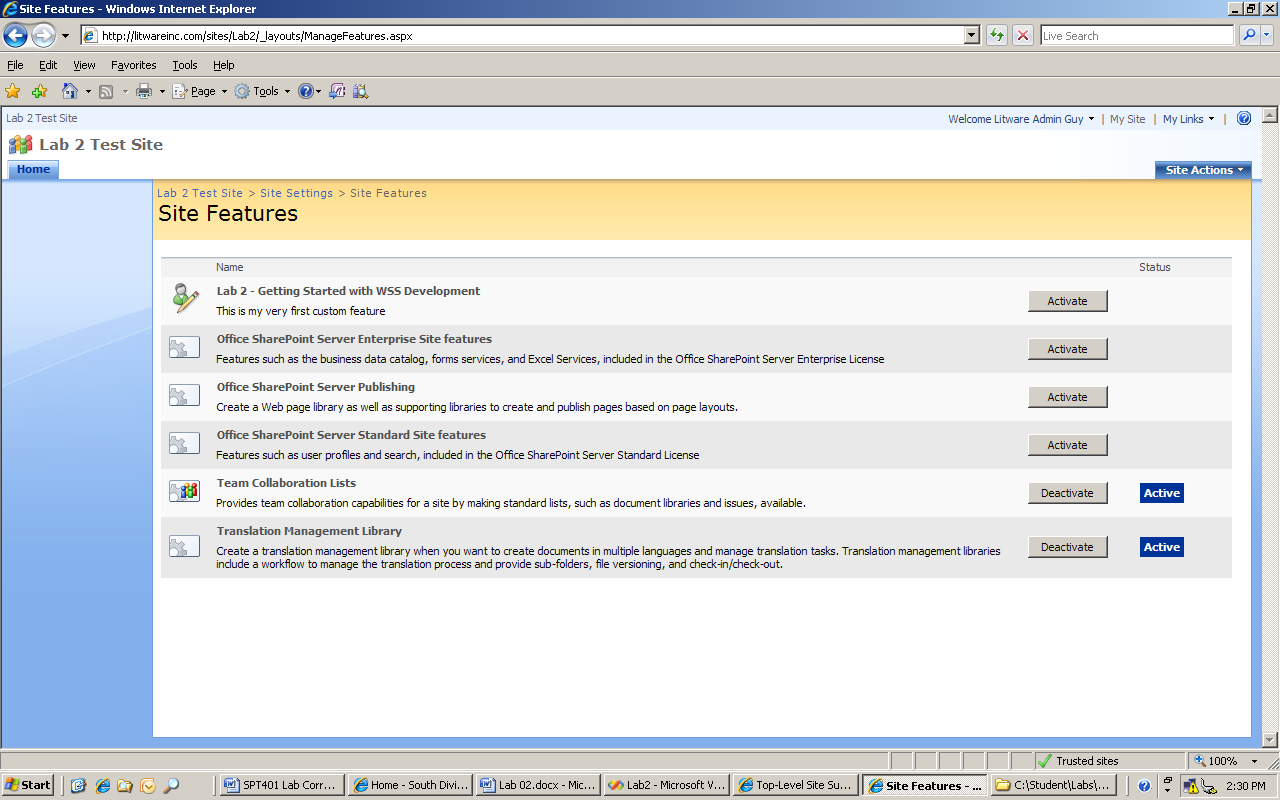
* 1. **C# directions:**
     1. Navigate to the **Build Events** tab.
     2. Add the following Post-build event command line instructions.

cd $(ProjectDir)

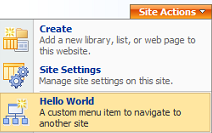
Install.bat

Note that the first line with **cd $(ProjectDir)** is required to change the current directory to that of the project directory. The second line runs the batch file **Install.bat** to copy the Feature files to the correct location and install the Feature with the **InstallFeature** operation of the command-line stsadm.exe utility.

1. Build your project (i.e. **Build** drop down menu > **Build Lab2** ) (this will build and install your feature for use)
2. Once the Feature has been properly installed, you should now be able to activate it within context of a site.
   1. Go to the top-level site of the site collection you created in the previous exercise (http://litwareinc.com/sites/Lab2)
   2. Navigate to the **Site Settings** page through the **Site Actions** menu.
   3. Under the **Site Administration** section, click the link with the title **Site Features**. This should take you to a page where you can activate the Feature.



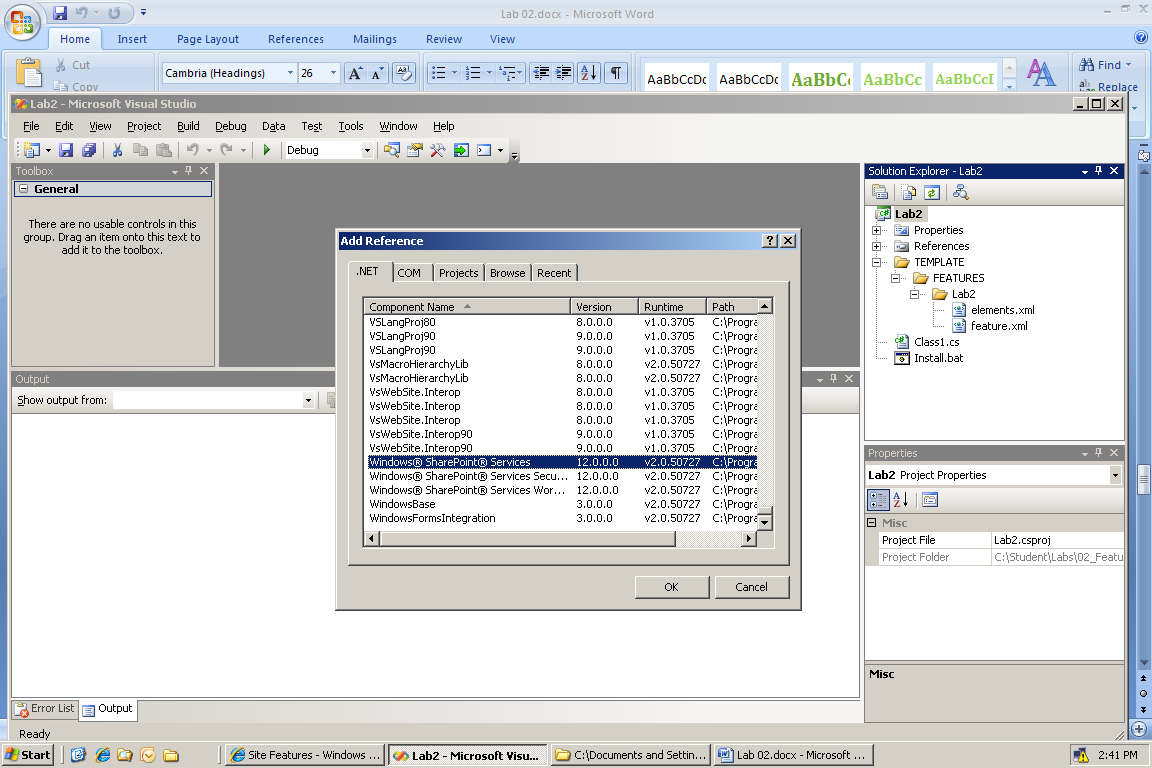
* 1. Activate the **Feature** named **Lab 2 - Getting Started with WSS Development** by clicking the **Activate** button next to the Lab 2 feature.
  2. Make sure your new menu item has been added to the **Site Actions** menu.



1. Now that you have successfully activated the Feature and tested the custom **Site Actions** menu command, you will learn how to deactivate a feature.
   1. Return to the Site Features page (**Site Actions > Site Settings >** **Site Features** (underneath the **Site Administration column**)
   2. Deactivate the Feature by clicking the **Deactivate** button next to the Lab2 feature.
   3. Now that you have deactivated the Lab2 Feature, you should be able to verify that the custom Site Actions menu command has been removed from the **Site Actions** drop down.

You have now witnessed the fundamental principle behind Features. Developers create various types of site elements that can be added or removed from a site through the process of activation and deactivation.

# Exercise 3: Adding an event handler to your Feature

1. Now it’s time to take the example of the Lab2 Feature a little further by adding an event handler with code that programs against the WSS object model.
   1. First, start by adding a project reference to **Microsoft.SharePoint.dll** which is listed as **Windows SharePoint® Services** in Visual Studio's **Add Reference** dialog.
      1. In **Visual Studio** go to the **Solution Explorer** and right click on your project **Lab2** and choose **Add Reference...**
      2. From the **Add Reference** dialog box go to the **.Net** tab, scroll down the list of assemblies and select the **Windows SharePoint® Services** component.
      3. Click the **OK** button. 
2. Now we need to set our **FeatureReceiver** class up to inject a new site title via the object model on Feature Activation and Reset this title to the original on Feature Deactivation. A feature receiver class always inherits from **SPFeatureReceiver**.
   1. **If Using VB.NET**
      1. Next, locate the source file named **Class1.vb** and rename it to **FeatureReceiver.vb**. Next, replace the content of **FeatureReceiver.vb** with the following code.

Imports System

Imports Microsoft.SharePoint

Public Class FeatureReceiver

    Inherits SPFeatureReceiver

    Public Overrides Sub FeatureActivated(ByVal properties As SPFeatureReceiverProperties)

        Dim site As SPWeb = properties.Feature.Parent

        ' track original site Title using SPWeb property bag

        site.Properties("OriginalTitle") = site.Title

        site.Properties.Update()

        ' update site title

        site.Title = "Hello World"

        site.Update()

    End Sub

    Public Overrides Sub FeatureDeactivating(ByVal properties As SPFeatureReceiverProperties)

        ' reset site Title back to its original value

        Dim site As SPWeb = properties.Feature.Parent

        site.Title = site.Properties("OriginalTitle")

        site.Update()

    End Sub

Public Overrides Sub FeatureInstalled(ByVal properties As SPFeatureReceiverProperties)

    End Sub

Public Overrides Sub FeatureUninstalling(ByVal properties As SPFeatureReceiverProperties)

    End Sub

End Class

* 1. **If Using C#**
     1. Next, locate the source file named **Class1.cs** and rename it to **FeatureReceiver.cs**. Next, replace the content of **FeatureReceiver.cs** with the following code.

using System;

using Microsoft.SharePoint;

namespace Lab2 {

public class FeatureReceiver : SPFeatureReceiver {

public override void FeatureActivated(SPFeatureReceiverProperties properties){

SPWeb site = properties.Feature.Parent as SPWeb;

//track original site title using spweb property bag

site.Properties["OriginalTitle"] = site.Title;

site.Properties.Update();

// update site title

site.Title = "Hello World";

site.Update();

}

public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {

// reset site Title back to its original value

SPWeb site = properties.Feature.Parent as SPWeb;

site.Title = site.Properties["OriginalTitle"];

site.Update();

}

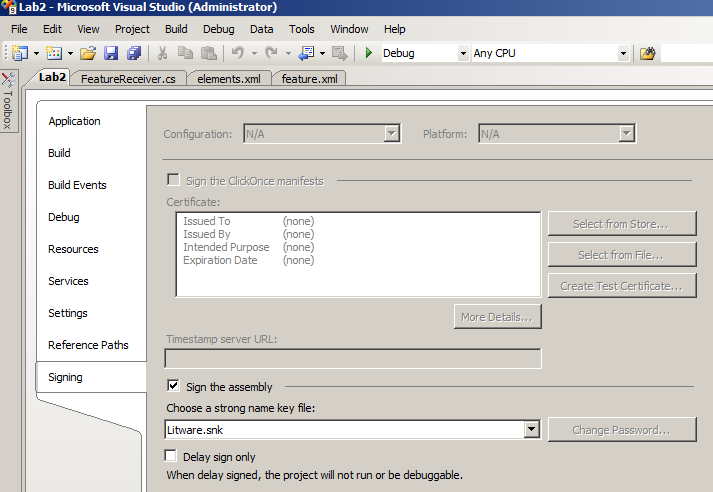
public override void FeatureInstalled(SPFeatureReceiverProperties properties){}

public override void FeatureUninstalling(SPFeatureReceiverProperties properties) {}

}

}

1. The next step is to update the **Lab2** project so that it's output assembly **Lab2.dll** is built with a strong name. This is a requirement for assemblies that contain Feature receiver classes which must be installed in the GAC in order to work properly.
   1. Copy the **Litware.snk** file from the **Resources** directory into your project directory
      1. In **Visual Studio** navigate to the **Solution Explorer** and rt click on your project **Lab2.**  Select **Add** > **Existing Item...**
      2. In the **Objects of type**: drop down select **All Files (\*.\*)**
      3. In the **Look in:** drop down navigate to **\Student\Labs\02\_Features\Resources** and select the **Litware.snk** file and click **Add**
   2. Configure the project properties of the **Lab2** project to build its output assembly **Lab2.dll** using this key file.
      1. In **Visual Studio** navigate to the **Solution Explorer** and rt click on your project **Lab2** select **Properties**
      2. Navigate to the **Signing** tab
      3. Put a check in the **Sign the assembly** checkbox
      4. From the **Choose a strong name key file:** dropdown list select your **Litware.snk** file.
      5. Close the **Project Properties window.**



* + 1. **re-Build** your project to create a dll with this new strong name.

1. Now that you have implemented the Feature receiver class, the next step is to update the feature.xml file with two new attributes so that WSS knows that there is at least one event handler that should be fired whenever the Feature is activated or deactivated.
   1. In your **Solution Explorer** open the **Feature.xml** file
   2. Add the **ReceiverAssembly** attribute and the **ReceiverClass** attribute to the existing **Feature.xml** file as shown here.  
      **Note**: you can obtain the full 4-part assembly name (for use in the **ReceiverAssembly** Attribute) using the **Reflector** utility that is in your **\Student\Resources** directory.  
      **Further Note:** In order to make this work you must do this **AFTER** you have added the keyfile **Litware.snk** to the **Project Properties**  and then re-**build** your project (so that your assembly (dll) will contain the strong name attribute (i.e. **PublicKeyToken**)) but before you add the attributes to the **Elements.xml** file.
   3. In the **...\Student\Resources\Reflector.exe** Utility you will  need to examine the **Lab2.dll**.   **File > Open...** (VB.NET or C#) Navigate to the **...Lab2\bin\Debug directory** open the **Lab2.dll** file.
   4. Select the **Lab2** Assembly in the main Reflector utility window and look at the bottom for the **Name:** entry.  You can copy this entry to use for your **ReceiverAssembly** attribute.
   5. Your **ReceiverClass** attribute is made up of the **namespace**.**className**  **(Note**: these are typically the same as your **ProjectName**.**ClassFileName** if no alterations have been made to the Visual Studio Default choices.)

<Feature

Id="B2CB42E2-4F0A-4380-AABA-1EF9CD526F20"

Title="Lab 1 - Getting Started with WSS Development"

Description="This is my very first custom feature"

Version="1.0.0.0"

Scope="Web"

Hidden="FALSE"

ImageUrl="menuprofile.gif"

ReceiverAssembly="Lab2,Version=1.0.0.0,Culture=neutral,PublicKeyToken=b38a04419cc857d9"

ReceiverClass="Lab2.FeatureReceiver"

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

<ElementManifests>

<ElementManifest Location="elements.xml" />

</ElementManifests>

</Feature>

1. Next you need to update your **Install.bat** in order to add the strong named assembly into the GAC
   1. In **Visual Studio** navigate to **Solution Explorer** and open the **Install.bat** file.
   2. Remove just the **REM** keyword in the **REM** **%GACUTIL%**... line.  This will cause the Strongly Named dll to be injected into the GAC.
2. Once you have made these changes to the **feature.xml** and **install.bat** files, you should be able to test your work.
   1. **Rebuild** the Lab2 project so that Visual Studio runs the **Install.bat** file to copy the updated version of **feature.xml** file to the WSS Features directory and to reinstall the updated version of the Feature with WSS.  The build process should also compile **Lab2.dll** with a strong name and install it in the GAC.   
      **Note**: you would **normally** be required to run an **IISRESET** command any time you recompile to Lab2.dll to restart the IIS worker process. This is due to the fact that Features and assemblies loaded from the GAC are cached by WSS within the IIS worker process.  **HOWEVER** this is done for us automatically as the last line of our **Install.bat** file that we created earlier.
   2. Test your work by activating and deactivating the Feature. You should be able to see the site title changing back and forth each time you activate or deactivate the Lab2 Feature.