Lab 06: Creating Custom Web Parts

**Lab Time**: 60 Minutes

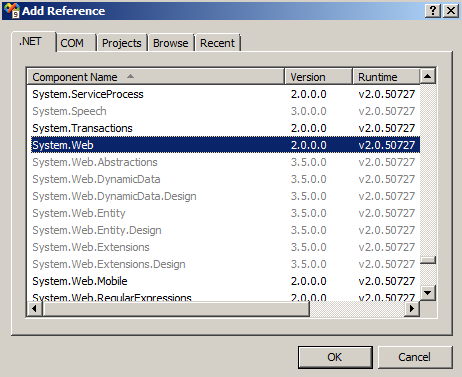
**Lab Directory**: C:/Student/Labs/06\_WebParts

**Lab Overview**: In this lab, you will create a simple "Hello World" Web Part and then proceed to modify it so that it will be able to calculate business information from field values within the lists of your sites.

* In exercise 1 you will create a simple Web Part based class library and see how to import it into SharePoint.
* In exercise 2 you will build on this by adding a conditional property to change the display of the text.
* In exercise 3 you will delve into the SharePoint object model to pull data out of a SharePoint list and aggregate it for display in our custom Web Part.

# Exercise 1: Create and test the traditional "Hello World" Web Part

In this exercise you will write a custom Web Part using Visual Studio 2008. You will be provided with a class library DLL project as a starting point.

1. Begin this lab by opening the  **..\06\_WebParts\Lab\VB\LitwareWebPartsLab.sln** or  **..\06\_WebParts\Lab\C#\LitwareWebPartsLab.sln** file in Visual Studio 2008.
   1. Take a moment and familiarize yourself with the project inside this solution named **LitwareWebPartsLab**. You should notice that the **LitwareWebPartsLab** project is a class library project that has **NOT** been configured to build an Assembly DLL with a strong name. (Hint: in your **Solution Explorer** right-click on your **LitwareWebPartsLab** project and select **Properties.** On the **Signing** tab you will find the **Sign the assembly** check box.  Verify that it is **NOT SELECTED**).
   2. **Note**: see <http://support.microsoft.com/kb/839300> for information about the issues with strongly named assemblies and \bin directory deployment in .Net.
2. Note there is already an existing class file named **RevenueWebPart.cs** or **RevenueWebPart.vb** with a class named **RevenueWebPart**. It is your mission to transform this standard boring class into a Web Part for use with the **Project Management** site you created earlier in this Course.
3. Add a reference to **System.Web.DLL**. This system assembly has types that you will need to write ASP.NET-style Web Parts.
   1. In your **Solution Explorer**, right-click on your **LitwareWebPartsLab** project and select **Add Reference...** Select **System.Web** from the list and click the **OK** button. 
4. Modify the **RevenueWebPart** class so that you have an Imports (**VB**) or using (**C#)** statement for this (**System.Web.UI.WebControls.WebParts**) namespace and then make your class inherit from the **WebPart** class defined inside the **WebParts** namespace.

C#

using System;

using System.Web.UI.WebControls.WebParts;

namespace LitwareWebPartsLab {

public class RevenueWebPart : WebPart {

// Web Part code goes here

}

}

VB.Net

Imports System

Imports System.Web.UI.WebControls.WebParts

Public Class RevenueWebPart Inherits WebPart

' Web Part code goes here

End Class

1. Inside the **RevenueWebPart** class, override the **RenderContents** method with a minimal "Hello, world" implementation.  (Note: you will need to add either an **Imports** (VB) or **using** (C#) statement for **System.Web.UI** as the **HtmlTextWriter** class exists in this namespace)

C#

using System;

using System.Web.UI;

using System.Web.UI.WebControls.WebParts;

namespace LitwareWebPartsLab {

public class RevenueWebPart: WebPart {

protected override void RenderContents(HtmlTextWriter writer) {

writer.Write("Hello, world");

}

}

}

VB.Net

Imports System

Imports System.Web.UI

Imports System.Web.UI.WebControls.WebParts

Public Class RevenueWebPart Inherits WebPart

Protected Overrides Sub RenderContents(ByVal writer As

System.Web.UI.HtmlTextWriter)

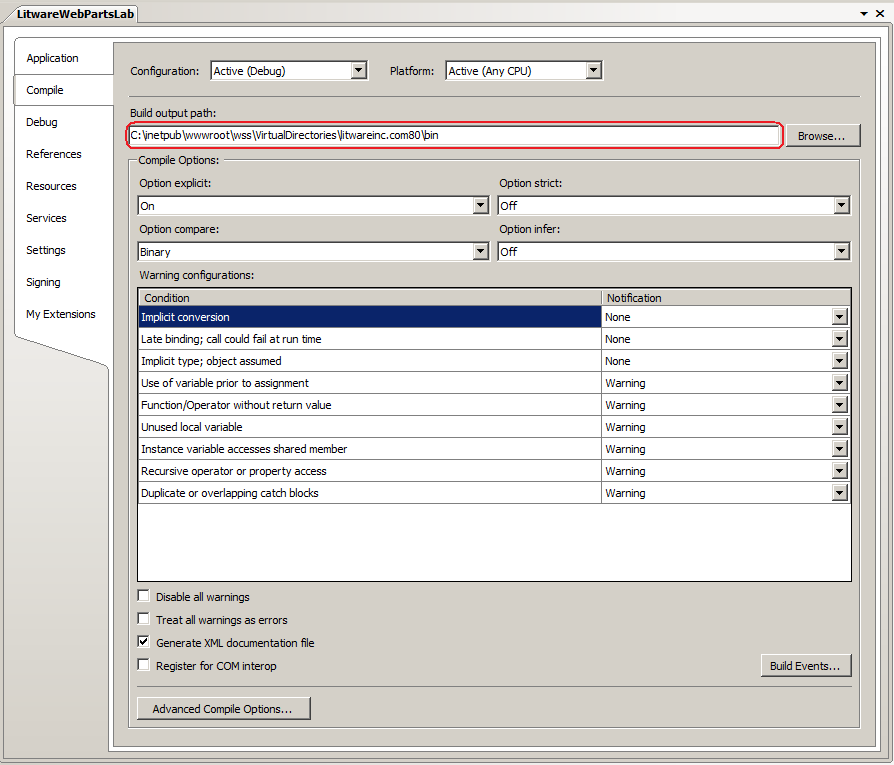
MyBase.RenderContents(writer)

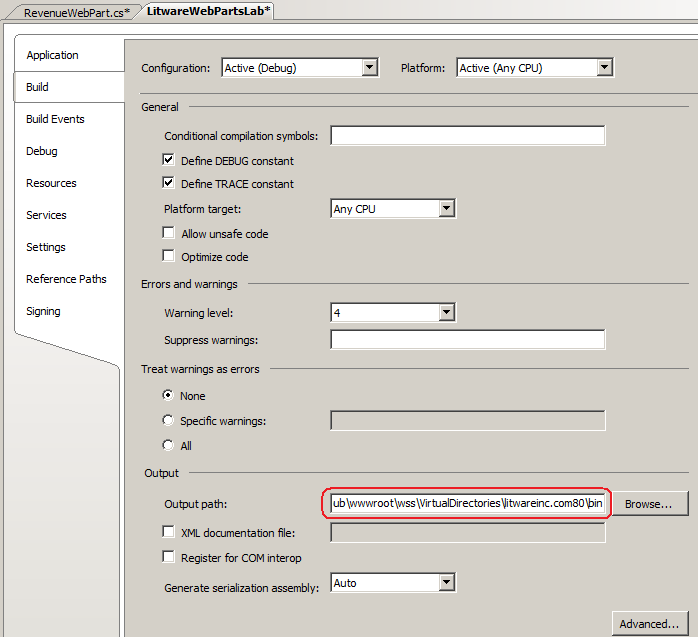
writer.Write("Hello, world")

End Sub

End Class

1. Remember that a Web Part DLL must be located in a place where the WSS runtime can find it. When you deploy a Web Part DLL, you have the option of putting it in either the local **\bin** directory or the **Global Assembly Cache (GAC)**. In this case, you are going to compile the assembly DLL output for the **LitwareWebPartsLab** project directly into the **\bin** directory of the default Web site.
   1. Modify the **LitwareWebPartsLab** project **Output path** to the following path by right clicking on your **LitwareWebPartsLab** project in **Solution Explorer** and select **Properties**
   2. In VB:   
      In your **Project Properties** window select **Compile\*** and then **Browse** and set the **Build output path** to **C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80\bin \**.



In C#:   
In your **Project Properties** window select **Build** and then **Browse** and set the **Output path** to **C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80\bin\**. 

1. Build the **LitwareWebPartsLab** project.
   1. After you have compiled the project, use the Windows Explorer to verify that the assembly DLL has been created in the bin subdirectory inside root directory of the Web Application running on port 80. Remember that the root directory is at the following path

C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80\

1. In order to be able to use Web Parts deployed into the local ...\bin directory you must modify the **<trust level=...>** tag in the **Web.config**
   1. By default, code access security permissions for the bin directory are low; only pure execution is allowed. Although our code in this exercise can currently run with the minimal trust level, in most cases you need to elevate these permissions to make your assembly run correctly, for example, if your Web Part requires access to the SharePoint object model.  In fact, we MUST modify this now as it will be required for this lab to run correctly before the end of exercise 3.
   2. **Note**: There are two ways to elevate permissions **(MAKE SURE YOU USE OPTION ii (EASY WAY))**:
      1. (**Recommended way** for deployment) Create a new trust policy file, and point your web.config file at the new file. This option is more complicated but it gives you a precise attribution of permissions for your Web Parts.  For more information about trust policy files, see [Microsoft Windows SharePoint Services and Code Access Security](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odc_sp2003_ta/html/sharepoint_wsscodeaccesssecurity.asp).
      2. (**Easy Way** for Development/Debugging) Raise the .Net trust level of the bin directory.
         1. Open the **Web.config** file in **C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80** in Visual Studio.
         2. In the **Web.config** file in the Web application root, there is a tag called **<trust>** with a default attribute of **level="WSS\_Minimal"**.   
            **Note**: this the easiest way to locate this tag is to do a find/replace function (**ctrl+h**) and search for **trust level**.
         3. Change this level to **WSS\_Medium**.  
            **Note**: While this option is simpler and preferred for testing and debugging, it grants arbitrary new permissions you might not need and is less secure than creating a new trust policy file.
         4. **Save** your changes to this file
2. In order to be able to use this newly created web part we must mark it as a "**SafeControl**" in the **Web.config** file for the Web Application.
   1. Edit the **Web.config** file from the **C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80** directory.
   2. Add a **SafeControl** setting to the **Web.config** file, in the preexisting **<SafeControls></SafeControls>** element, for your new Web Part that follows this format.

<SafeControl Assembly="[add assembly name]" Namespace="[add namespace]" TypeName="\*" />

* + 1. When you add the **SafeControl** element for your Web Part, the namespace should be written as **LitwareWebPartsLab** and the 4-part assembly format string should look like this.

LitwareWebPartsLab, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

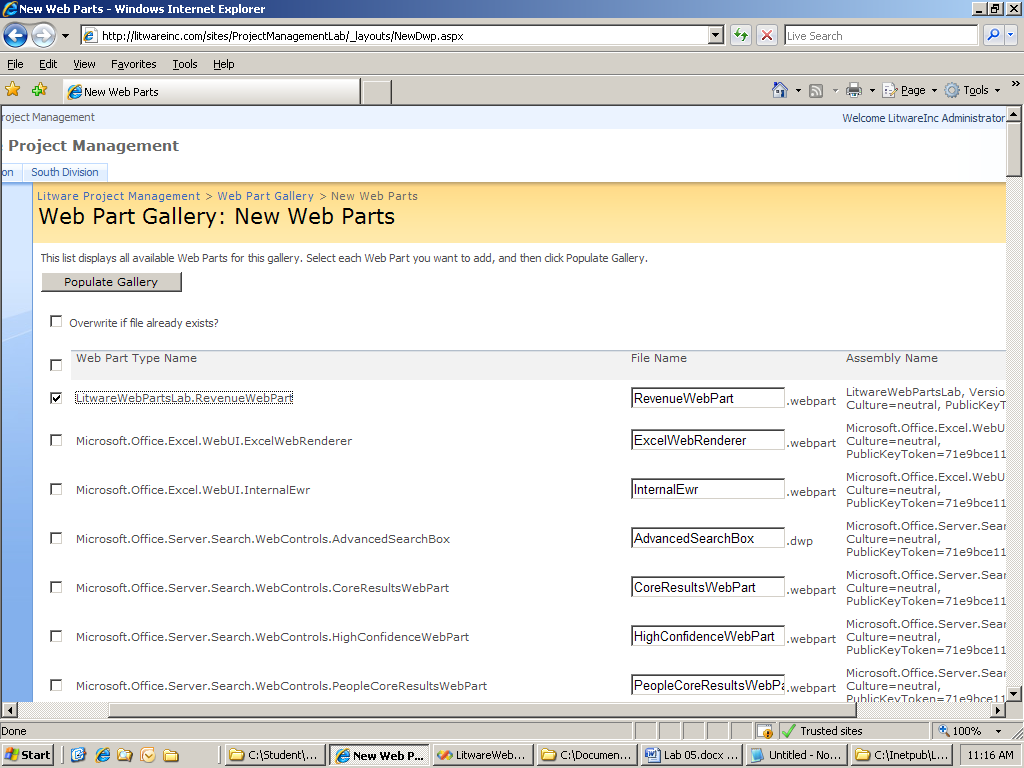
* + 1. **Note:** you can obtain the full 4-part assembly name using the **Reflector** utility that is in your \Student\Resources directory.
       1. In your **..\Student\Resources** directory open the **Reflector** utility (double click on **Reflector.exe**).
       2. In the Reflector Utility you will  need to examine the **LitwareWebPartsLab.dll**.  **File > Open...** (VB.NET or C#) Navigate to the  **C:\Inetpub\wwwroot\wss\VirtualDirectories\litwareinc.com80** directory open the **LitwareWebPartsLab.dll** file.
       3. Select the **LitwareWebPartsLab** 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 **SafeControl** setting.
       4. When finished your entry should look like this

<SafeControl Assembly="LitwareWebPartsLab, Version=1.0.0.0, Culture=neutral,

PublicKeyToken=null" Namespace="LitwareWebPartsLab" TypeName="\*" Safe="True" />

**Note**: this needs to be on **one line** even though it is spread across several  for display here  
**Extra Information**: The **Assembly** attribute comes from the dll name while the **Namespace** is the exact namespace where the class file exists. **TypeName** is the class name (\* means all classes in the namespace). **Safe** is whether you may use this on a SharePoint implementation.

* + 1. **Note**: As the **Web.config** is **case sensitive**, ensure that you type **LitwareWebPartsLab** in EXACTLY or it will not work.
    2. **Save** your changes.

1. Now that you have compiled the Web Part DLL into the **\bin** directory and configured it in the **SafeControl** list of the default Web Site, you can now add the **RevenueWebPart** Web Part to the Web Parts gallery of a top-level site.
   1. Go to the **Project Management** site  (**http://litwareinc.com/sites/ProjectManagementLab/**) and click the **Site Settings** command from the **Site Actions** menu.
2. Next we need to add the web part to our site collection so that we may utilize it on our Web Part Pages.
   1. On the **Site Settings** page, click on the **Web Parts** link in the **Galleries** section to navigate to the **Web Part Gallery** page.
   2. Click the **New** button on the toolbar to navigate to the page that allows you to add new Web Parts to the gallery.
   3. You should see **LitwareWebPartsLab.RevenueWebPart** as a selection. Click the check box to the left of the Web Part to select it, and then click the **Populate Gallery** button at the top of the page. 
   4. On the **Web Part Gallery** page, verify that the **RevenueWebPart.webpart** has been added to the list. It should be listed alphabetically, and also it should have the green **!NEW** marker next to it.
3. Now that you have added the **RevenueWebPart** Web Part to the gallery, you can add it to any page in the current site collection.
   1. Navigate to the home page of the **Project Management** site.
   2. On the **Litware Project Management** home page select **Edit Page** from the **Site Actions** menu.
   3. Click **Add a Web Part** in the **Web Part zone** on the right-hand side. Note:  the **RevenueWebPart** is located in the **Miscellaneous Web Part** Category; you will need to scroll down to locate it. Check the box next to it, and click **ADD**.
   4. Drag the newly added web part below the Litware Logo web part.
   5. When you are done, the Web Part should look like this:
   6. Be sure to click on the **Exit Edit Mode** hyperlink (near the top right corner of the page).

# Exercise 2: Add a persistent property to a Web Part

In this exercise you will add a property to your Web Part class and expose it to SharePoint.

1. Modify the **RevenueWebPart** so that it contains a persistent Boolean personalizable property named **ShowTextAsBold**. The property should give the user the ability to toggle it on and off within the task pane of the browser. When the user has the property disabled, the **RevenueWebPart** should render its output using a standard font as you have already done. When the property is enabled, the **RevenueWebPart** should render its output using a bold font with a larger size. If you're in the mood, you can also change the color of the font.
   1. Open the **RevenueWebPart** in Visual Studio 2008.
   2. First you need to create a protected Boolean variable called **\_ShowTextAsBold**.  Next we will create a public property called **ShowTextAsBold** utilizing the **\_ShowTextAsBold** for internal storage and add attributes to the property to expose this property in SharePoint.

C#

public class RevenueWebPart : WebPart {

protected bool \_ShowTextAsBold = true;

[ Personalizable(),

WebBrowsable(true),

WebDisplayName("Show Text As Bold"),

WebDescription("Enable to turn on bold font output") ]

public bool ShowTextAsBold {

get { return \_ShowTextAsBold; }

set { \_ShowTextAsBold = value; }

}

protected override void RenderContents(System.Web.UI.HtmlTextWriter writer)...

VB.Net

Public Class RevenueWebPart Inherits WebPart

Protected \_ShowTextAsBold As Boolean = True

< Personalizable(), \_

WebBrowsable(True), \_

WebDisplayName("Show Text As Bold"), \_

WebDescription("Enable to turn on bold font output")> \_

Public Property ShowTextAsBold() As Boolean

Get

Return \_ShowTextAsBold

End Get

Set

\_ShowTextAsBold = value

End Set

End Property

Protected Overrides Sub RenderContents(writer As System.Web.UI.HtmlTextWriter)...

* 1. Now we need to utilize this setting in our **RenderContents** method.  If the **ShowTextAsBold** property is set to "**true**" then we will Write out "**Hello, world**" in a **12pt, Bold, Red Font.**If "**false**" we will just write out "**Hello, world**" in a **plain** default font, as before.

C#

protected override void RenderContents(System.Web.UI.HtmlTextWriter writer) {

if (\_ShowTextAsBold) {

writer.AddStyleAttribute(HtmlTextWriterStyle.FontWeight, "Bold");

writer.AddStyleAttribute(HtmlTextWriterStyle.FontSize, "12pt");

writer.AddStyleAttribute(HtmlTextWriterStyle.Color, "Red");

writer.RenderBeginTag(HtmlTextWriterTag.Span);

writer.Write("Hello, world");

writer.RenderEndTag(); // </Span>

}

else {

writer.Write("Hello, world");

}

}

VB.Net

Protected Overrides Sub RenderContents(writer As System.Web.UI.HtmlTextWriter)

If \_ShowTextAsBold Then

writer.AddStyleAttribute(HtmlTextWriterStyle.FontWeight, "Bold")

writer.AddStyleAttribute(HtmlTextWriterStyle.FontSize, "12pt")

writer.AddStyleAttribute(HtmlTextWriterStyle.Color, "Red")

writer.RenderBeginTag(HtmlTextWriterTag.Span)

writer.Write("Hello, world")

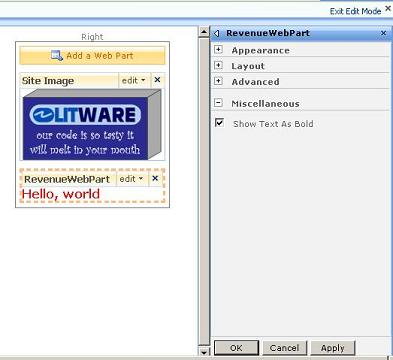
writer.RenderEndTag() ' </Span>

Else

writer.Write("Hello, world")

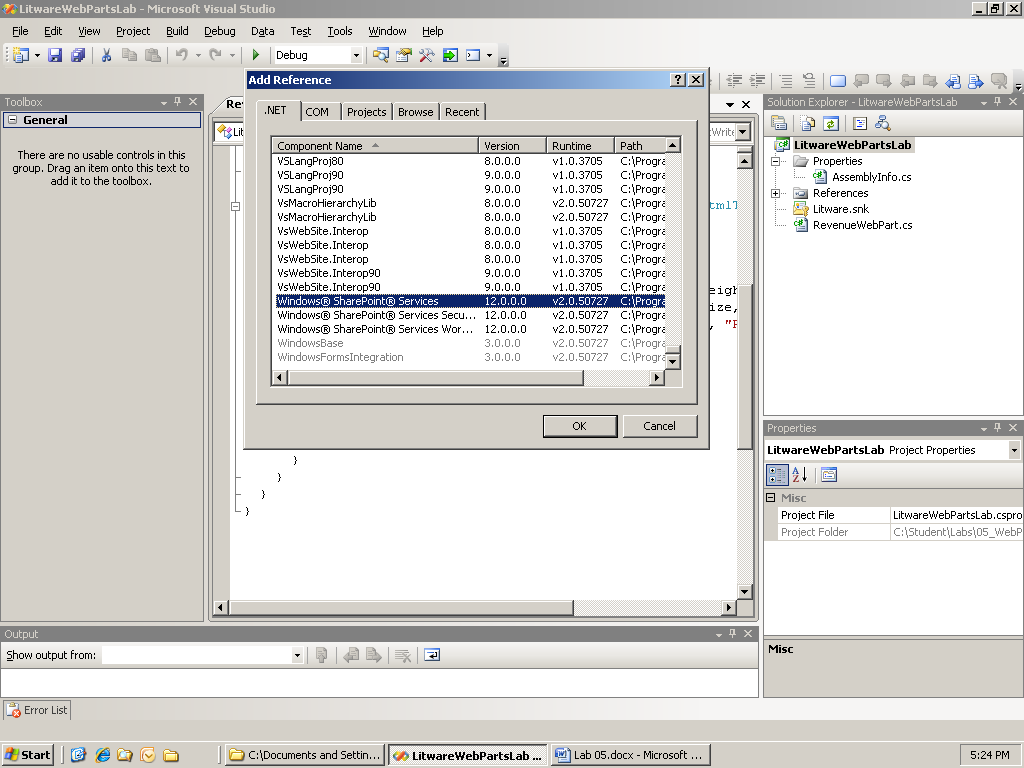
End If

End Sub

* 1. **Build** your project.
  2. Navigate to your **Litware Project Management** home page. (<http://litwareinc.com/sites/ProjectManagementLab/>)  
     **Note**: If you still have this page open, refresh it (**F5**) or the following steps will not work correctly.
  3. You should immediately notice that "**Hello, world**" is bigger, bolder, and redder than before.
  4. Click on the small drop down arrowin the upper right corner of your **RevenueWebPart** and select **Modify Shared Web Part**.
  5. Expand out the **Miscellaneous** section (i.e. click on the **+** next to the word **Miscellaneous**). The Web Part should appear as below in the browser.  
       
     
  6. Remove the check from the **Show Text As Bold** checkbox and click the **OK** button
  7. Your **RevenueWebPart** should now show "**Hello, world**" in regular black text.

# Exercise 3: Use the WSS object model to read data from a SharePoint list

In this exercise, you will modify the **RevenueWebPart** to show data from the **Projects** list. Up to this point, the Web Part has been written as a standard ASP.NET Web Part that can run in any ASP.NET site.  Now, it's time to add WSS-specific code to the DLL so it can access data within a WSS list.

1. Begin by adding a reference to **Microsoft.SharePoint.dll**.
   1. In your **Solution Explorer** right click on your **LitwareWebPartsLab** project and select **Add Reference...**
   2. Scroll down and select **Windows SharePoint Services** and click **OK.** 
2. Now you should bring the **Microsoft.SharePoint** namespace into scope in your **RevenueWebPart** class.

C#

using Microsoft.SharePoint;

VB.Net

Imports Microsoft.SharePoint

1. Next, inside the **RenderContents** method we will modify this code to access data within our **Projects** list.
   1. You will add this new code **before** the existing **IF ELSE** statement.
   2. Write the code to acquire a reference to the current site (i.e. **SPWeb** object) and then obtain a reference to the **Projects** list (i.e. **SPList** object).

C#

SPWeb ProjectManagementSite = SPContext.Current.Web;

SPList Projects = ProjectManagementSite.Lists["Projects"];

VB.Net

Dim ProjectManagementSite As SPWeb = SPContext.Current.Web

Dim Projects As SPList = ProjectManagementSite.Lists("Projects")

* 1. Next, enumerate through the list items one by one and add the contract amounts together to calculate a total.

C#

decimal TotalRevenue = 0;

foreach (SPListItem Project in Projects.Items) {

TotalRevenue += Convert.ToDecimal(Project["Contract Amount"]);

}

VB.Net

Dim TotalRevenue As Decimal = 0

Dim Project As SPListItem

For Each Project In Projects.Items

TotalRevenue += Convert.ToDecimal(Project("Contract Amount"))

Next Project

* 1. Finally, modify the existing **IF ELSE** statement to display the total summation of the contract amounts as the **total revenue**.  Be sure to format this total so that it has a US currency display.

C#

if (\_ShowTextAsBold) {

writer.AddStyleAttribute(HtmlTextWriterStyle.FontWeight, "Bold");

writer.AddStyleAttribute(HtmlTextWriterStyle.FontSize, "12pt");

writer.AddStyleAttribute(HtmlTextWriterStyle.Color, "Red");

writer.RenderBeginTag(HtmlTextWriterTag.Span);

writer.Write("Total Revenue: " + TotalRevenue.ToString("$#,###"));

writer.RenderEndTag(); // </Span>

}

else {

writer.Write("Total Revenue: " + TotalRevenue.ToString("$#,###"));

}

VB.Net

If \_ShowTextAsBold Then

writer.AddStyleAttribute(HtmlTextWriterStyle.FontWeight, "Bold")

writer.AddStyleAttribute(HtmlTextWriterStyle.FontSize, "12pt")

writer.AddStyleAttribute(HtmlTextWriterStyle.Color, "Red")

writer.RenderBeginTag(HtmlTextWriterTag.Span)

writer.Write(("Total Revenue: " + TotalRevenue.ToString("$#,###")))

writer.RenderEndTag() ' </Span>

Else

writer.Write(("Total Revenue: " + TotalRevenue.ToString("$#,###")))

End If

* 1. **Build** your project.
  2. Before you test your code, make sure you have updated the trust level in the **web.config** file to a setting value of **WSS\_Medium** as discussed in Exercise 1 Step 8. Since you have just begun to program against the WSS object model, this is the point in the lab where your code will begin to fail if you have not changed the trust level from the default setting of **WSS\_Minimal**. Note that you should always run an **IISRESET** operation when you have changed the trust level in the **web.config** file. Do this now by going to **START > RUN**, and then entering **iisreset** into the textbox.
  3. Navigate to your **Litware Project Management** home page (<http://litwareinc.com/sites/ProjectManagementLab/>)  
     **Note**: If you still have this page open refresh it (**F5**) or the following steps will not work correctly.
  4. You should immediately notice that "**Hello, world**" has been changed to "**Total Revenue: $1,645,000**".
  5. Click on the s**mall drop down arrow** in the upper right corner of your **RevenueWebPart** and select Modify Shared Web Part.
  6. Expand out the **Miscellaneous** section (i.e. click on the **+** next to the word **Miscellaneous**).
  7. Add a check in the **Show Text As Bold** checkbox and click the **OK** button
  8. Your **RevenueWebPart** should now display "**Total Revenue: $1,645,000**" in a bigger, bolder, and redder font than before.
  9. Your Web Part output should look like the screenshot below.  When you test your web part, verify that you can see your Web Part dynamically update its output when you add and/or modify projects in the Projects list.

