Lab 19: Building a Word 2007 Add-In

**Lab Time**: 20 Minutes

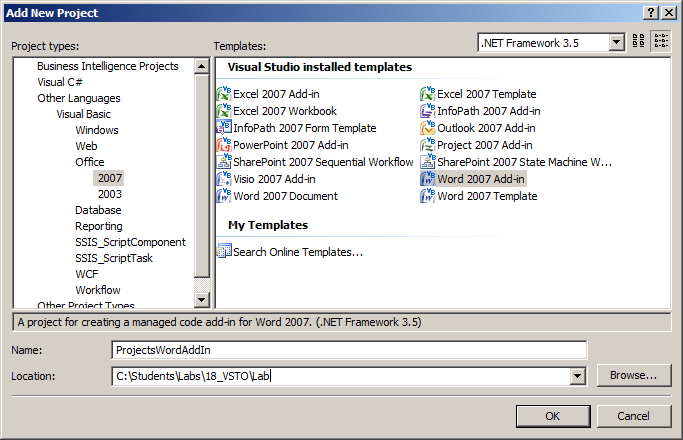
**Lab Directory**: C:\Student\Labs\19\_VSTO

**Lab Overview**: In this lab you will create a Word 2007 add-in that displays a custom ribbon with one toggle button. This button will show or hide a custom task pane where the user can select a project from a dropdown list. This dropdown is populated from the Projects list you created in the first lab on custom branding. When the user selects a project from the dropdown, project details will be displayed in the custom task pane. The user can then decide to insert the selected project in the word document.

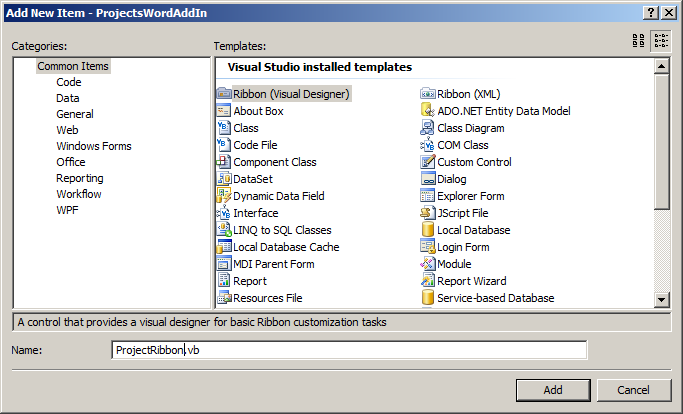
This lab contains directions to develop an Office Add-in using VB.NET.

# Exercise 1: Create the custom ribbon for the Word 2007 Add-In

1. Launch Visual Studio and create a new project of type Office 2007 and choose the Word 2007 template. Give the project the name **ProjectsWordAddIn** and save it in the **C:\Students\Labs\19\_VSTO\Lab**



1. Click the **OK** button to create the project.
2. Add a new item of type **Ribbon (Visual Designer)** and give it the name **ProjectRibbon.vb**. Click the **Add** button.



1. The ribbon designer opens with one tab and one group. Select the tab and set the following properties for the tab:
   1. Name: **LitwareTab**
   2. Label: **Litware**
2. Select the group and set the following properties for the group:
   1. Name: **LitwareGroup**
   2. Label: **Litware Projects**
3. Expand the **Toolbox** and expand the **Office Ribbon Controls** group. Drag a **ToggleButton** from the Toolbox into the group.
4. Select the button and set the following properties:
   1. Name: **ProjectsButton**
   2. ControlSize: **RibbonControlSizeLarge**
   3. Label: **Show Projects**
   4. OfficeImageId: **AddressBook**
5. In first instance we are only going to change the label of the button when the toggle button is clicked. Double-click the button to create an event handler for the **Click** event.

Private Sub MyToggleButton\_Click(ByVal sender As System.Object, \_

ByVal e As Microsoft.Office.Tools.Ribbon.RibbonControlEventArgs) \_

Handles MyToggleButton.Click

End Sub

1. When the button is clicked we want to change the label of the button. Place this code in a separate method because later on it will be called from outside the ribbon:

Public Sub RefreshToggleButton()

If (MyToggleButton.Checked) Then

MyToggleButton.Label = "Hide Products"

Else

MyToggleButton.Label = "Show Products"

End If

End Sub

1. Add the following lines of code in the Click event handler:

Private Sub MyToggleButton\_Click(ByVal sender As System.Object, \_

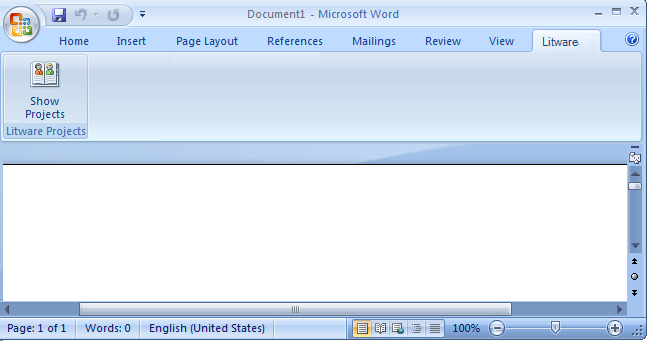
ByVal e As Microsoft.Office.Tools.Ribbon.RibbonControlEventArgs) \_

Handles MyToggleButton.Click

RefreshToggleButton()

End Sub

1. This is the easiest part of the project. Press F5 to build and run the project.
2. Word 2007 opens with a new document and your **Litware** tab should be visible.
3. Click the **Litware** tab and click the button. Notice that the label changes.



1. Close Word.

# Exercise 2: Create the custom task pane

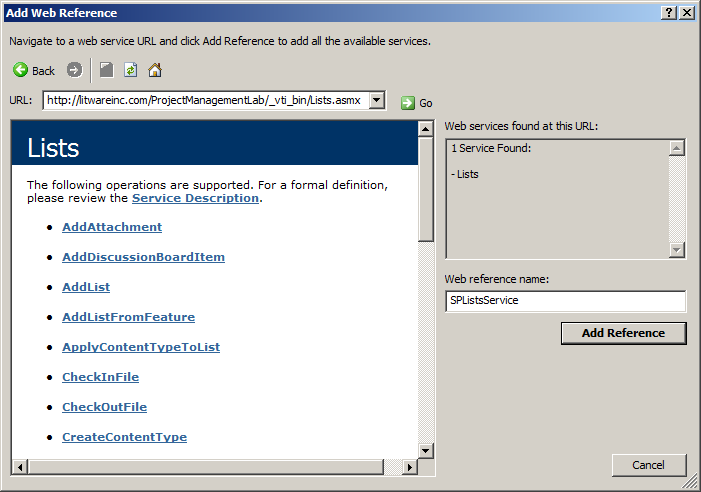
1. Custom task panes are developed in user controls. Return to Visual Studio and add a user control to the project. Give it the name **ProjectsPane**.
2. First you are going to design the user interface of the task pane.
   1. Set the **Width** property to **220** and the **Height** property to **400**. (The height doesn’t really matter because it will take the height of the Word instance.
   2. Add the following controls to the user interface so that it looks as follows:

|  |  |
| --- | --- |
| Name | Label |
| ProjectLabel | Select a project: |
| ProjectComboBox |  |
| NameLabel | Project name: |
| ClientLabel | Client: |
| ClientTextBox |  |
| AmountLabel | Contract amount: |
| AmountTextBox |  |
| BeginDateLabel | Begin date: |
| BeginDateTextBox |  |
| EndDateLabel | End date: |
| EndDateTextBox |  |
| ContractCheckBox | Contract signed |
| InsertButton | << Insert |

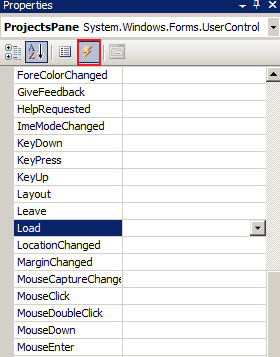
1. When the task pane is loaded, you want the dropdown to be populated with the projects available in the **Projects** list that you created on the **http://litwareinc.com/sites/ProjectManagementLab**. This data can be retrieved using the **GetListItems** method of the SharePoint **Lists.asmx** web service.
2. First add a reference to the Microsoft.SharePoint.dll. You can find this dll as **Windows SharePoint Services** assembly in the .**NET** tab of the **Add Reference** dialog.
3. You also need to add a service reference to the **Lists.asmx** web service.
   1. In Solution Explorer right click the project and choose **Add Service Reference**.
   2. Click the **Advanced** button at the bottom of the dialog.
   3. Click the **Add Web Reference** button at the bottom of the **Service Reference Settings** dialog.
   4. Enter the URL to the Project Management SharePoint site, completed with the path to the **Lists.asmx** web service. The SharePoint web services are accessible from the **\_vti\_bin** directory:

**http://litwareinc.com/ProjectManagementLab/\_vti\_bin/Lists.asmx**

* 1. Click the **Go** button to load the web service. The URL is entered correctly when the methods of the web service appear.
  2. Enter **SPListsService** as name for the web reference.



1. To avoid having to write all the plumbing code to retrieve the data, you can find a helper class in the **Starter Files** of this lab with the name **ProjectsHelper.vb**. Copy it to your project directory and include it in the project. Take your time to inspect the code. The helper file contains a class definition for the Project object and a helper class. The project data is downloaded using the **GetListItems** method of the web service. The result is an xml string that is parsed to a collection of project object using LINQ.
2. Return to the **ProjectsPane** in Designer mode. The projects dropdown from this custom task pane needs to be populated when the task pane loads.
   1. Go to the **Properties** box of the task pane and click the **Events** button.
   2. Locate the Load event and double-click it to generate the appropriate event handler.



* 1. First declare 2 class level variables: one for the list of projects and one for the selected project:

Dim projectList As List(Of Project)

Dim selectedProject As Project

* 1. Place your cursor in the **Load** event handler and add code to instantiate the helper class and retrieve the list of projects from the SharePoint web site:

Dim helper As New ProjectsHelper(“Projects”)

projectList = helper.GetProjects()

ProjectComboBox.DataSource = projectList

1. When a user selects a project from the dropdown the details should be displayed in the **TextBox** controls on the pane.
   1. Return to the **ProjectsPane** in Designer mode and double click the dropdown to generate the **SelectedIndexChanged** event handler.
   2. Add code to retrieve the selected project and populate the TextBox controls:

selectedProject = projectList(ProjectComboBox.SelectedIndex)

NameTextBox.Text = selectedProject.Name

ClientTextBox.Text = selectedProject.Client

AmountTextBox.Text = selectedProject.ContractAmount.ToString( \_

System.Globalization.CultureInfo.InvariantCulture) + "$"

BeginDateTextBox.Text = selectedProject.ProjectBeginDate.ToString()

EndDateTextBox.Text = selectedProject.ProjectEndDate.ToString()

ContractCheckBox.Checked = selectedProject.ContractSigned

1. When a user clicks the **Insert** button the selected project name should be filled out in the word document.
   1. Return to the **ProjectsPane** in Designer mode and double click the button to generate the event handler for the **Click** event.
   2. Add code to

Globals.ThisAddIn.Application.Selection.Range.Text = \_

String.Format("The project {0} is signed for an amount of {1}, “ + \_

“starting {2} and ending {3}.", \_

selectedProject.Name, AmountTextBox.Text, \_

BeginDateTextBox.Text, EndDateTextBox.Text)

1. This finishes off the work to the custom task pane. Now you need to establish the interaction between the custom ribbon button and the custom task pane. In order to achieve this you have to change the code behind the ribbon button.
   1. Open the **ThisAddIn.vb** class.
   2. Declare a class level variable for the task pane:

Friend projectsTaskPane As Microsoft.Office.Tools.CustomTaskPane

* 1. Locate the **Startup** event handler for the add-in and write code to add the custom task pane to the collection of custom task panes:

projectsTaskPane = Me.CustomTaskPanes.Add(New ProjectsPane(), "Projects Info")

* 1. Open the **MyRibbon.vb** in code view and add the following to the Click event of the toggle button in order to:

Globals.ThisAddIn.projectsTaskPane.Visible = MyToggleButton.Checked

1. This concludes the work on the Word AddIn. Press F5 to build and run the project. A Word document should open.
2. Click the **Litware** tab and click the button. You custom task pane appears.
3. Select a project and verify the details displayed in the task pane.
4. Click the **Insert** button and notice the sentence in the word document, containing information on the selected project.
5. If you click the ribbon button again, the task pane will disappear.
6. Close Word.

