**Revit API UI Labs**

**Lab 1 – Ribbon**

May, 2011 by A. Nagy

**<VB.NET>**VB.NET Version**</VB.NET>**

**Objective:** In this lab, we will learn how to create our own Ribbon tab and panels then populate them with controls. We’ll learn how to:

* Create a new Ribbon tab and panel
* Create various Ribbon controls

The following is the breakdown of step by step instructions in this lab:

1. Create a new External Application
2. Create new Ribbon tab and panel
3. Create push button
4. Create split button
5. Create combo box
6. Create other controls
7. Summary
8. **Define A New External Application**

We’ll create a new project and add an external application to it.

* 1. You can follow the same steps as in Revit API Intro Labs >> Lab1 >> 5, but this time name the Class Library project UiVb (or UiCs depending on the programming language you chose)
  2. Rename the Class1.vb (or cs) file
* File name: **1\_Ribbon.vb (or .cs)**
* Application class name: **UIRibbon**

**Required References:**

Additional references needed for this lab are:

* PresentationCore – used for handling bitmap images
* WindowsBase – used for handling bitmap images
* IntroCs/Vb – we will be using commands defined in the Intro labs

**Required Namespaces:**

Namespaces needed for this lab are:

* Autodesk.Revit.DB
* Autodesk.Revit.UI
* Autodesk.Revit.ApplicationServices
* Autodesk.Revit.Attributes
* System.Collections.Generic
* System.Diagnostics – used for debug
* System.IO – used for reading folders
* System.Windows.Media.Imaging – used for bitmap images
  1. Declare some variables that will contain certain paths that will be useful e.g. to retrieve images for our ribbon controls:

**<VB.NET>**  
 ''' <summary>

''' This is both the assembly name and the namespace

''' of the external command provider.

''' </summary>

Private Const \_dllExtension As String = ".dll"

Private Const \_introLabName As String = "IntroVb"

Private Const \_uiLabName As String = "UiVb"

''' <summary>

''' Name of subdirectory containing images.

''' </summary>

Private Const \_imageFolderName As String = "Images"

''' <summary>

''' Location of images for icons.

''' </summary>

Private \_imageFolder As String

''' <summary>

''' Location of managed dll where we have defined the commands.

''' </summary>

Private \_introLabPath As String

**</VB.NET>**

* 1. Let’s also add a couple of utility functions, which will help us find our image folder or create a BitmapImage object from a specific image file

**<VB.NET>**  
 ''' <summary>

''' Starting at the given directory, search upwards for

''' a subdirectory with the given target name located

''' in some parent directory.

''' </summary>

''' <param name="path">Starting directory, e.g. GetDirectoryName( GetExecutingAssembly().Location ).</param>

''' <param name="target">Target subdirectory name, e.g. "Images".</param>

''' <returns>The full path of the target directory if found, else null.</returns>

Private Function FindFolderInParents( \_

ByVal path As String, ByVal target As String) As String

Debug.Assert(Directory.Exists(path), \_

"expected an existing directory to start search in")

Do

Dim s As String = System.IO.Path.Combine(path, target)

If Directory.Exists(s) Then

Return s

End If

path = System.IO.Path.GetDirectoryName(path)

Loop While (path IsNot Nothing)

Return Nothing

End Function

''' <summary>

''' Load a new icon bitmap from our image folder.

''' </summary>

Function NewBitmapImage(ByVal imageName As String) As BitmapImage

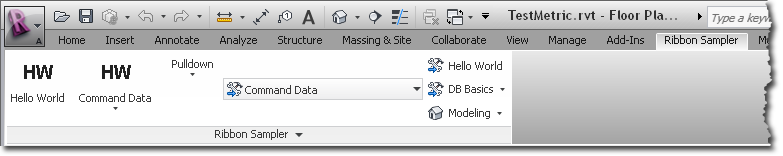
Return New BitmapImage(New Uri(Path.Combine(Me.\_imageFolder, imageName)))

End Function

**</VB.NET>**

We’ll also need to copy the ‘Images’ folder from the ‘Labs\2\_Revit\_UI\_API’ folder to our project’s folder.

1. **Create new Ribbon tab and panel**



Since Revit 2012 we can also add our own Ribbon tab using CreateRibbonTab(). Depending on which override of CreateRibbonPanel() we are using, the newly created panel will either be added to the Ribbon tab we specify or to the default ‘Add-Ins’ tab.

Let’s create a function that will create our Ribbon tab and add a panel to it.

**<VB.NET>**   
 Sub AddRibbonSampler(ByVal app As UIControlledApplication)

' (1) create a ribbon tab and panel

app.CreateRibbonTab("Ribbon Sampler")

Dim panel As RibbonPanel = \_

app.CreateRibbonPanel("Ribbon Sampler", "Ribbon Sampler")

End Sub

**</VB.NET>**

Before going any further let’s initialize the helper variables in the class inside the OnStartup() function.

**<VB.NET>**

' External application directory:

Dim dir As String = Path.GetDirectoryName( \_

System.Reflection.Assembly.GetExecutingAssembly.Location)

\_introLabPath = Path.Combine(dir, \_introLabName + \_dllExtension)

' External command path:

If Not File.Exists(\_introLabPath) Then

TaskDialog.Show( \_

"UIRibbon", "External command assembly not found: " + \_introLabPath)

Return Result.Failed

End If

' Image path:

\_imageFolder = FindFolderInParents(dir, \_imageFolderName)

If \_imageFolder Is Nothing Or Not Directory.Exists(\_imageFolder) Then

TaskDialog.Show( \_

"UIRibbon", \_

String.Format( \_

"No image folder named '{0}' found in the parent directories of '{1}.", \_

\_imageFolderName, dir))

Return Result.Failed

End If

**</VB.NET>**

1. **Create push button**

Now let’s add some controls to the Ribbon panel we’ve just created. First we’ll create a push button. Controls can be created through the various [Control]Data classes which will specify the properties of the control we are creating and the actual [Control] will be created once we called AddItem() – e.g. you use PushButtonData when creating a PushButton.   
In case of the various buttons we need to specify the assembly path and full class name of the ExternalCommand that should be invoked when the button is clicked.   
Note: If you specify an external command in the \*.addin manifest file then the command will be available through the ‘Add-Ins tab >> External >> External Tools’. By creating various Ribbon controls you can make your commands available through those controls as well. If the latter is enough for you then no need to declare your external commands in the \*.addin manifest file.

**<VB.NET>**

Sub AddPushButton(ByVal panel As RibbonPanel)

' Set the information about the command we will be assigning

' to the button

Dim pushButtonDataHello As \_

New PushButtonData("PushButtonHello", "Hello World", \_

\_introLabPath, \_introLabName + ".HelloWorld")

' Add a button to the panel

Dim pushButtonHello As PushButton = panel.AddItem(pushButtonDataHello)

' Add an icon

' Make sure you reference WindowsBase and PresentationCore,

' and import System.Windows.Media.Imaging namespace.

pushButtonHello.LargeImage = NewBitmapImage("ImgHelloWorld.png")

' Add a tooltip

pushButtonHello.ToolTip = "simple push button"

End Sub

**</VB.NET>**

1. **Create split button**

A split button is basically just a button that groups push buttons together. So you’ll need to use PushButtonData’s and add them to a SplitButtonData object to create the control. Here as well, each button will be hooked up to a specific external command.

**<VB.NET>**

Sub AddSplitButton(ByVal panel As RibbonPanel)

' Create three push buttons for split button drop down

' #1

Dim pushButtonData1 As \_

New PushButtonData("SplitCommandData", "Command Data", \_

\_introLabPath, \_introLabName + ".CommandData")

pushButtonData1.LargeImage = NewBitmapImage("ImgHelloWorld.png")

' #2

Dim pushButtonData2 As \_

New PushButtonData("SplitDbElement", "DB Element", \_

\_introLabPath, \_introLabName + ".DbElement")

pushButtonData2.LargeImage = NewBitmapImage("ImgHelloWorld.png")

' #3

Dim pushButtonData3 As \_

New PushButtonData("SplitElementFiltering", "ElementFiltering", \_

\_introLabPath, \_introLabName + ".ElementFiltering")

pushButtonData3.LargeImage = NewBitmapImage("ImgHelloWorld.png")

' Make a split button now

Dim splitBtnData As New SplitButtonData("SplitButton", "Split Button")

Dim splitBtn As SplitButton = panel.AddItem(splitBtnData)

splitBtn.AddPushButton(pushButtonData1)

splitBtn.AddPushButton(pushButtonData2)

splitBtn.AddPushButton(pushButtonData3)

End Sub

**</VB.NET>**

1. **Create combo box**

In case of e.g. the combo box, the functionality comes from handling selection events instead of hooking up each combo item to a specific external command. Once we created the combo box, we need to subscribe to the CurrentChanged event to handle selection changes.

**<VB.NET>**

Sub AddComboBox(ByVal panel As RibbonPanel)

' Create five combo box members with two groups

' #1

Dim comboBoxMemberData1 As \_

New ComboBoxMemberData("ComboCommandData", "Command Data")

comboBoxMemberData1.Image = NewBitmapImage("Basics.ico")

comboBoxMemberData1.GroupName = "DB Basics"

' #2

Dim comboBoxMemberData2 As \_

New ComboBoxMemberData("ComboDbElement", "DB Element")

comboBoxMemberData2.Image = NewBitmapImage("Basics.ico")

comboBoxMemberData2.GroupName = "DB Basics"

' #3

Dim comboBoxMemberData3 As \_

New ComboBoxMemberData("ComboElementFiltering", "Filtering")

comboBoxMemberData3.Image = NewBitmapImage("Basics.ico")

comboBoxMemberData3.GroupName = "DB Basics"

' #4

Dim comboBoxMemberData4 As \_

New ComboBoxMemberData("ComboElementModification", "Modify")

comboBoxMemberData4.Image = NewBitmapImage("Basics.ico")

comboBoxMemberData4.GroupName = "Modeling"

' #5

Dim comboBoxMemberData5 As \_

New ComboBoxMemberData("ComboModelCreation", "Create")

comboBoxMemberData5.Image = NewBitmapImage("Basics.ico")

comboBoxMemberData5.GroupName = "Modeling"

' Make a combo box now

Dim comboBxData As New ComboBoxData("ComboBox")

Dim comboBx As ComboBox = panel.AddItem(comboBxData)

comboBx.ToolTip = "Select an Option"

comboBx.LongDescription = "select a command you want to run"

comboBx.AddItem(comboBoxMemberData1)

comboBx.AddItem(comboBoxMemberData2)

comboBx.AddItem(comboBoxMemberData3)

comboBx.AddItem(comboBoxMemberData4)

comboBx.AddItem(comboBoxMemberData5)

AddHandler comboBx.CurrentChanged, New EventHandler(Of \_

ComboBoxCurrentChangedEventArgs)(AddressOf comboBx\_CurrentChanged)

End Sub

''' <summary>

''' Event handler for the above combo box

''' </summary>

Sub comboBx\_CurrentChanged(ByVal sender As Object, \_

ByVal e As ComboBoxCurrentChangedEventArgs)

' Cast sender as TextBox to retrieve text value

Dim combodata As ComboBox = TryCast(sender, ComboBox)

Dim member As ComboBoxMember = combodata.Current

TaskDialog.Show("Combobox Selection", "Your new selection: " + \_

member.ItemText)

End Sub

**</VB.NET>**

1. **Create other controls**

There are other controls as well that you could use. Look through the Autodesk.Revit.UI namesapce to find them and try to create some of them yourself.

Now call all the control creation functions from the AddRibbonSampler() function.

**</VB.NET>**

Sub AddRibbonSampler(ByVal app As UIControlledApplication)

' (1) create a ribbon tab and panel

app.CreateRibbonTab("Ribbon Sampler")

Dim panel As RibbonPanel = \_

app.CreateRibbonPanel("Ribbon Sampler", "Ribbon Sampler")

AddPushButton(panel)

AddSplitButton(panel)

AddComboBox(panel)

End Sub

**</VB.NET>**

Then call AddRibbonSampler() at the end of OnStartup()

You could also place some of the controls on the slide out part of the panel – this part is only visible if the user clicks on the bottom strip of the Ribbon panel.

All controls added to the panel after a call to panel.AddSlideOut() will be added to the slide out part of the panel.

1. **Summary**

In this lab, we learned how to create your own Ribbon tab and panels and will them with controls. We have learned how to:

* Create a new Ribbon tab and panel
* Create various Ribbon controls

Autodesk Developer Network