MS Office Power Point Plugin

Sometime we are very frustrated with repetitive work, especially when we are working on office product. In this conditions it's desirable to integrate added functionality directly into the environment, we need to automate our work. Add-ins gives power to do this type of work.

In this example we are focusing on small essential chucks for creating a PowerPoint Add-in. like setting up project, reading slide data etc.

Prerequisites:

- Visual Studio (I am working on VS-2013).
- MS Power Point (I am using MS-Power Point 2013).
- Working knowledge of C#.

Index (What I am going to cover):

- Setting up your power point plugin project in visual studio.
- Showing your plugin in power point plugin tab.
- Changing the image of your icon.
- Using the resource file.
- Invoking your methods.
- Displaying the result on power point.
- Reading the slide data.
- Writing a power point file.

How to set up a power point plugin project in visual studio?

Follow the below steps

File => New Project => Templates => Visual C# => Office/Share Point => PowerPoint 2013 Add-In (check screenshot in git repo) => Project Setup

How to show your plugin in power point plugin tab?

For this you need to create a ribbon.

Right Click on your project (solution pane)=> Select add new item => Ribbon (XML) (check screenshot in git repo) => ribbon

After adding this, visual studio will open a <ribbon name>.cs file and when you read the comment you will get to know what to do next.

Example

TODO: Follow these steps to enable the Ribbon (XML) item:

1: Copy the following code block into the "ThisAddin", "ThisWorkbook", or "ThisDocument" class.

```
Protected override Microsoft.Office.Core.IRibbonExtensibility
CreateRibbonExtensibilityObject()
{
    return new Ribbon1();
}
```

2. Create callback methods in the "Ribbon Callbacks" region of this class to handle user actions, such as clicking a button.

Note: If you have exported this Ribbon from the Ribbon designer, move your code from the event handlers to the callback methods and modify the code to work with the Ribbon extensibility (RibbonX) programming model.

3. Assign attributes to the control tags in the Ribbon XML file to identify the appropriate callback methods in your code.

1. Start with copying the code from the first comment and pasting it into ThisAddin class

2. Next we will add a button in the ribbon to trigger an event from PowerPoint

Paste this code into your xml or you can write your own code: D

```
<?xml version="1.0" encoding="UTF-8"?>
<customUI
xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="Ribbon_Load">
 <ribbon>
  <tabs>
   <tab idMso="TabAddIns">
    <group id="Mygroup" label="TestAdd-in" >
     <button id="AddAnimationButton"
      label="Check"
      size="large"
      getImage="GetImage"
      supertip="Check"
      enabled="true"
      onAction="AddAnimationButtonClick"/>
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

Now run the program. We see one ADD-IN tab and inside this tab a button named "check".

Here the getImage tag calls a function "GetImage" to get the icon image.

Changing the image of your icon and using the resource file

We have an inbuilt resource file. This file can save any object in XML format.

So first, we need to insert the icon file in the resource file.

1. Type **resource** in your solution explorer search box.
2. Now we see the Resources.resx file. Open it.
3. Click on Add Resources => add existing file and add one image file.
4. Now we have to implement the "GetImage" method.

This is a ribbon function, so we have to implement this into Ribbon.cs file.

This function takes one argument which is Microsoft.Office.Core.IRibbonControl

And in the end we have to return a bitmap file.

Example:

```
public Bitmap GetImage(Office.IRibbonControl control)
{
  return new Bitmap(PluginPowerPoint.Properties.Resources.icon);
 }
```

5. Now run the program and see the result.

   It’s not working. That's because we didn’t implement an action on this button.

6. We can see “ onAction="AddAnimationButtonClick"/> “ line on ribbon.xml file. This is the trigger point and we have to implement this method in our ribbon.cs file.

   Let's say we want to display a message box saying “Hello world”.

7. Copy and paste the following method in your ribbion.cs file. This function takes IRibbonControl interface object as an argument to gain control.

```
public void AddAnimationButtonClick(Office.IRibbonControl control)
    {
        MessageBox.Show("Hello World");
    }
```

Now run the code ..

Success!

Deep Dive Into code

Till now we have only set up our project. Now the actual work starts here.

What we need [technically]

1. Slide object of every slide: for handling operation on slides like reading data
2. Handle of every object in the every slide

For the slide objects, you need:

"Microsoft.Office.Interop.PowerPoint.Slides" object

How can you get this? It's very simple and straight:

"Globals.slideQAddIn.Application.ActivePresentation.Slides" (in power point plugin only)

This syntax will give you array of slide objects.

Let's tinker with it.

Previously we have shown “Hello World” on button click. Can we display the number of slides instead?

If syntax above gives us an array then we can show the array count. So replace your code with this:

```
public void AddAnimationButtonClick(Office.IRibbonControl control)
{
    MessageBox.Show(Globals.ThisAddIn.Application.ActivePresentation.Slides.Count.ToString();
}
```

Now run the application

Now we need the handle of every object in each slide.

For this you have to extract shapes property from the slide object, and iterate:

```
foreach (PPT.Shape shape in slide.Shapes)
    {
    }
```

Where PPT is “using PPT = Microsoft.Office.Interop.PowerPoint;”

So now you have the shape objects. But what if you have a group shape (group of many shapes)? The code above will only give you a group object. Also, what if you have placeholder in your slide?

If you have group object you can easily inherit objects with the help of a single loop. But if you have a placeholder you cannot. First you need to know the type of object inside the placeholder and then you can fetch the object.

So what should we do?

1. We have to use recursion for the extraction of every single shape in the slide.
2. We should know how to compare type of object.

We won't be discussing recursion here. Let's start with point number 2.

To get the type of shape you need to visit https://msdn.microsoft.com/en-us/library/office/ff860759.aspx page. And you have to search for enum MsoTriState.

First we will see how to extract shape from group object

There are two ways to do this:

1. Ungroup all the shapes.
2. With simple iteration of the shape object property.

First we have to judge whether it's a group item or not

You can do this with MsoShapeType as I already mentioned in the link

Example:

```
if (shape.Type == MsoShapeType.msoGroup)
        {
        }
```

"Type" is the property of shape object and "MsoShapeType" is an enum.

After this condition you have to iterate all the shapes contained by shape object with the help of “GroupItems” property.

Example

```
foreach (PPT.Shape myShape in shape.GroupItems)
{
}
```

Now come to the second shape

Firstly you have to judge the type of object/shape of the placeholder.

You need to use the same approach as mentioned earlier.

Here is a function which will cover most of the shapes.

```
private void extractInfoFromShape(PPT.Shape shape)
    {
        if (shape.Type == MsoShapeType.msoGroup)
        {
        }
        else if (shape.Type == MsoShapeType.msoSmartArt  )
        {

        }
        else if (shape.Type == MsoShapeType.msoTable)
        {

        }
        else if (shape.Type == MsoShapeType.msoPlaceholder)
        {
         }
        if (shape.HasTextFrame == MsoTriState.msoTrue)
        {
        }
    }
```

Ok now you can check the type of shape now what right now I don't know how to extract shape directly so you have to hit and trial for every shape.

Like :

```
private void GetDataFromPlaceHolder(PPT.Shape shape)
  {
      try
      {
          GetDataFromSmartArt(shape);
      }
      catch
      {
          try
          {
              GetDataFromTable(shape);
          }
          catch
          {
              try
              {
                  GetDataFromGroupItem(shape);
              }
              catch
              {


              }
          }
      }
  }
```

To check what type of shape the object contains, you can use “Has” property.

For example, if you want to check TextFrame in a shape:

```
if (shape.HasTextFrame == MsoTriState.msoTrue)
        {
            extractInfo(shape);
        }
```

Let’s see a live example:

In this example we will show the total number of different shapes in a PowerPoint file.

What we need for this

1. Every slide object
2. Every shape object

First try yourself ..

Ok then hope you did it.

expand your project

Note: I am not going to maintain code quality here.

So add a class here named counter (you can change the name)

Create a method which takes Slides interface object as an argument.

Now follow the steps:

3. Iterate every slide
4. Use recursive method to extract every single shape object
5. Check for three shapes SmartArt, group, Placeholder
6. If it's SmartArt count the nodes
7. If it's group or placeholder extract all the shapes
8. Again start with step 3

Here is the code for this

class counter

```
    {

        public int detectSmartArt(Slides slides)
        {
            int count = 0;
            GetCount(slides, ref count);
            return count;
        }

         void GetCount(Slides slides,ref int count)
        {
            foreach (Slide slide in slides)
            {
                extractSlideInfo( slide,ref count);
            }

        }

        private void extractSlideInfo(Slide slide,ref int count)
        {
            foreach (PPT.Shape shape in slide.Shapes)
                extractInfoFromShape(shape,ref count);

        }
```

And ribbon.cs class replace this method

```
  public void AddAnimationButtonClick(Office.IRibbonControl control)
      {
          counter contobj = new counter();
          MessageBox.Show( contobj.detectSmartArt(
Globals.ThisAddIn.Application.ActivePresentation.Slides).ToString());
      }
```

So what's your count? ;)

What if you want to show your output in better way? "Displaying the result on power point"

We can use a pane for this.

Pane is used to display information and also for user interaction.

Here's how can we use a pane.

1. Add a wpf user control in your project
    a. Add a label in this control
2. In your xaml.cs file create a public label object
    a. Like : public static Label counter ;
3. Initialize this object with your UI label object
4. Bind your UI label with this object.

Code :

XAML CODE

```
<UserControl x:Class="PluginPowerPoint.Display"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid Background="#FFFDFDFD">
        <Label x:Name="count"  HorizontalAlignment="Left" Margin="55,74,0,0" VerticalAlignment="Top" Height="142" Width="171" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Content="{Binding counter}"/>
    </Grid>
</UserControl>
```

C# code XAML.CS file

```
  public static Label counter ;
        public Display()
        {
            InitializeComponent();
            counter = count;
        }
```

5. Now add WinForm User control in your project
6. Build your project.
7. You can see the wpf control above in your toolbox. Dock it into your winform controller.

8. Now add two private properties in your thisaddin class
    a. private BackPane panebase;
    b. private static Microsoft.Office.Tools.CustomTaskPane TaskPaneObj;
9. For controlling task pane from the outside you have to make a public property of the taskpane object.

```
public Microsoft.Office.Tools.CustomTaskPane TaskPane
{
    get
    {
        return TaskPaneObj;
    }
}
```

10. Now you have to initialize this pane at the time of plugin startup. There is a method named ThisAddIn_Startup in thisaddin class.
11. Initialize all the objects in ThisAddIn_Startup method
    a. panebase = new BackPane();
    b. TaskPaneObj = this.CustomTaskPanes.Add(panebase, "SmartArt Count");
    c. TaskPaneObj.Visible = false;
12. As you can see at time of taskpane object initialization you have to pass one winform control object and name of the task pane.
13. Now come to ribbon.cs file.
14. Go to cbutton_click_event method (“AddAnimationButtonClick”)
15. Use wpf public label and set the value of the label.
16. Update the layout.
17. And set the visibility of the taskpane to true with the help of global object.

CODE:

```
counter contobj = new counter();
    int count= contobj.detectSmartArt(
Globals.ThisAddIn.Application.ActivePresentation.Slides);
```

```
Display.counter.Content= count;
Display.counter.UpdateLayout();
Globals.ThisAddIn.TaskPane.Visible = true;
```

Now run your program..

Now write in the slide

It’s the simplest part of this tutorial.

Lets take an example to add a table in every slide.

We need:

1. Slide object
2. Position of shape

You already have the slide object

Position of the shape is defined as (Left,Top), meaning distance of left-top corner of the shape from left-top corner of the slide.

(left,top) of slide

So add a class in your project and create a method which takes the slide objects.

Iterate all the slides and add this syntax :

```
Microsoft.Office.Interop.PowerPoint.Shape pptShape = slide.Shapes.AddTable(i, i);
```

Its done.

Code:

```
class AddTable
{
    public void Add(Slides slides)
    {

        GetCount(slides);

    }

    void GetCount(Slides slides)
    {
        int i=1;
        foreach (Slide slide in slides)
        {

            Microsoft.Office.Interop.PowerPoint.Shape pptShape =
slide.Shapes.AddTable(i, i);
            i++;
            if(i==7)
            {
                i = 1;
            }

        }

    }
}
```