

Lab 1 – リボン

May, 2011 by A. Nagy
Updated by DevTech AEC WG
Last modified: 8/4/2015

<C#>C#バージョン</C#>

目的:この実習では、リボン タブとリボン パネルを作成して、コントロールをそれらに配置する方法を学習します。
学習する項目は次のとおりです。

- 新しいリボン タブおよびパネルを作成する
- 様々なリボン コントロールを作成する

この実習の実装と確認の手順は、下記のとおりです:

1. 新しい外部アプリケーションを作成する
2. 新しいリボン タブおよびパネルを作成する
3. プッシュ ボタンを作成する
4. スプリット ボタンを作成する
5. コンボ ボックスを作成する
6. 他のコントロールを作成する
7. サマリ

1. 新しい外部アプリケーションを定義する

新しいプロジェクトを作成して、外部アプリケーションを追加します

1.1 Revit API 入門実習の手順に沿ってプロジェクトを作成します。ただし、今回のクラス ライブラリ プロジェクトの名前は **UiVb** または **UiCs** とします(プログラミング言語に依存して変更)。ここでは C#を選択します。

1.2 Class1.cs ファイルの名前を変更します。

- ファイル名: **1_Ribbon.cs**
- アプリケーション クラス名: **UIRibbon**

必要となる参照:

この実習で必要となる追加の参照は、次のとおりです:

- PresentationCore — ビットマップ イメージ処理で使用
- WindowsBase — ビットマップ イメージ処理で使用
- System.Xaml — ビットマップ イメージ処理で使用
- IntroCs/Vb — 入門実習で定義したコマンドを使用

要求される名前空間:

この実習で必要とする名前空間は次のとおりです:

- Autodesk.Revit.DB
- Autodesk.Revit.UI
- Autodesk.Revit.ApplicationServices
- Autodesk.Revit.Attributes
- System.Collections.Generic
- System.Xaml
- System.Diagnostics — デバッグで使用
- System.IO — フォルダを参照するために使用
- System.Windows.Media.Imaging — ビットマップ イメージ処理で使用

1.3 パスを含むいくつかの変数を宣言します(例、リボン コントロール用のイメージ取得のため):

```
<C#>  
/// <summary>
```

```

    /// This is both the assembly name and the namespace
    /// of the external command provider.
    /// </summary>

    const string _introLabName = "IntroCs";
    const string _uiLabName = "UiCs";
    const string _dllExtension = ".dll";

    /// <summary>
    /// Name of subdirectory containing images.
    /// </summary>

    const string _imageFolderName = "Images";

    /// <summary>
    /// Location of managed dll where we have defined the commands.
    /// </summary>

    string _introLabPath;

    /// <summary>
    /// Location of images for icons.
    /// </summary>

    string _imageFolder;
</C#>

```

1.4 ユーティリティ関数を2つ追加します。これらは、イメージフォルダの検索や特定の画像ファイルから `BitmapImage` オブジェクトを作成する手助けをするものです。

```

<C#>
    /// <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>

    string FindFolderInParents( string path, string target )
    {
        Debug.Assert( Directory.Exists( path ),
            "expected an existing directory to start search in" );

        string s;

        do {
            s = Path.Combine( path, target );
            if( Directory.Exists( s ) )

```

```

        {
            return s;
        }
        path = Path.GetDirectoryName( path );
    } while( null != path );

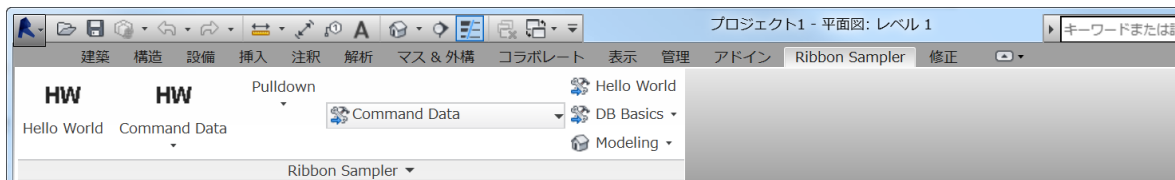
    return null;
}

/// <summary>
/// Load a new icon bitmap from our image folder.
/// </summary>

BitmapImage NewBitmapImage( string imageName)
{
    return new BitmapImage( new Uri(
        Path.Combine( _imageFolder, imageName ) ) );
}
</C#>

```

2. 新しいリボン タブおよびパネルを作成する



Revit 2012 以降のバージョンでは、CreateRibbonTab()を使用して独自のリボン タブを加えることができます。使用する CreateRibbonPanel() のオーバーロードによって、新しく作成されたパネルは、指定するリボン タブや既定の [アドイン] タブのいずれかに追加されます。それでは、リボン タブを作成して、そこにパネルを追加する関数を作成します。

```

<C#>
public void AddRibbonSampler( UIControlledApplication app )
{
    app.CreateRibbonTab( "Ribbon Sampler" );

    RibbonPanel panel =
        app.CreateRibbonPanel( "Ribbon Sampler", "Ribbon Sampler" );
}
</C#>

```

リボンの処理を記述する前に、クラス内のOnStartup() 関数の内側でヘルパー変数を初期化していきます。

```

<C#>
    // External application directory:

    string dir = Path.GetDirectoryName(
        System.Reflection.Assembly
            .GetExecutingAssembly().Location );

```

```

// External command path:

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

if( !File.Exists( _introLabPath ) )
{
    TaskDialog.Show( "UIRibbon", "External command assembly not found: "
        + _introLabPath );
    return Result.Failed;
}

// Image path:

_imageFolder = FindFolderInParents( dir, _imageFolderName );

if( null == _imageFolder
    || !Directory.Exists( _imageFolder ) )
{
    TaskDialog.Show(
        "UIRibbon",
        string.Format(
            "No image folder named '{0}' found in the parent directories of
'{1}'.",
            _imageFolderName, dir ) );

    return Result.Failed;
}
</C#>

```

3. プッシュ ボタンを作成する

いくつかのコントロールを、上で作成したリボン パネルに追加していきます。最初に、プッシュ ボタンを作成します。コントロールを作成する際には、まず、作成しようとしているコントロールのプロパティを設定するための、[コントロール名]Data クラスを作成する必要があります。また、実際のコントロールは、AddItem() を呼び出すことで、作成されることになります(例、コントロール [PushButton] を作成する場合には、PushButtonData インスタンスを作成してプロパティを設定し、AddItem() の引数に渡します)。

ボタンの場合には、ボタンがクリックされた時に呼び出される外部コマンドの完全なクラス名と、そのクラスを実装するアセンブリのパスを指定する必要があります。

注意: アドイン マニフェスト ファイル(*.addin) 中で外部コマンドを指定すれば、コマンドは、[アドイン]リボンタブ >> [外部] リボンパネル >> [外部ツール] から利用可能になります。あるいは、様々なリボン コントロールを作成し、それらのコントロールからコマンドを利用することも可能です。後者の場合、アドイン マニフェスト ファイル内で外部コマンドを宣言する必要はありません。

```

<C#>
public void AddPushButton(RibbonPanel panel)
{
    // Set the information about the command we will be assigning

```

```

// to the button

PushButtonData pushButtonDataHello =
    new PushButtonData( "PushButtonHello", "Hello World",
        _introLabPath, _introLabName + ".HelloWorld" );

// Add a button to the panel

PushButton pushButtonHello =
    panel.AddItem( pushButtonDataHello ) as PushButton;

// 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";
}
</C#>

```

4. スプリット ボタンを作成する

スプリット ボタンは、基本的にプッシュ ボタンをまとめて表示するボタンです。したがって、このコントロールを作成するためには、複数の PushButtonData を作成して、それらを SplitButtonData オブジェクトに加える必要があります。ここでも、ボタンはそれぞれ特定の外部コマンドに関連付けられます。

```

<C#>
public void AddSplitButton(RibbonPanel panel)
{
    // Create three push buttons for split button drop down

    // #1
    PushButtonData pushButtonData1 =
        new PushButtonData( "SplitCommandData", "Command Data",
            _introLabPath, _introLabName + ".CommandData" );
    pushButtonData1.LargeImage = NewBitmapImage( "ImgHelloWorld.png" );

    // #2
    PushButtonData pushButtonData2 =
        new PushButtonData( "SplitDbElement", "DB Element",
            _introLabPath, _introLabName + ".DbElement" );
    pushButtonData2.LargeImage = NewBitmapImage( "ImgHelloWorld.png" );

    // #3
    PushButtonData pushButtonData3 =
        new PushButtonData( "SplitElementFiltering", "ElementFiltering",
            _introLabPath, _introLabName + ".ElementFiltering" );
    pushButtonData3.LargeImage = NewBitmapImage( "ImgHelloWorld.png" );
}

```

```

// Make a split button now
SplitButtonData splitBtnData =
    new SplitButtonData( "SplitButton", "Split Button" );

SplitButton splitBtn = panel.AddItem( splitBtnData ) as SplitButton;
splitBtn.AddPushButton( pushButtonData1 );
splitBtn.AddPushButton( pushButtonData2 );
splitBtn.AddPushButton( pushButtonData3 );
}
</C#>

```

5. コンボ ボックスを作成する

コンボ ボックスの場合は、コンボ ボックス内の各項目に特定の外部コマンドを関連付けるのではなく、選択イベントで処理します。コンボ ボックスを作成したら、選択項目の変更を処理するために CurrentChanged イベントを監視します。

```

<C#>
public void AddComboBox( RibbonPanel panel )
{
    // Create five combo box members with two groups

    // #1
    ComboBoxMemberData comboBoxMemberData1 =
        new ComboBoxMemberData( "ComboCommandData", "Command Data" );
    comboBoxMemberData1.Image = NewBitmapImage( "Basics.ico" );
    comboBoxMemberData1.GroupName = "DB Basics";

    // #2
    ComboBoxMemberData comboBoxMemberData2 =
        new ComboBoxMemberData( "ComboDbElement", "DB Element" );
    comboBoxMemberData2.Image = NewBitmapImage( "Basics.ico" );
    comboBoxMemberData2.GroupName = "DB Basics";

    // #3
    ComboBoxMemberData comboBoxMemberData3 =
        new ComboBoxMemberData( "ComboElementFiltering", "Filtering" );
    comboBoxMemberData3.Image = NewBitmapImage( "Basics.ico" );
    comboBoxMemberData3.GroupName = "DB Basics";

    // #4
    ComboBoxMemberData comboBoxMemberData4 =
        new ComboBoxMemberData( "ComboElementModification", "Modify" );
    comboBoxMemberData4.Image = NewBitmapImage( "Basics.ico" );
    comboBoxMemberData4.GroupName = "Modeling";

    // #5
    ComboBoxMemberData comboBoxMemberData5 =
        new ComboBoxMemberData( "ComboModelCreation", "Create" );
    comboBoxMemberData5.Image = NewBitmapImage( "Basics.ico" );
    comboBoxMemberData5.GroupName = "Modeling";
}

```

```

// Make a combo box now
ComboBoxData comboBxData = new ComboBoxData("ComboBox");
ComboBox comboBx = panel.AddItem(comboBxData) as ComboBox;
comboBx.ToolTip = "Select an Option";
comboBx.LongDescription = "select a command you want to run";
comboBx.AddItem(comboBxMemberData1);
comboBx.AddItem(comboBxMemberData2);
comboBx.AddItem(comboBxMemberData3);
comboBx.AddItem(comboBxMemberData4);
comboBx.AddItem(comboBxMemberData5);

comboBx.CurrentChanged += new
EventHandler<Autodesk.Revit.UI.Events.ComboBoxCurrentChangedEventArgs>(comboB
x_CurrentChanged);
}

void comboBx_CurrentChanged(object sender,
Autodesk.Revit.UI.Events.ComboBoxCurrentChangedEventArgs e)
{
// Cast sender as TextBox to retrieve text value
ComboBox combodata = sender as ComboBox;
ComboBoxMember member = combodata.Current;
TaskDialog.Show("Combobox Selection", "Your new selection: " +
member.ItemText);
}
</C#>

```

6. 他のコントロールを作成する

これまでに紹介したコントロール以外にも様々なコントロールがあります。他のコントロールについては、Autodesk.Revit.UI 名前空間から辿って見つけることができます。

ここでは、AddRibbonSampler() 関数から、すべてのコントロール生成関数を呼び出します。

```

</C#>
public void AddRibbonSampler(UIControlledApplication app)
{
app.CreateRibbonTab("Ribbon Sampler");

RibbonPanel panel =
app.CreateRibbonPanel("Ribbon Sampler", "Ribbon Sampler");

AddPushButton(panel);

AddSplitButton(panel);

AddComboBox(panel);
}
</C#>

```

次に、OnStartup() の終わりで AddRibbonSampler() を呼び出してください。

また、リボン パネル上にスライドアウト コントロールを置くことも出来ます。これは、ユーザがリボン パネルの最下段をクリックすることで表示される部分を指します。

`panel.AddSlideOut()` の呼び出し後にすべてのコントロールがパネルに追加されます。

7. サマリ

この実習では、リボン タブとリボン パネルを作成して、コントロールをそれらに配置する方法を学習しました。学習した項目は次のとおりです。

- 新しいリボン タブおよびパネルを作成する
- 様々なリボン コントロールを作成する