MP2 Notes – Sample plugin see SLAMP2Test

Workflow - Controls the plugin transitions and also which screens get loaded

Workflowstate - The Workflowstate Id is used to define the xaml file that the state is tied to and also the corresponding model id number. The MainScreen should be equal to the corresponding xaml file and the workflowmodel should equal the corresponding model.cs file

Workflowaction – The workflow action controls the transition to the different defined workflow states the source state if using the home page should be 7F702D9C-F2DD-42da-9Ed8-0BA92F07787F other source state would be equal to the previous workflow state id and the target state should be equal to the intended Workflowstate Id.

General Beginning workflow

Home .xaml loads first which has a master_bare.xaml reference in its source

Every time a menu item is clicked it pushes down the workflow state of that plugin which should load master_menu.xaml if the source is set for master_menu.xaml allowing for different home screens from content loaded by plugins.

Plugin.xml – Used to define and register the plugin

When registering models the Classname must match the model namespace and then the model name

Skin Files - Used to change the look and feel of MP2

themes folder - contains the different themes available for that skin
can copy themes from different skins directly

NOTE This is the place to control the tile effect of the metro skin
styles folder under themes contains xaml files to manipulate:
buttons (animation, color, size, etc...)
colors (buttons, backgrounds, etc…)

When changing styles in a xaml file the style must come before it can be used a resource
For example if using a `BasedOn="{ThemeResource MetroStyle}"`
`The MetroStyle must be defined before`

`In general terms any style for a ThemeResource must come before it is used`

screens folder – contains the different screens for that skin
NOTE This is the place to control the screen navigation of the metro skin

XAML files - Used to display the information

When identifying the Model the Id should match the Model Id string in the “project”model.cs file

Don’t forget to include the DataContext element in the xaml file to be able to bind data to the model.

```
DataContext="{Binding Source={StaticResource MetroModel}}">  <!--Must Add DataContext to match the key
of model from above examplbe Model for x:key="Model"-->
```

Buttons must have a command property set which can tie into just a public void method in the model. Can change the style through the ThemeResource to change the look of the buttons.

```
<Button Grid.Column="0" Grid.Row="1" Style="{ThemeResource DialogButtonStyle}" Content="MAX"
              HorizontalAlignment="Center" Command="{Command Tile1CMD}"/>
```

Buttons (most controls) must have a style resource defined. Also in that style resource a content presenter needs to be part of the style to display the content and content must be defined in the xaml button (either binding or static) in order for the button to show up.

Must utilize `WProperty` for binding elements that will update in codebehind per the following example…

```
protected AbstractProperty m_samplebtncontent = new WProperty(typeof(string), string.Empty);

public string SampleBtnContent
        {
            get { return (string)m_samplebtncontent.GetValue(); }
            internal set { m_samplebtncontent.SetValue(value); }
        }

public AbstractProperty SampleBtnContentProperty
        {
            get { return m_samplebtncontent; }
        }
```

Bind xaml to `SampleBtnContent (ie actual property not abstract property)`

```
<Label DataContext="{Binding Source={StaticResource StartModel}}" Grid.Column="0" Grid.Row="1"
Content="{Binding Path=SampleBtnContent,Mode=TwoWay}"/>
```

Models – Used to keep information for the plugin to use much like MVVM model

Must add model registration to plugin.xml regardless of plugin or skin anytime a model will be used.

Build – Notes on how to setup plugin

Plugin structure must be added to the MP2 folder under (until fully installed)

C:\Users\Chad\MP2\MediaPortal\Bin\MP2-Client\bin\x86\Debug\Plugins

Can Auto add the structure by adding the following code to the project properties under
build events tab – Post build events this will place the project in its proper location from above
Also Make sure that the build output path is set for bin\x86\Debug\

```
xcopy /Y "$(ProjectDir)plugin.xml" "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\"

xcopy /Y "$(ProjectDir)$(OutDir)SLAMP2Test.dll"
"$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\"

mkdir "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\Language"

robocopy "$(ProjectDir)Language" "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\Language"
/MIR /NP /XD .svn

mkdir "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\Skin"

robocopy "$(ProjectDir)Skin" "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\Skin" /MIR /XD
.svn

xcopy /Y "$(TargetDir)$(ProjectName).dll"  "$(SolutionDir)..\Bin\$(SolutionName)\$(OutDir)Plugins\$(ProjectName)\"
```

General C# notes

To view all settings if there isn't an option to set to debug mode

Go to Tools – Settings – Select Expert settings

Next go to Tools – Options (make sure show all settings is checked in bottom left) under Projects and Solutions – General select Show Advanced build configurations