The use of PART_ in Control Templates

One of the most useful features of WPF is the ability to completely redesign the look of a control such a button, progress bar or slider bar relatively easily through the use of control templates. Meaning a progress bar can become round, a slider bar 3D or some other shape.

With the more complex controls (such as the progress bar) you can also keep the functionality of the original control and save having to re-write the base logic. This is done by specifying that elements in your design are the same as part of the original control. The link is created through simply naming your elements by set names usually prefixed with PART_.

In the example below we have a ControlTemplate for a ProgressBar. In order for it to keep it's function as a progress bar the I have named the elements "PART_Track" and "PART_Indicator" to link to the two parts of original functionality. In the case of a progress bar without this when the value property is set on the progresses bar it would not be reflected in the redesigned control.

```
<ControlTemplate x:Key="DarkBar" TargetType="{x:Type ProgressBar}">
    <Grid>
        <Border Name="PART_Track" CornerRadius="10" BorderThickness="2" BorderBrush="DarkG
        <Border Name="PART_Indicator" CornerRadius="10" Background="Gray" HorizontalAlignm
    </Grid>
</ControlTemplate>
```

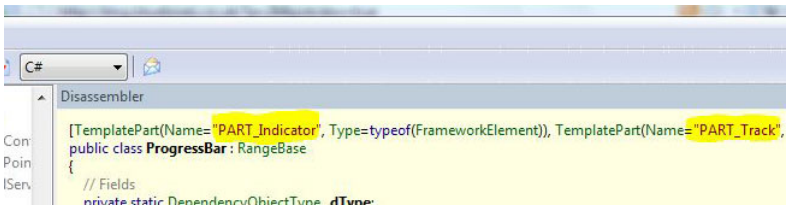
For each control type the names that you use to specify the link are different, although they are all prefixed with the word PART_ . These is no full list online however as these are specified as TemplatePartAttributeattributes on the controls it means that they are relatively easy to discover.

Using reflection (code sample below) it is possible to get a full list.

```
Dim extAssembly As System.Reflection.Assembly = System.Reflection.Assembly.LoadFile("C:Program FilesReference AssembliesMicrosoftFrameworkv3.0PresentationFramework.dll")
For Each t As System.Type In extAssembly.GetTypes
For Each att As System.Windows.TemplatePartAttribute In t.GetCustomAttributes(GetType(System.Windows.TemplatePartAttribute), True)
Console.WriteLine(t.ToString & " : " & att.Name.ToString & " Type of " & att.Type.ToString)

Next
Next
```

Alternatively and often more practical is to use Red Gates Reflector tool. With this free application you can search through and look at the source code of the controls. The part will be specified at the top along with the other attributes as shown below.



This has been a quick overview and further reading is often need to discover just how to use the parts as they differ per control however hopefully you can start to see how useful they are when redesigning controls.