Modelos e controles personalizados

José Antônio da Cunha

IFRN

Modelos e controles personalizados

Modelos (Templates)

Os estilos permitem alterar a aparência de um elemento. No entanto, os estilos são limitados a definição de propriedades que são definidas na classe do elemento.

O Silverlight tem outra ferramenta de personalização chamada templates. Embora você possa usar estilos com qualquer elemento do Silverlight, os modelos são limitados a controles Silverlight, em outras palavras, elementos que herda da classe Control no namespace System.Windows.Controls.

Modelos e controles personalizados

Modelos (Templates)

Estes elementos adquirem uma propriedade chamada de ***template***, que pode ser configurado para aplicar um modelo personalizado, efetivamente substituindo o visual padrão do controle.

Por exemplo, mudando o modelo usado por um objeto de botão, você pode criar muitos outros tipos e botão que seria impensável só com um estilo.

Modelos e controles personalizados

Criando um Modelos (Templates)

```
<Button Content="Um botão modelado" Height="30" Width="300">
        <Button.Template>
            <ControlTemplate TargetType="Button" >
                <Border BorderBrush="Orange" BorderThickness="3" CornerRadius="10"
Background="Red" >
                    <TextBlock Foreground="White" Text="A Custom Template"></TextBlock>
                </Border>
            </ControlTemplate>
        </Button.Template>
</Button>
```

Modelos e controles personalizados

Reutilização de Control Templates

No exemplo anterior, a definição do modelo está aninhado dentro do elemento. Mas é muito mais comum definir o modelo de um controle através de um estilo. Isso dá a possibilidade de reutilizar o modelo para várias instâncias do mesmo controle.

Para acomodar este projeto, você precisa definir o seu modelo de controle como um recurso:

```
<UserControl.Resources>
      <ControlTemplate x:Key="ButtonTemplate" TargetType="Button" >
         <Border BorderBrush="Orange" BorderThickness="3" CornerRadius="10"
                        Background="Red">
            <TextBlock Foreground="White" Text="A Custom Template"></TextBlock>
         </Border>
      </ControlTemplate>
</UserControl.Resources>
```

Modelos e controles personalizados

Você pode consultá-lo usando uma referência **StaticResource**, como mostrado aqui:

```
<Button Template="{StaticResource ButtonTemplate}"
        Content="A Template Button"
        Height="30" Width="300">
</Button>
```

Modelos e controles personalizados

O ContentPresenter

O ContentPresenter é necessário para todos os controles de conteúdo e é uma espécie de marcador que diz ao Silverlight para onde o conteúdo deve ser inserido.

```
<Button Height="30" Width="300" Margin="50,191,50,79">
        <Button.Template>
            <ControlTemplate x:Name="ButtonTemplate" TargetType="Button" >
                <Border BorderBrush="Orange" BorderThickness="3"
                        CornerRadius="10" Background="Red" >
                    <ContentPresenter>
                        <TextBlock Foreground="White" Text="Um novo
                                                    botão"></TextBlock>
                    </ContentPresenter>
                </Border>
            </ControlTemplate>
        </Button.Template>
</Button>
```

Modelos e controles personalizados

Estado do Controle Button

Estados são colocados juntos em grupo. Os grupos são mutuamente exclusivos, o que significa que, um controle tem um estado em cada grupo. Por exemplo, o botão tem dois grupos estado: CommonStates e FocusStates.

Por exemplo, se você usar a tecla Tab para passar de um botão para outro, os seus estados passaram de normal (de CommonStates) e foco (do FocusStates). Se você mova o mouse sobre o botão, os seus estados serão MouseOver ((de CommonStates) e foco ((de FocusStates).

Modelos e controles personalizados

Estado do Controle Button

Para definir os grupos do estado, você deve adicionar VisualStateManager.VisualSateGroups no elemento raiz de seu control template, como mostrado aqui:

Modelos e controles personalizados

```
<Grid x:Name="LayoutRoot" Background="White">
      <Button Height="30" Width="300">
         <Button.Template>
            <ControlTemplate x:Name="ButtonTemplate" TargetType="Button" >
               <Grid>
                  <VisualStateManager.VisualStateGroups>
                     <VisualStateGroup x:Name="CommState">
                        <VisualState x:Name="MouseOver">
                           <Storyboard>
                              <ColorAnimation Duration="0:0:0"
                                          Storyboard.TargetName="ButtonBackgroundBrush"
                                          Storyboard.TargetProperty="Color" To="Orange" />
                           </Storyboard>
                        </VisualState>
                        <VisualState x:Name="Normal">
                           <Storyboard>
                              <ColorAnimation Duration="0:0:0"
                                          Storyboard.TargetName="ButtonBackgroundBrush"
                                          Storyboard.TargetProperty="Color" />
                           </Storyboard>
                        </VisualState>
                     </VisualStateGroup>
                  </VisualStateManager.VisualStateGroups>
                  <Border x:Name="ButtonBorder" BorderBrush="OrangeRed" BorderThickness="3" CornerRadius="15">
                     <Border.Background>
                        <SolidColorBrush x:Name="ButtonBackgroundBrush" Color="Red" />
                     </Border.Background>
                     <ContentPresenter></ContentPresenter>
                  </Border>
               </Grid>
            </ControlTemplate>
         </Button.Template>
      </Button>
   </Grid>
```

Modelos e controles personalizados

Mostrando o Focus

No exemplo anterior, você usou os estados normal e MouseOver do grupo CommonStates para controlar como o botão aparece quando o mouse passa por cima do mesmo. Você também pode adicionar os estados Pressionado e Disabilitado para personalizar suas outras duas alternativas. Esses quatro estados são mutuamente exclusivos.

Muitos controles usam um foco para indicar quando o controle está selecionado. O é indicado por um retângulo com uma borda pontilhada. O exemplo a seguir mostra isto:

Modelos e controles personalizados

```
<Button Height="30" Width="300" x:Name="btnButton" Margin="63,95,37,175">
        <Button.Template>
          <ControlTemplate x:Name="ButtonTemplate" TargetType="Button" >
            <Grid>
              <VisualStateManager.VisualStateGroups>
                <VisualStateGroup x:Name="FocusStates">
                  <VisualState x:Name="Focused">
                    <Storyboard>
                      <DoubleAnimation Duration="0" Storyboard.TargetName="FocusVisualElement"
                                       Storyboard.TargetProperty="Opacity" To="1"/>
                    </Storyboard>
                  </VisualState>

                  <VisualState x:Name="UnFocused">
                    <!-- Coloque o código aqui para quando o botão perder o foco-->
                  </VisualState>
                </VisualStateGroup>
              </VisualStateManager.VisualStateGroups>
              <Border x:Name="ButtonBorder" BorderBrush="OrangeRed" BorderThickness="3" CornerRadius="15">
                <Border.Background>
                  <SolidColorBrush x:Name="ButtonBackgroundBrush" Color="Red" />
                </Border.Background>
                <ContentPresenter></ContentPresenter>
              </Border>

              <Rectangle x:Name="FocusVisualElement" Stroke="Black" Margin="8" Opacity="0"
                         StrokeThickness="1" StrokeDashArray="1 2"></Rectangle>

            </Grid>
          </ControlTemplate>
        </Button.Template>
      </Button>
```

Modelos e controles personalizados

Transições

Você pode aumentar a duração para criar um efeito que vai misturando gradualmente as cores. Aqui está um exemplo onde a cor desaparece e uma nova cor surge a cada 0,2 segundo:

```
<VisualStateManager.VisualStateGroups>
                    <VisualStateGroup x:Name="CommonStates">
                        <VisualStateGroup.Transitions>
                            <VisualTransition GeneratedDuration="0:0:0.2"/>
                        </VisualStateGroup.Transitions>
                        <VisualState x:Name="MouseOver">
                            <Storyboard>
                                <ColorAnimation Duration="0:0:0"
                                                Storyboard.TargetName="ButtonBackgroundBrush"
                                                Storyboard.TargetProperty="Color" To="Orange" />
                            </Storyboard>
                        </VisualState>
                        <VisualState x:Name="Normal">

                        </VisualState>
                    </VisualStateGroup>
                </VisualStateManager.VisualStateGroups>
```

Modelos e controles personalizados

Transições outro exemplo

```
<VisualStateManager.VisualStateGroups>
      <VisualStateGroup x:Name="CommonStates">
            <VisualStateGroup.Transitions>
                  <VisualTransition To="MouseOver" GeneratedDuration="0:0:0.5"/>
                   <VisualTransition From="MouseOver" GeneratedDuration="0:0:0.1"/>
            </VisualStateGroup.Transitions>
            <VisualState x:Name="MouseOver">
                 <Storyboard>
                      <ColorAnimation Duration="0:0:0"
                              Storyboard.TargetName="ButtonBackgroundBrush"
                              Storyboard.TargetProperty="Color" To="Orange" />
                 </Storyboard>
             </VisualState>
             <VisualState x:Name="Normal">

             </VisualState>
      </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```