Silverllight

Templates

Templates

Apesar da aplicação de estilos permitir a personalização de muitos aspectos visuais de um controle, a verdade é que podemos ir um pouco mais longe e efetuar a personalização completa do seu aspecto através da definição do *template* do controle.

A classe *Control* disponibiliza uma propriedade, designada por *Template,* espera um elemento do tipo ContentTemplate*.* É este objeto que define o aspecto gráfico do controle em rutime. Em outras palavras, é este objeto que define a árvore visual utilizada pelo controle.

Templates

Definição da Interface Básica

O primeiro passo necessário à construção de um *template* passa pela criação de um objeto ***ContentTemplate*** que serve de container aos outros elementos visuais que definem a interface gráfica do controle. Neste caso, vamos criar um *template* que usa uma elipse para delimitar a área do controle.

No exemplo, vamos criar um botão em forma de elipse. Veja o código a seguir:

Templates

```
<Window.Resources>
      <ControlTemplate TargetType="Button" x:Key="btTemplate">
         <Grid>
            <Ellipse x:Name="background"
                     Stroke="{TemplateBinding BorderBrush}"
                     Width="{TemplateBinding Width}"
                     Height="{TemplateBinding Height}">
               <Ellipse.Fill>
                  <LinearGradientBrush EndPoint=".7,1" StartPoint=".7,0">
                     <GradientStop Color="#FFFFFFFF" Offset="0"/>
                     <GradientStop Color="#F9FFFFFF" Offset="0.375"/>
                     <GradientStop Color="#E5FFFFFF" Offset="0.625"/>
                     <GradientStop Color="#C6FFFFFF" Offset="1"/>
                  </LinearGradientBrush>
               </Ellipse.Fill>
            </Ellipse>
         </Grid>
      </ControlTemplate>
   </Window.Resources>
```

Templates

Atribuindo o *template* ao botão.

```
<Grid>
      <Button Content="Estilo inLine" Height="70" Width="200"
            Template="{StaticResource btTemplate}"/>
   </Grid>
```

Observação: se você executar esta aplicação, irá perceber que o conteúdo não foi impresso no botão, ou seja, o Content do botão, não foi inserido ao mesmo. Isto se deve, porque a renderização do conteúdo, fica a cargo do controle *ContentPresenter*.

Então, acrescente o código a seguir ao recurso.

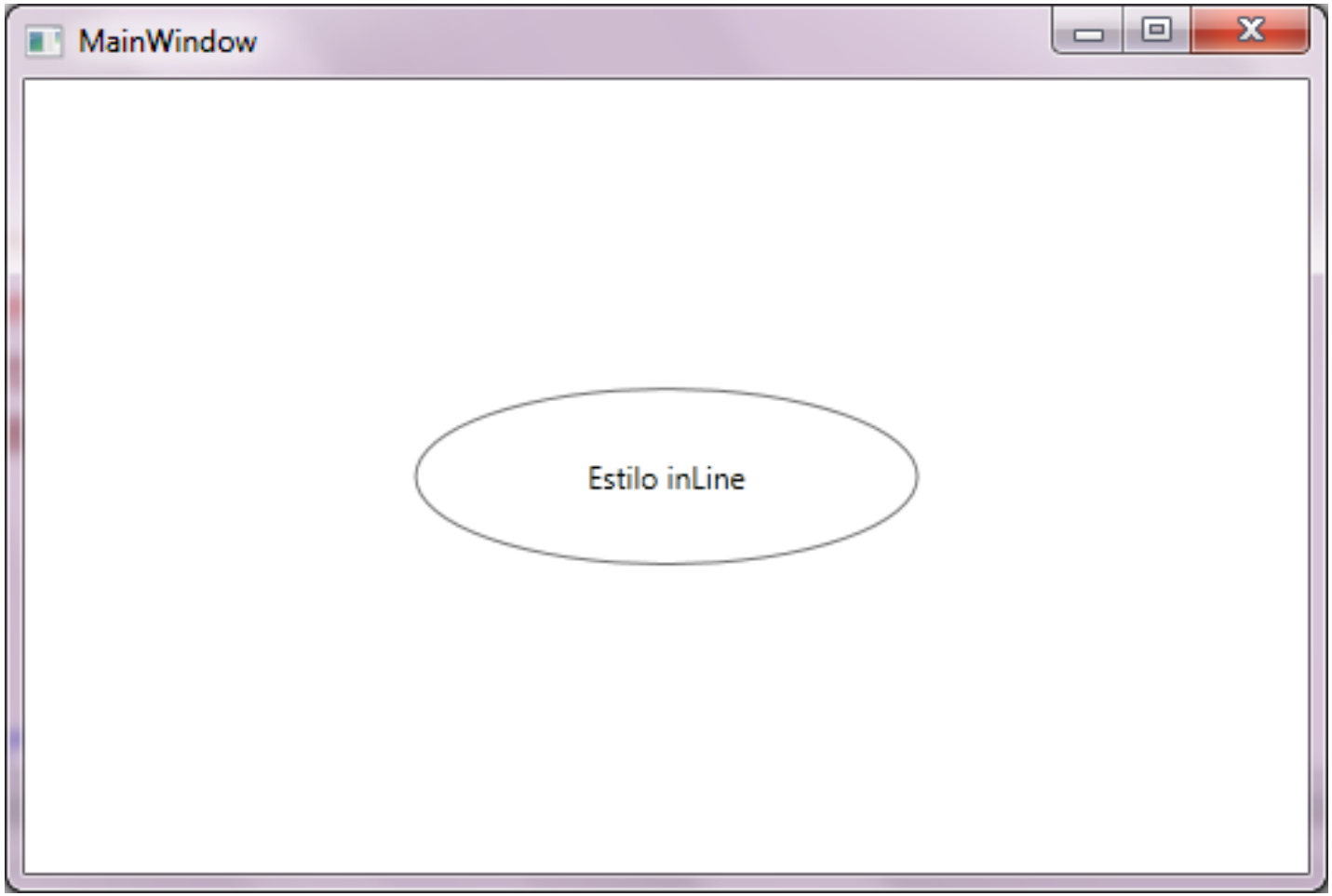
Templates

```
<ContentPresenter x:Name="contentPresenter"
                  ContentTemplate="{TemplateBinding
ContentTemplate}"
                  Content="{TemplateBinding Content}"
                  HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                  VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
```

Se você executar a aplicação novamente, verá que o conteúdo foi apresentado.

A extensão de markup ***TemplateBinding*** faz com que os valores das propriedades do controle sejam propagados para as propriedades dos elementos usados no interior do *template*.

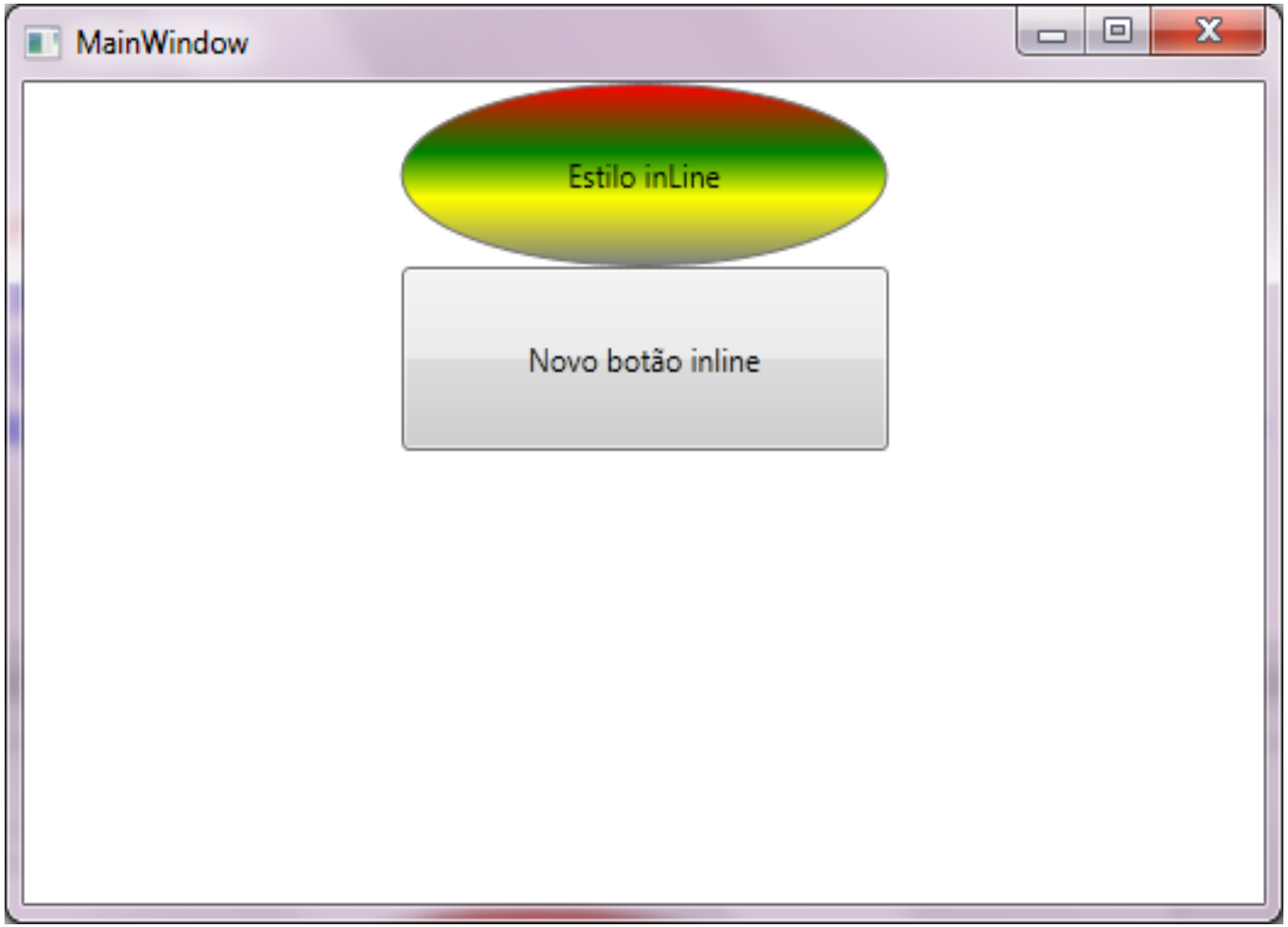
Templates



Templates

Acrescente um outro botão ao projeto. Agora não use o template definido na página. Faça isso, usando um StackPanel, ou seja coloque os dois botões, dentro de um StackPanel Execute a aplicação e veja a diferença entre os dois. Veja o resultado na página seguinte.

Templates



Templates

Definição de estados

Ao executar a aplicação, observe que quando passamos o mouse sobre o botão elipse, nenhum evento é disparado. Enquanto que, quando passamos o mouse sobre o botão normal, o mesmo muda seu estado. Para resolver esta situação, temos que modificar o *template* e adicionar os elementos necessários ao suporte dos vários estados visuais pelos quais o botão pode passar.