Criar um botão usando XAML

O objetivo deste laboratório é mostrar como criar um botão animado para uso em um aplicativo Windows Presentation Foundation (WPF). Vamos usar estilos e um modelo para criar um botão personalizado que permite a reutilização de código e a separação da lógica do botão da declaração botão. Vamos fazer tudo em Extensible Application Markup Language (XAML).

A figura a seguir mostra o resultado final.

Criando um Botão básico

Vamos iniciar criando um novo projeto e adicionando alguns botões para a janela.

Para criar um novo projeto WPF e adicionar botões para a janela (window)

1. Inicie o Visual Studio.
2. Criar um novo projeto WPF: atribua o seguinte nome ao projeto "AnimatedButton".
3. **Adicione botões padrão básico:** abra o arquivo xaml . Remova o Grid e adicione um StackPanel e alguns botões. Veja o código XAML a seguir:

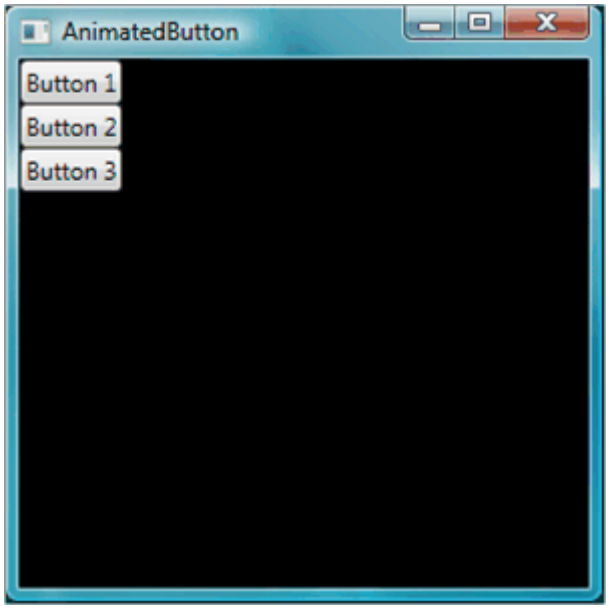
```
<Window x:Class="AnimatedButton.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="AnimatedButton" Height="300" Width="300"
  Background="Black">

  <!-- Buttons arumados verticalmente dentro do StackPanel. -->
  <StackPanel HorizontalAlignment="Left">
```

```
    <Button>Button 1</Button>
    <Button>Button 2</Button>
    <Button>Button 3</Button>
  </StackPanel>

</Window>
```

Pressione F5 para rodar o aplicativo; veja o resultado na figura seguinte.



Agora que você criou os botões, vamos definir os estilos e um template para os botões. Só que vamos fazer isto no arquivo app.xaml.

Definido as propriedades Básicas

Em seguida, vamos definir algumas propriedades destes botões para definir a aparência dos mesmos. Ao invés de definir as propriedades individuais dos botões vamos usar recursos para definir as propriedades do botão para o aplicativo inteiro. Recursos de aplicativo são conceitualmente semelhantes a Cascading Style Sheets (CSS) para páginas da Web; No entanto, os recursos são muito mais poderosos do que Cascading Style Sheets (CSS), como você verá no final deste laboratório.

Para usar estilos para definir as propriedades básicas sobre botões

1. **Defina um bloco de Application.Resources:** Abra o arquivo App.xaml. Veja a seguir:

```
<Application x:Class="AnimatedButton.App"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  StartupUri="Window1.xaml"
  >
  <Application.Resources>

    <!—Recurso para aplicação pode ser definida aqui. -->

  </Application.Resources>
</Application>
```

O escopo de um recurso é determinado por onde você define o recurso. Recursos definidos no arquivo app.xaml, ou seja, definidos na aplicação, permite que o mesmo seja aplicado em qualquer lugar do aplicativo.

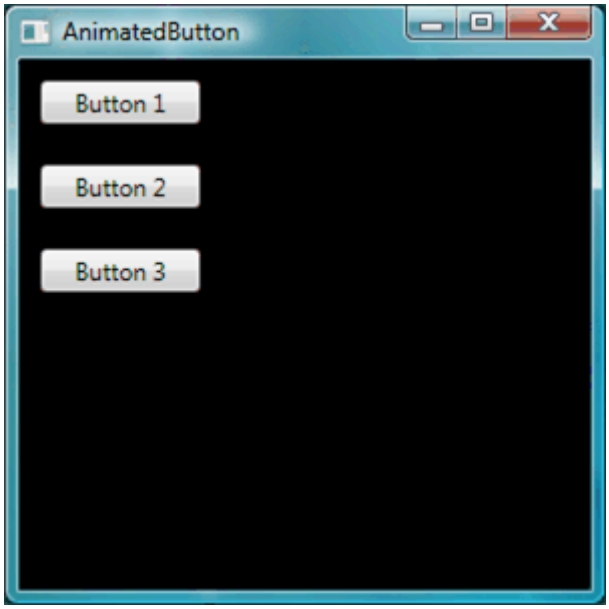
2. Vamos criar um estilo e definir os valores em sua propriedade: Adicione a seguinte tag no bloco Application.Resources. Esta marcação criar um estilo(Style) que aplica a todos os botões no aplicativo, definido o tamanho (Width) pata 90 e a margem (Margin) para 10:

```
<Application.Resources>

  <Style TargetType="Button">
    <Setter Property="Width" Value="90" />
    <Setter Property="Margin" Value="10" />
  </Style>

</Application.Resources>
```

A propriedade TargetType especifica que o estilo se aplica a todos os objetos do tipo Button. Cada Setter define um valor diferente para a propriedade Style. Portanto, neste momento todos os botões no aplicativo, têm uma largura de 90 e uma margem de 10. Se você pressionar F5 para executar o aplicativo, você vê a janela a seguir.



3. **Defina um valor de propriedade de estilo a um recurso:** Recursos permitem uma maneira simples de reutilizar objetos e valores definidos comumente. É especialmente útil para definir valores complexos para tornar seu código mais modular. Adicione a marcação a seguir ao arquivo app.xaml.

```
<Application.Resources>

  <LinearGradientBrush x:Key="GrayBlueGradientBrush"
    StartPoint="0,0" EndPoint="1,1">
    <GradientStop Color="DarkGray" Offset="0" />
    <GradientStop Color="#CCCCFF" Offset="0.5" />
    <GradientStop Color="DarkGray" Offset="1" />
```

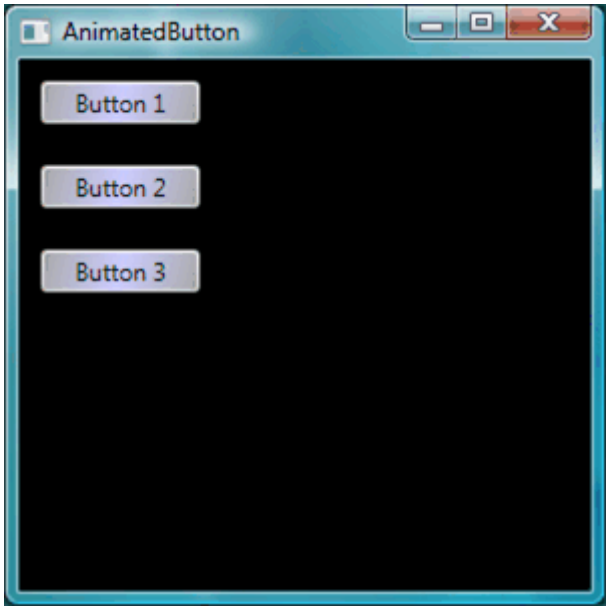
```
    </LinearGradientBrush>

    <Style TargetType="{x:Type Button}">
      <Setter Property="Background"
        Value="{StaticResource GrayBlueGradientBrush}" />
      <Setter Property="Width" Value="80" />
      <Setter Property="Margin" Value="10" />
    </Style>

</Application.Resources>
```

Você criou diretamente no bloco Application.Resources, um recurso chamado "GrayBlueGradientBrush". Este recurso define um gradiente horizontal. Este recurso pode ser usado como um valor de propriedade em qualquer lugar no aplicativo, incluindo a propriedade Background, no interior de Setter, no Estilo. Agora, todos os botões tem a propriedade Background deste gradiente.

Pressione F5 para rodar o aplicativo. Ele deve ter a seguinte aparência:



Criando um modelo que define o aspecto do Botão

Nesta seção, você vai criar um modelo para personalizar a aparência (apresentação) do botão. A aparência do botão, será composta de vários objetos incluindo triângulos e outros componentes para dar uma aparência única.

Os modelos permitem poderosos controles sobre a aparência de um objeto. Como os modelos podem ser usados dentro de estilos, você pode aplicar um modelo para todos os objetos que o estilo se aplica.

Para usar o modelo definido para o botão

1. **Configure o modelo:** Os controles, como Button tem uma propriedade Template, e você pode definir o valor da propriedade de modelo, da mesma forma que você definiu as

propriedade em um Style usando um Setter. Adicione a seguinte marcação para o estilo do botão.

```
<Application.Resources>

  <LinearGradientBrush x:Key="GrayBlueGradientBrush"
    StartPoint="0,0" EndPoint="1,1">
    <GradientStop Color="DarkGray" Offset="0" />
    <GradientStop Color="#CCCCFF" Offset="0.5" />
    <GradientStop Color="DarkGray" Offset="1" />
  </LinearGradientBrush>

  <Style TargetType="{x:Type Button}">
  <Setter Property="Background" Value="{StaticResource GrayBlueGradien
tBrush}" />
    <Setter Property="Width" Value="80" />
    <Setter Property="Margin" Value="10" />
    <Setter Property="Template">
      <Setter.Value>
        <!—o template do botão é definodo aqui. -->
      </Setter.Value>
    </Setter>
  </Style>

</Application.Resources>
```

2. **Altere a apresentação do botão:** Neste ponto, você precisará definir o modelo. Adicione a seguinte marcação destacada (negrito). Esta marcação especifica dois retângulos como bordas arredondadas, seguido por um DockPanel. O DockPanel é usado para hospedar o ContentPresenter do botão. A ContentPresenter exibe o conteúdo do botão. No nosso caso, o conteúdo em text ("Button 1", "Button 2", "Button 3"). Todos os componentes do modelo (o retângulo e o DockPanel) são definidos dentro de um Grid.

   Copiar

```
<Setter.Value>
  <ControlTemplate TargetType="Button">
    <Grid Width="{TemplateBinding Width}"
     Height="{TemplateBinding Height}" ClipToBounds="True">

      <!-- Retângulo com cantos arredondados. -->
      <Rectangle x:Name="UmRectangle"
        HorizontalAlignment="Stretch"
        VerticalAlignment="Stretch"
        Stroke="{TemplateBinding Background}"
        RadiusX="20" RadiusY="20" StrokeThickness="5"
        Fill="Transparent" />

      <!-- Retângulo com cantos arredondados -->
      <Rectangle x:Name="innerRectangle"
        HorizontalAlignment="Stretch"
        VerticalAlignment="Stretch" Stroke="Transparent"
        StrokeThickness="20"
        Fill="{TemplateBinding Background}"
        RadiusX="20" RadiusY="20"   />

      <!-- Apresenta o Content (text) do the button. -->
      <DockPanel Name="myContentPresenterDockPanel">
```

```
        <ContentPresenter x:Name="myContentPresenter" Margin="20"
          Content="{TemplateBinding  Content}"
          TextBlock.Foreground="Black" />
      </DockPanel>
    </Grid>
  </ControlTemplate>
</Setter.Value>
```

Pressione F5 para rodar a aplicação. Ele deve ter a seguinte aparência.