Silverlight

JOSÉ ANTÔNIO DA CUNHA
IFRN

Layout com Painéis

Roteiro

- Canvas
- StackPanel
- WrapPanel
- DockPanel
- Grid

Layout Painéis

A janela do Silverlight pode conter apenas um único elemento. Para caber mais de um elemento e criar uma interface de usuário mais prática, você precisa colocar um recipiente na sua página e, em seguida, adicionar outros elementos para o recipiente. Seu layout é determinado pelo recipiente que você usa.

Todos os recipientes de layout Silverlight são painéis que derivam da classe abstrata **System.Windows.Controls.Panel**.

Layout Painéis

Canvas

Canvas é o painel mais básico. Canvas oferece suporte somente para a notação “clássica” do posicionamento de elementos com coordenadas explícitas. Você posiciona elementos em um Canvas usando propriedades: Left, Top.

```
<Canvas>
    <Button Background="Red">Left=0, Top=0</Button>
    <Button Canvas.Left="18" Canvas.Top="18" Background="Orange">Left=18, Top=18</Button>
  </Canvas>
```

Layout Painéis

StackPanel

StackPanel é um painel popular pela sua simplicidade e utilidade. Como seu nome sugere, ele simplesmente empilha seus filhos seqüencialmente.

```
<StackPanel Orientation="Horizontal" >
      <Button Background="Red">1</Button>
      <Button Background="Red">2</Button>
      <Button Background="Red">3</Button>
      <Button Background="Red">4</Button>
      <Button Background="Red">5</Button>
   </StackPanel>
```

Layout Painéis

WrapPanel

WrapPanel é similar a StackPanel. Mas além de empilhar seus elementos filhos, distribui-os em linhas ou colunas adicionais quando não houver espaço suficiente para um única pilha.

```
<toolkit:WrapPanel Margin="3">
    <Button VerticalAlignment="Top" Content="Top Button"></Button>
    <Button MinHeight="60" Content="Tall Button"></Button>
    <Button VerticalAlignment="Bottom" Content="Bottom Button"></Button>
    <Button Content="Stretch Buttom"></Button>
    <Button VerticalAlignment="Center" Content="Centered Buttom"></Button>
  </toolkit:WrapPanel>
```

Layout Painéis

DockPanel

O toolkit Silverlight também inclui um recipiente de layout, chamado de DockPanel. Ele estende os controles ao longo de suas bordas. Como exemplo de DockPanel, podemos citar os toolbar, ou seja, um toolbar é um DockPanel.

A questão óbvia é esta. Como elementos filho sabe o lado onde se deseja encaixar? A resposta é através de uma propriedade anexada Dock, que pode ser definida para a esquerda, direita, superior ou inferior.

```
<toolkit:DockPanel LastChildFill="True">
      <Button toolkit:DockPanel.Dock="Top" Content="Top Buttom"></Button>
      <Button toolkit:DockPanel.Dock="Bottom" Content="Bottom Buttom"></Button>
      <Button toolkit:DockPanel.Dock="Left" Content="Left Buttom"></Button>
      <Button toolkit:DockPanel.Dock="Right" Content="Right Buttom"></Button>
      <Button Content="Ramining Space"></Button>
   </toolkit:DockPanel>
```

Layout Painéis

Grid

O Grid é o recipiente de layout mais poderoso do Silverlight. Na verdade, o Grid é tão útil que, quando você adicionar um documento XAML novo para uma página no Visual Studio, ele automaticamente adiciona as tags Grid como o contêiner de primeiro nível, aninhado dentro do elemento UserControls raiz.

O Grid separa os elementos em uma grade invisível de linhas e colunas. Você pode colocar mais de uma elemento por célula. No entanto, o bom senso pede que você coloque apenas um elemento por célula.

Veja o exemplo a seguir:

Layout Painéis

Grid

```
<Grid x:Name="LayoutRoot" Background="White" ShowGridLines="True" >
      <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
        <RowDefinition></RowDefinition>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
      </Grid.ColumnDefinitions>
      <Button Grid.Row="0" Grid.Column="0" Content="Top Left"></Button>
      <Button Grid.Row="0" Grid.Column="1" Content="Middle Left"></Button>
      <Button Grid.Row="1" Grid.Column="2" Content="Bottom Right"></Button>
      <Button Grid.Row="1" Grid.Column="1" Content="Bottom Middle"></Button>
   </Grid>
```

Layout Painéis

Grid

Como você viu, o Grid fornece a capacidade para criar uma coleção de linhas e colunas proporcionais, que muitas vezes é bastante útil. No entanto, você pode mudar a maneira como cada linha e coluna é dimensionada.

O Grid suporta estratégias de tamanho:
•**Absoluto**: Você indica o tamanho exato usando pixels.
•**Automático**: Cada linha ou coluna é exatamente a quantidade de espaço que ele precisa e não mais.
•**Proporcional**: O espaço é dividido entre um grupo de linhas ou colunas.

Layout Painéis

Grid

Definir o tamanho da coluna em valor absoluto de 100 pixes:

```
<ColumnDefinition Width="100"></ColumnDefinition>
```

Para usar o tamanho automático, você deve usar o valor Auto:

```
<ColumnDefinition Width="Auto"></ColumnDefinition>
```

Finalmente, para usar o tamanho proporcional, você usa um asterisco (*):

```
<ColumnDefinition Width="*"></ColumnDefinition>
```

```
<RowDefinition Height=*></RowDefinition>
<RowDefinition Height=2*></RowDefinition>
```

Layout Painéis

Aninhando Grid e StackPanel

```
<Grid x:Name="LayoutRoot" Background="SteelBlue"  ShowGridLines="True"
                HorizontalAlignment="Center" VerticalAlignment="Center">
      <Grid.RowDefinitions>
        <RowDefinition Height="*"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
      </Grid.RowDefinitions>
      <TextBlock Margin="10" Grid.Row="0" Foreground="White" Text="Isto é     um simples teste de
layout."></TextBlock>
      <StackPanel Grid.Row="1" HorizontalAlignment="Right"           Orientation="Horizontal" >
        <Button Margin="10,10,2,10" Padding="3" Content="OK"></Button>
        <Button Margin="2,10,10,10" Padding="3" Content="Cancel"></Button>
      </StackPanel>
   </Grid>
```

Layout Painéis

Span de Row e Colunas

```
<Grid x:Name="LayoutRoot" Background="SteelBlue" HorizontalAlignment="Center"
VerticalAlignment="Center">
      <Grid.RowDefinitions>
        <RowDefinition Height="*"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
      </Grid.ColumnDefinitions>
      <TextBlock Margin="10" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3"
            Foreground="White" Text="Isto é um exemplo de layout simples."></TextBlock>
      <Button Margin="10,10,2,10" Padding="3" Grid.Row="1" Grid.Column="1"
            Content="OK"></Button>
      <Button Margin="2,10,10,10" Padding="3" Grid.Row="1" Grid.Column="2"
            Content="Cancel"></Button>
   </Grid>
```

Layout Painéis

GridSplitter

Todo usuário do Windows tem visto janelas com barras de separação arrastáveis. Por exemplo, quando você usa o Windows Explorer, você tem do lado esquerdo uma lista de pastas e, do lado direito uma lista com os arquivos da pasta selecionada. Você pode arrastar a barra de separação entre os painéis, aumentando ou diminuindo o espaço de cada painel.

No Silverlight você pode criar algo similar, adicionando um Splitter bar para um Grid. Veja o exemplo a seguir:

Layout Painéis

GridSplitter

```
<Grid x:Name="LayoutRoot" Background="White">
      <Grid.ColumnDefinitions>
        <ColumnDefinition MinWidth="100"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition MinWidth="50"></ColumnDefinition>
      </Grid.ColumnDefinitions>
      <Button Grid.Column="0" Margin="3" Content="Lado esquerdo do Grid“Height="50"
Width="100></Button>
      <sdk:GridSplitter Grid.Column="1" Grid.RowSpan="2" Background="LightGray"          Width="3“
VerticalAlignment="Stretch" HorizontalAlignment="Center"
                ShowsPreview="False"></sdk:GridSplitter>
      <Button Grid.Column="2" Margin="3" Content="Lado direito do Grid“ Height=“50”
Width=“100”></Button>
   </Grid>
```

Layout Painéis

Camadas com Zindex - Canvas

Todo elemento tem seu Zindex = 0. Você pode promover um elemento para um nível mais alto incrementando seu Zindex. Veja o exemplo a seguir:

```
<Canvas x:Name="canvasBackground" Width="200" Height="500" Background="AliceBlue">
    <Button Canvas.Left="60" Canvas.Top="80" Canvas.ZIndex="1" Width="50" Height="50"
Content="(60,80)"></Button>
    <Button Canvas.Left="70" Canvas.Top="120" Width="100" Height="50"
Content="(70,120)"></Button>
  </Canvas>
```

Layout Painéis

Scrolling

Nenhum dos containers tem suporte direto a scrolling. Para que você consiga o efeito de scrolling você deve utilizar o controle ScrollViewer. Veja o exemplo a seguir:

```
<ScrollViewer Background="AliceBlue" >
   <Grid x:Name="LayoutRoot" Background="White" Margin="3,3,10,3">
         <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
         </Grid.ColumnDefinitions>
         <TextBox  Grid.Row="0" Grid.Column="0" Margin="3" Height="Auto"
VerticalAlignment="Center"></TextBox>
         <Button Grid.Row="0" Grid.Column="1" Margin="3" Padding="2"
Content="Brwose"></Button>
      </Grid>
   </ScrollViewer>
```

Layout Painéis

ViewBox

```
<Viewbox>
  <Grid x:Name="LayoutRoot" Background="White" Width="200" Height="225"
            Margin="3,3,10,3">
        <Grid.RowDefinitions>

        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
           <ColumnDefinition Width="*"></ColumnDefinition>
           <ColumnDefinition Width="Auto"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <TextBox Grid.Row="0" Grid.Column="0" Margin="3" Height="auto"
             VerticalAlignment="Center"
              Text="Exemplo Texto"></TextBox>
        <Button Grid.Row="0" Grid.Column="1" Margin="3" Padding="2"
             Content="Browse"></Button>
     </Grid>
  </Viewbox>
```

Layout Painéis

ViewBox

Viewbox é um controle disponível em WPF. O mesmo controle é fornecido também no Silverlight. Viewbox define um decorador de conteúdo que pode se ajustar de acordo com a escala de um único filho para preencher o espaço disponível.

Layout Painéis

Full Screen

```
Application.Current.Host.Content.IsFullScreen = true;
```