Shapes e Geometries

José Antônio da Cunha

IFRN

Shapes e Geometries

Silverlight suporta um subconjuntos surpreendentemente grande de recursos de desenho. Você vai explorar o modelo de forma, os quais permite-lhe construir retângulos, elipses, linhas e curvas. Você também verá como você pode converter formas vetoriais existente para o formato XAML, as vezes você precisa reutiliza elementos gráficos existentes ao invés de construí-los de zero.

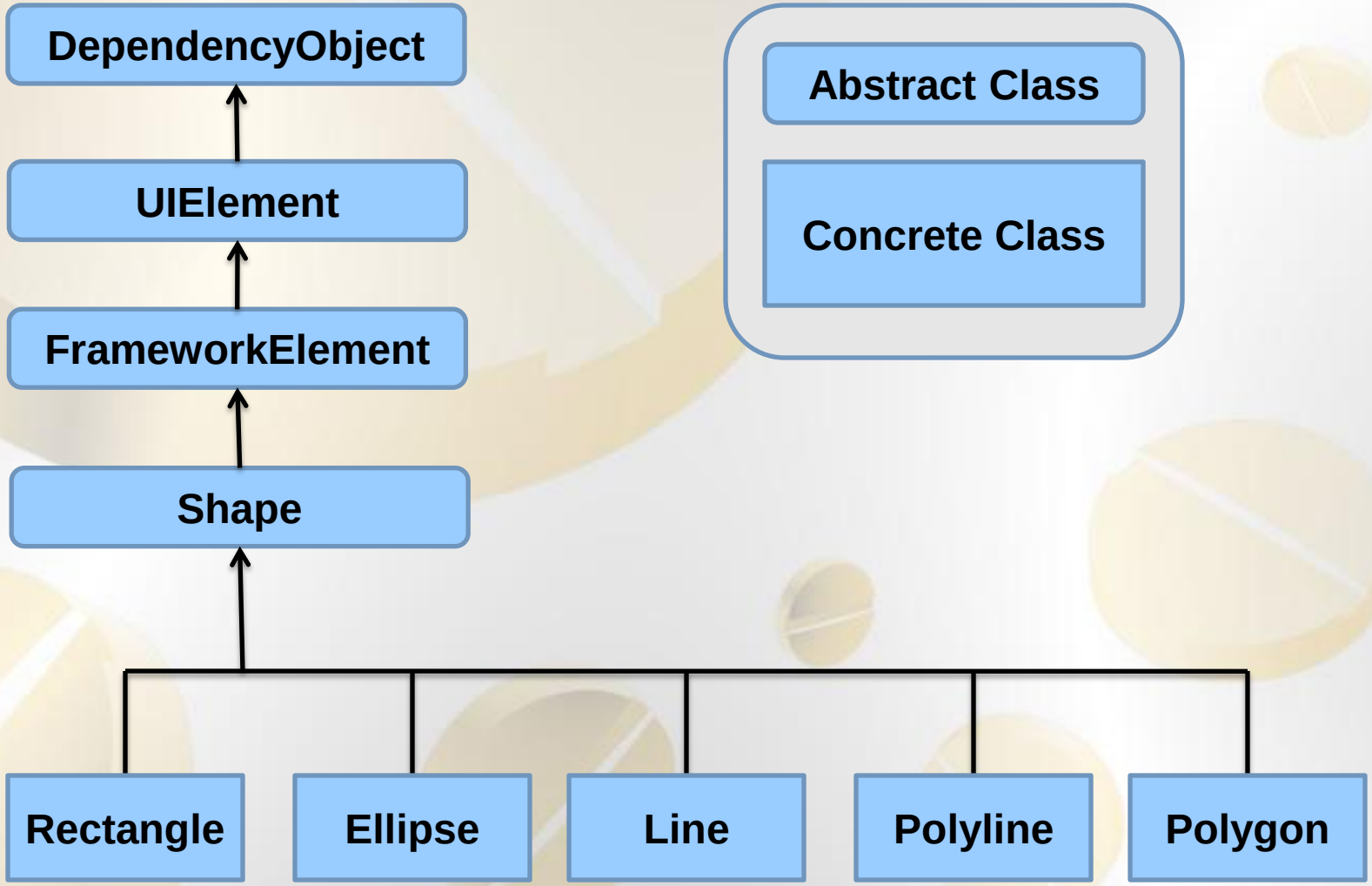
Shapes e Geometries

Formas básicas

A maneira mais simples de exibir gráfico 2-D em uma interface de usuário do Silverlight é a utilização de formas: classes dedicadas que representam simples linhas, elipses, retângulos e polígonos. Tecnicamente, as formas são conhecidas como primitivas de desenho. É possível combinar estes ingredientes básicos para criar gráficos mais complexos.

Shapes e Geometries

Formas básicas



Shapes e Geometries

Tabela 1: propriedades das formas

Propriedade	Descrição
Fill	Define o objeto pincel que pinta a superfície da forma (tudo dentro das suas fronteiras).
Stroke	Define o objeto pincel que pinta a borda da forma (a fronteira).
StrokeThickness	Define a espessura da borda, em pixels.
StrokeStartLineCap StrokeEndLineCap	Determinar o contorno da borda do início e fim da linha.
StrokeDashArray StrokeDashOffset StrokeDashCap	Permite-lhe criar uma borda tracejada ao redor de uma forma. Você pode controlar o tamanho e a freqüência dos traços.
StrokeLineJoin StrokeMiterLimit	Determinar o contorno dos cantos de uma forma.
Stretch	Determina como uma forma preencher o espaço disponível. Você pode usar essa propriedade para criar uma forma que se expande para ajustar seu recipiente.
GeometryTransform	Permite-lhe aplicar um objeto de transformação que muda o sistema de coordenadas usado para desenhar uma forma. Isso lhe permite inclinar, girar ou deslocar uma forma.

Shapes e Geometries

Retângulo e Ellipse

Retângulo e Elipse são duas formas mais simples. Para criar, defina as propriedades familiar, altura e Largura (herdado do FrameworkElement) para definir o tamanho da forma, e defina a propriedade Fill ou Stroke (ou ambos) para tornar a forma visível.

```
<StackPanel>
    <Ellipse Fill="Yellow" Stroke="Blue" Height="50" Width="100"
            HorizontalAlignment="Left"></Ellipse>
    <Rectangle Fill="Yellow" Stroke="Blue" Height="50" Width="100"
            Margin="5" HorizontalAlignment="Left"></Rectangle>
      </StackPanel>
```

Shapes e Geometries

Retângulo e Ellipse

```
<Canvas>
      <Ellipse Fill="Yellow" Stroke="Blue" Canvas.Left="100"
Canvas.Top="50"
               Width="100" Height="50"></Ellipse>
      <Rectangle Fill="Yellow" Stroke="Blue" Canvas.Left="30"
Canvas.Top="40"
                Width="100" Height="50"></Rectangle>
   </Canvas>
```

Shapes e Geometries

Retângulo e Ellipse

```
<Grid Margin="5">
      <Grid.RowDefinitions>
         <RowDefinition Height="Auto"></RowDefinition>
         <RowDefinition Height="*"></RowDefinition>
      </Grid.RowDefinitions>
      <TextBlock>A primeira linha da grade</TextBlock>
      <Viewbox Grid.Row="1" HorizontalAlignment="Left">
         <Canvas Width="200" Height="150">
            <Ellipse Fill="Yellow" Stroke="Blue" Canvas.Left="10" Canvas.Top="50"
                  Width="100" Height="50" HorizontalAlignment="Left"></Ellipse>
            <Rectangle Fill="Yellow" Stroke="Blue" Canvas.Left="30" Canvas.Top="40"
                   Width="100" Height="50" HorizontalAlignment="Left"></Rectangle>
         </Canvas>
      </Viewbox>
   </Grid>
```

Shapes e Geometries

Line

A linha representa uma linha reta que liga um ponto a outro. Os pontos inicial e final são definidos por quatro propriedades: X1 e Y1 (para o primeiro ponto) e X2 e Y2 (segundo). Por exemplo, aqui está uma linha que se estende de (0,0) a (10, 100).

```
<Line Stroke="Blue" X1="0" Y1="0" X2="10" Y2="100"></Line>
```

Shapes e Geometries

Polyline

A classe Polyline permite desenhar uma seqüência de linhas retas conectadas. Você fornece uma lista de coordenadas X e Y usando a propriedade Points.

```
<Polyline Stroke="Blue" Points="5 100 15 200"></Polyline>
```

```
<Polyline Stroke="Blue" Points="5,100 15, 200"></Polyline>
```

```
<Polyline Stroke="Red" StrokeThickness="5"
          Points="10,150 30,140 50,160 70,130 90,170 110,120 130,
                  180 150,110 170,190 190,100 210,240"></Polyline>
```

Shapes e Geometries

Polygon

Polígono é praticamente o mesmo que Polyline. A única diferença é que o Polígono adiciona um segmento de reta final, que conecta o ponto final para o ponto de partida.

```
<Grid x:Name="LayoutRoot" Background="White">
        <Polygon Stroke="Blue" StrokeThickness="5" Points="10,150 30,140
50,160 70,130,90,170 110,120 130,180 150,110 170,190 190,100 210,240"
Fill="Yellow"></Polygon>
    </Grid>
```

Abaixo, temos a estrela de cinco pontas.

```
<Canvas>
        <Polygon Stroke="Blue" StrokeThickness="1" Fill="Yellow"
Canvas.Left="10" Canvas.Top="175"
                 FillRule="Nonzero" Points="15,200 68,70 110,200 0,125
135,125"></Polygon>
    </Canvas>
```

Shapes e Geometries

Dashes

Em vez de desenhar linhas sólidas para as fronteiras de sua forma, você pode desenhar linhas tracejadas - linhas que são quebradas com os espaços de acordo com um padrão que você especificar.

```
<Grid x:Name="LayoutRoot" Background="White">
        <Polyline Stroke="Black" StrokeThickness="14" StrokeDashArray="1 2"
                Points="10,30 60,0 90,40 120,10 350,10"></Polyline>
        <Polyline Stroke="Black" StrokeThickness="14" StrokeDashArray="2 1"
                Points="10,30 60,0 90,40 120,10 350,10"></Polyline>
        <Polyline Stroke="Black" StrokeThickness="14" StrokeDashArray="5 0.2 3 0.2"
                Points="10,30 60,0 90,40 120,10 350,10"></Polyline>
    </Grid>
```

Shapes e Geometries

Paths e Geometries

Até agora, você já olhou para uma série de classes que derivar de Shape, incluindo Retângulo, Elipse, Linha, Polígono, e Polyline. No entanto, há uma outra categoria de classe muito mais potente. A classe **Path** tem a capacidade de abranger toda a forma simples, os grupos de formas, e os ingredientes mais complexos, tais como curvas.

A classe Path inclui uma única propriedade, de dados (Data), que aceita um objeto Geometry que define a forma (ou formas) incluindo o caminho. Você não pode criar um objeto Geometry diretamente porque é uma classe abstrata.Em vez disso, você precisa usar uma das classes derivadas listadas na Tabela 2.Todas essas classes são encontrados no **System.Windows.Media**.

Shapes e Geometries

Tabela 2. Classes Geometry

Nome	Descrição
LineGeometry	Representa uma linha reta. Equivale a forma Line.
RectangleGeometry	Representa uma retângulo. Equivale a forma Rectangle.
EllipseGeometry	Representa uma Elipse. Equivale a forma Ellipse.
GeometryGroup	Adiciona um número de objetos a um caminho, usando a regra EvenOdd ou Nonzero para determinar qual região será preenchida.
PathGeometry	Representa uma figura mais complexa que é composto de arcos, curvas e linhas, e pode ser aberta ou fechada.

Shapes e Geometries

Classes Geometry

```
<Grid x:Name="LayoutRoot" Background="White">
        <Rectangle Fill="Yellow" Stroke="Blue" Width="100" Height="50"></Rectangle>
        <Path Fill="Green" Stroke="Aqua" >
            <Path.Data>
                <RectangleGeometry Rect="0,0 100,50"></RectangleGeometry>
            </Path.Data>
        </Path>
        <Path Stroke="Black" >
            <Path.Data>
                <LineGeometry StartPoint="0,0" EndPoint="10,100"></LineGeometry>
            </Path.Data>
        </Path>
        <Path Stroke="Black" >
            <Path.Data>
                <EllipseGeometry RadiusX="50" RadiusY="25" Center="50,25"></EllipseGeometry>
            </Path.Data>
        </Path>
    </Grid>
```

Shapes e Geometries

Combinando formas com GeometryGroup

A maneira mais simples de combinar formas (figuras) é usar GeometryGroup para aninhar os objetos. Aqui está um exemplo de uma elipse ao lado de um quadrado:

```
<Canvas>
      <Path Fill="Yellow" Stroke="Blue" Margin="5" Canvas.Left="10" Canvas.Top="10">
         <Path.Data>
            <GeometryGroup>
               <RectangleGeometry Rect="0,0 100,100"></RectangleGeometry>
               <EllipseGeometry Center="50,50" RadiusX="35" RadiusY="25"></EllipseGeometry>
            </GeometryGroup>
         </Path.Data>
      </Path>
      <TextBlock Canvas.Left="20" Canvas.Top="50" FontSize="25" FontWeight="Bold" >Olá
Mundo!</TextBlock>
   </Canvas>
```

Shapes e Geometries

Curvas e linhas com PathGeometry

PathGeometry é um objeto para construir uma ou mais PathFigure. Um PathFigue é um continuo conjunto de linhas conectadas que podem ser aberta ou fechada.

Shapes e Geometries

Tabela 4. propriedades de PathFigure

StartPoint	Este é o ponto que indica onde a figura inicia.
Segments	Coleção de objetos PathSegment que são usados para desenhar a figura.
IsClosed	Se True, Silverlight adiciona linhas para conectar o ponto inicial e o ponto final.
IsFilled	Se true, a área dentro da figura é preenchida com a cor de Path.Fill.

Shapes e Geometries

Tabela 5. Classes de PathSegment

Nome	Descrição
LineSegment	Cria uma linha reta entre dois pontos.
ArcSegment	Cria um arco elípico entre dois pontos.
BezierSegment	Cria uma curva Bézier entre dois pontos.
QuadraticBezierSegmente	Cria uma curva Bézier que tem um ponto de controle, em vez de dois, e é mais rápido de calcular.
PolyLineSegment	Cria uma série de linhas retas. Você pode obter o mesmo efeito se você usar múltiplas linhas, mais um simples objeto PolyLineSegment é mais conciso.
PolyLineBezierSegment	Cria uma série de curvas Bézier.
PolyQuadraticBezierSegment	Cria uma série de simples curvas Bézier quadráticas.

Shapes e Geometries

Linhas

```
<Grid x:Name="LayoutRoot" Background="White">
      <Path Stroke="Blue">
         <Path.Data>
            <PathGeometry>
               <PathFigure IsClosed="True" StartPoint="10,100">
                  <LineSegment Point="100,100"/>
                  <LineSegment Point="100,50"/>
               </PathFigure>
            </PathGeometry>
         </Path.Data>
      </Path>
</Grid>
```

Shapes e Geometries

Arcos

```
<Grid x:Name="LayoutRoot" Background="White">
        <Path Stroke="Blue" StrokeThickness="3">
            <Path.Data>
                <PathGeometry>
                    <PathFigure IsClosed="False" StartPoint="10,100">
                        <ArcSegment Point="250,150" Size="200,300"/>
                    </PathFigure>
                </PathGeometry>
            </Path.Data>
        </Path>
    </Grid>
```

Shapes e Geometries

Curvas Bézier

```
<Canvas>
    <Path Stroke="Blue" StrokeThickness="5" Canvas.Top="20">
      <Path.Data>
        <PathGeometry>
          <PathFigure StartPoint="10,10">
            <BezierSegment Point1="130,30" Point2="40,140" Point3="150,150"></BezierSegment>
          </PathFigure>
        </PathGeometry>
      </Path.Data>
    </Path>
    <Path Stroke="Green" StrokeThickness="2" StrokeDashArray="5 2" Canvas.Top="20">
      <Path.Data>
        <GeometryGroup>
          <LineGeometry StartPoint="10,10" EndPoint="130,30"></LineGeometry>
          <LineGeometry StartPoint="40,140" EndPoint="150,150"></LineGeometry>
        </GeometryGroup>
      </Path.Data>
    </Path>
    <Path Fill="Red" Stroke="Red" StrokeThickness="8" Canvas.Top="20">
      <Path.Data>
        <GeometryGroup>
          <EllipseGeometry Center="130,30"></EllipseGeometry>
          <EllipseGeometry Center="40,140"></EllipseGeometry>
        </GeometryGroup>
      </Path.Data>
    </Path>
  </Canvas>
```

Shapes e Geometries

Geometry Mini-Language

As figuras geométricas vistas até o momento, são relativamente concisas, com poucos pontos. No entanto, figuras geométricas mais complexas, podem envolver centenas de segmentos. Definir cada segmento, arco e curva é uma tarefa complexa. O Silverlight adicionou uma forma alternativa mais concisa para definir figuras geométricas. Esta sintaxe é freqüentemente descrita **como geometry mini-language**.

Mini-language é essencialmente uma série de comandos em uma string. Cada comando é uma letra simples e opcionalmente é seguido por alguns bits numéricos (tais como coordenadas X e y) separados por espaços. Cada comando é também separado do comando anterior com um espaço.

Shapes e Geometries

Triângulo

```
<Path Stroke="Blue" >
        <Path.Data>
            <PathGeometry>
                <PathFigure IsClosed="True" StartPoint="10,100">
                    <LineSegment Point="100,100"/>
                    <LineSegment Point="100,50"/>
                </PathFigure>
            </PathGeometry>
        </Path.Data>
</Path>
```

Geometry Mini-Language

```
<Path Stroke="Blue" Data="M 10,100 L 100,100 L 100,50 Z"></Path>
```

```
<Path Stroke="Blue" Data="M10 100 L100 100 L100 50 Z"></Path>
```

Shapes e Geometries

Tabela 6. Comandos para Geometry Mini_Language

Comando	Descrição
F value	Configura a propriedade Geometry.FillRule. Usa 0 para EvenOdd ou 1 para Nonzero. Deve aparecer no início da string.
M x,y	Cria um nov elemento PathFigure e configura seus pontos iniciais.
L x,y	Cria um LineSegment para os específicos pontos.
H x	Cria um LineSegment horizontal.
V y	Cria um LineSegment vertical.
A radiusX, radiusY Degrees isLargArc, isClockwise x,y	Cria um ArcSegment indicado pelos pontos. Você especifica o raio, o grau e se é o maior arco da elipse.
C x1,y1 x2,y2 x,y	Cria um BezierSegment indicado pelos pontos.
Q x1,y1 x,y	Cria um QuadraticBezierSegment indicado pelos pontos.
S x2,y2 x,y	Cria um smooth BezierSegment.
T x2,y2 x,y	Cria um smooth QuadraticBezierSegment.
Z	Finaliza a corrente PathFigure.

Shapes e Geometries

Clipping com Geometry

Como você viu, as geometrias são a maneira mais poderosa para criar uma forma. No entanto, as geometrias não estão limitadas ao elemento Path.

A propriedade Clip permite restringir os limites exteriores de um elemento para atender a uma geometria específica. Você pode usar a propriedade Clip para criar uma série de efeitos exóticos. Você pode usar a propriedade clipe com qualquer elemento.

Shapes e Geometries

Clipping com Geometry

```
<Grid x:Name="LayoutRoot" Background="White">
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Button Content="A button">
            <Button.Clip>
                <GeometryGroup FillRule="Nonzero" >
                    <EllipseGeometry RadiusX="75" RadiusY="50" Center="100,150"/>
                    <EllipseGeometry RadiusX="100" RadiusY="25" Center="200,150"/>
                    <EllipseGeometry RadiusX="75" RadiusY="130" Center="140,140"/>
                </GeometryGroup>
            </Button.Clip>
        </Button>
        <Image Grid.Column="1" Stretch="None" Source="/koala.jpg">
            <Image.Clip>
                <GeometryGroup FillRule="Nonzero" >
                    <EllipseGeometry RadiusX="75" RadiusY="50" Center="100,150"/>
                    <EllipseGeometry RadiusX="100" RadiusY="25" Center="200,150"/>
                    <EllipseGeometry RadiusX="75" RadiusY="130" Center="140,140"/>
                </GeometryGroup>
            </Image.Clip>
        </Image>
    </Grid>
```