# Shapes and Brushes

## Shapes

<Window x:Class="Shapes.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Shapes" Height="350" Width="525">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

<StackPanel>

<!--Для того что бы фигура стала видимой нужно задать значение свойствам Stroke или Fill. По умолчанию заданы прозрачные кисти.-->

<Ellipse

Fill="Yellow"

Stroke="Blue"

Width="100"

Height="50"

Margin="5"

HorizontalAlignment="Left"></Ellipse>

<Rectangle

Fill="Yellow"

Stroke="Blue"

Width="100"

Height="50"

Margin="5"

HorizontalAlignment="Left"></Rectangle>

</StackPanel>

<!--Для фигур часто применяется Canvas так как позволяет задать координаты.-->

<Canvas Grid.Row="1">

<!--Если фигуры накладываются одна на одну, то верхней будет та фигура, которая в разметке указана последней-->

<Ellipse Canvas.Left="100" Canvas.Top="50"

Fill="Yellow" Stroke="Blue"

Width="100" Height="50" ></Ellipse>

<!--Будет над эллипсом-->

<Rectangle Canvas.Left="30" Canvas.Top="40"

Fill="Yellow" Stroke="Blue"

Width="100" Height="50" ></Rectangle>

</Canvas>

</Grid>

</Window>

## Rounded Rectangles

<Window x:Class="RoundetRectangles.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Rounded Rectangles" Height="500" Width="250">

<StackPanel>

<!--Для задания округлых углов прямоугольника - RadiusX и RadiusY-->

<TextBlock Margin="5,5,0,0">Радиус углов 5.</TextBlock>

<Rectangle RadiusX="5" RadiusY="5"

Fill="Yellow" Stroke="Blue"

Width="100" Height="60"

Margin="5" HorizontalAlignment="Left">

</Rectangle>

<TextBlock Margin="5,5,0,0">Радиус углов 10.</TextBlock>

<Rectangle RadiusX="10" RadiusY="10"

Fill="Yellow" Stroke="Blue"

Width="100" Height="60"

Margin="5" HorizontalAlignment="Left">

</Rectangle>

<TextBlock Margin="5,5,0,0">Радиус углов 10 (X) и 25 (Y).</TextBlock>

<Rectangle RadiusX="10" RadiusY="25"

Fill="Yellow" Stroke="Blue"

Width="100" Height="60"

Margin="5" HorizontalAlignment="Left">

</Rectangle>

<TextBlock Margin="5,5,0,0">Радиус углов 100 (X) и 60 (Y).</TextBlock>

<Rectangle RadiusX="50" RadiusY="30"

Fill="Yellow" Stroke="Blue"

Width="100" Height="60"

Margin="5" HorizontalAlignment="Left"></Rectangle>

</StackPanel>

</Window>

## ViewBox

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

<TextBlock>Первая строка таблицы</TextBlock>

<!--ViewBox для пропорционального изменения содержимого при масштабировании-->

<Viewbox Grid.Row="1"

HorizontalAlignment="Left"

MaxHeight="500">

<!--Если не задать размеры вложенного во Viewbox элемента, viewbox будет невидимым-->

<Canvas Width="200" Height="150">

<Ellipse Canvas.Left="10" Canvas.Top="50"

Fill="Yellow" Stroke="Blue"

Width="100" Height="50"

HorizontalAlignment="Left">

</Ellipse>

<Rectangle Canvas.Left="30" Canvas.Top="40"

Fill="Yellow" Stroke="Blue"

Width="100" Height="50"

HorizontalAlignment="Left">

</Rectangle>

</Canvas>

</Viewbox>

</Grid>

## FillMode

<Grid ShowGridLines="True" Margin="5">

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<!--Fill - заполняет контейнер, растягиваясь по высоте и ширине-->

<Ellipse Stretch="Fill"

Fill="Yellow" Stroke="Blue"></Ellipse>

<!--Uniform – ширина и высота увеличиваются пропорционально.-->

<Ellipse Stretch="Uniform"

Fill="Yellow" Stroke="Blue" Grid.Column="1"></Ellipse>

<!--UniformToFill - Ширина и высота увеличиваются пропорционально пока фигура не заполнит весь контейнер.-->

<Ellipse Stretch="UniformToFill"

Fill="Yellow" Stroke="Blue" Grid.Column="2"></Ellipse>

<TextBlock Grid.Row="1" TextAlignment="Center">Fill</TextBlock>

<TextBlock Grid.Row="1" Grid.Column="1" TextAlignment="Center">Uniform</TextBlock>

<TextBlock Grid.Row="1" Grid.Column="2" TextAlignment="Center">UniformToFill</TextBlock>

</Grid>

## Polygons

<Canvas>

<!--Координаты задаются попарно X,Y -->

<Polyline

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" >

</Polyline>

<!--Последняя точка всегда соединяется с первой точкой.-->

<Polygon

Stroke="Blue"

StrokeThickness="5"

Fill="Yellow"

Canvas.Top="200"

Points="10,150 30,140 50,160 70,130 90,170 110,120 130,180 150,110 170,190 190,100 210,240" >

</Polygon>

</Canvas>

<Canvas Width="200" Height="400" >

<!--По умолчанию правило, по которому заполняется полигон цветом (FillRule) - EvenOdd-->

<Polygon Canvas.Left="10"

FillRule="EvenOdd"

Stroke="Blue"

StrokeThickness="1"

Fill="Yellow"

Points="15,200 68,70 110,200 0,125 135,125" >

</Polygon>

<Polygon Canvas.Left="10" Canvas.Top="175"

Stroke="Blue"

StrokeThickness="1"

Fill="Yellow"

FillRule="Nonzero"

Points="15,200 68,70 110,200 0,125 135,125" >

</Polygon>

</Canvas>

<Grid Margin="7">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

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

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<TextBlock VerticalAlignment="Center">Not Snapped:</TextBlock>

<Rectangle SnapsToDevicePixels="False"

Grid.Column="1"

Margin="10" Height="10" Fill="Red"></Rectangle>

<TextBlock VerticalAlignment="Center" Grid.Row="1">Snapped:</TextBlock>

<Rectangle

Grid.Column="1" Grid.Row="1"

Margin="10" Height="10" Fill="Red"></Rectangle>

</Grid>

## Gradients

<Grid Margin="5">

<!--Линейный градиент относится к кистям (Brush) его можно применить на любом элементе.-->

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

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

</Grid.ColumnDefinitions>

<Rectangle Width="150" Height="100" Margin="5">

<Rectangle.Fill>

<LinearGradientBrush>

<GradientStop Color="Blue" Offset="0"/>

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Grid.Column="1" VerticalAlignment="Center" Margin="5">Линейный градиент по диагонали</TextBlock>

<Rectangle Width="150" Height="100" Margin="5" Grid.Row="1">

<Rectangle.Fill>

<LinearGradientBrush>

<GradientStop Color="Blue" Offset="0"/>

<GradientStop Color="White" Offset="0.5" />

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" Margin="5">Значение Offset = 0.5 для белого цвета</TextBlock>

<Rectangle Width="150" Height="100" Grid.Row="2" Margin="5">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="0,0" EndPoint="0,1" >

<GradientStop Color="Blue" Offset="0"/>

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Grid.Row="2" Grid.Column="1" VerticalAlignment="Center" Margin="5">Горизонтальный линейный градиент</TextBlock>

<Rectangle Width="150" Height="100" Grid.Row="3" Margin="5">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="0,0" EndPoint="0,0.5" SpreadMethod="Reflect">

<GradientStop Color="Blue" Offset="0"/>

<GradientStop Color="White" Offset="1" />

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Grid.Row="3" Grid.Column="1" VerticalAlignment="Center" Margin="5">SpreadMethod="Reflect"</TextBlock>

<Rectangle Width="150" Height="100" Grid.Row="4" Margin="5">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="Yellow" Offset="0.0" />

<GradientStop Color="Red" Offset="0.25" />

<GradientStop Color="Blue" Offset="0.75" />

<GradientStop Color="LimeGreen" Offset="1.0" />

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<TextBlock Grid.Row="4" Grid.Column="1" VerticalAlignment="Center" Margin="5">Многоцветный градиент</TextBlock>

</Grid>

<Grid>

<TextBox Margin="5" FontWeight="Bold" FontSize="65" TextWrapping="Wrap" TextAlignment="Center">

<TextBox.Text>Это текст с градиентом.</TextBox.Text>

<!--Градиент в качестве Foreground-->

<TextBox.Foreground>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="Yellow" Offset="0.0" />

<GradientStop Color="Red" Offset="0.25" />

<GradientStop Color="Blue" Offset="0.75" />

<GradientStop Color="LimeGreen" Offset="1.0" />

</LinearGradientBrush>

</TextBox.Foreground>

</TextBox>

</Grid>

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

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

</Grid.ColumnDefinitions>

<Ellipse Margin="5" Stroke="Black" StrokeThickness="1">

<Ellipse.Fill>

<!--Настройки градиента-->

<RadialGradientBrush RadiusX="1" RadiusY="1" >

<GradientStop Color="White" Offset="0" />

<GradientStop Color="Blue" Offset="1" />

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

<TextBlock Grid.Column="1" VerticalAlignment="Center" Margin="5">Радиальный градиент.</TextBlock>

<Ellipse Margin="5" Grid.Row="1" Stroke="Black" StrokeThickness="1">

<!--GradientOrigin - центр распространения градиента.-->

<Ellipse.Fill>

<RadialGradientBrush RadiusX="1" RadiusY="1" GradientOrigin="0.7,0.3">

<GradientStop Color="White" Offset="0" />

<GradientStop Color="Blue" Offset="1" />

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

<TextBlock Grid.Row="1" Grid.Column="1" VerticalAlignment="Center" Margin="5" TextWrapping="Wrap">Радиальный градиент со смещенным центром.</TextBlock>

</Grid>

## Tranform

<Canvas>

<Rectangle Canvas.Left="100" Canvas.Top="100"

Width="80" Height="10"

Stroke="Blue" Fill="Yellow">

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="100">

<Rectangle.RenderTransform>

<RotateTransform Angle="25" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Canvas.Left="100" Canvas.Top="100"

Width="80" Height="10"

Stroke="Blue" Fill="Yellow" >

<Rectangle.RenderTransform>

<RotateTransform Angle="50" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="100">

<Rectangle.RenderTransform>

<RotateTransform Angle="75" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="100">

<Rectangle.RenderTransform>

<RotateTransform Angle="100" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="300">

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="300">

<Rectangle.RenderTransform>

<RotateTransform Angle="25"

CenterX="40"

CenterY="5" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="300">

<Rectangle.RenderTransform>

<RotateTransform Angle="50"

CenterX="45"

CenterY="5" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10"

Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="300">

<Rectangle.RenderTransform>

<RotateTransform Angle="75"

CenterX="45"

CenterY="5" />

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="80" Height="10" Stroke="Blue" Fill="Yellow"

Canvas.Left="100" Canvas.Top="300">

<Rectangle.RenderTransform>

<RotateTransform Angle="100"

CenterX="45"

CenterY="5" />

</Rectangle.RenderTransform>

</Rectangle>

</Canvas>

<StackPanel>

<StackPanel Margin="25" Background="LightYellow">

<Button Padding="5" HorizontalAlignment="Left">

<!--RenderTransform - меняет отображение фигуры-->

<Button.RenderTransform>

<RotateTransform Angle="35"

CenterX="45"

CenterY="5" />

</Button.RenderTransform>

<Button.Content>Под углом 35 градусов</Button.Content>

</Button>

<Button Padding="5" HorizontalAlignment="Left">Кнопка</Button>

</StackPanel>

<StackPanel Margin="25" Background="LightYellow">

<Button Padding="5" HorizontalAlignment="Left">

<!--LayoutTransform - меняет отображение фигуры на этапе компоновки.-->

<Button.LayoutTransform>

<RotateTransform Angle="35"

CenterX="45"

CenterY="5" />

</Button.LayoutTransform>

<Button.Content>Под углом 35 градусов</Button.Content>

</Button>

<Button Padding="5" HorizontalAlignment="Left">Кнопка</Button>

</StackPanel>

</StackPanel>

## VisualBrush

<StackPanel Margin="3">

<Button Name="cmd" Margin="3" Padding="5">Кнопка ?</Button>

<Rectangle Margin="3" Height="{Binding ElementName=cmd, Path=ActualHeight}"

RenderTransformOrigin="0 0.5">

<Rectangle.Fill>

<!--VisualBrush - в качестве заливки фона прямоугольника.-->

<VisualBrush Visual="{Binding ElementName=cmd}"></VisualBrush>

</Rectangle.Fill>

</Rectangle>

<Rectangle Margin="3" Height="50">

<Rectangle.Fill>

<VisualBrush Visual="{Binding ElementName=cmd}"></VisualBrush>

</Rectangle.Fill>

</Rectangle>

<Rectangle Margin="3" Height="150">

<Rectangle.Fill>

<VisualBrush Visual="{Binding ElementName=cmd}"></VisualBrush>

</Rectangle.Fill>

</Rectangle>

</StackPanel>

## OpacityMask

<Grid Margin="10,50">

<Button Background="Purple" FontSize="14" FontWeight="Bold">

<!--OpacityMask свойство позволяет сделать элемент частично прозрачным. Цвет не имеет значения при указании OpacityMask-->

<Button.OpacityMask>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,0">

<GradientStop Offset="0" Color="Yellow"></GradientStop>

<GradientStop Offset="0.8" Color="Transparent"></GradientStop>

</LinearGradientBrush>

</Button.OpacityMask>

<Button.Content>Частично прозрачная кнопка</Button.Content>

</Button>

</Grid>

## ReflactionEffect

<Grid Margin="10" VerticalAlignment="Center">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<TextBox Name="txt" FontSize="30">Тестовая строка</TextBox>

<Rectangle Grid.Row="1" Height="50" RenderTransformOrigin="0, 0.5">

<Rectangle.Fill>

<VisualBrush Visual="{Binding ElementName=txt}"></VisualBrush>

</Rectangle.Fill>

<Rectangle.OpacityMask>

<LinearGradientBrush StartPoint="0,0" EndPoint="0,1">

<GradientStop Offset="0.3" Color="Transparent"></GradientStop>

<GradientStop Offset="1" Color="#44000000"></GradientStop>

</LinearGradientBrush>

</Rectangle.OpacityMask>

<Rectangle.RenderTransform>

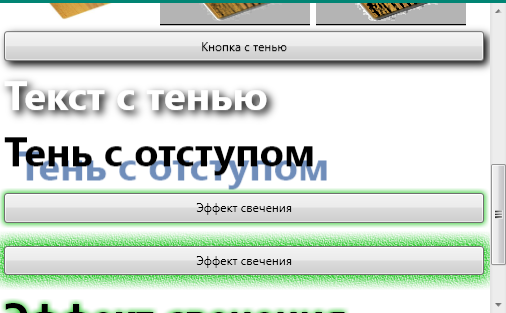
<ScaleTransform ScaleY="-1"></ScaleTransform>

</Rectangle.RenderTransform>

</Rectangle>

</Grid>

## BitmapEffect



<ScrollViewer>

<StackPanel Margin="3">

<StackPanel.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Padding" Value="5"></Setter>

<Setter Property="Margin" Value="3"></Setter>

</Style>

</StackPanel.Resources>

<!--BlurBitmapEffect-->

<Button>Обычная кнопка</Button>

<Button>

<Button.Content>Обычная кнопка с размытием</Button.Content>

<Button.BitmapEffect>

<BlurBitmapEffect Radius="1"></BlurBitmapEffect>

</Button.BitmapEffect>

</Button>

<Button>

<Button.Content>Обычная кнопка и сильное размытие</Button.Content>

<Button.BitmapEffect>

<BlurBitmapEffect Radius="5"></BlurBitmapEffect>

</Button.BitmapEffect>

</Button>

<!--BevelBitmapEffect-->

<Button>

<Button.Content>Beveled эффект</Button.Content>

<Button.BitmapEffect>

<BevelBitmapEffect></BevelBitmapEffect>

</Button.BitmapEffect>

</Button>

<TextBlock FontSize="80" FontWeight="Bold" Foreground="Gray">

<TextBlock.Text>Beveled текст</TextBlock.Text>

<TextBlock.BitmapEffect>

<BevelBitmapEffect></BevelBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

<!--EmbossBitmapEffect-->

<TextBlock FontSize="40" FontWeight="Bold" Foreground="Gray">

<TextBlock.Text>Embossed эффект</TextBlock.Text>

<TextBlock.BitmapEffect>

<EmbossBitmapEffect></EmbossBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

<WrapPanel>

<Image Margin="3" Source="cube.jpg" Width="150"></Image>

<Image Margin="3" Source="cube.jpg" Width="150">

<Image.BitmapEffect>

<EmbossBitmapEffect Relief="0.9"></EmbossBitmapEffect>

</Image.BitmapEffect>

</Image>

<Image Margin="3" Source="cube.jpg" Width="150">

<Image.BitmapEffect>

<EmbossBitmapEffect Relief="0.5"></EmbossBitmapEffect>

</Image.BitmapEffect>

</Image>

</WrapPanel>

<!--DropShadowBitmapEffect-->

<Button>

<Button.Content>Кнопка с тенью</Button.Content>

<Button.BitmapEffect>

<DropShadowBitmapEffect></DropShadowBitmapEffect>

</Button.BitmapEffect>

</Button>

<TextBlock Margin="3" FontSize="40" FontWeight="Bold" Foreground="White">

<TextBlock.Text>Текст с тенью</TextBlock.Text>

<TextBlock.BitmapEffect>

<DropShadowBitmapEffect></DropShadowBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

<TextBlock Margin="3,0,3,13" FontSize="40" FontWeight="Bold">

<TextBlock.Text>Тень с отступом</TextBlock.Text>

<TextBlock.BitmapEffect>

<DropShadowBitmapEffect ShadowDepth="20" Color="LightSteelBlue" Softness="0"></DropShadowBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

<!--OuterGlowBitmapEffect-->

<Button>

<Button.Content>Эффект свечения</Button.Content>

<Button.BitmapEffect>

<OuterGlowBitmapEffect GlowColor="LimeGreen"></OuterGlowBitmapEffect>

</Button.BitmapEffect>

</Button>

<Button Margin="3,20,3,10">

<Button.Content>Эффект свечения</Button.Content>

<Button.BitmapEffect>

<OuterGlowBitmapEffect GlowColor="LimeGreen" GlowSize="15" Noise="0.5"></OuterGlowBitmapEffect>

</Button.BitmapEffect>

</Button>

<TextBlock FontSize="40" FontWeight="Bold" Margin="3">

<TextBlock.Text>Эффект свечения</TextBlock.Text>

<TextBlock.BitmapEffect>

<OuterGlowBitmapEffect GlowColor="LimeGreen"></OuterGlowBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

<TextBlock Margin="3" FontSize="40" FontWeight="Bold" Foreground="Blue">

<TextBlock.Text>Эффект свечения</TextBlock.Text>

<TextBlock.BitmapEffect>

<OuterGlowBitmapEffect GlowColor="LimeGreen" GlowSize="20" Noise="0.5"></OuterGlowBitmapEffect>

</TextBlock.BitmapEffect>

</TextBlock>

</StackPanel>

</ScrollViewer>

# Geometry

## GeometryCombine

<Window.Resources>

<RectangleGeometry x:Key="rect"

Rect="0 0 100 100"></RectangleGeometry>

<EllipseGeometry x:Key="ellipse"

Center="85 50"

RadiusX="65"

RadiusY="35"></EllipseGeometry>

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="10"></Setter>

</Style>

</Window.Resources>

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="198\*" />

<ColumnDefinition Width="212\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="120" />

<RowDefinition Height="120" />

<RowDefinition Height="120" />

<RowDefinition Height="120" />

</Grid.RowDefinitions>

<!--Union-->

<Path Grid.Row="0" Grid.Column="0" Fill="Yellow" Stroke="Black">

<Path.Data>

<CombinedGeometry GeometryCombineMode="Union"

Geometry1="{StaticResource rect}"

Geometry2="{StaticResource ellipse}">

</CombinedGeometry>

</Path.Data>

</Path>

<TextBlock Grid.Row="0" Grid.Column="1">Union</TextBlock>

<!--Intersect-->

<Path Grid.Row="1" Grid.Column="0" Fill="Yellow" Stroke="Black">

<Path.Data>

<CombinedGeometry GeometryCombineMode="Intersect"

Geometry1="{StaticResource rect}"

Geometry2="{StaticResource ellipse}">

</CombinedGeometry>

</Path.Data>

</Path>

<TextBlock Grid.Row="1" Grid.Column="1">Intersect</TextBlock>

<!--Xor-->

<Path Grid.Row="2" Grid.Column="0" Fill="Yellow" Stroke="Black">

<Path.Data>

<CombinedGeometry GeometryCombineMode="Xor"

Geometry1="{StaticResource rect}"

Geometry2="{StaticResource ellipse}">

</CombinedGeometry>

</Path.Data>

</Path>

<TextBlock Grid.Row="2" Grid.Column="1">Xor</TextBlock>

<!--Exclude-->

<Path Grid.Row="3" Grid.Column="0" Fill="Yellow" Stroke="Black">

<Path.Data>

<CombinedGeometry GeometryCombineMode="Exclude"

Geometry1="{StaticResource rect}"

Geometry2="{StaticResource ellipse}">

</CombinedGeometry>

</Path.Data>

</Path>

<TextBlock Grid.Row="3" Grid.Column="1">Exclude</TextBlock>

</Grid>

## Lines

<Grid>

<Path Stroke="Black">

<Path.Data>

<PathGeometry>

<PathFigure IsClosed="True" StartPoint="10, 100">

<LineSegment Point="100, 100"></LineSegment>

<LineSegment Point="100, 50"></LineSegment>

</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

</Grid>

## MiniLanguageGeometry

<Grid>

<Path Stroke="Blue">

<Path.Data>

<PathGeometry>

<PathFigure IsClosed="True" StartPoint="10, 100">

<LineSegment Point="100, 100"></LineSegment>

<LineSegment Point="100, 50"></LineSegment>

</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

<!--Фигура описана с помощью мини языка-->

<Path Stroke="Blue" Data="M 10, 250 H 100 L 100, 200 Z"></Path>

<!--

F значение - Устанавливает свойство Geometry.FillRule. 0 - EvenOdd 1 - NonZero

M x,y - создает новый объект PathFigure и устанавливает его начальную точку. Эта команда должна использоватся перед

любой другой за исключением F.

L x,y - LineSegment

H x - Горизонтальный LineSegment используется указанное значение X и неизменное Y

V y - Вертикальный LineSegment

A radiusX, radiusY degrees isLargeArc, isClockwise x,y - Создает объект ArcSegment

Z - Завершает текущую PathFigure и устанавливает IsClosed=true

-->

</Grid>

## ButtonClip

<Grid>

<Button Click="Button\_Click" Content="Click me!" Height="100" Width="100" BorderThickness="0">

<Button.Clip>

<!--CombinedGeometry первая комбинация, которая соединит прямоугольник и окружность-->

<CombinedGeometry GeometryCombineMode="Union">

<!--В качестве Geometry1 используется еще одна комбинация сплайнов-->

<CombinedGeometry.Geometry1>

<CombinedGeometry GeometryCombineMode="Exclude">

<CombinedGeometry.Geometry1>

<EllipseGeometry Center="50, 50" RadiusX="50" RadiusY="50"></EllipseGeometry>

</CombinedGeometry.Geometry1>

<CombinedGeometry.Geometry2>

<EllipseGeometry Center="50, 50" RadiusX="40" RadiusY="40"></EllipseGeometry>

</CombinedGeometry.Geometry2>

</CombinedGeometry>

</CombinedGeometry.Geometry1>

<CombinedGeometry.Geometry2>

<RectangleGeometry Rect="5, 40 90, 20">

</RectangleGeometry>

</CombinedGeometry.Geometry2>

</CombinedGeometry>

</Button.Clip>

</Button>

</Grid>

## DrawingContext

Grid>

<local:TestDrawing Height="50" Width="50"></local:TestDrawing>

</Grid>

class TestDrawing : Control

{

// DrawingContext - Описывает визуальное содержимое, имеет методы для рисования.

protected override void OnRender(System.Windows.Media.DrawingContext drawingContext)

{

drawingContext.DrawLine(new Pen(Brushes.Red, 2),

new Point(0, 0),

new Point(this.Height, this.Width));

drawingContext.DrawLine(new Pen(Brushes.Red, 2),

new Point(0, this.Width),

new Point(this.Height, 0));

base.OnRender(drawingContext);

}

}

## ClassVisual

<Grid>

<Grid.ColumnDefinitions>

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

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<!--Вертикальный toolbar-->

<ToolBarTray Orientation="Vertical">

<ToolBar>

<RadioButton Margin="3" Name="cmdSelectMove">

<StackPanel>

<Image Source="pointer.png" Width="35" Height="35"></Image>

<TextBlock>Выбрать/Двигать</TextBlock>

</StackPanel>

</RadioButton>

<RadioButton Margin="3" IsChecked="True" Name="cmdAdd">

<StackPanel>

<Rectangle Width="30"

Height="30"

Stroke="SteelBlue"

StrokeThickness="3"

Fill="AliceBlue"></Rectangle>

<TextBlock>Добавить квадрат</TextBlock>

</StackPanel>

</RadioButton>

<RadioButton Margin="3" Name="cmdDelete">

<StackPanel>

<Path Stroke="SteelBlue" StrokeThickness="4"

StrokeEndLineCap="Round" StrokeStartLineCap="Round"

Fill="Red" HorizontalAlignment="Center">

<Path.Data>

<GeometryGroup>

<PathGeometry>

<PathFigure StartPoint="0,0">

<LineSegment Point="18,18"></LineSegment>

</PathFigure>

<PathFigure StartPoint="0,18">

<LineSegment Point="18,0"></LineSegment>

</PathFigure>

</PathGeometry>

</GeometryGroup>

</Path.Data>

</Path>

<TextBlock>Удалить квадрат</TextBlock>

</StackPanel>

</RadioButton>

<RadioButton Margin="0,3" Name="cmdSelectMultiple">

<StackPanel>

<Image Source="pointer.png" Width="35" Height="35"></Image>

<TextBlock>Выбрать несколько</TextBlock>

</StackPanel>

</RadioButton>

</ToolBar>

</ToolBarTray>

<Border Grid.Column="1" Margin="3" BorderBrush="SteelBlue" BorderThickness="1">

<!--Панель, на которой отображаются квадраты-->

<local:DrawingCanvas x:Name="drawingSurface"

Background="White"

ClipToBounds="True"

MouseLeftButtonDown="drawingSurface\_MouseLeftButtonDown"

MouseLeftButtonUp="drawingSurface\_MouseLeftButtonUp"

MouseMove="drawingSurface\_MouseMove">

</local:DrawingCanvas>

</Border>

</Grid>

public partial class MainWindow : Window

{

private bool isDragging = false;

private Vector clickOffset;

// DrawingVisual - класс позволяет рисовать графическое содержимое.

private DrawingVisual selectedVisual;

public MainWindow()

{

InitializeComponent();

}

// Переменные для отображения рамки выбора прямоугольников.

private bool isMultiSelecting = false; // Выбор нескольких прямоугольников.

private Point selectionSquareTopLeft; // Координаты прямоугольника, который рисуется при выборе нескольких квадратов.

// Переменные для рисования.

private Brush drawingBrush = Brushes.AliceBlue;

private Brush selectedDrawingBrush = Brushes.LightGoldenrodYellow;

private Pen drawingPen = new Pen(Brushes.SteelBlue, 3);

private Size squareSize = new Size(30, 30);

private DrawingVisual selectionSquare;

// Отображение прямоугольника.

private void DrawSquare(DrawingVisual visual, Point topLeftCorner, bool isSelected)

{

// Для начала рисования следует вызвать метод RenderOpen, который вернет объект DrawingContext.

// По окончанию рисования нужно вызвать метод Close()

using (DrawingContext dc = visual.RenderOpen())

{

Brush brush = drawingBrush;

if (isSelected) brush = selectedDrawingBrush;

dc.DrawRectangle(brush, drawingPen,

new Rect(topLeftCorner, squareSize));

}

}

private Brush selectionSquareBrush = Brushes.Transparent;

private Pen selectionSquarePen = new Pen(Brushes.Black, 2);

private void DrawSelectionSquare(Point point1, Point point2)

{

selectionSquarePen.DashStyle = DashStyles.Dash;

using (DrawingContext dc = selectionSquare.RenderOpen())

{

dc.DrawRectangle(selectionSquareBrush, selectionSquarePen,

new Rect(point1, point2));

}

}

// Обработчик на нажатие левой клавишей мыши по поверхности, на которой отображаются квадраты.

private void drawingSurface\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

// Получение координат мыши.

Point pointClicked = e.GetPosition(drawingSurface);

// Если выбран RadioButton "Добавить квадрат"

if (cmdAdd.IsChecked == true)

{

DrawingVisual visual = new DrawingVisual();

DrawSquare(visual, pointClicked, false); // Рисуем новый квадрат.

drawingSurface.AddVisual(visual); // Добавляем на поверхность объект DrawingVisual

}

// Если выбран RadioButton "Удалить"

else if (cmdDelete.IsChecked == true)

{

DrawingVisual visual = drawingSurface.GetVisual(pointClicked); // Получаем объект по указанным координатам.

if (visual != null) drawingSurface.DeleteVisual(visual);

}

// Если выбран RadioButton "Выбрать/Двигать"

else if (cmdSelectMove.IsChecked == true)

{

DrawingVisual visual = drawingSurface.GetVisual(pointClicked); // Получаем объект по указанным координатам.

if (visual != null)

{

// Подсчет координат верхнего левого угла квадрата

Point topLeftCorner = new Point(

visual.ContentBounds.TopLeft.X + drawingPen.Thickness / 2,

visual.ContentBounds.TopLeft.Y + drawingPen.Thickness / 2);

DrawSquare(visual, topLeftCorner, true);

// Получение смещения координат.

clickOffset = topLeftCorner - pointClicked;

isDragging = true;

if (selectedVisual != null && selectedVisual != visual)

{

// Очистка предыдущего выбора.

ClearSelection();

}

selectedVisual = visual;

}

}

// Если выбран RadioButton "Выбрать несколько"

else if (cmdSelectMultiple.IsChecked == true)

{

selectionSquare = new DrawingVisual();

drawingSurface.AddVisual(selectionSquare);

selectionSquareTopLeft = pointClicked;

isMultiSelecting = true;

// Для того, чтобы получить событие MouseLeftButtonUp даже если пользователь выведет

// мышь за пределы Canvas, делаем захват мыши.

drawingSurface.CaptureMouse();

}

}

private void drawingSurface\_MouseMove(object sender, MouseEventArgs e)

{

if (isDragging)

{

Point pointDragged = e.GetPosition(drawingSurface) + clickOffset;

DrawSquare(selectedVisual, pointDragged, true);

}

else if (isMultiSelecting)

{

Point pointDragged = e.GetPosition(drawingSurface);

DrawSelectionSquare(selectionSquareTopLeft, pointDragged);

}

}

private void drawingSurface\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

isDragging = false;

if (isMultiSelecting)

{

RectangleGeometry geometry = new RectangleGeometry(new Rect(selectionSquareTopLeft, e.GetPosition(drawingSurface)));

List<DrawingVisual> visualsInRegion = drawingSurface.GetVisuals(geometry); // Получаем все элементы которые находятся в прямоугольной области.

MessageBox.Show(String.Format("You selected {0} square(s).", visualsInRegion.Count));

isMultiSelecting = false;

drawingSurface.DeleteVisual(selectionSquare); // Удаляем рамку выделения.

drawingSurface.ReleaseMouseCapture();

}

}

private void ClearSelection()

{

Point topLeftCorner = new Point(

selectedVisual.ContentBounds.TopLeft.X + drawingPen.Thickness / 2,

selectedVisual.ContentBounds.TopLeft.Y + drawingPen.Thickness / 2);

DrawSquare(selectedVisual, topLeftCorner, false);

selectedVisual = null;

}

}

public class DrawingCanvas : Panel

{

// Коллекция уже нарисованных квадратов.

private List<Visual> visuals = new List<Visual>();

// Панель будет использовать этот метод для доступа к каждому элементу, который надо визуализировать.

protected override Visual GetVisualChild(int index)

{

return visuals[index];

}

// Получение общего количества элементов для визуализации.

protected override int VisualChildrenCount

{

get

{

return visuals.Count;

}

}

public void AddVisual(Visual visual)

{

visuals.Add(visual);

// Эти методы нужны для того, что бы корректно работали функции взаимодействия элементов,

// например, такие как проверка попадания курсора в элемент.

base.AddVisualChild(visual);

base.AddLogicalChild(visual);

}

public void DeleteVisual(Visual visual)

{

visuals.Remove(visual);

base.RemoveVisualChild(visual);

base.RemoveLogicalChild(visual);

}

// Метод для проверки попадания.

public DrawingVisual GetVisual(Point point)

{

HitTestResult hitResult = VisualTreeHelper.HitTest(this, point);

return hitResult.VisualHit as DrawingVisual;

}

private List<DrawingVisual> hits = new List<DrawingVisual>();

public List<DrawingVisual> GetVisuals(Geometry region)

{

hits.Clear(); // Очищение коллекции результатов проверки попадания.

GeometryHitTestParameters parameters = new GeometryHitTestParameters(region);

HitTestResultCallback callback = new HitTestResultCallback(this.HitTestCallback); // Метод обратного вызова.

VisualTreeHelper.HitTest(this, null, callback, parameters);

return hits;

}

// Метод, который будет находить элементы попавшие в регион.

private HitTestResultBehavior HitTestCallback(HitTestResult result)

{

GeometryHitTestResult geometryResult = (GeometryHitTestResult)result;

DrawingVisual visual = result.VisualHit as DrawingVisual;

if (visual != null &&

geometryResult.IntersectionDetail == IntersectionDetail.FullyInside)

{

hits.Add(visual);

}

return HitTestResultBehavior.Continue;

}

}

# Templates

## ControlTypes

<Grid Margin="10" Name="grid">

<Grid.ColumnDefinitions>

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

<ColumnDefinition Width="3\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<ListBox DisplayMemberPath="Name"

Name="lstTypes"

SelectionChanged="lstTypes\_SelectionChanged">

</ListBox>

<TextBox Grid.Column="1"

Name="txtTemplate"

TextWrapping="Wrap"

VerticalScrollBarVisibility="Visible"

FontFamily="Consolas">

</TextBox>

</Grid>

public partial class Window1 : System.Windows.Window

{

public Window1()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, EventArgs e)

{

Type controlType = typeof(Control);

List<Type> derivedTypes = new List<Type>();

// Получаем все типы сборки, в которой объявлен класс Control

Assembly assembly = Assembly.GetAssembly(controlType);

foreach (Type type in assembly.GetTypes())

{

// Добавляем в список только те классы, которые производные от Control не абстрактные и открытые.

if (type.IsSubclassOf(controlType) && !type.IsAbstract && type.IsPublic)

{

derivedTypes.Add(type);

}

}

// Сортируем по имени.

derivedTypes.Sort(new TypeComparer());

// Привязываем коллекцию к списку.

lstTypes.ItemsSource = derivedTypes;

}

private void lstTypes\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

// Получаем выбранный элемент в списке и приводим его к типу Type

Type type = (Type)lstTypes.SelectedItem;

// Создаем экземпляр выбранного типа. (для того чтобы можно было получить XAML разметку шаблона)

ConstructorInfo info = type.GetConstructor(System.Type.EmptyTypes);

Control control = (Control)info.Invoke(null);

// Добавление элемента в грид, но при этом элемент остается спрятанным.

control.Visibility = Visibility.Collapsed;

grid.Children.Add(control);

// Получаем шаблон контрола.

ControlTemplate template = control.Template;

// Получаем XAML разметку выбранного элемента.

XmlWriterSettings settings = new XmlWriterSettings();

settings.Indent = true; // для добавления отступов.

StringBuilder sb = new StringBuilder();

XmlWriter writer = XmlWriter.Create(sb, settings);

XamlWriter.Save(template, writer);

// Отображаем элемент управления.

txtTemplate.Text = sb.ToString();

// Удаление элемента из грида.

grid.Children.Remove(control);

}

catch (Exception err)

{

txtTemplate.Text = "<< Ошибка чтения шаблона: " + err.Message + ">>";

}

}

}

public class TypeComparer : IComparer<Type>

{

public int Compare(Type x, Type y)

{

return x.Name.CompareTo(y.Name);

}

}

## ButtonTemplate

<Window.Resources>

<!--Шаблон для кнопок-->

<ControlTemplate x:Key="MyButtonTemplate" TargetType="{x:Type Button}">

<!--В стандартном шаблоне кнопки вместо Border используется ButtonChrome-->

<Border

Name="Border"

Background="Red"

BorderBrush="Black"

CornerRadius="3"

TextBlock.Foreground="White"

TextBlock.FontWeight="Bold">

<Grid>

<Rectangle Name="StrokeBorder" Stroke="Black" StrokeThickness="0" StrokeDashArray="1, 1"></Rectangle>

<!--{TemplateBinding Padding} позволяет получить значение свойства Padding

контрола который использует этот шаблон-->

<!--ContentPresenter этот элемент указывает куда нужно помещать содержимое.-->

<ContentPresenter Margin="{TemplateBinding Padding}"></ContentPresenter>

</Grid>

</Border>

<ControlTemplate.Triggers>

<!--При наведении мыши-->

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="Border" Property="Background" Value="DarkRed"></Setter>

</Trigger>

<Trigger Property="IsFocused" Value="True">

<Setter TargetName="StrokeBorder" Property="StrokeThickness" Value="3"></Setter>

</Trigger>

<!--При нажатии-->

<Trigger Property="IsPressed" Value="True">

<Setter TargetName="Border" Property="Background" Value="IndianRed"></Setter>

<Setter TargetName="Border" Property="TextBlock.Foreground" Value="Black"></Setter>

</Trigger>

<!--Если кнопка отключена-->

<Trigger Property="IsEnabled" Value="False">

<Setter TargetName="Border" Property="TextBlock.Foreground" Value="Gray" />

<Setter TargetName="Border" Property="Background" Value="MistyRose" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Window.Resources>

<StackPanel>

<Button Padding="10" Margin="10" Template="{StaticResource MyButtonTemplate}" BorderThickness="3">

Button with template</Button>

<Button Padding="10" Margin="10" Template="{StaticResource MyButtonTemplate}" IsEnabled="False">

Button with template (Disabled)</Button>

</StackPanel>

## ListBoxTemplate

<Window.Resources>

<!--Стиль для кнопок, которые находятся между кнопкой вверх, вниз и кареткой-->

<Style x:Key="RepeatButtonStyle" TargetType="{x:Type RepeatButton}">

<Setter Property="IsTabStop" Value="False" />

<Setter Property="Focusable" Value="False" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type RepeatButton}">

<Border Background="Transparent"></Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<!--Стиль для каретки на полосе прокрутки-->

<Style x:Key="ThumbStyle" TargetType="{x:Type Thumb}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Thumb}">

<Ellipse Stroke="Black" Fill="Red"></Ellipse>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<!--Шаблон для полосы прокрутки, его использует стиль объявленный ниже-->

<ControlTemplate x:Key="VerticalScrollBar" TargetType="{x:Type ScrollBar}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition MaxHeight="18"/>

<RowDefinition Height="\*"/>

<RowDefinition MaxHeight="18"/>

</Grid.RowDefinitions>

<RepeatButton Grid.Row="0" Background="Transparent" Height="18" Command="ScrollBar.LineUpCommand" >

<Path Fill="Red" Data="M 0 4 L 8 4 L 4 0 Z"></Path>

</RepeatButton>

<Track Name="PART\_Track" Grid.Row="1" IsDirectionReversed="True">

<Track.DecreaseRepeatButton>

<RepeatButton Command="ScrollBar.PageUpCommand" Style="{StaticResource RepeatButtonStyle}"></RepeatButton>

</Track.DecreaseRepeatButton>

<Track.Thumb>

<Thumb Style="{StaticResource ThumbStyle}"></Thumb>

</Track.Thumb>

<Track.IncreaseRepeatButton>

<RepeatButton Command="ScrollBar.PageDownCommand" Style="{StaticResource RepeatButtonStyle}"></RepeatButton>

</Track.IncreaseRepeatButton>

</Track>

<RepeatButton Grid.Row="2" Height="18"

Command="ScrollBar.LineDownCommand">

<Path Fill="Red" Data="M 0 0 L 4 4 L 8 0 Z"></Path>

</RepeatButton>

</Grid>

</ControlTemplate>

<!--Стиль для полосы прокрутки устанавливается если она вертикальная-->

<Style TargetType="{x:Type ScrollBar}">

<Setter Property="SnapsToDevicePixels" Value="True"/>

<Style.Triggers>

<Trigger Property="Orientation" Value="Vertical">

<Setter Property="Width" Value="18"/>

<Setter Property="Height" Value="Auto" />

<Setter Property="Template" Value="{StaticResource VerticalScrollBar}" />

</Trigger>

</Style.Triggers>

</Style>

<!--Стиль для элементов ListBox, изменяет свойство Template-->

<Style TargetType="{x:Type ListBoxItem}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ListBoxItem}">

<Border Name="Bd" Margin="{TemplateBinding Control.Padding}">

<ContentPresenter></ContentPresenter>

</Border>

<ControlTemplate.Triggers>

<!--При наведении мыши-->

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="TextBlock.FontSize" Value="20"></Setter>

<Setter TargetName="Bd" Property="BorderThickness" Value="1"></Setter>

<Setter TargetName="Bd" Property="BorderBrush" Value="Gray"></Setter>

</Trigger>

<!--При выборе-->

<Trigger Property="IsSelected" Value="True">

<Setter Property="TextBlock.Foreground" Value="White"></Setter>

<Setter Property="TextBlock.FontSize" Value="20"></Setter>

<Setter Property="TextBlock.FontWeight" Value="Bold"></Setter>

<Setter TargetName="Bd" Property="Background" Value="Gray"></Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<ListBox>

<ListBoxItem>Item 1</ListBoxItem>

<ListBoxItem>Item 2</ListBoxItem>

<ListBoxItem>Item 3</ListBoxItem>

<ListBoxItem>Item 4</ListBoxItem>

<ListBoxItem>Item 5</ListBoxItem>

<ListBoxItem>Item 6</ListBoxItem>

<ListBoxItem>Item 7</ListBoxItem>

<ListBoxItem>Item 8</ListBoxItem>

<ListBoxItem>Item 9</ListBoxItem>

<ListBoxItem>Item 10</ListBoxItem>

<ListBoxItem>Item 11</ListBoxItem>

<ListBoxItem>Item 12</ListBoxItem>

</ListBox>

</Grid>

## GradientButton

<Window x:Class="GradientButton.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Использование шаблонов" Height="350" Width="350">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="TemplateResources/GradientButton1.xaml"></ResourceDictionary>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<StackPanel>

<Button Margin="10" Padding="10">Button 1</Button>

<Button Margin="10" Padding="10">Button 2</Button>

<Button Margin="10" Padding="10">Button 3</Button>

<CheckBox Margin="10" Click="CheckBox\_Click">Изменить стиль кнопок</CheckBox>

</StackPanel>

</Window>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

bool flag = false;

private void CheckBox\_Click(object sender, RoutedEventArgs e)

{

string path = string.Empty;

if (flag)

{

path = "TemplateResources/GradientButton1.xaml";

}

else

{

path = "TemplateResources/GradientButton2.xaml";

}

ResourceDictionary resourceDictionary = new ResourceDictionary();

resourceDictionary.Source = new Uri(path, UriKind.Relative);

this.Resources.MergedDictionaries[0] = resourceDictionary;

flag = !flag;

}

}

}

<!--ResourceDictionary был добавлен с помощью Add -> New Item -> Resource Dictionary (WPF)

Определяет хэш-таблицу или реализацию словаря с ресурсами WPF, которые используются компонентами и другими элементами приложения WPF

Эти ресурсы будут использоваться шаблонами.-->

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<!--Если мышь над контролом-->

<RadialGradientBrush

x:Key="HighlightBackground"

RadiusX="1"

RadiusY="5"

GradientOrigin="0.5,0.3">

<GradientStop Color="White" Offset="0" />

<GradientStop Color="Blue" Offset=".4" />

</RadialGradientBrush>

<!--При нажатии на кнопку-->

<RadialGradientBrush

x:Key="PressedBackground"

RadiusX="1" RadiusY="5"

GradientOrigin="0.5,0.3">

<GradientStop Color="White" Offset="0" />

<GradientStop Color="Blue" Offset="1" />

</RadialGradientBrush>

<!--Кисти-->

<SolidColorBrush x:Key="DefaultBackground" Color="Blue"></SolidColorBrush>

<SolidColorBrush x:Key="DisabledBackground" Color="Gray"></SolidColorBrush>

<!--Градиент для Border-->

<RadialGradientBrush x:Key="BorderBrush"

RadiusX="1"

RadiusY="5"

GradientOrigin="0.5,0.3">

<GradientStop Color="White" Offset="0" />

<GradientStop Color="Blue" Offset="1" />

</RadialGradientBrush>

<!--Шаблон кнопки-->

<ControlTemplate x:Key="GradientButtonTemplate"

TargetType="{x:Type Button}">

<Border

Name="Border"

BorderBrush="{StaticResource BorderBrush}"

BorderThickness="2"

CornerRadius="2"

Background="{StaticResource DefaultBackground}"

TextBlock.Foreground="White">

<Grid>

<!--Для отображения клавиатурного фокуса.-->

<Rectangle

Name="FocusCue"

Visibility="Hidden"

Stroke="White"

StrokeThickness="2"

StrokeDashArray="1 2"

SnapsToDevicePixels="True">

</Rectangle>

<ContentPresenter

Margin="{TemplateBinding Padding}"

RecognizesAccessKey="True">

</ContentPresenter>

</Grid>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="Border" Property="Background" Value="{StaticResource HighlightBackground}" />

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter TargetName="Border" Property="Background" Value="{StaticResource PressedBackground}" />

</Trigger>

<Trigger Property="IsKeyboardFocused" Value="True">

<Setter TargetName="FocusCue" Property="Visibility" Value="Visible"></Setter>

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter TargetName="Border" Property="Background" Value="{StaticResource DisabledBackground}"></Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

<Style TargetType="{x:Type Button}">

<Setter Property="Control.Template" Value="{StaticResource GradientButtonTemplate}"></Setter>

</Style>

</ResourceDictionary>

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<!--Градиент для заливки кнопки в обычном состоянии-->

<LinearGradientBrush x:Key="NormalBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#FFF" Offset="0.0"/>

<GradientStop Color="#CCC" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Градиент для рамки в нормальном состоянии-->

<LinearGradientBrush x:Key="NormalBorderBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#CCC" Offset="0.0"/>

<GradientStop Color="#444" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Градиент для рамки кнопки по умолчанию-->

<LinearGradientBrush x:Key="DefaultedBorderBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#777" Offset="0.0"/>

<GradientStop Color="#000" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Темная кисть-->

<LinearGradientBrush x:Key="DarkBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#FFF" Offset="0.0"/>

<GradientStop Color="#AAA" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Градиент для заливки нажатой кнопки-->

<LinearGradientBrush x:Key="PressedBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#BBB" Offset="0.0"/>

<GradientStop Color="#EEE" Offset="0.1"/>

<GradientStop Color="#EEE" Offset="0.9"/>

<GradientStop Color="#FFF" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Градиент для рамки нажатой кнопки-->

<LinearGradientBrush x:Key="PressedBorderBrush" StartPoint="0,0" EndPoint="0,1">

<GradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Color="#444" Offset="0.0"/>

<GradientStop Color="#888" Offset="1.0"/>

</GradientStopCollection>

</GradientBrush.GradientStops>

</LinearGradientBrush>

<!--Кисти для рисования отключенной кнопки-->

<SolidColorBrush x:Key="DisabledBackgroundBrush" Color="#EEE" />

<SolidColorBrush x:Key="DisabledBorderBrush" Color="#AAA" />

<SolidColorBrush x:Key="DisabledForegroundBrush" Color="#888" />

<!--Шаблон для кнопок-->

<ControlTemplate x:Key="ButtonTemplate" TargetType="{x:Type Button}">

<Border x:Name="Border"

CornerRadius="2"

BorderThickness="1"

Background="{StaticResource NormalBrush}"

BorderBrush="{StaticResource NormalBorderBrush}">

<ContentPresenter

Margin="{TemplateBinding Button.Padding}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

RecognizesAccessKey="True"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsKeyboardFocused" Value="True">

<Setter TargetName="Border" Property="BorderBrush" Value="{StaticResource DefaultedBorderBrush}" />

</Trigger>

<Trigger Property="IsDefaulted" Value="true">

<Setter TargetName="Border" Property="BorderBrush" Value="{StaticResource DefaultedBorderBrush}" />

</Trigger>

<Trigger Property="IsMouseOver" Value="true">

<Setter TargetName="Border" Property="Background" Value="{StaticResource DarkBrush}" />

</Trigger>

<Trigger Property="IsPressed" Value="true">

<Setter TargetName="Border" Property="Background" Value="{StaticResource PressedBrush}" />

<Setter TargetName="Border" Property="BorderBrush" Value="{StaticResource PressedBorderBrush}" />

</Trigger>

<Trigger Property="IsEnabled" Value="false">

<Setter TargetName="Border" Property="Background" Value="{StaticResource DisabledBackgroundBrush}" />

<Setter TargetName="Border" Property="BorderBrush" Value="{StaticResource DisabledBorderBrush}" />

<Setter Property="Foreground" Value="{StaticResource DisabledForegroundBrush}"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

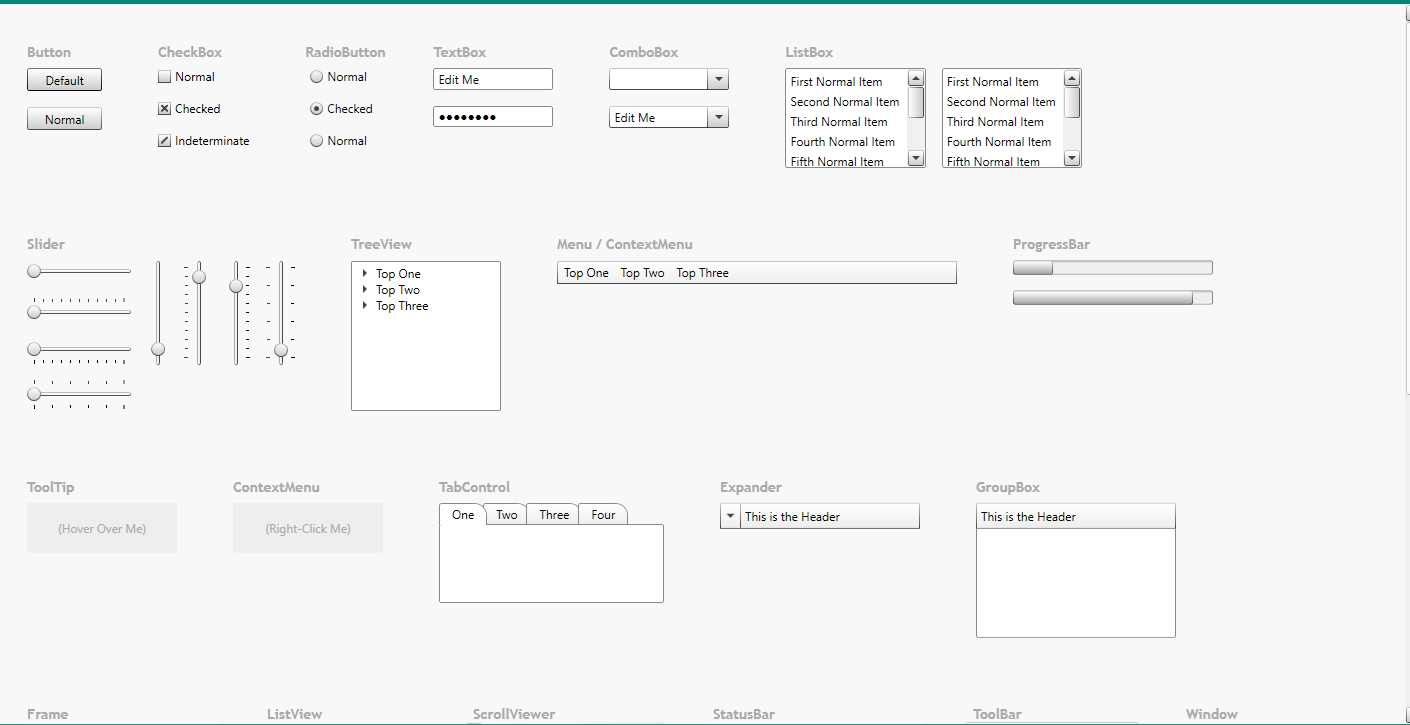
<Style TargetType="{x:Type Button}">

<Setter Property="Control.Template" Value="{StaticResource ButtonTemplate}"></Setter>

</Style>

</ResourceDictionary>

## Styles



<ScrollViewer>

<WrapPanel>

<!-- Button -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="Button">

<Button Margin="8" IsDefault="True">Default</Button>

<Button Margin="8">Normal</Button>

</HeaderedItemsControl>

<!-- CheckBox -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="CheckBox">

<CheckBox Margin="8">Normal</CheckBox>

<CheckBox Margin="8" IsChecked="true">Checked</CheckBox>

<CheckBox Margin="8" IsThreeState="true" IsChecked="{x:Null}">Indeterminate</CheckBox>

</HeaderedItemsControl>

<!-- RadioButton -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="RadioButton">

<RadioButton Margin="8">Normal</RadioButton>

<RadioButton Margin="8" IsChecked="true">Checked</RadioButton>

<RadioButton Margin="8">Normal</RadioButton>

</HeaderedItemsControl>

<!-- TextBox -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="TextBox">

<StackPanel>

<TextBox HorizontalAlignment="Center" Margin="8" Text="Edit Me" />

<PasswordBox HorizontalAlignment="Center" Margin="8" Password="Password" />

</StackPanel>

</HeaderedItemsControl>

<!-- ComboBox -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ComboBox">

<StackPanel>

<ComboBox Margin="8">

<ComboBoxItem Content="First Normal Item" />

<ComboBoxItem Content="Second Normal Item" />

<ComboBoxItem Content="Third Normal Item" />

<ComboBoxItem Content="Fourth Normal Item" />

<ComboBoxItem Content="Fifth Normal Item" />

</ComboBox>

<ComboBox Margin="8" IsEditable="True" Text="Edit Me">

<ComboBoxItem Content="First Normal Item" />

<ComboBoxItem Content="Second Normal Item" />

<ComboBoxItem Content="Third Normal Item" />

<ComboBoxItem Content="Fourth Normal Item" />

<ComboBoxItem Content="Fifth Normal Item" />

</ComboBox>

</StackPanel>

</HeaderedItemsControl>

<!-- ListBox -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ListBox">

<StackPanel Orientation="Horizontal">

<ListBox Margin="8" SelectionMode="Extended" Height="100">

<ListBoxItem Content="First Normal Item" />

<ListBoxItem Content="Second Normal Item" />

<ListBoxItem Content="Third Normal Item" />

<ListBoxItem Content="Fourth Normal Item" />

<ListBoxItem Content="Fifth Normal Item" />

<ListBoxItem Content="Sixth Normal Item" />

<ListBoxItem Content="Seventh Normal Item" />

<ListBoxItem Content="Eighth Normal Item" />

</ListBox>

<ListBox Margin="8" SelectionMode="Extended" ScrollViewer.VerticalScrollBarVisibility="Visible" Height="100">

<ListBoxItem Content="First Normal Item" />

<ListBoxItem Content="Second Normal Item" />

<ListBoxItem Content="Third Normal Item" />

<ListBoxItem Content="Fourth Normal Item" />

<ListBoxItem Content="Fifth Normal Item" />

<ListBoxItem Content="Sixth Normal Item" />

<ListBoxItem Content="Seventh Normal Item" />

<ListBoxItem Content="Eighth Normal Item" />

</ListBox>

</StackPanel>

</HeaderedItemsControl>

<!-- Slider -->

<HeaderedItemsControl Style="{StaticResource HorizontalHIC}" Header="Slider">

<StackPanel>

<Slider Margin="8" Orientation="Horizontal" />

<Slider Margin="8" Orientation="Horizontal" TickPlacement="TopLeft" />

<Slider Margin="8" Orientation="Horizontal" TickPlacement="BottomRight" />

<Slider Margin="8" Orientation="Horizontal" TickPlacement="Both" TickFrequency="2" />

</StackPanel>

<StackPanel Orientation="Horizontal" VerticalAlignment="Top">

<Slider Margin="8" Orientation="Vertical" />

<Slider Margin="8" Orientation="Vertical" TickPlacement="TopLeft" />

<Slider Margin="8" Orientation="Vertical" TickPlacement="BottomRight" />

<Slider Margin="8" Orientation="Vertical" TickPlacement="Both" TickFrequency="2" />

</StackPanel>

</HeaderedItemsControl>

<!-- TreeView -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="TreeView">

<StackPanel>

<TreeView Width="150" Height="150" Margin="8">

<TreeViewItem Header="Top One">

<TreeViewItem Header="Sub One" />

<TreeViewItem Header="Sub Two (Longer than Normal)" />

<TreeViewItem Header="Sub Three" />

<TreeViewItem Header="Sub Four">

<TreeViewItem Header="Sub One" />

<TreeViewItem Header="Sub Two" />

<TreeViewItem Header="Sub Three" />

</TreeViewItem>

<TreeViewItem Header="Sub Five" />

</TreeViewItem>

<TreeViewItem Header="Top Two">

<TreeViewItem Header="Sub One" />

<TreeViewItem Header="Checkable" />

<TreeViewItem Header="Sub Three" />

<TreeViewItem Header="Sub Four">

<TreeViewItem Header="Sub One" />

<TreeViewItem Header="Sub Two" />

<TreeViewItem Header="Sub Three" />

</TreeViewItem>

<TreeViewItem Header="Sub Five" />

</TreeViewItem>

<TreeViewItem Header="Top Three" >

<TreeViewItem Header="Sub One" />

<TreeViewItem Header="Sub Two" />

<TreeViewItem Header="Sub Three" />

<TreeViewItem Header="Sub Five" />

</TreeViewItem>

</TreeView>

</StackPanel>

</HeaderedItemsControl>

<!-- Menu -->

<HeaderedItemsControl Style="{StaticResource HorizontalHIC}" Header="Menu / ContextMenu">

<Menu Margin="8" Width="400">

<MenuItem Header="Top One">

<MenuItem Header="Sub One" InputGestureText="Ctrl+L" />

<MenuItem Header="Sub Two (With an Icon)" InputGestureText="Ctrl+A">

<MenuItem.Icon>

<Ellipse Width="16" Height="16" Fill="LightBlue" />

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="Sub Three" />

<Separator />

<MenuItem Header="Sub Four">

<MenuItem Header="Sub One" />

<MenuItem Header="Sub Two" />

<MenuItem Header="Sub Three" />

</MenuItem>

<MenuItem Header="Sub Five" />

</MenuItem>

<MenuItem Header="Top Two">

<MenuItem Header="Sub One" />

<MenuItem Header="Sub Tow (Checkable)" IsCheckable="True" />

<MenuItem Header="Sub Three" />

<Separator />

<MenuItem Header="Sub Four">

<MenuItem Header="Sub One" />

<MenuItem Header="Sub Two" />

<MenuItem Header="Sub Three" />

</MenuItem>

<MenuItem Header="Sub Five" />

</MenuItem>

<MenuItem Header="Top Three" >

<MenuItem Header="Sub One" />

<MenuItem Header="Sub Two" />

<MenuItem Header="Sub Three" />

<Separator />

<MenuItem Header="Sub Five" />

</MenuItem>

</Menu>

</HeaderedItemsControl>

<!-- ProgressBar -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ProgressBar">

<StackPanel>

<ProgressBar HorizontalAlignment="Center" Margin="8" Value="20" />

<ProgressBar HorizontalAlignment="Center" Margin="8" Value="90" />

</StackPanel>

</HeaderedItemsControl>

<!-- ToolTip -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ToolTip">

<StackPanel>

<Border Margin="8" Background="#EEE" Width="150" Height="50" CornerRadius="2">

<Border.ToolTip>

This is a test tooltip! Woohoo!

</Border.ToolTip>

<TextBlock Foreground="#AAA" VerticalAlignment="Center" HorizontalAlignment="Center">(Hover Over Me)</TextBlock>

</Border>

</StackPanel>

</HeaderedItemsControl>

<!-- ContextMenu -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ContextMenu">

<StackPanel>

<Border Margin="8" Background="#EEE" Width="150" Height="50" CornerRadius="2">

<Border.ContextMenu>

<ContextMenu>

<MenuItem Header="Sub One" InputGestureText="Ctrl+L" />

<MenuItem Header="Sub Two (With an Icon)" InputGestureText="Ctrl+A">

<MenuItem.Icon>

<Ellipse Width="16" Height="16" Fill="LightBlue" />

</MenuItem.Icon>

</MenuItem>

<MenuItem Header="Sub Three" />

<Separator />

<MenuItem Header="Sub Four">

<MenuItem Header="Sub One" />

<MenuItem Header="Sub Two" />

<MenuItem Header="Sub Three" />

</MenuItem>

<MenuItem Header="Sub Five" />

</ContextMenu>

</Border.ContextMenu>

<TextBlock Foreground="#AAA" VerticalAlignment="Center" HorizontalAlignment="Center">(Right-Click Me)</TextBlock>

</Border>

</StackPanel>

</HeaderedItemsControl>

<!-- TabControl -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="TabControl">

<StackPanel Orientation="Horizontal">

<TabControl Margin="8" Height="100" Width="225">

<TabItem Header="One" />

<TabItem Header="Two">

<Label Content="Content goes here..." />

</TabItem>

<TabItem Header="Three" />

<TabItem Header="Four" />

</TabControl>

</StackPanel>

</HeaderedItemsControl>

<!-- Expander -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="Expander">

<StackPanel Orientation="Horizontal">

<Expander Width="200" Margin="8" Header="This is the Header">

<Border Height="100" />

</Expander>

</StackPanel>

</HeaderedItemsControl>

<!-- GroupBox -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="GroupBox">

<StackPanel>

<GroupBox Margin="8" Header="This is the Header" Width="200" >

<Border Height="100" />

</GroupBox>

</StackPanel>

</HeaderedItemsControl>

<!-- Frame -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="Frame">

<StackPanel>

<Frame Source="page1.xaml" Width="200" Height="200"/>

</StackPanel>

</HeaderedItemsControl>

<!-- ListView -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ListView">

<StackPanel>

<ListView Margin="8" Height="100" Width="150">

<ListView.View>

<GridView AllowsColumnReorder="true">

<GridViewColumn DisplayMemberBinding="{Binding}" Header="Content" Width="100"/>

<GridViewColumn DisplayMemberBinding="{Binding Path=Length}" Header="Length" Width="100"/>

</GridView>

</ListView.View>

<ListViewItem Content="First Item" />

<ListViewItem Content="Second Item" />

<ListViewItem Content="Third Item" />

<ListViewItem Content="Fourth Item" />

<ListViewItem Content="Fifth Item" />

<ListViewItem Content="Sixth Item" />

<ListViewItem Content="Seventh Item" />

<ListViewItem Content="Eighth Item" />

</ListView>

</StackPanel>

</HeaderedItemsControl>

<!-- ScrollViewer -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ScrollViewer">

<StackPanel>

<ScrollViewer Width="200" Height="200" Style="{StaticResource LeftScrollViewer}">

<Canvas Width="400" Height="400">

<Canvas.Background>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="#FFF" Offset="0"/>

<GradientStop Color="#AAA" Offset="1"/>

</LinearGradientBrush>

</Canvas.Background>

</Canvas>

</ScrollViewer>

</StackPanel>

</HeaderedItemsControl>

<!-- StatusBar -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="StatusBar">

<StackPanel Width="220">

<StatusBar VerticalAlignment="Bottom">

<StatusBarItem DockPanel.Dock="Right">Ln 181, Col 20</StatusBarItem>

<Separator DockPanel.Dock="Right"/>

<StatusBarItem>80%</StatusBarItem>

</StatusBar>

</StackPanel>

</HeaderedItemsControl>

<!-- ToolBar -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="ToolBar">

<StackPanel>

<ToolBarTray>

<ToolBar Grid.Row="1">

<Button>Button</Button>

<CheckBox>CheckBox</CheckBox>

<TextBox>TextBox</TextBox>

</ToolBar>

</ToolBarTray>

</StackPanel>

</HeaderedItemsControl>

<!-- Window -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="Window">

<StackPanel>

<Border Margin="8" Background="#EEE" Width="150" Height="50" CornerRadius="2" MouseLeftButtonDown="OpenWindow">

<TextBlock Foreground="#AAA" VerticalAlignment="Center" HorizontalAlignment="Center">(Click Me)</TextBlock>

</Border>

</StackPanel>

</HeaderedItemsControl>

<!-- NavigationWindow -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="Navigation Window">

<StackPanel>

<Border Margin="8" Background="#EEE" Width="150" Height="50" CornerRadius="2" MouseLeftButtonDown="OpenNavWindow">

<TextBlock Foreground="#AAA" VerticalAlignment="Center" HorizontalAlignment="Center">(Click Me)</TextBlock>

</Border>

</StackPanel>

</HeaderedItemsControl>

<!-- DocumentViewer -->

<HeaderedItemsControl Style="{StaticResource VerticalHIC}" Header="DocumentViewer">

<StackPanel>

<DocumentViewer Width="275"/>

</StackPanel>

</HeaderedItemsControl>

</WrapPanel>

</ScrollViewer>

# Binding

## Binding ElementName

<Window x:Class="WpfBinding.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Простая привязка данных" Height="350" Width="525">

<StackPanel Margin="12">

<!--Элемент источник данных-->

<Slider Name="silderFontSize"

Minimum="12"

Maximum="80"

Margin="10"></Slider>

<!--Элемент с целевым свойством привязки-->

<TextBlock Margin="10"

FontSize="{Binding ElementName=silderFontSize, Path=Value}">

Hello world

</TextBlock>

</StackPanel>

</Window>

## TwoWayMode

<StackPanel Margin="5">

<!--TickFrequency="1" размер одного тика-->

<!--IsSnapToTickEnabled="True" каретка может находиться только на тике значение в случае с TickFrequency="1" будет целым-->

<Slider Name="sliderFontSize" Margin="3"

Minimum="1" Maximum="40"

Value="10" TickFrequency="1"

IsSnapToTickEnabled="True" TickPlacement="TopLeft">

</Slider>

<!--Mode=TwoWay в привязке означает, что изменение свойства изменит источник.-->

<TextBlock Margin="10" Name="lblSampleText"

FontSize="{Binding ElementName=sliderFontSize, Path=Value, Mode=TwoWay}"

Text="Simple Text">

</TextBlock>

<!--При нажатии на кнопки будет меняться значение целевого свойства.

Если Mode для привязки на строке 18 не равен TwoWay, кнопки не повлияют на изменение позиции каретки,

но занчение шрифта изменится-->

<StackPanel Orientation="Horizontal" >

<Button Margin="5" Padding="3" Click="cmd\_SetSmall">Small</Button>

<Button Margin="5" Padding="3" Click="cmd\_SetNormal">Normal</Button>

<Button Margin="5" Padding="3" Click="cmd\_SetLarge">Large</Button>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="5">

<!--Для того чтобы не удалялись лишние пробелы установлен атрибут xml:space="preserve"-->

<TextBlock VerticalAlignment="Center" xml:space="preserve">Exact Size: </TextBlock>

<!--UpdateSourceTrigger=PropertyChanged обновление привязки будет происходить при изменении значения свойства

для поля ввода по умолчанию LostFocus-->

<TextBox Text="{Binding ElementName=lblSampleText,

Path=FontSize,

UpdateSourceTrigger=LostFocus,

Mode=TwoWay}"

Width="100"></TextBox>

</StackPanel>

</StackPanel>

## BindingSource

<Window.Resources>

<!--Источник данных-->

<local:PersonList x:Key="ListObject"></local:PersonList>

<!--Шаблон для одного элемента списка-->

<DataTemplate x:Key="ListDataTemplate">

<Border BorderBrush="Black"

BorderThickness="1"

Padding="10"

HorizontalAlignment="Stretch"

CornerRadius="5"

Margin="5">

<StackPanel>

<StackPanel Orientation="Horizontal" TextBlock.FontSize="20" TextBlock.FontWeight="Bold">

<TextBlock Text="{Binding FirstName}"></TextBlock>

<TextBlock Text=" "></TextBlock>

<TextBlock Text="{Binding LastName}"></TextBlock>

</StackPanel>

<StackPanel>

<TextBlock Text="{Binding Path=Age}"></TextBlock>

<TextBlock Text="{Binding Path=Position}"></TextBlock>

</StackPanel>

</StackPanel>

</Border>

</DataTemplate>

</Window.Resources>

<Grid>

<ListBox ItemsSource="{Binding Source={StaticResource ListObject}}"

HorizontalContentAlignment="Stretch">

</ListBox>

</Grid>

class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

public string Position { get; set; }

public Person(string firstName, string lastName, int age, string position)

{

FirstName = firstName;

LastName = lastName;

Age = age;

Position = position;

}

}

class PersonList : List<Person>

{

public PersonList()

{

this.Add(new Person("John", "Doe", 23, "Developer"));

this.Add(new Person("Kent", "Elgas", 29, "Tester"));

this.Add(new Person("Rea", "Ostrom", 31, "Team Lead"));

this.Add(new Person("Lupe", "Campen", 42, "Project Manager"));

this.Add(new Person("Alexander", "Heys", 35, "Developer"));

}

}

## BindClassPropertyToText

<Window.Resources>

<local:MyData x:Key="DataItem"

MyProperty="Test Data String">

</local:MyData>

</Window.Resources>

<StackPanel>

<TextBlock Foreground="Black"

FontSize="20"

Text="{Binding Source={StaticResource DataItem}, Path=MyProperty}">

</TextBlock>

</StackPanel>

using System.Windows;

namespace BindNotControl

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

public class MyData

{

public string MyProperty { get; set; }

}

}

## RelativeSource

<StackPanel Margin="5">

<StackPanel Margin="10">

<!--Привязка к своим свойствам используя относительный источник данных RelativeSource-->

<TextBlock FontSize="20"

FontFamily="Courier New"

Text="{Binding RelativeSource={RelativeSource Mode=Self}, Path=FontFamily}">

</TextBlock>

<!--Привязка к своим свойствам родителя типа Window-->

<TextBlock FontSize="20"

Text="{Binding RelativeSource={RelativeSource Mode=FindAncestor,

AncestorType=Window},

Path=FontFamily}">

</TextBlock>

<!--Привязка к своим свойствам второго родительского элемента типа StackPanel-->

<TextBlock FontSize="20"

Text="{Binding RelativeSource={RelativeSource Mode=FindAncestor,

AncestorType=StackPanel,

AncestorLevel=2},

Path=Margin}">

</TextBlock>

</StackPanel>

</StackPanel>

## DataContext

<Window.Resources>

<local:Person x:Key="PersonDataSource"

FirstName="Jhon"

LastName="Doe">

</local:Person>

</Window.Resources>

<Grid>

<!--Установив DataContext, вложенные элементы могут использовать Path без указания источника данных.-->

<StackPanel Margin="10" Name="panel1"

DataContext="{StaticResource PersonDataSource}">

<TextBlock>First name</TextBlock>

<TextBox Text="{Binding FirstName}"></TextBox>

<TextBlock></TextBlock>

<TextBlock>Last name</TextBlock>

<TextBox Text="{Binding LastName}"></TextBox>

</StackPanel>

</Grid>

using System.Windows;

namespace DataContextSample

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

public class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

}

## BindingToSQL

<Window x:Class="BindingToSQL.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Book Database" Height="350" Width="512" SizeToContent="Height">

<Grid Background="BlanchedAlmond" Name="gridMain">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="136\*" />

<ColumnDefinition Width="367\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="40" />

<RowDefinition Height="40" />

<RowDefinition Height="40" />

<RowDefinition Height="170\*" />

</Grid.RowDefinitions>

<ListBox Name="listBooks"

Visibility="Collapsed"

Grid.ColumnSpan="2"

Height="90"

Margin="10"

SelectionChanged="listBooks\_SelectionChanged">

</ListBox>

<StackPanel Grid.Row="1" Grid.ColumnSpan="2"

Orientation="Horizontal" Background="White">

<TextBox Name="textBoxID" VerticalAlignment="Center" Width="120" Margin="10,0,10,0" />

<Button Content="Получить книгу по ID" Name="buttonGetBook" VerticalAlignment="Center" Padding="2" Click="buttonGetBook\_Click" />

<Button Content="Обновить данные в базе" Name="UpdateBook" Padding="2" VerticalAlignment="Center" Margin="10,0,0,0" Click="UpdateBook\_Click"></Button>

<Button Content="Список" Name="buttonShowList" Margin="10,0,0,0" Padding="2" VerticalAlignment="Center" Click="buttonShowList\_Click" />

</StackPanel>

<!--Для полей ввода установленна привязка к свойствам класса Book.

Как источник данных объект Book будет устанавливатся из кода.-->

<TextBox Text="{Binding Title}" HorizontalAlignment="Stretch" Name="textBoxTitle" VerticalAlignment="Stretch" Grid.Row="2" Margin="5" Grid.Column="1" />

<TextBox Text="{Binding Author}" HorizontalAlignment="Stretch" Name="textBoxAuthor" VerticalAlignment="Stretch" Grid.Row="3" Margin="5" Grid.Column="1" />

<TextBox Text="{Binding Description}" TextWrapping="Wrap" HorizontalAlignment="Stretch" Name="textBoxDesc" VerticalAlignment="Stretch" Grid.Row="4" Margin="5" Grid.Column="1" MinHeight="150" />

<TextBlock Grid.Row="2" HorizontalAlignment="Center" Text="Книга" VerticalAlignment="Center" />

<TextBlock Grid.Row="3" HorizontalAlignment="Center" Text="Автор" VerticalAlignment="Center" />

<TextBlock Grid.Row="4" HorizontalAlignment="Center" Text="Описание" VerticalAlignment="Center" />

</Grid>

</Window>

namespace BindingToSQL

{

public partial class MainWindow : Window

{

Book \_currentBook = null;

StoreDB \_db = new StoreDB();

public MainWindow()

{

InitializeComponent();

}

private void buttonGetBook\_Click(object sender, RoutedEventArgs e)

{

try

{

int bookID = Convert.ToInt32(textBoxID.Text);

\_currentBook = \_db.GetBook(bookID);

gridMain.DataContext = \_currentBook;

}

catch (Exception)

{

MessageBox.Show("Ошибка при запросе к базе данных.");

}

}

private void UpdateBook\_Click(object sender, RoutedEventArgs e)

{

try

{

int bookID = Convert.ToInt32(textBoxID.Text);

\_db.UpdateBook(\_currentBook, bookID);

}

catch (Exception)

{

MessageBox.Show("Ошибка при обновлении записи в базе данных.");

}

}

bool isCollapsed = true;

private void buttonShowList\_Click(object sender, RoutedEventArgs e)

{

if (isCollapsed)

{

listBooks.Visibility = System.Windows.Visibility.Visible;

try

{

listBooks.ItemsSource = \_db.GetAllBooks();

listBooks.DisplayMemberPath = "Title";

}

catch

{

MessageBox.Show("Ошибка при запросе к базе данных.");

}

}

else

{

listBooks.Visibility = System.Windows.Visibility.Collapsed;

}

isCollapsed = !isCollapsed;

}

private void listBooks\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

gridMain.DataContext = listBooks.SelectedItem;

}

}

}

namespace BindingToSQL.DB

{

public class Book

{

public string Title { get; set; }

public string Description { get; set; }

public string Author { get; set; }

public Book() { }

public Book(string title, string description, string author)

{

Title = title;

Description = description;

Author = author;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient;

namespace BindingToSQL

{

// Для работы примера в Settings.settings нужно поменять путь к

class StoreDB

{

public Book GetBook(int bookId)

{

Book book = null;

SqlDataReader reader = null;

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.ConnectionString))

{

SqlCommand command = new SqlCommand("SELECT \* FROM Books WHERE ID=@id", connection);

command.Parameters.AddWithValue("id", bookId);

connection.Open();

try

{

reader = command.ExecuteReader();

reader.Read();

book = new Book()

{

Title = reader[1].ToString(),

Description = reader[2].ToString(),

Author = reader[3].ToString()

};

}

catch (Exception e)

{

if (reader != null)

{

reader.Close();

}

throw e;

}

connection.Close();

return book;

}

}

public void UpdateBook(Book book, int bookId)

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.ConnectionString))

{

SqlCommand command = new SqlCommand("UPDATE Books SET Title=@title, Description=@description, Author=@author WHERE ID=@id", connection);

command.Parameters.AddWithValue("title", book.Title);

command.Parameters.AddWithValue("description", book.Description);

command.Parameters.AddWithValue("author", book.Author);

command.Parameters.AddWithValue("ID", bookId);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

public List<Book> GetAllBooks()

{

List<Book> books = new List<Book>();

SqlDataReader reader = null;

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.ConnectionString))

{

connection.Open();

SqlCommand command = new SqlCommand("SELECT \* FROM Books", connection);

try

{

reader = command.ExecuteReader();

while (reader.Read())

{

Book book = new Book()

{

Title = reader[1].ToString(),

Description = reader[2].ToString(),

Author = reader[3].ToString()

};

books.Add(book);

}

}

catch (Exception ex)

{

reader.Close();

throw ex;

}

connection.Close();

}

return books;

}

}

}

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=4.4.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<connectionStrings>

<add name="BindingToSQL.Properties.Settings.ConnectionString" connectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\BookDatabase.mdf;Integrated Security=True;User Instance=True" providerName="System.Data.SqlClient" />

<add name="BookDatabaseEntities" connectionString="metadata=res://\*/Model1.csdl|res://\*/Model1.ssdl|res://\*/Model1.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=.\SQLEXPRESS;attachdbfilename=|DataDirectory|\BookDatabase.mdf;integrated security=True;user instance=True;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

</connectionStrings>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

</entityFramework>

</configuration>

## MultipleBindings

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Slider Name="sliderFontSize" Margin="3" Minimum="1" Maximum="40" Value="10"></Slider>

<TextBox Name="txtContent" Margin="3" Grid.Row="2">Sample Content</TextBox>

<ListBox Margin="3" Grid.Row="3" Name="lstColors">

<ListBoxItem Tag="Blue">Blue</ListBoxItem>

<ListBoxItem Tag="DarkBlue">Dark Blue</ListBoxItem>

<ListBoxItem Tag="LightBlue">Light Blue</ListBoxItem>

</ListBox>

<!--Привязка сразу к нескольким свойствам.

Для получения свойств объекта используется запись SelectedItem.Tag-->

<TextBlock Margin="3" Name="lblSampleText"

FontSize="{Binding ElementName=sliderFontSize, Path=Value}" Grid.Row="4"

Text="{Binding ElementName=txtContent, Path=Text}"

Foreground="{Binding ElementName=lstColors, Path=SelectedItem.Tag}">

</TextBlock>

</Grid>

## ObservableCollection

public partial class MainWindow : Window

{

//List<string> list = new List<string>();

ObservableCollection<string> list = new ObservableCollection<string>();

public MainWindow()

{

InitializeComponent();

string[] arr = {"First", "Second", "Third", "Fourth" };

foreach (string item in arr)

{

list.Add(item);

}

listBox1.ItemsSource = list;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

// Новое значение добавляется в коллекцию но ListBox его не отобразит.

// Для того, чтобы контрол реагировал на изменения коллекции,

// Коллекция, которая используется как ItemsSource, должна реализовывать интерфейс INotifyCollectionChanged.

// В WPF есть только одна коллекция, которая реализующая этот интерфейс ObservableCollection<T>

list.Add(DateTime.Now.ToLongTimeString());

}

## SliderConverter

<Window.Resources>

<local:Converter x:Key="ConverterObject"></local:Converter>

</Window.Resources>

<StackPanel>

<Slider Minimum="0" Maximum="7" Name="Slider1"></Slider>

<TextBox Text="{Binding ElementName=Slider1,

Path=Value,

Converter={StaticResource ConverterObject},

UpdateSourceTrigger=PropertyChanged}">

</TextBox>

</StackPanel>

namespace ConverterSample

{

// Интерфейс для добавления пользовательской логики при привязке данных.

public class Converter: IValueConverter

{

// Parameters:

// value:

// Значение которое задает источник данных

//

// targetType:

// Тип целевого свойства. (Тип который нужно вернуть)

//

// parameter:

// Дополнительные параметры

//

// culture:

// Культура

//

// Returns:

// Конвертированное значение.

/// От источника к цели.

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

int number = System.Convert.ToInt32(value);

switch(number)

{

case 0:

return "Zero";

case 1:

return "One";

case 2:

return "Two";

case 3:

return "Three";

case 4:

return "Four";

case 5:

return "Five";

default:

return "ERROR";

}

}

// От цели к источнику

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

string numberString = value.ToString();

switch (numberString)

{

case "Zero":

return 0;

case "One":

return 1;

case "Two":

return 2;

case "Three":

return 3;

case "Four":

return 4;

case "Five":

return 5;

default:

return 0;

}

}

}

}

# DataTemplates

## ComboBoxBinding

<Grid>

<Grid.Resources>

<local:ImageConverter x:Key="Converter"></local:ImageConverter>

</Grid.Resources>

<ComboBox Name="comboBox1"

Height="90"

VerticalAlignment="Top"

HorizontalContentAlignment="Stretch"

IsEditable="False" IsReadOnly="True">

<ComboBox.ItemTemplate>

<DataTemplate>

<DockPanel HorizontalAlignment="Stretch">

<TextBlock DockPanel.Dock="Left"

Margin="10"

FontSize="18"

Text="{Binding Path=Name}">

</TextBlock>

<Image DockPanel.Dock="Right"

HorizontalAlignment="Right"

Height="70"

Width="70"

Margin="5"

Source="{Binding Path=Picture, Converter={StaticResource Converter}, ConverterParameter='Pictures'}"></Image>

</DockPanel>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</Grid>

namespace ComboBoxBinding

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

List<MyItem> items = new List<MyItem>();

items.Add(new MyItem() { Name = "Яблоко", Picture = "1.jpg" });

items.Add(new MyItem() { Name = "Апельсин", Picture = "2.jpg" });

items.Add(new MyItem() { Name = "Ананас", Picture = "3.jpg" });

items.Add(new MyItem() { Name = "Авокадо", Picture = "4.jpg" });

items.Add(new MyItem() { Name = "Банан", Picture = "5.jpg" });

Binding binding = new Binding();

binding.Source = items;

comboBox1.SetBinding(ComboBox.ItemsSourceProperty, binding);

}

}

}

namespace ComboBoxBinding

{

class ImageConverter : IValueConverter

{

#region IValueConverter Members

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

return Path.Combine(parameter.ToString(), value.ToString());

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotSupportedException();

}

#endregion

}

}

namespace ComboBoxBinding

{

public class MyItem

{

public string Name { get; set; }

public string Picture { get; set; }

public override string ToString()

{

return Name;

}

}

}

## GridView

<Grid>

<Grid.Resources>

<local:PersonList x:Key="PersonList"></local:PersonList>

</Grid.Resources>

<ListView Name="listView1"

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

HorizontalContentAlignment="Stretch"

ItemsSource="{StaticResource PersonList}"

MouseDoubleClick="ListView\_MouseDoubleClick">

<ListView.View>

<GridView>

<GridViewColumn DisplayMemberBinding="{Binding Path=FirstName}" Header="First Name" />

<GridViewColumn DisplayMemberBinding="{Binding Path=LastName}" Header="Last Name" />

<GridViewColumn DisplayMemberBinding="{Binding Path=Age}" Header="Age" />

<GridViewColumn DisplayMemberBinding="{Binding Path=Position}" Header="Position" />

</GridView>

</ListView.View>

</ListView>

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void ListView\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

Person current = listView1.SelectedItem as Person;

if (current != null)

{

string result = string.Format("{0} {1}\nPosition : {2}\nAge: {3}",

current.FirstName,

current.LastName,

current.Position,

current.Age);

MessageBox.Show(result);

}

}

}

namespace GridViewSample

{

class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

public string Position { get; set; }

public Person(string firstName, string lastName, int age, string position)

{

FirstName = firstName;

LastName = lastName;

Age = age;

Position = position;

}

}

class PersonList : List<Person>

{

public PersonList()

{

this.Add(new Person("John", "Doe", 23, "Developer"));

this.Add(new Person("Kent", "Elgas", 29, "Tester"));

this.Add(new Person("Rea", "Ostrom", 31, "Team Lead"));

this.Add(new Person("Lupe", "Campen", 42, "Project Manager"));

this.Add(new Person("Alexander", "Heys", 35, "Developer"));

}

}

}

## TreeView

<Grid>

<TreeView Name="treeView1">

<!--Шаблон для описания структуры дерева-->

<TreeView.ItemTemplate>

<HierarchicalDataTemplate ItemsSource="{Binding Path=trafficList}">

<TextBlock Text="{Binding Path=Name}" />

<HierarchicalDataTemplate.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path=TrafficDescription}"></TextBlock>

</DataTemplate>

</HierarchicalDataTemplate.ItemTemplate>

</HierarchicalDataTemplate>

</TreeView.ItemTemplate>

</TreeView>

</Grid>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

treeView1.ItemsSource = CategoryCreator.GetCreatorList();

}

}

}

## DirectoryTreeView

<Grid Margin="3">

<TreeView Name="treeFileSystem"

TreeViewItem.Expanded="item\_Expanded"

MouseDoubleClick="treeFileSystem\_MouseDoubleClick" >

</TreeView>

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

BuildTree();

}

private void BuildTree()

{

treeFileSystem.Items.Clear();

foreach (DriveInfo drive in DriveInfo.GetDrives())

{

TreeViewItem item = new TreeViewItem();

item.Tag = drive;

item.Header = drive.ToString();

// \* отображаться не будет, так как узел находится в закрытом состоянии,

item.Items.Add("\*");

treeFileSystem.Items.Add(item);

}

}

private void item\_Expanded(object sender, RoutedEventArgs e)

{

TreeViewItem item = (TreeViewItem)e.OriginalSource;

// выполняем обновление каждый раз, когда узел разворачивается.

item.Items.Clear();

DirectoryInfo dir;

if (item.Tag is DriveInfo)

{

DriveInfo drive = (DriveInfo)item.Tag;

dir = drive.RootDirectory;

}

else

{

dir = (DirectoryInfo)item.Tag;

}

try

{

foreach (DirectoryInfo subDir in dir.GetDirectories())

{

TreeViewItem newItem = new TreeViewItem();

newItem.Tag = subDir;

newItem.Header = subDir.ToString();

newItem.Items.Add("\*");

item.Items.Add(newItem);

}

}

catch

{

MessageBox.Show("Error");

}

}

private void treeFileSystem\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

TreeViewItem item = treeFileSystem.SelectedItem as TreeViewItem;

if (item != null)

{

DirectoryInfo dirInfo = item.Tag as DirectoryInfo;

if (dirInfo != null)

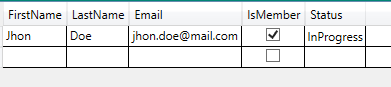
Title = dirInfo.FullName;

}

}

}

## DataGrid



<Grid>

<DataGrid AutoGenerateColumns="True" Name="dataGrid1" CanUserAddRows="True" />

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

dataGrid1.ItemsSource = Customer.GetCustomerList();

}

}

namespace DataGridSample

{

public enum OrderStatus

{

InProgress, Delivered

}

public class Customer

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Email { get; set; }

public bool IsMember { get; set; }

public OrderStatus Status { get; set; }

public static ObservableCollection<Customer> GetCustomerList()

{

ObservableCollection<Customer> collection = new ObservableCollection<Customer>();

collection.Add(new Customer() { FirstName = "Jhon", LastName = "Doe", Email = "jhon.doe@mail.com", IsMember = true, Status = OrderStatus.InProgress });

return collection;

}

}

## DataGridSelector

<Window.Resources>

<local:StatusTemplateSelector x:Key="statusTemplateSelector">

<local:StatusTemplateSelector.DeliveredTemplate>

<DataTemplate>

<Grid Background="LightGreen">

<TextBlock>Delivered</TextBlock>

</Grid>

</DataTemplate>

</local:StatusTemplateSelector.DeliveredTemplate>

<local:StatusTemplateSelector.InProgressTemplate>

<DataTemplate>

<Grid Background="Salmon">

<TextBlock>In Progress</TextBlock>

</Grid>

</DataTemplate>

</local:StatusTemplateSelector.InProgressTemplate>

</local:StatusTemplateSelector>

<local:Customer x:Key="customers"></local:Customer>

</Window.Resources>

<Grid>

<DataGrid AutoGenerateColumns="False"

ItemsSource="{Binding Source={StaticResource customers}, Path=CustomersList}"

RowDetailsTemplateSelector="{StaticResource statusTemplateSelector}">

<DataGrid.Columns>

<DataGridTextColumn Header="Email" Binding="{Binding FirstName}"></DataGridTextColumn>

<DataGridTextColumn Header="Email" Binding="{Binding LastName}"></DataGridTextColumn>

<DataGridTextColumn Header="Email" Binding="{Binding Email}"></DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Window>

namespace \_7DataGridTemplateSelector

{

public class StatusTemplateSelector : DataTemplateSelector

{

public DataTemplate DeliveredTemplate { get; set; }

public DataTemplate InProgressTemplate { get; set; }

public override DataTemplate SelectTemplate(object item, DependencyObject container)

{

Customer customer = item as Customer;

if (customer == null)

{

return base.SelectTemplate(item, container);

};

if (customer.Status == OrderStatus.Delivered)

{

return DeliveredTemplate;

}

else

{

return InProgressTemplate;

}

}

}

}

## Menu

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<!--В WPF меню может находиться в любой части окна-->

<StackPanel>

<Button Padding="3" Margin="5">

<Menu VerticalAlignment="Top">

<MenuItem Header="File">

<MenuItem Command="ApplicationCommands.Open"></MenuItem>

</MenuItem>

<MenuItem Header="Help"></MenuItem>

</Menu>

</Button>

</StackPanel>

<TextBox Grid.Row="1" Margin="5">Поле ввода</TextBox>

<StackPanel Grid.Row="1" Grid.Column="2">

<CheckBox Margin="5">CheckBox</CheckBox>

<Menu VerticalAlignment="Top">

<MenuItem Header="File"></MenuItem>

<MenuItem Header="Help"></MenuItem>

</Menu>

</StackPanel>

<Menu Grid.Row="3" Margin="5" VerticalAlignment="Top" HorizontalAlignment="Left">

<MenuItem Header="File">

<MenuItem Header="New"></MenuItem>

<MenuItem Header="Open"></MenuItem>

<MenuItem Header="Save"></MenuItem>

</MenuItem>

</Menu>

</Grid>

## SideBarMenu

<DockPanel LastChildFill="True" Margin="5">

<Border BorderBrush="SteelBlue" BorderThickness="1">

<ScrollViewer DockPanel.Dock="Left">

<Menu>

<!--Для изменения панели, в которой отображается меню.-->

<Menu.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Background="White"></StackPanel>

</ItemsPanelTemplate>

</Menu.ItemsPanel>

<MenuItem Header="File">

<MenuItem Header="New"></MenuItem>

<MenuItem Header="Open"></MenuItem>

<MenuItem Header="Save"></MenuItem>

</MenuItem>

<MenuItem Header="Help"></MenuItem>

</Menu>

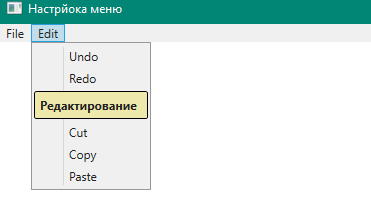
</ScrollViewer>

</Border>

<TextBox Margin="5,0,0,0" TextWrapping="Wrap">Other content goes here.</TextBox>

</DockPanel>

## Menu



<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Menu>

<MenuItem Header="File">

<MenuItem Header="New"></MenuItem>

<MenuItem Header="Open"></MenuItem>

<MenuItem Header="Save"></MenuItem>

<Separator></Separator>

<MenuItem Header="Exit"></MenuItem>

</MenuItem>

<MenuItem Header="Edit">

<MenuItem Header="Undo"></MenuItem>

<MenuItem Header="Redo"></MenuItem>

<Separator>

<!--Создание своего сепаратора-->

<Separator.Template>

<ControlTemplate>

<Border CornerRadius="2" Padding="5" BorderBrush="Black" BorderThickness="1" Background="PaleGoldenrod">

<TextBlock FontWeight="Bold">

Редактирование

</TextBlock>

</Border>

</ControlTemplate>

</Separator.Template>

</Separator>

<MenuItem Header="Cut"></MenuItem>

<MenuItem Header="Copy"></MenuItem>

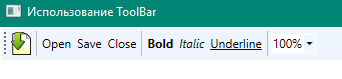
<MenuItem Header="Paste"></MenuItem>

</MenuItem>

</Menu>

</Grid>

## Toolbar



<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Icon.xaml"></ResourceDictionary>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<ToolBar>

<Button Content="{StaticResource DownloadFile}"></Button>

<Separator></Separator>

<Button>Open</Button>

<Button>Save</Button>

<Button>Close</Button>

<Separator></Separator>

<CheckBox FontWeight="Bold">Bold</CheckBox>

<CheckBox FontStyle="Italic">Italic</CheckBox>

<CheckBox>

<TextBlock TextDecorations="Underline">Underline</TextBlock>

</CheckBox>

<Separator></Separator>

<ComboBox SelectedIndex="0">

<ComboBoxItem>100%</ComboBoxItem>

<ComboBoxItem>50%</ComboBoxItem>

<ComboBoxItem>25%</ComboBoxItem>

</ComboBox>

</ToolBar>

</Grid>

<ResourceDictionary

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >

<!-- http://www.mikeswanson.com/XAMLExport для импорта векторного изображения из Adobe Illustrator в XAML -->

<Viewbox x:Key="DownloadFile" x:Shared="False" Width="20">

<Canvas Width="117.764648" Height="135.559570">

<Canvas>

<Canvas>

<Path StrokeThickness="5.216650" Stroke="#ff000000" StrokeMiterLimit="1.000000" Data="F1 M 85.454102,9.938965 C 80.053711,9.938965 10.747070,9.938965 10.747070,9.938965 L 10.747070,132.951172 L 115.156250,132.951172 C 115.156250,132.951172 115.156250,56.140137 115.156250,42.341309 C 115.156250,33.933105 94.643555,9.938965 85.454102,9.938965 Z">

<Path.Fill>

<LinearGradientBrush StartPoint="0.500000,0.513863" EndPoint="0.500000,1.101590">

<LinearGradientBrush.GradientStops>

<GradientStop Offset="0.000000" Color="#ffffffff"/>

<GradientStop Offset="1.000000" Color="#ffffff90"/>

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Path.Fill>

</Path>

<Path Fill="#ff4c4c4c" Data="F1 M 111.562500,40.507813 C 108.388672,32.172852 95.361328,35.732422 95.361328,35.732422 C 97.299805,27.265625 92.250977,14.162598 85.419922,13.693848 C 85.375977,13.693848 83.706055,13.693848 83.657227,13.693848 C 90.488281,14.162598 91.752930,31.853027 89.804688,40.314941 C 89.804688,40.314941 108.647461,32.553711 111.679688,41.201172 C 111.655273,40.974121 111.596680,40.739746 111.562500,40.507813 Z"/>

<Canvas>

<Canvas>

<Path Opacity="0.500000" Fill="#ff000000" Data="F1 M 33.603516,0.000000 C 29.707031,0.000000 25.034180,3.076172 24.282227,8.039551 L 24.165039,8.676758 C 24.165039,8.676758 24.165039,48.803711 24.165039,51.240234 C 21.962891,51.240234 8.745117,51.240234 8.745117,51.240234 C 6.103516,51.240234 4.013672,52.416992 3.017578,54.467773 L 2.504883,56.528320 C 2.500000,56.577148 2.500000,56.628418 2.500000,56.679688 C 2.500000,58.286133 3.022461,59.638672 3.979492,60.864258 L 37.128906,102.846680 C 38.500977,104.592285 40.512695,105.595703 42.636719,105.595703 C 44.750977,105.595703 46.752930,104.592285 48.125000,102.844238 L 81.264648,60.866699 C 82.875977,58.833008 83.232422,56.501465 82.241211,54.467773 C 81.250000,52.416992 79.165039,51.240234 76.513672,51.240234 C 76.513672,51.240234 63.291016,51.240234 61.088867,51.240234 C 61.088867,48.889160 61.088867,14.440918 61.088867,12.089844 C 63.378906,12.089844 83.364258,12.089844 83.364258,12.089844 L 82.353516,8.754883 C 81.987305,7.788086 78.623047,0.000000 63.759766,0.000000 L 33.603516,0.000000 Z"/>

</Canvas>

<Canvas>

<Path Fill="#ff000000" Data="F1 M 31.103516,0.000000 C 27.207031,0.000000 22.534180,3.076172 21.782227,8.039551 L 21.665039,8.676758 C 21.665039,8.676758 21.665039,48.803711 21.665039,51.240234 C 19.462891,51.240234 6.245117,51.240234 6.245117,51.240234 C 3.603516,51.240234 1.513672,52.416992 0.517578,54.467773 L 0.004883,56.528320 C 0.000000,56.577148 0.000000,56.628418 0.000000,56.679688 C 0.000000,58.286133 0.522461,59.638672 1.479492,60.864258 L 34.628906,102.846680 C 36.000977,104.592285 38.012695,105.595703 40.136719,105.595703 C 42.250977,105.595703 44.252930,104.592285 45.625000,102.844238 L 78.764648,60.866699 C 80.375977,58.833008 80.732422,56.501465 79.741211,54.467773 C 78.750000,52.416992 76.665039,51.240234 74.013672,51.240234 C 74.013672,51.240234 60.791016,51.240234 58.588867,51.240234 C 58.588867,48.889160 58.588867,14.440918 58.588867,12.089844 C 60.878906,12.089844 80.864258,12.089844 80.864258,12.089844 L 79.853516,8.754883 C 79.487305,7.788086 76.123047,0.000000 61.259766,0.000000 L 31.103516,0.000000 Z"/>

</Canvas>

</Canvas>

<Path Data="F1 M 74.013672,55.488281 L 54.340820,55.488281 L 54.340820,10.161133 L 75.844727,10.161133 C 75.844727,10.161133 73.549805,4.243164 61.259766,4.243164 C 48.964844,4.243164 33.212891,4.243164 31.103516,4.243164 C 29.155273,4.243164 26.401367,5.893555 25.971680,8.676758 L 25.917969,8.676758 L 25.917969,55.488281 L 6.245117,55.488281 C 4.277344,55.488281 3.627930,56.726074 4.829102,58.239746 L 37.968750,100.212402 C 39.150391,101.726074 41.108398,101.726074 42.304688,100.212402 L 75.434570,58.239746 C 76.621094,56.726074 75.986328,55.488281 74.013672,55.488281 Z">

<Path.Fill>

<LinearGradientBrush StartPoint="0.500000,0.145879" EndPoint="0.500000,1.179259">

<LinearGradientBrush.GradientStops>

<GradientStop Offset="0.000000" Color="#ffb2da0f"/>

<GradientStop Offset="1.000000" Color="#ff329f2b"/>

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

</Path.Fill>

</Path>

<Canvas>

<Path Fill="#ffffff00" Data="F1 M 74.013672,55.488281 L 56.831055,55.488281 L 56.831055,57.155762 L 75.991211,57.155762 C 76.210938,56.179199 75.483398,55.488281 74.013672,55.488281 Z"/>

<Path Fill="#ffffff00" Data="F1 M 7.314453,59.904785 C 6.118164,58.391113 6.762695,57.155762 8.730469,57.155762 L 28.403320,57.155762 L 28.403320,10.344238 L 28.461914,10.344238 C 28.891602,7.563477 31.645508,5.913086 33.593750,5.913086 C 35.698242,5.913086 51.450195,5.913086 63.745117,5.913086 C 68.623047,5.913086 71.918945,6.848145 74.130859,7.976074 C 72.241211,6.289063 68.486328,4.243164 61.259766,4.243164 C 48.964844,4.243164 33.212891,4.243164 31.103516,4.243164 C 29.155273,4.243164 26.401367,5.893555 25.971680,8.676758 L 25.917969,8.676758 L 25.917969,55.488281 L 6.245117,55.488281 C 4.277344,55.488281 3.627930,56.726074 4.829102,58.239746 L 37.968750,100.212402 C 38.520508,100.932617 39.267578,101.298828 40.019531,101.335449 L 7.314453,59.904785 Z"/>

</Canvas>

</Canvas>

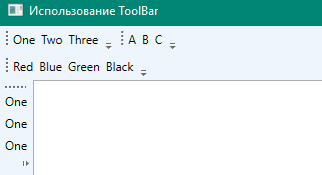
</Canvas>

</Canvas>

</Viewbox>

</ResourceDictionary>

## ToolBarTrays



<DockPanel>

<ToolBarTray DockPanel.Dock="Top">

<ToolBar>

<!--ToolBar.OverflowMode="Never" кнопки на тулбаре будут перекрываться, но не прятаться-->

<Button>One</Button>

<Button ToolBar.OverflowMode="Never">Two</Button>

<Button>Three</Button>

</ToolBar>

<ToolBar>

<Button ToolBar.OverflowMode="Never">A</Button>

<Button ToolBar.OverflowMode="Never">B</Button>

<Button>C</Button>

</ToolBar>

<ToolBar Band="1">

<Button>Red</Button>

<Button>Blue</Button>

<Button>Green</Button>

<Button>Black</Button>

</ToolBar>

</ToolBarTray>

<ToolBarTray DockPanel.Dock="Left" Orientation="Vertical">

<ToolBar>

<Button>One</Button>

<Button>One</Button>

<Button>One</Button>

</ToolBar>

</ToolBarTray>

<TextBox></TextBox>

</DockPanel>

## StatusBar

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<StatusBar Grid.Row="1">

<TextBlock>Status</TextBlock>

<ProgressBar Value="5" Width="100" Maximum="10" Height="10"></ProgressBar>

</StatusBar>

</Grid>

# DocumentsPrinting

## FollowDocument

<FlowDocumentScrollViewer Name="docViewer" IsToolBarVisible="True">

<FlowDocument>

<!--BlockUIContainer - используется для добавление в документ объектов наследников UIElement-->

<BlockUIContainer>

<Button Click="cmdCreateDynamicDocument\_Click" HorizontalAlignment="Left" Padding="5">Создать динамический документ</Button>

</BlockUIContainer>

<Paragraph FontSize="20">Самые большие города в 100 году</Paragraph>

<Table>

<Table.Columns>

<TableColumn Width="\*"></TableColumn>

<TableColumn Width="3\*"></TableColumn>

<TableColumn Width="\*"></TableColumn>

</Table.Columns>

<TableRowGroup>

<TableRow FontWeight="Bold" >

<TableCell >

<Paragraph>Позиция</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Город</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Население</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>1</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Rome</Paragraph>

</TableCell>

<TableCell>

<Paragraph>450,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>2</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Luoyang (Honan), China</Paragraph>

</TableCell>

<TableCell>

<Paragraph>420,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>3</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Seleucia (on the Tigris), Iraq</Paragraph>

</TableCell>

<TableCell>

<Paragraph>250,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>4</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Alexandria, Egypt</Paragraph>

</TableCell>

<TableCell>

<Paragraph>250,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>5</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Antioch, Turkey</Paragraph>

</TableCell>

<TableCell>

<Paragraph>150,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>6</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Anuradhapura, Sri Lanka</Paragraph>

</TableCell>

<TableCell>

<Paragraph>130,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>7</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Peshawar, Pakistan</Paragraph>

</TableCell>

<TableCell>

<Paragraph>120,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>8</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Carthage, Tunisia</Paragraph>

</TableCell>

<TableCell>

<Paragraph>100,000</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>9</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Suzhou, China</Paragraph>

</TableCell>

<TableCell>

<Paragraph>n/a</Paragraph>

</TableCell>

</TableRow>

<TableRow>

<TableCell>

<Paragraph>10</Paragraph>

</TableCell>

<TableCell>

<Paragraph>Smyrna, Turkey</Paragraph>

</TableCell>

<TableCell>

<Paragraph>90,000</Paragraph>

</TableCell>

</TableRow>

</TableRowGroup>

</Table>

</FlowDocument>

</FlowDocumentScrollViewer>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdCreateDynamicDocument\_Click(object sender, RoutedEventArgs e)

{

// Создание первой части предложения.

Run runFirst = new Run();

runFirst.Text = "Hello world ";

// Создание текста с полужирным начертанием.

Bold bold = new Bold();

Run runBold = new Run();

runBold.Text = "динамически сгенерированный";

bold.Inlines.Add(runBold);

// Создание второй части приложения.

Run runLast = new Run();

runLast.Text = " документ";

// Добавление предложений в параграф

Paragraph paragraph = new Paragraph();

paragraph.Inlines.Add(runFirst);

paragraph.Inlines.Add(bold);

paragraph.Inlines.Add(runLast);

// Создание документа и добавление параграфа.

FlowDocument document = new FlowDocument();

document.Blocks.Add(paragraph);

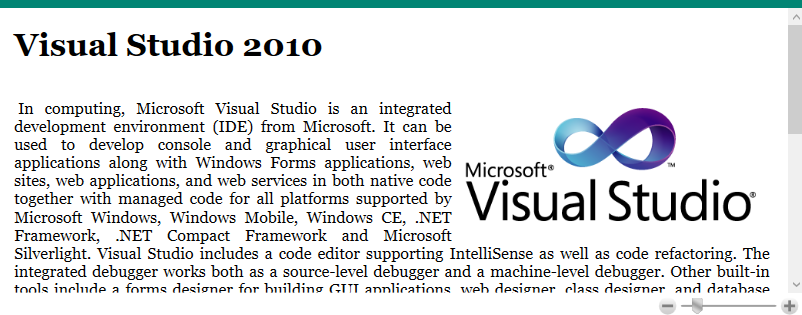
// Отображение документа.

docViewer.Document = document;

}

}

## Paragraphs



<Window.Resources>

<Style x:Key="PullQuote">

<Setter Property="Paragraph.FontSize" Value="20" ></Setter>

<Setter Property="Paragraph.FontStyle" Value="Italic"></Setter>

<Setter Property="Paragraph.Foreground" Value="Blue"></Setter>

<Setter Property="Paragraph.Padding" Value="5"></Setter>

<Setter Property="Paragraph.Margin" Value="5,10,15,10"></Setter>

</Style>

</Window.Resources>

<FlowDocumentScrollViewer Name="docViewer" MinZoom="1" MaxZoom="1000" IsToolBarVisible="True">

<FlowDocument>

<Paragraph FontWeight="Bold" FontSize="24pt">Visual Studio 2010</Paragraph>

<Paragraph>

<Floater Width="300" Padding="5,0,5,0" HorizontalAlignment="Right">

<BlockUIContainer>

<Image Source="logo.png"></Image>

</BlockUIContainer>

</Floater>

Text

</Paragraph>

<Paragraph>

<Floater Style="{StaticResource PullQuote}" Width="205" HorizontalAlignment="Left">

<Paragraph>http://en.wikipedia.org/wiki/Visual\_studio</Paragraph>

</Floater>

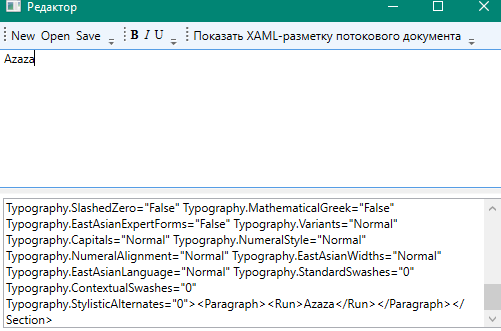
Text

</Paragraph>

</FlowDocument>

</FlowDocumentScrollViewer>

## TextEditor



<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<ToolBarTray>

<ToolBar>

<Button Click="cmdNew\_Click">New</Button>

<Button Click="cmdOpen\_Click">Open</Button>

<Button Click="cmdSave\_Click">Save</Button>

</ToolBar>

<ToolBar FontFamily="Times New Roman">

<Button FontWeight="Bold" Command="EditingCommands.ToggleBold">B</Button>

<!--Кнопки с командами для редактирования стиля текста.-->

<Button FontStyle="Italic" Command="EditingCommands.ToggleItalic">I</Button>

<Button Command="EditingCommands.ToggleUnderline">U</Button>

</ToolBar>

<ToolBar>

<Button Click="cmdShowXAML\_Click">Показать XAML-разметку потокового документа</Button>

</ToolBar>

</ToolBarTray>

<!--RichTextBox для редактирования потокового документа. Хранит свое содержимое в FlowDocument-->

<RichTextBox Name="richTextBox" Grid.Row="1">

<FlowDocument>

<Paragraph FontSize="20" FontWeight="Bold" TextDecorations="Underline">

Chapter 1

</Paragraph>

<Paragraph>

It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

</Paragraph>

<Paragraph>

The hallway smelt of boiled cabbage and old rag mats. At one end of it a coloured poster, too large for indoor display, had been tacked to the wall. It depicted simply an enormous face, more than a metre wide: the face of a man of about forty-five, with a heavy black moustache and ruggedly handsome features. Winston made for the stairs. It was no use trying the lift. Even at the best of times it was seldom working, and at present the electric current was cut off during daylight hours. It was part of the economy drive in preparation for Hate Week. The flat was seven flights up, and Winston, who was thirty-nine and had a varicose ulcer above his right ankle, went slowly, resting several times on the way. On each landing, opposite the lift-shaft, the poster with the enormous face gazed from the wall. It was one of those pictures which are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption beneath it ran.

</Paragraph>

</FlowDocument>

</RichTextBox>

<GridSplitter Grid.Row="2" Height="5" HorizontalAlignment="Stretch" VerticalAlignment="Center"></GridSplitter>

<!--Поле ввода для редактирования XAML разметки-->

<TextBox Grid.Row="3" TextWrapping="Wrap" VerticalScrollBarVisibility="Visible" Margin="5" Name="txtFlowDocumentMarkup"></TextBox>

</Grid>

namespace ReachTextBoxEditor

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdShowXAML\_Click(object sender, RoutedEventArgs e)

{

UpdateMarkupDisplay();

}

private void UpdateMarkupDisplay()

{

// TextRange - Представляет выделение содержимого между двумя позициями TextPointer.

TextRange range;

range = new TextRange(richTextBox.Document.ContentStart, richTextBox.Document.ContentEnd);

MemoryStream stream = new MemoryStream();

range.Save(stream, DataFormats.Xaml);

stream.Position = 0;

StreamReader r = new StreamReader(stream);

txtFlowDocumentMarkup.Text = r.ReadToEnd();

r.Close();

stream.Close();

}

private void cmdOpen\_Click(object sender, RoutedEventArgs e)

{

// Запускаем диалоговое окно для открытия файлов с расширением XAML или RTF

OpenFileDialog openFile = new OpenFileDialog();

openFile.Filter = "XAML Files (\*.xaml)|\*.xaml|RichText Files (\*.rtf)|\*.rtf|All Files (\*.\*)|\*.\*";

if (openFile.ShowDialog() == true)

{

TextRange documentTextRange = new TextRange(richTextBox.Document.ContentStart, richTextBox.Document.ContentEnd);

using (FileStream fs = File.Open(openFile.FileName, FileMode.Open))

{

if (System.IO.Path.GetExtension(openFile.FileName).ToLower() == ".rtf")

{

documentTextRange.Load(fs, DataFormats.Rtf);

}

else

{

documentTextRange.Load(fs, DataFormats.Xaml);

}

}

}

}

private void cmdSave\_Click(object sender, RoutedEventArgs e)

{

// Диалоговое окно для сохранения файлов.

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Filter = "XAML Files (\*.xaml)|\*.xaml|RichText Files (\*.rtf)|\*.rtf|All Files (\*.\*)|\*.\*";

if (saveFile.ShowDialog() == true)

{

// TextRange - Представляет выделение содержимого между двумя позициями TextPointer.

TextRange documentTextRange = new TextRange(richTextBox.Document.ContentStart, richTextBox.Document.ContentEnd);

// Если файл существует он будет перезаписан.

using (FileStream fs = File.Create(saveFile.FileName))

{

if (System.IO.Path.GetExtension(saveFile.FileName).ToLower() == ".rtf")

{

documentTextRange.Save(fs, DataFormats.Rtf);

}

else

{

documentTextRange.Save(fs, DataFormats.Xaml);

}

}

}

}

private void cmdNew\_Click(object sender, RoutedEventArgs e)

{

// Создаем новый потоковый документ в RichTextBox

richTextBox.Document = new FlowDocument();

}

}

## PrintFlowDocument

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<FlowDocumentReader Name="docReader">

<FlowDocument Background="LightYellow" IsOptimalParagraphEnabled="True" IsHyphenationEnabled="True" >

<Paragraph>

It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

</Paragraph>

<Paragraph>

The hallway smelt of boiled cabbage and old rag mats. At one end of it a coloured poster, too large for indoor display, had been tacked to the wall. It depicted simply an enormous face, more than a metre wide: the face of a man of about forty-five, with a heavy black moustache and ruggedly handsome features. Winston made for the stairs. It was no use trying the lift. Even at the best of times it was seldom working, and at present the electric current was cut off during daylight hours. It was part of the economy drive in preparation for Hate Week. The flat was seven flights up, and Winston, who was thirty-nine and had a varicose ulcer above his right ankle, went slowly, resting several times on the way. On each landing, opposite the lift-shaft, the poster with the enormous face gazed from the wall. It was one of those pictures which are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption beneath it ran.

</Paragraph>

<Paragraph>

Inside the flat a fruity voice was reading out a list of figures which had something to do with the production of pig-iron. The voice came from an oblong metal plaque like a dulled mirror which formed part of the surface of the right-hand wall. Winston turned a switch and the voice sank somewhat, though the words were still distinguishable. The instrument (the telescreen, it was called) could be dimmed, but there was no way of shutting it off completely. He moved over to the window: a smallish, frail figure, the meagreness of his body merely emphasized by the blue overalls which were the uniform of the party. His hair was very fair, his face naturally sanguine, his skin roughened by coarse soap and blunt razor blades and the cold of the winter that had just ended.

</Paragraph>

<Paragraph>

Outside, even through the shut window-pane, the world looked cold. Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemed to be no colour in anything, except the posters that were plastered everywhere. The blackmoustachio'd face gazed down from every commanding corner. There was one on the house-front immediately opposite. BIG BROTHER IS WATCHING YOU, the caption said, while the dark eyes looked deep into Winston's own. Down at streetlevel another poster, torn at one corner, flapped fitfully in the wind, alternately covering and uncovering the single word INGSOC. In the far distance a helicopter skimmed down between the roofs, hovered for an instant like a bluebottle, and darted away again with a curving flight. It was the police patrol, snooping into people's windows. The patrols did not matter, however. Only the Thought Police mattered.

</Paragraph>

<Paragraph>

Behind Winston's back the voice from the telescreen was still babbling away about pig-iron and the overfulfilment of the Ninth Three-Year Plan. The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it, moreover, so long as he remained within the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even conceivable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live -- did live, from habit that became instinct -- in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinized.

</Paragraph>

<Paragraph>

Winston kept his back turned to the telescreen. It was safer, though, as he well knew, even a back can be revealing. A kilometre away the Ministry of Truth, his place of work, towered vast and white above the grimy landscape. This, he thought with a sort of vague distaste -- this was London, chief city of Airstrip One, itself the third most populous of the provinces of Oceania. He tried to squeeze out some childhood memory that should tell him whether London had always been quite like this. Were there always these vistas of rotting nineteenth-century houses, their sides shored up with baulks of timber, their windows patched with cardboard and their roofs with corrugated iron, their crazy garden walls sagging in all directions? And the bombed sites where the plaster dust swirled in the air and the willow-herb straggled over the heaps of rubble; and the places where the bombs had cleared a larger patch and there had sprung up sordid colonies of wooden dwellings like chicken-houses? But it was no use, he could not remember: nothing remained of his childhood except a series of bright-lit tableaux occurring against no background and mostly unintelligible.

</Paragraph>

<Paragraph>

The Ministry of Truth -- Minitrue, in Newspeak -- was startlingly different from any other object in sight. It was an enormous pyramidal structure of glittering white concrete, soaring up, terrace after terrace, 300 metres into the air. From where Winston stood it was just possible to read, picked out on its white face in elegant lettering, the three slogans of the Party:

</Paragraph>

<Paragraph>

WAR IS PEACE

</Paragraph>

<Paragraph>

FREEDOM IS SLAVERY

</Paragraph>

<Paragraph>

IGNORANCE IS STRENGTH

</Paragraph>

<Paragraph>

The Ministry of Truth contained, it was said, three thousand rooms above ground level, and corresponding ramifications below. Scattered about London there were just three other buildings of similar appearance and size. So completely did they dwarf the surrounding architecture that from the roof of Victory Mansions you could see all four of them simultaneously. They were the homes of the four Ministries between which the entire apparatus of government was divided. The Ministry of Truth, which concerned itself with news, entertainment, education, and the fine arts. The Ministry of Peace, which concerned itself with war. The Ministry of Love, which maintained law and order. And the Ministry of Plenty, which was responsible for economic affairs. Their names, in Newspeak: Minitrue, Minipax, Miniluv, and Miniplenty.

</Paragraph>

<Paragraph>

The Ministry of Love was the really frightening one. There were no windows in it at all. Winston had never been inside the Ministry of Love, nor within half a kilometre of it. It was a place impossible to enter except on official business, and then only by penetrating through a maze of barbed-wire entanglements, steel doors, and hidden machine-gun nests. Even the streets leading up to its outer barriers were roamed by gorilla-faced guards in black uniforms, armed with jointed truncheons.

</Paragraph>

</FlowDocument>

</FlowDocumentReader>

<Button Click="cmdPrint\_Click" Margin="3" Padding="3" Grid.Row="1">Распечатать с текущими настройками страниц</Button>

<Button Click="cmdPrintCustom\_Click" Margin="3" Padding="3" Grid.Row="2">Напечатать в две колонки</Button>

</Grid>

namespace PrintFlowDocument

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdPrint\_Click(object sender, RoutedEventArgs e)

{

PrintDialog printDialog = new PrintDialog();

if (printDialog.ShowDialog() == true)

{

// для печати потоковых документов используется метод PrintDocument

printDialog.PrintDocument(((IDocumentPaginatorSource)docReader.Document).DocumentPaginator, "A Flow Document");

}

}

private void cmdPrintCustom\_Click(object sender, RoutedEventArgs e)

{

PrintDialog printDialog = new PrintDialog();

if (printDialog.ShowDialog() == true)

{

FlowDocument doc = docReader.Document;

// Сохранить все существующие настройки.

double pageHeight = doc.PageHeight;

double pageWidth = doc.PageWidth;

Thickness pagePadding = doc.PagePadding;

double columnGap = doc.ColumnGap;

double columnWidth = doc.ColumnWidth;

// Подгоняем размеры FlowDocument под размеры страницы для печати

doc.PageHeight = printDialog.PrintableAreaHeight;

doc.PageWidth = printDialog.PrintableAreaWidth;

doc.PagePadding = new Thickness(50);

// Применяем две колонки.

doc.ColumnGap = 25; // Пробелы между колонками.

doc.ColumnWidth = (doc.PageWidth - doc.ColumnGap

- doc.PagePadding.Left - doc.PagePadding.Right) / 2;

printDialog.PrintDocument(((IDocumentPaginatorSource)doc).DocumentPaginator, "A Flow Document");

// Возобновляем старые настройки.

doc.PageHeight = pageHeight;

doc.PageWidth = pageWidth;

doc.PagePadding = pagePadding;

doc.ColumnGap = columnGap;

doc.ColumnWidth = columnWidth;

}

}

}

# Animation

## Animation

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Button Name="cmdGrow" Padding="10" Click="cmdGrow\_Click" Height="40" Width="160"

HorizontalAlignment="Center" VerticalAlignment="Center">

Увеличение размеров кнопки

</Button>

<Button Name="cmdShrink" Grid.Row="1" Padding="10" Click="cmdShrink\_Click"

HorizontalAlignment="Center" VerticalAlignment="Center">

Вернуть в исходное состояние

</Button>

<Button Name="cmdGrowIncrementally" Grid.Row="2" Padding="10" Click="cmdGrowIncrementally\_Click"

HorizontalAlignment="Center" VerticalAlignment="Center" Width="240">

Увеличение размеров кнопки (Поэтапно)

</Button>

</Grid>

namespace SimpleAnimation

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdGrow\_Click(object sender, RoutedEventArgs e)

{

// Если не устанавливать значение From его значение будет равно текущему значению свойства.

DoubleAnimation widthAnimation = new DoubleAnimation();

widthAnimation.To = this.Width - 30;

widthAnimation.Duration = TimeSpan.FromSeconds(5);

widthAnimation.Completed += animation\_Completed; // событие при завершении анимации.

// Анимация является временной, а это значит, что в действительности она не меняет

// значения свойств. Пока анимация активна, она просто перекрывает значения свойств.

DoubleAnimation heightAnimation = new DoubleAnimation();

heightAnimation.To = (this.Height - 50) / 3;

heightAnimation.Duration = TimeSpan.FromSeconds(5);

cmdGrow.BeginAnimation(Button.WidthProperty, widthAnimation);

cmdGrow.BeginAnimation(Button.HeightProperty, heightAnimation);

}

private void animation\_Completed(object sender, EventArgs e)

{

MessageBox.Show("Completed!");

// Запоминаем свойства, которые анимировались.

//double currentWidth = cmdGrow.Width;

//double currentHeight = cmdGrow.Height;

//cmdGrow.BeginAnimation(Button.WidthProperty, null);

//cmdGrow.BeginAnimation(Button.HeightProperty, null);

//cmdGrow.Width = currentWidth;

//cmdGrow.Height = currentHeight;

}

private void cmdShrink\_Click(object sender, RoutedEventArgs e)

{

// Если не устанавливать значения свойств:

// From - всегда принимает текущее значение свойства.

// To - принимает значение, которое было последний раз установлено из кода или в разметке.

DoubleAnimation widthAnimation = new DoubleAnimation();

widthAnimation.Duration = TimeSpan.FromSeconds(5);

DoubleAnimation heightAnimation = new DoubleAnimation();

heightAnimation.Duration = TimeSpan.FromSeconds(5);

cmdGrow.BeginAnimation(Button.WidthProperty, widthAnimation);

cmdGrow.BeginAnimation(Button.HeightProperty, heightAnimation);

}

private void cmdGrowIncrementally\_Click(object sender, RoutedEventArgs e)

{

DoubleAnimation widthAnimation = new DoubleAnimation();

widthAnimation.By = 10; // Увеличение на 10 единиц.

widthAnimation.Duration = TimeSpan.FromSeconds(0.5);

cmdGrowIncrementally.BeginAnimation(Button.WidthProperty, widthAnimation);

}

}

}

## XAMLAnimation

<Window x:Class="XAMLAnimation.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Настройки анимации в XAML разметке" Height="350" Width="525">

<Button Padding="10" Name="cmdGrow"

Height="40" Width="160"

HorizontalAlignment="Center" VerticalAlignment="Center">

<Button.Triggers>

<!--Триггер сработает на событие-->

<EventTrigger RoutedEvent="Button.Click">

<EventTrigger.Actions>

<BeginStoryboard>

<Storyboard>

<DoubleAnimation Storyboard.TargetProperty="Width"

To="300"

Duration="0:0:1">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger.Actions>

</EventTrigger>

</Button.Triggers>

<Button.Content>

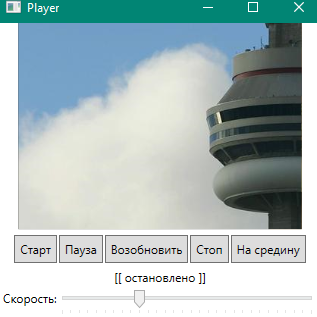
Нажать

</Button.Content>

</Button>

</Window>

## ChangePhoto



<Window.Triggers>

<!--Привязка скорости воспроизведения к слайдеру на окне-->

<EventTrigger SourceName="cmdStart" RoutedEvent="Button.Click">

<BeginStoryboard Name="fadeStoryboardBegin">

<Storyboard Name="fadeStoryboard"

CurrentTimeInvalidated="storyboard\_CurrentTimeInvalidated"

SpeedRatio="{Binding ElementName=sldSpeed, Path=Value}">

<DoubleAnimation Storyboard.TargetName="imgDay"

Storyboard.TargetProperty="Opacity"

From="1" To="0" Duration="0:0:10">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

<!--Управление воспроизведением анимации-->

<EventTrigger SourceName="cmdPause" RoutedEvent="Button.Click">

<PauseStoryboard BeginStoryboardName="fadeStoryboardBegin"></PauseStoryboard>

</EventTrigger>

<EventTrigger SourceName="cmdResume" RoutedEvent="Button.Click">

<ResumeStoryboard BeginStoryboardName="fadeStoryboardBegin"></ResumeStoryboard>

</EventTrigger>

<EventTrigger SourceName="cmdStop" RoutedEvent="Button.Click">

<StopStoryboard BeginStoryboardName="fadeStoryboardBegin"></StopStoryboard>

</EventTrigger>

<EventTrigger SourceName="cmdMiddle" RoutedEvent="Button.Click">

<SeekStoryboard BeginStoryboardName="fadeStoryboardBegin" Offset="0:0:5"></SeekStoryboard>

</EventTrigger>

</Window.Triggers>

<!--Ресурс со стилями для кнопок-->

<Window.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Padding" Value="5"></Setter>

<Setter Property="Margin" Value="1"></Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid>

<Image Source="night.jpg"></Image>

<Image Source="day.jpg" Name="imgDay"></Image>

</Grid>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" Margin="5">

<Button Name="cmdStart">Старт</Button>

<Button Name="cmdPause">Пауза</Button>

<Button Name="cmdResume">Возобновить</Button>

<Button Name="cmdStop">Стоп</Button>

<Button Name="cmdMiddle">На средину</Button>

</StackPanel>

<TextBlock Grid.Row="2" Name="lblTime" HorizontalAlignment="Center">[[ остановлено ]]</TextBlock>

<Grid Grid.Row="3" Margin="5">

<Grid.ColumnDefinitions>

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

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<TextBlock>Скорость:</TextBlock>

<Slider Grid.Column="1"

Name="sldSpeed"

Minimum="0.1"

Maximum="3"

Value="1"

TickPlacement="BottomRight"

TickFrequency="0.1"

ValueChanged="sldSpeed\_ValueChanged"></Slider>

</Grid>

<ProgressBar Grid.Row="4"

Margin="0,5,0,0"

Height="10"

Name="progressBar"

Minimum="0"

Maximum="1"></ProgressBar>

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void storyboard\_CurrentTimeInvalidated(object sender, EventArgs e)

{

// Sender - объект clock, который бул создан для storyboard.

Clock storyboardClock = (Clock)sender;

if (storyboardClock.CurrentProgress == null)

{

lblTime.Text = "[[ остановлено ]]";

progressBar.Value = 0;

}

else

{

lblTime.Text = storyboardClock.CurrentTime.ToString();

progressBar.Value = (double)storyboardClock.CurrentProgress;

}

}

private void sldSpeed\_ValueChanged(object sender, RoutedEventArgs e)

{

fadeStoryboard.SetSpeedRatio(this, sldSpeed.Value);

}

}

## FPSAnimation

<Window.Resources>

<BeginStoryboard x:Key="beginStoryboard">

<!--DesiredFrameRate частота кадров в секунду, по умолчанию = 60-->

<Storyboard Timeline.DesiredFrameRate="{Binding ElementName=txtFrameRate, Path=Text}">

<!--Если анимированное свойство не принадлежит целевому объекту его надо взять в скобки. Например, (Canvas.Left)-->

<DoubleAnimation Storyboard.TargetName="ellipse" Storyboard.TargetProperty="(Canvas.Left)"

From="0" To="300" Duration="0:0:5">

</DoubleAnimation>

<!--AutoReverse="True" анимация будет проигрываться в обратном направлении-->

<DoubleAnimation Storyboard.TargetName="ellipse" Storyboard.TargetProperty="(Canvas.Top)"

From="300" To="0" AutoReverse="True" Duration="0:0:2.5"

DecelerationRatio="1">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</Window.Resources>

<Window.Triggers>

<!--Запускаем анимацию при загрузке окна-->

<EventTrigger RoutedEvent="Window.Loaded">

<EventTrigger.Actions>

<!--Указываем анимацию из ресурсов.-->

<StaticResource ResourceKey="beginStoryboard"></StaticResource>

</EventTrigger.Actions>

</EventTrigger>

</Window.Triggers>

<Grid Background="LightGoldenrodYellow" >

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Border Background="White" BorderBrush="DarkGray" BorderThickness="3" Width="300" Height="300" HorizontalAlignment="Center" VerticalAlignment="Center">

<!--ClipToBounds="True" указывает на то, что все элементы, которые на помещаются на канвас отображаться не будут-->

<Canvas ClipToBounds="True">

<Ellipse Name="ellipse"

Fill="Red"

Width="10"

Height="10"></Ellipse>

</Canvas>

</Border>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<TextBlock VerticalAlignment="Center" xml:space="preserve">Частота кадров: </TextBlock>

<TextBox Grid.Column="2" Width="50" Name="txtFrameRate">60</TextBox>

</StackPanel>

<Button Grid.Row="2" HorizontalAlignment="Center" Padding="3" Margin="3">

<Button.Triggers>

<EventTrigger RoutedEvent="Button.Click">

<StaticResource ResourceKey="beginStoryboard"></StaticResource>

</EventTrigger>

</Button.Triggers>

<Button.Content>

Повторить

</Button.Content>

</Button>

</Grid>

## SpiningButtons

<Window.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="HorizontalAlignment" Value="Center"></Setter>

<Setter Property="Padding" Value="20,15"></Setter>

<Setter Property="Margin" Value="2"></Setter>

<Setter Property="LayoutTransform">

<Setter.Value>

<RotateTransform></RotateTransform>

</Setter.Value>

</Setter>

<Style.Triggers>

<EventTrigger RoutedEvent="Button.MouseEnter">

<EventTrigger.Actions>

<BeginStoryboard Name="rotateStoryboardBegin">

<Storyboard>

<!--LayoutTransform анимация применяется на первом этапе компоновки еще до того как произведется замер элементов управления-->

<DoubleAnimation Storyboard.TargetProperty="LayoutTransform.Angle"

To="360" Duration="0:0:0.8"

RepeatBehavior="Forever">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger.Actions>

</EventTrigger>

<EventTrigger RoutedEvent="Button.MouseLeave">

<EventTrigger.Actions>

<!-- <RemoveStoryboard BeginStoryboardName="rotateStoryboardBegin"></RemoveStoryboard> -->

<BeginStoryboard>

<Storyboard>

<DoubleAnimation Storyboard.TargetProperty="LayoutTransform.Angle" Duration="0:0:5">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger.Actions>

</EventTrigger>

</Style.Triggers>

</Style>

</Window.Resources>

<StackPanel Margin="5" Button.Click="cmd\_Clicked">

<Button>One</Button>

<Button>Two</Button>

<Button>Three</Button>

<Button>Four</Button>

<TextBlock Name="lbl" Margin="5"></TextBlock>

</StackPanel>

# MediaUserControl

## CheckPhone

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="271\*" />

</Grid.RowDefinitions>

<!--Для работы элемента WindowsFormsHost в проекте должна быть ссылка на сборку WindowsFormsIntegration-->

<WindowsFormsHost Margin="10" Name="host">

<wf:MaskedTextBox x:Name="maskedTextBox"

Mask="(999)-000-0000"

MaskInputRejected="maskedTextBox\_MaskInputRejected">

</wf:MaskedTextBox>

</WindowsFormsHost>

<Label Margin="5" Name="lblErrorText" Grid.Row="1"></Label>

</Grid>

/\*

\* 0 - Обязательная цифра.

\* 9 - Необязательная цифра.

\* # - Необязательная десятичная цифра, пробел или знак плюс-минус. Если остается пустым пробел вставляется автоматически

\* L - Обязательный символ ASCII (a-z или A-Z)

\* ? - Необязательный символ ASCII (a-z или A-Z)

\* & - Oбязательный Unicode

\* С - Необязательный символ Unicode

\* > - все что после преобразуется в верхний регистр

\* < - Все что после преобразуется в нижний регистр

\*/

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void maskedTextBox\_MaskInputRejected(object sender, System.Windows.Forms.MaskInputRejectedEventArgs e)

{

lblErrorText.Content = "Error: " + e.RejectionHint.ToString();

}

}

## SoundPlayer

<StackPanel Margin="5">

<Button Click="cmdPlayAudio\_Click">Воспроизвести аудио синхронно</Button>

<Button Click="cmdPlayAudioAsync\_Click">Воспроизвести аудио асинхронно</Button>

<Button>

<Button.Content>Проигрывание из XAML</Button.Content>

<Button.Style>

<Style>

<Style.Triggers>

<EventTrigger RoutedEvent="Button.Click">

<EventTrigger.Actions>

<!--Декларативный подход при указании проигрываемого файла-->

<SoundPlayerAction Source="test.wav"></SoundPlayerAction>

</EventTrigger.Actions>

</EventTrigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

<Button Click="cmdPlayWithMediaPlayer\_Click">Проигрывание через MediaPlayer (mp3)</Button>

</StackPanel>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdPlayAudio\_Click(object sender, RoutedEventArgs e)

{

SoundPlayer player = new SoundPlayer();

// Файл находится в ресурсах приложения.

player.Stream = Properties.Resources.chord;

try

{

player.Load();

player.PlaySync(); // Синхронное воспроизведение, в этом же потоке.

//MessageBox.Show("Test");

}

catch (FileNotFoundException)

{

MessageBox.Show("Файл не найден");

}

catch (FormatException)

{

MessageBox.Show("Не верный формат аудио.");

}

}

private void cmdPlayAudioAsync\_Click(object sender, RoutedEventArgs e)

{

SoundPlayer player = new SoundPlayer();

player.SoundLocation = "test.wav";

try

{

player.Load();

player.Play(); // асинхронное воспроизведение в отдельном потоке.

}

catch (FileNotFoundException)

{

MessageBox.Show("Файл не найден");

}

catch (FormatException)

{

MessageBox.Show("Не верный формат аудио.");

}

}

// Объект MediaPlayer будет работать только в том случае, если в системе установлен Windows Media Player 10 или выше

private MediaPlayer player = new MediaPlayer();

private void cmdPlayWithMediaPlayer\_Click(object sender, RoutedEventArgs e)

{

player.MediaFailed += new EventHandler<ExceptionEventArgs>(player\_MediaFailed);

player.Open(new Uri("test.mp3", UriKind.Relative));

player.Play(); // Метод Play не выбрасывает исключений, для отлова ошибок нужно делать обработчики на события

}

void player\_MediaFailed(object sender, ExceptionEventArgs e)

{

MessageBox.Show("Ошибка во время открытия файла.");

}

private void window\_Closed(object sender, EventArgs e)

{

player.Close(); // При закрытии окна освобождаем объект MediaPlayer

}

}

## PlayerToggles

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<!--LoadedBehavior="Manual" Для воспроизведения аудио вручную из кода.-->

<!--UnloadedBehavior="Close" закрытие файла и освобождение ресурсов по завершению проигрыша.-->

<MediaElement Name="media"

LoadedBehavior="Manual"

UnloadedBehavior="Close"

Source="test.mp3"

MediaOpened="media\_MediaOpened">

</MediaElement>

<StackPanel Orientation="Horizontal">

<Button Click="cmdPlay\_Click" Padding="5" Margin="1">Play</Button>

<Button Click="cmdStop\_Click" Padding="5" Margin="1">Stop</Button>

<Button Click="cmdPause\_Click" Padding="5" Margin="1">Pause</Button>

</StackPanel>

<Grid Grid.Row="1" Margin="0,10,0,0">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

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

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<TextBlock Margin="5">Volume:</TextBlock>

<Slider Grid.Column="1" Minimum="0" Maximum="1"

Value="{Binding ElementName=media, Path=Volume, Mode=TwoWay}">

</Slider>

<TextBlock Grid.Row="1" Margin="5">Balance:</TextBlock>

<Slider Grid.Row="1" Grid.Column="1" Minimum="-1" Maximum="1"

Value="{Binding ElementName=media, Path=Balance, Mode=TwoWay}">

</Slider>

<TextBlock Grid.Row="2" Margin="5">Speed:</TextBlock>

<Slider Grid.Column="1" Grid.Row="2" Minimum="0" Maximum="2"

Value="{Binding ElementName=media, Path=SpeedRatio}">

</Slider>

<TextBlock Grid.Row="3" Margin="5,20,5,5">Seek To:</TextBlock>

<Slider Minimum="0" Grid.Column="1" Grid.Row="3" Name="sliderPosition"

Margin="0,20,0,0" TickPlacement="BottomRight" TickFrequency="1"

ValueChanged="sliderPosition\_ValueChanged">

</Slider>

</Grid>

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdPlay\_Click(object sender, RoutedEventArgs e)

{

media.Play();

}

private void cmdPause\_Click(object sender, RoutedEventArgs e)

{

media.Pause();

}

private void cmdStop\_Click(object sender, RoutedEventArgs e)

{

media.Stop();

media.SpeedRatio = 1;

}

private void media\_MediaOpened(object sender, RoutedEventArgs e)

{

// Получение длины дорожки.

sliderPosition.Maximum = media.NaturalDuration.TimeSpan.TotalSeconds;

}

private void sliderPosition\_ValueChanged(object sender, RoutedEventArgs e)

{

media.Position = TimeSpan.FromSeconds(sliderPosition.Value);

}

}

## SpeechSynthesis

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<TextBox Name="txtWords" Margin="5"

ScrollViewer.HorizontalScrollBarVisibility="Visible"

TextWrapping="Wrap">Hello, world

</TextBox>

<Button Margin="5" Grid.Row="1" Click="cmdSpeak\_Click">Воспроизвести</Button>

<Button Margin="5" Grid.Row="2" Click="cmdPromptTest\_Click">Тест</Button>

</Grid>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdSpeak\_Click(object sender, RoutedEventArgs e)

{

// Воспроизведение речи.

SpeechSynthesizer synthesizer = new SpeechSynthesizer();

synthesizer.Speak(txtWords.Text);

}

private void cmdPromptTest\_Click(object sender, RoutedEventArgs e)

{

// Воспроизведение речи с указанием ударений и интонации.

PromptBuilder prompt = new PromptBuilder();

prompt.AppendText("How are you");

prompt.AppendBreak(TimeSpan.FromSeconds(2));

prompt.AppendText("How ", PromptEmphasis.Reduced);

PromptStyle style = new PromptStyle();

style.Rate = PromptRate.ExtraSlow;

style.Emphasis = PromptEmphasis.Strong;

prompt.StartStyle(style);

prompt.AppendText("are ");

prompt.EndStyle();

prompt.AppendText("you?");

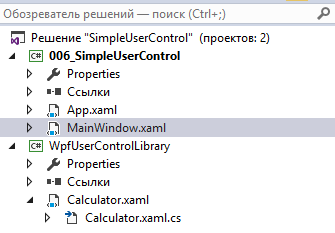
SpeechSynthesizer synthesizer = new SpeechSynthesizer();

synthesizer.Speak(prompt);

}

}

## UserControl



<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="35\*" />

<RowDefinition Height="276\*" />

</Grid.RowDefinitions>

<my:Calculator HorizontalAlignment="Left"

Margin="83,44,0,0"

Name="calculator1"

VerticalAlignment="Top"

Grid.Row="1" />

</Grid>

UserCotrol library

<UserControl x:Class="WpfUserControlLibrary.Calculator"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

d:DesignHeight="70" d:DesignWidth="300">

<StackPanel>

<StackPanel Orientation="Horizontal">

<TextBox Margin="4" Name="Operand1TextBox" Width="75"></TextBox>

<TextBlock Margin="4">+</TextBlock>

<TextBox Margin="4" Name="Operand2TextBox" Width="75"></TextBox>

<TextBlock Margin="4">=</TextBlock>

<TextBox Margin="4" Name="ResultTextBox" Width="75"></TextBox>

</StackPanel>

<Button Name="Sum" Padding="4" Margin="4" Click="Sum\_Click">Sum</Button>

</StackPanel>

</UserControl>

public partial class Calculator : UserControl

{

public Calculator()

{

InitializeComponent();

}

private void Sum\_Click(object sender, RoutedEventArgs e)

{

try

{

int a, b;

a = Convert.ToInt32(Operand1TextBox.Text);

b = Convert.ToInt32(Operand2TextBox.Text);

ResultTextBox.Text = (a + b).ToString();

}

catch

{

MessageBox.Show("Error");

}

}

}

## UserControlColorPicker

<Window x:Class="UserControlSample.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:lib="clr-namespace:WpfControlLibrary1;assembly=WpfControlLibrary1"

Title="ColorPicker" Height="350" Width="525">

<StackPanel>

<lib:ColorPickerUserControl

Name="colorPicker"

Margin="2"

Padding="3"

ColorChanged="colorPicker\_ColorChanged"

Color="Yellow">

</lib:ColorPickerUserControl>

<Button Click="cmdGet\_Click" Margin="5,20,5,0" Padding="2">Get Color</Button>

<Button Click="cmdSet\_Click" Margin="5,0,5,0" Padding="2">Reset Color</Button>

<Button Command="Undo" CommandTarget="{Binding ElementName=colorPicker}" Margin="5,0,5,0" Padding="2">Undo</Button>

<TextBlock Name="lblColor" Margin="10"></TextBlock>

</StackPanel>

</Window>

namespace UserControlSample

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void cmdGet\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show(colorPicker.Color.ToString());

}

private void cmdSet\_Click(object sender, RoutedEventArgs e)

{

colorPicker.Color = Colors.Beige;

}

private void colorPicker\_ColorChanged(object sender, RoutedPropertyChangedEventArgs<Color> e)

{

if (lblColor != null)

{

lblColor.Text = "The new color is " + e.NewValue.ToString();

}

}

}

}

WPFControllLibrary

<UserControl x:Class="WpfControlLibrary1.ColorPickerUserControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

mc:Ignorable="d"

d:DesignHeight="300" d:DesignWidth="300" Name="colorPicker">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

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

</Grid.ColumnDefinitions>

<Slider Name="sliderRed" Minimum="0" Maximum="255"

Margin="{Binding ElementName=colorPicker,Path=Padding}"

Value="{Binding ElementName=colorPicker, Path=Red}"></Slider>

<Slider Grid.Row="1" Name="sliderGreen" Minimum="0" Maximum="255"

Margin="{Binding ElementName=colorPicker, Path=Padding}"

Value="{Binding ElementName=colorPicker, Path=Green}"></Slider>

<Slider Grid.Row="2" Name="sliderBlue" Minimum="0" Maximum="255"

Margin="{Binding ElementName=colorPicker, Path=Padding}"

Value="{Binding ElementName=colorPicker, Path=Blue}"></Slider>

<Rectangle Grid.Column="1" Grid.RowSpan="3"

Margin="{Binding ElementName=colorPicker, Path=Padding}"

Width="50" Stroke="Black" StrokeThickness="1">

<Rectangle.Fill>

<SolidColorBrush Color="{Binding ElementName=colorPicker, Path=Color}"></SolidColorBrush>

</Rectangle.Fill>

</Rectangle>

</Grid>

</UserControl>

namespace WpfControlLibrary1

{

public partial class ColorPickerUserControl : UserControl

{

public ColorPickerUserControl()

{

InitializeComponent();

}

private Color? previousColor;

// Проверка возможности запуска команды.

private static void UndoCommand\_CanExecute(object sender, CanExecuteRoutedEventArgs e)

{

ColorPickerUserControl colorPicker = (ColorPickerUserControl)sender;

e.CanExecute = colorPicker.previousColor.HasValue;

}

private static void UndoCommand\_Executed(object sender, ExecutedRoutedEventArgs e)

{

// Возвращаем старое значение цвета.

ColorPickerUserControl colorPicker = (ColorPickerUserControl)sender;

colorPicker.Color = (Color)colorPicker.previousColor;

}

// Конструктор для регистрации свойств зависимостей.

static ColorPickerUserControl()

{

ColorProperty = DependencyProperty.Register("Color", typeof(Color),

typeof(ColorPickerUserControl),

new FrameworkPropertyMetadata(Colors.Black, new PropertyChangedCallback(OnColorChangedCallback)));

RedProperty = DependencyProperty.Register("Red", typeof(byte),

typeof(ColorPickerUserControl),

new FrameworkPropertyMetadata(new PropertyChangedCallback(OnColorRGBChangedCallback)));

GreenProperty = DependencyProperty.Register("Green", typeof(byte),

typeof(ColorPickerUserControl),

new FrameworkPropertyMetadata(new PropertyChangedCallback(OnColorRGBChangedCallback)));

BlueProperty = DependencyProperty.Register("Blue", typeof(byte),

typeof(ColorPickerUserControl),

new FrameworkPropertyMetadata(new PropertyChangedCallback(OnColorRGBChangedCallback)));

CommandManager.RegisterClassCommandBinding(typeof(ColorPickerUserControl),

new CommandBinding(ApplicationCommands.Undo,

UndoCommand\_Executed, UndoCommand\_CanExecute));

}

#region Обертки для свойств зависимостей

public static DependencyProperty ColorProperty;

public static DependencyProperty RedProperty;

public static DependencyProperty GreenProperty;

public static DependencyProperty BlueProperty;

public Color Color

{

get

{

return (Color)GetValue(ColorProperty);

}

set

{

SetValue(ColorProperty, value);

}

}

public byte Red

{

get { return (byte)GetValue(RedProperty); }

set { SetValue(RedProperty, value); }

}

public byte Green

{

get { return (byte)GetValue(GreenProperty); }

set { SetValue(GreenProperty, value); }

}

public byte Blue

{

get { return (byte)GetValue(BlueProperty); }

set { SetValue(BlueProperty, value); }

}

#endregion

// Действия при изменении цвета.

private static void OnColorChangedCallback(DependencyObject sender, DependencyPropertyChangedEventArgs e)

{

ColorPickerUserControl colorPicker = (ColorPickerUserControl)sender;

Color oldColor = (Color)e.OldValue;

Color newColor = (Color)e.NewValue;

colorPicker.Red = newColor.R;

colorPicker.Green = newColor.G;

colorPicker.Blue = newColor.B;

colorPicker.previousColor = oldColor;

colorPicker.OnColorChanged(oldColor, newColor);

}

// Действия при изменнении интенсивности одного из каналов.

private static void OnColorRGBChangedCallback(DependencyObject sender, DependencyPropertyChangedEventArgs e)

{

ColorPickerUserControl colorPicker = (ColorPickerUserControl)sender;

Color color = colorPicker.Color;

// Определяем свойство, из-за которого этот метод сработал.

if (e.Property == RedProperty)

{

color.R = (byte)e.NewValue;

}

else if (e.Property == GreenProperty)

{

color.G = (byte)e.NewValue;

}

else if (e.Property == BlueProperty)

{

color.B = (byte)e.NewValue;

}

colorPicker.Color = color;

}

// Маршрутизируемое событие.

public static readonly RoutedEvent ColorChangedEvent =

EventManager.RegisterRoutedEvent("ColorChanged", RoutingStrategy.Bubble,

typeof(RoutedPropertyChangedEventHandler<Color>), typeof(ColorPickerUserControl));

public event RoutedPropertyChangedEventHandler<Color> ColorChanged

{

add { AddHandler(ColorChangedEvent, value); }

remove { RemoveHandler(ColorChangedEvent, value); }

}

private void OnColorChanged(Color oldValue, Color newValue)

{

RoutedPropertyChangedEventArgs<Color> args = new RoutedPropertyChangedEventArgs<Color>(oldValue, newValue);

args.RoutedEvent = ColorPickerUserControl.ColorChangedEvent;

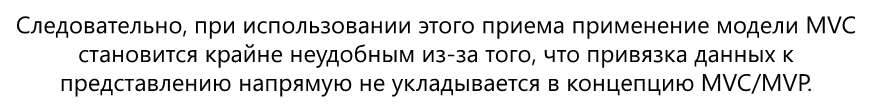
RaiseEvent(args);

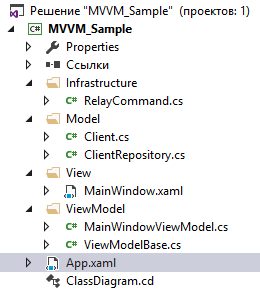
}

}

}

# MVVM





namespace MVVM\_Sample.Model

{

public class Client

{

public string FirstName { get; set; }

public string LastName { get; set; }

public Client()

{

}

public Client(string firstName, string lastName)

{

FirstName = firstName;

LastName = lastName;

}

public override string ToString()

{

return string.Format("{0} {1}", FirstName, LastName);

}

}

}

namespace MVVM\_Sample.Model

{

public static class ClientRepository

{

private static ObservableCollection<Client> \_clients;

public static ObservableCollection<Client> AllClients

{

get

{

if (\_clients == null)

\_clients = GenerateClientRepository();

return \_clients;

}

}

private static ObservableCollection<Client> GenerateClientRepository()

{

ObservableCollection<Client> clients = new ObservableCollection<Client>();

clients.Add(new Client("Jhon", "Doe"));

clients.Add(new Client("Tom", "Ronald"));

clients.Add(new Client("Jane", "Roe"));

return clients;

}

}

}

namespace MVVM\_Sample.ViewModel

{

public abstract class ViewModelBase : INotifyPropertyChanged, IDisposable

{

protected ViewModelBase()

{

}

public event PropertyChangedEventHandler PropertyChanged;

public virtual void OnPropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = this.PropertyChanged;

if (handler != null)

{

handler.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

public void Dispose()

{

this.OnDispose();

}

protected virtual void OnDispose()

{

}

}

}

namespace MVVM\_Sample.ViewModel

{

public class MainWindowViewModel : ViewModelBase

{

Client \_currentClient;

public Client CurrentClient

{

get

{

if (\_currentClient == null)

\_currentClient = new Client();

return \_currentClient;

}

set

{

\_currentClient = value;

OnPropertyChanged("CurrentClient");

}

}

ObservableCollection<Client> \_clients;

public ObservableCollection<Client> Clients

{

get

{

if (\_clients == null)

\_clients = ClientRepository.AllClients;

return \_clients;

}

}

RelayCommand \_addClientCommand;

public ICommand AddClient

{

get

{

if (\_addClientCommand == null)

\_addClientCommand = new RelayCommand(ExecuteAddClientCommand, CanExecuteAddClientCommand);

return \_addClientCommand;

}

}

public void ExecuteAddClientCommand(object parameter)

{

Clients.Add(CurrentClient);

CurrentClient = null;

}

public bool CanExecuteAddClientCommand(object parameter)

{

if (string.IsNullOrEmpty(CurrentClient.FirstName) ||

string.IsNullOrEmpty(CurrentClient.LastName))

return false;

return true;

}

protected override void OnDispose()

{

this.Clients.Clear();

}

}

}

<Window x:Class="MVVM\_Sample.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:MVVM\_Sample.ViewModel"

Title="MVVM. Главное окно" Height="293" Width="525">

<Window.DataContext>

<local:MainWindowViewModel></local:MainWindowViewModel>

</Window.DataContext>

<StackPanel>

<ListBox Name="ListBoxPersons" Height="100" ItemsSource="{Binding Path=Clients}">

</ListBox>

<Border BorderBrush="Black" CornerRadius="5" Margin="15,15,15,162" Padding="15" BorderThickness="1">

<Grid>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="77"></ColumnDefinition>

<ColumnDefinition Width="131"></ColumnDefinition>

<ColumnDefinition Width="211\*" />

</Grid.ColumnDefinitions>

<TextBlock Height="24" Grid.ColumnSpan="2" Margin="0,5">

First Name

</TextBlock>

<TextBox Grid.Column="1" Name="TextBoxFirstName" Height="24"

Text="{Binding Path=CurrentClient.FirstName, UpdateSourceTrigger=PropertyChanged}">

</TextBox>

<TextBlock Grid.Row="1" Height="24" Grid.ColumnSpan="2" Margin="0,5">

Last Name

</TextBlock>

<TextBox Grid.Row="1" Grid.Column="1" Name="TextBoxLastName" Height="24"

Text="{Binding Path=CurrentClient.LastName, UpdateSourceTrigger=PropertyChanged}">

</TextBox>

<Button Grid.Row="2" Grid.ColumnSpan="2" Height="24"

Command="{Binding AddClient}">

Add

</Button>

</Grid>

</Border>

</StackPanel>

</Window>

namespace MVVM\_Sample

{

public partial class App : Application

{

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

//MainWindow window = new MainWindow();

//MainWindowViewModel viewModel = new MainWindowViewModel();

//window.DataContext = viewModel;

//window.Show();

}

}

}

namespace MVVM\_Sample.Infrastructure

{

class RelayCommand : ICommand

{

readonly Action<object> \_execute;

readonly Predicate<object> \_canExecute;

public RelayCommand(Action<object> execute)

: this(execute, null)

{

}

public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)

{

if (execute == null)

{

throw new ArgumentNullException("execute");

}

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null ? true : \_canExecute.Invoke(parameter);

}

public event EventHandler CanExecuteChanged

{

add

{

CommandManager.RequerySuggested += value;

}

remove

{

CommandManager.RequerySuggested -= value;

}

}

public void Execute(object parameter)

{

\_execute.Invoke(parameter);

}

}

}