

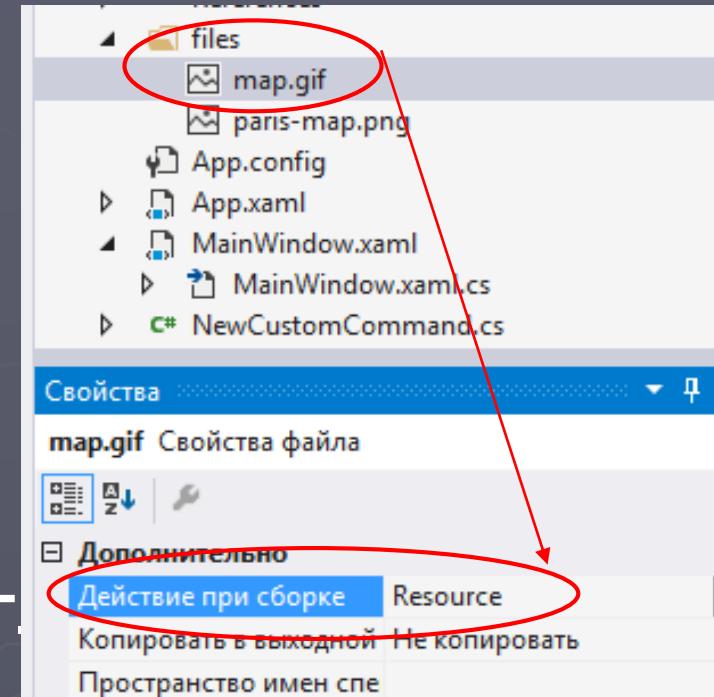
WPF Windows Presentation Foundation

3 часть

Ресурсы

► Ресурс сборки – блок двоичных данных, встроенный в сборку

► Ресурс объекта – .NET объект который объявляется в одном месте и используется в других (логический ресурс – кнопки, кисти и т.д.)



- 1) эффективность: определить один раз и многократно использовать его в различных местах приложения
- 2) поддержка : изменение ресурса в одном месте

```
<Window.Resources>
```

```
    <ImageBrush x:Key="CommonImBrush"  
        ViewportUnits="RelativeToBoundingBox"  
        ImageSource="files/map.gif"  
        Opacity="0.3"></ImageBrush>
```

```
</Window.Resources>
```

определяет ключ в словаре

Свойство **Resources** представляет объект **ResourceDictionary** или словарь ресурсов



```
<Button Background="{StaticResource CommonImBrush}"  
        FontWeight="Bold"  
        FontSize="14"  
        Height="60"  
        >Кнопка с фоном</Button>
```

применить ресурс используя кли

► Добавление и установка ресурса

```
ImageBrush CommonimBrush = new ImageBrush();
//...

// добавление ресурса в словарь ресурсов окна
this.Resources.Add("CommonimBrush", CommonimBrush);

// установка ресурса у кнопки
button1.Background = (Brush)this.TryFindResource("CommonimBrush");
```

► ResourceDictionary:

- Метод **Add(string key, object resource)** добавляет объект с ключом key в словарь
- Метод **Remove(string key)** удаляет из словаря ресурс с ключом key
- Свойство **Uri** устанавливает источник словаря
- Свойство **Keys** возвращает все имеющиеся в словаре ключи
- Свойство **Values** возвращает все имеющиеся в словаре объекты

- ▶ **Статический** ресурс - свойство инициализируется один раз и не меняет свое значение, даже если ресурс был изменен
- ▶ **Динамический** ресурса - свойство элемента обновляется при обновлении ресурса
- ▶ Один и тот же ресурс может быть и стат. и динамич.

Пример. Определение ресурса

```
<Window.Resources>
    <ImageBrush x:Key="MunBrush"
        TileMode="Tile"
        ViewportUnits="Absolute"
        Viewport="0 0 64 64"
        ImageSource="files/munich.jpg"
        Opacity="0.5"></ImageBrush>
</Window.Resources>
```

```
<Grid>
```

```
  <StackPanel Margin="5">
```

```
    <Button Background="{DynamicResource MunBrush}"
```

```
      Padding="5"
```

```
      FontWeight="Bold"
```

```
      FontSize="14"
```

```
      Click="Change_OnClick"
```

```
      Margin="5">Динамический ресурс</Button>
```

```
    <Button Name ="Change" Padding="5"
```

```
      Margin="5"
```

```
      Click="Change_OnClick"
```

```
      FontWeight="Bold"
```

```
      FontSize="14">Изменить фон (ресурс)</Button>
```

```
    <Button Background="{StaticResource MunBrush}"
```

```
      Padding="5"
```

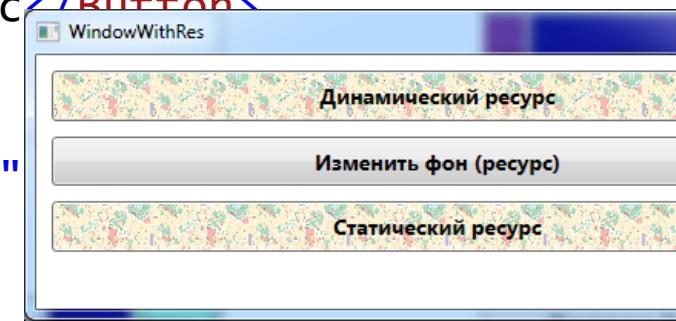
```
      Margin="5"
```

```
      FontWeight="Bold"
```

```
      Click="Change_OnClick"
```

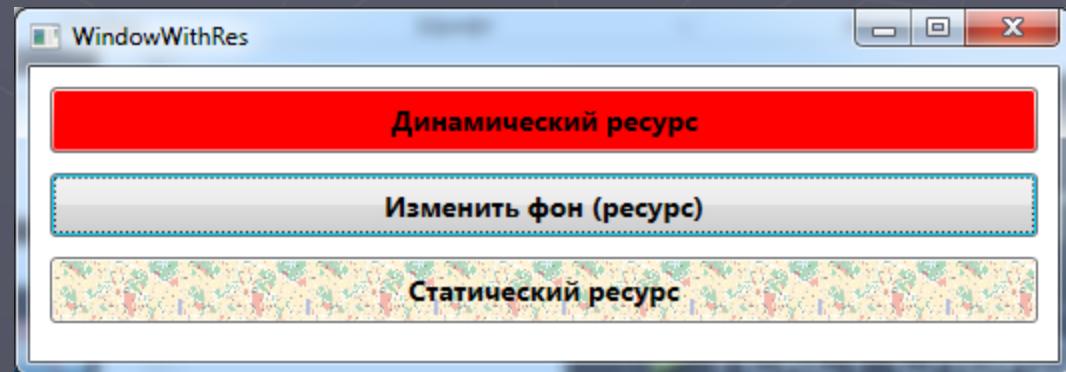
```
      FontSize="14">Статический ресурс</Button>
```

```
</StackPanel>
```



```
private void Change_OnClick(object sender, RoutedEventArgs e)
{
    this.Resources["MunBrush"] =
        new SolidColorBrush(Colors.Red);

}
```



► Установка динамического ресурса в коде

```
ImageBrush CommonimBrush = new ImageBrush();  
//...
```

// добавление ресурса в словарь ресурсов окна

```
this.Resources.Add("CommonimBrush", CommonimBrush);
```

// установка ресурса у кнопки

```
button1.SetResourceReference(Button.BackgroundProperty, "CommonimBrush")  
}
```



Управление ресурсами

The screenshot shows a portion of a C# code editor in Visual Studio. The code is part of a class named `WindowWithResources` within the `WpfAppDemo` namespace. The code includes imports for `System.Windows.Media`, `System.Windows.Media.Imaging`, and `System.Windows.Shapes`. A tooltip is open over the `this.Resources` line, listing methods related to the `ResourceDictionary` class:

- Add
- BeginInit
- Clear
- Contains
- CopyTo
- EndInit
- FindName
- GetEnumerator
- RegisterName
- Remove
- UnregisterName

The tooltip also shows the `Brush(Colors.Red);` completion suggestion. A status bar at the bottom right indicates that `Brush(Colors.Red);` implements the `IResourceDictionary` interface.

```
using System;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace WpfAppDemo
{
    /// <summary>
    /// Логика взаимодействия для WindowWithResources.xaml
    /// </summary>
    public partial class WindowWithResources : Window
    {
        public WindowWithResources()
        {
            InitializeComponents();
        }

        private void ChangeColor(object sender, edEventArgs e)
        {
            this.Resources[<span style="background-color: red; border: 1px solid black; padding: 2px; border-radius: 5px; display: inline-block;>2</span>] = Brush(Colors.Red);
        }
    }
}
```

Ресурсы приложения

App.xaml

```
<Application.Resources>

    <ImageBrush x:Key="MunBrush" TileMode="Tile"
        ViewportUnits="Absolute" Viewport="0 0 32 32"
        ImageSource="files/munich.jpg"
        Opacity="0.3"></ImageBrush>

</Application.Resources>
```

Системные ресурсы

- ▶ SystemColors
- ▶ SystemFonts
- ▶ SystemParameters

```
<Label  
Foreground="{x:Static SystemColors.WindowTextBrush}">  
Статически</Label>
```

```
<Label  
Foreground="{DynamicResource  
{x:Static SystemColors.WindowTextBrushKey}}}">  
Динамически</Label>
```

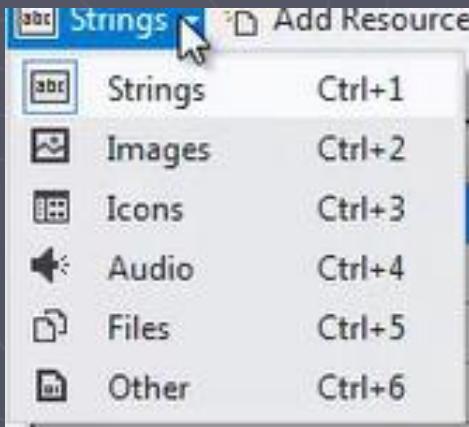
Локализация приложений

► Содержит

The screenshot shows the 'LocalizeResources.resx' file open in the Visual Studio resource editor. At the top, there's a toolbar with 'Строки' (Strings), 'Добавить ресурс' (Add Resource), 'Удалить ресурс' (Delete Resource), and a dropdown for 'Модификатор доступа' (Access Modifier) set to 'Public'. Below the toolbar is a tree view showing two resources: 'LocalizeResources.resx' and 'LocalizeResources.ru.resx'. The main area is a table with columns 'Имя' (Name) and 'Значение' (Value). The table contains three entries:

Имя	Значение
BtnLabel	Enter
FNLabel	First Name
LNLabel	Last Name

► Может



en_GB
ru_RU

Язык культуры

► Обращение к ресурсу в разметке

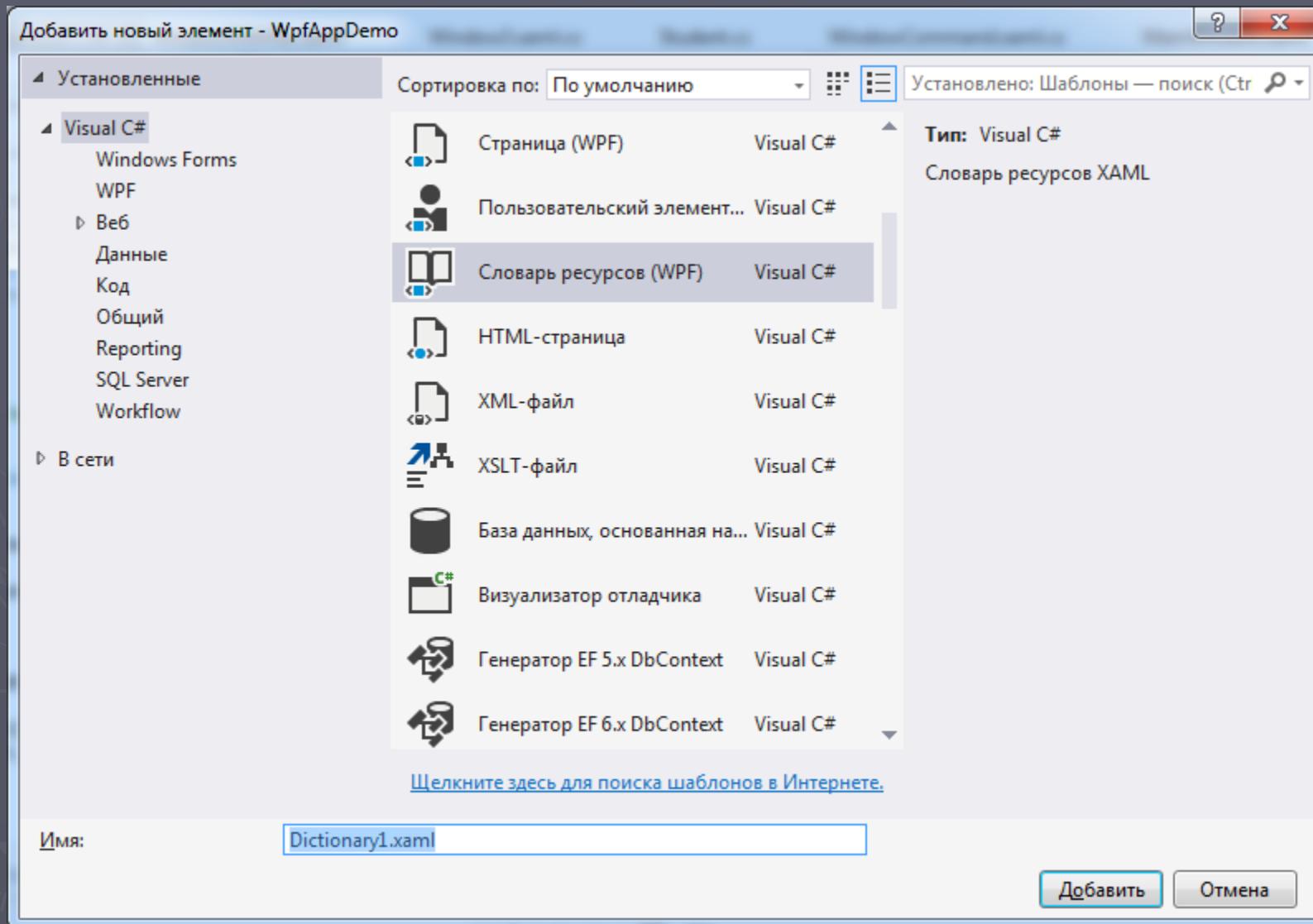
```
<TextBlock Text="{x:Static local:LocalizeResources.FNLabel}">  
</TextBlock>
```

► Переключение культуры (до инициализации компонент и требует перезапуска)

```
Thread.CurrentThread.CurrentCulture =  
    new CultureInfo(Settings.Default.Culture);
```

```
Thread.CurrentThread.CurrentCulture =  
    new CultureInfo("ru_RU");
```

Словари ресурсов



коллекция объектов `ResourceDictionary`, которые добавляются к ресурсам

```
<ResourceDictionary>
  <ResourceDictionary.MergedDictionaries>
    <ResourceDictionary Source="DictionaryMono.xaml" />
    <ResourceDictionary Source="Dictionary2.xaml" />
    <ResourceDictionary Source="ButtonStyles.xaml" />
    <SolidColorBrush Color="Green" x:Key="GButton" />
  </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
```

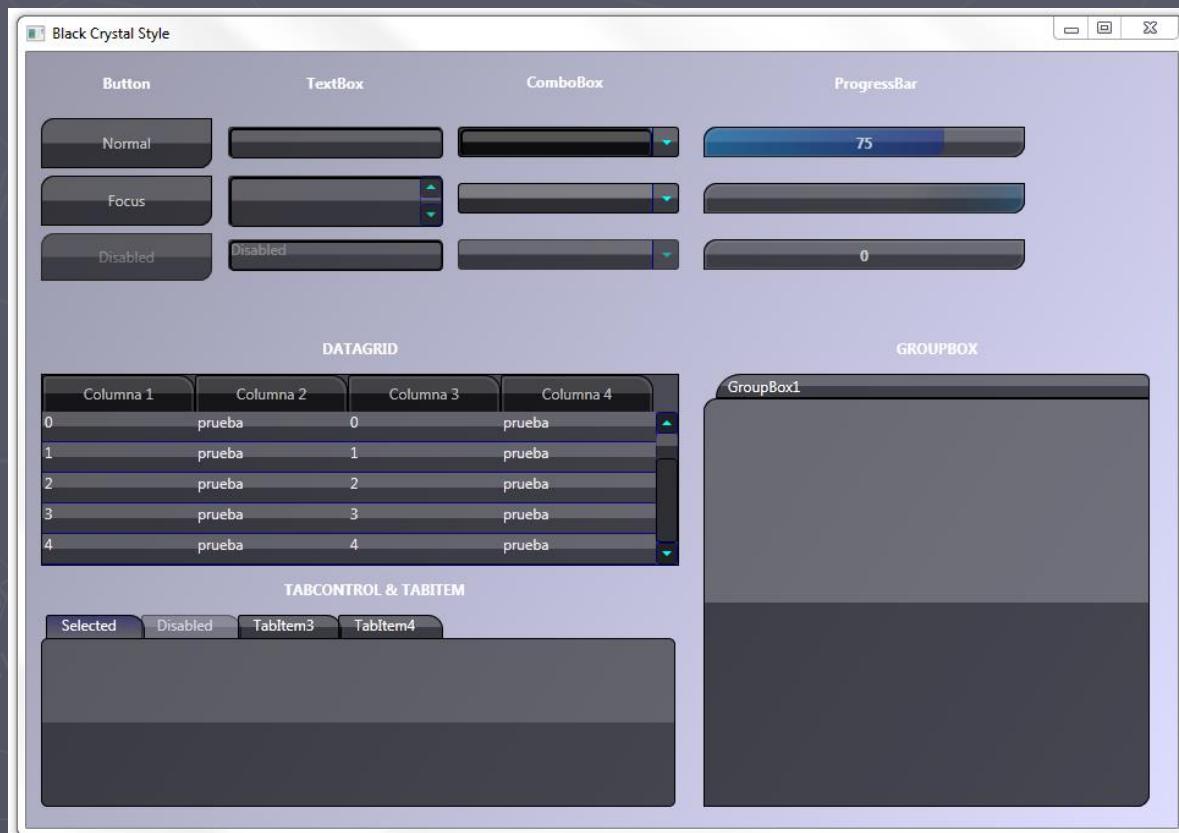
файла ресурсов подсоединяется к
ресурсам окна

```
<Window.Resources>
  <ResourceDictionary Source="Dictionary1.xaml" />
</Window.Resources>
```

Стили

Стиль – коллекция значений свойств, которые могут быть применены к элементу (CSS)

Хранятся в ресурсах



Пример. Работа с ресурсами

1) Объявление ресурса

```
<Window.Resources>
    <FontWeight x:Key="PNVWeigth">
        Bold
    </FontWeight>
    <system:Double x:Key="PNVSize">
        20
    </system:Double>
</Window.Resources>
```

Нет связи
между
ресурсами

2) Применение ресурса

```
<Button Command="local>NewCustomCommand.PnvCommand"
        Margin="100"
        FontWeight="{StaticResource PNVWeigth}"
        FontSize="{StaticResource PNVSize}"
        >
    Команда
</Button>
```

Объемный код

Определение стиля окна

```
<Window.Resources>
    <Style x:Key="PNVStyle">
        <Setter Property="Control.FontSize" Value="20"/>
        <Setter Property="Control.FontFamily" Value="Times New Roman"/>
        <Setter Property="Control.FontWeight" Value="UltraLight"/>
    </Style>
</Window.Resources>
```

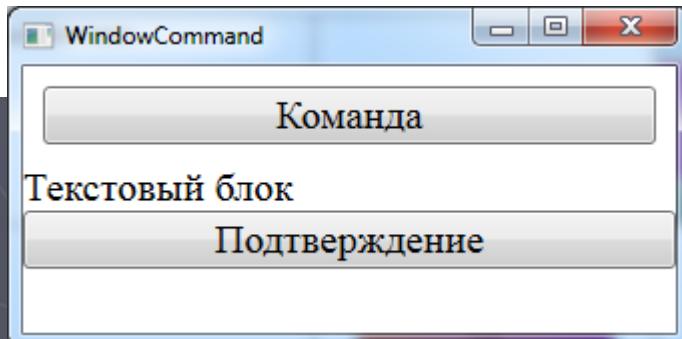
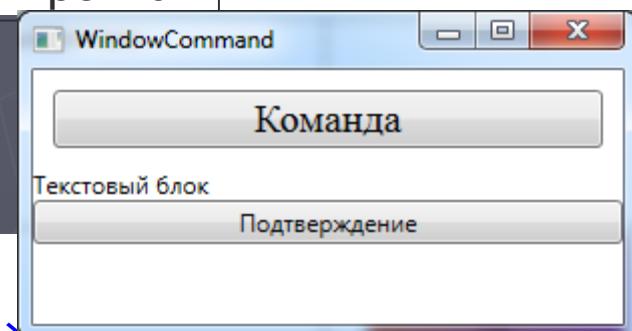
Тип_элемента.Свойство_элемента

Группировка

представляет
класс **System.Windows.Style**

Применение стиля

```
<Button Margin="100"
        Style="{DynamicResource PNVStyle}">
    Команда
</Button>
```



- Значения параметров
элементов приоритетнее
стиля

Ключевые свойства стиля

- ▶ **Setters** – коллекция объектов, которые автоматически устанавливают значение свойств элементов управления
- ▶ **Triggers** – коллекция объектов, которые позволяют автоматически изменять параметры стиля
- ▶ **BasedOn** – для создания стиля, который наследует другой стиль и переопределяет его значения
- ▶ **TargetType** – указывает тип элементов на которые действует стиль

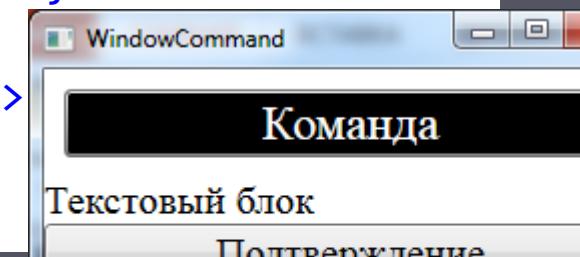
Пример. Наследование стилей

```
<Style x:Key="PNVStyle">
    <Setter Property="Control.FontSize" Value="20"/></Setter>
    <Setter Property="Control.FontFamily"
           Value="Times New Roman"/></Setter>
    <Setter Property="Control.FontWeight"
           Value="UltraLight"/></Setter>
</Style>
<Style x:Key="PNVStyleNext" BasedOn="{StaticResource PNVStyle}">
    <Setter Property="Control.Background" Value="Black"/>
    <Setter Property="Control.FontSize" Value="24"/>
    <Setter Property="Control.Foreground" Value="White"/>
</Style>
```

Наследование

```
<StackPanel>
    <Button Margin="10" Style="{DynamicResource PNVStyleNext}">
        Команда </Button>
    <TextBlock Style="{DynamicResource PNVStyle}">
        Текстовый блок </TextBlock>
    <Button Style="{DynamicResource PNVStyle}">
        Подтверждение </Button>
</StackPanel>
```

переопределение

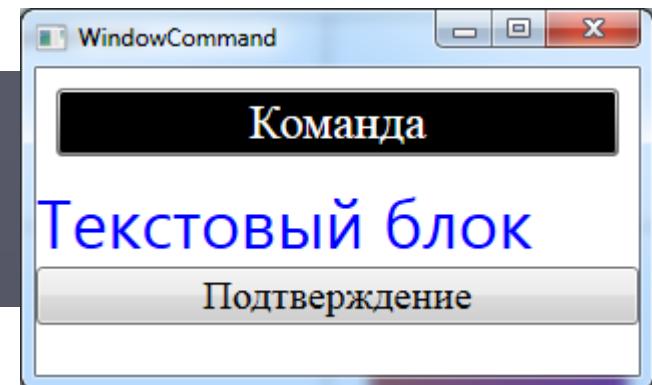


► Автоматическое применение свойств к ЭУ

```
<Style TargetType="TextBlock"
```

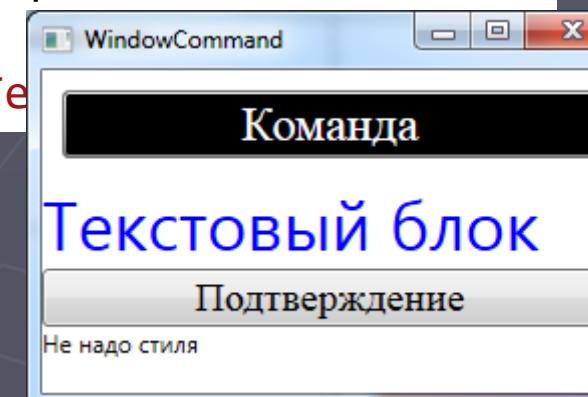
применяется

```
<TextBlock> Текстовый блок </TextBlock>
```



```
<Button Style="{DynamicResource PNVStyle}"> Подтверждение </Button>
```

```
<TextBlock Style="{x:Null}"> Не надо стиля </TextBlock>
```



Задание событий и обработчиков (редко)

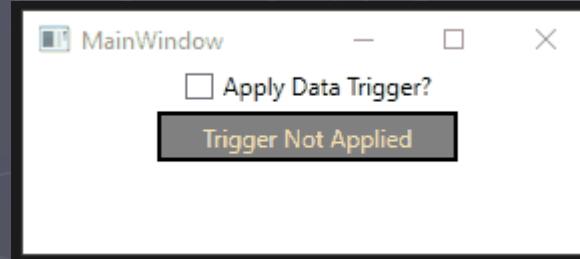
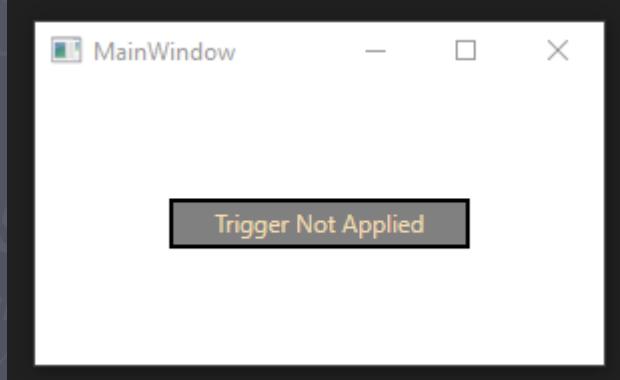
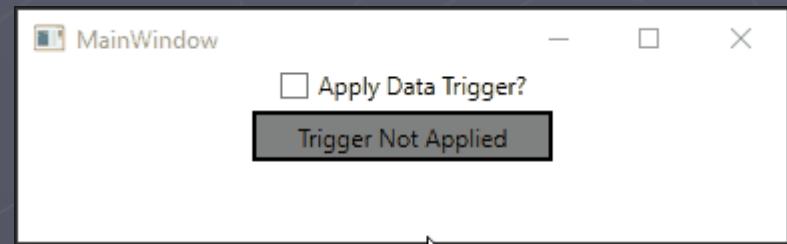
```
<Style TargetType="TextBlock">
    <Setter Property="Control.FontSize" Value="34"></Setter>
    <Setter Property="Control.BorderBrush" Value="Aqua"/>
    <Setter Property="Control.Foreground" Value="Blue"/>

    <EventSetter
        _____
    Event="MouseDown"
    Handler="control_MouseDown"/>
</Style>
```

Триггеры

Триггеры – декларативное определение некоторых действий, которые выполняются при изменении свойств (свойств зависимостей) стиля

Определяется в XAML



Основные типы триггеров

System.Windows.TriggerBase

- ▶ **Trigger** – простой триггер. Следит за изменением значения свойства
- ▶ **MultiTrigger** – срабатывает при выполнении множества условий
- ▶ **DataTrigger** – срабатывает при изменении в связанных с ним данных
- ▶ **MultiDataTrigger** – множество триггеров данных
- ▶ **EventTrigger** – применяется при возникновении события

Простой триггер (триггер свойств)

```
<Style TargetType="TextBlock">
    <Setter Property="Control.FontSize" Value="34"></Setter>
    <Setter Property="Control.BorderBrush" Value="Aqua"/>
    <Setter Property="Control.Foreground" Value="Blue"/>

    <!--<EventSetter Event="MouseDown"
Handler="control_MouseDown"/><!--&gt;
    &lt;Style.Triggers&gt;
        &lt;Trigger Property="IsMouseOver" Value="True"&gt;
            &lt;Setter Property="FontSize" Value = "10"/&gt;
        &lt;/Trigger&gt;
        &lt;Trigger Property="IsFocused" Value="True"&gt;
            &lt;Setter Property="FontSize" Value="40"/&gt;
        &lt;/Trigger&gt;
    &lt;/Style.Triggers&gt;
&lt;/Style&gt;</pre>

по наведению на  
textblock высота  
шрифта  
устанавливается в 10

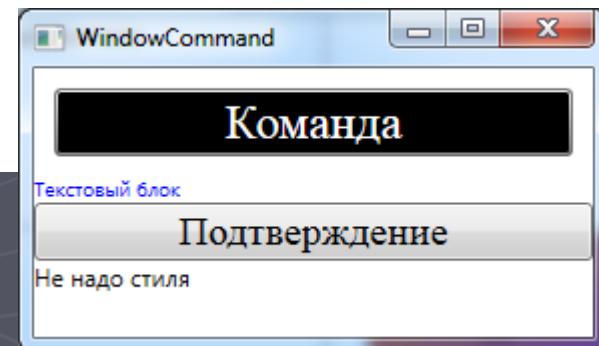


Если/отлеживаем  
свойство



Установить


```



MultiTrigger - содержит коллекцию элементов Condition

```
Style x:Key="PNVStyleNext" BasedOn="{StaticResource PNVStyle}">
    <Setter Property="Control.Background" Value="Black"/>
    <Setter Property="Control.FontSize" Value="24"/>
    <Setter Property="Control.Foreground" Value="White"/>

    <Style.Triggers>
        <MultiTrigger>

            <!--Список условий-->
            <MultiTrigger.Conditions>
                <Condition Property="Control.IsMouseOver" Value="True"></Condition>
                <Condition Property="Control.IsPressed" Value="True"></Condition>
            </MultiTrigger.Conditions>

            <!--Список изменений, которые вступят в силу, если все условия выполняются-->
            <MultiTrigger.Setters>
                <Setter Property="Control.Foreground" Value="DarkBlue"></Setter>
                <Setter Property="Control.FontSize" Value="20"></Setter>
            </MultiTrigger.Setters>

        </MultiTrigger>
    </Style.Triggers>
</Style>
```

DataTrigger

```
public class Student
{
    public String FName { get; set; }
    public String LName { get; set; }
    public int Number { get; set; }
    public override string ToString()
```

```
<Style TargetType="ListBoxItem">
    <Style.Triggers>
```

Задает значение отслеживаемого свойства, при котором сработает триггер
--Если значение свойства объекта будет равно 0 поменять свойства--

```
<!--Если значение свойства объекта будет равно 0 поменять свойства-->
<DataTrigger Binding="{Binding Path=Number}" Value="0">
    <Setter Property="Foreground" Value="Red" />
</DataTrigger>
```

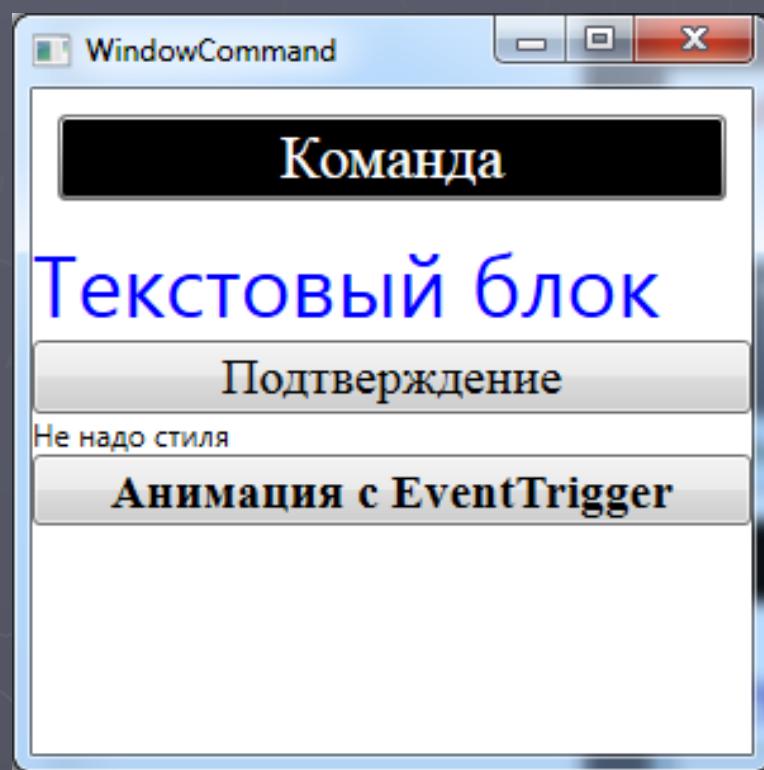
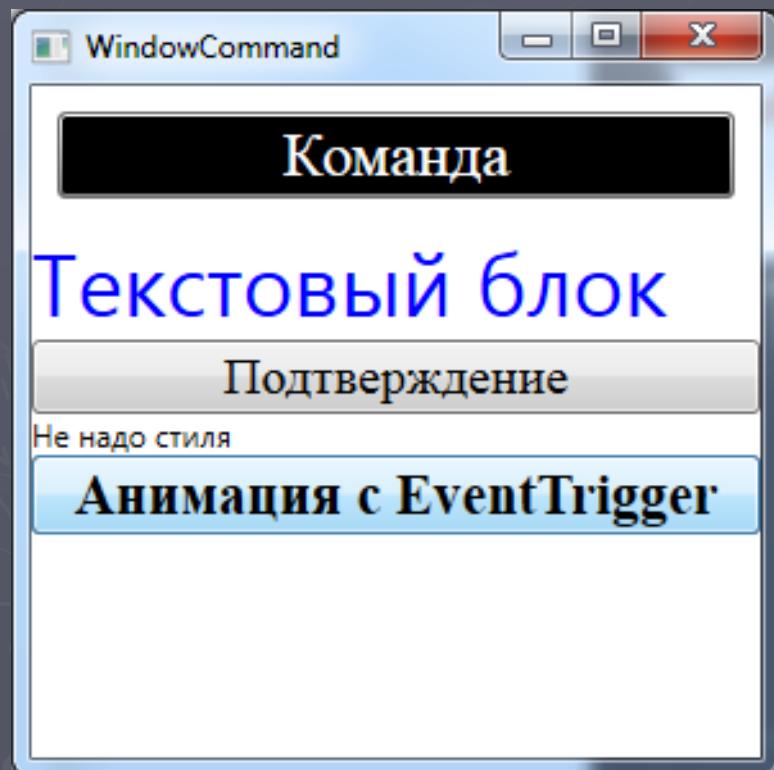
```
<MultiDataTrigger>
    <MultiDataTrigger.Conditions>
        <Condition Binding="{Binding Path=Number}"
                    Value="10" />
        <Condition Binding="{Binding Path=FName}"
                    Value="July" />
    </MultiDataTrigger.Conditions>
    <Setter Property="Background" Value="Green" />
</MultiDataTrigger>
</Style.Triggers>
</Style>
```

Для соединения с отслеживаемыми свойствами триггеры данных используют выражения привязки

EventTrigger

```
<Style x:Key="EventAnimation">
    <!--Стили-->
    <Style.Setters>
        <Setter Property="Control.FontFamily" Value="Times New Roman" />
        <Setter Property="Control.FontSize" Value="18" />
        <Setter Property="Control.FontWeight" Value="Bold" />
    </Style.Setters>
    <!--Триггеры
EventTrigger - ожидает определенного события-->
    <Style.Triggers>
        <!--Действие на событие MouseEnter-->
        <EventTrigger RoutedEvent="Mouse.MouseEnter">
            <EventTrigger.Actions>
                <BeginStoryboard>
                    <Storyboard>
                        <DoubleAnimation
                            Duration="0:0:0.3"
                            Storyboard.TargetProperty="FontSize"
                            To="22" />
                    </Storyboard>
                </BeginStoryboard>
            </EventTrigger.Actions>
        </EventTrigger>
        <!--Действие на событие MouseLeave-->
        <EventTrigger RoutedEvent="Mouse.MouseLeave">
            <EventTrigger.Actions>
                <BeginStoryboard>
                    <Storyboard>
                        <DoubleAnimation
                            Duration="0:0:3"
                            Storyboard.TargetProperty="FontSize" To="18" />
                    </Storyboard>
                </BeginStoryboard>
            </EventTrigger.Actions>
        </EventTrigger>
    </Style.Triggers>
```

```
<Button Style="{StaticResource EventAnimation}">  
    Анимация с EventTrigger  
</Button>
```



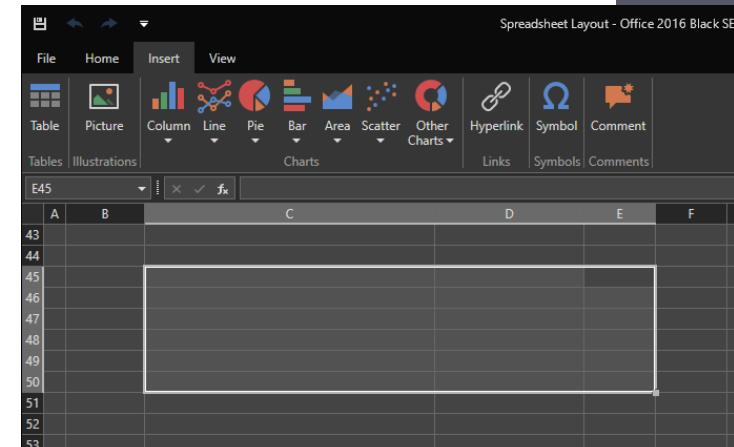
Темы

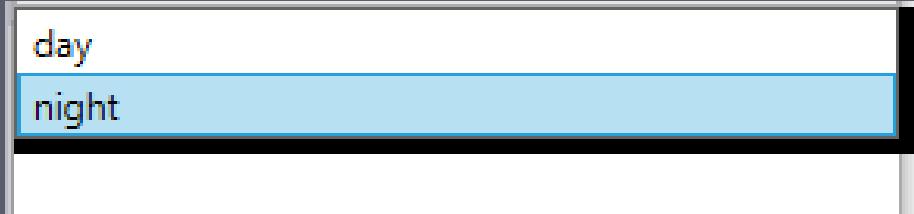
► объединение стилей

1) Создается файл словаря ресурсов - *nigth.xaml*, и определяется некоторый набор ресурсов:

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:WpfAppDemo">

    <Style x:Key="TextBlockNigth" TargetType = "TextBlock">
        <Setter Property="Background" Value="Black" />
        <Setter Property="Foreground" Value="Gray" />
    </Style>
    <Style x:Key="WindowNigth" TargetType="Window">
        <Setter Property="Background" Value="Black" />
    </Style>
    <Style x:Key="ButtonNigth" TargetType="Button">
        <Setter Property="Background" Value="Black" />
        <Setter Property="Foreground" Value="Gray" />
        <Setter Property="BorderBrush" Value="Gray" />
    </Style>
</ResourceDictionary>
```





► 2) ВЫВОДИМ ЭУ ДЛЯ СМЕНЫ ТЕМ И МЕНЯЕМ

```
public Window6()
{
    InitializeComponent();
    List<string> styles = new List<string> { "day", "night" };
    Theme.SelectionChanged += ThemeChange;
    Theme.ItemsSource = styles;
    Theme.SelectedItem = "nigth";

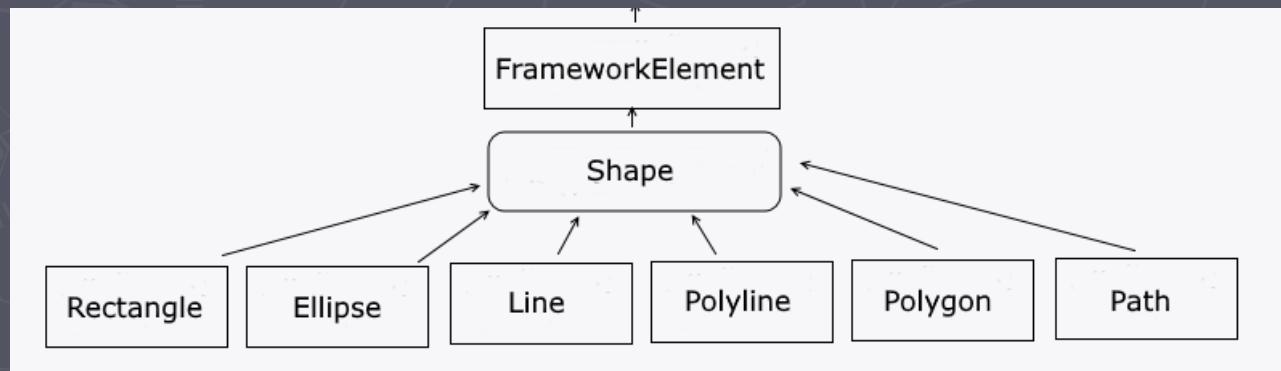
}
private void ThemeChange(object sender, SelectionChangedEventArgs e)
{
    string style = Theme.SelectedItem as string;
    // определяем путь к файлу ресурсов
    var uri = new Uri(style + ".xaml", UriKind.Relative);
    // загружаем словарь ресурсов
    ResourceDictionary resourceDict = Application.LoadComponent(uri) as
ResourceDictionary;
    // очищаем коллекцию ресурсов приложения
    Application.Current.Resources.Clear();
    // добавляем загруженный словарь ресурсов
    Application.Current.Resources.MergedDictionaries.Add(resourceDict);
}
```

Shapes

- **Фигуры(Shapes)**—объекты для отображения примитивов, таких как линии, прямоугольники, эллипсы и т.д.

FrameworkElement:

- прорисовка
- размещаются в контейнерах компоновки
- поддерживают события



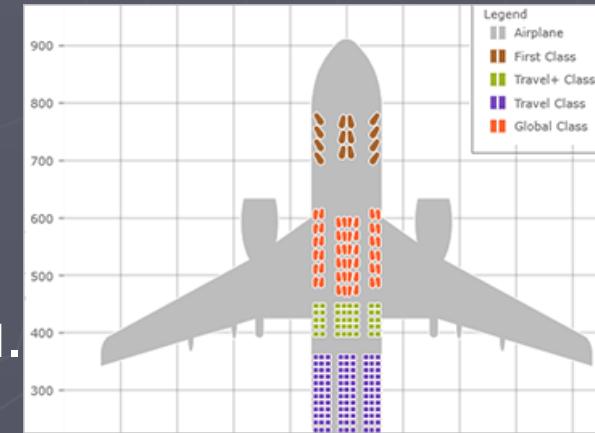
- ▶ Line
- ▶ Ellipse
- ▶ Rectangle
- ▶ Polyline
- ▶ Polygon
- ▶ Path

PathGeometry
PathFigure

Visual – функциональность визуализации
элементов в WPF (создание новых элементов
управления)

LinearGradientBrush
RadialGradientBrush
ImageBrush
DrawingBrush
VisualBrush

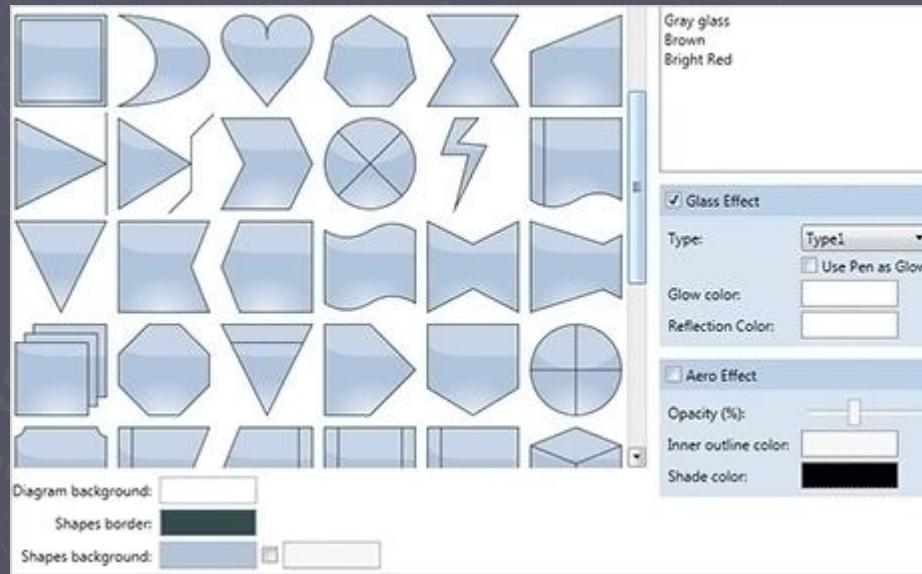
Path - любую
фигуру, группы
фигур и более
сложные элементы.



```
<Line X1="100" Y1="50" X2="20" Y2="30" Stroke="Blue" />
```

```
<Ellipse Fill="Blue" Width="20" Height="200" />
```

```
<Rectangle Width="100" Height="30" Stroke="Blue" Fill="LightBlue">
    <Rectangle.RenderTransform>
        <RotateTransform Angle="45" />
    </Rectangle.RenderTransform>
</Rectangle>
```



2-D, 3-D и изображения

► Графический API WPF:

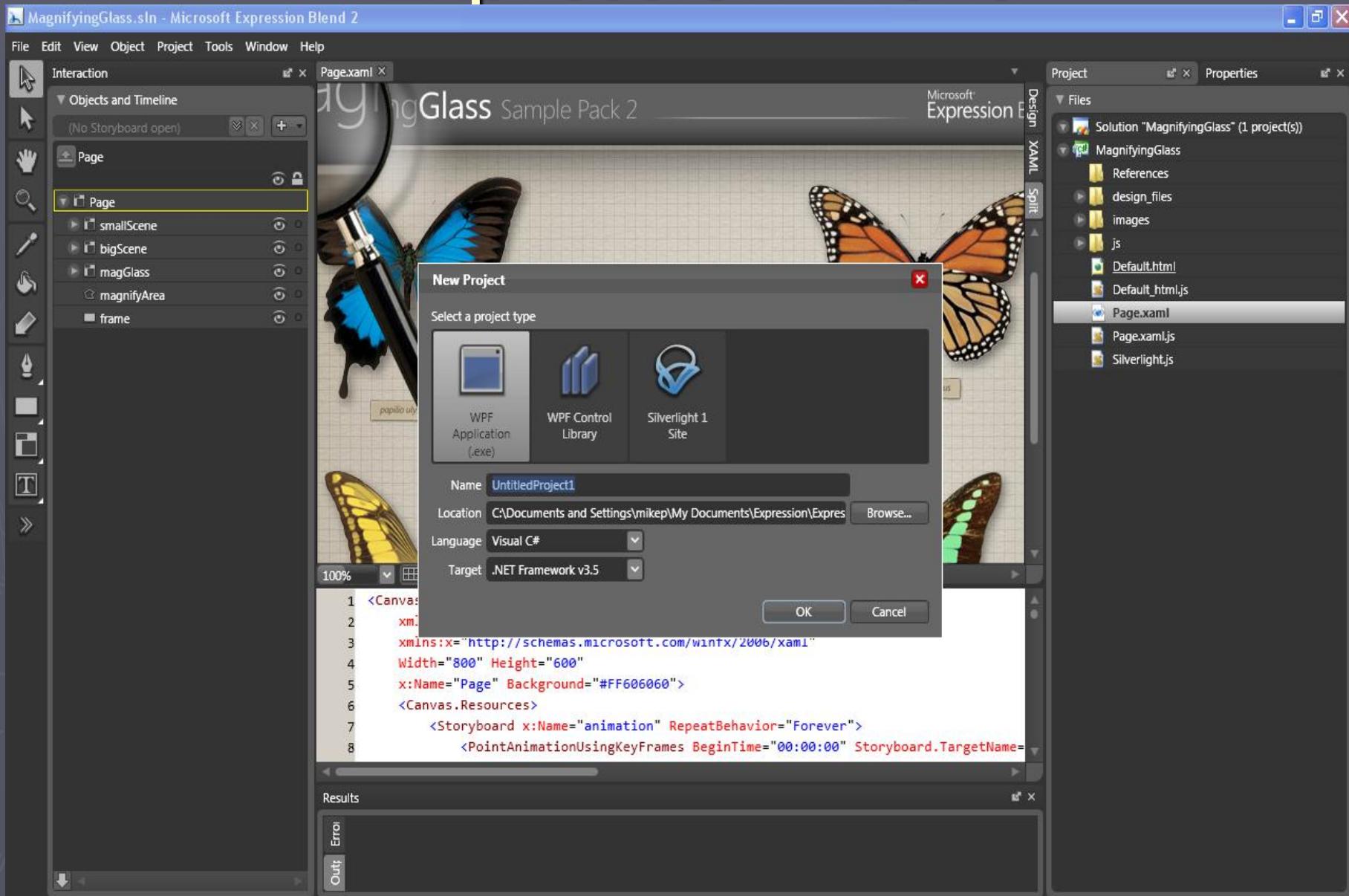
- Brushes (Кисти)
- Shapes (Примитивы)
- Imaging (Изображения)
- Geometries (Геометрии)
- Transformations (Трансформации)
- Animations (Анимации)
- Visuals (Визуальные элементы)
- 3-D графика

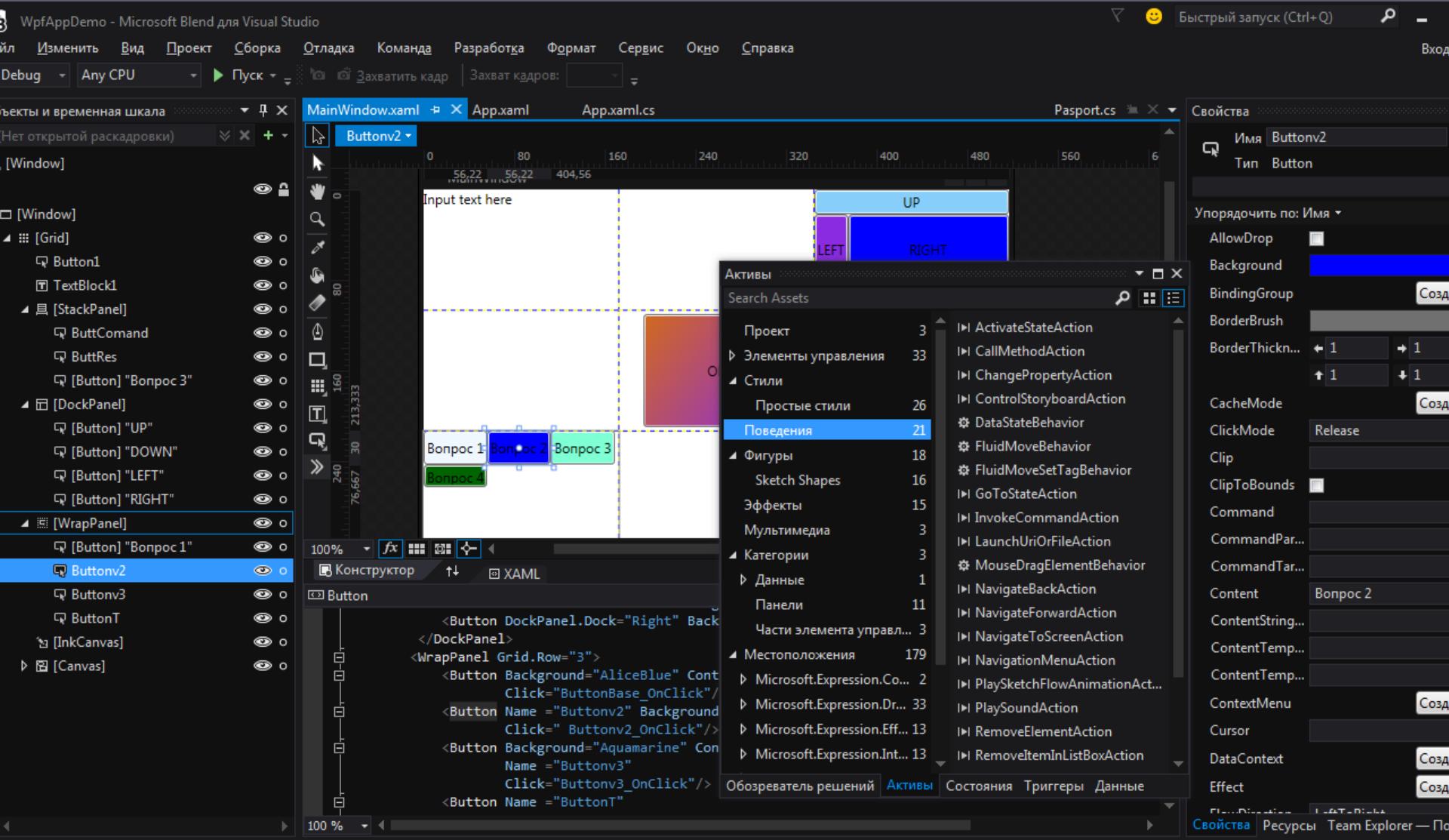


<Viewport3D>

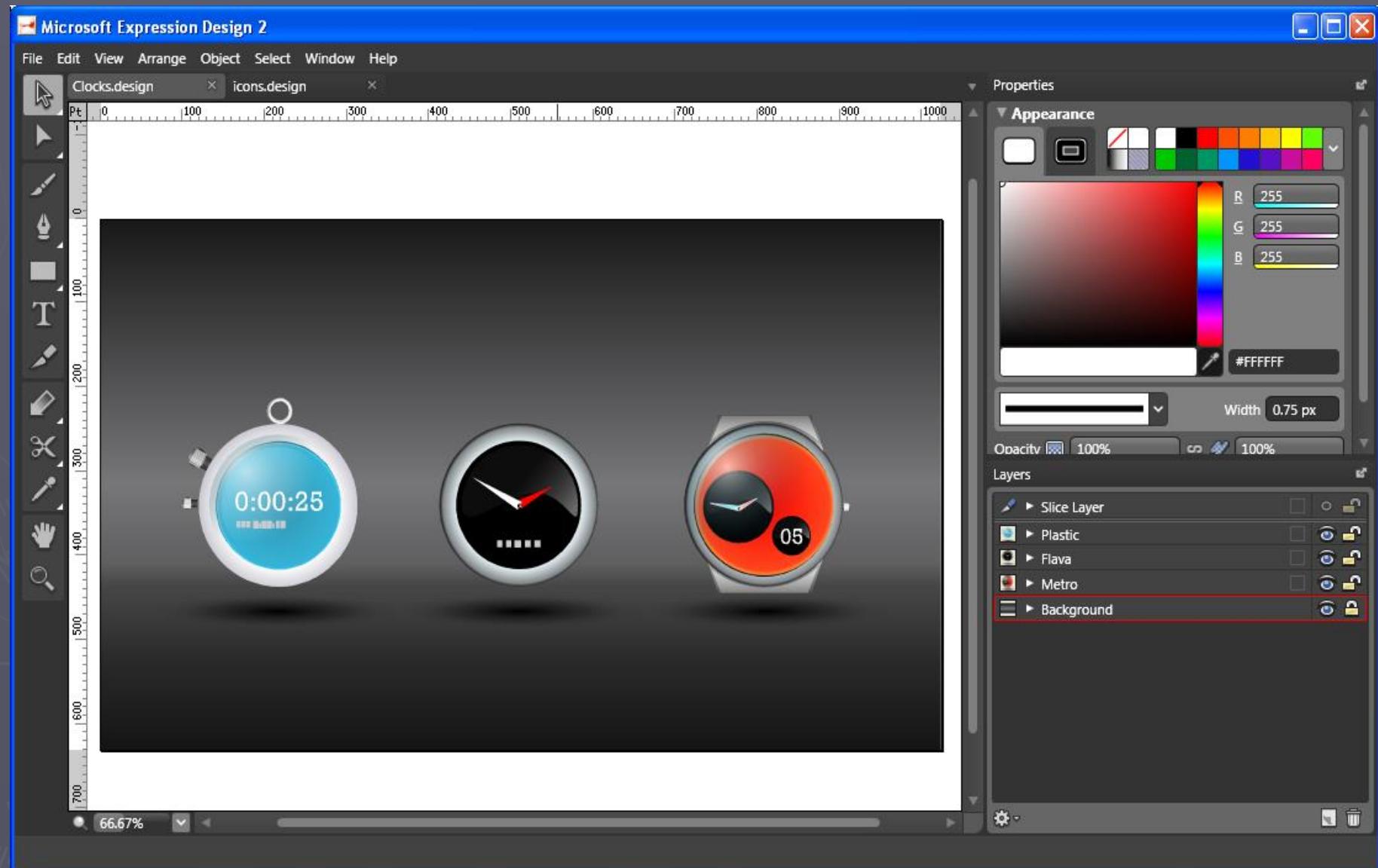
```
<Viewport3D.Camera>
  <Viewport3D.Children>
    <ModelVisual3D>
      <ModelVisual3D.Content>
        <GeometryModel3D>
```

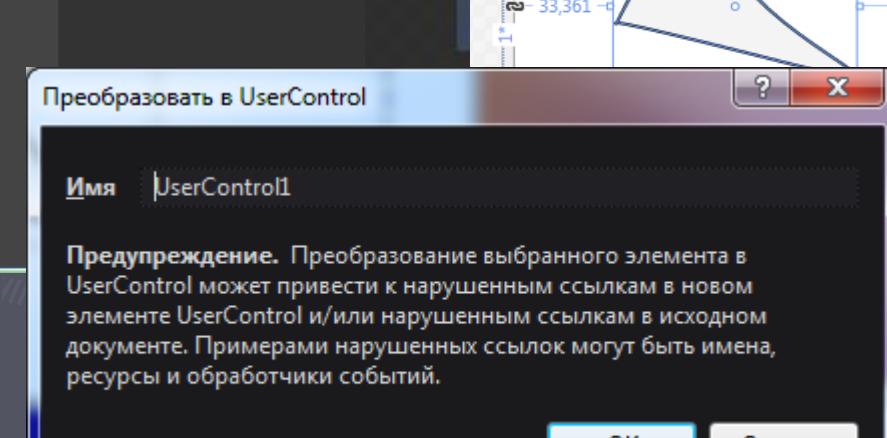
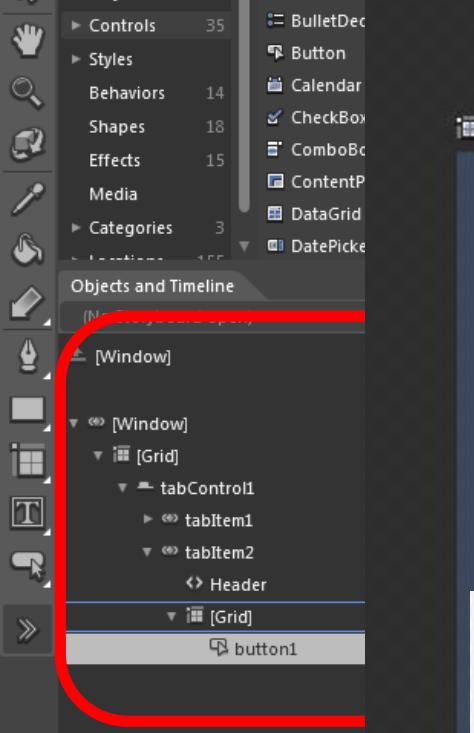
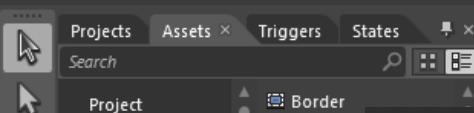
Expression Blend





Expression Design





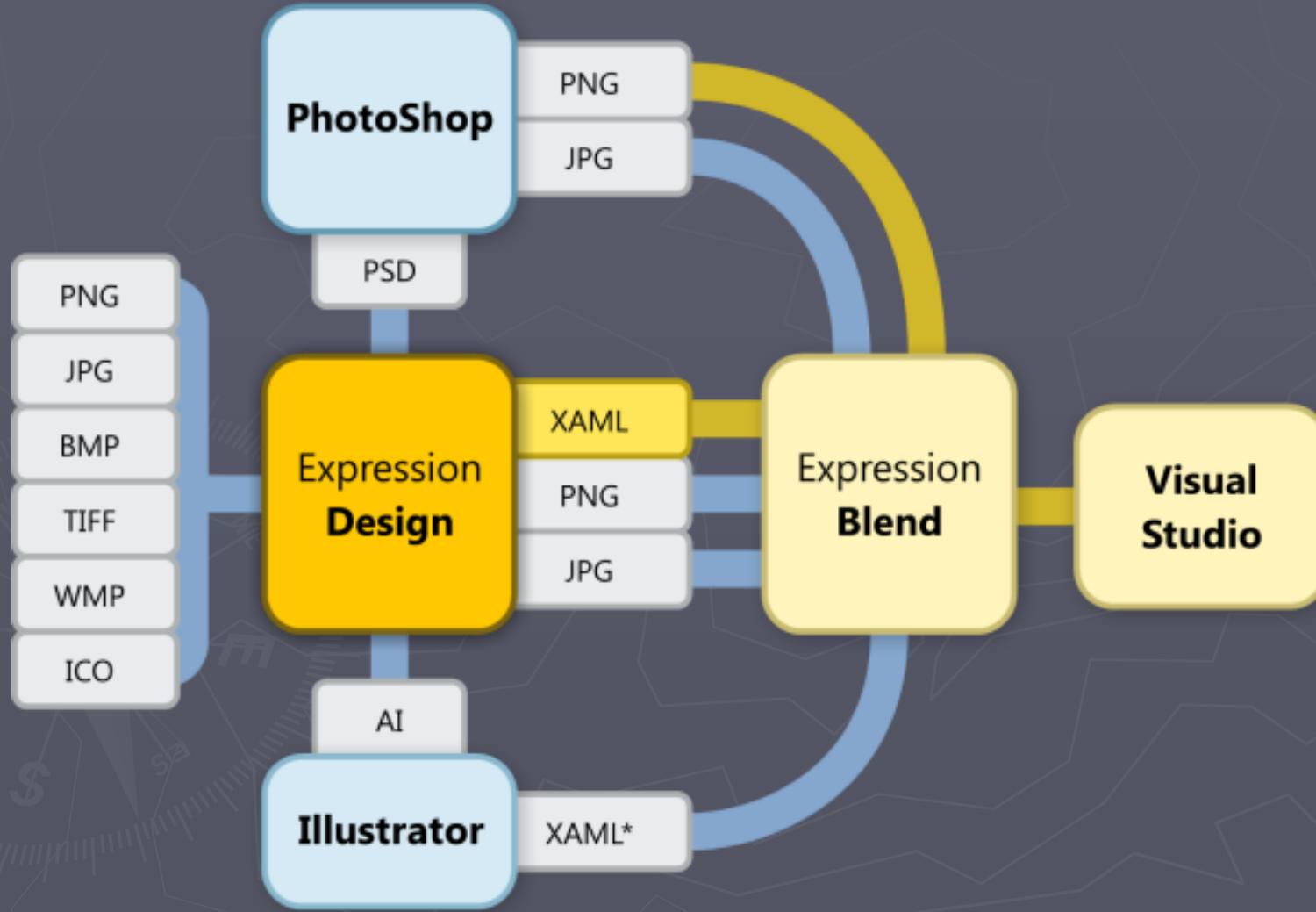
```

</> Path
Content="Вопрос 3" Canvas.Bottom="30" Canvas.Left="20"/>
Content="Вопрос 4" Canvas.Bottom="20" Canvas.Right="40"/>

HorizontalAlignment="Left" Height="1" Margin="34,0,0,46.333" Grid.Row="1" Stretch="Fill" Stroke="Black" VerticalAlignment="Top"/>
Content="Вопрос 5" Canvas.Bottom="30" Canvas.Left="20"/>
Content="Вопрос 6" Canvas.Bottom="20" Canvas.Right="40"/>

```

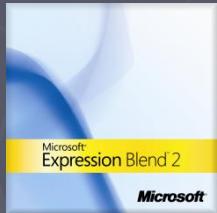
Работа графического дизайнера



*Upcoming 3rd party plugins



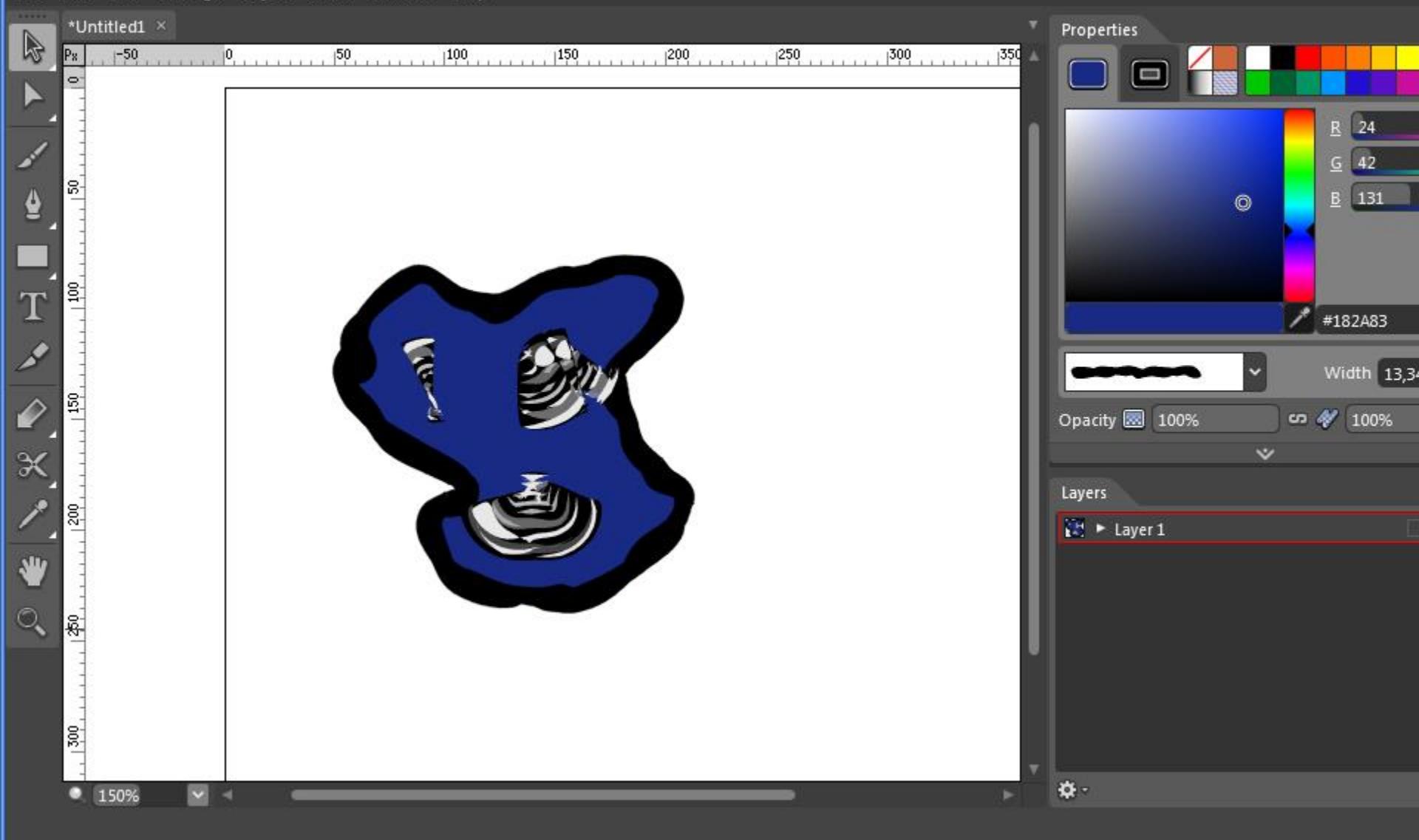
Изображения, пути,
эффекты, импорт (ai, psd)



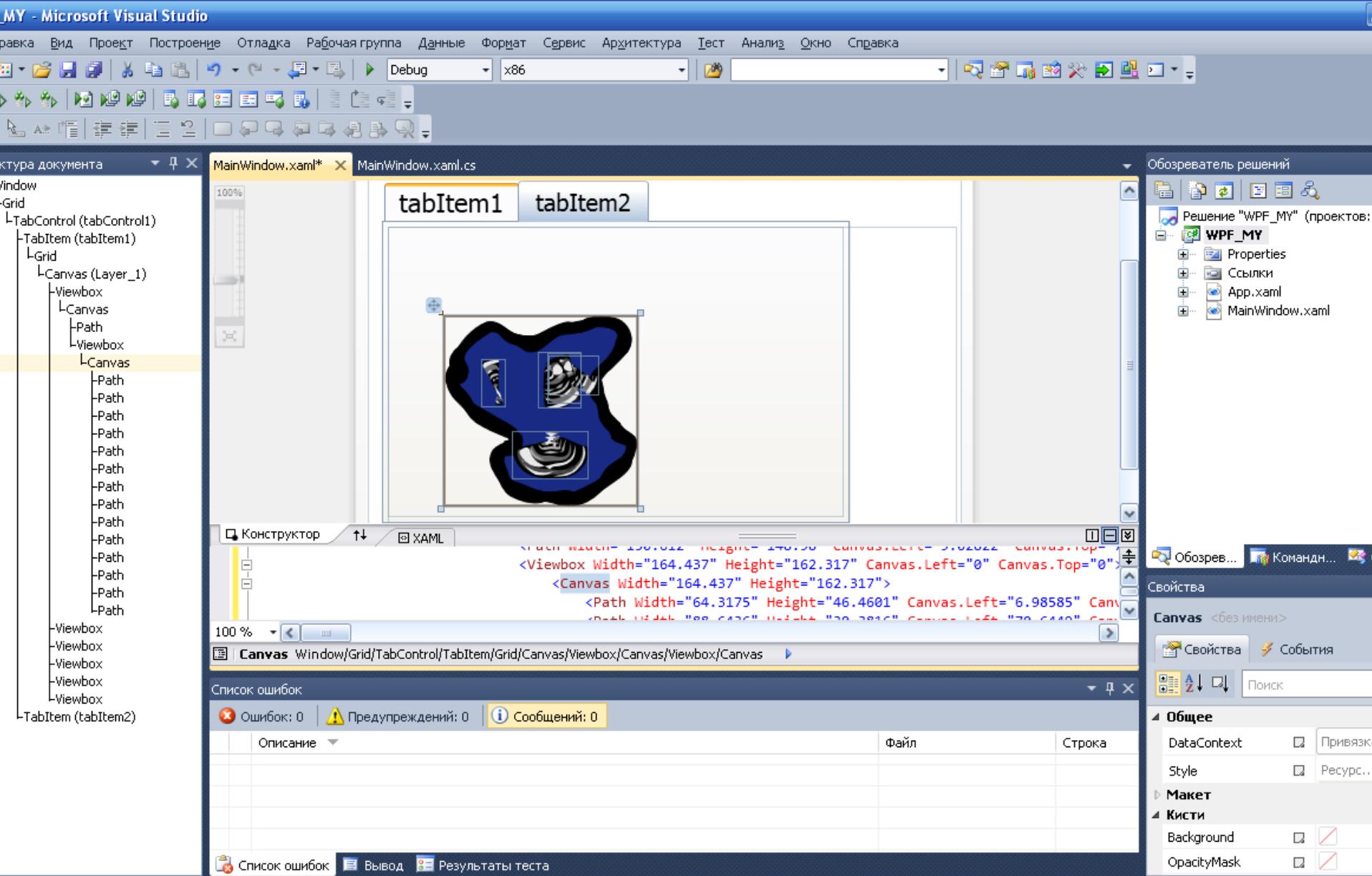
Шаблоны расположения,
Кисти, Шаблоны, Стили,
Ресурсы, Анимации,
Триггеры



XAML intellisense, Отладка,
подключение обработчиков
событий, структура проекта,
source control.



Нарисовать, скопировать



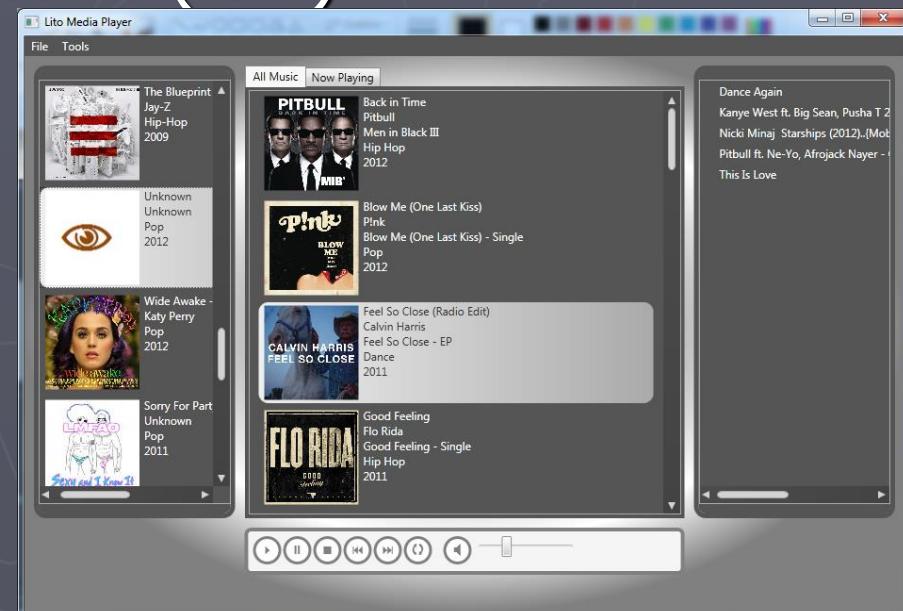
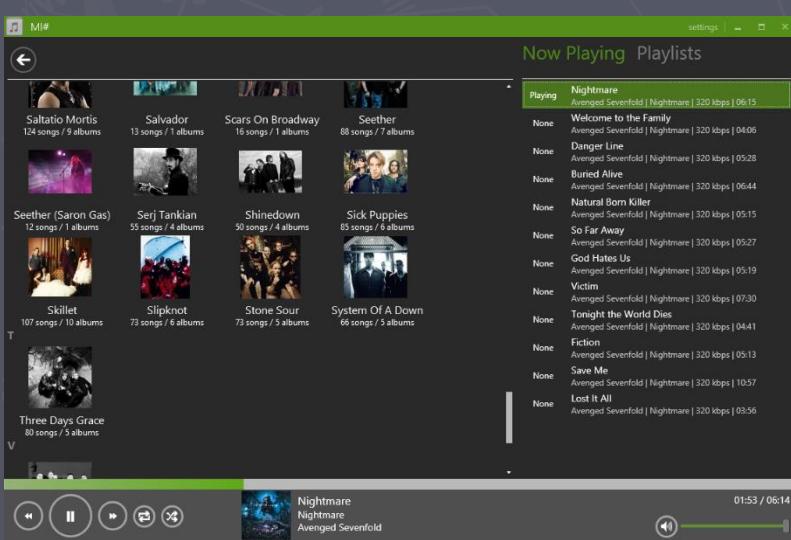
Вставить как XAML- отобразиться

Аудио и Видео

► WPF поддерживает:

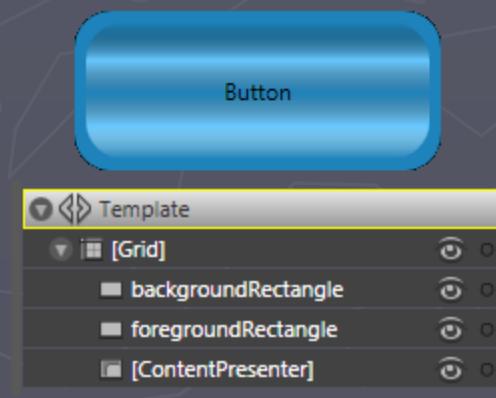
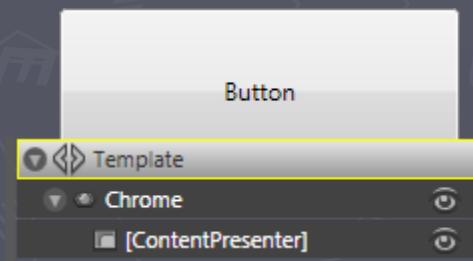
- Windows Media Video (.wmv)
- Advanced Systems Format (.ASF)
- Windows Media Audio (.wma)
- Moving Picture Experts Group (.mpeg)
- Audio Video Interleave (.avi)

■ и др.



Шаблоны (Control Templates)

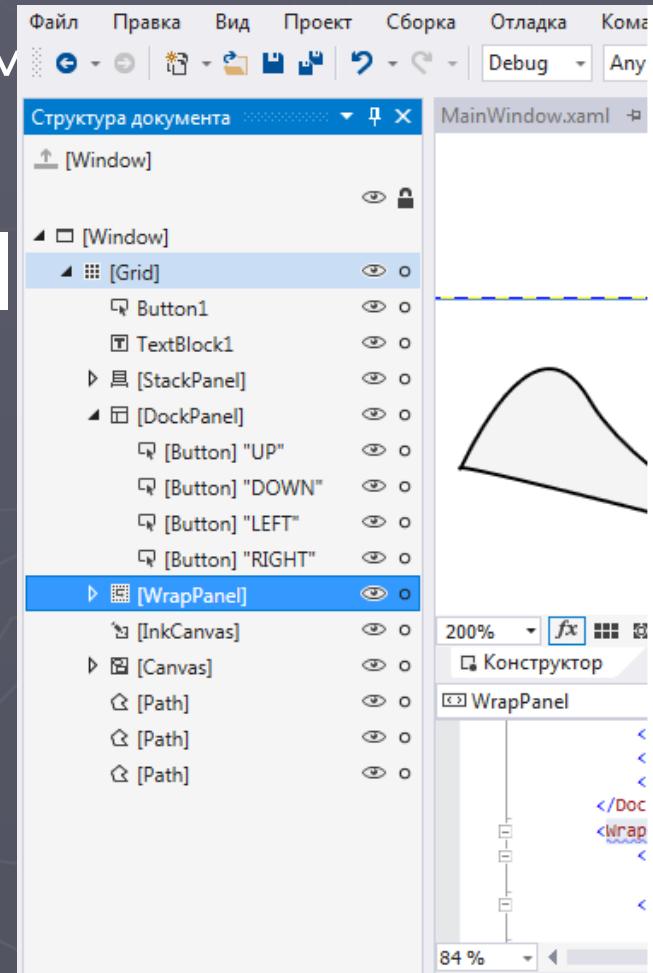
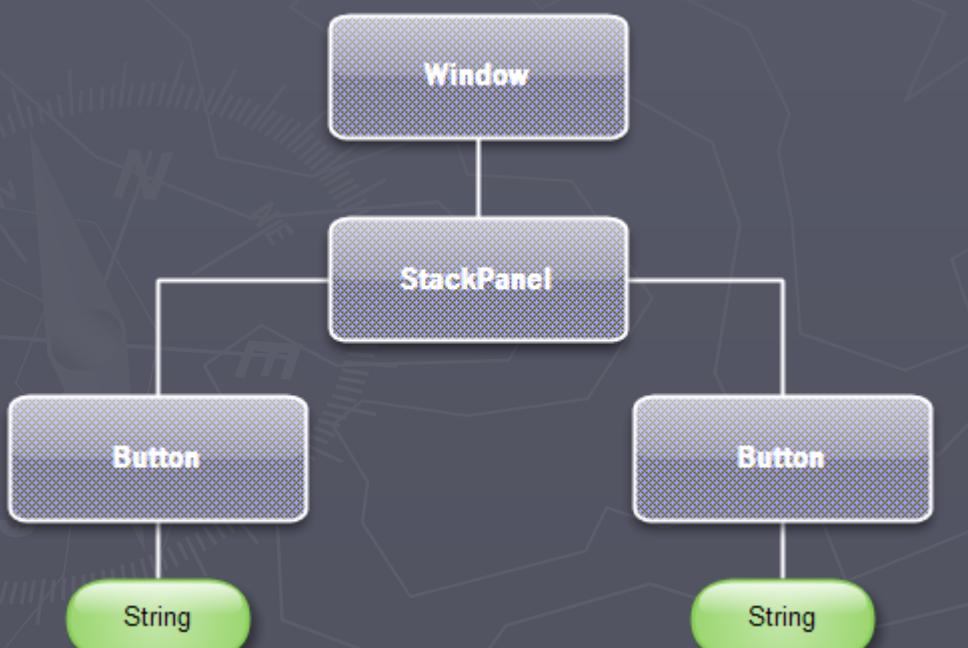
- ▶ Позволяют менять модель визуализации элемента
- ▶ Визуальное дерево шаблона разворачивается для каждого экземпляра элемента



Логические и визуальные деревья

- Множество добавленных элементов называется логическим деревом, образует модель доступа к дочерним элементам

Структура элементов – логическое дерево



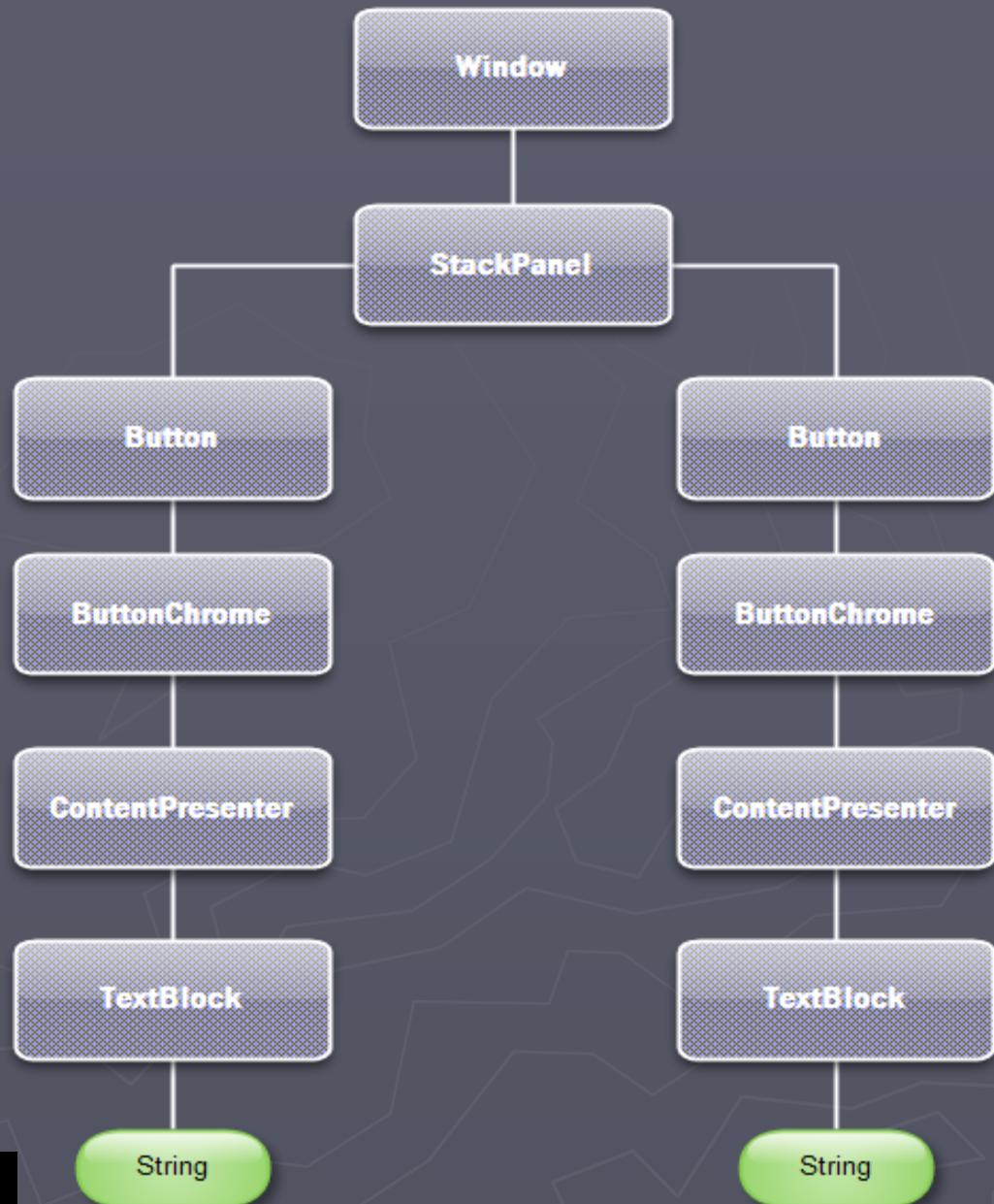
Представлено классом **System.Windows.LogicalTreeHelper**

► Визуальное дерево — это расширенная версия логического дерева.

показывает, как с визуальной точки зрения устроен элемент

представленное классом **System.Windows.Media.VisualTreeHelper**

WPF Spy utility - snoop
(<http://snoopwpf.codeplex.com/>)



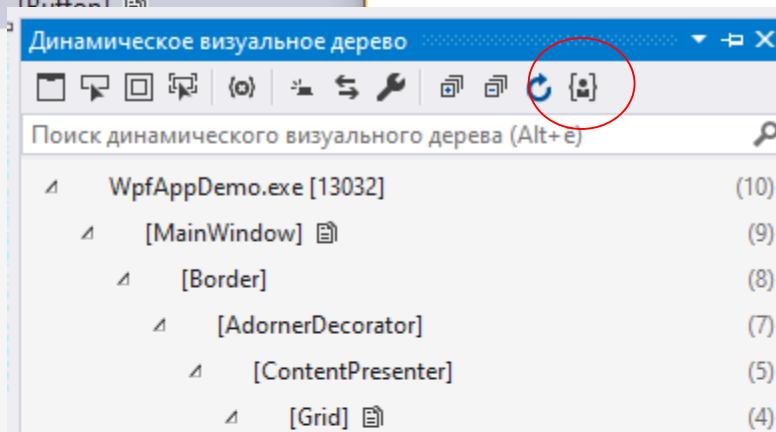
Процесс: [84] WpfAppDemo.exe

MainWindow

Динамическое визуальное дерево

Поиск динамического визуального дерева (Alt+e)

- WpfAppDemo.exe [84]
 - [MainWindow]
 - [Grid]



Динамический обозреватель свойств

Имя <Без имени>
Тип Button

Поиск свойств (Alt+ë)

Свойства

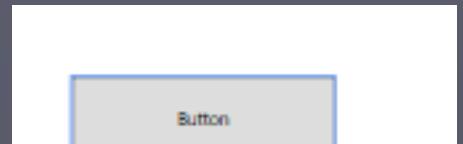
Style (по умолчанию)	SolidColorBrush
Background	SolidColorBrush
BorderBrush	SolidColorBrush
BorderThickness	1,1,1,1
FocusVisualStyle	System.Windows.Style
Foreground	DynamicResourceExt
HorizontalContentAlignment	Center
Padding	1,1,1,1
Template	System.Windows.Control
VerticalContentAlignment	Center

Вычисленные значения

ActualHeight	19.960000000000008
ActualWidth	39.62333333333335
AreAnyTouches...	False

Динамический обозреватель... Средства диагностики

► Все визуальные элементы в WPF имеют встроенные шаблоны



```
<Setter Property="Template">
    <Setter.Value>
        <ControlTemplate TargetType="{x:Type Button}">
            <Border x:Name="border" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness={TemplateBinding BorderThickness} Background="{TemplateBinding Background}" SnapsToDevicePixels="true">
                <ContentPresenter x:Name="contentPresenter" Focusable="False" HorizontalAlignment={TemplateBinding HorizontalContentAlignment} Margin="{TemplateBinding Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}" VerticalContentAlignment={TemplateBinding VerticalContentAlignment} />
            </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsDefaulted" Value="true">
                    <Setter Property="BorderBrush" TargetName="border" Value="{DynamicResource ButtonBorderBrush}" />
                </Trigger>
                <Trigger Property="IsMouseOver" Value="true">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource ButtonBackground}" />
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource ButtonBorderBrush}" />
                </Trigger>
                <Trigger Property="IsPressed" Value="true">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource ButtonBackground}" />
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource ButtonBorderBrush}" />
                </Trigger>
                <Trigger Property="IsEnabled" Value="false">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource ButtonBackground}" />
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource ButtonBorderBrush}" />
                    <Setter Property="TextElement.Foreground" TargetName="contentPresenter" Value="Black" />
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Setter.Value>
```

Шаблон элемента управления

Создание шаблона (в ресурсах):

```
<ControlTemplate x:Key="MyButtonTemplate" TargetType="{x:Type Button}">
    <Border ...>
        <ControlTemplate.Triggers ...>
    </ControlTemplate>
```

Использование шаблона для кнопки:

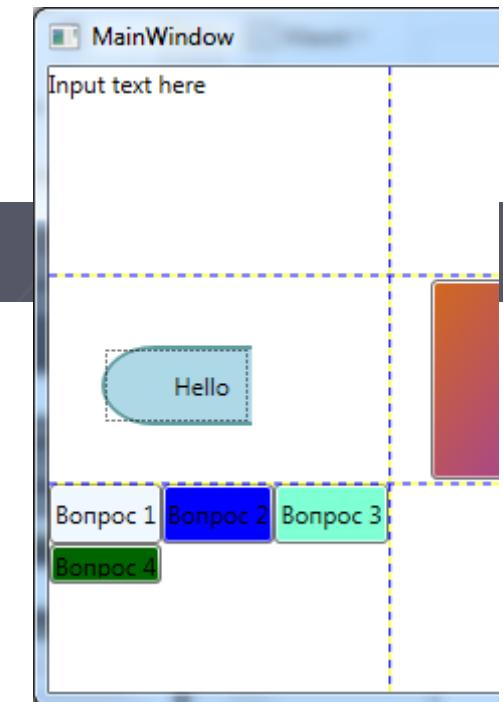
```
<Button Template="{StaticResource MyButtonTemplate}">OK</Button>
```

Варианты определения :

- 1) через стили
- 2) в виде отдельных ресурсов

```
<Window.Resources>
    <ControlTemplate TargetType="Button" x:Key="MyButtonTemplate">
        <Border CornerRadius="25"
            BorderBrush="CadetBlue"
            BorderThickness="2"
            Background="LightBlue" Height="40" Width="100" >
            <ContentControl Margin="5"
                HorizontalAlignment="Center"
                VerticalAlignment="Center"
                Content="Hello" />
        </Border>
    </ControlTemplate>
</Window.Resources>
```

```
<Button x:Name="button"
        Content="Button"
        HorizontalAlignment="Left"
        Margin="26,35,0,0"
        Grid.Row="1"
        VerticalAlignment="Top"
        Width="75"
        Template="{StaticResource MyButtonTemplate}" />
```



```

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

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Shared.xaml" />
    </ResourceDictionary.MergedDictionaries>

    <!-- Focus Visual -->

    <Style x:Key="ButtonFocusVisual">
        <Setter Property="Control.Template">
            <Setter.Value>
                <ControlTemplate>
                    <Border>
                        <Rectangle
                            Margin="2"
                            StrokeThickness="1"
                            Stroke="#60000000"
                            StrokeDashArray="1 2"/>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <!-- SimpleStyles: Button -->

    <Style TargetType="Button">
        <Setter Property="SnapsToDevicePixels" Value="true"/>
        <Setter Property="OverridesDefaultStyle" Value="true"/>
        <Setter Property="FocusVisualStyle" Value="{StaticResource ButtonFocusVisual}" />
        <Setter Property="MinHeight" Value="23"/>
        <Setter Property="MinWidth" Value="75"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border
                        x:Name="Border"
                        CornerRadius="2"
                        BorderThickness="1"
                        Background="{StaticResource NormalBrush}"
                        BorderBrush="{StaticResource NormalBorderBrush}">
                        <ContentPresenter
                            Margin="2"
                            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">

```



```
<Application x:Class="WpfTempl.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:WpfTempl"
    StartupUri="MainWindow.xaml">
<Application.Resources>
    <ControlTemplate TargetType="Button" x:Key="PnvTemplate">
        <Border CornerRadius="5"
            BorderBrush="Chocolate"
            BorderThickness="6"
            Background="LightSeaGreen"
            Height="100" Width="200" >
            <ContentControl Margin="5"
                HorizontalAlignment="Center"
                VerticalAlignment="Center" />
        </Border>
    </ControlTemplate>
```



устанавливает параметры, которые
нельзя изменить

```
<Grid>
    <Button Content="Button"
        HorizontalAlignment="Left" Margin="79,57,0,0"
        VerticalAlignment="Top" Width="302" Height="154"
        Template="{DynamicResource PnvTemplate}"/>
</Grid>
```

Пример создания шаблона для кнопки

► Свойство TemplateBinding

Для влияния из элемента, к которому применяется шаблон на свойства, определенные в шаблоне
Для установки в шаблоне привязки к свойствам элемента.

```
<ControlTemplate TargetType="Button" x:Key="PnvTemplate">
    <Border CornerRadius= "5"
        BorderBrush="{TemplateBinding BorderBrush}"
        BorderThickness="{TemplateBinding BorderThickness}"
        Background="{TemplateBinding Background}"
        Height="100" Width="200" >
    </Border>
</ControlTemplate>
```

фон элемента Border будет привязан к
свойству Background элемента Button



```
<Button Content="Button" HorizontalAlignment="Left" Margin="79,57,0,0"
    VerticalAlignment="Top" Width="302" Height="154"
    Background="Khaki"
    BorderBrush="BlueViolet"
    Template="{DynamicResource PnvTemplate}"/>
```

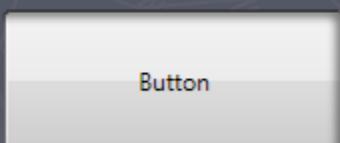
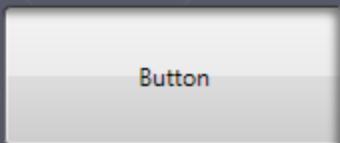
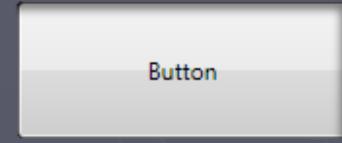
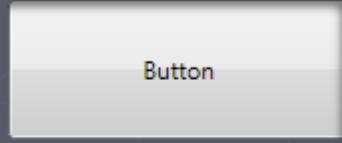
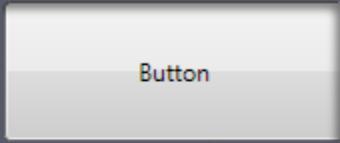
► Свойство Template

Позволяет определить шаблон напрямую в самом элементе

```
<Button Content="Button" HorizontalAlignment="Left" Margin="79,57,0,0"
        VerticalAlignment="Top" Width="302" Height="154"
        Background="Khaki"
        BorderBrush="BlueViolet"
        >
    <Button.Template>
        ←
        <ControlTemplate TargetType="Button">
            <Border CornerRadius="25"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="{TemplateBinding BorderThickness}"
                    Background="{TemplateBinding Background}"
                    Height="{TemplateBinding Height}"
                    Width="{TemplateBinding Width}" />
        </ControlTemplate>
    </Button.Template>
```

```
<Style TargetType="{x:Type Button}">
    <Setter Property="Control.Template" Value="{StaticResource ButtonTemplate}">
</Style>
```

Элементы управления, Стили, Шаблоны и Ресурсы



Элементы
управления
(Controls)

Стили
(Styles)

Шаблоны
(Templates)

Документы

► Фиксированные документы (fixed documents)

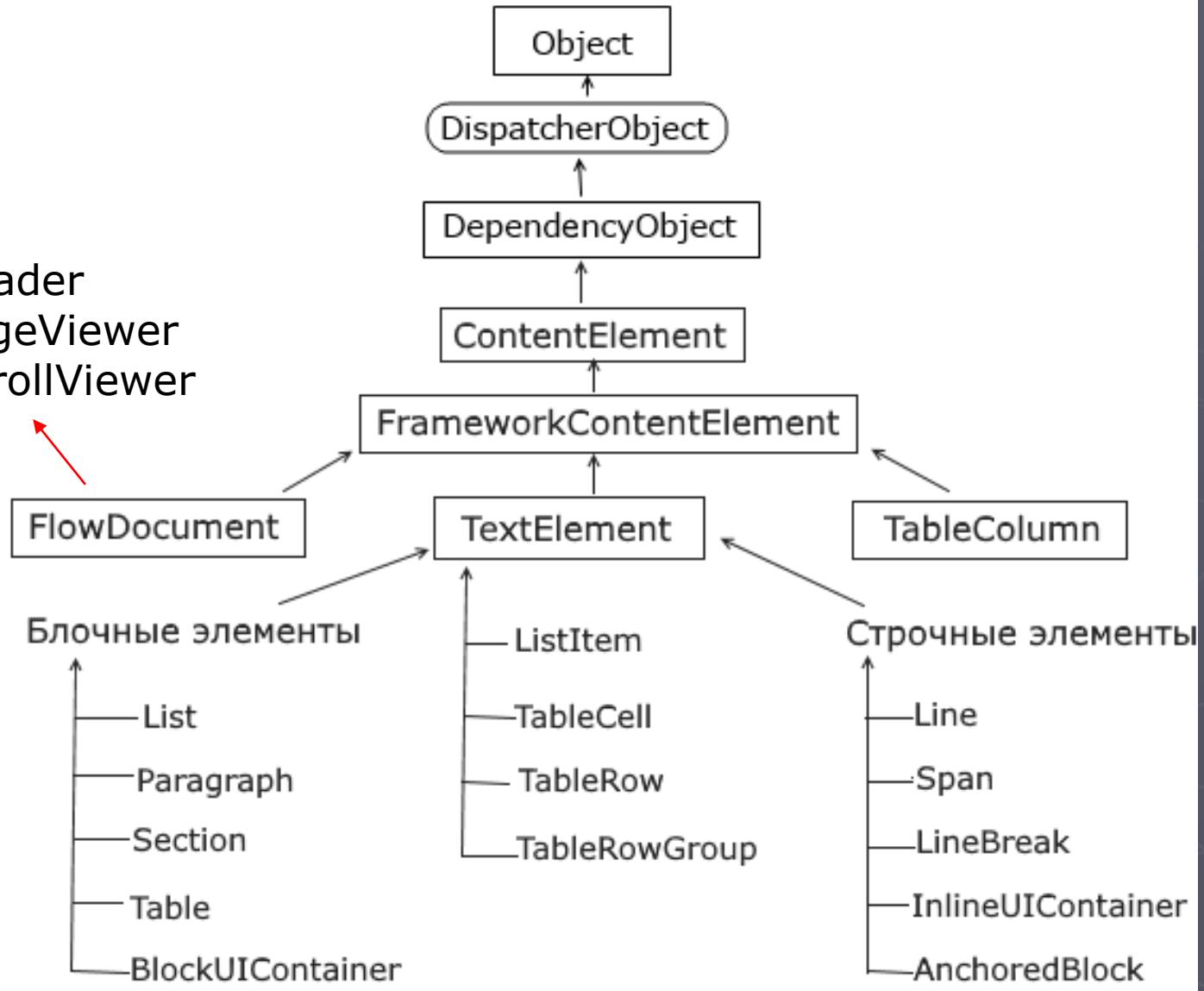
- для печати
- формат и расположение не может быть изменено
- будут выглядеть одинаково
- не оптимизированы
- использует стандарт XPS (XML Paper Specification)

► Потоковые документы (flow documents)

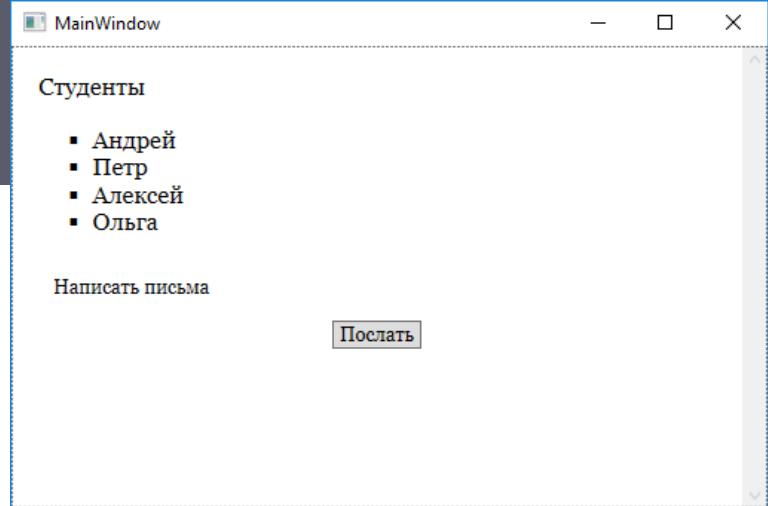
- для просмотра на экране
- выполняет оптимизацию документа под конкретные параметры среды

Потоковые документы

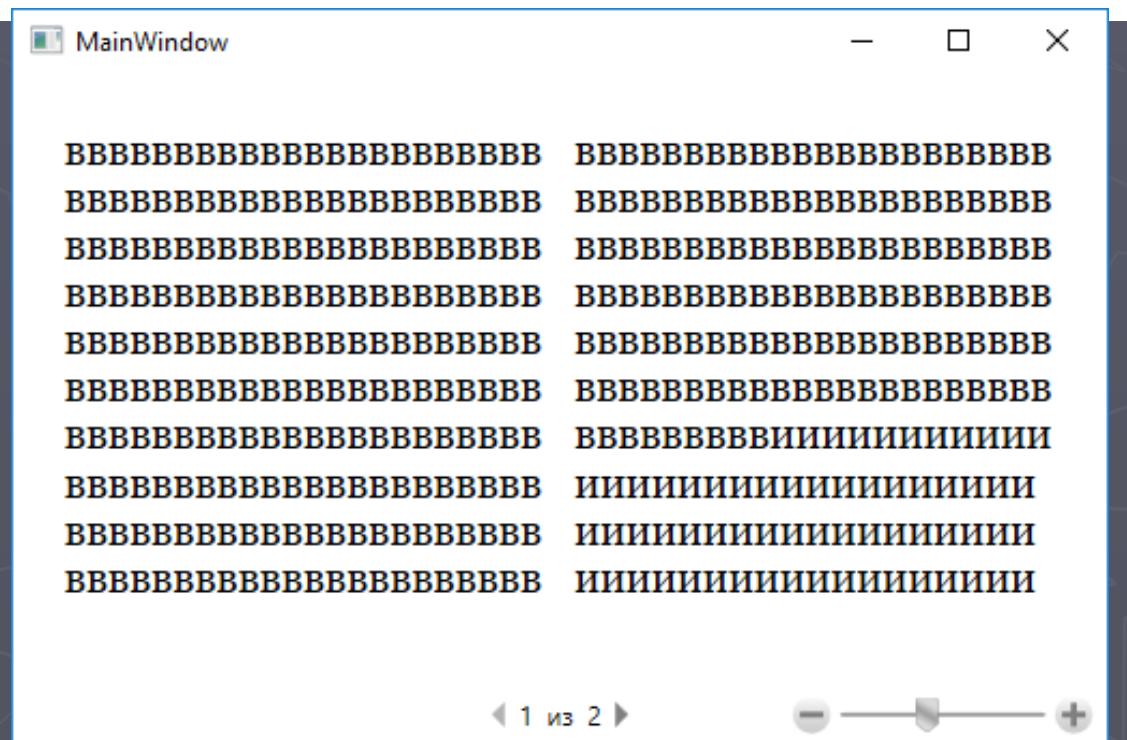
Контейнеры:
FlowDocumentReader
FlowDocumentPageViewer
FlowDocumentScrollView

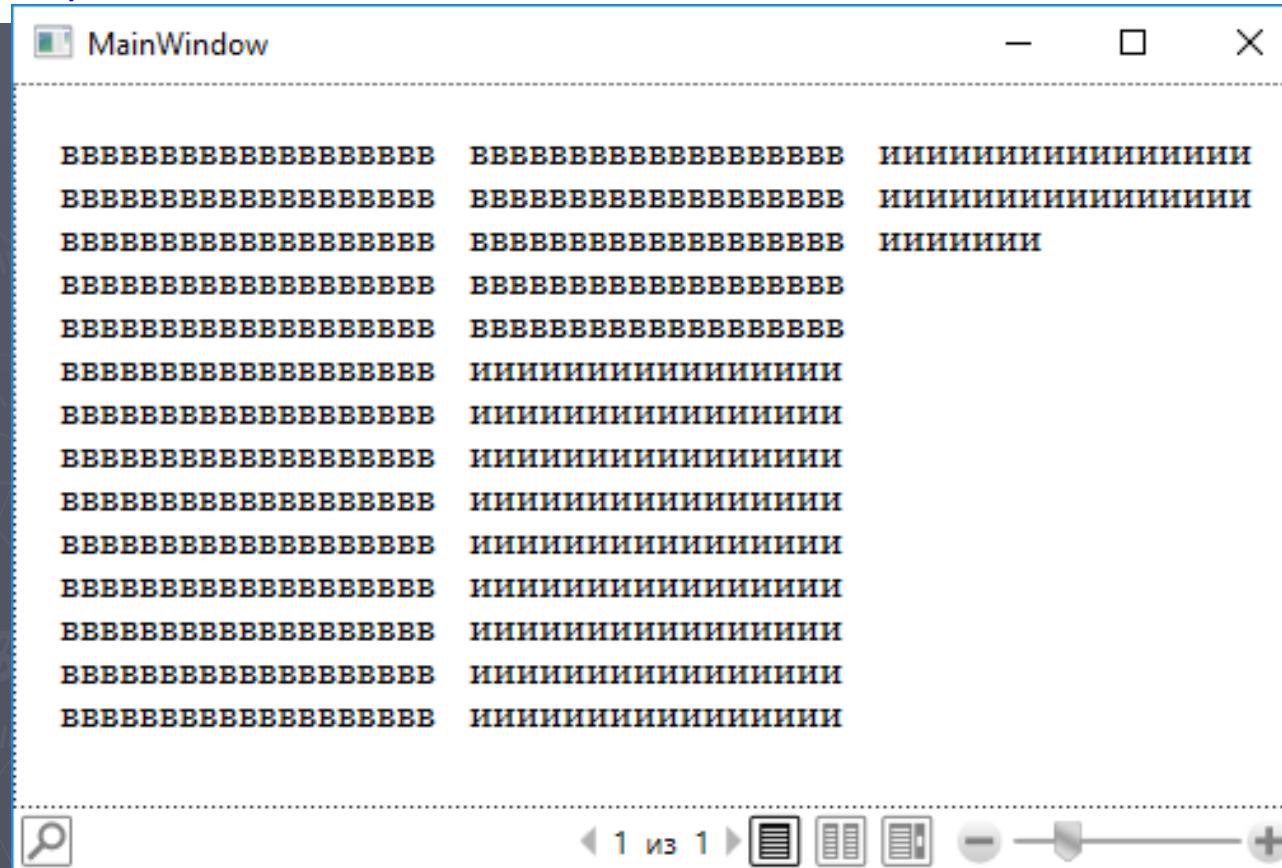


```
<FlowDocumentScrollViewer>
    <FlowDocument>
        <Paragraph>Студенты</Paragraph>
        <List MarkerStyle="Box">
            <ListItem>
                <Paragraph>Андрей</Paragraph>
            </ListItem>
            <ListItem>
                <Paragraph>Петр</Paragraph>
            </ListItem>
            <ListItem>
                <Paragraph>Алексей</Paragraph>
            </ListItem>
            <ListItem>
                <Paragraph>Ольга</Paragraph>
            </ListItem>
        </List>
        <BlockUIContainer FontSize="13">
            <StackPanel Orientation="Vertical">
                <TextBlock Height="40" Padding="10">Написать письма</TextBlock>
                <Button Width="60">Послать</Button>
            </StackPanel>
        </BlockUIContainer>
    </FlowDocument>
</FlowDocumentScrollViewer>
```



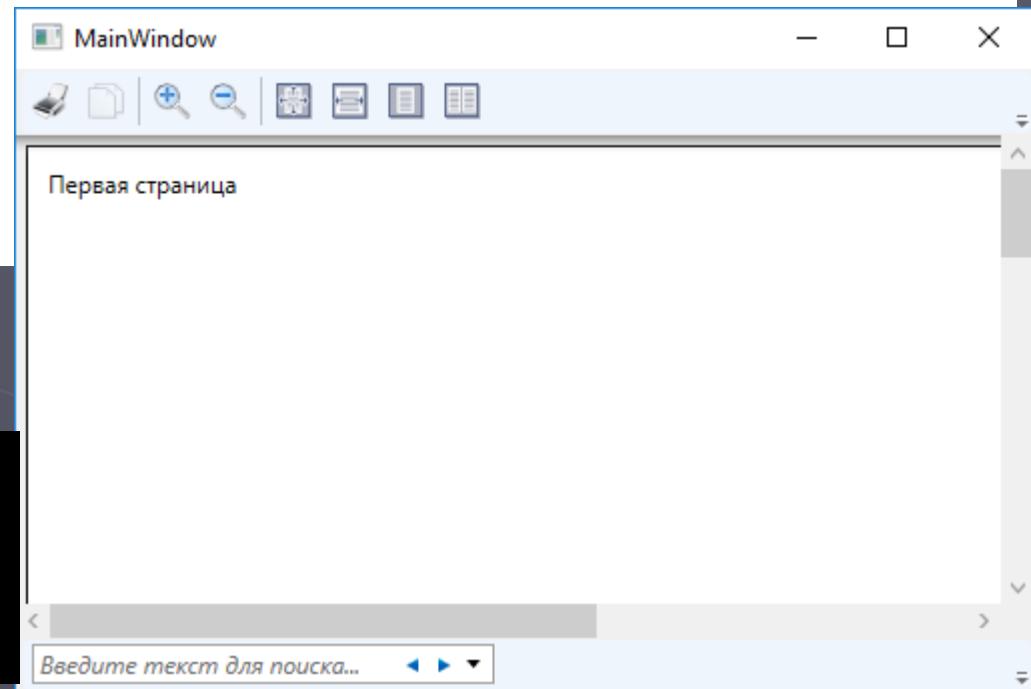
```
<FlowDocumentPageViewer>
    <FlowDocument ColumnWidth="150" ColumnGap="10">
        <Paragraph TextAlignment="Left" FontSize="15">
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBB...
        </Paragraph>
    </FlowDocument>
</FlowDocumentPageViewer>
```





Фиксированные документы

```
<DocumentViewer>
    <FixedDocument>
        <PageContent>
            <FixedPage>
                <Grid Margin="10" Width="450" Height="600">
                    <TextBlock Text="Первая страница" />
                    <Rectangle Stroke="Green" Width="50"
                               Height="50" Fill="Green" />
                </Grid>
            </FixedPage>
        </PageContent>
    </FixedDocument>
</DocumentViewer>
```



точная неизменная компоновка и
предназначены для печати,
могут использоваться для чтения
текста