

WPF Windows Presentation Foundation

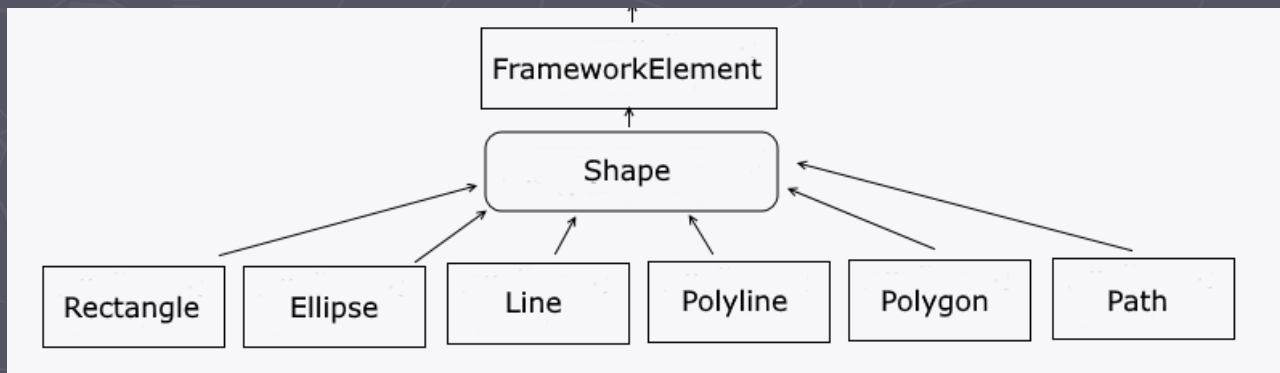
3 часть

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

LinearGradientBrush
RadialGradientBrush
ImageBrush
DrawingBrush
VisualBrush

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

PathGeometry
PathFigure

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

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

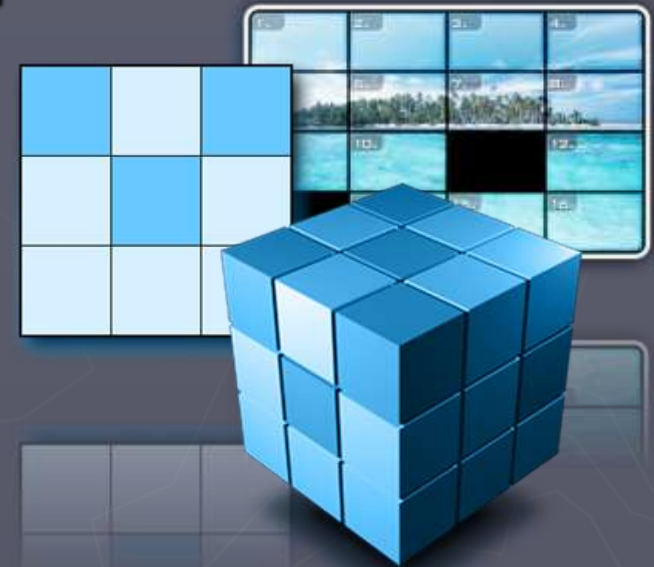
```
<Ellipse Fill="Blue" Width="20" Height="200" />
```

```
<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>
```

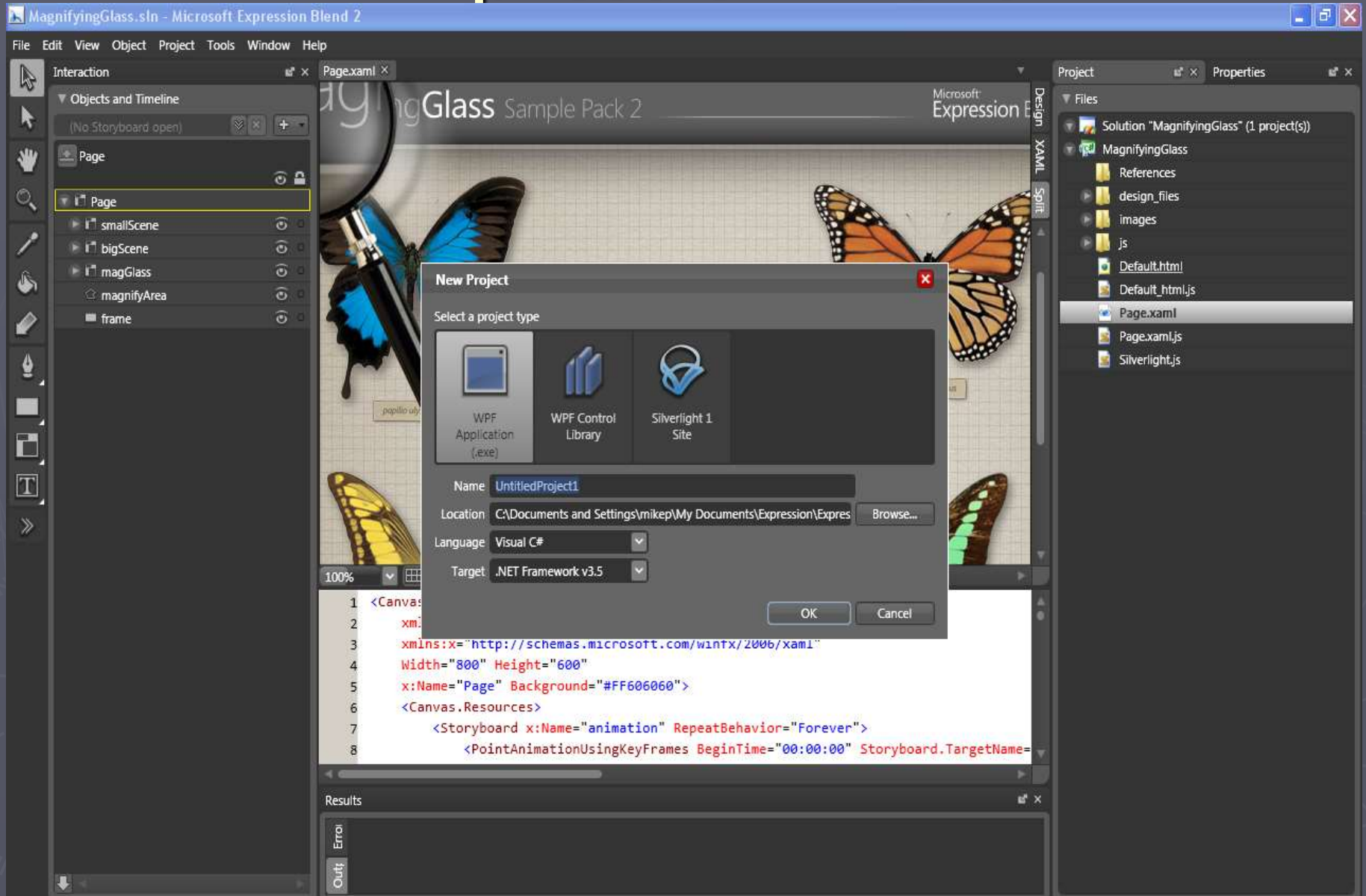
```
        <Viewport3D.Children>
```

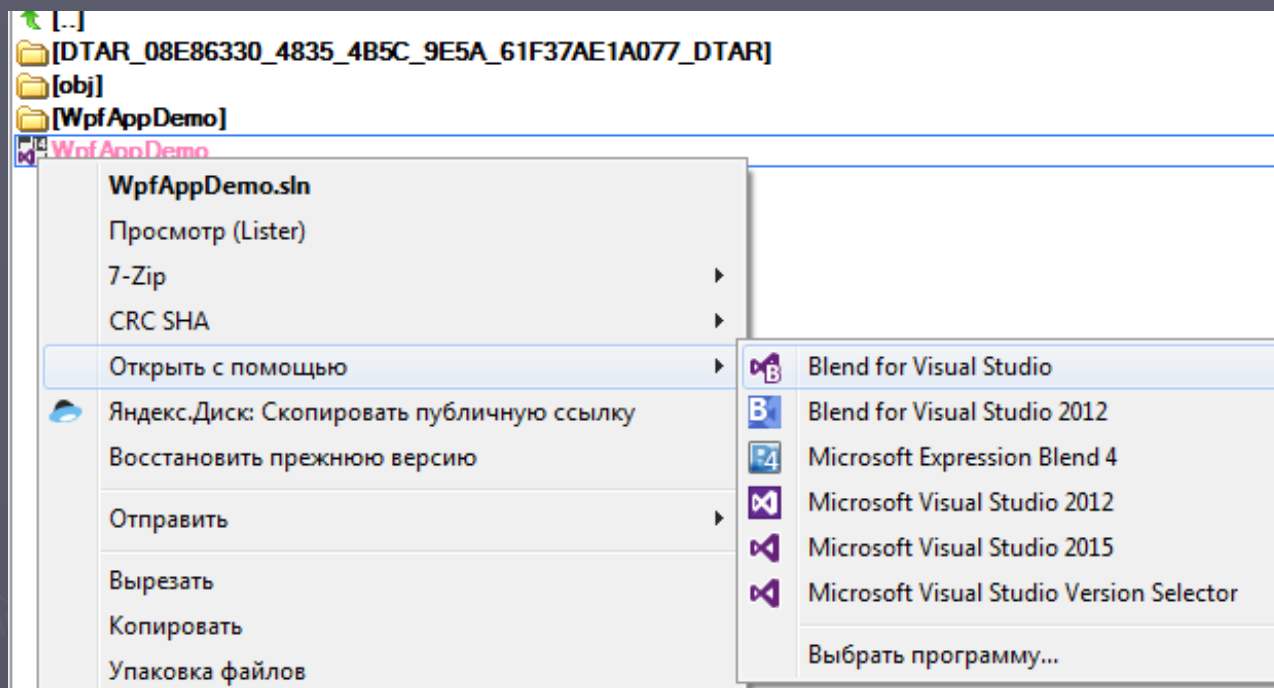
```
            <ModelVisual3D>
```

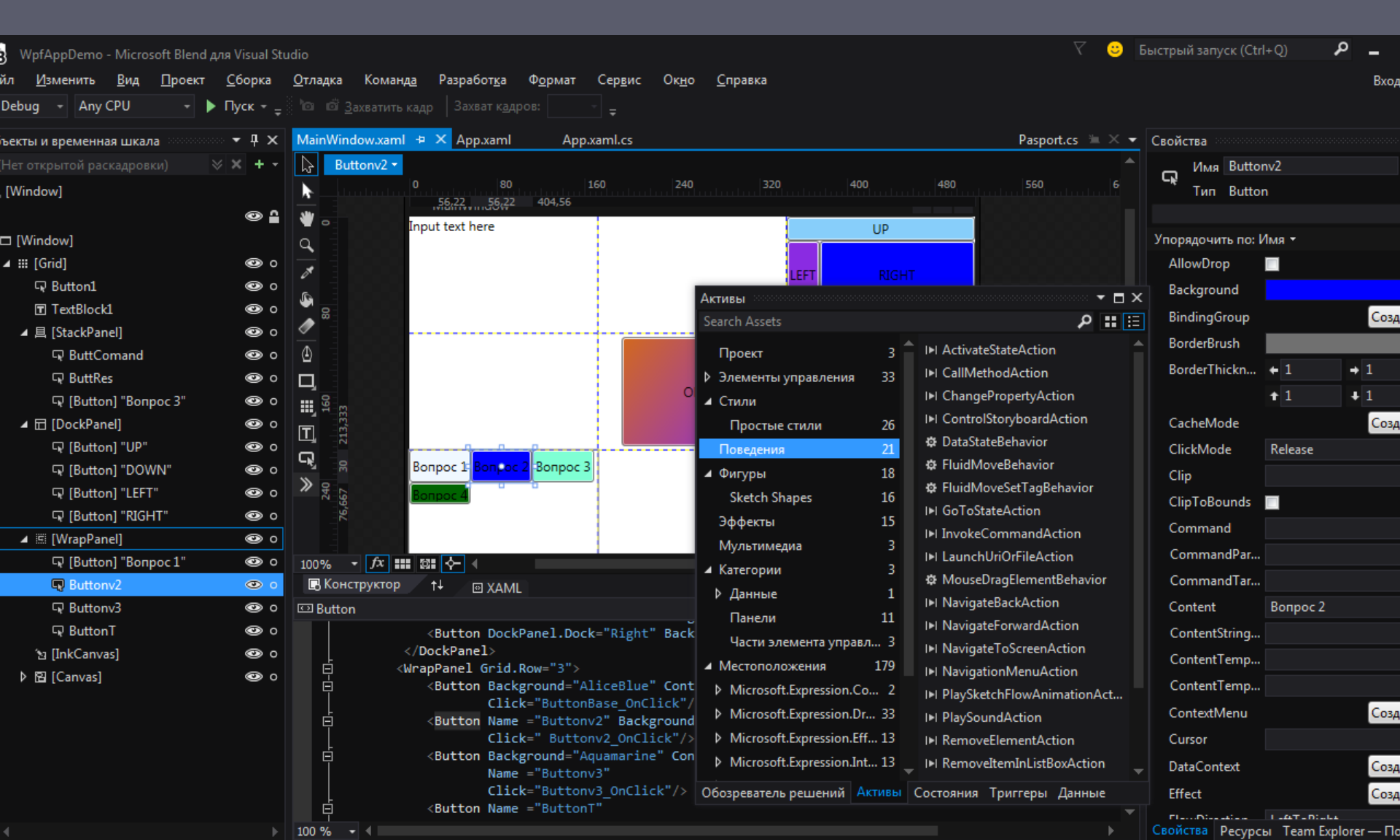
```
                <ModelVisual3D.Content>
```

```
                    <GeometryModel3D>
```

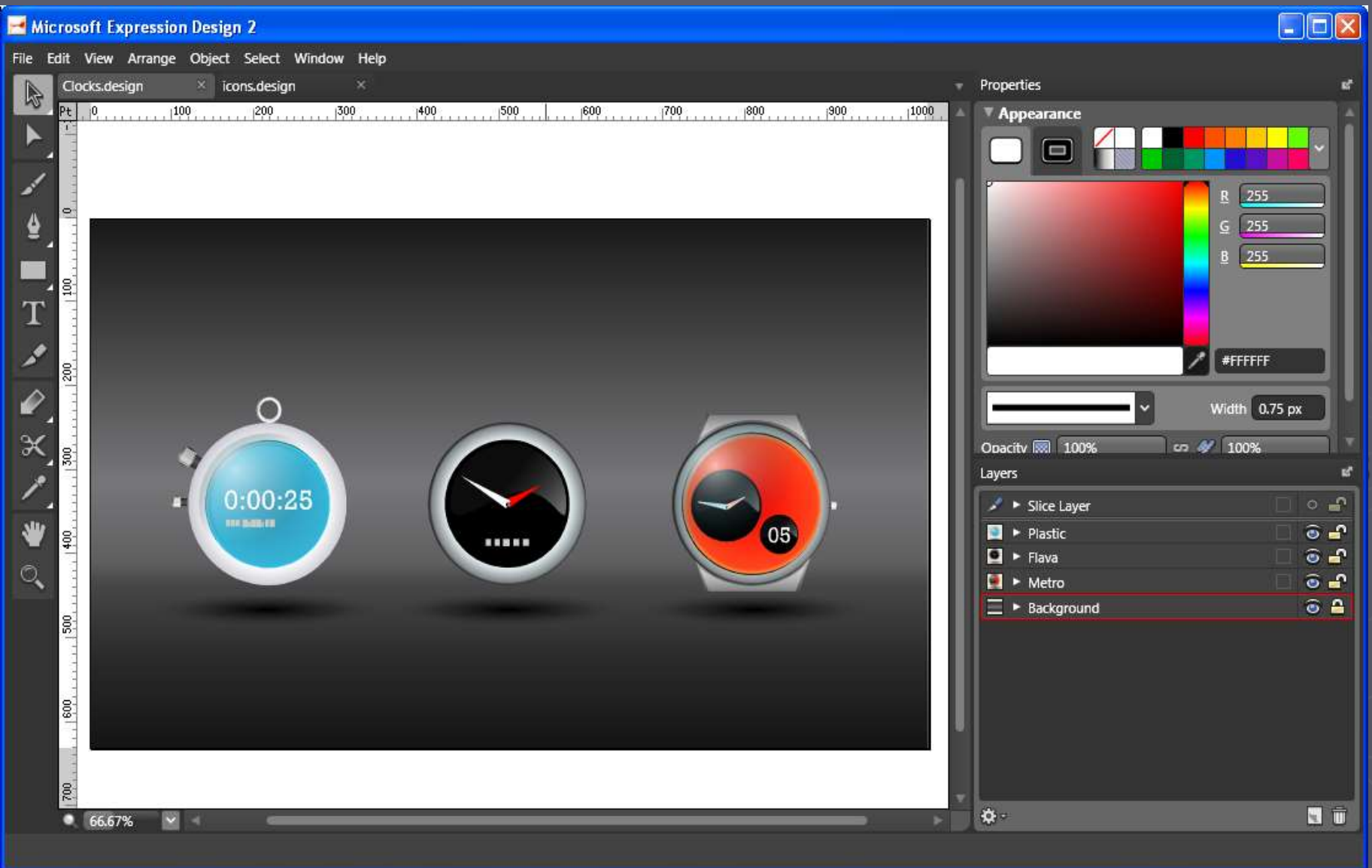
Expression Blend

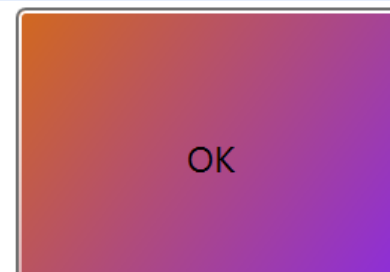
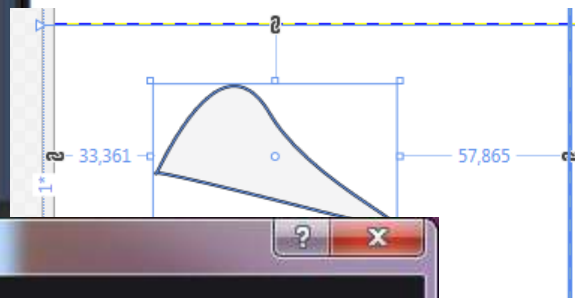
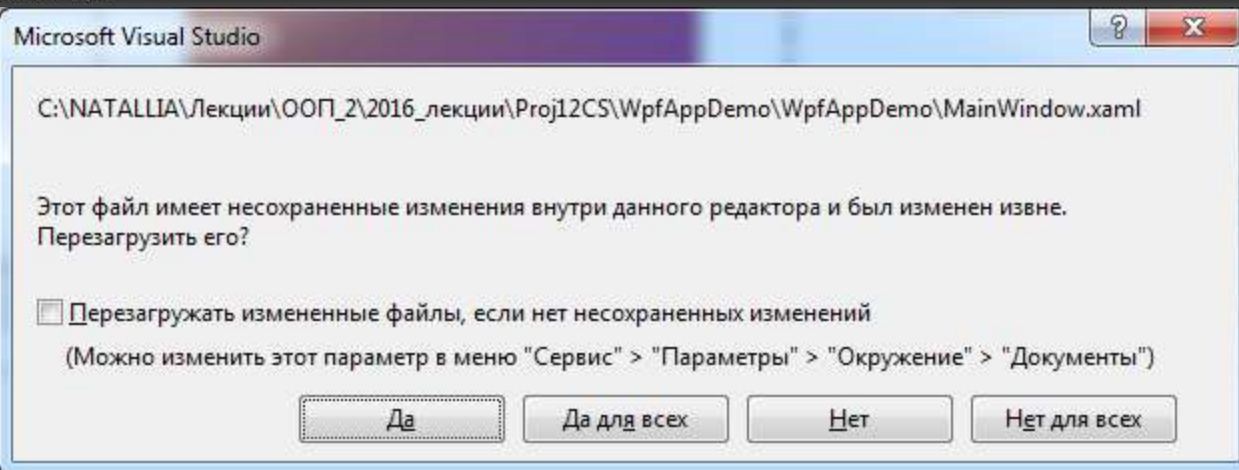
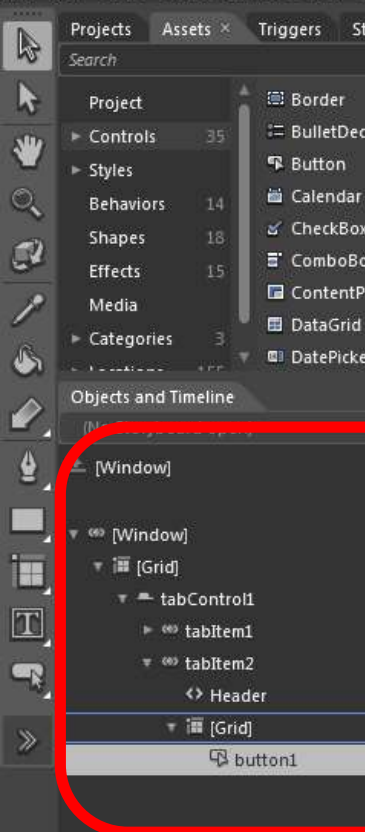






Expression Design





Вопрос 1

Вопрос 3

Вопрос 4

Преобразовать в UserControl

Имя UserControl1

Предупреждение. Преобразование выбранного элемента в UserControl может привести к нарушенным ссылкам в новом элементе UserControl и/или нарушенным ссылкам в исходном документе. Примерами нарушенных ссылок могут быть имена, ресурсы и обработчики событий.

```
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" Width="100"/>  
HorizontalAlignment="Left" Height="1" Margin="85,0,0,32.833" Grid.Row="1" Stretch="Fill" Stroke="Black" VerticalAlignment="Top" Width="100"/>
```



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

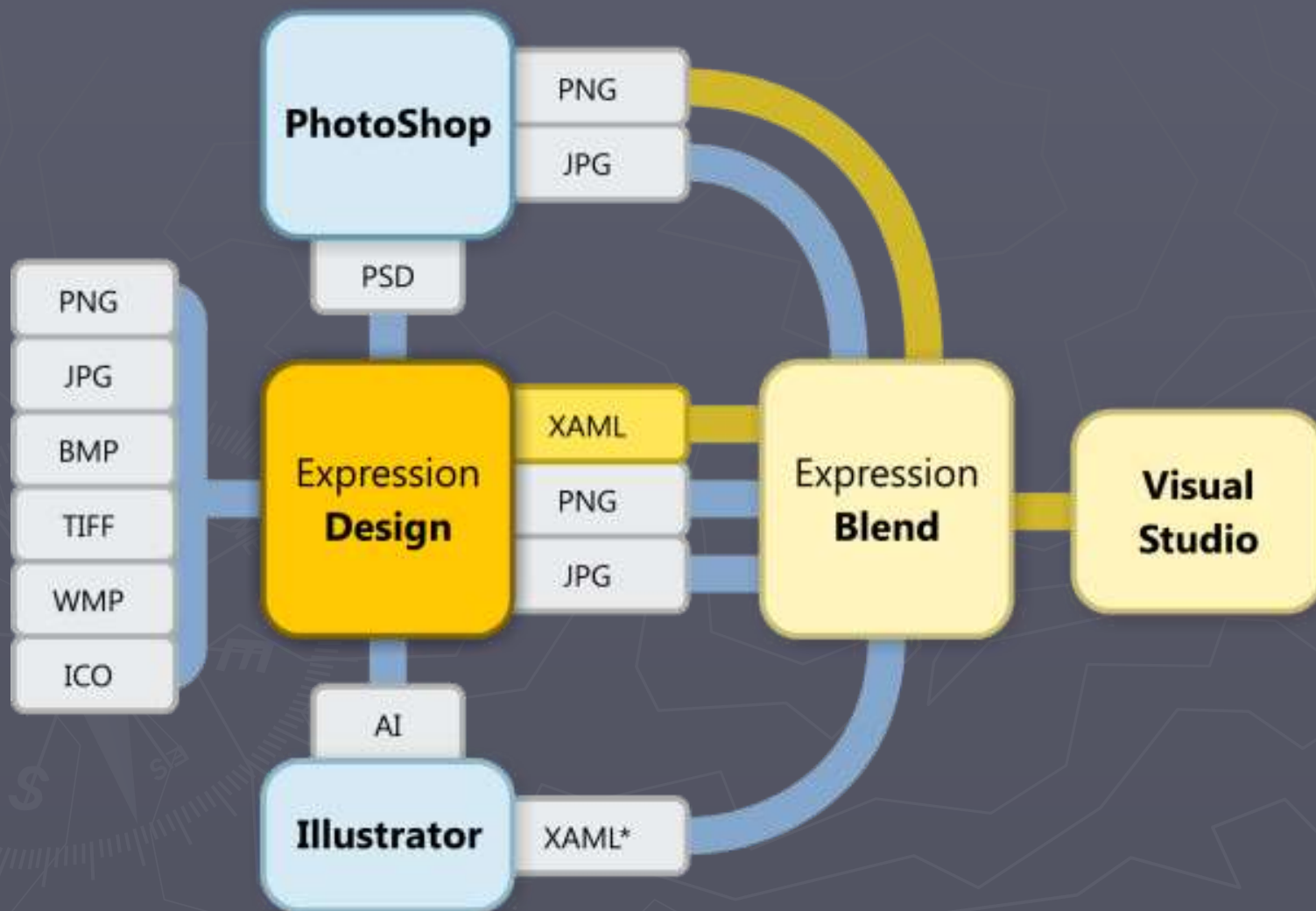


Шаблоны расположения,
Кисти, Шаблоны, Стили,
Ресурсы, Анимации,
Триггеры

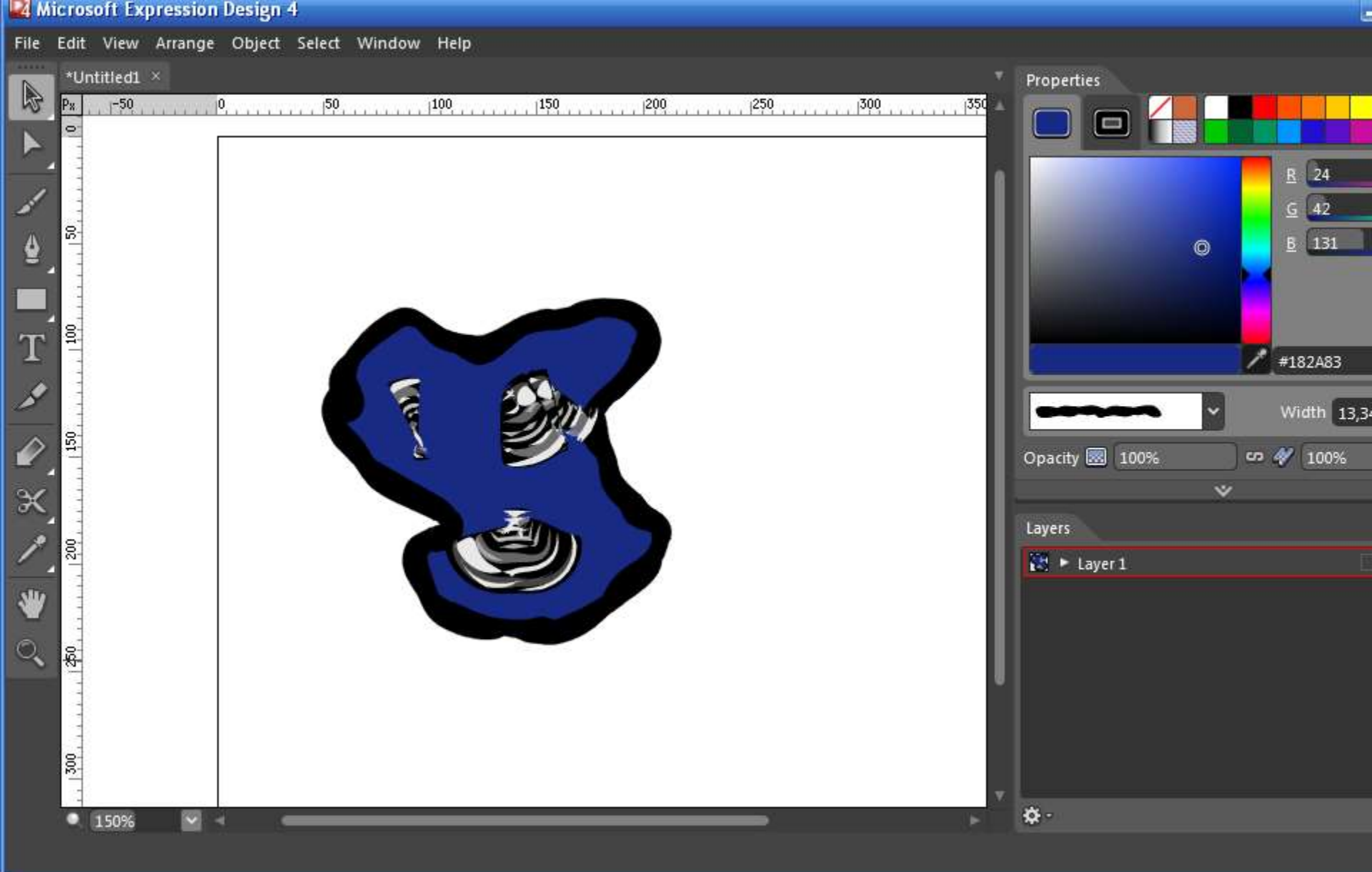


XAML intellisense, Отладка,
подключение обработчиков
событий, структура проекта,
source control.

Работа графического дизайнера



*Upcoming 3rd party plugins



Нарисовать, скопировать

Вставить как XAML-отобразиться

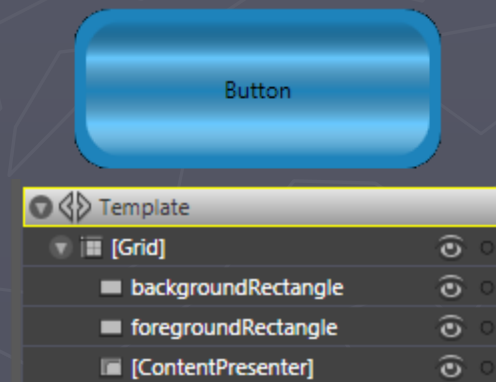
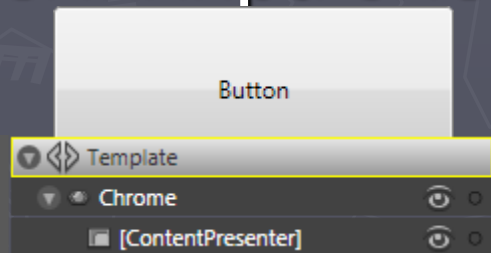
Аудио и Видео

► 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.

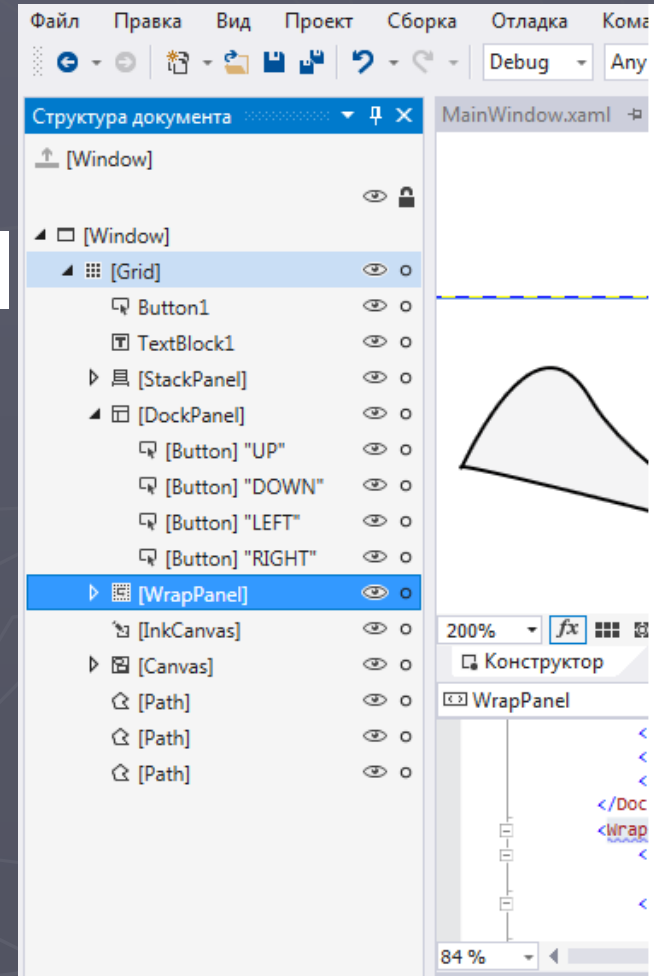
Шаблоны (Control Templates)

- ▶ Визуальный скелет элемента управления
- ▶ Позволяют полностью менять модель визуализации элемента
- ▶ Визуальное дерево шаблона разворачивается для каждого экземпляра элемента



Логические и визуальные деревья

- ▶ Множество добавленных элементов называется логическим деревом, Структура элементов – логическое дерево



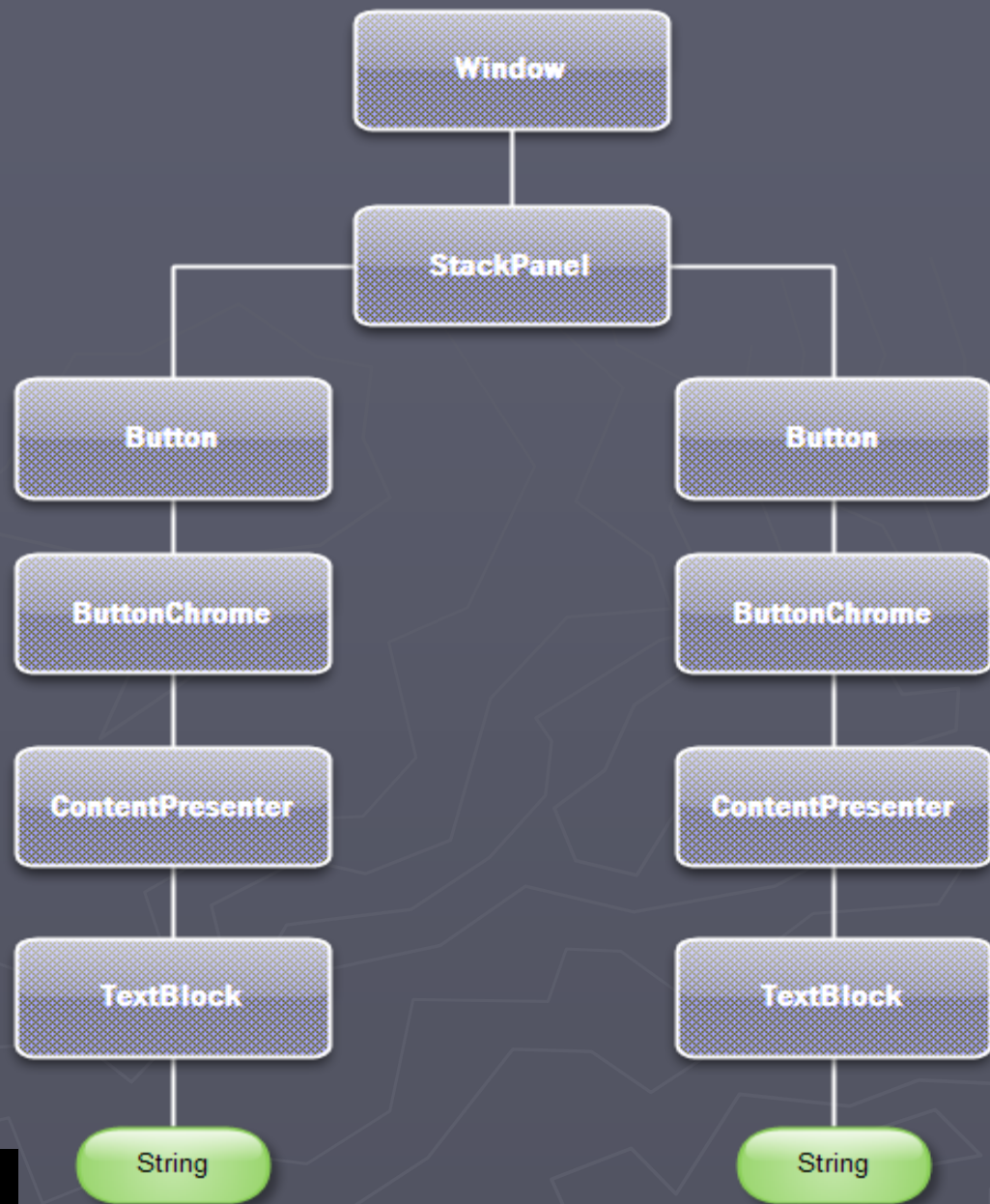
Представлено классом `System.Windows.LogicalTreeHelper`

► Визуальное
дерево — это
расширенная
версия
логического
дерева.

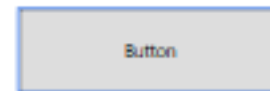
показывает, как с
визуальной точки зрения
устроен элемент

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

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



► 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=
Background}" SnapsToDevicePixels="true">
                <ContentPresenter x:Name="contentPresenter" Focusable="False" HorizontalAlignmen
Margin="{TemplateBinding Padding}" RecognizesAccessKey="True" SnapsToDevicePixels="{TemplateBinding SnapsToD
VerticalContentAlignment}"/>
            </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsDefaulted" Value="true">
                    <Setter Property="BorderBrush" TargetName="border" Value="{DynamicResource {
                </Trigger>
                <Trigger Property="IsMouseOver" Value="true">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource But
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Bu
                </Trigger>
                <Trigger Property="IsPressed" Value="true">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource But
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Bu
                </Trigger>
                <Trigger Property="IsEnabled" Value="false">
                    <Setter Property="Background" TargetName="border" Value="{StaticResource But
                    <Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Bu
                    <Setter Property="TextElement.Foreground" TargetName="contentPresenter" Valu
                </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 :

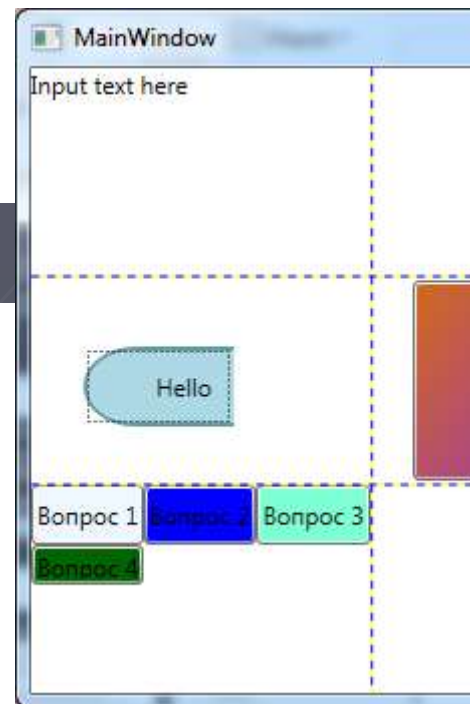
1) через стили

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}">/>

```

```

<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>

```

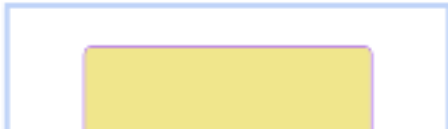
Пример создания шаблона для кнопки

► 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}"/>
```

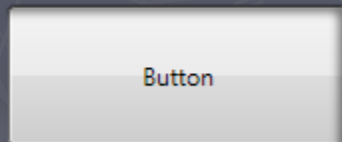
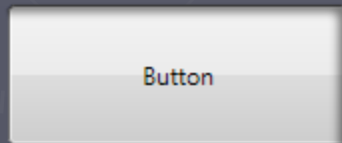
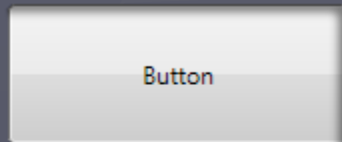
► Свойство Template

Позволяет определить шаблон напрямую в самом элементе

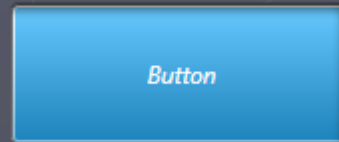
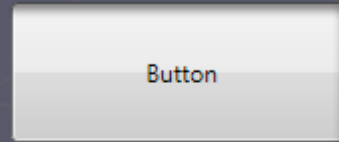
```
<Button Content="Button" HorizontalAlignment="Left" Margin="79,57,0,0"
    VerticalAlignment="Top" Width="302" Height="154"
    Background="Khaki"
    BorderBrush="BlueViolet"
    >
<Button.Template>
    ←
    <ControlTemplate TargetType="Button">
    <Border CornerRadius="25"
    BorderBrush="{TemplateBinding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

</Button.Template>
```

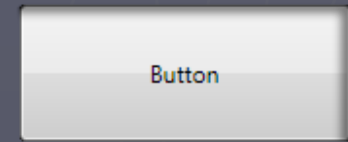

Элементы управления, Стили, Шаблоны и Ресурсы



**Элементы
управления
(Controls)**



**Стили
(Styles)**



**Шаблоны
(Templates)**

Привязка Binding

Источник

Приемник

в случае модификации
приемник также будет
модифицирован

создает привязку к
определенному свойству
объекта-источника



```
<TextBlock x:Name="TextBlock1"
```

```
Text="{Binding ElementName=Button1,Path=Content}"  
Height="30" />
```

Целевой объект привязки

Источник привязки

```
{Binding ElementName=Имя_объекта-источника, Path=Свойство_объекта-источника}
```

```
Binding binding = new Binding();
```





```
binding.ElementName = "TextBox1"; // источник  
binding.Path = new PropertyPath("Text"); // свойство  
TextBlock1.SetBinding(TextBlock.TextProperty, binding);  
// установка привязки
```

- ▶ **ElementName**: имя элемента, к которому создается привязка
- ▶ **IsAsync**: асинхронный режим (по умолчанию равно False)
- ▶ **Mode**: режим привязки
- ▶ **Path**: ссылка на свойство объекта, к которому идет привязка
- ▶ **TargetNullValue**: устанавливает значение по умолчанию, если привязанное свойство источника привязки имеет значение null

TargetNullValue="по умолчанию"

- ▶ **RelativeSource**: создает привязку относительно текущего объекта
- ▶ **Source**: указывает на объект-источник, если он не является элементом управления.
- ▶ **XPath**: используется вместо свойства path для указания пути к xml-данным

Направление привязок Mode

- ▶ **OneWay**—целевое свойство обновляется при изменении значения источника.

- ▶ **TwoWay**—при изменении источника меняется целевое свойство и наоборот.

- ▶ **OneTime**—целевое свойство устанавливается изначально на основе свойства источника и с этого момента изменения значений в источнике игнорируются.

- ▶ **OneWayToSource**—свойство источника обновляется при изменении целевого свойства.

- ▶ **Default**—тип привязки зависит от целевого свойства. `TextBox.Text`—`TwoWay` для всех прочих `OneWay`.

```
<TextBlock x:Name="TextBlock"  
    Text="{Binding  
    ElementName=Button1,  
    Path=Content,  
    Mode=OneWay} "  
    Height="30" />
```

Обновление привязки

Значения перечисления UpdateSourceTrigger

- ▶ **PropertyChanged**—обновление происходит сразу после изменения значения свойства.
- ▶ **LostFocus**—обновление происходит после изменения значения и потери фокуса.
- ▶ **Explicit**—обновления происходят после вызова метода `BindingExpression.UpdateSource()`;
- ▶ **Default**—Для большинства свойств значение `PropertyChanged` для `TextBox.Text`-`LostFocus`

```
Text="{Binding ElementName=textBox1,  
    Path=Text,  
    Mode=TwoWay,  
    UpdateSourceTrigger=PropertyChanged}"  
/>
```

Привязка к объектам

- **Source**—ссылка на объект источник.

```
< TextBlock x:Name="nameTextBlock"  
Text="{Binding Source={StaticResource Student}, Path=FName}"  
Foreground="White"/>
```

- **DataContext**—указание источника для группы элементов управления.

```
<Grid DataContext="{StaticResource Student}" >  
    <TextBlock Text="Студент" />  
    <TextBlock Text="{Binding FName}" />  
    <TextBlock Text="{Binding Number}" />
```

вложенные элементы могут использовать объект `Binding` для привязки к конкретным свойствам этого контекста

RelativeSource

RelativeSource – позволяет создать привязку относительно элемента-источника, который связан какими-нибудь отношениями с элементом-приемником или на другой элемент вверх по дереву.

- `Self`: привязка осуществляется к свойству этого же элемента. То есть элемент-источник привязки в то же время является и приемником привязки.
- `FindAncestor`: привязка осуществляется к свойству элемента-контейнера.

```
<TextBox Text="{Binding  
    RelativeSource={RelativeSource Mode=Self},  
    Path=Background,  
    Mode=TwoWay,  
    UpdateSourceTrigger=PropertyChanged}" />
```

INotifyPropertyChanged

Для реализации механизма привязки, надо реализовать интерфейс

```
class Student : INotifyPropertyChanged
```

```
{
```

```
    private string name;
```

```
    public string Name
```

```
    {
```

```
        get { return name; }
```

```
        set
```

```
        {
```

```
            name = value;
```

```
            OnPropertyChanged("Name");
```

```
        }
```

```
    }
```

```
    public event PropertyChangedEventHandler PropertyChanged;
```

```
    public void OnPropertyChanged([CallerMemberName]string prop = "")
```

```
    {
```

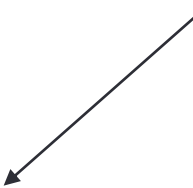
```
        if (PropertyChanged != null)
```

```
            PropertyChanged(this, new PropertyChangedEventArgs(prop));
```

```
    }
```

```
}
```

Когда объект класса изменяет значение свойства, то он через событие PropertyChanged извещает систему об изменении свойства. А система обновляет все привязанные объекты.



Провайдеры данных. ObjectDataProvider

- ▶ ПОЗВОЛЯЮТ СВЯЗЫВАТЬ ИСТОЧНИКИ ДАННЫХ И ЭЛЕМЕНТЫ ИНТЕРФЕЙСА.
- ▶ ObjectDataProvider (для работы с объектами)
- ▶ XmlDataProvider (для работы с xml-файлами)

Пример

Классы, который представляют модель данных, и классы, которые будут поставлять данные

```
public class Team
{
    string _name;

    public Team(string name)
    {
        _name = name;
    }

    public string Name { get { return _name; } }

    public override string ToString()
    {
        return _name.ToString();
    }
}

public class TeamList : ObservableCollection<Team>
{
    public TeamList() : base()
    {
    }
}
```

```
public class Division
{
    string _name;
    TeamList _teams;

    public Division(string name)
    {
        _name = name;
        _teams = new TeamList();
    }

    public string Name { get { return _name; } }

    public override string ToString()
    {
        return _name.ToString();
    }

    public TeamList Teams { get { return _teams; } }
}

public class DivisionList : ObservableCollection<Division>
{
    public DivisionList() : base()
    {
    }
}
```

```
{  
    League l;  
    Division d;  
  
    Add(l = new League("Лесхоз"));  
    l.Divisions.Add((d = new Division("Садовопарковое")));  
    d.Teams.Add(new Team("1"));  
    d.Teams.Add(new Team("2"));  
    d.Teams.Add(new Team("3"));  
    d.Teams.Add(new Team("4"));  
    d.Teams.Add(new Team("5"));  
    l.Divisions.Add((d = new Division("Лесозаготовительное")));  
    d.Teams.Add(new Team("6"));  
    d.Teams.Add(new Team("7"));  
    d.Teams.Add(new Team("8"));  
    d.Teams.Add(new Team("9"));  
    d.Teams.Add(new Team("10"));  
    l.Divisions.Add((d = new Division("Еще какой -то")));  
    d.Teams.Add(new Team("11"));  
    d.Teams.Add(new Team("12"));  
    d.Teams.Add(new Team("13"));  
    d.Teams.Add(new Team("14"));  
    Add(l = new League("ИДиП"));  
    l.Divisions.Add((d = new Division("Редакторы")));  
    d.Teams.Add(new Team("Группа 1.1"));  
    d.Teams.Add(new Team("Группа 1.2"));  
    d.Teams.Add(new Team("Группа 1.3"));  
    d.Teams.Add(new Team("Группа 1.4"));  
    d.Teams.Add(new Team("Группа 1.5"));  
}
```

```

Title="As"
Background="Silver">
<Window.Resources>
    <ObjectDataProvider x:Key="MyList" ObjectType="{x:Type src:LeagueList}" />
    <Style TargetType="StackPanel">
        <Setter Property="DockPanel.Dock" Value="Left"/>
        <Setter Property="Margin" Value="10,0,10,0"/>
    </Style>
    <Style TargetType="ListBox">
        <Setter Property="Height" Value="100"/>
    </Style>
    <Style TargetType="Label">
        <Setter Property="FontSize" Value="12"/>
    </Style>
</Window.Resources>

<DockPanel DataContext="{Binding Source={StaticResource MyList}}">
    <StackPanel>
        <Label>Facultet</Label>
        <ListBox ItemsSource="{Binding}"
            IsSynchronizedWithCurrentItem="true"/>
    </StackPanel>

```

Document Outline

- [-] Window
 - [+] Resources
 - [-] DockPanel
 - [-] StackPanel
 - Label
 - ListBox
 - [-] StackPanel
 - Label
 - ListBox
 - [-] StackPanel
 - Label
 - ListBox

```
<DockPanel DataContext="{Binding Source={StaticResource MyList}}">
  <StackPanel>
    <Label>Facultet</Label>
    <ListBox ItemsSource="{Binding}"
      IsSynchronizedWithCurrentItem="true"/>
  </StackPanel>

  <StackPanel>
    <Label Content="Spec"/>
    <ListBox ItemsSource="{Binding Path=Divisions}"
      IsSynchronizedWithCurrentItem="true"/>
  </StackPanel>

  <StackPanel>
    <Label Content="Gruops"/>
    <ListBox ItemsSource="{Binding Path=Divisions/Teams}"/>
  </StackPanel>
</DockPanel>
</Window>
```

Binding.exe

Документы

► Фиксированные документы (fixed documents)

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

► Потокотые документы (flow documents)

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

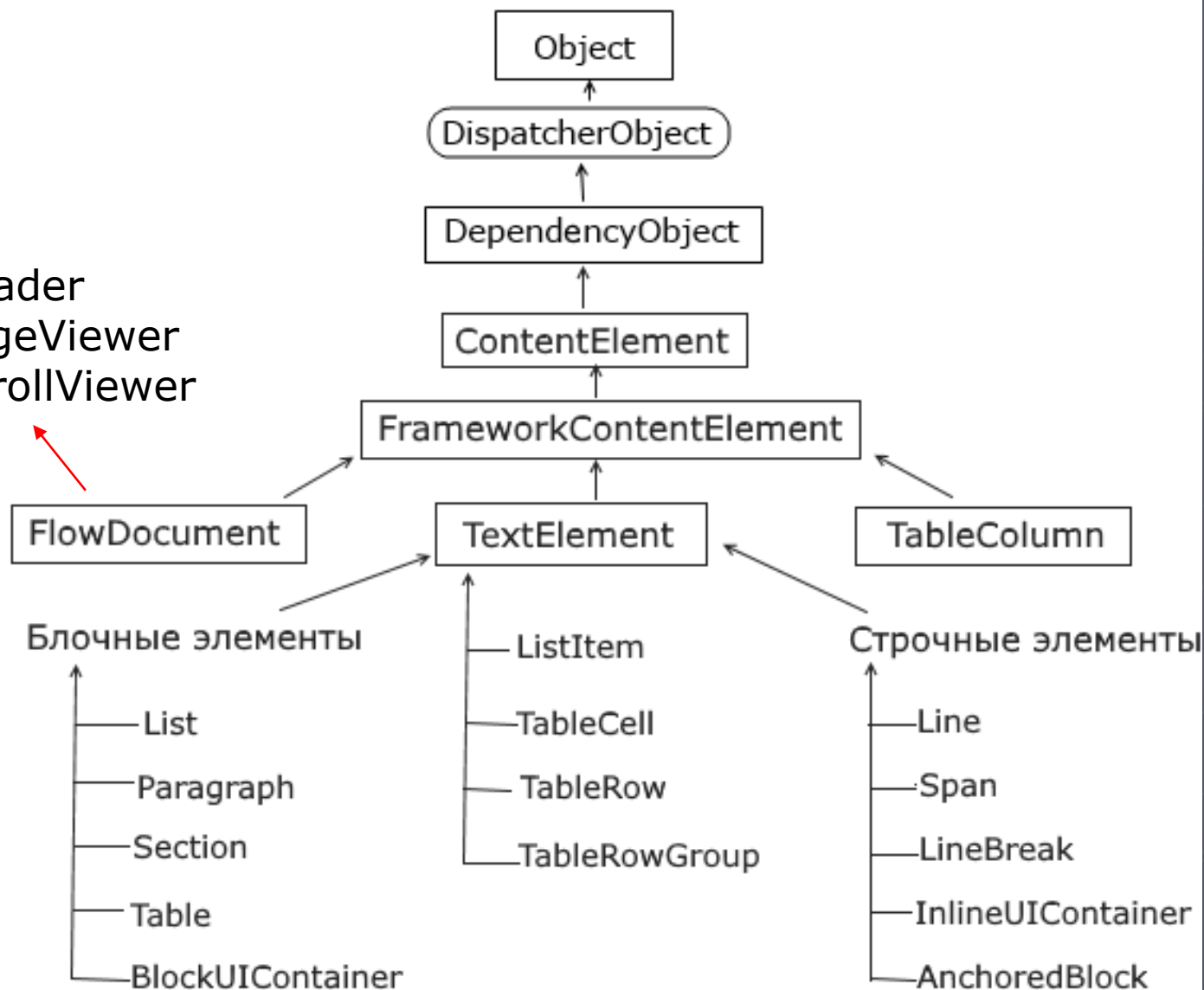
Потоковые документы

Контейнеры:

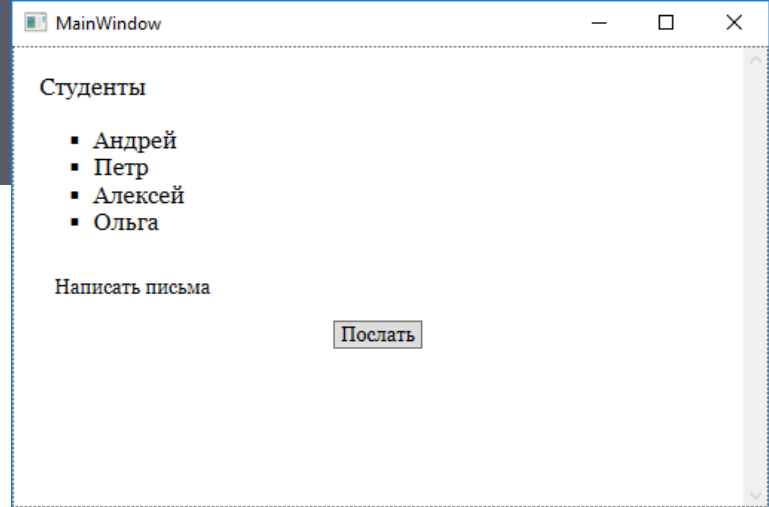
FlowDocumentReader

FlowDocumentPageViewer

FlowDocumentScrollViewer



```
<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>
```



<FlowDocumentReader>

```
<FlowDocument ColumnWidth="150" ColumnGap="10">
```

```
<Paragraph TextAlignment="Left" FontSize="15">
```

[illegible]

</Paragraph>

</FlowDocument>

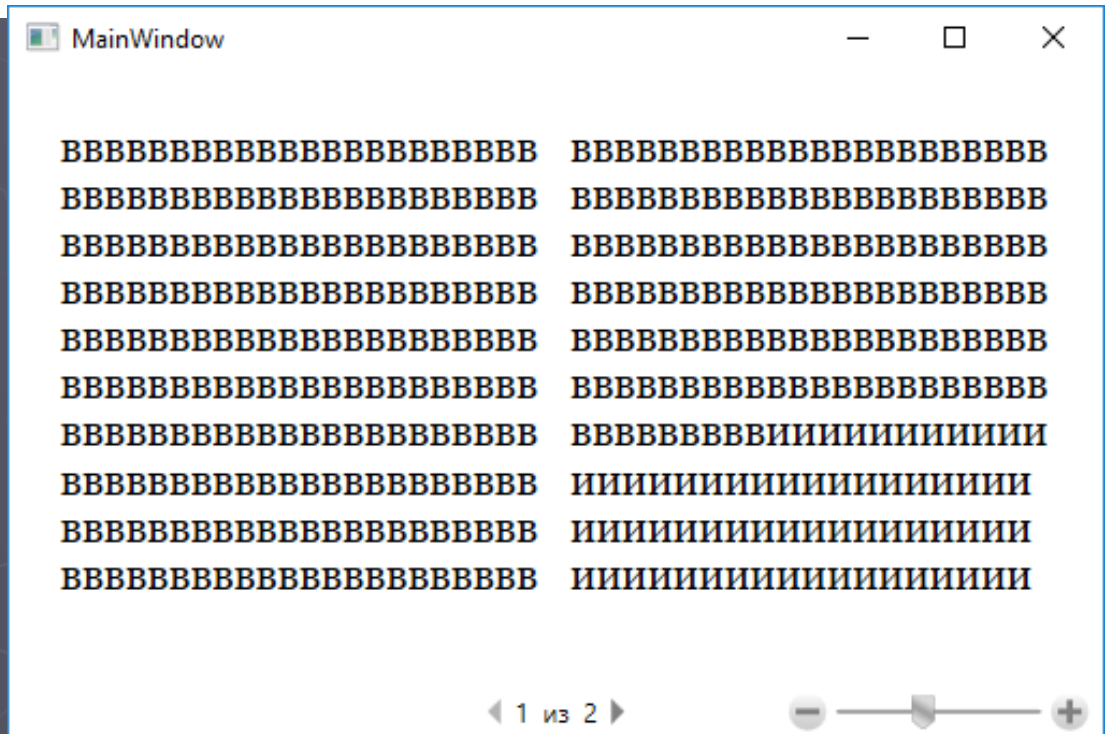
</FlowDocumentReader>



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

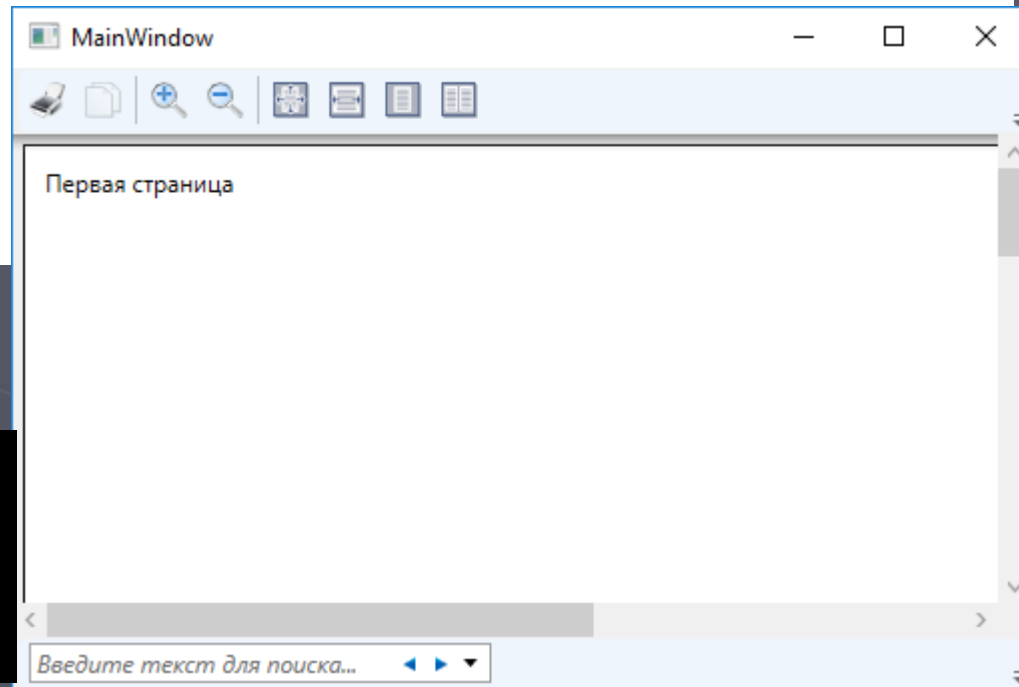
[illegible]

```
</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