# Лекция 3

## Введение в графические приложения на платформе WPF

# Разработка графических приложений на C#

Раньше - Windows Forms

- Основано на MFC (1992 год)
- Дизайн жестко привязан к коду
- Нет поддержки аппаратного ускорения

# Разработка графических приложений на C#

Сейчас - Windows Presentation Foundation

- Создано в этом тысячелетии
- Дизайнер и программист могут работать независимо друг от друга
- Интерфейс отрисовывается через DirectX

# WPF состоит из двух частей

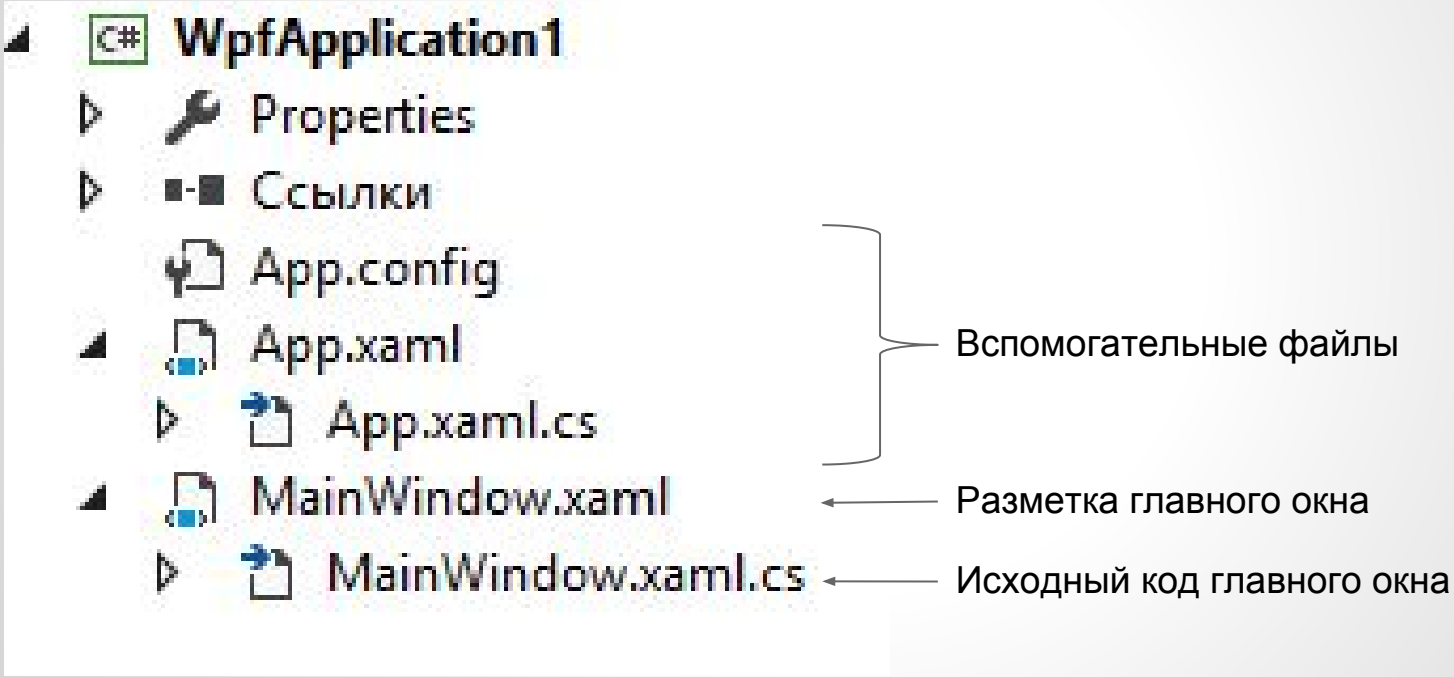
## Дизайн

XAML (зэмл)
html подобный язык разметки, отвечающий за то как выглядит приложение

## Код

C# код реализующий логику работы приложения

# Состав проекта WPF

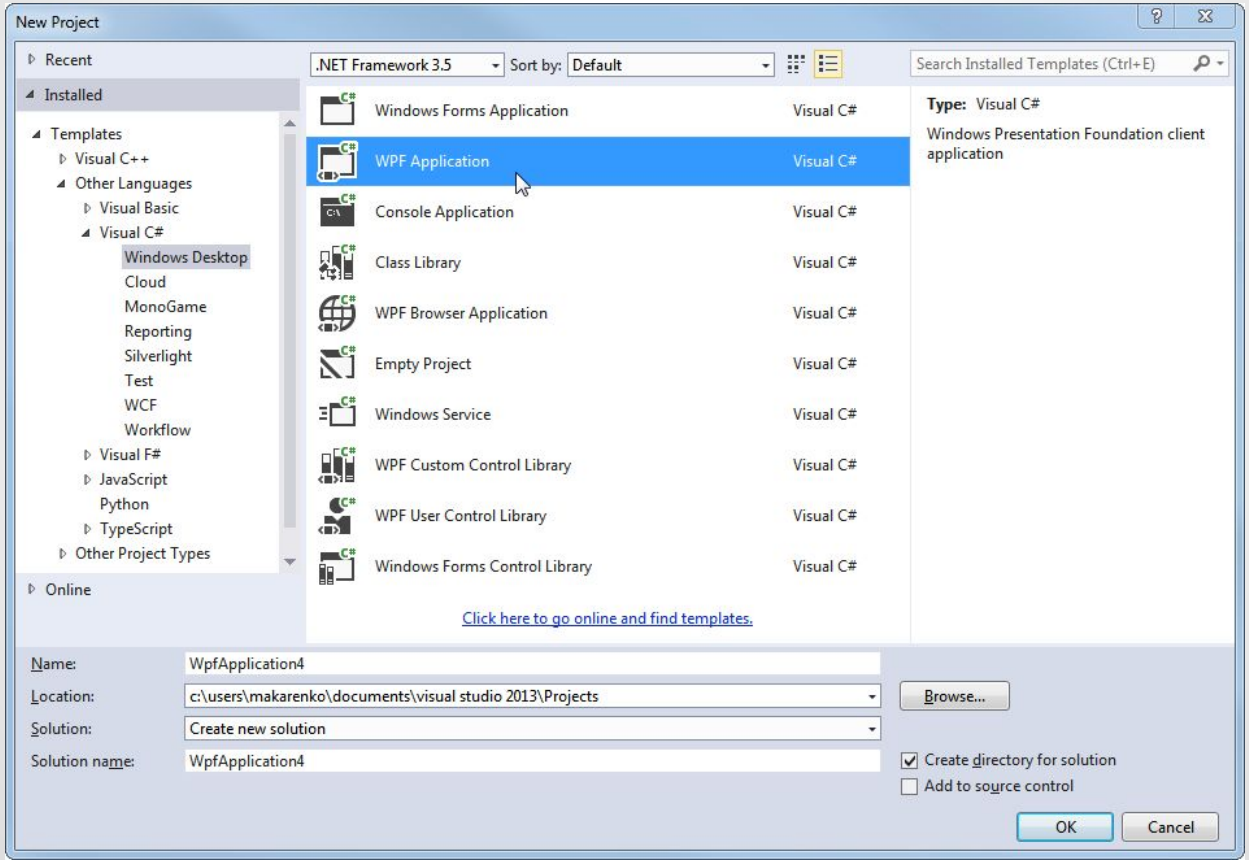


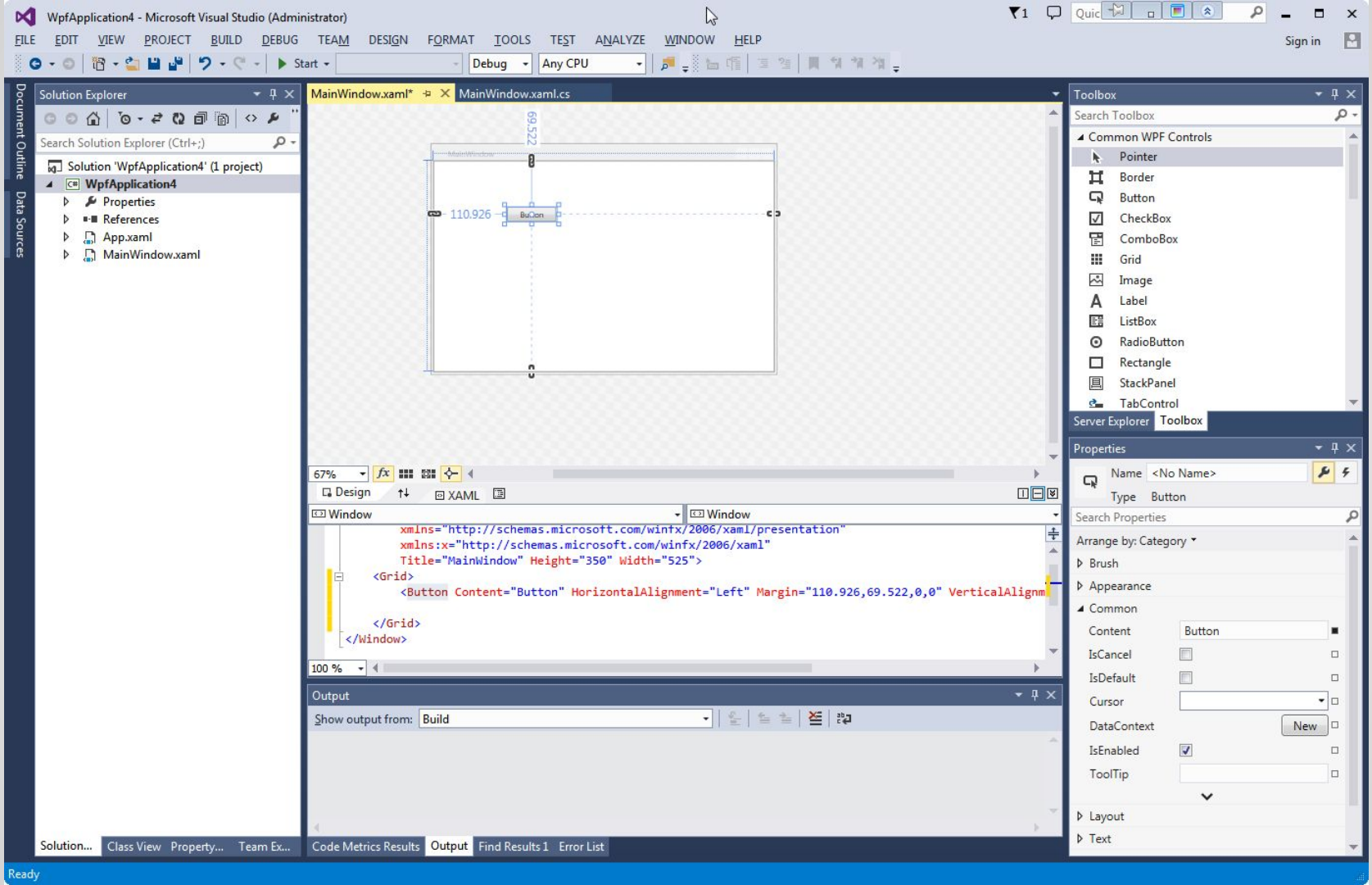
# Подходы к разработке WPF приложений

- Code behind - вся логика приложения описывается в связанном классе

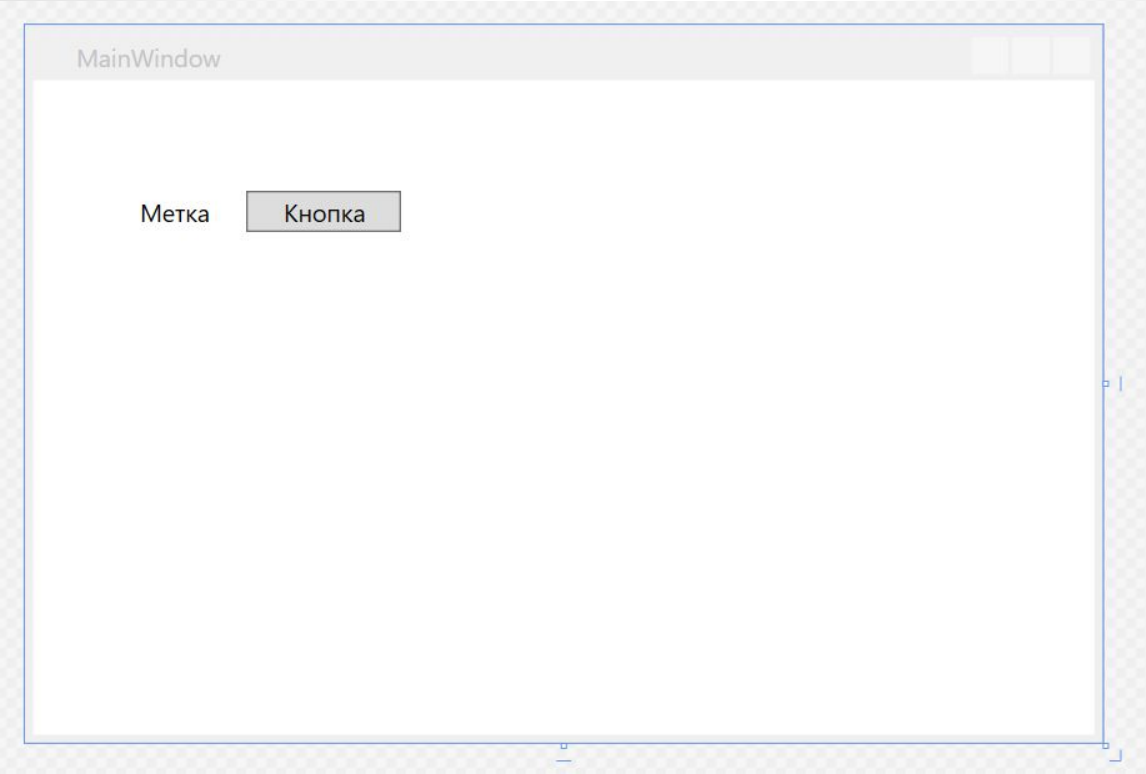
- MVVM - класс окна не содержит кода. Приложение разделяется на Model (бизнес-логика) и View (окно).

# Создание проекта WPF




WpfApplication4 - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM DESIGN FORMAT TOOLS TEST ANALYZE WINDOW HELP
Sign in
Start
Debug
Any CPU
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WpfApplication4' (1 project)
WpfApplication4
Properties
References
App.xaml
MainWindow.xaml
MainWindow.xaml*
MainWindow.xaml.cs
69.522
110.926
BuCon
67%
Design
XAML
Window
Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="350" Width="525">
<Grid>
<Button Content="Button" HorizontalAlignment="Left" Margin="110.926,69.522,0,0" VerticalAlignm
</Grid>
</Window>
100 %
Output
Show output from: Build
Toolbox
Search Toolbox
Common WPF Controls
Pointer
Border
Button
CheckBox
ComboBox
Grid
Image
Label
ListBox
RadioButton
Rectangle
StackPanel
TabControl
Server Explorer
Toolbox
Properties
Name <No Name>
Type Button
Search Properties
Arrange by: Category
Brush
Appearance
Common
Content Button
IsCancel
IsDefault
Cursor
DataContext New
IsEnabled
ToolTip
Layout
Text
Solution...
Class View
Property...
Team Ex...
Code Metrics Results
Output
Find Results 1
Error List
Ready

# XAML



# XAML

```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApplication2"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button x:Name="button" Content="Кнопка"
                HorizontalAlignment="Left" Margin="104,54,0,0"
                VerticalAlignment="Top" Width="75"/>
        <Label x:Name="label" Content="Метка"
            HorizontalAlignment="Left" Margin="47,51,0,0"
            VerticalAlignment="Top"/>

    </Grid>
</Window>
```

# Из чего состоит XAML

- Элементы - Elements / tags
- Атрибуты - Attributes / properties
- Содержимое - Content

```
<Grid>
      <Button x:Name="button" Content="Кнопка"
              HorizontalAlignment="Left"
               Margin="104,54,0,0"/>
</Grid>
```

# Элементы XAML

- Элементы отвечающие за расположение дочерних элементов - **Layouts**
- Графические элементы интерфейса
- Служебные элементы используемые для настройки первых двух типов

# Атрибуты XAML

Относятся к элементам, и используются для задания их свойств и поведения

```
<Button x:Name="button" Content="Кнопка"
                    HorizontalAlignment="Left"
                     Margin="104,54,0,0"/>
```

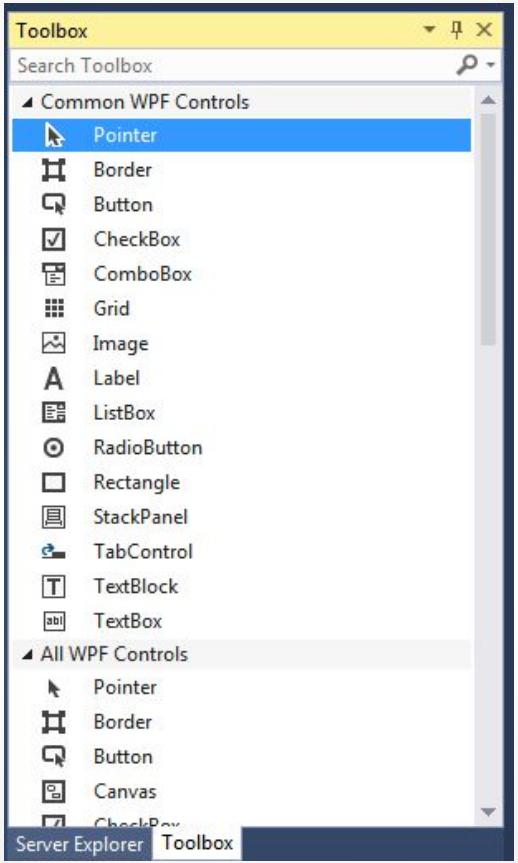
# Управление расположением элементов внутри окна

- **Grid** - представляет собой область разбитую на столбцы и строки. Элементы располагаются в табличном виде

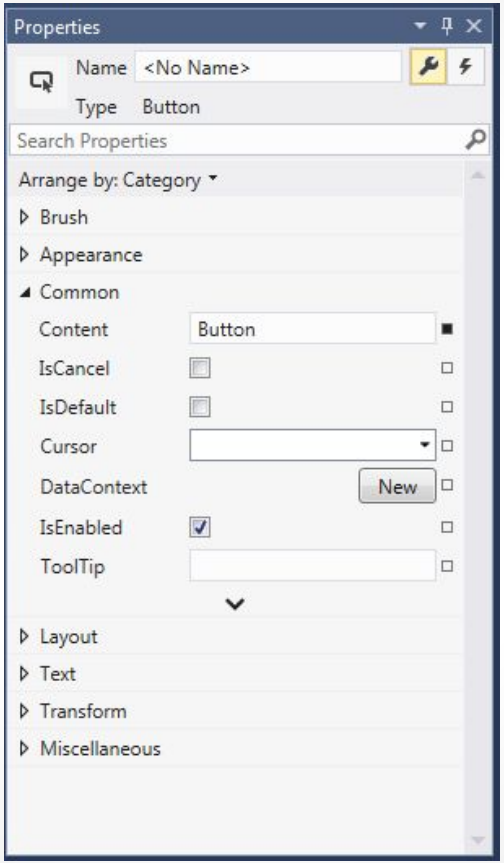
- **StackPanel** - элементы располагаются в одну линию, вертикально или горизонтально

- **WrapPanel** - аналогично StackPanel, но при заполнении строки происходит перенос на следующую строку

# Панель элементов

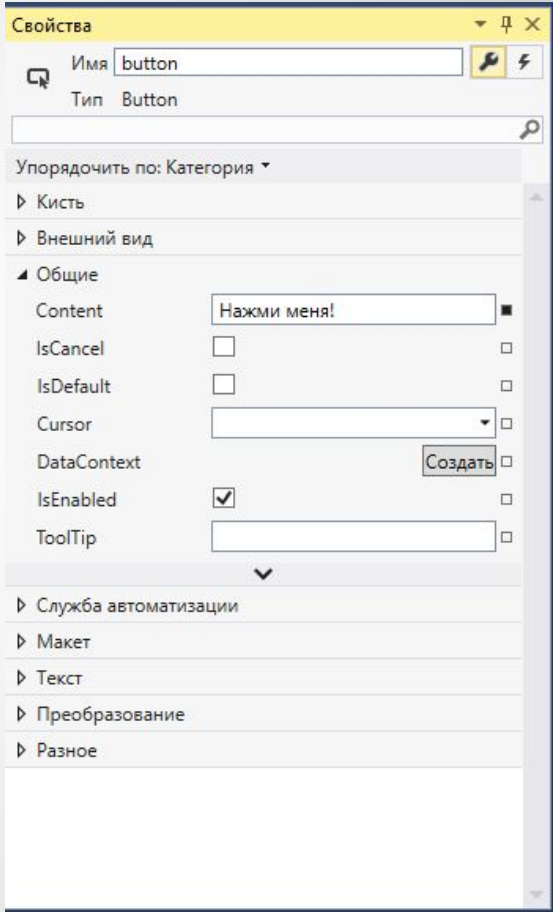


# Панель свойств

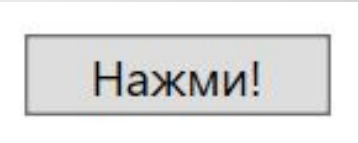


# Панель свойств и XAML

```
<Button x:Name="button" Content="Нажми
меня!"
                HorizontalAlignment="Left"
                    Margin="104,54,0,0"/>
```



# Визуальные элементы. Кнопка



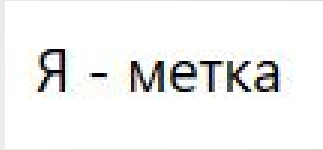

Позволяет пользователю совершить действие нажатием на элемент

XAML: **<Button>**

Основные свойства:

- Content - текст кнопки
- IsEnabled - “включена” ли кнопка

# Визуальные элементы. Метка




Используется для вывода текста небольшого объема и подписи к другим элементам

XAML: **<Label>**

Основные свойства:

- **Content -** текст метки

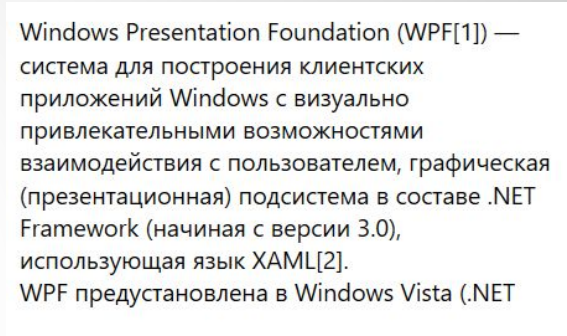
# Визуальные элементы. Текст

Используется для вывода многострочного текста

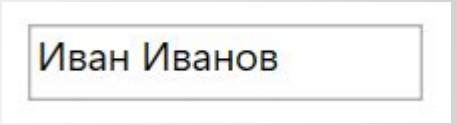
XAML: **<TextBlock>**

Основные свойства:

- **Text -** текст для отображения



# Визуальные элементы. Текстовое поле




Используется для предоставления пользователю возможности ввести текст

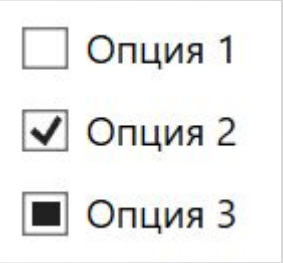
XAML: **<TextBox>**

Основные свойства:

- Text - введенный текст
- AcceptsReturn - разрешает вводить многострочный текст

# Визуальные элементы. Переключатель

Используется для предоставления пользователю возможности включения и выключения некоторых опций

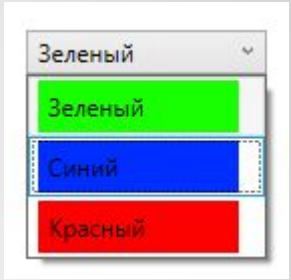



XAML: **<CheckBox>**

Основные свойства:

- Content - текст метки
- IsChecked - установлена ли галочка
- IsThreeState - возможно ли “третье” состояние

# Визуальные элементы. Поле со списком




Используется для предоставления пользователю возможности выбора варианта из списка

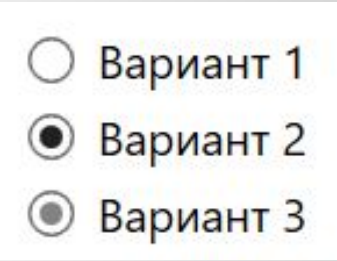
XAML: **<ComboBox>**

Основные свойства:

- Items - элементы списка
- SelectedIndex - номер выбранного элемента
- SelectedItem - выбранный элемент

# Визуальные элементы. Переключатель

Используется для предоставления пользователю возможности выбора варианта



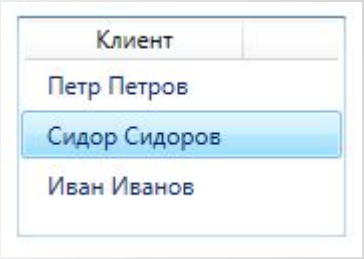

XAML: **<RadioButton>**

Основные свойства:

- Content - текст метки
- IsChecked - установлена ли галочка
- IsThreeState - возможно ли “третье” состояние
- GroupName - имя группы переключателей

# Визуальные элементы. Список

Используется для отображения набора данных в виде списка




XAML: **<ListView>**

Основные свойства:

- Items - элементы списка
- SelectedIndex - номер выбранного элемента
- View - отображение списка, настройка колонок данных

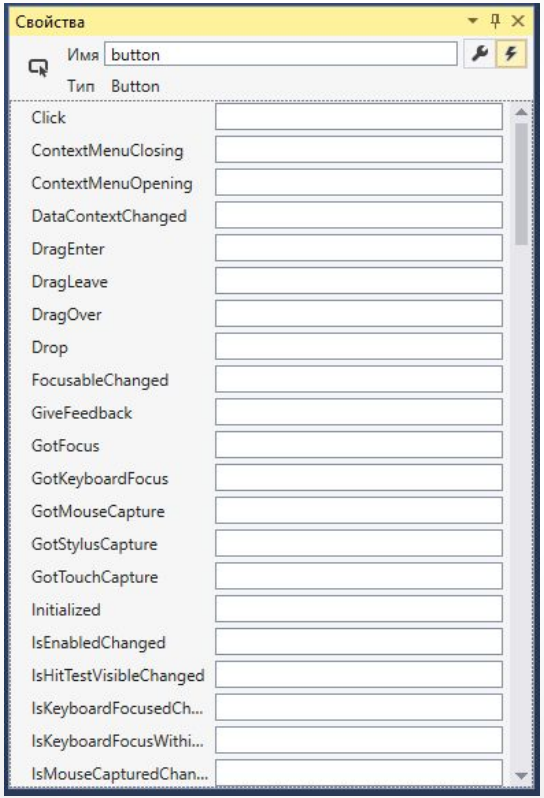
# События

Событие - это сообщение посылаемое элементом для уведомления кода о том, что произошло что-то важное

Примеры:

- Нажата кнопка
- Изменен текст в поле ввода
- Выбран элемент списка
- Переместился курсор мыши

# Панель событий



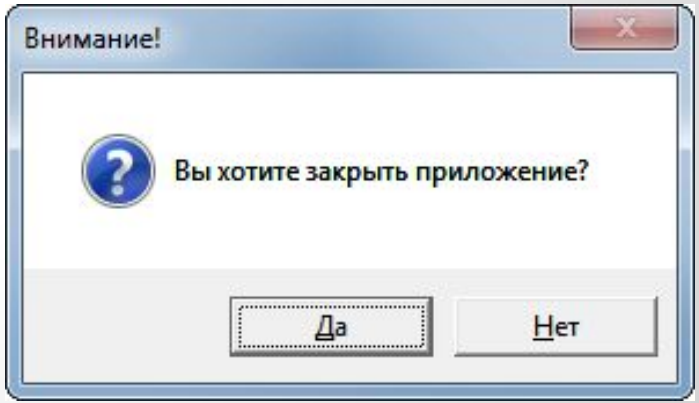
# Обработчик события

```
private void button_Click(object sender, RoutedEventArgs e)
{
    // тут пишем код, который нужно выполнить по нажатию кнопки
}
```

# Информационные окна

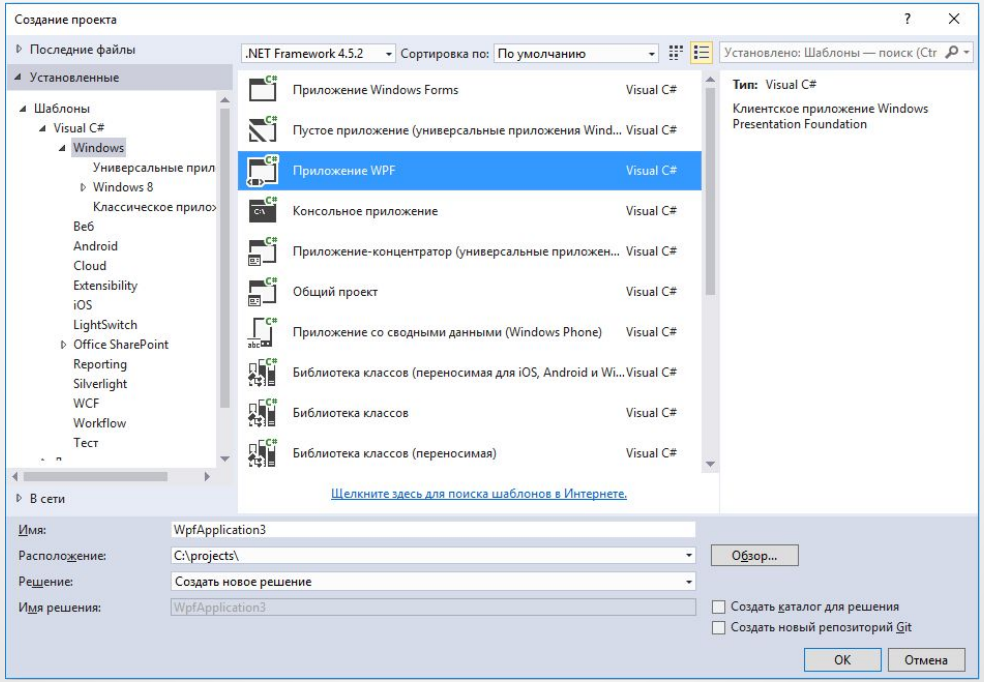
```
private void button_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Hello world!");
}
```

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
     var result = MessageBox.Show("Вы хотите закрыть приложение?",
                                  "Внимание!",
                                  MessageBoxButton.YesNo,
                                  MessageBoxImage.Question);
        if (result == MessageBoxResult.No)
        {
           e.Cancel = true;
     }
}
```



# Пример создания приложения WPF

Создаем проект “Приложение WPF”



# Добавляем элементы в окно

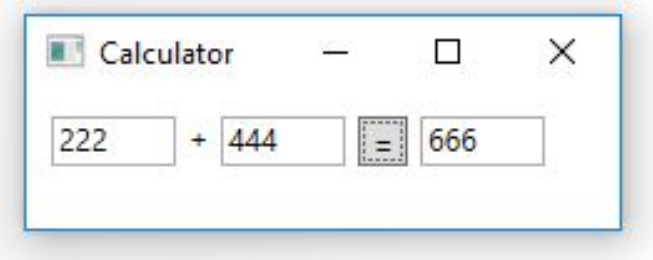
```
<WrapPanel Orientation="Horizontal" Margin="10, 10, 10, 10">
  <TextBox x:Name="tbOperandA" Text="0" Width="50"/>
  <Label x:Name="lblPlus" Content="+" Padding="5, 0, 5, 0"/>
  <TextBox x:Name="tbOperandB" Text="0" Width="50"/>
  <Button x:Name="btnCalc" Content="=" Margin="5, 0, 5, 0"
                                       Width="20" />
  <TextBox x:Name="tbResult" Text="0" Width="50"
                                IsReadOnly="True"/>
</WrapPanel>
```

# Добавляем обработчик события

```
<Button     x:Name="btnCalc" Content="="
            Margin="5, 0, 5, 0" Width="20"
            Click="btnCalc_Click"
/>
```

```
  private void btnCalc_Click(object sender, RoutedEventArgs e)
  {
      double opA = double.Parse(tbOperandA.Text);
      double opB = double.Parse(tbOperandB.Text);
      double result = opA + opB;
      tbResult.Text = result.ToString();
  }
```

# Запускаем программу



# Добавим другие арифметические операции

```
<ComboBox x:Name="cbOperation" Padding="5, 0, 5, 0">
        <ComboBoxItem Content="+" IsSelected="True"/>
        <ComboBoxItem Content="-" />
        <ComboBoxItem Content="*" />
        <ComboBoxItem Content="/" />
</ComboBox>
```

# Доработаем обработчик события

```
private void btnCalc_Click(object sender, RoutedEventArgs e)
{
      double opA = double.Parse(tbOperandA.Text);
      double opB = double.Parse(tbOperandB.Text);
      double result = 0;
      switch (cbOperation.SelectedIndex)
      {
        case 0:
            result = opA + opB;
            break;
        case 1:
            result = opA - opB;
            break;
        case 2:
            result = opA * opB;
            break;
        case 3:
            result = opA / opB;
            break;
      }
      tbResult.Text = result.ToString();
}
```

# Проверка деления на ноль

```
case 3:
    if (opB == 0)
    {
       MessageBox.Show("На ноль делить нельзя!",
           "Ошибка!",
           MessageBoxButton.OK,
           MessageBoxImage.Error);
       break;
    }
    result = opA / opB;
    break;
```