Свойства

- Свойство – это член, предоставляющий гибкий механизм для чтения, записи и вычисления значения частного (private) поля.
- Свойства очень удобны в контроле значений определенной переменной, и используются повсеместно.

Пример:

```
private double seconds;
public double Hours
{
  get { return seconds / 3600; }
  set { seconds = value * 3600; }
}
```

Интерфейсы

- Интерфейсы очень похожи на классы, но они служат всего лишь как шаблоны.
- Можно сказать, что интерфейсы похожи на абстрактные классы, но в отличие от них, в интерфейсах нельзя реализовывать логику, только объявлять свойства и методы.
- Интерфейсы нужны там, где необходимо несколько классов сгруппировать как единое целое.

Пример:

```
public interface MyInterface
{
   public float MyFloat {get; set;} - свойство
   public void MyMethod(); - метод
}
```

События

- События (events) позволяют классу или объекту уведомлять другие классы или объекты о возникновении каких-либо ситуаций. Класс, отправляющий событие, называется издателем, а классы, принимающие событие, называются подписчиками.

- Но прежде, чем изучать события, необходимо разобрать, что такое делегаты (delegats).

Делегаты

- Делегат это объект, указывающий на функцию. Вызывая делегат, мы вызываем функцию, на которую он указывает.

Пример:

```
delegate возвращаемый_тип имя (список_параметров);
delegate string MyDelegate(int i);
…
MyDelegate _myD = new MyDelegate(MyMethod);
string result = _myD(5);
…
public string MyMethod(int a) { return a.ToString(); }
```

Объявление событий

- События основаны на делегатах, и прежде, чем объявить событие, необходимо объявить делегат:

```
public event MyDelegate MyEvent;
```

- Теперь необходимо в нужный нам момент вызывать событие, чтобы все классы, которые подписаны на него, получили необходимое сообщение:

```
if(MyEvent != null)
   MyEvent(список_параметров);
```

Подписка на события

- Для получение сообщения от события, необходимо на него подписаться. Это можно сделать с помощью выражения +=

Пример:

MyEvent += Метод_с_параметрами_как_у_делегата;

- Если необходимо отписаться от события, то используем -=

Пример:

MyEvent -= Метод_с_параметрами_как_у_делегата;

Технология WPF

Что такое WPF и с чем его едят?

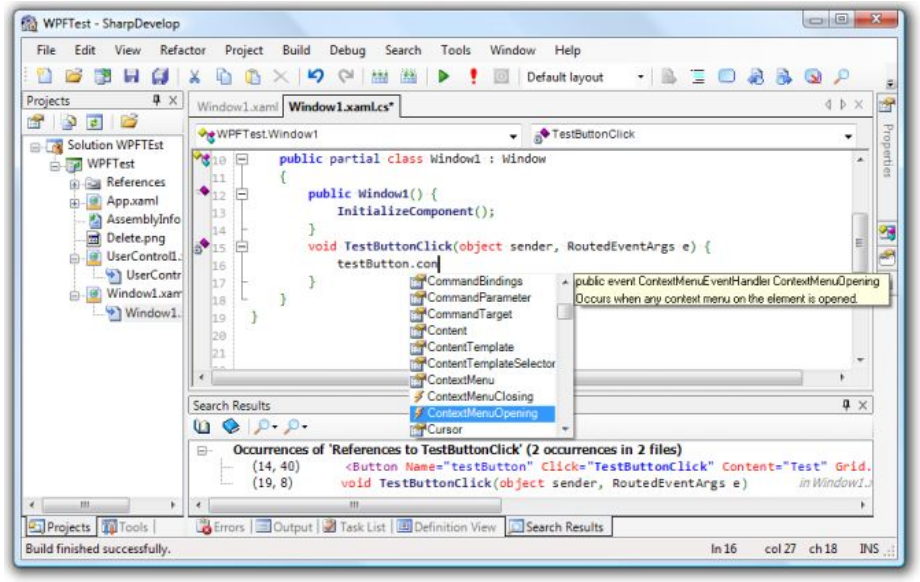
- WPF (Windows Presentation Foundation) – технология корпорации Microsoft, которая представляет собой систему для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, использующая язык XAML.
- На WPF реализованы системы Windows, начиная с Windows Vista.
- На этой системе можно реализовать как автономные приложения, так и запускаемые в браузере.

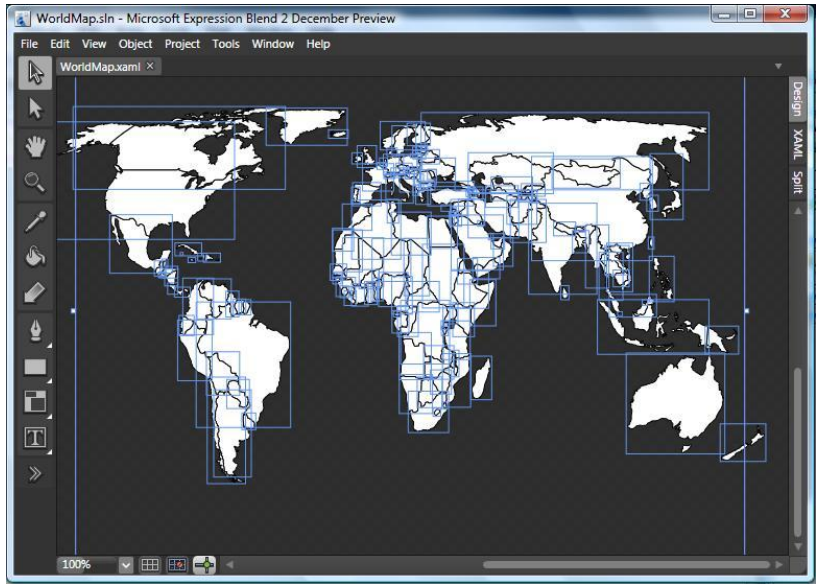
Язык XAML

- XAML представляет собой язык декларативного описания интерфейса, основанный на XML.

```
<ListBox Name="CategoryListBox"
         ScrollViewer.HorizontalScrollBarVisibility="Disabled"
         ItemsSource="{Binding Path=RefValues,
                UpdateSourceTrigger=PropertyChanged}"
                SelectionMode="Multiple">
    <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
            <WrapPanel />
        </ItemsPanelTemplate>
    </ListBox.ItemsPanel>
    <ListBox.ItemTemplate>
     <DataTemplate >
        <StackPanel Orientation="Horizontal"
                    MinWidth="150" MaxWidth="150"
                    Margin="0,5, 0, 5" >
            <CheckBox
                Name="checkedListBoxItem"
                IsChecked="{Binding
                        RelativeSource={RelativeSource FindAncestor,
                        AncestorType={x:Type ListBoxItem} },
                        Path=IsSelected, Mode=TwoWay}" />
            <ContentPresenter
                Content="{Binding
                        RelativeSource={RelativeSource TemplatedParent},
                        Path=Content}"
                Margin="5,0, 0, 0" />
        </StackPanel>
    </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

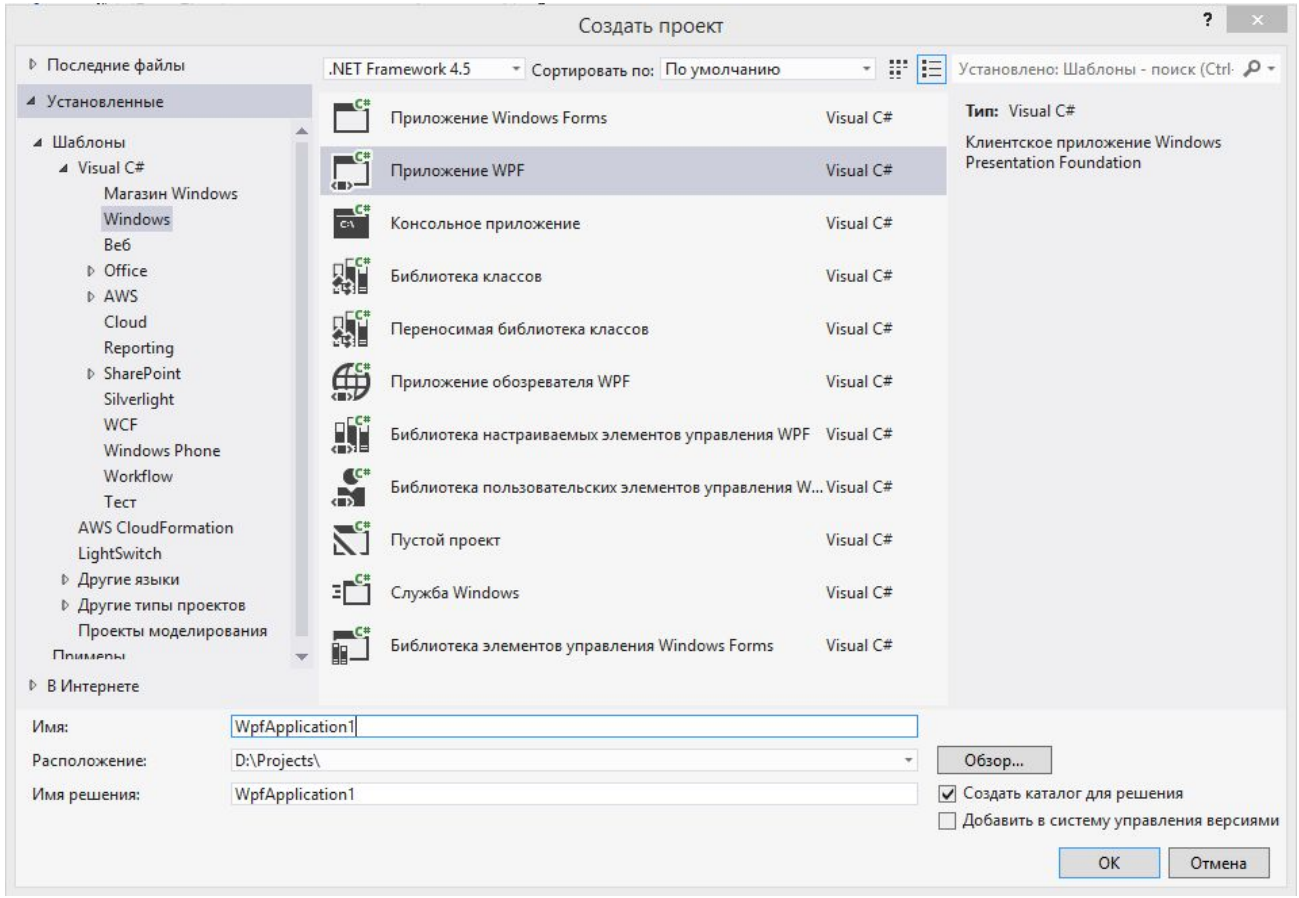
- Для работы с WPF требуется любой .NET-совместимый язык. В этот список входит и C#. Для полноценной работы может быть использована Visual Studio или Expression Blend.





Первая программа на WPF

- Создайте проект:



Первая программа на WPF

- После создания, у нас автоматически появится окно, и сгенерируется код для него:

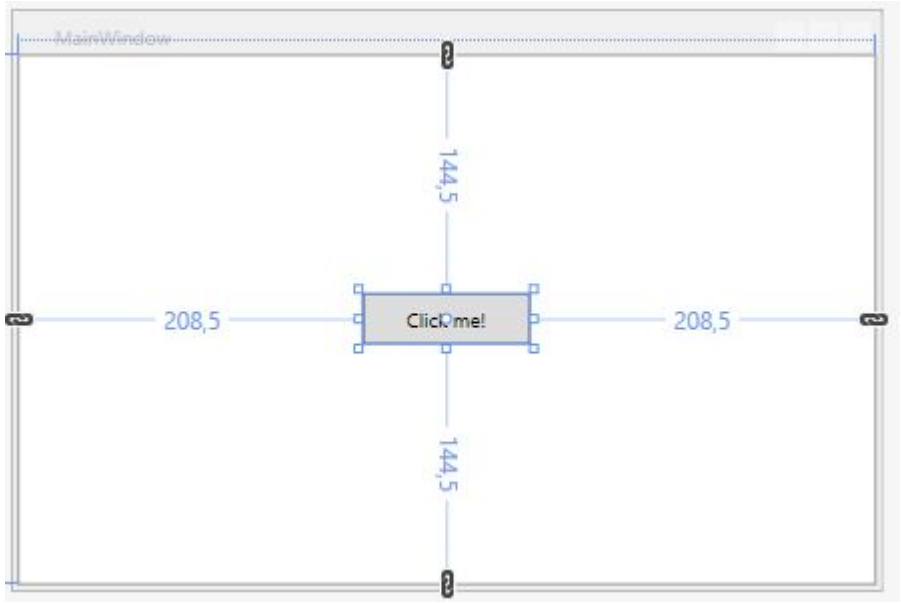
Первая программа на WPF

- Давайте добавить кнопку по середине этого окошка:

```
<Window x:Class="WpfApplication1.MainWindow"
        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 Width="100" Height="30" Content="Click me!"></Button>
    </Grid>
</Window>
```

Первая программа на WPF

- У вас должно получить что-то подобное:



Первая программа на WPF

- Добавим еще два контрола - TextBox и Label:

```
<Window x:Class="WpfApplication1.MainWindow"
        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>
        <Label Content="Text:" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-70,75,0,0"></Label>
        <TextBox Width="100" Height="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,100,0,0"></TextBox>
        <Button Width="100" Height="30" Content="Click me!"></Button>
    </Grid>
</Window>
```

Первая программа на WPF

- Попробуем очищать текст из TextBox при клике на кнопку - изменим Content у Button, добавим имя TextBox’у и добавим обработчик события Click:

```
<Window x:Class="WpfApplication1.MainWindow"
        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>
        <Label Content="Text:" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-70,75,0,0"></Label>
        <TextBox x:Name="MyTextBox" Width="100" Height="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,100,0,0"></TextBox>
        <Button Width="100" Height="30" Content="Clear!" Click="ButtonBase_OnClick"></Button>
    </Grid>
</Window>
```

Первая программа на WPF

- При добавлении обработчика событий, у вас сгенерируется метод:

```
1  using System;
2  using System.Windows;
3
4  namespace WpfApplication1
5  {
6      /// <summary> ...
9      public partial class MainWindow : Window
10     {
11         public MainWindow()
12         {
13             InitializeComponent();
14         }
15
16         private void ButtonBase_OnClick(object sender, RoutedEventArgs e)
17         {
18             throw new NotImplementedException();
19         }
20     }
21 }
22
```

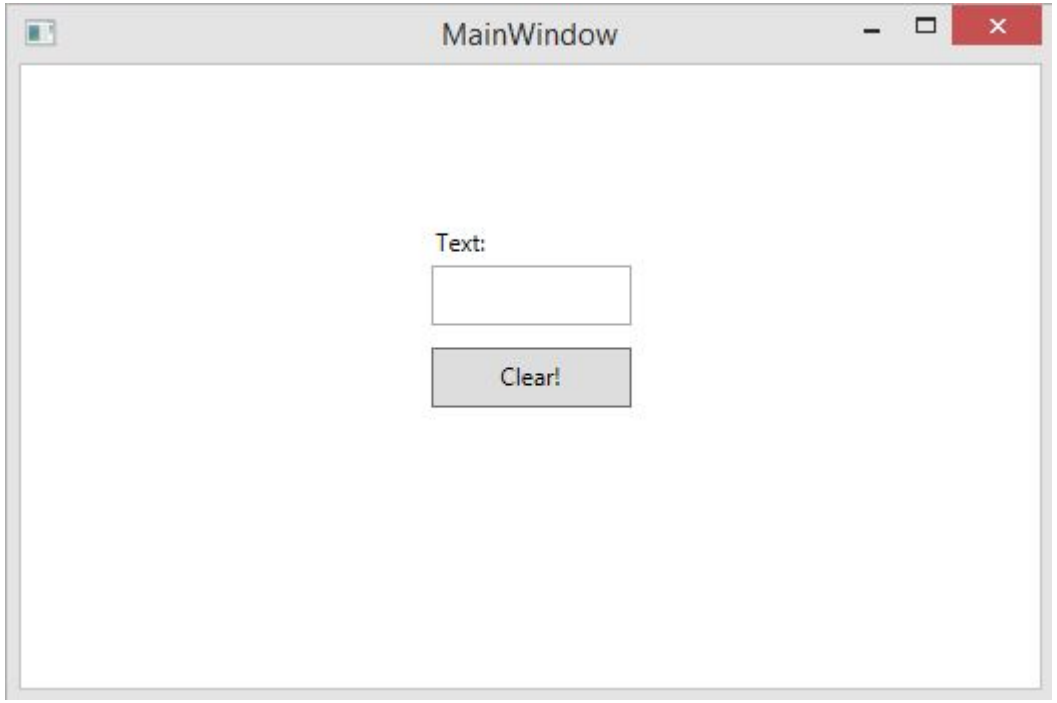
Первая программа на WPF

- Удаляем все из метода, и добавляем вот такую строчку кода:

```
1  using System;
2  using System.Windows;
3
4  namespace WpfApplication1
5  {
6      /// <summary> ...
9      public partial class MainWindow : Window
10     {
11         public MainWindow()
12         {
13             InitializeComponent();
14         }
15
16         private void ButtonBase_OnClick(object sender, RoutedEventArgs e)
17         {
18             MyTextBox.Text = ""; // Очистили текст
19         }
20     }
21 }
22
```

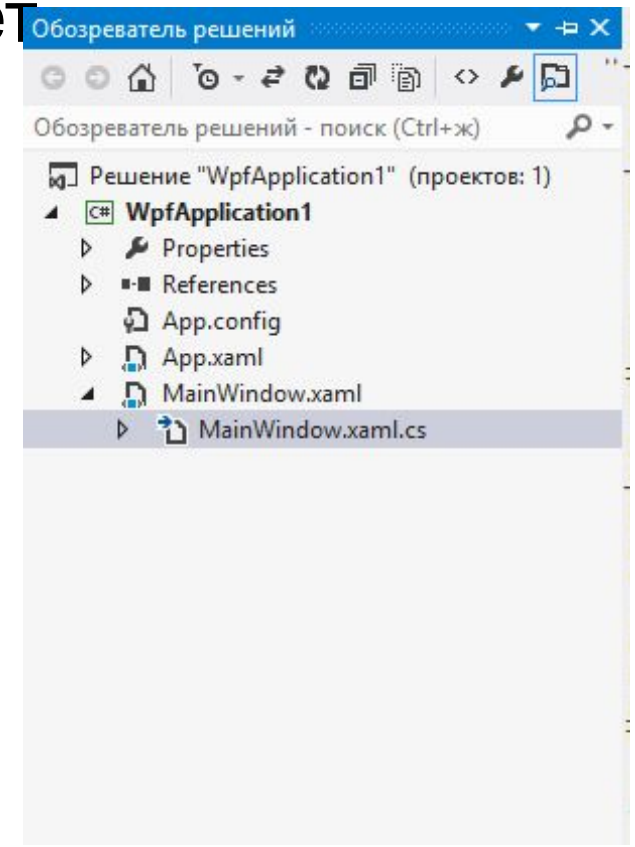
Первая программа на WPF

- Что должно получиться:



Проект на WPF. Структура.

- Каждое окно, или контрол пользователя имеет расширение .xaml. Внутри окна или контрола содержится не только файл разметки (xaml), но и файл с кодом логики на языке .NET (в нашем случае – C#). Этот файл имеет расширение .xaml.cs



Панель элементов в WPF