WPF Basics

abelski

Introduction

What is WPF?

- WPF stands for Windows Presentation Foundation.
- It is Microsoft's newest graphical subsystem for rendering user interfaces in Windows based applications.
- Based on DirectX, WPF provides us with hardware acceleration and special capabilities such as transparency and gradients.
- WPF allows us having a clear separation between the user interface and the business logic.

What is WPF?

- WPF includes classes that assist us in building cool user interfaces.
- Those classes include classes that describe user interface controls and classes that manage animations and resources, handle events and allow the creation of styles and templates.

What is XAML?

- XAML is part of WPF. It is a new markup language that allows us to define the user interface and their relationships with each other.
- We can create XAML (zammel) files using a simple text editor. We can alternatively use the Expression Blend tool.
- Using XAML isn't a MUST. We can hard code the entire user interface without writing a single XAML file.

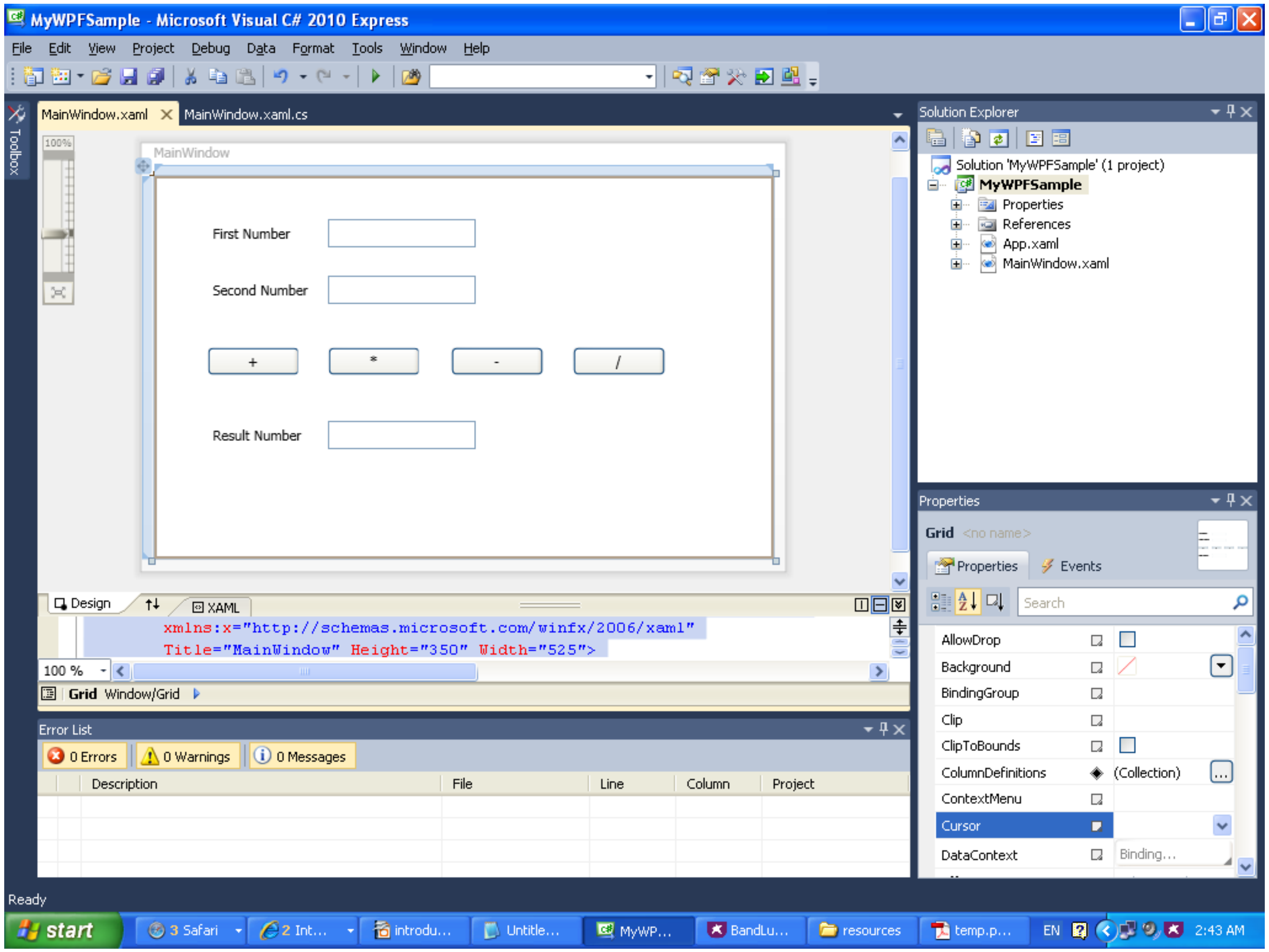
What is XAML?

```
<Window x:Class="MyWPFSample.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 Content="+" Height="23" HorizontalAlignment="Center"
            Margin="44,139,386,0" Name="btPlus" VerticalAlignment="Top"
            Width="75" Click="btPlus_Click" />
        <Button Content="-" Height="23" HorizontalAlignment="Center"
            Margin="241,139,187,0" Name="btMinus" VerticalAlignment="Top"
            Width="75" Click="btMinus_Click" />
        <Button Content="*" Height="23" HorizontalAlignment="Center"
            Margin="142,139,288,0" Name="btMultiply" VerticalAlignment="Top"
            Width="75" Click="btMultiply_Click" />
        <Button Content="/" Height="23" HorizontalAlignment="Center"
            Margin="340,139,88,0" Name="btDivide" VerticalAlignment="Top"
            Width="75" Click="btDivide_Click" />
```

What is XAML?

```
        <TextBox Height="23" HorizontalAlignment="Left" Margin="141,35,0,0"
            Name="firstNumber" VerticalAlignment="Top" Width="120" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="141,81,0,0"
            Name="secondNumber" VerticalAlignment="Top" Width="120" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="141,199,0,0"
            Name="resultNumber" VerticalAlignment="Top" Width="120" />
        <Label Content="First Number" Height="28" HorizontalAlignment="Left"
            Margin="43,35,0,0" Name="label1" VerticalAlignment="Top"
            Width="120" />
        <Label Content="Second Number" Height="28"
            HorizontalAlignment="Left" Margin="43,81,0,0" Name="label2"
            VerticalAlignment="Top" Width="120" />
        <Label Content="Result Number" Height="28"
            HorizontalAlignment="Left" Margin="43,199,0,0" Name="label3"
            VerticalAlignment="Top" Width="120" />
    </Grid>

</Window>
```

What is XAML?



```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="350" Width="525">
```

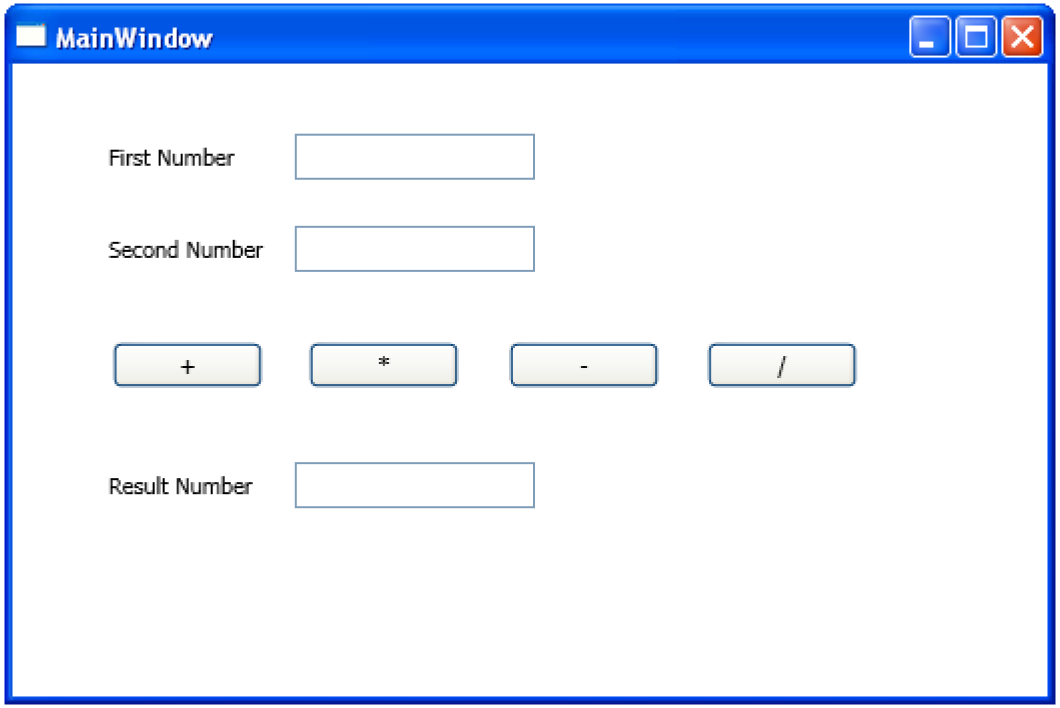
Events Handling

```
namespace MyWPFSample
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void btPlus_Click(object sender, RoutedEventArgs e)
        {
            resultNumber.Text = ""  +   (
                double.Parse(firstNumber.Text) +
                double.Parse(secondNumber.Text));
        }

        private void btMinus_Click(object sender, RoutedEventArgs e)
        {
            resultNumber.Text = ""  +   (
                double.Parse(firstNumber.Text) -
                double.Parse(secondNumber.Text));
        }
```

Events Handling

```
        private void btMultiply_Click(object sender, RoutedEventArgs e)
        {
            resultNumber.Text = "" + (
                double.Parse(firstNumber.Text) *
                double.Parse(secondNumber.Text));
        }
        private void btDivide_Click(object sender, RoutedEventArgs e)
        {
            resultNumber.Text = "" + (
                double.Parse(firstNumber.Text) /
                double.Parse(secondNumber.Text));
        }
    }
}
```

Events Handling



The Web Browser

- Using WPF we can develop web pages that will be able to run within a web browser.

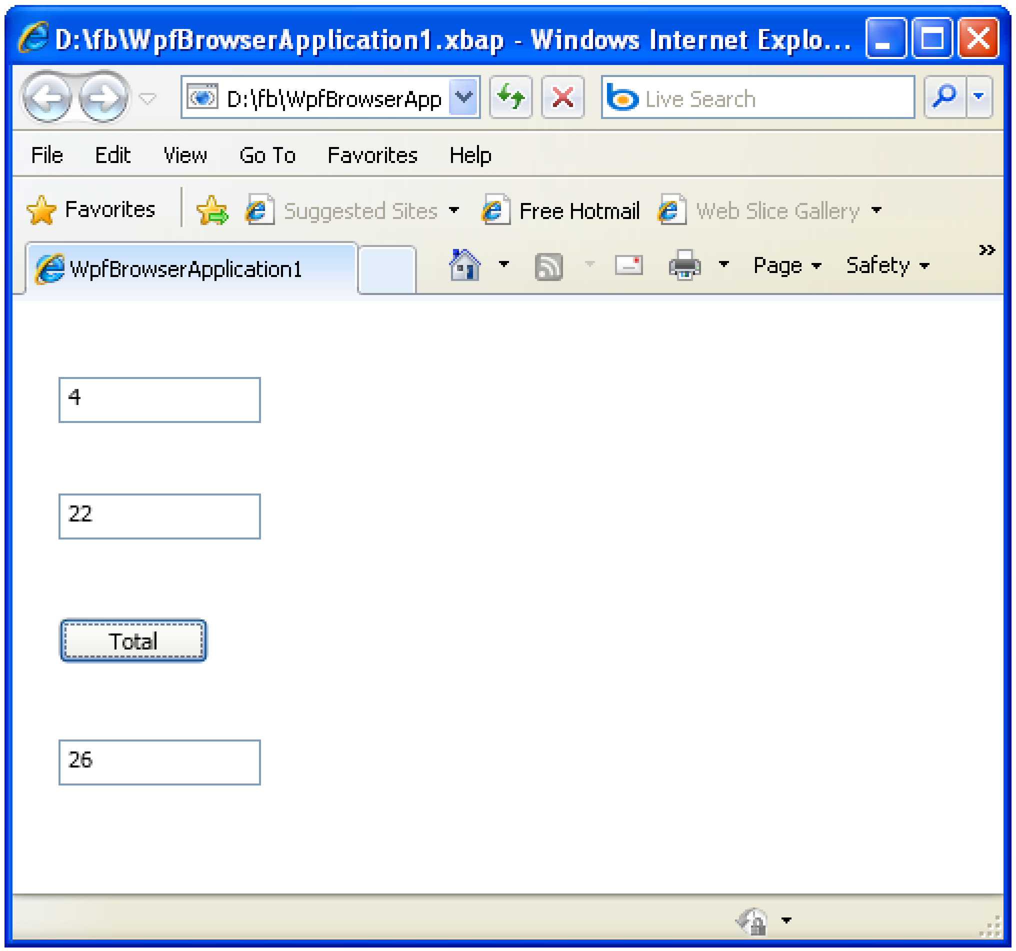
The Web Browser

```
namespace WpfBrowserApplication1
{
    public partial class Page1 : Page
    {
        public Page1()
        {
            InitializeComponent();
        }
        private void totalButton_Click(object sender, RoutedEventArgs e)
        {
            resultTextBox.Text = "" +
                (double.Parse(firstNumberTextBox.Text)
                +double.Parse(secondNumberTextBox.Text));
        }
    }
}
```

The Web Browser

```
<Page x:Class="WpfBrowserApplication1.Page1"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="300" d:DesignWidth="300"
      Title="Page1">
    <Grid>
        <Button Content="Total" Height="23" HorizontalAlignment="Left"
            Margin="23,158,0,0" Name="totalButton" VerticalAlignment="Top"
            Width="75" Click="totalButton_Click" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="23,38,0,0"
            Name="firstNumberTextBox" VerticalAlignment="Top" Width="101" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="23,96,0,0"
            Name="secondNumberTextBox" VerticalAlignment="Top" Width="101" />
        <TextBox Height="23" HorizontalAlignment="Left" Margin="23,219,0,0"
            Name="resultTextBox" VerticalAlignment="Top" Width="101" />
    </Grid>
</Page>
```

The Web Browser



What is Sliverlight?

- Silverlight is a restricted version of WPF that targets the web browser environment and aims at providing a rich user interface.

Silverlight Supported Web Browsers

- The Silverlight web browsers plug-in is currently available for the following web browsers: Firefox, Internet Explorer & Safari. The Safari version is available for windows platform only.

WPF Projects Types

- WPF allows us to develop three types of projects: standalone, XAML browser application and loose XAML pages.

WPF Standalone Application

- The WPF standalone application runs locally on our computer just as any other standalone application.
- The WPF standalone application has a full access to the computer resources. It can read and write files, modify the system registry and do nearly everything a stand alone application can.

XAML Web Browser Application

- The XAML web browser application is a compiled application that runs within a web browser.
- It doesn't have full trust and cannot access all computer parts a stand-alone application can.
- It can run in browsers that support this option and it requires .NET Framework version 3 or above to be installed.
- This type of applications is also known as ex-bap.

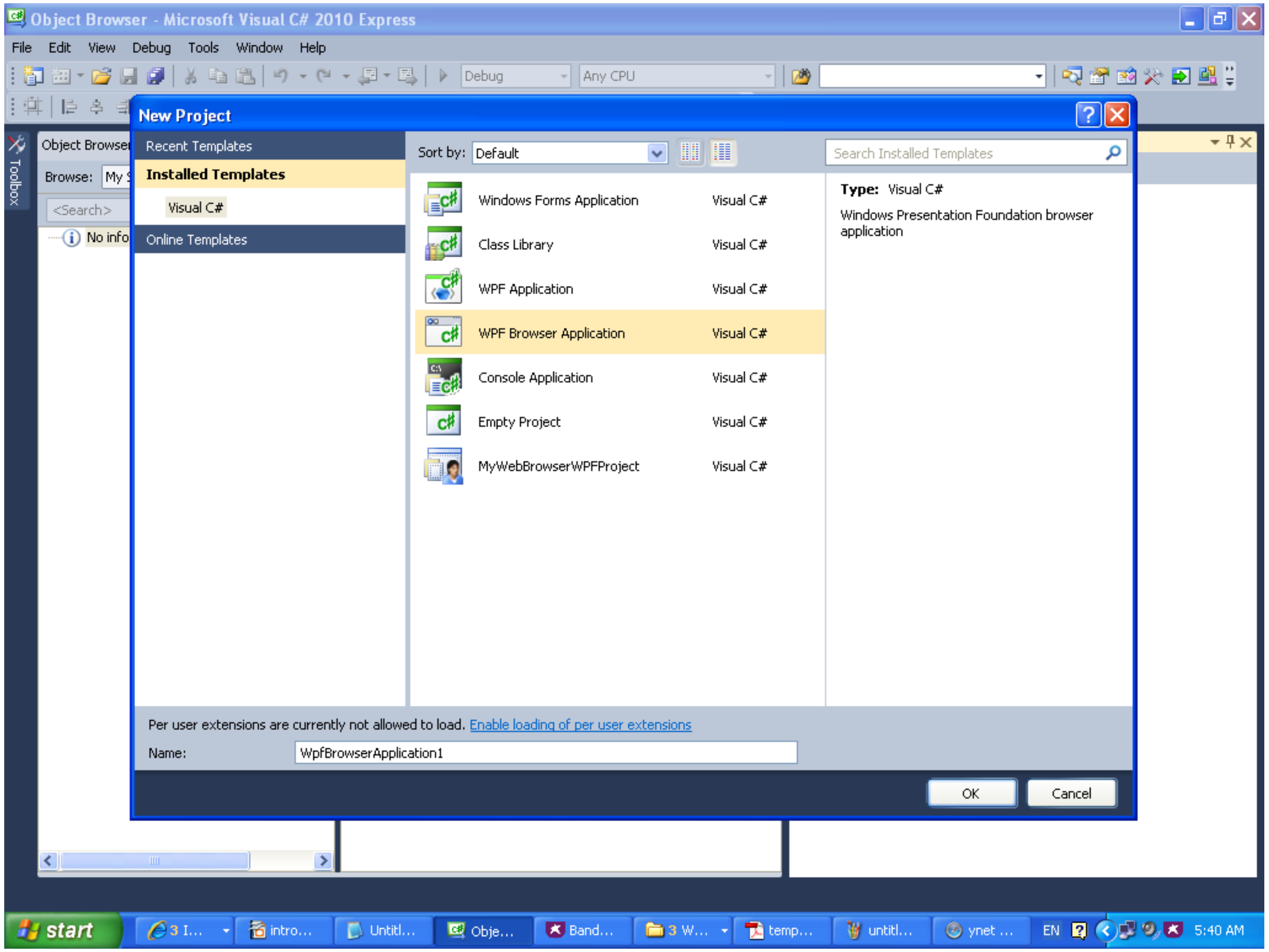
Loose XAML Pages

- The loose XAML pages are simple XAML files displayed in a web browser. They can be viewed by any browser that understands XAML.
- The loose XAML pages don't require the .NET framework. We can run them on operating systems that cannot install the .NET framework, such as Unix, Linux, Solaris and Mac.

WPF Visual Studio Support

- When creating a new project we can choose between WPF Browser Application and WPF Application.

WPF Visual Studio Support


Object Browser - Microsoft Visual C# 2010 Express
File Edit View Debug Tools Window Help
Debug
Any CPU
Toolbox
Object Browser
Browse: My
<Search>
No info
start

New Project

Recent Templates
Installed Templates
Visual C#
Online Templates

Sort by: Default

Search Installed Templates

Windows Forms Application	Visual C#
Class Library	Visual C#
WPF Application	Visual C#
WPF Browser Application	Visual C#
Console Application	Visual C#
Empty Project	Visual C#
MyWebBrowserWPFProject	Visual C#

Type: Visual C#

Windows Presentation Foundation browser application

Per user extensions are currently not allowed to load. Enable loading of per user extensions

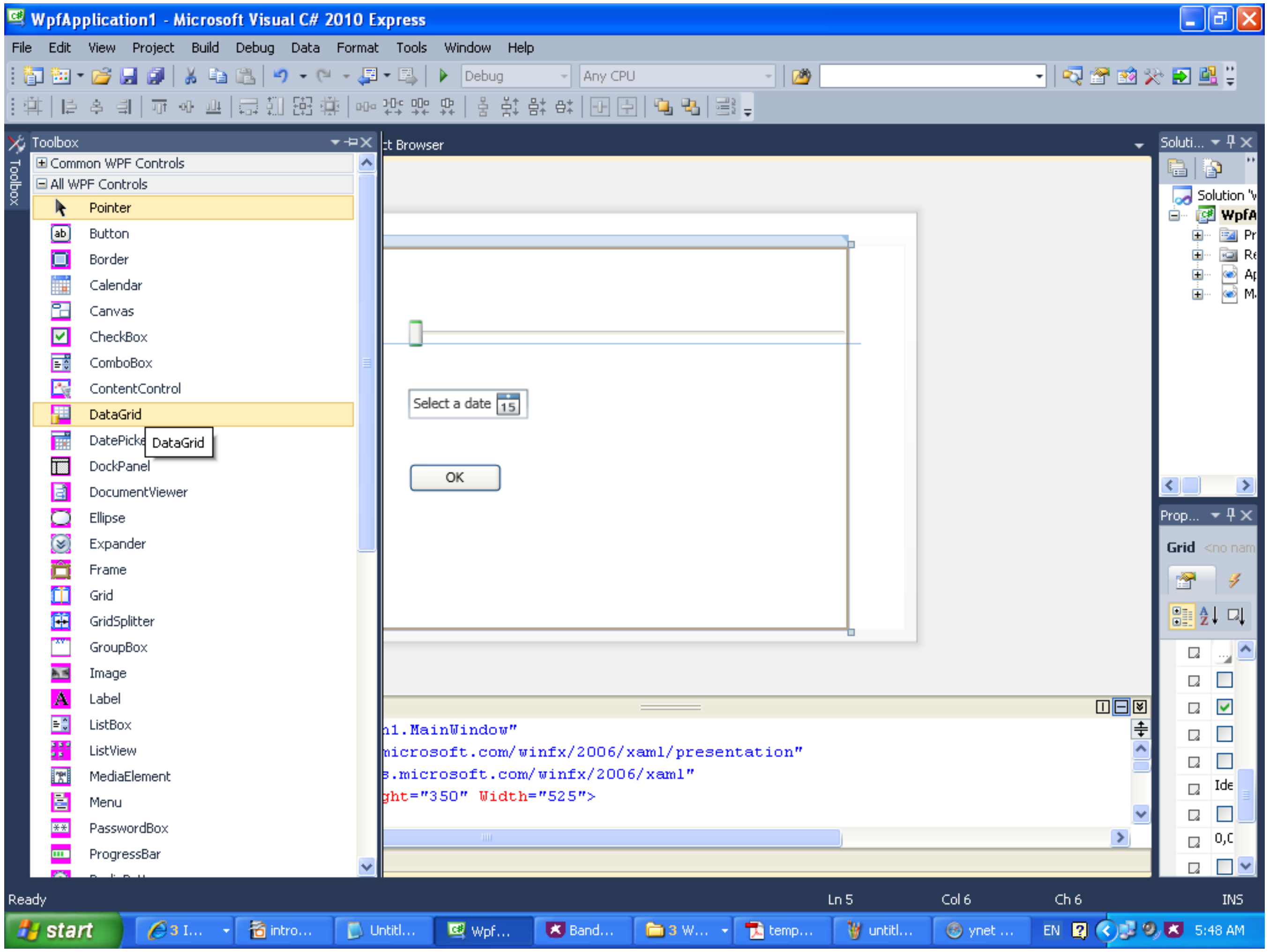
Name: WpfBrowserApplication1

OK Cancel

WPF Visual Studio Designer

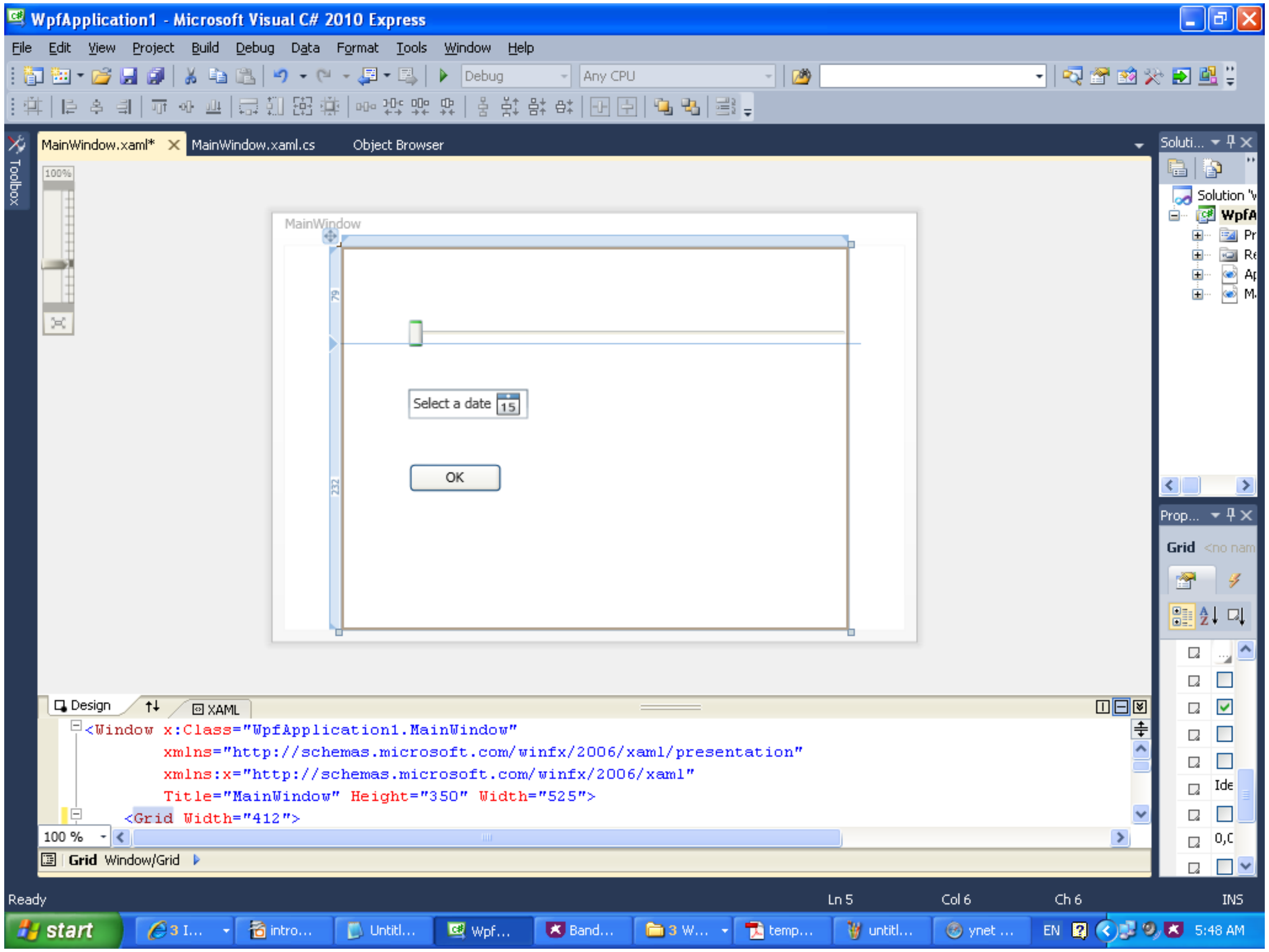
- The visual studio designer allows us to design the user interface.
- Using the visual studio designer we can add controls to the windows, resize and arrange them, place new controls inside others, set controls' properties and more.
- The scale slider on left changes the magnification level so we can zoom in and out in according with our needs.

WPF Visual Studio Designer



```
n1.MainWindow"
microsoft.com/winfx/2006/xaml/presentation"
s.microsoft.com/winfx/2006/xaml"
ght="350" Width="525">
```

WPF Visual Studio Designer

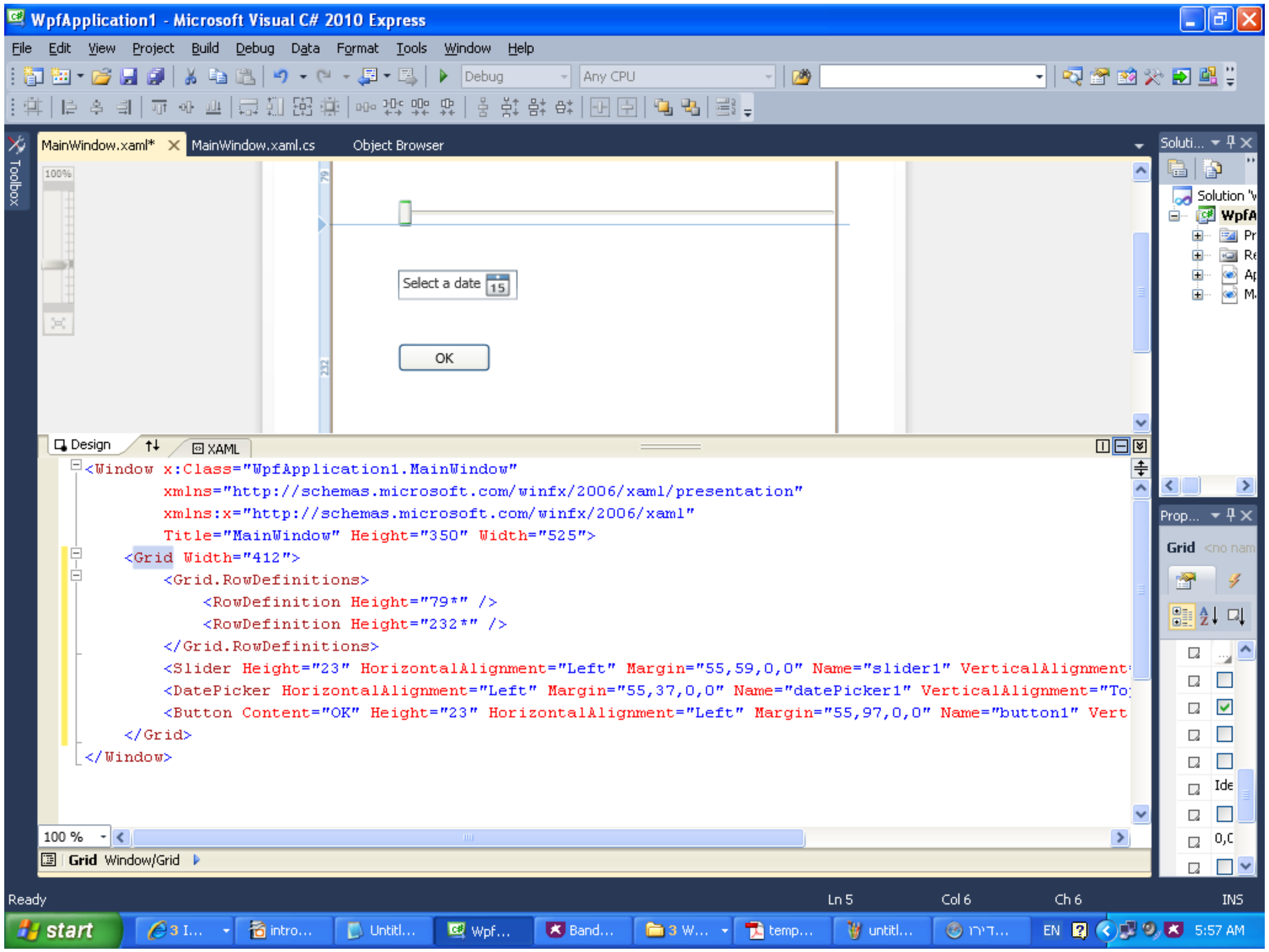


```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid Width="412">
```

The XAML Editor

- The XAML editor allows us to manually edit the XAML code.
- There are cases when it is simpler to edit the XAML code directly.

The XAML Editor

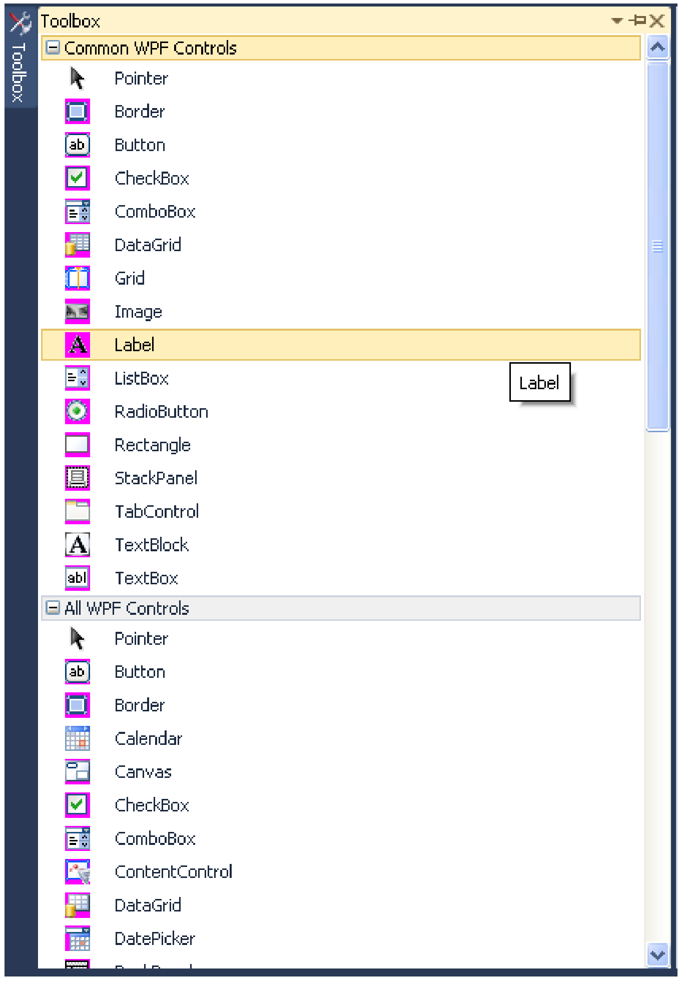

WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Format Tools Window Help
MainWindow.xaml* MainWindow.xaml.cs Object Browser
Select a date
OK
Design
XAML
Grid Window/Grid
Ready Ln 5 Col 6 Ch 6 INS

```
<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 Width="412">
        <Grid.RowDefinitions>
            <RowDefinition Height="79*" />
            <RowDefinition Height="232*" />
        </Grid.RowDefinitions>
        <Slider Height="23" HorizontalAlignment="Left" Margin="55,59,0,0" Name="slider1" VerticalAlignment
        <DatePicker HorizontalAlignment="Left" Margin="55,37,0,0" Name="datePicker1" VerticalAlignment="To
        <Button Content="OK" Height="23" HorizontalAlignment="Left" Margin="55,97,0,0" Name="button1" Vert
    </Grid>
</Window>
```

The Toolbox

- The toolbox holds controls we can place on the window we design.
- We place the selected controls on our window using a simple drag&drop.

The Toolbox

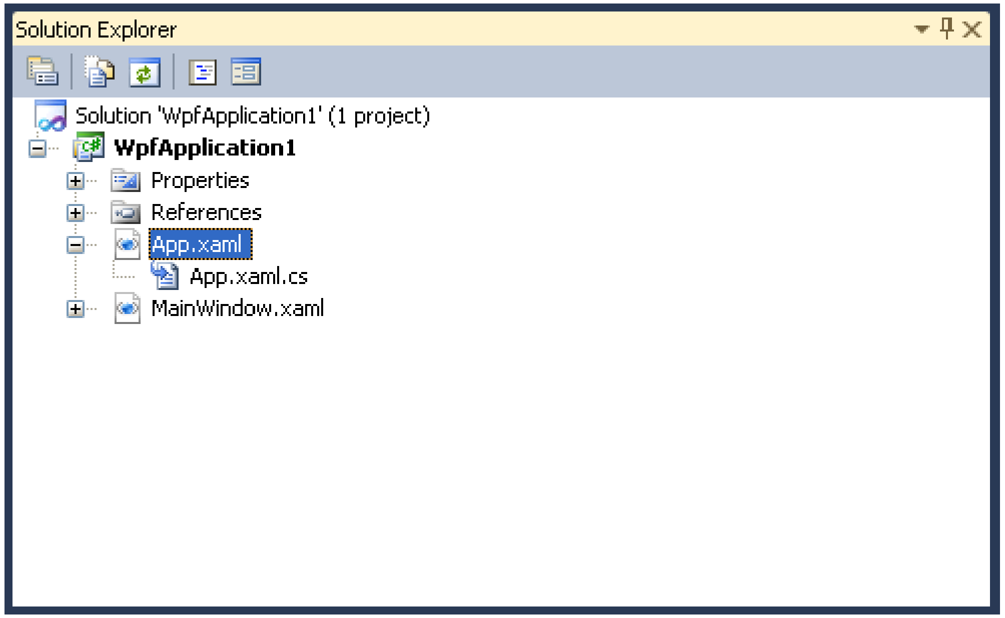


The separation between the common WPF controls and the rest assists when using them in our code.

The Solution Explorer

- The solution explorer lists the files. Double click a file opens it in its appropriate editor.
- Using the solution explorer we can manage all files, including the resources ones (images etc.) we add to our project.

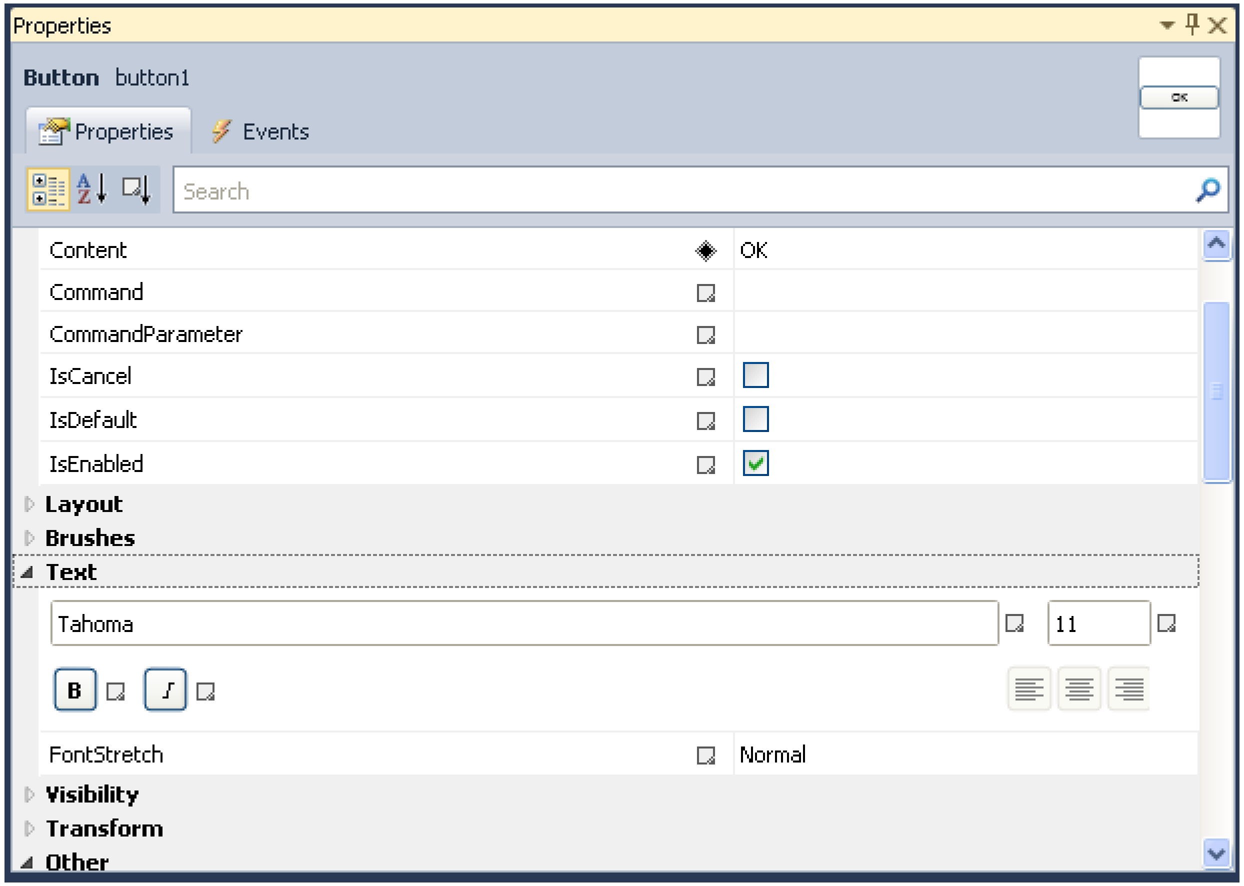
The Solution Explorer



The Properties Window

- The properties window allows us to view and edit properties values of the control we select.
- Some of the properties cannot be edited using this window and in order to edit them we must use the XAML editor.

The Properties Window



Properties

Button button1

Properties | Events

Search

Content	OK
Command	
CommandParameter	
IsCancel	☐
IsDefault	☐
IsEnabled	✓

Layout

Brushes

Text

Tahoma 11

FontStretch Normal

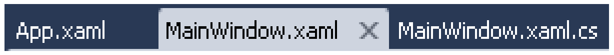
Visibility

Transform

Other

The Window Tabs

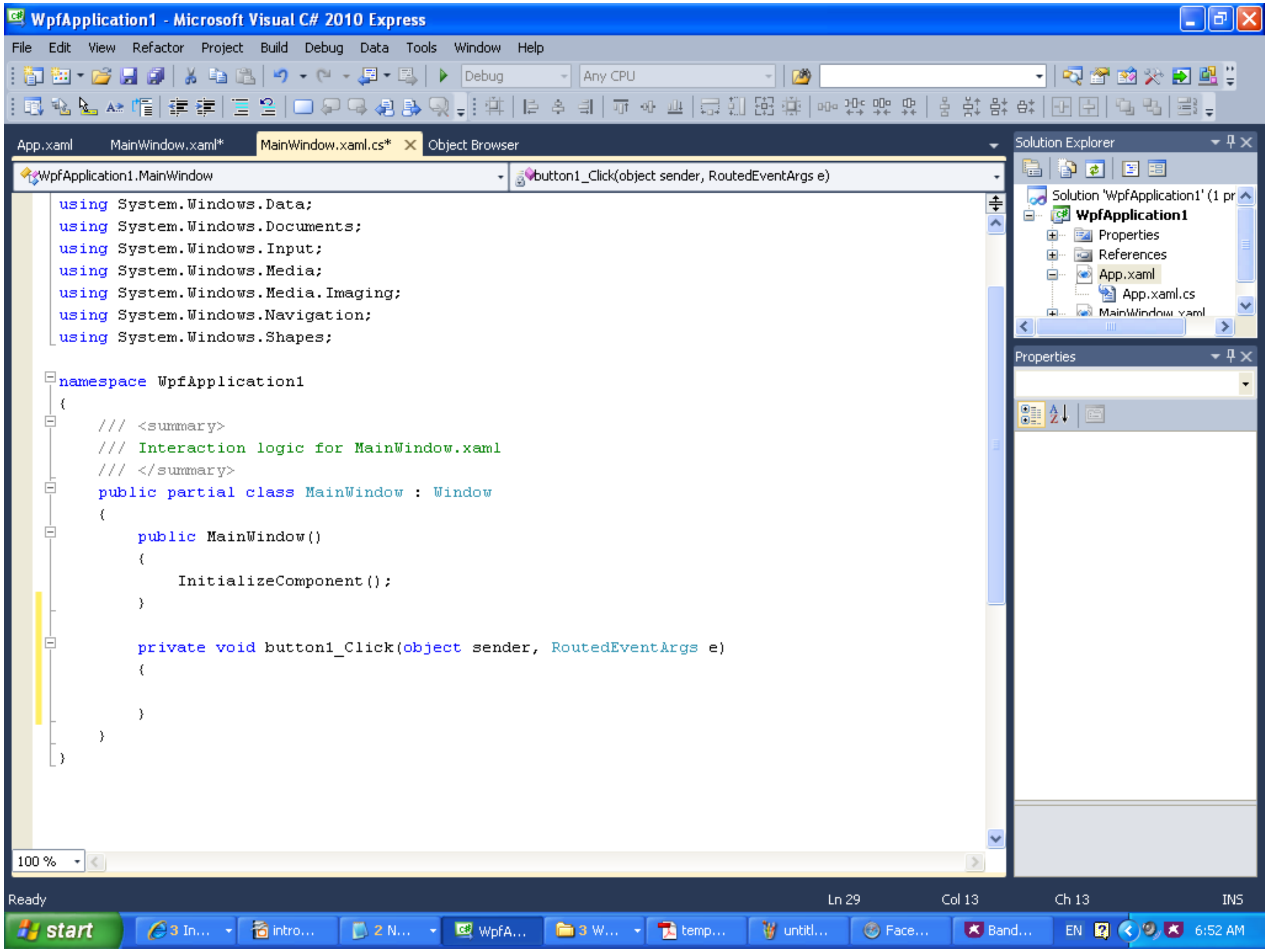
- The window tabs allows us to us select out of those opened files the one we want to edit.



Events Handlers

- The simplest way to attach code to the user interface is double click the user control we want to handle (e.g. a button we want to handle its clicks) and write our code within the automatically opened code editor.

Events Handlers



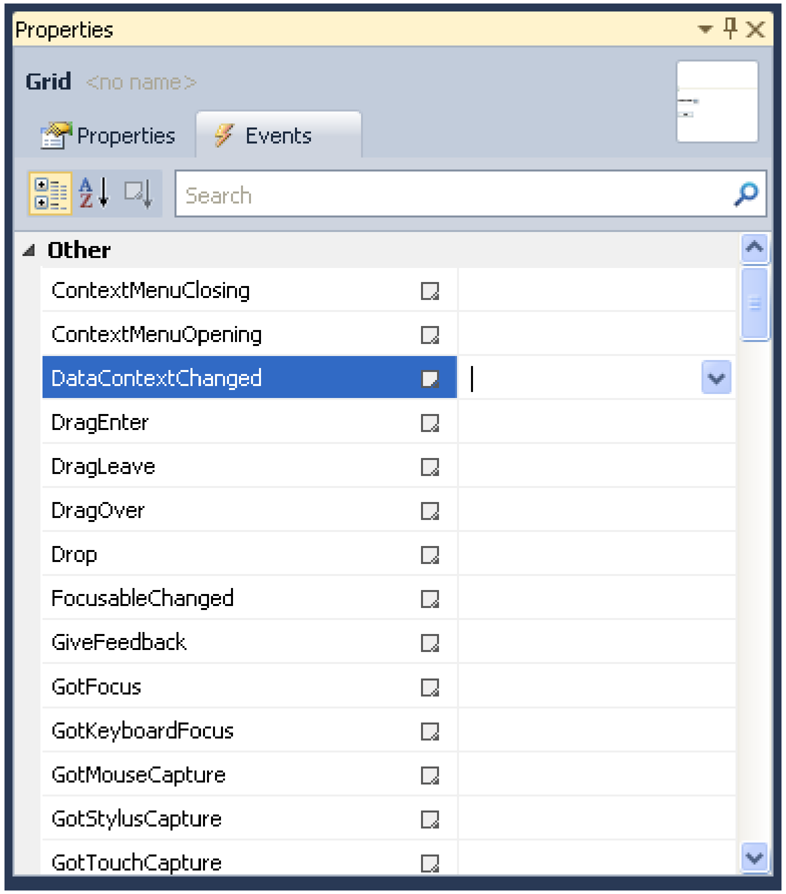
WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Refactor Project Build Debug Data Tools Window Help
App.xaml MainWindow.xaml* MainWindow.xaml.cs* Object Browser
WpfApplication1.MainWindow
button1_Click(object sender, RoutedEventArgs e)
Solution Explorer
Solution 'WpfApplication1' (1 pr
WpfApplication1
Properties
References
App.xaml
App.xaml.cs
Properties
Ready Ln 29 Col 13 Ch 13 INS

```
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

Events Handlers

- When dealing with non default events handlers we can add our own events handlers.
- Double click on the event will create an appropriate event handler and open it in the Code Editor.

Events Handlers



ContextMenuClosing
ContextMenuOpening
DataContextChanged
DragEnter
DragLeave
DragOver
Drop
FocusableChanged
GiveFeedback
GotFocus
GotKeyboardFocus
GotMouseCapture
GotStylusCapture
GotTouchCapture

Events Handlers

- We can add a new event handler manually as well. Within the XAML code we just need to add the appropriate event attribute to the XAML code. Its value should be the name of the handling method.

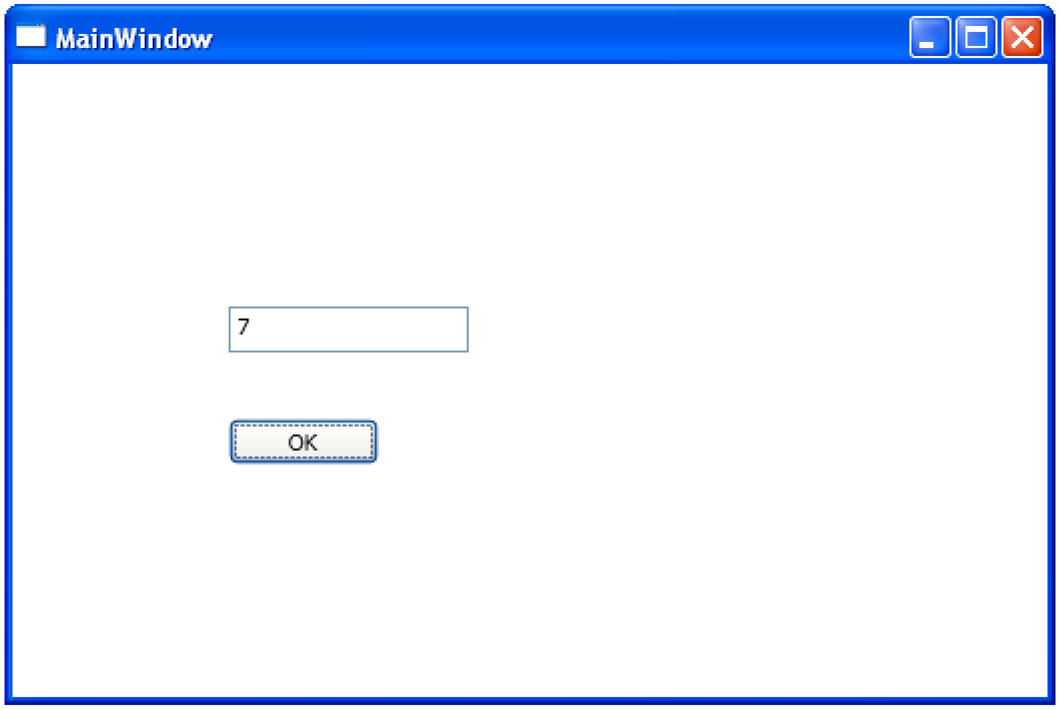
Events Handlers

```
namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            bt.Click += button1_Click;
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            tf.Text = ""+(1+double.Parse(tf.Text));
        }
    }
}
```

Events Handlers

```
<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 Width="412">
        <Grid.RowDefinitions>
            <RowDefinition Height="79*" />
            <RowDefinition Height="232*" />
        </Grid.RowDefinitions>
        <Button Content="OK" Height="23" HorizontalAlignment="Left"
                Margin="55,97,0,0" Name="bt" VerticalAlignment="Top"
                Width="75" Grid.Row="1" />
        <TextBox Grid.Row="1" Text="1" Height="23" HorizontalAlignment="Left"
                Margin="55,41,0,0" Name="tf" VerticalAlignment="Top"
                Width="120" />
    </Grid>
</Window>
```

Events Handlers



The Expression Blend

- The Expression Blend environment provides with excellent tools for creating XAML interfaces.
- The Expression Blend can be easily integrated within the Visual Studio.
- Expression Blend doesn't have a free version. You can get a trial version at `expression.microsoft.com/cc136530.aspx`.