IAP C# Lecture 5
XAML and the
Windows Presentation Foundation

Geza Kovacs

What is Windows Presentation Foundation (WPF)?

- A toolkit for building graphical user interfaces (GUI) for an application
- Ships as part of the .NET platform for Windows, Windows Phone 7, and Silverlight

Elements of a GUI application

- The **frontend / interface**: where are the **controls** (buttons, labels, sliders, text boxes) placed, how does the application look like to your user?
- The **backend** code: when the user clicks a button, what code gets executed?

Coding a GUI

- Each control (button, label, slider, text box) is represented by an object
- Each object representing a control has some properties: (ex: the text on a button)
- Writing an GUI in C#: create objects representing your Window and each of the controls, set all their properties, then show the interface

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Defines a Window

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Sets the Window's title

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Defines a Button control

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Sets the text in the button

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Sets the button text font size

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Adds the button to the window

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Shows the window

Displaying a Window with a Button

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass
{
    [STAThread]
    static void Main(string[] args)
    {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Needed to start a WPF application

Adding Interactivity to an application

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs (ie, a button is clicked), by **subscribing** to the event

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

An event

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

Subscribing to an event

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

Which method to call whenever event occurs

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

Method's signature depends on the event

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintSomethingElse(object sender, RoutedEventArgs e) {
    Console.WriteLine("Something Else");
}

static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    …
}
```

- Whenever the user does something to the GUI (like, clicking a button), an **event** will be triggered
- You can have a method be called whenever an event occurs, by **subscribing** to the event

```
static void PrintSomethingElse(object sender, RoutedEventArgs e) {
    Console.WriteLine("Something Else");
}

static void PrintHelloWorld(object sender, RoutedEventArgs e) {
    Console.WriteLine("Hello World");
}

static void Main(string[] args)
{
    Button button = new Button();
    button.Click += PrintHelloWorld;
    button.Click += PrintSomethingElse;
    …
}
```

Can subscribe multiple methods to an event; all will be called whenever button is pressed

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Define a window, set its title

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Define a new button, set the button text and font size

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Subscribe methods to the button's click event (both called when clicked)

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Add button to the window

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

Show the window

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

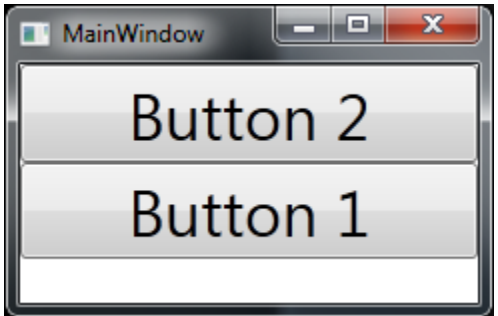
Start the application

```
using System;
using System.Windows;
using System.Windows.Controls;
static class MyMainClass {
    static void PrintHelloWorld(object sender, RoutedEventArgs e) {
        Console.WriteLine("Hello World");
    }

    static void PrintSomethingElse(object sender, RoutedEventArgs e) {
        Console.WriteLine("Something Else");
    }

    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        Button button = new Button();
        button.Content = "Click Me";
        button.FontSize = 32.0;
        button.Click += PrintHelloWorld;
        button.Click += PrintSomethingElse;
        window.Content = button;
        window.Show();
        Application app = new Application();
        app.Run();
    }
}
```

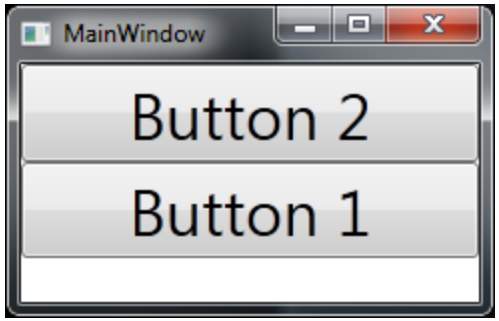
Start the application

Layouts

- So far, we've had only 1 button. What if we want multiple buttons (or textbox, label, or other controls) on screen?
- In WPF, we usually use a layout to organize multiple widgets on screen
  - StackPanel: stacks items horizontally or vertically
  - Grid: organizes them into columns and rows



```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass {
    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        window.Show();
        Button button1 = new Button();
        button1.FontSize = 36.0;
        button1.Content = "Button 1";
        Button button2 = new Button();
        button2.FontSize = 36.0;
        button2.Content = "Button 2";
        StackPanel panel = new StackPanel();
        panel.Children.Add(button2);
        panel.Children.Add(button1);
        window.Content = panel;
        Application app = new Application();
        app.Run();
    }
}
```



StackPanel is a layout for organizing our 2 buttons

```
using System;
using System.Windows;
using System.Windows.Controls;

static class MyMainClass {
    [STAThread]
    static void Main(string[] args) {
        Window window = new Window();
        window.Title = "Hello World";
        window.Show();
        Button button1 = new Button();
        button1.FontSize = 36.0;
        button1.Content = "Button 1";
        Button button2 = new Button();
        button2.FontSize = 36.0;
        button2.Content = "Button 2";
        StackPanel panel = new StackPanel();
        panel.Orientation = Orientation.Horizontal;
        panel.Children.Add(button2);
        panel.Children.Add(button1);
        window.Content = panel;
        Application app = new Application();
        app.Run();
    }
}
```

Can change orientation

Separation of frontend and backend

- The frontend / interface: where are the buttons placed, how does the application look like to your user?
- The backend code: when the user clicks a button, what code gets executed?
- GUI applications should try to keep these are separate as possible
  - Makes it easy to replace interface while keeping the backend code working correctly
- In WPF, this is accomplished via **XAML**

XAML

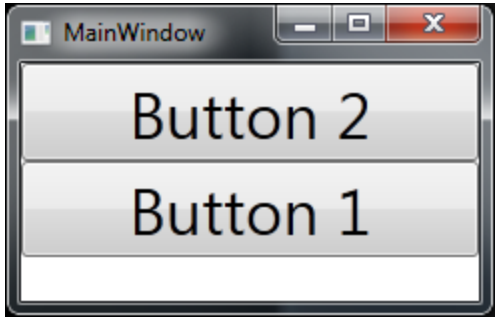
- A specialized (XML-based) language for defining an interface in WPF
  - Can describe an interface more concisely than simply using C#
- Ex: defining a button in WPF vs in C#:

```
<Button Content="Click Me" FontSize="32" />
```

```
Button button = new Button();
button.Content = "Click Me";
button.FontSize = 32.0;
```

Defining a stack layout in XAML vs C#

```
<StackPanel>
    <Button Content="Button 2" FontSize="32" />
    <Button Content="Button 1" FontSize="32" />
</StackPanel>
```




```
Button button1 = new Button();
button1.FontSize = 36.0;
button1.Content = "Button 1";
Button button2 = new Button();
button2.FontSize = 36.0;
button2.Content = "Button 2";
StackPanel panel = new StackPanel();
panel.Children.Add(button2);
panel.Children.Add(button1);
```

Defining a stack layout in XAML vs C#

```
<StackPanel Orientation="Horizontal">
    <Button Content="Button 2" FontSize="32" />
    <Button Content="Button 1" FontSize="32" />
</StackPanel>
```


```
Button button1 = new Button();
button1.FontSize = 36.0;
button1.Content = "Button 1";
Button button2 = new Button();
button2.FontSize = 36.0;
button2.Content = "Button 2";
StackPanel panel = new StackPanel();
panel.Orientation = Orientation.Horizontal;
panel.Children.Add(button2);
panel.Children.Add(button1);
```

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel Orientation="Horizontal">
        <Button Content="Button 2" FontSize="32" />
        <Button Content="Button 1" FontSize="32" />
    </StackPanel>
</Window>
```

MyWindow.xaml – the interface

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel Orientation="Horizontal">
        <Button Content="Button 2" FontSize="32" />
        <Button Content="Button 1" FontSize="32" />
    </StackPanel>
</Window>
```

MyWindow.xaml – the interface

```csharp
using System.Windows;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
}
```

MyWindow.xaml.cs – the backend logic for MyWindow (none at the moment)

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel Orientation="Horizontal">
        <Button Content="Button 2" FontSize="32" />
        <Button Content="Button 1" FontSize="32" />
    </StackPanel>
</Window>
```

MyWindow.xaml – the interface

```csharp
using System.Windows;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
}
```

MyWindow.xaml.cs – the backend logic for MyWindow (none at the moment)

```csharp
using System;
using System.Windows;
static class MyMainClass {
    [STAThread]
    static void Main(string[] args) {
        MyWindow window = new MyWindow();
        window.Show();
        new Application().Run();
    }
}
```

MyMainClass.cs – creates a MyWindow, shows it, and starts the WPF application

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel Orientation="Horizontal">
        <Button Content="Button 2" FontSize="32" />
        <Button Content="Button 1" FontSize="32" />
    </StackPanel>
</Window>
```

```csharp
using System.Windows;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void DoSomething1(object sender, RoutedEventArgs e) {
        Console.WriteLine("button 1 clicked");
    }

}
```

- Suppose we want some method in MyWindow.xaml.cs to be called whenever button 1 gets clicked (an event)

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel Orientation="Horizontal">
        <Button Content="Button 2" FontSize="32" />
        <Button Content="Button 1" FontSize="32" Click="DoSomething1" />
    </StackPanel>
</Window>
```

Subscribes DoSomething1 method to button1's Click event

```
using System.Windows;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void DoSomething1(object sender, RoutedEventArgs e) {
        Console.WriteLine("button 1 clicked");
    }

}
```

Executed whenever button is clicked

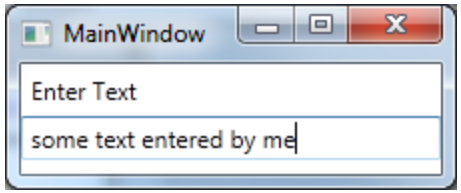
- Suppose we want some method in MyWindow.xaml.cs to be called whenever button 1 gets clicked
  - Add "Click=DoSomething1" in XAML to subscribe the method to the event

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Label Content="Enter Text" />
        <TextBox />
    </StackPanel>
</Window>
```

Label

TextBox

```
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
}
```

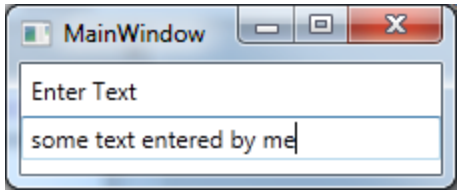


- TextBox control – user can enter text in it
- Label control – displays some message

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Label Content="Enter Text" />
        <TextBox TextChanged="RunWhenTextChanges“ />
    </StackPanel>
</Window>
```

Subscribe RunWhenTextChanges to TextChanged event

```
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void RunWhenTextChanges(object sender, TextChangedEventArgs e) {
        Console.WriteLine("text changed");
    }
}
```



Executed whenever text changes

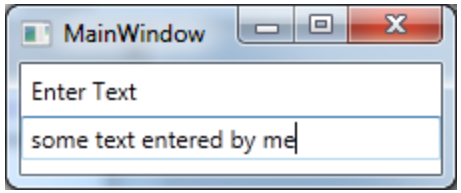
- TextBox’s TextChanged event is triggered whenever user enters text into the textbox

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Label Content="Enter Text" />
        <TextBox TextChanged="RunWhenTextChanges" Name="textBoxA" />
    </StackPanel>
</Window>
```

Can refer to this TextBox as textBoxA in code

```csharp
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void RunWhenTextChanges(object sender, TextChangedEventArgs e) {
        Console.WriteLine(textBoxA.Text);
    }
}
```

textBoxA.Text: text entered in the TextBox



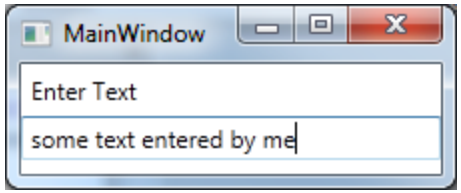
- Suppose we want to retrieve text from the TextBox – we'll need the name of the instance. Can specify this in XAML using the "Name" property

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Label Content="Enter Text" />
        <TextBox TextChanged="RunWhenTextChanges" Name="textBoxA" />
    </StackPanel>
</Window>
```

Can refer to this TextBox as textBoxA in code

```csharp
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void RunWhenTextChanges(object sender, TextChangedEventArgs e) {
        Console.WriteLine(textBoxA.Text);
    }
}
```

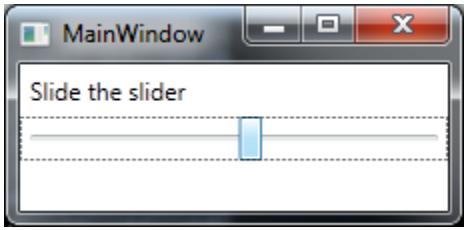
textBoxA.Text: text entered in the TextBox



- Suppose we want to retrieve text from the TextBox – we'll need the name of the instance. Can specify this in XAML using the "Name" property

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Label Content="Slide the slider" />
        <Slider Minimum="0" Maximum="100" Name="sliderA"
                ValueChanged="sliderA_ValueChanged" />
    </StackPanel>
</Window>
```

```csharp
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void sliderA_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
        Console.WriteLine(sliderA.Value);
    }
}
```

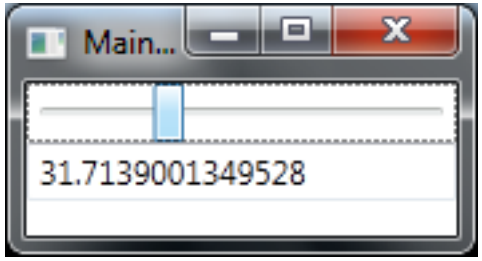


sliderA.Value: value in the slider

- Slider: can be slid by user, between a Minimum and Maximum value. ValueChanged event occurs when user slides the slider

```xml
<Window x:Class="MyWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow">
    <StackPanel>
        <Slider Minimum="0" Maximum="100" Name="sliderA"
                ValueChanged="sliderA_ValueChanged" />
        <TextBox Name="someTextBox" />
    </StackPanel>
</Window>
```

```csharp
using System;
using System.Windows;
using System.Windows.Controls;
public partial class MyWindow : Window {
    public MyWindow() {
        InitializeComponent();
    }
    public void sliderA_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
        someTextBox.Text = sliderA.Value.ToString();
    }
}
```



- Sets text in someTextBox whenever slider value changes