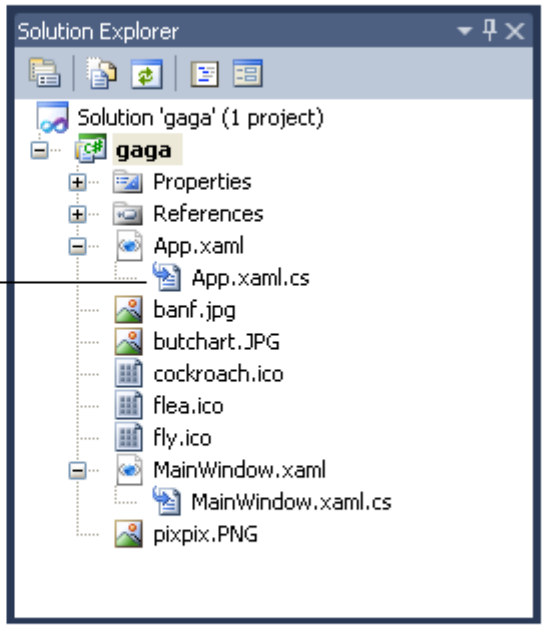
Events

Introduction

- The code responsible for handling the user events is known as the 'code behind'.
- Working with the Visual Studio 2010 we can easily get access to that code.
- Each `XAML` file has a 'code behind' one. Its name is identical to the name of the `XAML` file + the '`.cs`' extension.

Introduction

This is the 'code behind' file of the XAML file.



Event Name Attributes

- The controls have attributes we can refer when handling their related user events.
- The name of the event handling attribute is identical to the name of the event.
- The value of that attribute is the name of the method that will be called.

```
...
<Button Content="Click Me" Name="btnApply" Click="bt_handle"/>
...
```

The Event Handling Method

- Using the Visual Studio, when double clicking the control we are interested in handling its related event we will be taken to the relevant method within the 'code behind' file.
- The name of that method is identical to the value we assign the event handling attribute.

```
...
private void bt_handle(object sender, RoutedEventArgs e)
{


}
...
```

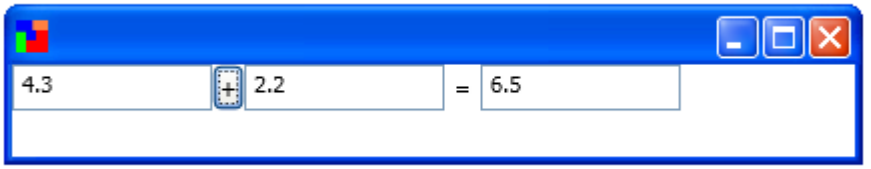
The Event Handling Method

```
<Window     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            x:Class="gaga.MainWindow" Width="800" Height="200" Icon="pixpix.png">

    <WrapPanel HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
            Orientation="Horizontal">

        <TextBox Name="numA" Width="100"/>
        <Button Name="plus" Content="+" Click="bt_handle"/>
        <TextBox Name="numB" Width="100"/>
        <Label Content="="/>
        <TextBox Name="result" Width="100"/>

    </WrapPanel>

</Window>
```

The Event Handling Method

```
namespace gaga
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
        }


        private void bt_handle(object sender, RoutedEventArgs e)
        {
            result.Text = "" + (double.Parse(numA.Text) +
                                        double.Parse(numB.Text));
        }
     }
}
```

The Event Handling Method



Setting Event Handlers at Run-Time

- We can easily set event handlers at run time. The event handling attribute of the control we refer is actually an event.

```
...
bt.Click += MyNewHanderMethod;
bt.Click += AnotherNewHandlerMethod;
...
```

Setting Event Handlers at Run-Time

- This technique allows the designer to work separately from the developer.
- The programmer can write code that will attach the event handling methods during run-time. The only requirement is having each one of the controls assigned with a name.

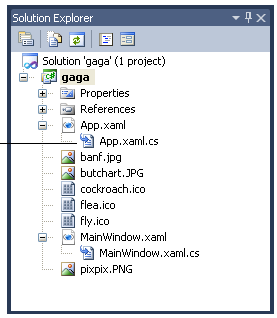
Events

Introduction

- The code responsible for handling the user events is known as the 'code behind'.
- Working with the Visual Studio 2010 we can easily get access to that code.
- Each `XAML` file has a 'code behind' one. Its name is identical to the name of the `XAML` file + the '`.cs`' extension.

Introduction

This is the 'code behind' file of the XAML file.



Event Name Attributes

- The controls have attributes we can refer when handling their related user events.
- The name of the event handling attribute is identical to the name of the event.
- The value of that attribute is the name of the method that will be called.

```
...
<Button Content="Click Me" Name="btnApply" Click="bt_handle"/>
...
```

The Event Handling Method

- Using the Visual Studio, when double clicking the control we are interested in handling its related event we will be taken to the relevant method within the 'code behind' file.
- The name of that method is identical to the value we assign the event handling attribute.

```
...
private void bt_handle(object sender, RoutedEventArgs e)
{

}
...
```

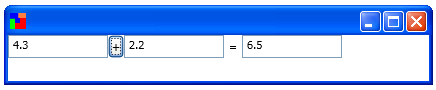
The Event Handling Method

```
<Window     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            x:Class="gaga.MainWindow" Width="800" Height="200" Icon="pixpix.png">

    <WrapPanel HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
            Orientation="Horizontal">

        <TextBox Name="numA" Width="100"/>
        <Button Name="plus" Content="+" Click="bt_handle"/>
        <TextBox Name="numB" Width="100"/>
        <Label Content="="/>
        <TextBox Name="result" Width="100"/>

    </WrapPanel>

</Window>
```

The Event Handling Method

```
namespace gaga
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
        }


        private void bt_handle(object sender, RoutedEventArgs e)
        {
            result.Text = "" + (double.Parse(numA.Text) +
                                double.Parse(numB.Text));
        }
     }
}
```

The Event Handling Method



Setting Event Handlers at Run-Time

- We can easily set event handlers at run time. The event handling attribute of the control we refer is actually an event.

```
...
bt.Click += MyNewHanderMethod;
bt.Click += AnotherNewHandlerMethod;
...
```

Setting Event Handlers at Run-Time

- This technique allows the designer to work separately from the developer.
- The programmer can write code that will attach the event handling methods during run-time. The only requirement is having each one of the controls assigned with a name.