



# Windows® Phone

## Making an application with Silverlight

Session 2.3



# Topics

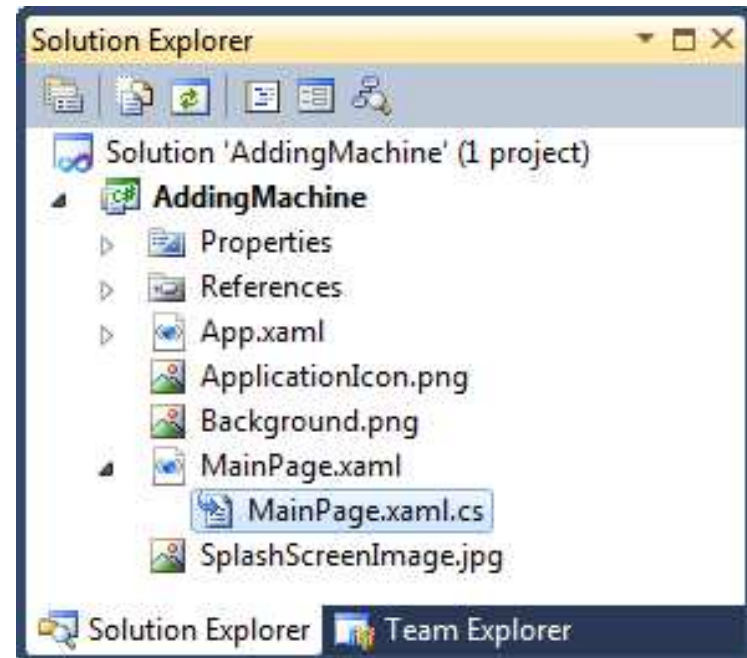
- XAML files and code behind pages
- Adding a calculation method
- Events and programs
- Binding to Silverlight events
- Managing event properties
- Events and XAML

# Silverlight and C#

- Up until now all we have done is create XAML designs for our application
  - We have used the editing surface and the properties pane, along with a text editor
- Now we are going to see where the program content lives
- This code will provide the active content for our calculator
  - This will work out the result

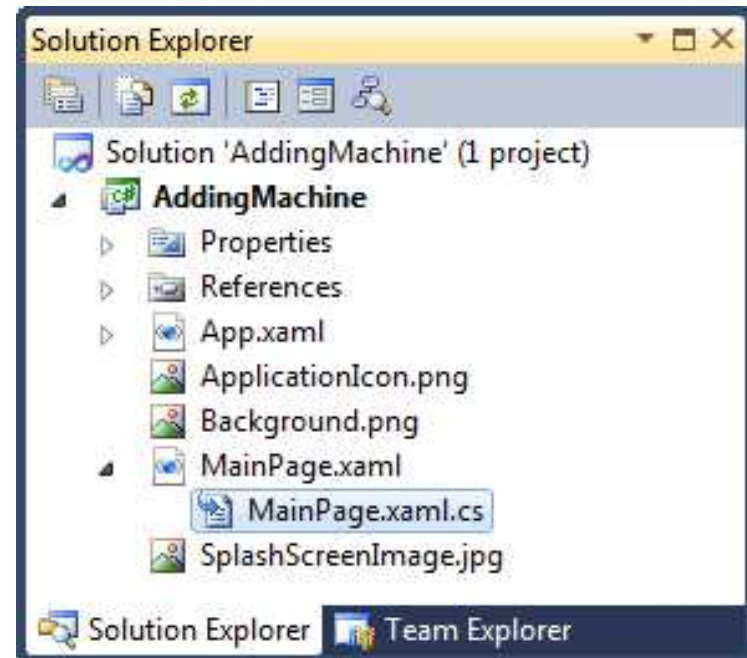
# The Visual Studio Solution Explorer

- The Solution Explorer is another pane in Visual Studio
- It shows all the elements in a particular solution
- This includes all the application resources and XAML files



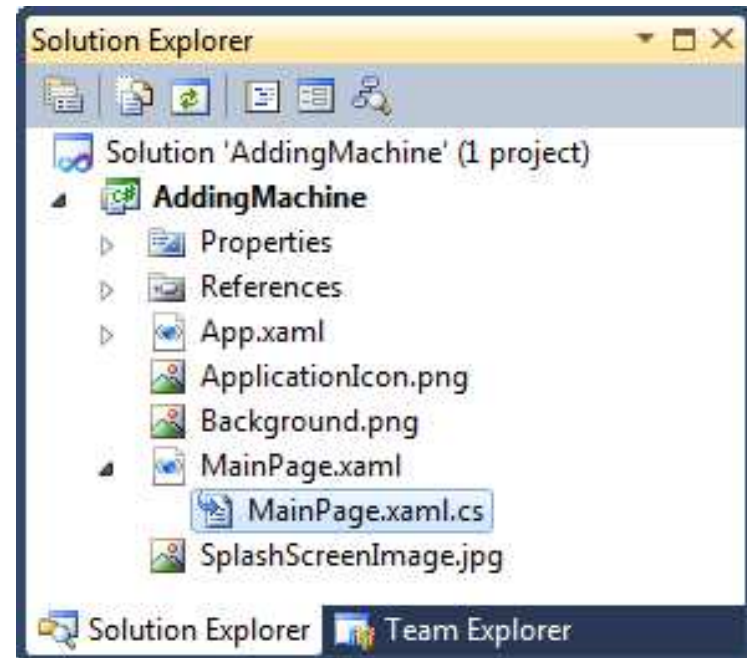
# The MainPage.xaml file

- The `MainPage.xaml` file contains the XAML that describes how the main page looks
- If you add further pages to the application they will appear as xaml pages in the solution



# The code behind a xaml page

- Each xaml page has a file of C# behind it
- We can find this by opening up the xaml file as shown in Solution Explorer



# XAML file content

```
namespace AddingMachine
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

- This is all the actual code behind our adding machine main page

# MainPage class

```
namespace AddingMachine
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

- This code is in a class called **MainPage** which extends the **PhoneApplicationPage** class



# MainPage constructor

```
namespace AddingMachine
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

- The code just contains the constructor that is called when the page instance is loaded

# Initialise Components

```
namespace AddingMachine
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

- The constructor calls a method that initialises all the components on the page

# Demo

Demo 1: Code Behind



# Health Warning

- We have just had a quick look in the “engine room” of Silverlight
- This is somewhere you should never go
- Do not make any changes to this code or Visual Studio may get very upset and you will almost certainly break your program

# Running our application

- We can run our application if we like
- We can type in values and press the equals button
- But nothing happens
- Next we need to make a method that will do the calculation and get it to run



# calculateResult method

```
private void calculateResult()  
{  
    float v1 = float.Parse(firstNumberTextBox.Text);  
    float v2 = float.Parse(secondNumberTextBox.Text);  
  
    float result = v1 + v2;  
  
    resultTextBlock.Text = result.ToString();  
}
```

- This method will calculate the result and display it
- If we call this the display will be updated

# Obtaining values

```
private void calculateResult()
{
    float v1 = float.Parse(firstNumberTextBox.Text);
    float v2 = float.Parse(secondNumberTextBox.Text);

    float result = v1 + v2;

    resultTextBlock.Text = result.ToString();
}
```

- This line gets the text out of the **firstNumberTextbox** and parses it to produce a floating point value

# Calculating a result

```
private void calculateResult()
{
    float v1 = float.Parse(firstNumberTextBox.Text);
    float v2 = float.Parse(secondNumberTextBox.Text);

    float result = v1 + v2;

    resultTextBlock.Text = result.ToString();
}
```

- This statement calculates the result we get by adding the values together



# Displaying the result

```
private void calculateResult()
{
    float v1 = float.Parse(firstNumberTextBox.Text);
    float v2 = float.Parse(secondNumberTextBox.Text);

    float result = v1 + v2;

    resultTextBlock.Text = result.ToString();
}
```

- This statement gets a string version of the result value and puts it into the text property of **resultTextBlock** – causing it to be displayed

# Silverlight elements and properties

- The method works because the Silverlight elements expose properties that the program can read (to get the numbers typed in) and write (to display the required result)
- Our program is able to work with the display elements because they are objects as far as we are concerned
- The objects are created when the program starts

# Getting the result

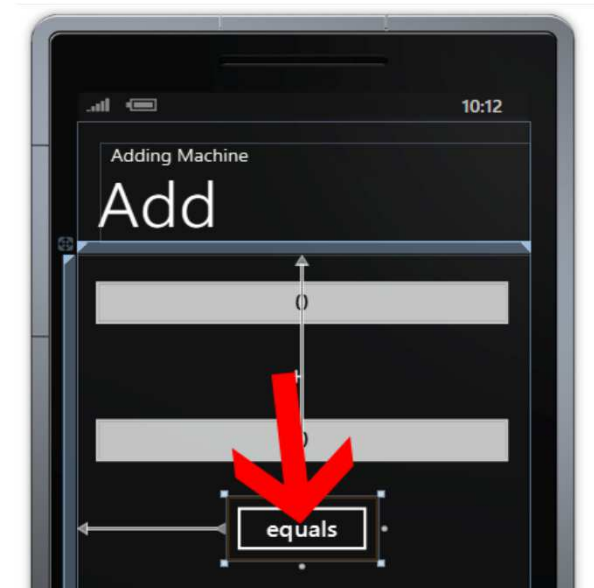
- Now all we need is to run the **calculateResult** method when the Equals button is pressed
- To do this we have to connect an event generator (the button) with the method that will run when the button produces an event
- The event is generated when the user presses the equals button

# Programs and events

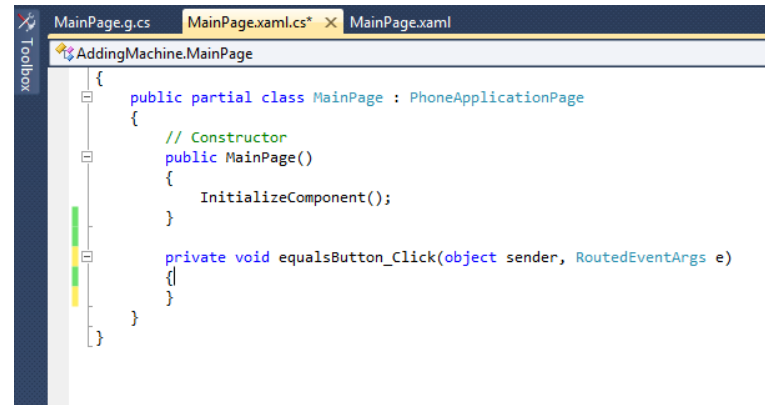
- For events to work a program must have a way of managing an event as a data object
- We need to be able to give the button an object that represents a method to be called when the event occurs
- C# provides a delegate mechanism that can create references to methods
- Fortunately Silverlight and Visual Studio make this very easy to use

# Events in Visual Studio

- We can ask Visual Studio to create an event delegate for us by double clicking on the element in the designer
- Visual Studio will do all the “plumbing” to make this work



# The Event Handler Method



```
MainWindow.g.cs | MainPage.xaml.cs* | MainPage.xaml
AddingMachine.MainPage
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        private void equalsButton_Click(object sender, RoutedEventArgs e)
        {
        }
    }
}
```

- When we have double clicked the button we find that there is now a new method in the **MainPage** class
- This will be called when the button is clicked in our program

# Displaying the result

```
private void equalsButton_Click(  
    object sender, RoutedEventArgs e)  
{  
    calculateResult();  
}
```

- Visual Studio has given this method a sensible name
  - This is based on the component name
- We just need to add a call of **calculateResult** to get the program to work

# Demo

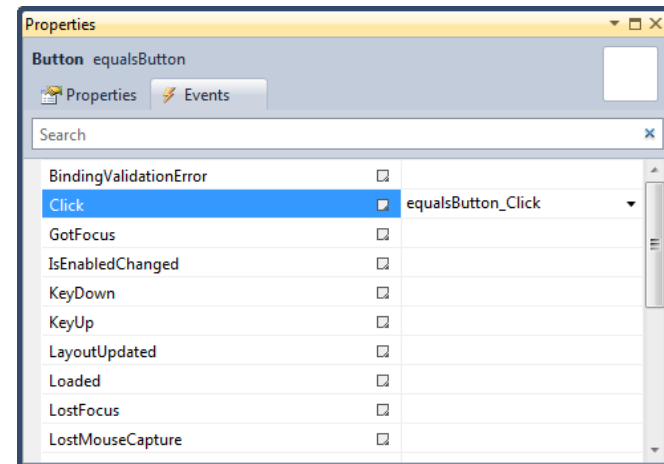
Demo 2: A Complete Solution





# Managing Events

- The properties of a Silverlight element include information about the events and what methods they are bound to
- You can see this by selecting Events in the Properties pane



# Events and XAML

```
<Button Content="equals" Height="72"  
HorizontalAlignment="Left" Margin="158,275,0,0"  
Name="equalsButton" VerticalAlignment="Top" Width="160"  
Click="equalsButton_Click" />
```

- We know that everything in Silverlight begins and ends with the XAML
- When we add an event binding in Visual Studio this adds a property to the XAML for that component
- You can see it above

# Review

- Behind each XAML file is a C# source file that contains the behaviour for that page
- You add your own behaviours by putting methods into this code file
- If you bind Silverlight events to elements the event handlers are also placed in this file
- You connect these handlers to your code to produce a working application