Windows® Phone

Program design with Silverlight

Session 2.1

Topics

- Programming and Design
- Silverlight elements
- A sample application
- The Toolbox and Design Surface
  - Creating a user interface
- Managing element names and properties
  - Customising elements in an application

Software and Design

- It is a sad fact that most programmers are not very good at graphic design
  - Although some are (lucky people)
- Also, most designers do not program much

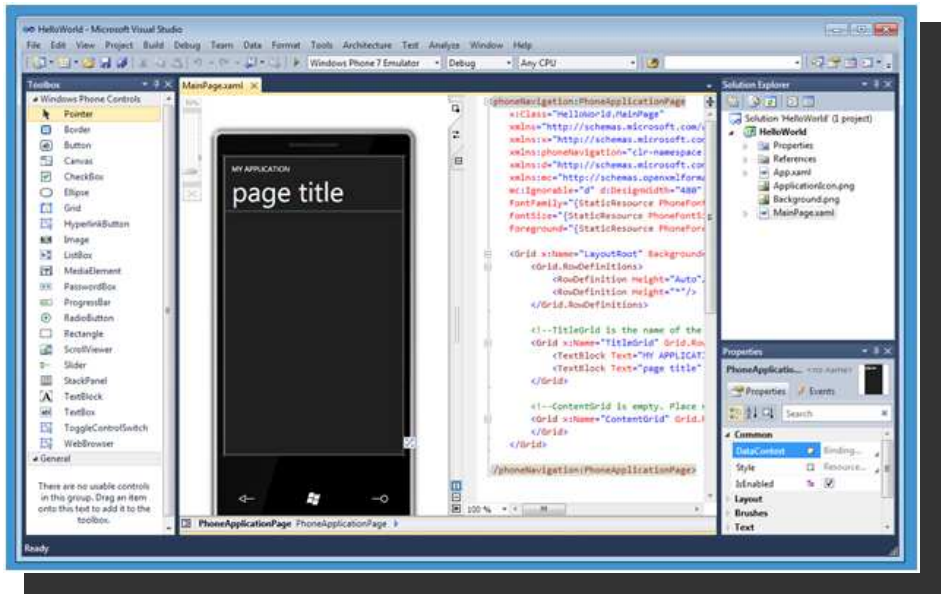
Separation of tasks

- One way to get good looking programs is to separate the graphical design aspects from the programming
  - The designer can work on the look and feel of the application
  - The programmer can implement the required behaviours
- Silverlight supports this way of working

Tools for the job : Graphical Design

- A Silverlight designer can use the "Expression Blend" to specify the appearance of the user interface
  - A version of Blend for the phone is supplied as part of the phone SDK

Tools for the job : Code Creation



- A Developer can take the user interface design and use Visual Studio build the program to make it work
  - Windows Phone developers use Visual Studio

The Metro design style

- The Windows Phone team have taken a lot of trouble over the look and feel of the phone
- They have created a design style, "Metro" to express this
- Programs on the phone should reflect this style

Design Style and programming

- As programmers we probably start of just worrying about making the program work
  - This is a very good place to start
- But in modern systems the "look and feel" of the user interface is very important
  - No matter how good the code is, if the program is hard to use it will not be popular
- You should pay careful attention to user interface issues when making your programs

Silverlight and Metro

- To make life easier for us the Metro style is "baked in" to the Windows developer tools
- The default appearance, behaviour and fonts of the user elements all match the style
- If you want to find out more about Metro you can read the style guidelines for it:

`http://msdn.microsoft.com/en-us/library/hh202915.aspx`

Silverlight and Us

- We are going to use the Silverlight designer in Visual Studio
- This can be used to create a good quality user interface that adheres to the Metro principles
- If you know any Graphic Designers it is worth getting them on your development team when you make a Marketplace application
  - They can make your programs much more impressive

Software Objects

- From our study of C# we know that we can create software objects to represent things
- We talk to our customer to establish their requirements and then identify elements that our software will manipulate
- We then create software classes which contain the data and behaviours
- For example, if we were creating a bank application..

Software Objects

```
public class Account
 {
   private decimal balance ;
   private string name ;
   public string GetName ()
   {
      return name;
   }
   public bool SetName (string newName){
   {
     // Final version will validate the name
     name = newName;
     return true;
   }
   // Other get and set methods here
 }
```

Data members

```
public class Account
 {
   private decimal balance ;
   private string name ;
    public string GetName ()
    {
        retu
    }
    public
    {
      // Final version will validate the name
      name = newName;
      return true;
    }
    // Other get and set methods here
 }
```

This is the data our bank account will hold:
The name of the holder and the balance

Data members

```
public class Account
 {
   private string name ;
   public string GetName ()
   {
      return name;
   }
   public bool SetName (string newName){
   {
     // Final version will validate the name
     name = newName;
     return true;
   }
   // Other get and set methods here
 }
```

These are the behaviours the account provides

Using the Account class

```
Account rob = new Account();
rob.SetName("Rob");
```

- The bank program can create instances of the account class and use them to hold customer information
- Each instance of an account represents a particular customer
- The bank software calls methods in the Account class to work with the data in it

Objects Oriented Design

- Object Oriented Design is a great way to manage complex projects
- It lets you break the solution down into manageable chunks
- It allows you to put off detailed implementation of the elements until you know what they need to do
- It allows you to test objects separately before they are incorporated into the product

The Silverlight Adding Machine

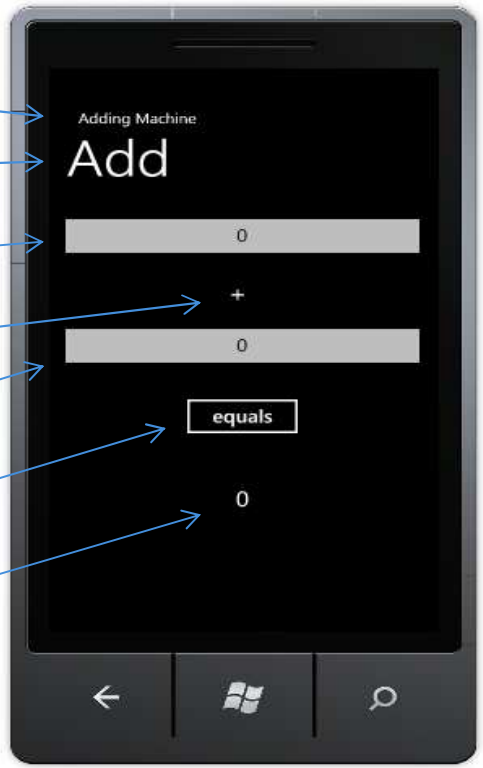
- This is a very simple calculator
  - All it can do is add two numbers
- The user types the numbers into the text boxes and presses the equals button
- The answer is displayed at the bottom of the screen

Silverlight and Objects

- Silverlight is implemented using objects to represent the elements on a User Interface
- Each of the items on the screen of the application shown is graphical rendering of a software object

Silverlight display elements

- Application title
- Page title
- First number
- Plus text
- Second number
- Equals button
- Result text



Display element data

- Each of the elements contains data elements that define how it appears on the screen
  - Position on the screen
  - Height and width
  - Font colour and size etc..
- These values are used by Silverlight when the display is drawn
- If these value are changed by the program the appearance of the element will change

Types of element

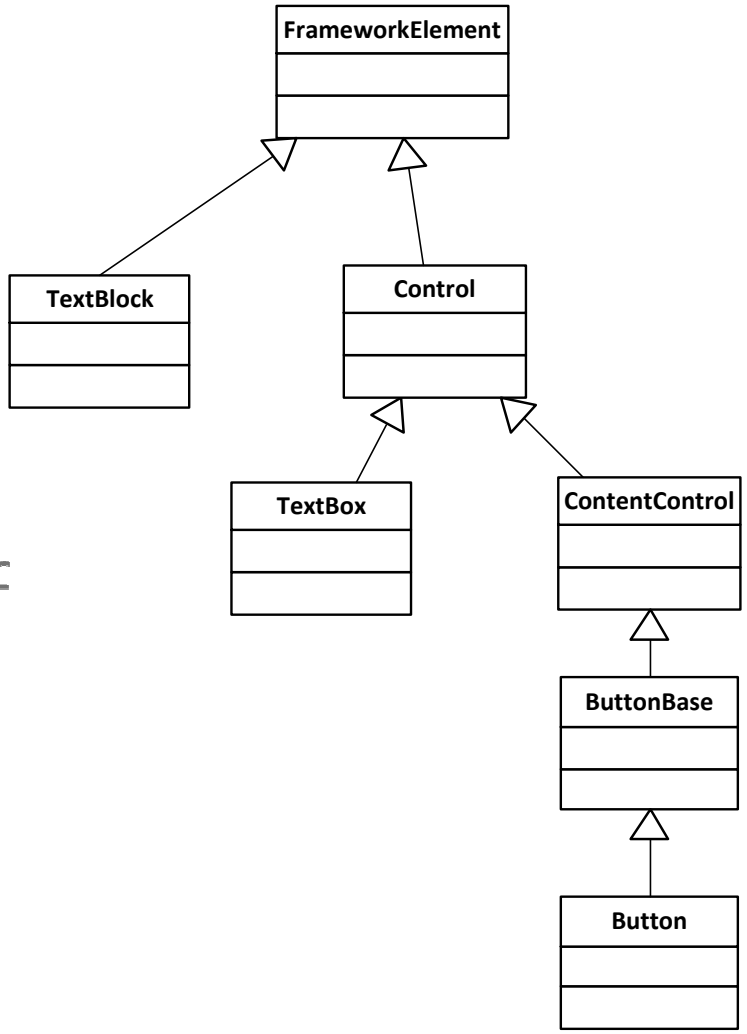
- The adding machine actually contains three different types of Silverlight display element
- `TextBox`
  - Used to receive user input from the keyboard
- `TextBlock`
  - Used to display messages to the user
- `Button`
  - Used to cause events in the application

Class Hierarchies and Silverlight

- The elements will have some properties in common
  - All will have a position on the screen and a height and width
- But there will be some properties that are specific to that element
  - Only a **`TextBox`** needs to track the cursor position for user input
- A class hierarchy is a great way to implement this

Silverlight element class hierarchy

- The Silverlight class hierarchy is quite complex
- Everything is based on the `FrameworkElement` class which contains the fundamental properties of all elements
- There are many other classes



Class Hierarchies

- Using a class hierarchy for Silverlight elements has many advantages
  - A particular set of element data only needs to be stored once, at the appropriate position in the hierarchy
  - The display drawing system can treat them all display elements in exactly the same way
  - We can add new Silverlight elements that extend the behaviour of existing ones

Silverlight and Code

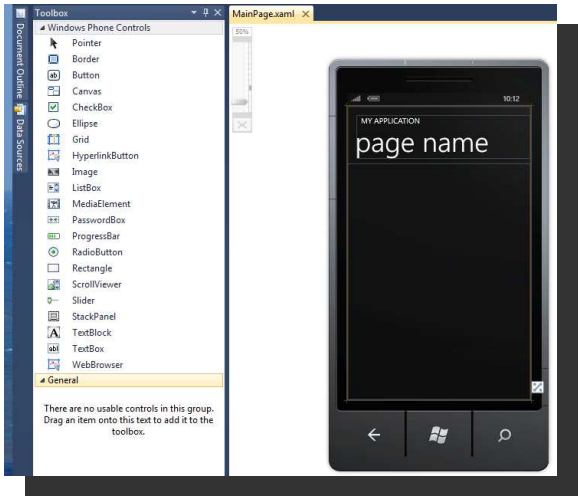
- This means that to update the items on a display our program will be calling methods or updating properties on the appropriate software object
- The next time that object is drawn the Silverlight rendering system will use the new data values and the appearance of the object will change to match

Silverlight and Design

- When we design a user interface we set values to give the elements position and size
- We do not do this from a program that we write (although we could)
- Instead we are going to use the design tools in Visual Studio to set up the elements
- We are going to start by using the element toolbox and the design surface

The Toolbox and Designer

- Visual Studio provides a menu of Silverlight elements that can be placed on the display page
- You can do this by simply dragging the elements from the toolbox onto the page



Demo

Demo 1: Adding display elements

Silverlight element names

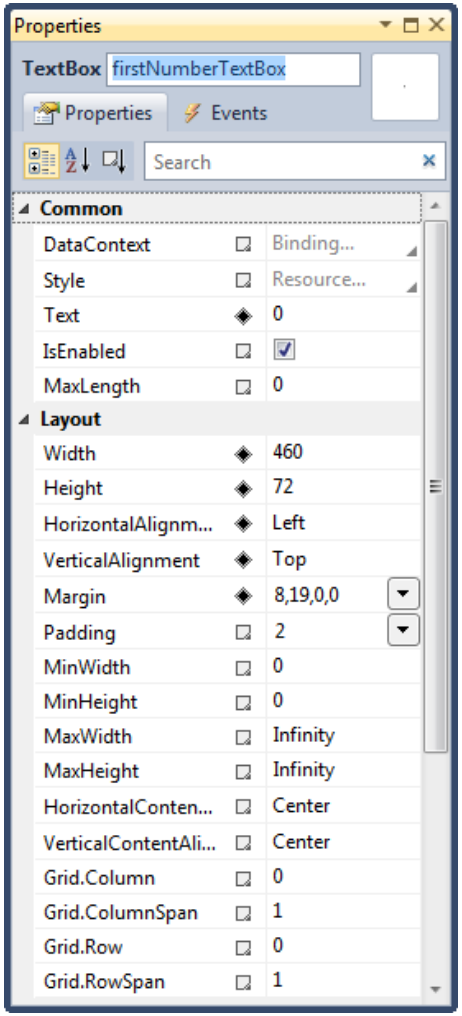
- When you drag an element onto the page Visual Studio creates a new variable in your program
- These are declared in a special area of the project that Visual Studio looks after for you
  - And you should not fiddle with this or bad things will happen
- However, you should always make sure that your variables have sensible names

Setting the name of an element

- The default name of an element is not useful
  - `TextBox1`, `Button1` etc
- We want to have more meaningful names
  - `firstsNumberTextbox`, `EqualsButton` etc
- We can use the Properties window in Visual Studio to update the name of an element in our program

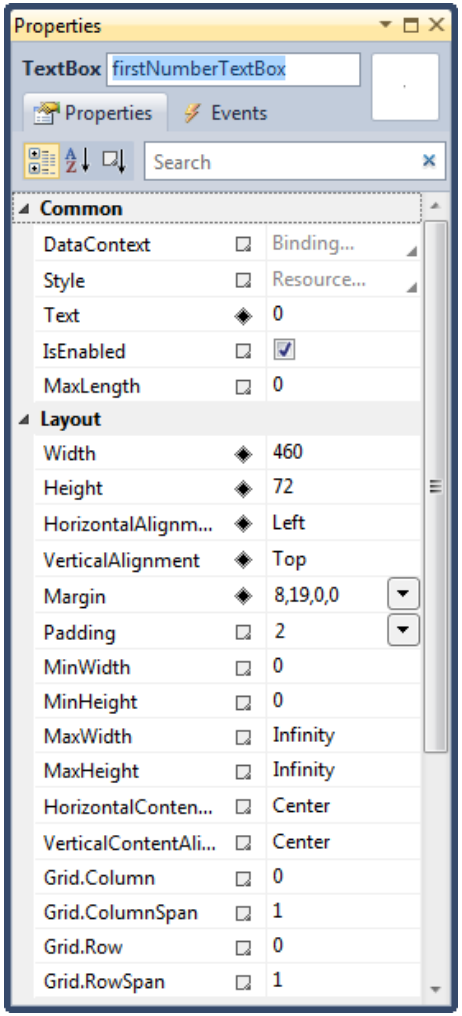
Selecting an element

- The name of an element is right at the top of the properties pane in Visual Studio
- To display the properties pane for an object, click on the object in the design surface
- The pane will usually be located at the bottom right of the Visual Studio display



Editing the name of an element

- You can simply type in the new name for your element here
- Make sure that it is a valid C# variable name, or Visual Studio will complain
- When you navigate away from this textbox the name will change



Demo

Demo 2: Display element names

Silverlight element properties

- We have seen that Visual Studio lets us manipulate the properties of the display elements
  - We can drag them around the display page
  - We can set values using the property pane
- Now we are going to consider how the properties actually from the perspective of C#
- To do this we have to revisit how properties work in C# classes

Properties in C#

- A program is made up of classes
- A class can contain data and behaviours
- The data is held in member variables
- The behaviours are provided by member methods
- When you design an application you decide what classes are required and the data and behaviours they have

Private and Public

- Data in a class is usually made `private`
- Methods are usually `public`
- You call methods to make changes to the data inside the class
- This means that all changes to data in the class is controlled
  - Programmers love having control.....

Managing Class Data

- We might decide to create a class called `Account` to store information in a bank
- The `Account` class will have lots of data items that are stored about each customer
- Each of these items will map onto a data member of the class
- The programmer must then provide methods to get and set the values of the items

Bank accounts and data

- Perhaps our bank needs to store the age of the account holder
  - Perhaps they are concerned about allowing very young people to draw out too much cash
- This information must be stored in the account
- It will be a data member within it

NameProperty

```
public class Account
{
    private int age;
    /// rest of account here
}
```

- The age value is a `private` member of the `Account` class
- To work with the age value we have to provide methods that are public

Using Get and Set methods

```
public class Account
{
    private int age;
    public int GetAge()
    {
        return this.age;
    }
    public void SetAge( int inAge )
    {
        if ( (inAge > 0) && (inAge < 120) )
        {
            this.age = inAge;
        }
    }
}
```

Managing the Age

```
CurrentAccount a = new Account();
a.SetAge(21);
```

- When we want to control the age property we can call the **SetAge** method
- We can use the **GetAge** method to read the age of the account holder

Get and Set Methods

- Creating **`Get`** and **`Set`** methods is a standard programming pattern
- Frequently they will perform validation of incoming data and reject out of range values
- I have not added any validation to the set method yet
  - Nobody with an age less than 8 or greater than 100 can have an account

Using Properties

- The C# language provides *properties* to make getting and setting data easier
- They do not make things possible that we couldn't do before
- They must make the programs easier to write
  - Properties are used extensively in Silverlight for display elements

Age Property

```
public class Account
{
    private int ageValue;
    public int Age
    {
        set
        {
            if ( (value > 8) && (value < 100) )
                ageValue = value;
        }
        get
        {
            return ageValue;
        }
    }
}
```

Property Keywords

- The block of code after the `get` keyword is obeyed when the property is read by a program
  - It must return a value of the property type
- The block of code after the `set` keyword is obeyed when the property is written
  - Within the set block the keyword `value` means "The value being assigned to the property"

Using the Age Property

```
Account s = new Account ();
s.Age = 21;
Console.WriteLine ( "Age is : " + s.Age );
```

- When a program uses a property it looks just like direct access to a data member
- The difference is that the get or set behaviours are used the blocks of code in the property are used
- This makes the code much tidier

Property validation

```
Account s = new Account ();
int newAge = 150;
s.Age = newAge;
if (s.Age != newAge )
    Console.WriteLine ( "Age was not set" );
```

- If the property uses validation it is up to the user of the property to make sure that an assignment worked correctly

Multiple properties for one value

```
public int AgeInMonths
{
    get
    {
        return this.ageValue*12;
    }
}
```

- This property only has a `get` method
- It provides a way of reading the age in months

Properties and Silverlight

- The items that we see in the properties pane for an element are all C# properties
- When an assigning to an element property a program runs code inside a C# set behaviour
- This might seem inefficient
  - Accessing the data directly would be quicker
- But it is actually a necessary part of how Silverlight runs

Properties and Notification

- The `set` method in a Silverlight property does a lot more than just update the stored data
- This action must also trigger a display update for the text on the screen
- Setting a property in a program also sends a notification to the program that the property is being changed
- This idea of using property changes to trigger events is one we will return to

Page Design with Silverlight

- We can design all our pages by dragging items onto the page, giving them a name and then setting their properties
- Visual Studio will create all the code behind the program to make this work
- You can set many properties of element s including colour, texture, transparency and even animation
- But this is best left to Graphic Designers

Review

- Visual Studio provides a complete design tool for creating and managing user interfaces
- A Silverlight display is made up of elements
- Each element is implemented by a C# class which is part of the element hierarchy
- Silverlight elements expose properties which allow programs (and Visual Studio) to determine their location and appearance on the display page