

ICT365

# Software Development Frameworks

Dr Afaq Shah



**Murdoch**  
UNIVERSITY

# XAML



**Murdoch**  
UNIVERSITY

# Aims



Introduction to XAML

XAML Setup

XAML Namespace

XAML Building Block

Data Binding

Markup Extensions

# What is XAML

- Short for Extensible Application Markup Language, and pronounced “zammel”
- Markup (declarative) language used to instantiate .NET objects
- XAML, like all XML-based languages, is *case-sensitive*. That means you can't substitute `<button>` for `<Button>`
- can be applied to many different problem domains
- Visual Studio or Microsoft Expression Blend can be used to generate the XAML

| XAML Code   | C# Code   |
|---|---|
| <code>&lt;Button x:Name="button1" <u>IsEnabled="True"</u>/&gt;</code> | <code>Button button1=new Button();<br/>button1.IsEnabled=true;</code> |

# XAML Overview

XAML: eXtensible Application Markup Language

XML-based declarative language for UI

- Each XML element maps to an object instance

- Each attribute maps to a property of object

- Event handlers can be declared

  - Still the handlers should be implemented in the code-behind

- ... more features (will be explained shortly)

# Tools for Editing XAML

## Visual Studio

Mainly for programmers

Most of the XAML editing features are provided

Can program application logic

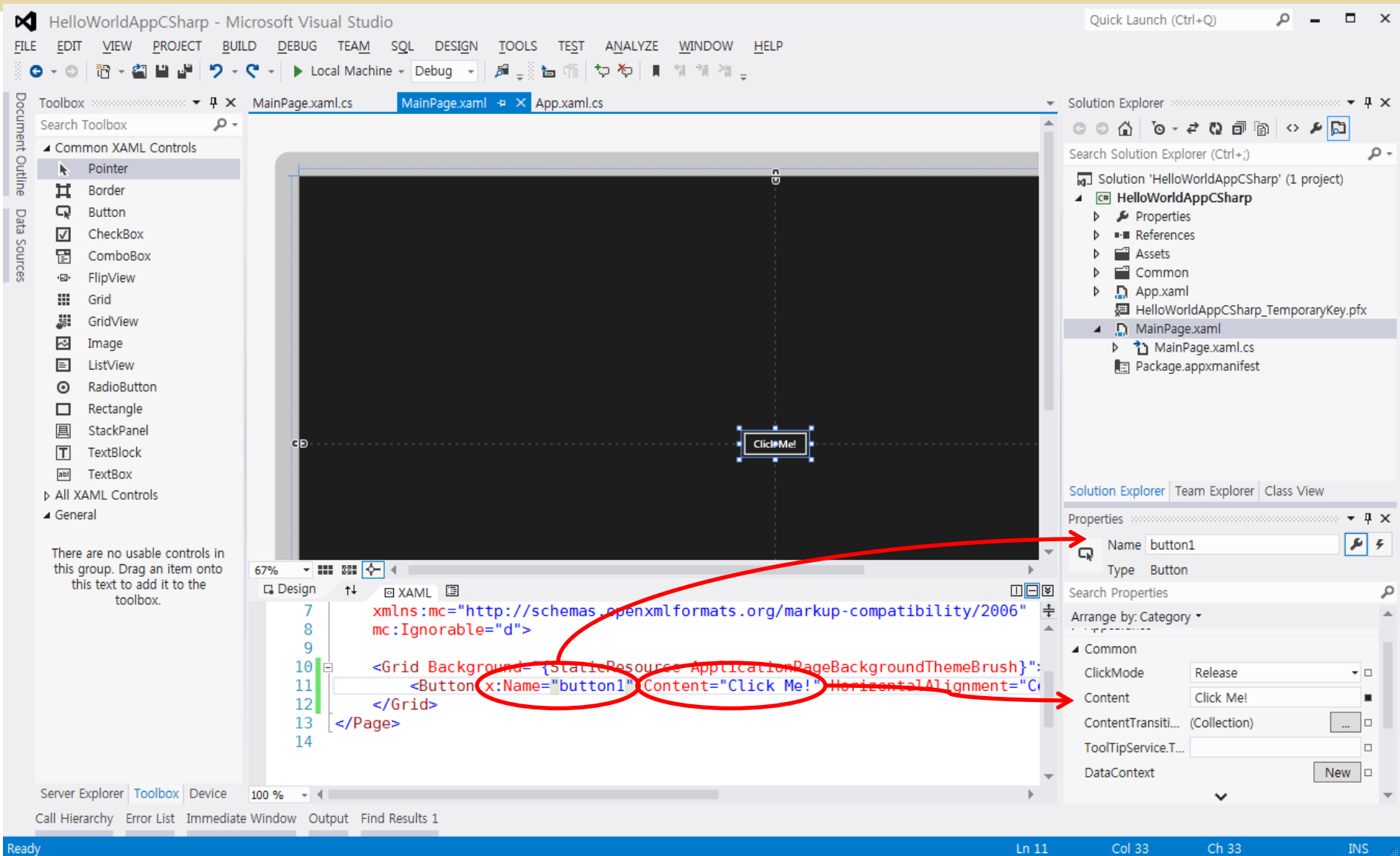
## Blend for VS

Mainly for UI designers

Visual states can be seen/edited without compiling

Can create complex animations

# XAML Editor in Visual Studio



The screenshot displays the Visual Studio IDE with the XAML Editor open. The design view shows a button with the text "Click Me!". The source view shows the following XAML code:

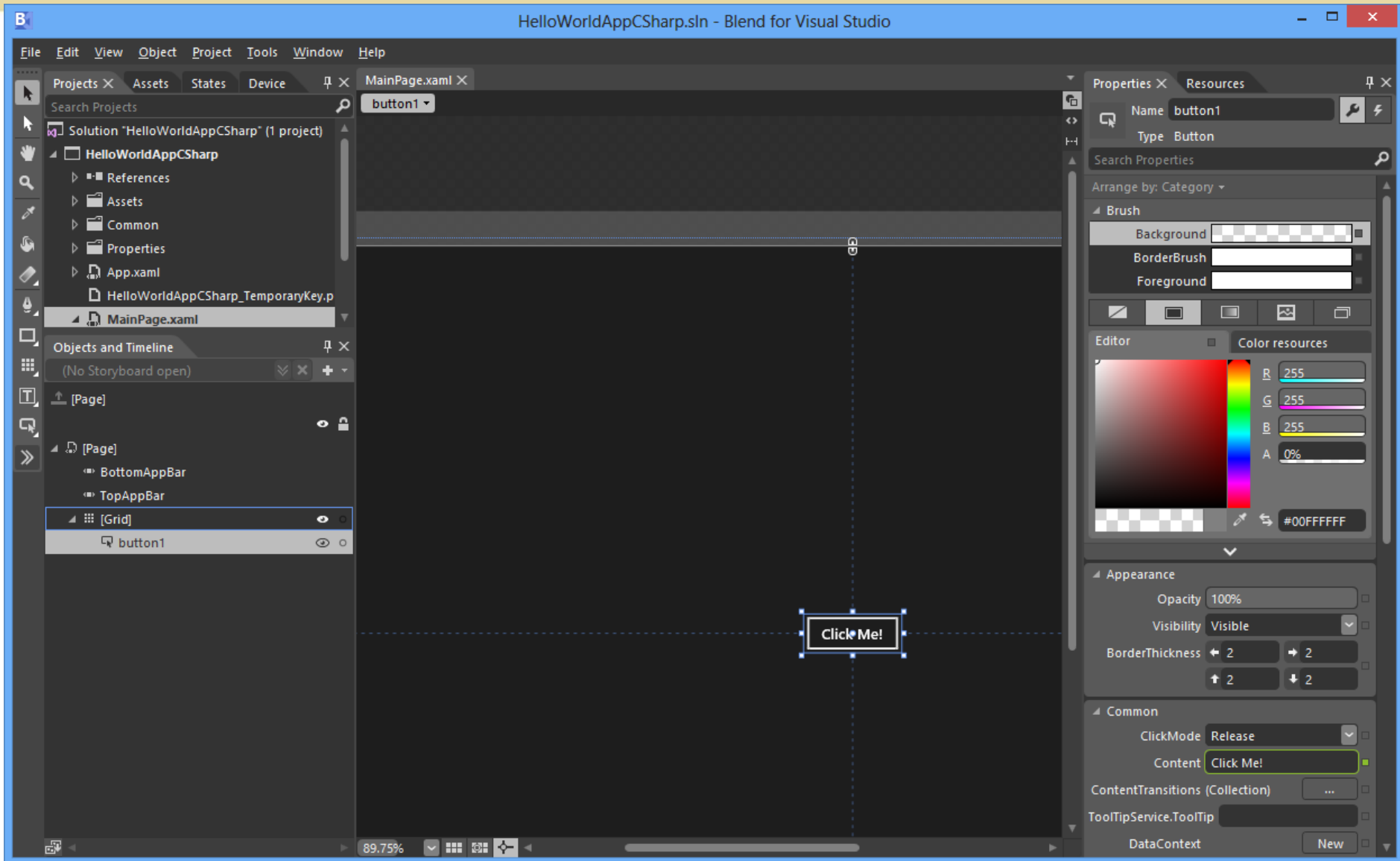
```
7 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8 mc:Ignorable="d">
9
10 <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
11 <Button x:Name="button1" Content="Click Me!" HorizontalAlignment="C
12 </Grid>
13 </Page>
14
```

The XAML code is highlighted in the source view, and the corresponding button is highlighted in the design view. Red circles and arrows indicate the mapping between the XAML code and the design view. The button's name is "button1" and its content is "Click Me!".

Properties window (Properties):

- Name: button1
- Type: Button
- ClickMode: Release
- Content: Click Me!
- ContentTransiti... (Collection)
- ToolTipService.T...
- DataContext: New

# Blend for Visual Studio





# Advantages of XAML

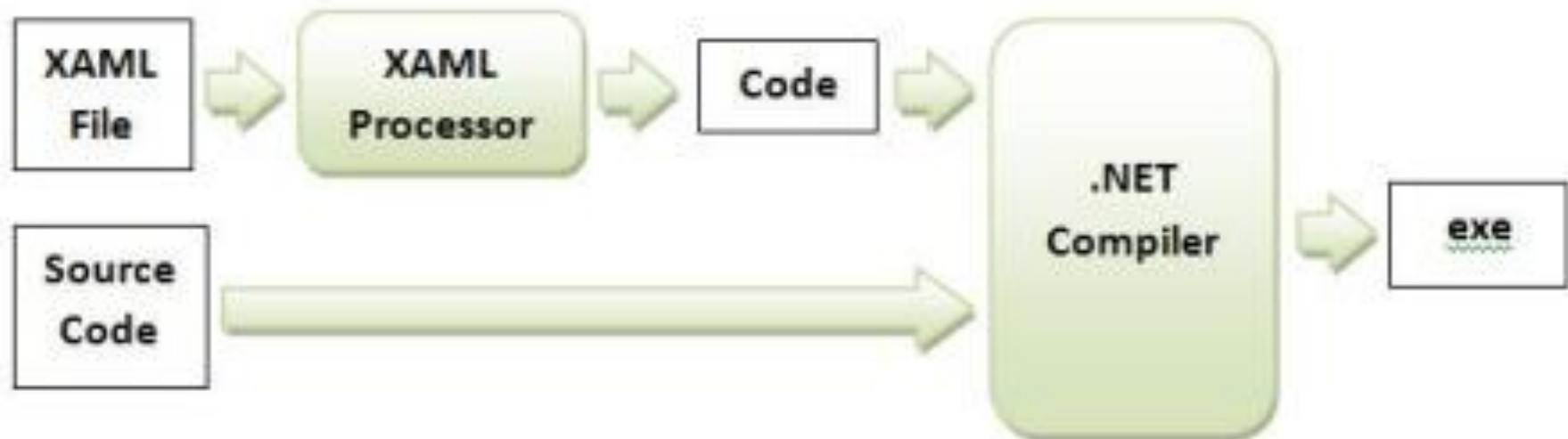
- XAML code is short and clear to read
- Separation of designer code and logic
- Graphical design tools like Expression Blend require XAML as source.
- The separation of XAML and UI logic allows it to clearly separate the roles of designer and developer

# The Variants of XAML

- **WPF XAML** encompasses the elements that describe WPF content, such as vector graphics, controls, and documents.
- **XPS XAML (XML Paper Specification)** is the part of WPF XAML that defines an XML representation for formatted electronic documents.
- **Silverlight XAML** is a subset of WPF XAML that's intended for browser applications.
- **WF XAML** encompasses the elements that describe Windows Workflow Foundation (WF) content.

# XAML Compilation

- When you compile a WPF application in Visual Studio, all your XAML files are converted into BAML and that BAML is then embedded as a resource into the final DLL or EXE assembly

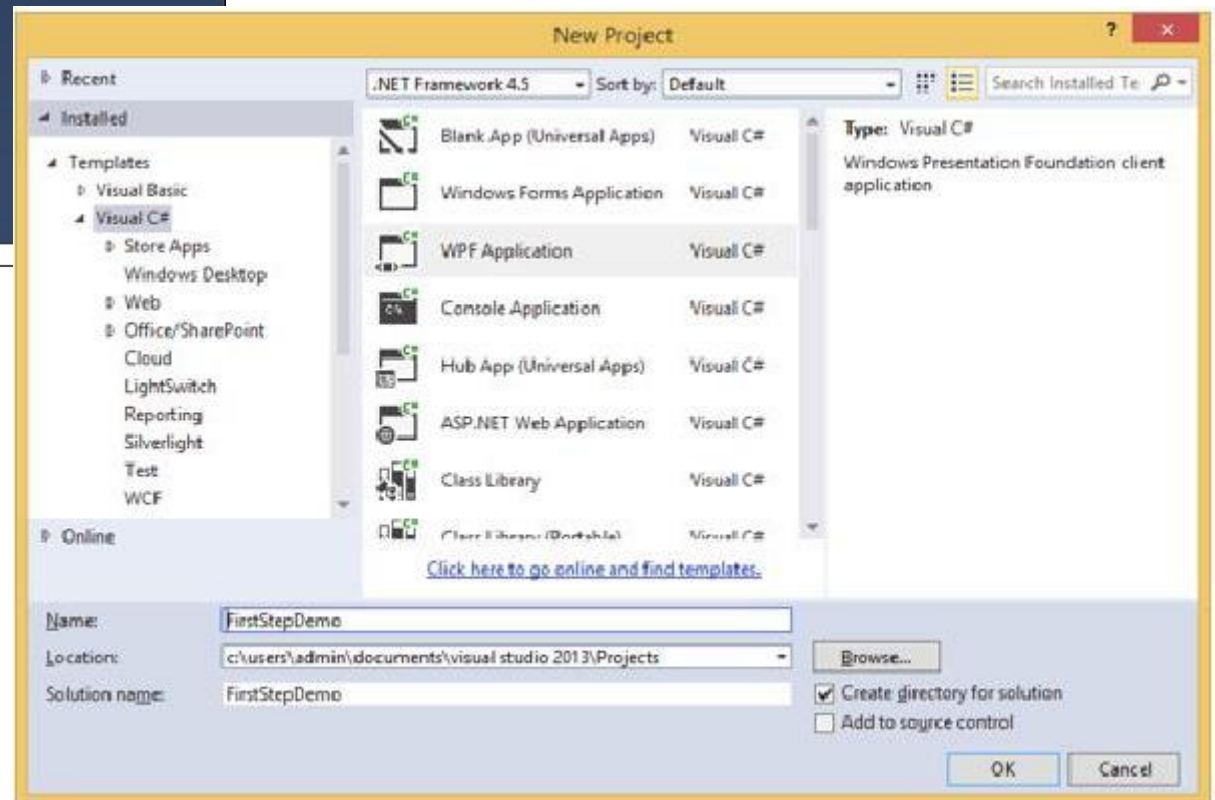
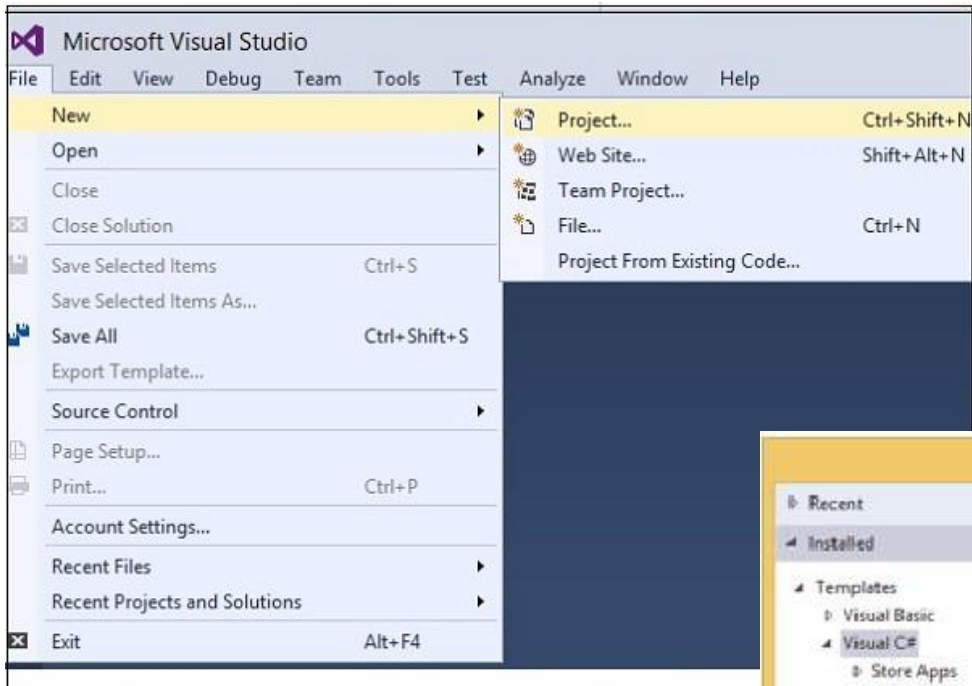


- A XAML file can be compiled into a Binary Application Markup Language file having the .BAML extension.), which may be inserted as a resource into a .NET Framework assembly.
- At run-time, the framework engine extracts the .BAML file from assembly resources, parses it, and creates a corresponding WPF visual tree or workflow. Having this format, the content is faster loadable during runtime.

# BAML Resources

- **Compiled XAML = BAML not IL**
- <http://blogs.microsoft.co.il/tomershamam/2007/05/25/compiled-xaml-baml-not-il/>
- **How to load BAML resources**
- <http://www.wpftutorial.net/baml.html>

# XAML Setup



# XAML Setup

```
<Window x:Class = "FirstStepDemo.MainWindow"  
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"  
  xmlns:d = "http://schemas.microsoft.com/expression/blend/2008"  
  xmlns:mc = "http://schemas.openxmlformats.org/markup-compatibility/2006"  
  xmlns:local = "clr-namespace:FirstStepDemo"  
  mc:Ignorable = "d" Title = "MainWindow" Height = "300" Width = "300">  
  <Grid>  
  </Grid>  
</Window>
```

- The top-level Window element, which represents the entire window
- The Grid, in which you can place all your controls
- For the Window element you'll find several interesting attributes, including a class name and two XML namespaces
- Also find the three properties which tells
- WPF to create a window with the caption MainWindow and to make it 300 by 300 units large

# XAML Setup

```
<Window x:Class = "FirstStepDemo.MainWindow"
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d = "http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc = "http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local = "clr-namespace:FirstStepDemo"
  mc:Ignorable = "d" Title = "MainWindow" Height = "350" Width = "604">

  <Grid>
    <Button Content = "First Button" Height = "30" Width = "80"/>
    <TextBlock Text = "Congratulations you have successfully build your first app"
      Height = "30" Margin = "162,180,122,109"/>
  </Grid>

</Window>
```



# XAML Setup



# XAML Namespaces

- To figure out which class you really want, the XAML parser examines the XML namespace that's applied to the element.
- The xmlns attribute is a specialized attribute in the world of XML that's reserved for declaring namespaces

```
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

# XAML Namespaces (cont.)

- In the example, two namespaces are defined:  

```
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```
- <http://schemas.microsoft.com/winfx/2006/xaml/presentation> is the core WPF namespace  
this namespace is declared without a namespace prefix, so every element is automatically placed in this namespace
- <http://schemas.microsoft.com/winfx/2006/xaml> is the XAML namespace.  
This namespace is mapped to the prefix *x*. *That means you can apply it by placing the namespace prefix before the element name (as in <x:ElementName>).*

# Using Other Namespaces

- You need to map the .NET namespace to an XML namespace.
- XAML has a special syntax for doing this

```
xmlns:Prefix="clr-namespace:Namespace;assembly=AssemblyName"
```

- Prefix is the XML prefix you want to use to indicate that namespace in your XAML markup. For example, the XAML language uses the x: prefix.
- Namespace is the fully qualified .NET namespace name.
- AssemblyName is the assembly where the type is declared, without the .dll extension. This assembly must be referenced in your project.

# Other Namespaces: Examples

- Here's how you would gain access to the basic types in the System namespace and map them to the prefix sys:

```
xmlns:sys="clr-namespace:System;assembly=mscorlib"
```

- Here's how you would gain access to the types you've declared in the MyProject namespace of the current project and map them to the prefix local:

```
xmlns:local="clr-namespace:MyNamespace"
```

- Now, to create an instance of a class in one of these namespaces, you use the namespace prefix:

```
<local:MyObject ...></local:MyObject>
```

# XAML vs C# Code

```
<Window x:Class = "XAMLVsCode.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">
```

```
<StackPanel>
```

```
  <TextBlock Text = "Welcome to XAML Tutorial" Height = "20" Width =  
"200" Margin = "5"/>
```

```
  <Button Content = "Ok" Height = "20" Width = "60" Margin = "5"/>
```

```
</StackPanel>
```

```
</Window>
```

# XAML vs C# Code



# XAML vs C# Code



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

namespace XAMLVsCode {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>

    public partial class MainWindow : Window {
        public MainWindow() {
            InitializeComponent();

            // Create the StackPanel
            StackPanel stackPanel = new StackPanel();
            this.Content = stackPanel;
        }

        // Create the TextBox
        TextBox textBox = new TextBox();
        textBox.Text = "Welcome to XAML Tutorial";
        textBox.Height = 20;
        textBox.Width = 200;
        textBox.Margin = new Thickness(5);
        stackPanel.Children.Add(textBox);

        // Create the Button
        Button button = new Button();
        button.Content = "OK";
        button.Height = 20;
        button.Width = 50;
        button.Margin = new Thickness(20);
        stackPanel.Children.Add(button);
    }
}
```



# XAML vs C# Code



# XML Building Blocks

- **Objects (create and initialize)**
- XAML is a typically declarative language which can create and instantiate objects. It is another way to describe objects based on XML, i.e., which objects need to be created and how they should be initialized before the execution of a program. Objects can be:
  - Containers (Stack Panel, Dock Panel)
  - UI Elements / Controls (Button, TextBox, etc.)
  - Resource Dictionaries

# XML Building Blocks

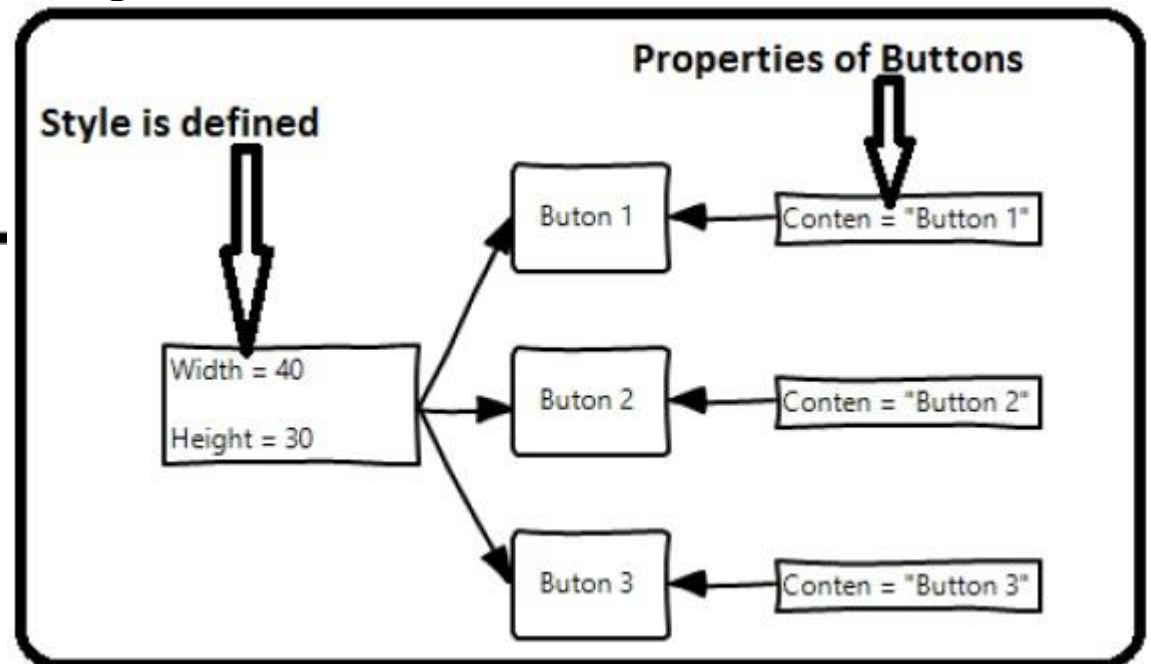
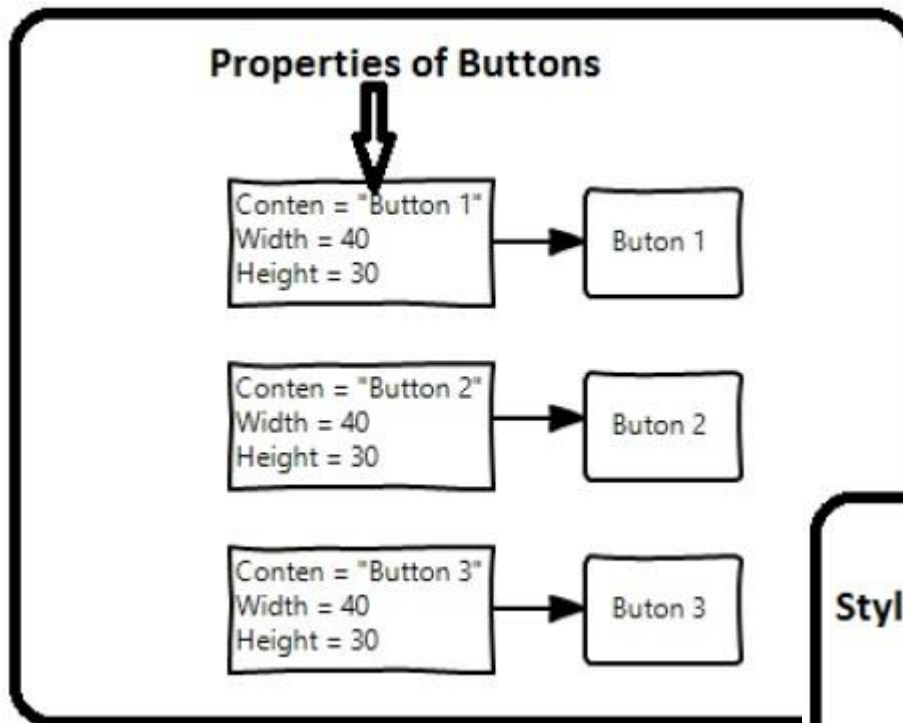
- Objects can be modified easily by using resources, styles and templates
- **Resources**

Resources are normally definitions connected with some object that you just anticipate to use more often than once. It is the ability to store data locally for controls or for the current window or globally for the entire applications.

# XML Building Blocks

- **Styles**
- XAML framework provides several strategies to personalize and customize the appearance of an application. Styles give us the flexibility to set some properties of an object and reuse these specific settings across multiple objects for a consistent look.
- In styles, you can set only the existing properties of an object such as Height, Width, Font size, etc.
- Only the default behavior of a control can be specified.
- Multiple properties can be added into a style.

# XML Building Blocks

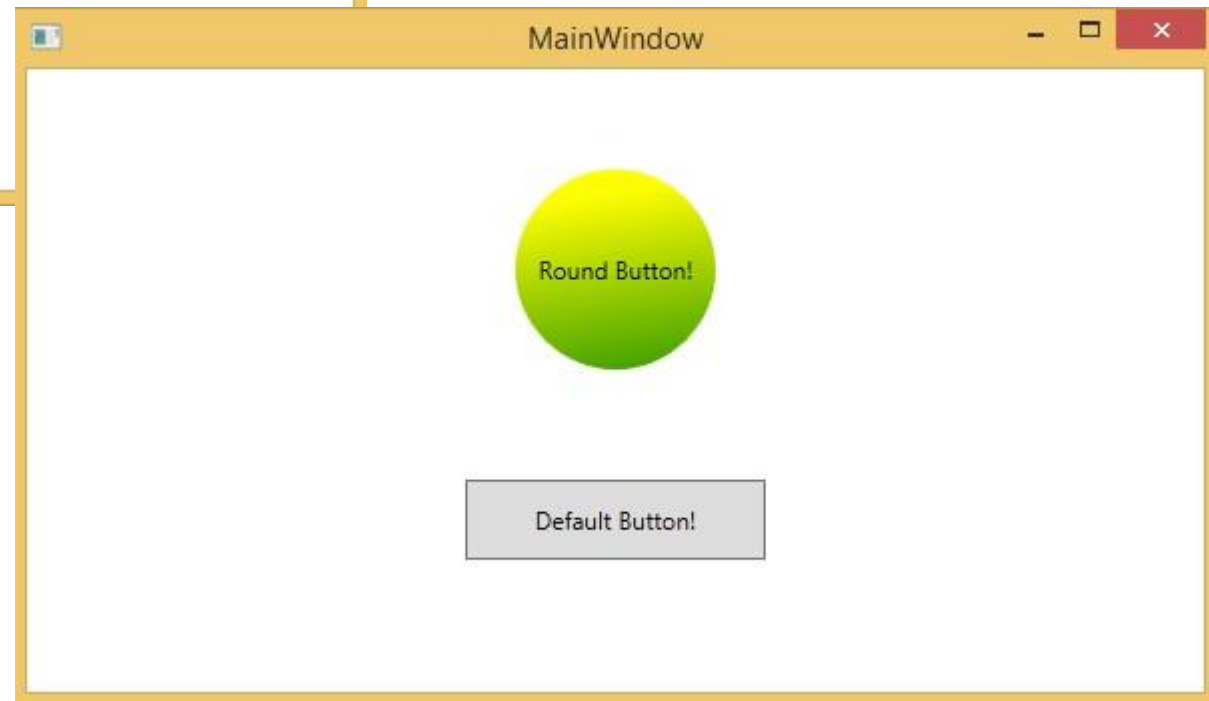
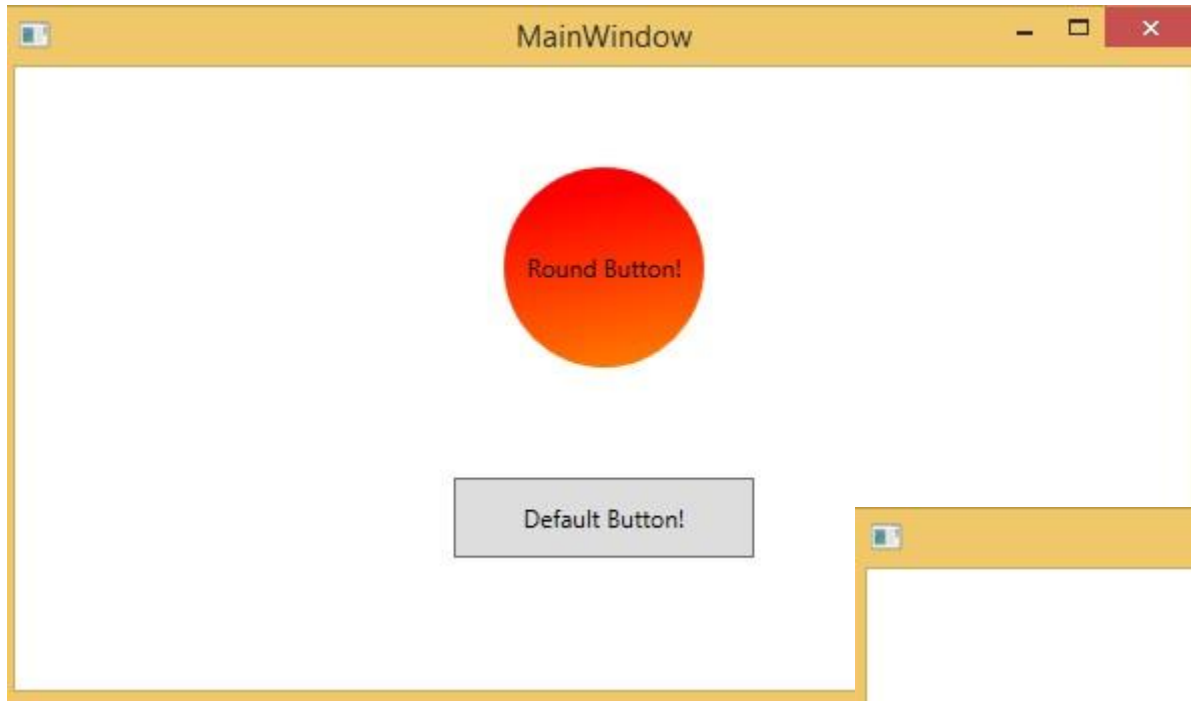


# XML Building Blocks

- **Templates**

A template describes the overall look and visual appearance of a control. For each control, there is a default template associated with it which gives the appearance to that control. In XAML, you can easily create your own templates when you want to customize the visual behavior and visual appearance of a control.

# XML Building Blocks



# Controls?

| Sr.No. | Controls & Description  |
|--------|---|
| 1      | <a href="#">Button</a> A control that responds to user input.   |
| 2      | <a href="#">Calendar</a> Represents a control that enables a user to select a date by using a visual calendar display.  |
| 3      | <a href="#">CheckBox</a> A control that a user can select or clear.   |
| 4      | <a href="#">ComboBox</a> A drop-down list of items a user can select from.  |
| 5      | <a href="#">ContextMenu</a> Gets or sets the context menu element that should appear whenever the context menu is requested through a user interface (UI) from within this element. |
| 6      | <a href="#">DataGrid</a> Represents a control that displays data in a customizable grid.  |
| 7      | <a href="#">DatePicker</a> A control that lets a user select a date.  |
| 8      | <a href="#">Dialogs</a> An application may also display additional windows to the user to gather or display important information.  |

|    |   |
|----|---|
| 9  | <a href="#">GridView</a> A control that presents a collection of items in rows and columns that can scroll horizontally.                                  |
| 10 | <a href="#">Image</a> A control that presents an image.   |
| 11 | <a href="#">ListBox</a> A control that presents an inline list of items that the user can select from.  |
| 12 | <a href="#">Menus</a> Represents a Windows menu control that enables you to hierarchically organize elements associated with commands and event handlers. |
| 13 | <a href="#">PasswordBox</a> A control for entering passwords.   |
| 14 | <a href="#">Popup</a> Displays content on top of existing content, within the bounds of the application window.   |
| 15 | <a href="#">ProgressBar</a> A control that indicates progress by displaying a bar.  |
| 16 | <a href="#">ProgressRing</a> A control that indicates indeterminate progress by displaying a ring.  |

|    |   |
|----|---|
| 17 | <a href="#">RadioButton</a> A control that allows a user to select a single option from a group of options.                               |
| 18 | <a href="#">RichEditBox</a> A control that lets a user edit rich text documents with content like formatted text, hyperlinks, and images. |
| 19 | <a href="#">ScrollViewer</a> A container control that lets the user pan and zoom its content.   |
| 20 | <a href="#">SearchBox</a> A control that lets a user enter search queries.  |
| 21 | <a href="#">Slider</a> A control that lets the user select from a range of values by moving a Thumb control along a track.                |
| 22 | <a href="#">TextBlock</a> A control that displays text.   |
| 23 | <a href="#">TimePicker</a> A control that lets a user set a time value.   |
| 24 | <a href="#">ToggleButton</a> A button that can be toggled between 2 states.   |



# Data Binding

- Data binding is a mechanism in XAML applications that provides a simple and easy way for Windows Runtime Apps using partial classes to display and interact with data.
- The management of data is entirely separated from the way the data is displayed in this mechanism.
- Data binding allows the flow of data between UI elements and data object on user interface.
- When a binding is established and the data or your business model changes, then it will reflect the updates automatically to the UI elements and vice versa. It is also possible to bind, not to a standard data source, but rather to another element on the page. Data binding can be of two types –
  - One-way data binding
  - Two-way data binding

# Data Binding

## One-Way Data Binding

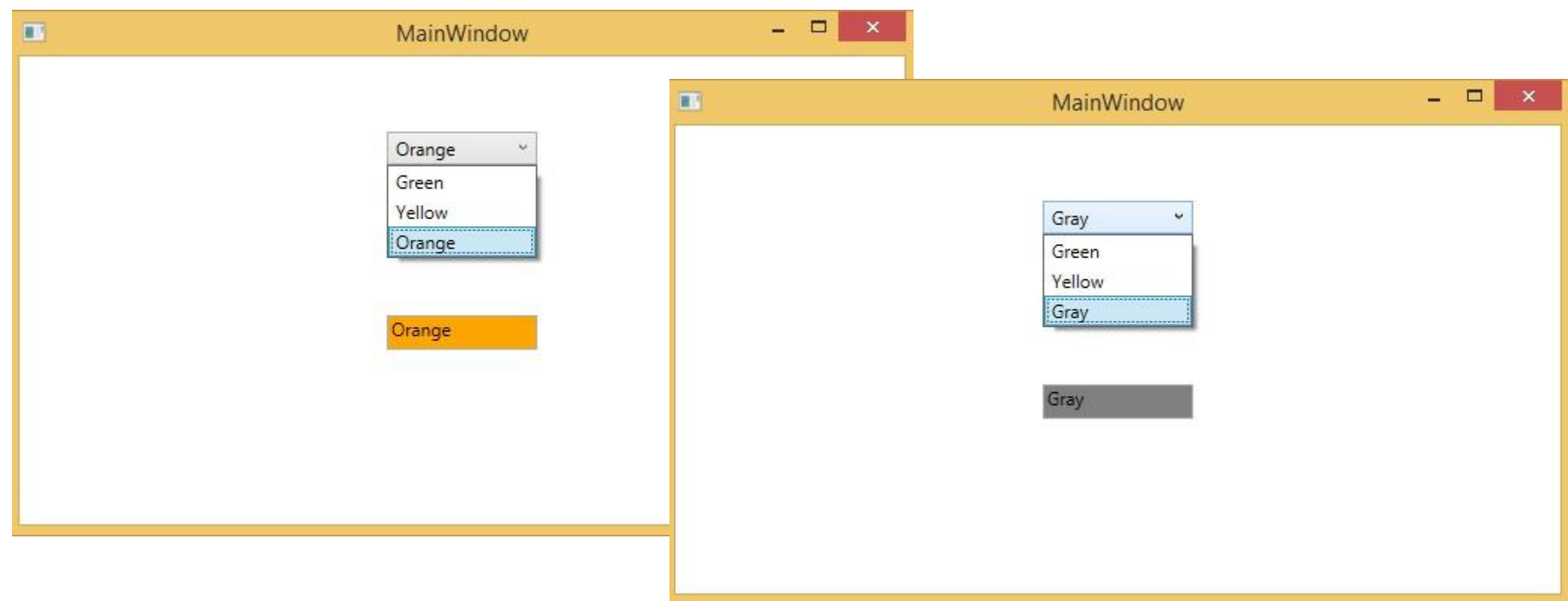
In one-way binding, data is bound from its source (that is the object that holds the data) to its target (that is the object that displays the data).



# Data Binding

## Two-Way Data Binding

In two-way binding, the user can modify the data through the user interface and have that data updated in the source. If the source changes while the user is looking at the view, you would want to update the view.



# Markup Extensions

- Markup extensions are a method/technique to gain a value that is neither a specific XAML object nor a primitive type
- Markup extensions can be defined by opening and closing curly braces and inside that curly braces, the scope of the markup extension is defined.
- **Data binding** and static **resources** are markup extensions.
- There are some predefined XAML markup extensions in **System.xaml** which can be used.

# Markup Extensions (cont.)

- `<Window x:Class = "XAMLStaticResourcesMarkupExtension.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">`
- `<Window.Resources>`
- `<SolidColorBrush Color = "Blue" x:Key = "myBrush"></SolidColorBrush>`
- `</Window.Resources>`
- `<Grid>`
- `<StackPanel Orientation = "Vertical">`
- `<TextBlock Foreground = "{StaticResource myBrush}" Text = "First Name"`
- `Width = "100" Margin = "10" />`
- `<TextBlock Foreground = "{StaticResource myBrush}" Text = "Last Name"`
- `Width = "100" Margin = "10" />`
- `</StackPanel>`
- `</Grid>`
- `</Window>`

In **Window.Resources**, you can see **x:Key** is used which uniquely identifies the elements that are created and referenced in an XAML defined dictionary to identify a resource in a resource dictionary.

# Markup Extensions (cont.)



# Markup Extensions (cont.)

```
<Window x:Class = "XAMLMarkupExtension.MainWindow"  
  xmlns =  
  "http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"  
  xmlns:my = "clr-namespace:XAMLMarkupExtension"  
  Title = "MainWindow" Height = "350" Width = "525">
```

```
<Grid>
```

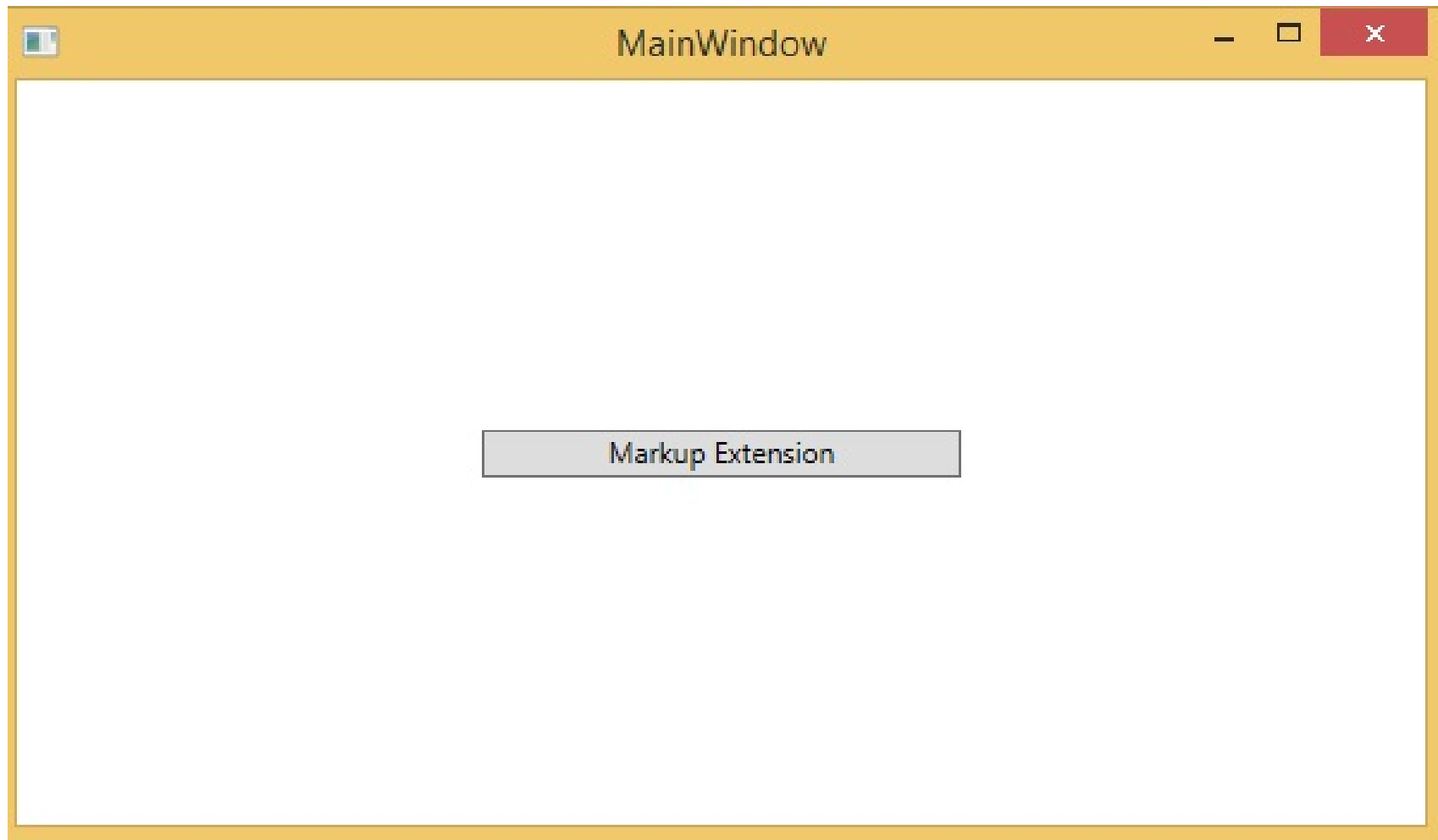
```
  <Button Content = "{my:MyMarkupExtension FirstStr =  
Markup, SecondStr = Extension}"
```

```
    Width = "200" Height = "20" />
```

```
</Grid>
```

```
</Window>
```

# Markup Extensions (cont.)





# Special Characters

- By the rules of XML, XAML pays special attention to a few specific characters, such as & and < and >.

| <b>Special Character</b> | <b>Character Entity</b> |
|--------------------------|-------------------------|
| Less than (<)            | &lt;                    |
| Greater than (>)         | &gt;                    |
| Ampersand (&)            | &amp;                   |
| Quotation mark (")       | &quot;                  |

# Special Characters (cont.)

```
<Button ... >  
    &lt;Click Me&gt;  
</Button>
```

- When the XAML parser reads this, it correctly understands that you want to add the text <Click Me> and it passes a string with this content, complete with angled brackets, to the Button.Content property.
- This limitation is a XAML detail and it won't affect you if you want to set the Button.Content property in code

# Shapes in XAML

Unlike many other declarative UI languages, non-widget shapes are supported in XAML

## Classes

The **Windows.UI.Xaml.Shapes** namespace has these classes.

| Class            | Description  |
|------------------|--|
| <b>Ellipse</b>   | Draws an ellipse.  |
| <b>Line</b>      | Draws a straight line between two points.  |
| <b>Path</b>      | Draws a series of connected lines and curves. The line and curve dimensions are declared through the Data property, and can be specified either with a path-specific mini-language, or with an object model. |
| <b>Polygon</b>   | Draws a polygon, which is a connected series of lines that form a closed shape.  |
| <b>Polyline</b>  | Draws a series of connected straight lines.  |
| <b>Rectangle</b> | Draws a rectangle shape, which can have a stroke and a fill.   |
| <b>Shape</b>     | Provides a base class for shape elements, such as Ellipse, Polygon, and Rectangle.   |

# Events

- In XAML, all of the controls expose some events so that they can be subscribed for specific purposes.
- Whenever an event takes place, the application will be notified and the program can react to them, e.g., close buttons are used to close a dialog.
  - Click
  - MouseDown
  - MouseEnter
  - MouseLeave
  - MouseUp
  - KeyDown
  - KeyUp

# Events (cont.)

```
<Window x:Class = "XAMLEventHandling.MainWindow"  
  xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"  
  Title = "MainWindow" Height = "350" Width = "604">  
  
  <Grid>  
    <Button x:Name = "button1" Content = "Click" Click = "OnClick"  
      Width = "150" Height = "30" HorizontalAlignment = "Center" />  
  </Grid>  
  
</Window>
```

# Events (cont.)

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

namespace XAMLEventHandling {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>

    public partial class MainWindow : Window {
        public MainWindow() {
            InitializeComponent();
        }
        private void OnClick(object sender, RoutedEventArgs e) {
            MessageBox.Show("Button is clicked!");
        }
    }
}
```

# Events (cont.)





**Murdoch**  
UNIVERSITY

# Examples



# StackPanel example

```
<Window x:Class="Demo4.Content.Window1"
xmlns="http://schemas.microsoft.com/winfx/avalon/2005"
xmlns:x="http://schemas.microsoft.com/winfx/xaml/2005"
Title="Demo4.Content">
  <StackPanel Orientation="Vertical">
    <Button Name="button1">Just text</Button>
    <Button Name="button2">
      <Image Source="banner.jpg" Name="image1" Width="100"/>
    </Button>
    <Button Name="button3">
      <StackPanel Orientation="Vertical">
        <TextBlock>Just text<LineBreak/>The next line</TextBlock>
        <Image Source="banner.jpg" Name="image1" Width="100"/>
      </StackPanel>
    </Button>
  </StackPanel>
</Window>
```

# Grid example

```
<Window x:Class="Demo4.Content.Window1"
xmlns="http://schemas.microsoft.com/winfx/avalon/2005"
xmlns:x="http://schemas.microsoft.com/winfx/xaml/2005"
Title="Demo4.Content">
  <Grid ShowGridLines="True">
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition/>
      <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition/>
      <RowDefinition/>
      <RowDefinition/>
    </Grid.RowDefinitions>
    <TextBlock Grid.Column="0" Grid.Row="0">Top left</TextBlock>
    <TextBlock Grid.Column="1" Grid.Row="1">Middle</TextBlock>
    <TextBlock Grid.Column="2" Grid.Row="2">LRight</TextBlock>
  </Grid>
</Window>
```

# Databinding example

```
<StackPanel Name="pnlMain">
    <TextBlock>Name: </TextBlock>
    <TextBox Name="txtName"
        Text="{Binding Path=Name}" />
    <TextBlock>City:</TextBlock>
    <TextBox Name="txtCity"
        Text="{Binding Path=City}" />
    <StackPanel Orientation="Horizontal">
        <Button Name="btnPrevious"
            Click="btnPrevious_Click">&lt;</Button>
        <Button Name="btnNext"
            Click="btnNext_Click">&gt;</Button>
    </StackPanel>
    <ListBox Name="lstCustomers"
        IsSynchronizedWithCurrentItem="True"
        ItemsSource="{Binding}" />
</StackPanel>
```



# Hello World

- C# example

```
using System;
using System.Windows;

namespace AvalonExample {
    class MyApp {
        [STAThread]
        static void Main() {
            MessageBox.Show("Hello World!");
        }
    }
}
```



# Hello World

- XAML example

```
<Page
xmlns="http://schemas.microsoft.com/winfx/a
valon/2005"
xmlns:x="http://schemas.microsoft.com/winfx
/xaml/2005">

  <TextBlock>Hello World!</TextBlock>

</Page>
```

# Application Object

Application object acts as container for more complex applications

MainWindow

Application events like  
Startup & Shutdown

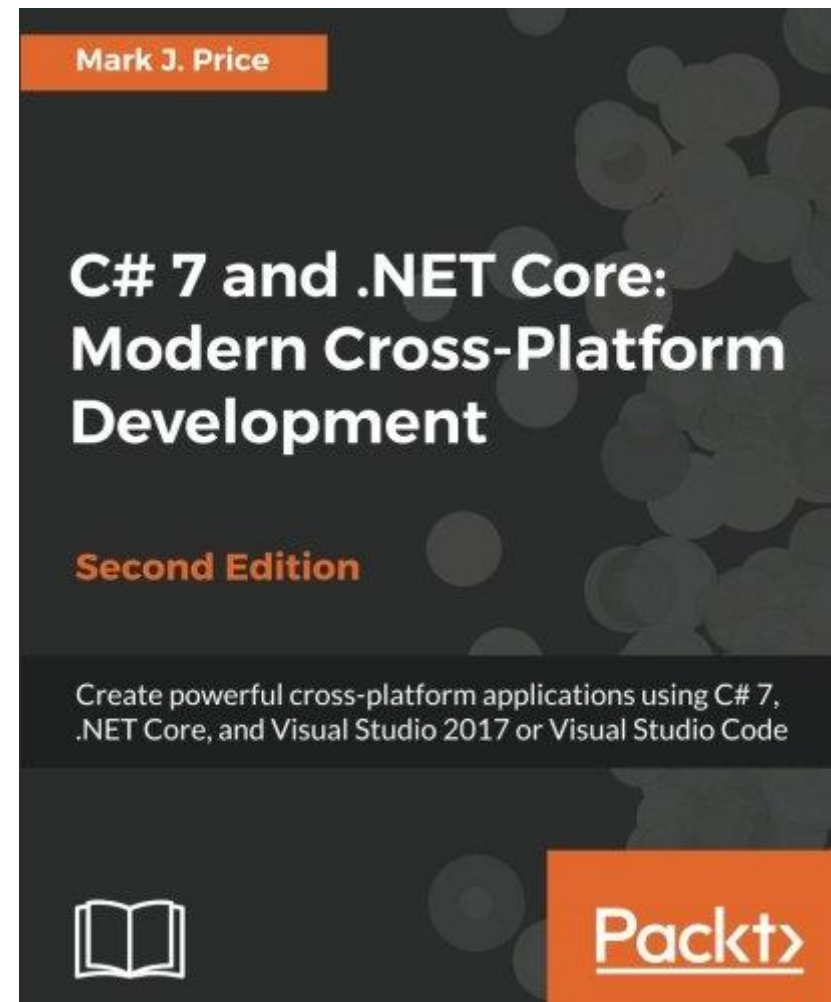
```
public class MyApp : Application
{
    [STAThread]
    static void Main(string[] args)
    {
        MyApp app = new MyApp();
        app.Startup += app.OnApplicationStartup;
        app.Run(args);
    }

    void OnApplicationStartup(object sender,
                             StartupEventArgs e)
    {
        Window w = new Window();
        w.Title = "Mark says: Hello World!";
        w.Show();
    }
}
```

# Reading/ reference

<http://prospero.murdoch.edu.au/record=b2962782~S1>

## Chapter 13. Building Universal Windows Platform Apps Using XAML



# Reading/ reference

<http://prospero.murdoch.edu.au/record=b2962780~S1>

Chapter: Introducing Windows  
Communication Foundation

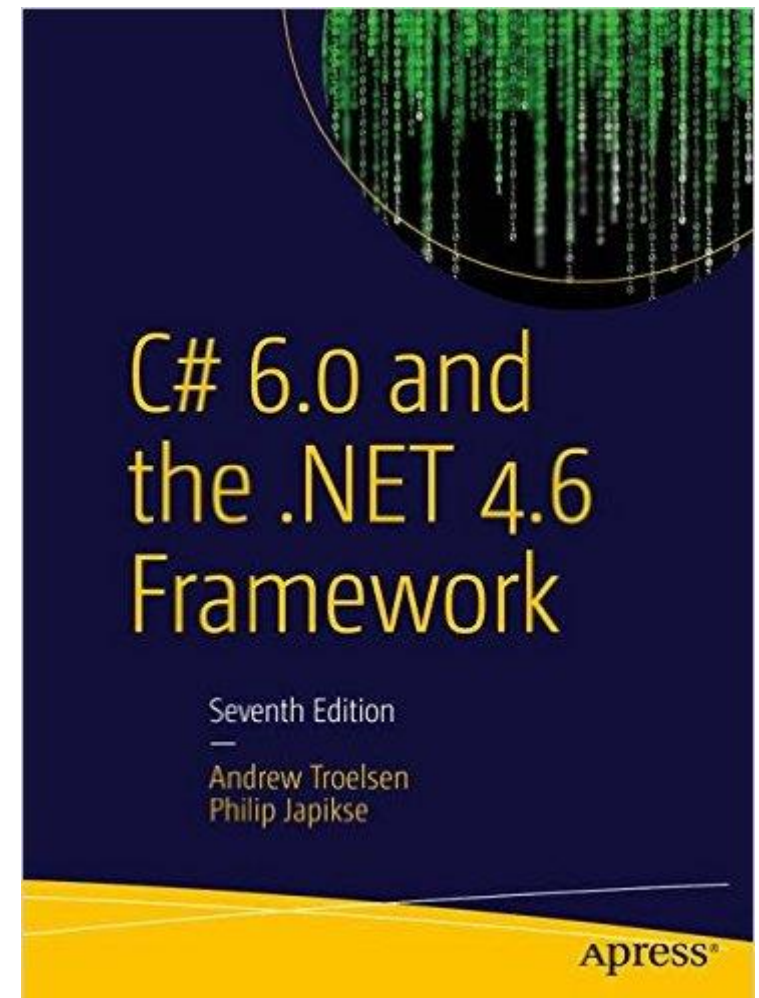
Chapter: Introducing Windows  
Presentation Foundation and XAML

Chapter: Programming with WPF  
Controls

Chapter: WPF Graphics Rendering  
Services

Chapter: WPF Resources,  
Animations, Styles, and Templates

Chapter: Notifications, Commands,  
Validation, and MVVM





- **Developing Universal Windows Apps with C# and XAML**

[https://mva.microsoft.com/en-us/training-courses/developing-universal-windows-apps-with-c-and-xaml-8363?l=8pXSyBGz\\_3904984382](https://mva.microsoft.com/en-us/training-courses/developing-universal-windows-apps-with-c-and-xaml-8363?l=8pXSyBGz_3904984382)

[https://mva.microsoft.com/en-us/training-courses/developing-universal-windows-apps-with-c-and-xaml-8363?l=XYrwDDGz\\_7804984382](https://mva.microsoft.com/en-us/training-courses/developing-universal-windows-apps-with-c-and-xaml-8363?l=XYrwDDGz_7804984382)

- **Create a "Hello, world" app (XAML)**
- <https://msdn.microsoft.com/en-us/windows/uwp/get-started/create-a-hello-world-app-xaml-universal>
- **What's a Universal Windows Platform (UWP) app?**
- <https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp>

## Create a "Hello, world" app (XAML)

<https://docs.microsoft.com/en-us/windows/uwp/get-started/create-a-hello-world-app-xaml-universal>

## Acknowledgment

Sources used in this presentation include:

XAML Tutorial <https://www.tutorialspoint.com/xaml/>