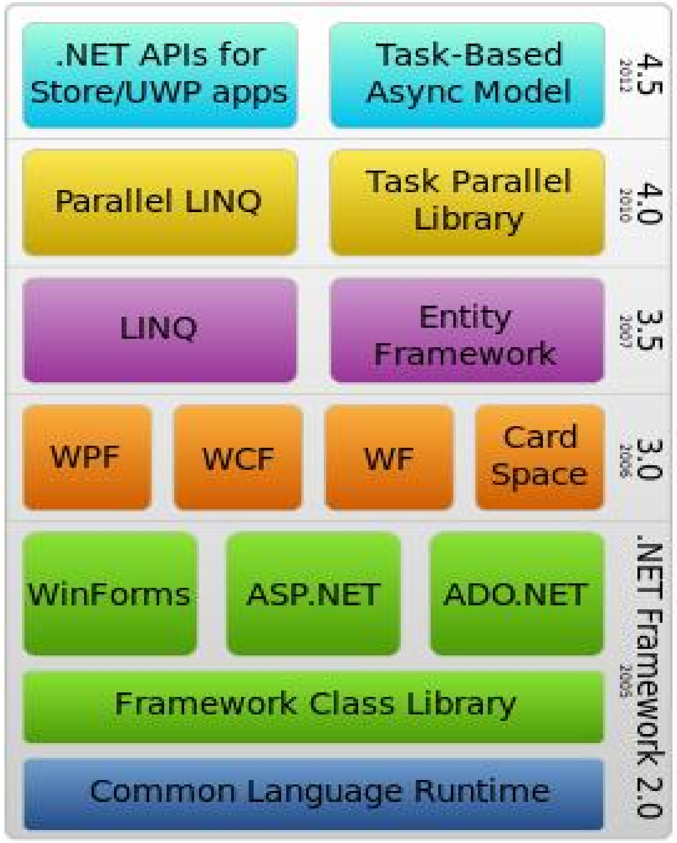
Windows Presentation Foundation

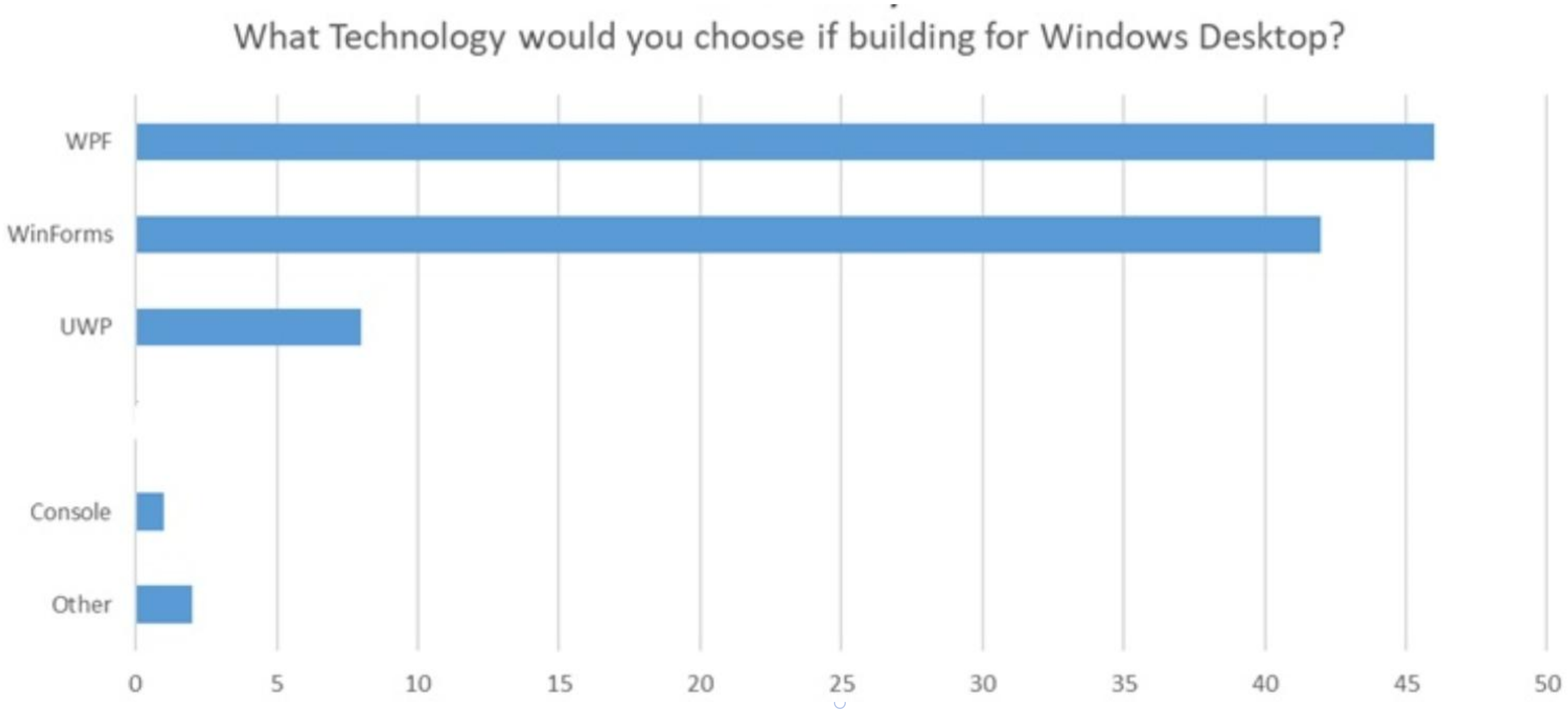
WPF

Agenda

- Winforms VS WPF
- Introduction WPF
- XAML
- Layout
- Types of properties


.NET APIs for Store/UWP apps
Task-Based Async Model
4.5
2012
Parallel LINQ
Task Parallel Library
4.0
2010
LINQ
Entity Framework
3.5
2007
WPF
WCF
WF
Card Space
3.0
2006
WinForms
ASP.NET
ADO.NET
Framework Class Library
Common Language Runtime
.NET Framework 2.0
2005

Winforms
VS
WPF

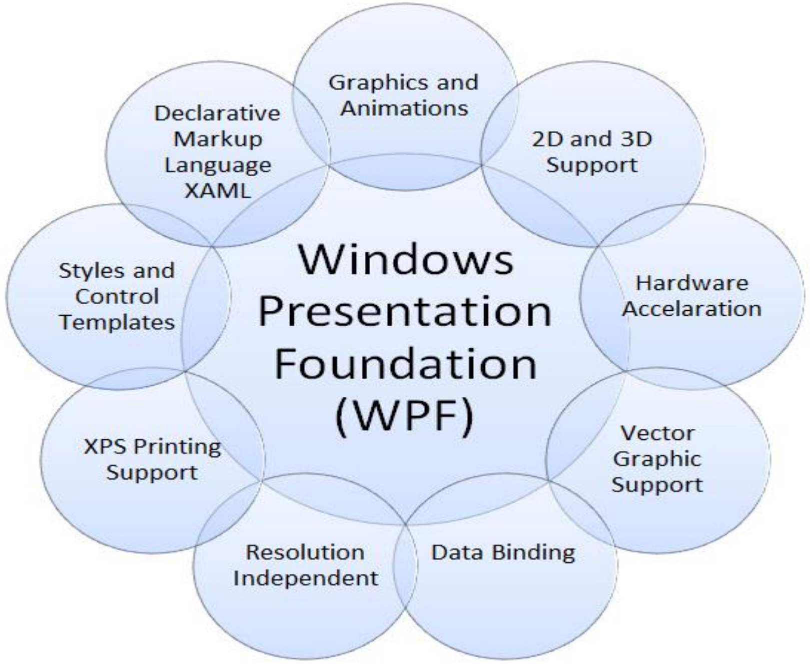

What Technology would you choose if building for Windows Desktop?
WPF
WinForms
UWP
Console
Other
0
5
10
15
20
25
30
35
40
45
50

Windows Presentation Foundation

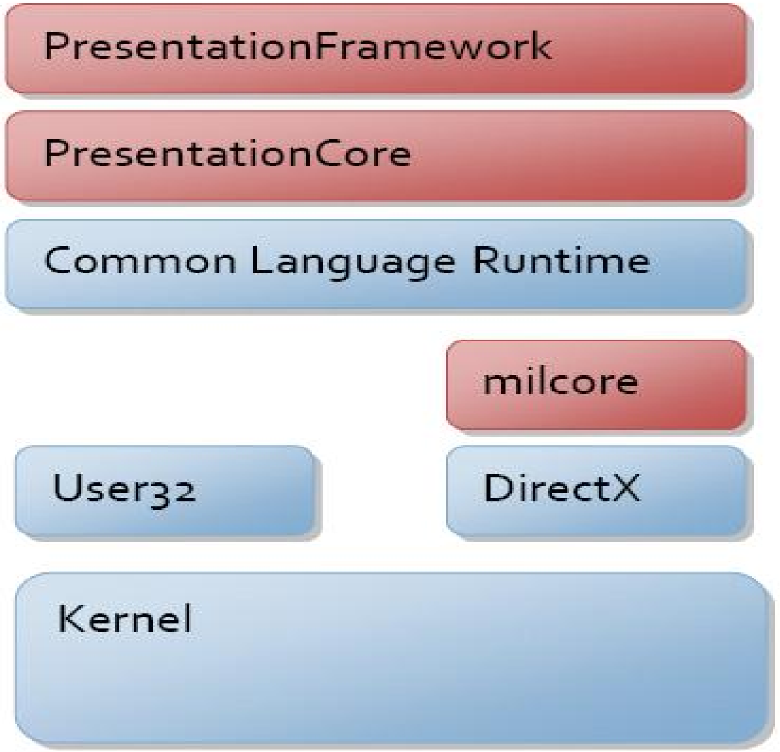
- Its formerly code name is **Avalon**
- It is a graphical subsystem in .NET Framework 3.0 (formerly called **WinFX**)
- **WPF** uses a markup language, known as **XAML**, for rich user interface development.
- It provides a consistent programming model for building applications and provides a clear separation between the user interface and the business logic
- **Microsoft Expression Blend**

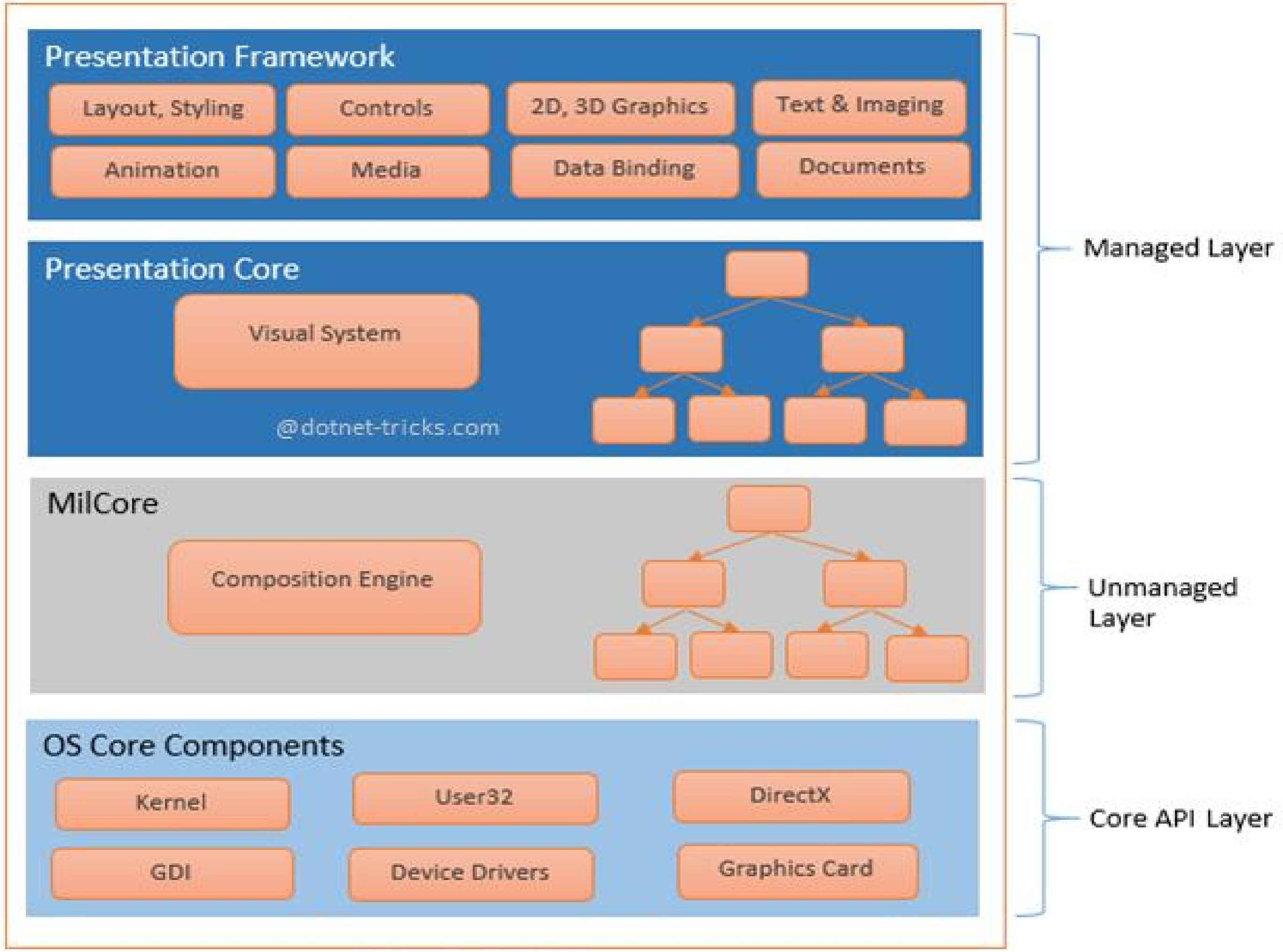
Features of WPF

- Data binding
- UI customization & graphics
- Easier implementation for the MVVM pattern
- The visual designer got much better



WPF Architecture

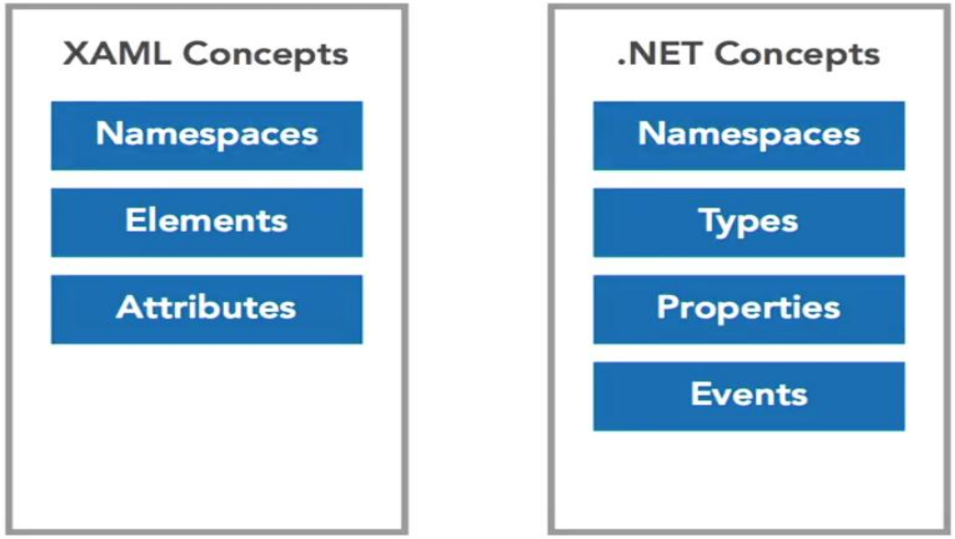




WPF Architecture

XAML

- Its eXtensible Application Markup Language
- It's the markup standard you use to define WPF user interfaces.
- **XAML** allows you to build a window without writing code



XAML Example

```
<Window x:Class="WpfApplication1.Window1“
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
Title="Window1"
Height="300" Width="300">
<Grid>
    <Button Height="23" Margin="94,76,108,0"
     Name="button1“ VerticalAlignment="Top”
     Click="button1_Click">Button
    </Button>
</Grid>
</Window>
```

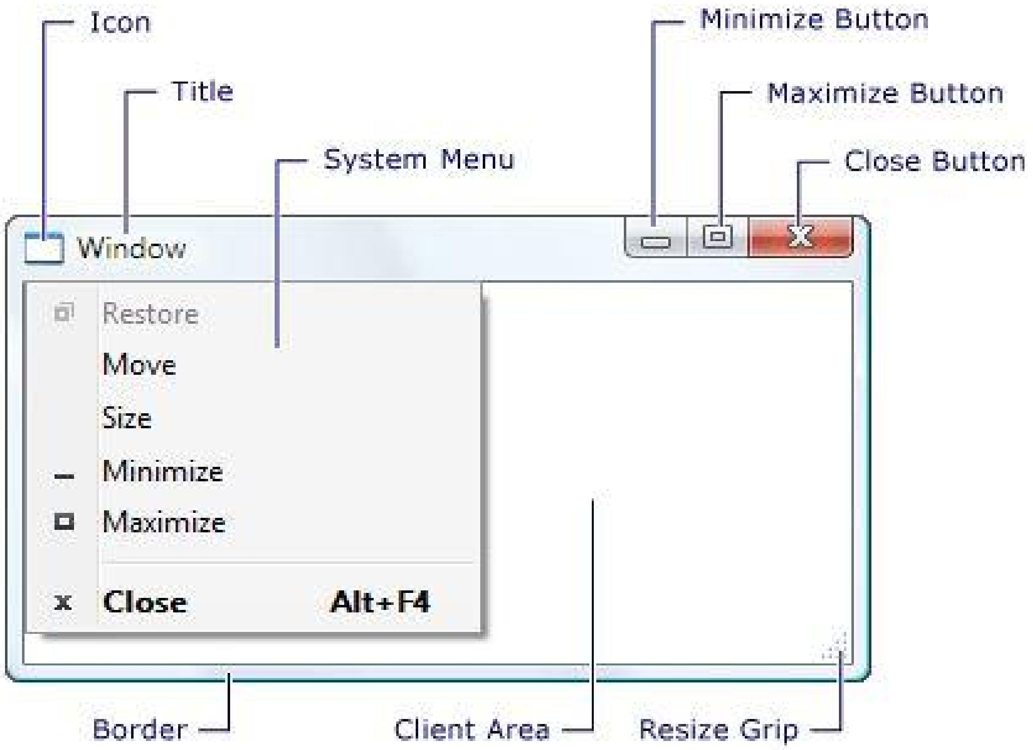
The Application Life Cycle

Run the program to show life cycle of WPF
Main>>APP>>Mainwindow

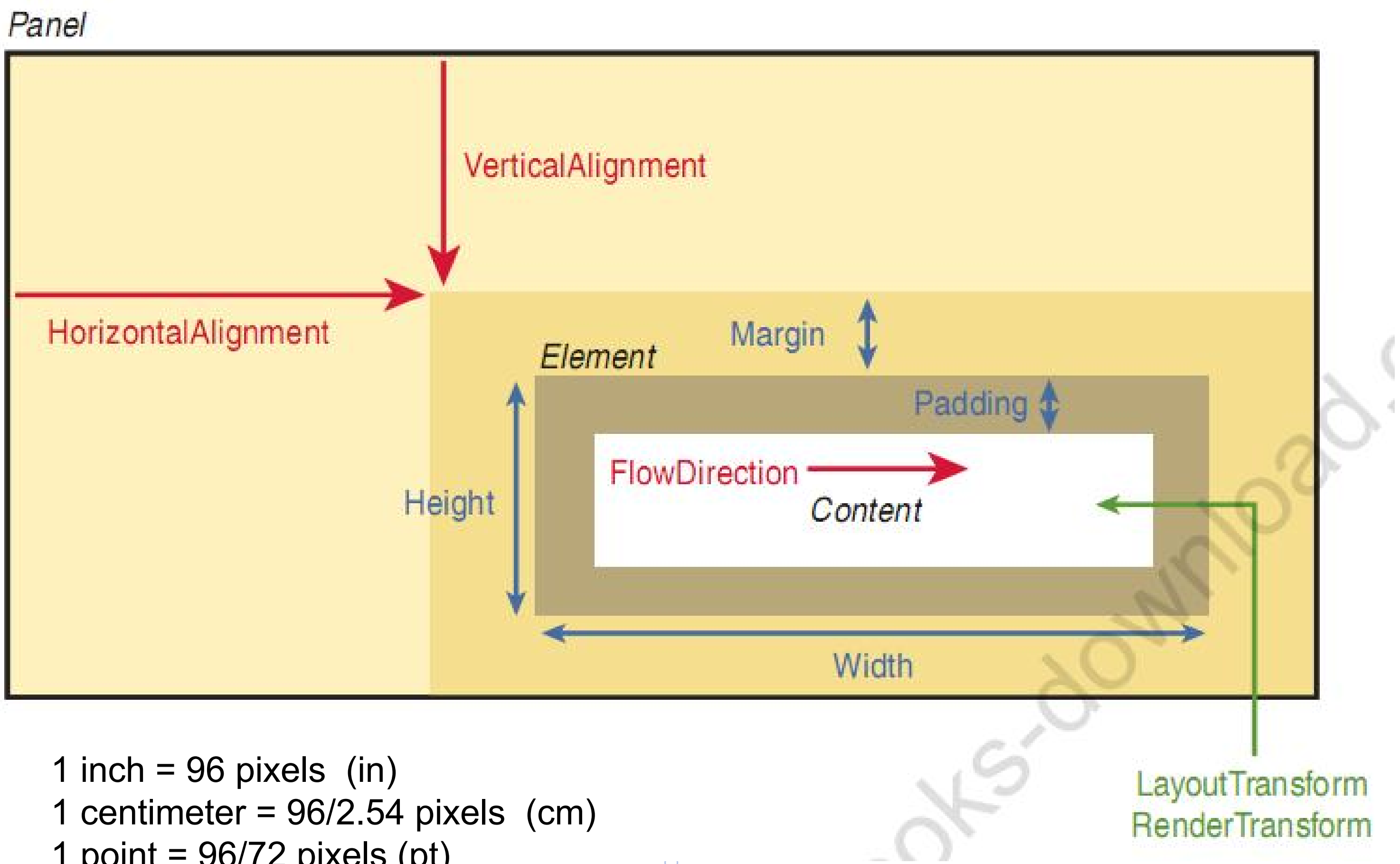
```
<Application
        x:Class="TestApplication.App”
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        StartupUri="Window1.xaml">
</Application>
```

The following figure illustrates the constituent parts of a window:



A window is divided into two areas: the non-client area and client area.

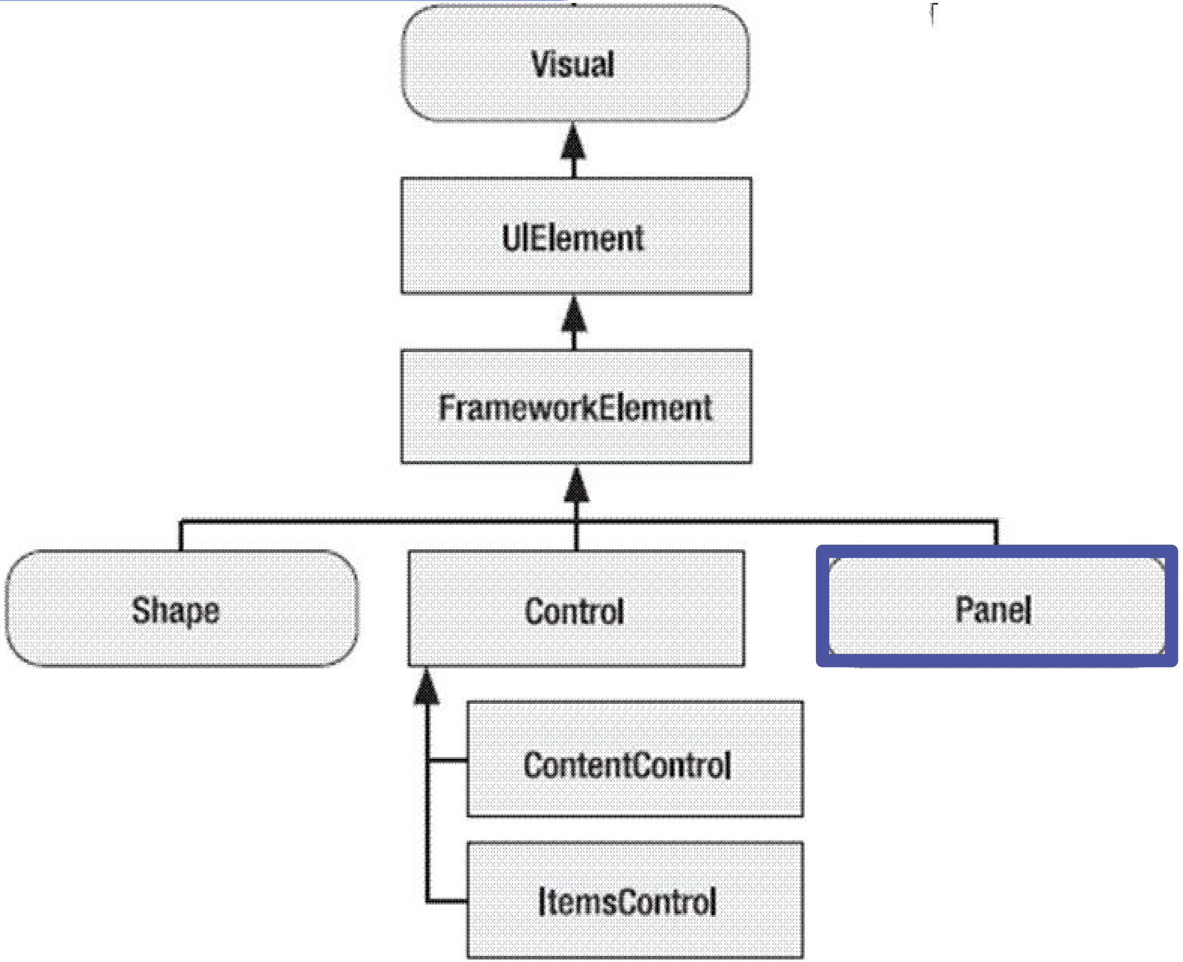


1 inch = 96 pixels (in)
1 centimeter = 96/2.54 pixels (cm)
1 point = 96/72 pixels (pt)

Types of windows

- Window
- Navigation
- Page
- DialogBox

Layout


Visual
UIElement
FrameworkElement
Shape
Control
Panel
ContentControl
ItemsControl

Panels

StackPanel :

Places elements in a horizontal or vertical stack

WrapPanel :

Places elements in a series of wrapped lines

DockPanel :

Aligns elements against an entire edge of the container

Grid :

Arranges elements in rows and columns according to an invisible table

UniformGrid :

Places elements in an invisible table but forces all cells to have the same size

Canvas :

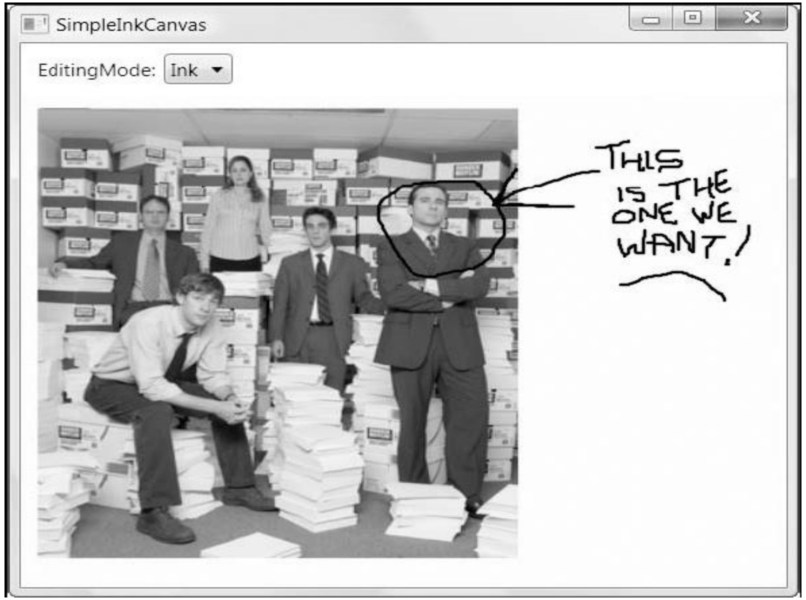
Allows elements to be positioned absolutely using fixed coordinates.

InkCanvas:

The primary purpose of the InkCanvas is to allow stylus input

InkCanvas:

- DefaultDrawingAttributes.
- EditingMode.
- Strokes.Clear().
- CopySelection().



Properties and Events in XAML

Types of properties

- Simple Property
- Complex Property
- Markup Extensions
- Attached Property

Event In XAML

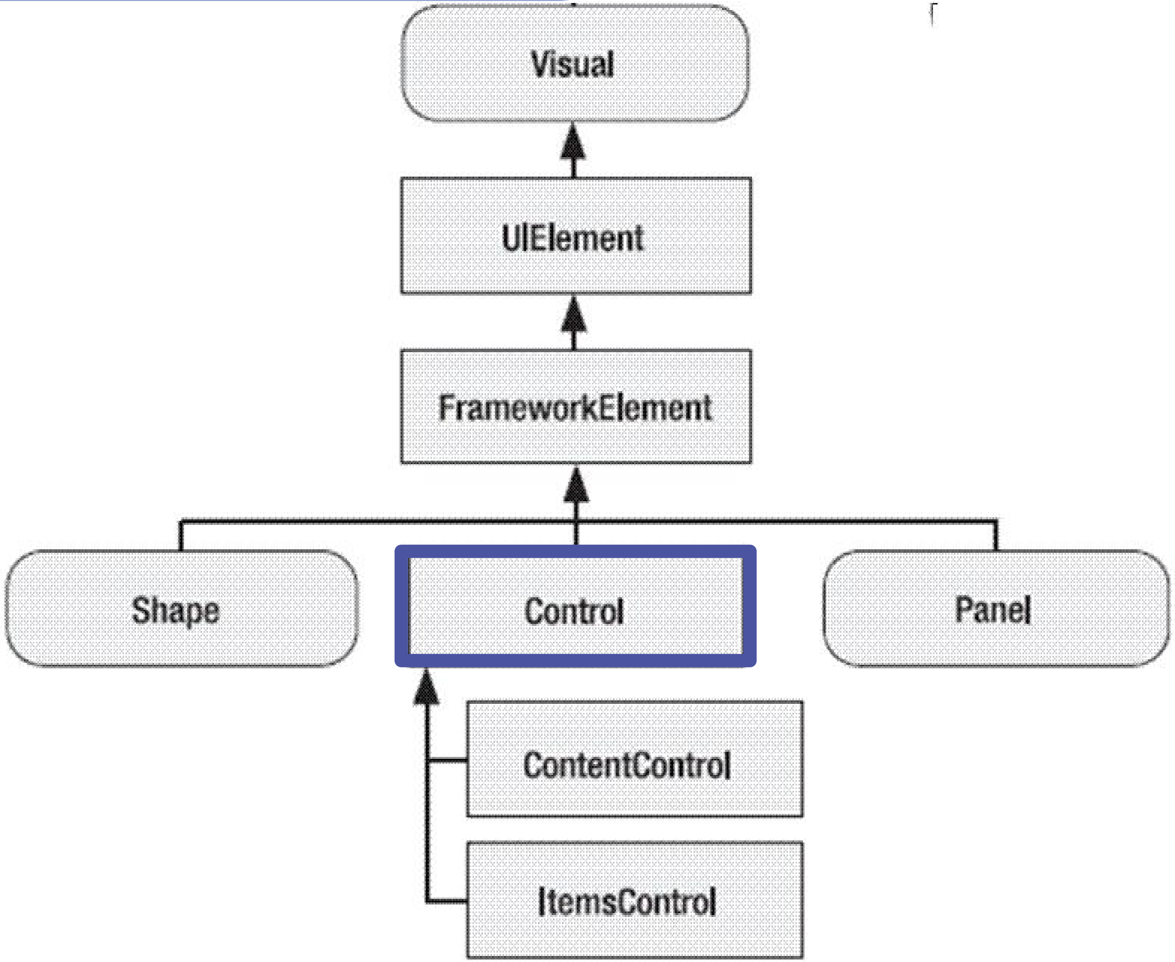
In XAML Each element have set of attribute used to give Property value to element ,and also attribute used to make event handler

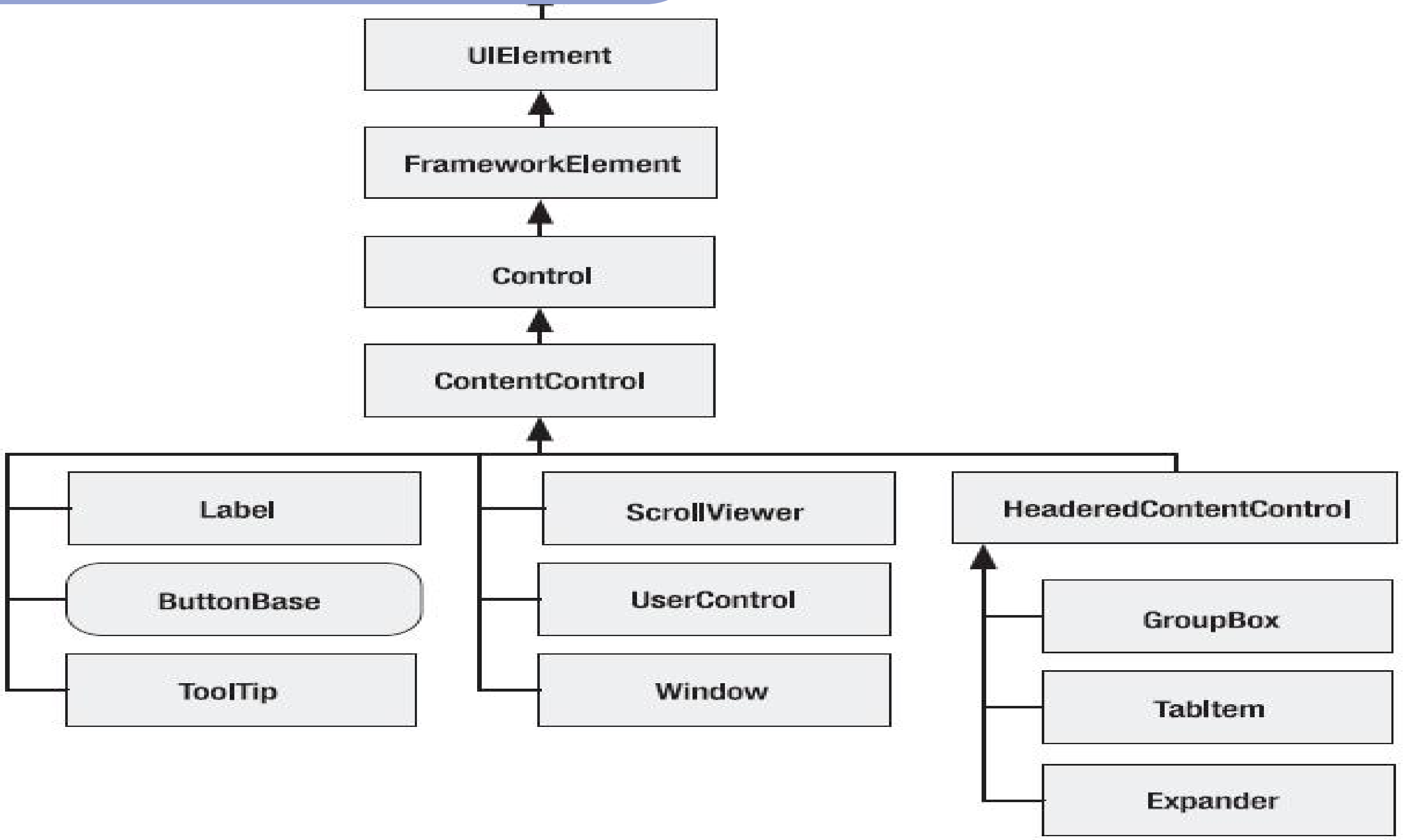
XAML :

```
<Button Name="btn1" Click="btn1_Click">
```

C# :

```
private void btn1_Click(object sender,
        RoutedEventArgs e)
```


Visual
UIElement
FrameworkElement
Shape
Control
Panel
ContentControl
ItemsControl


UIElement
FrameworkElement
Control
ContentControl
Label
ButtonBase
ToolTip
ScrollViewer
UserControl
Window
HeaderedContentControl
GroupBox
TabItem
Expander

The Content Property

- As Panel class adds the Children collection to hold nested elements, The **ContentControl** class adds a Content property, which accepts a single object

```
<Button Margin="3">Text content</Button>
```

```
<Button Margin="3">
    <Image Source="happyface.jpg“
                    Stretch="None" />
</Button>
```

The Content Property (Con.)

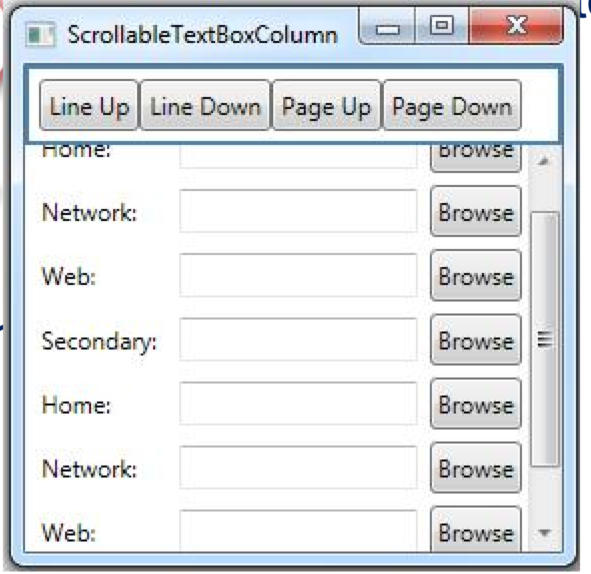
```
 <Button Margin="3">
<StackPanel>
<TextBlock Margin="3">
             Image and text button </TextBlock>
<Image Source="happyface.jpg"
Stretch="None" />
<TextBlock Margin="3">
             Courtesy of the StackPanel
</TextBlock>
</StackPanel>
</Button>
```

Specialized Containers

ScrollViewer

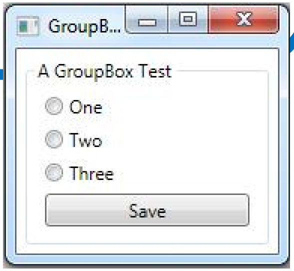
- In order to get scrolling support, you need to wrap the content you want to scroll inside a ScrollViewer.
- Although the ScrollViewer can hold anything, even to wrap a layout container

  such as Grid, StackPanel
- Specialized Containers
- **CanContentScroll** Proper



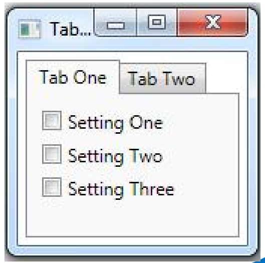
Headered Content Controls:GroupBox

```
<GroupBox Header="A GroupBox Test"
   Padding="5 " Margin="5" VerticalAlignment="Top">
   <StackPanel>
              <RadioButton Margin="3">One</RadioButton>
              <RadioButton Margin="3">Two</RadioButton>
              <RadioButton
Margin="3">Three</RadioButton>
              <Button Margin="3">Save</Button>
   </StackPanel>
</GroupBox>
```

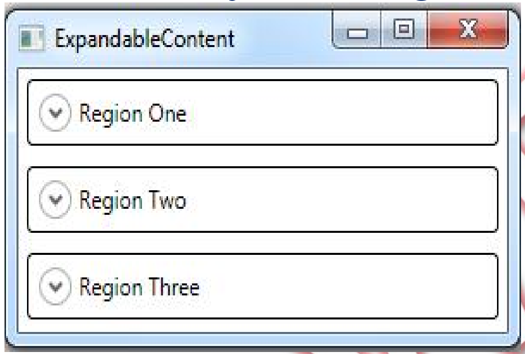


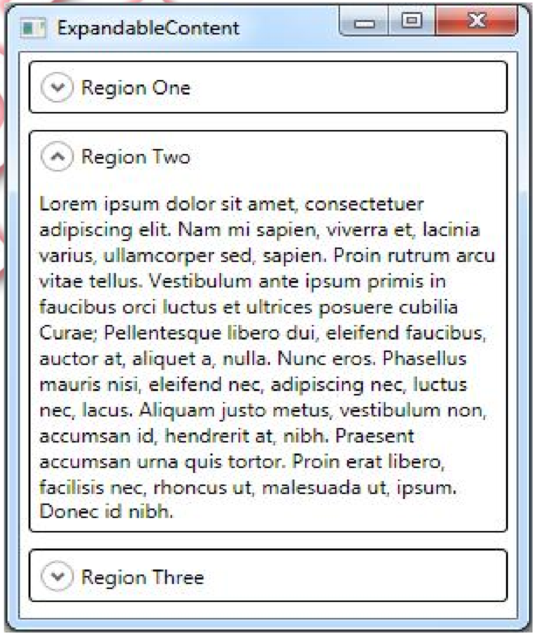
TabControl

```
<TabControl Margin="5" TabStripPlacement="Top">
<TabItem Header="Tab One">
<StackPanel Margin="3">
<CheckBox Margin="3">Setting One</CheckBox>
<CheckBox Margin="3">Setting Two</CheckBox>
<CheckBox Margin="3">Setting Three</CheckBox>
</StackPanel>
</TabItem>
<TabItem Header="Tab Two">…</TabItem>
</TabControl>
```



- The Expander
  It wraps a region of content that the user can show or hide by clicking a small arrow button



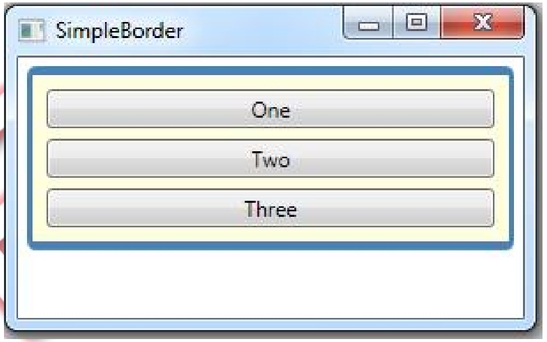


Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam mi sapien, viverra et, lacinia varius, ullamcorper sed, sapien. Proin rutrum arcu vitae tellus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque libero dui, eleifend faucibus, auctor at, aliquet a, nulla. Nunc eros. Phasellus mauris nisi, eleifend nec, adipiscing nec, luctus nec, lacus. Aliquam justo metus, vestibulum non, accumsan id, hendrerit at, nibh. Praesent accumsan urna quis tortor. Proin erat libero, facilisis nec, rhoncus ut, malesuada ut, ipsum. Donec id nibh.

Decorators

- The Border

```
<Border Margin="5" Padding="5"
    Background="LightYellow"
 BorderBrush="SteelBlue“
 BorderThickness="3,5,3,5"
 CornerRadius="3“ >
 <StackPanel>
        <Button Margin="3">One</Button>
        <Button Margin="3">Two</Button>
        <Button Margin="3">Three</Button>
 </StackPanel>
</Border>
```



Decorators (Con.)

- TheViewbox

  The basic principle behind the Viewbox any content you place inside the Viewbox is scaled up or down to fit the bounds of the Viewbox

Button

- When IsCancel is true

  This button is designated as the cancel button for a window. You press the Escape key while positioned anywhere on the current window, this button is triggered

- When IsDefault is true

  This button is designated as the default button(accept button)

- However, there should be only a single cancel button and a single default button in a window

ToggleButton

- ToggleButton
  - A button that has two states (pushed or unpushed). When you click a ToggleButton, it stays in its pushed state until you click it again to release it.
  - The ToggleButton is genuinely useful inside a ToolBar
  - Class derived from ButtonBase
  - RadioButton and Checkbox drived from ToggleButton Class

ToolTip

- The ToolTip property is defined in the FrameworkElement class, so it’s available on anything you’ll place in a WPF window

```
<Button ToolTip="This is my tooltip">
            I have a tooltip
 </Button>
```

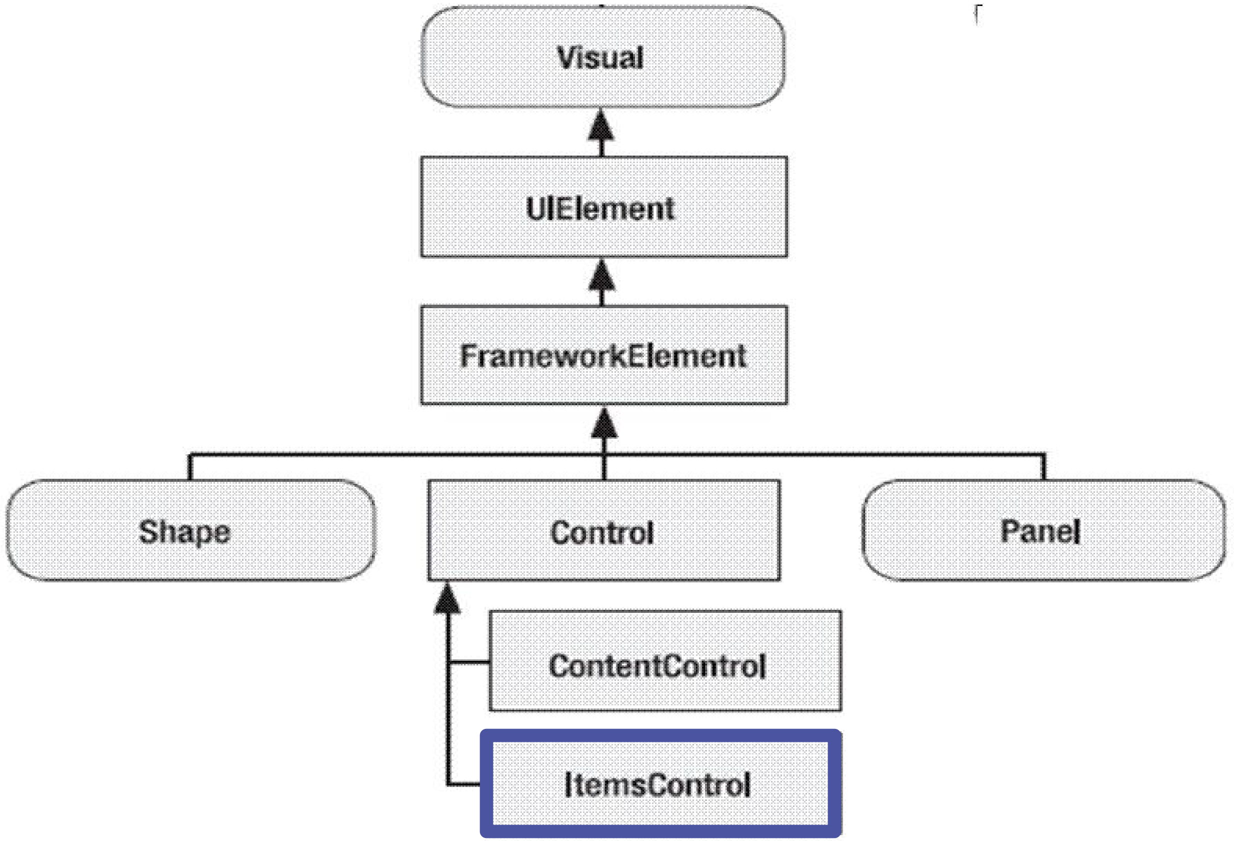
Text Controls

- WPF includes three text-entry controls:
  - TextBox
  - RichTextBox
  - PasswordBox
- The PasswordBox derives directly from Control.
- The TextBox and RichTextBox controls go through another level and derive from TextBoxBas

Text Controls & PasswordBox

- The **PasswordBox** looks like a TextBox, but it displays a string of circle symbols to mask the characters it shows
- You can choose a different mask character by setting the **PasswordChar** property
- **PasswordBox** does not support the clipboard, so you can't copy the text inside
- It provides a **MaxLength** property

Fundamental classes of WPF



ListBox

```
<ListBox>
<ListBoxItem>
        <Image Source="happyface1.jpg">
        </Image>
</ListBoxItem>
<ListBoxItem>
        <Image Source="happyface2.jpg">
        </Image>
</ListBoxItem>
</ListBox>
```