Introduction To Silverlight 2

Brij Mohan

brij.mohan@ascendum.com
http://www.dotnetglobe.com

Agenda

- Silverlight architecture
- XAML
- CoreCLR and Base Class Library
- Silverlight security
- DOM integration
- Controls and templates
- Styles
- File I/O and Networking

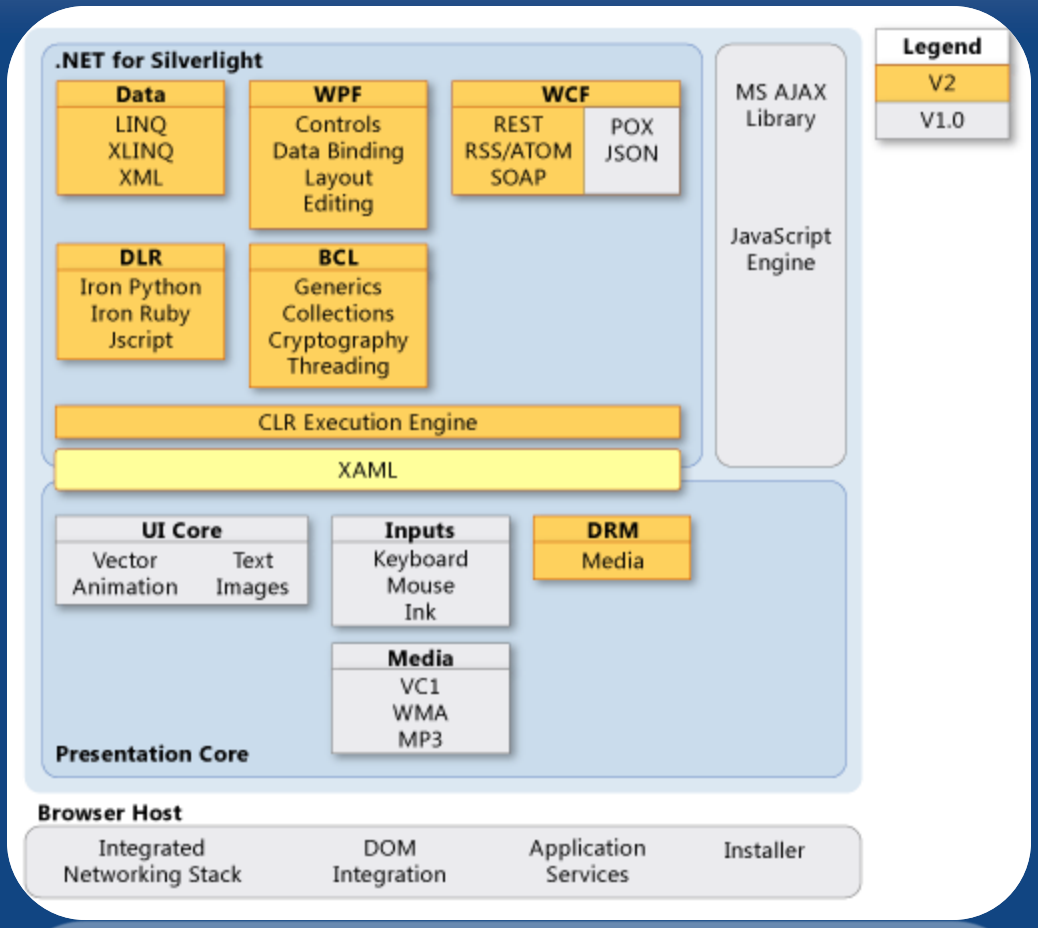
Silverlight

- Formerly known as "WPF/E"
- Microsoft's platform for rich, highly interactive Web experiences and RIAs
  - Cross-platform (browsers and OSes)
    - Windows, Mac OS, Linux ("Moonlight")
    - Internet Explorer, Firefox, Safari, and more
  - XAML-based rendering (subset of WPF XAML)
- Implemented as browser plug-in
  - Quick, easy install experience

Versions

- Silverlight 1.0
  - Shipped September 2007
  - XAML rendering and JavaScript API
- Silverlight 2
  - RTM already available
  - XAML, .NET Framework, managed code, dynamic languages (e.g., IronRuby)
  - 4.6 MB Download

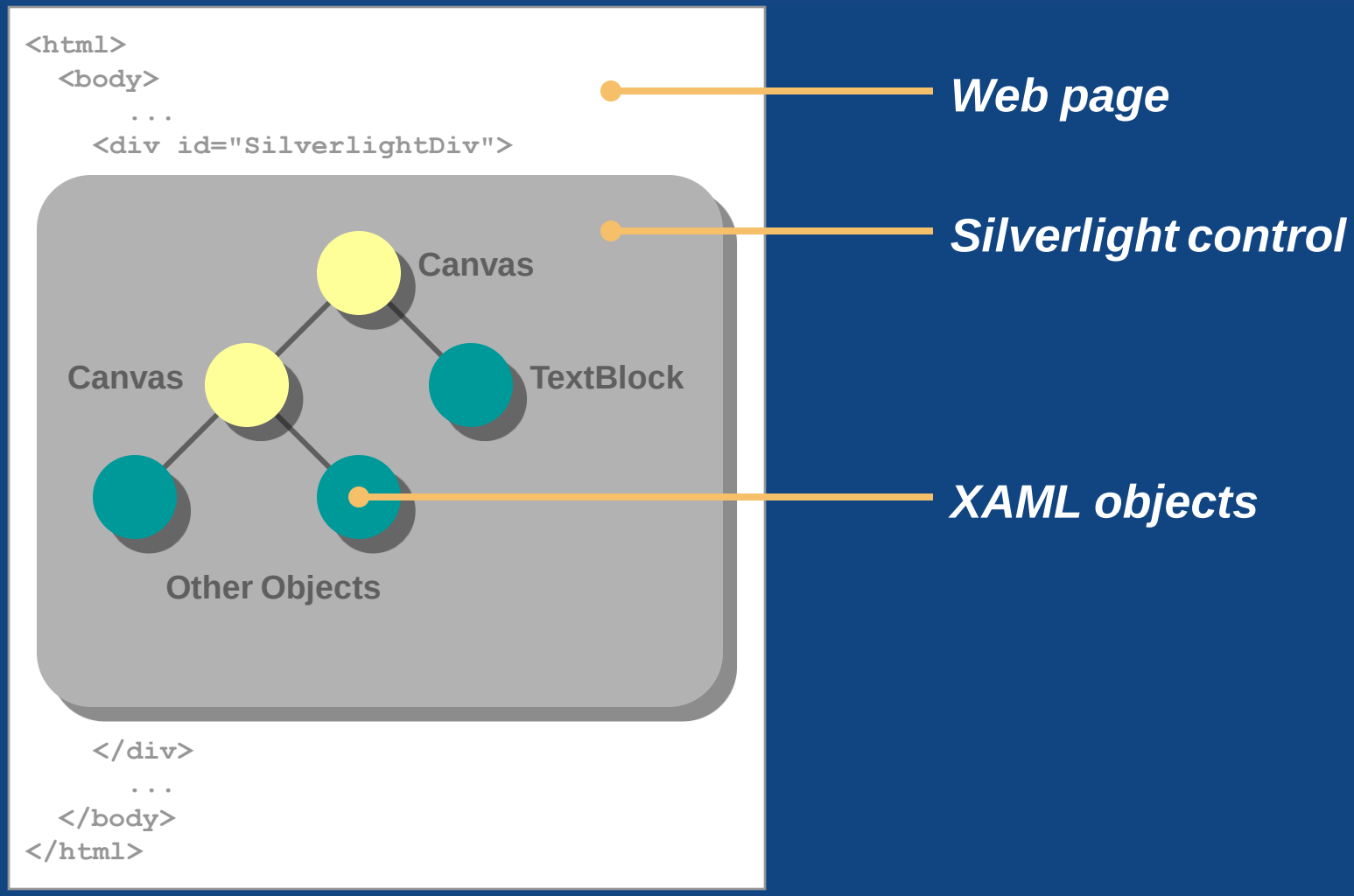
Silverlight Architecture



XAML

```
<Canvas Width="300" Height="300"
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <Ellipse Canvas.Left="20" Canvas.Top="20"
    Height="200" Width="200"
    Stroke="Black" StrokeThickness="10" Fill="Yellow" />
  <Ellipse Canvas.Left="80" Canvas.Top="80"
    Height="35" Width="25" Stroke="Black" Fill="Black" />
  <Ellipse Canvas.Left="140" Canvas.Top="80"
    Height="35" Width="25" Stroke="Black" Fill="Black" />
  <Path Data="M 70, 150 A 60, 60 0 0 0 170, 150"
    Stroke="Black" StrokeThickness="15"
    StrokeStartLineCap="Round" StrokeEndLineCap="Round" />
</Canvas>
```

XAML DOM


<html>
<body>
...

Canvas
Canvas
TextBlock
Other Objects

...
</body>
</html>
Web page
Silverlight control
XAML objects

Naming XAML Objects

- Use x:Name attributes to assign names to XAML objects (analogous to IDs in ASP.NET)

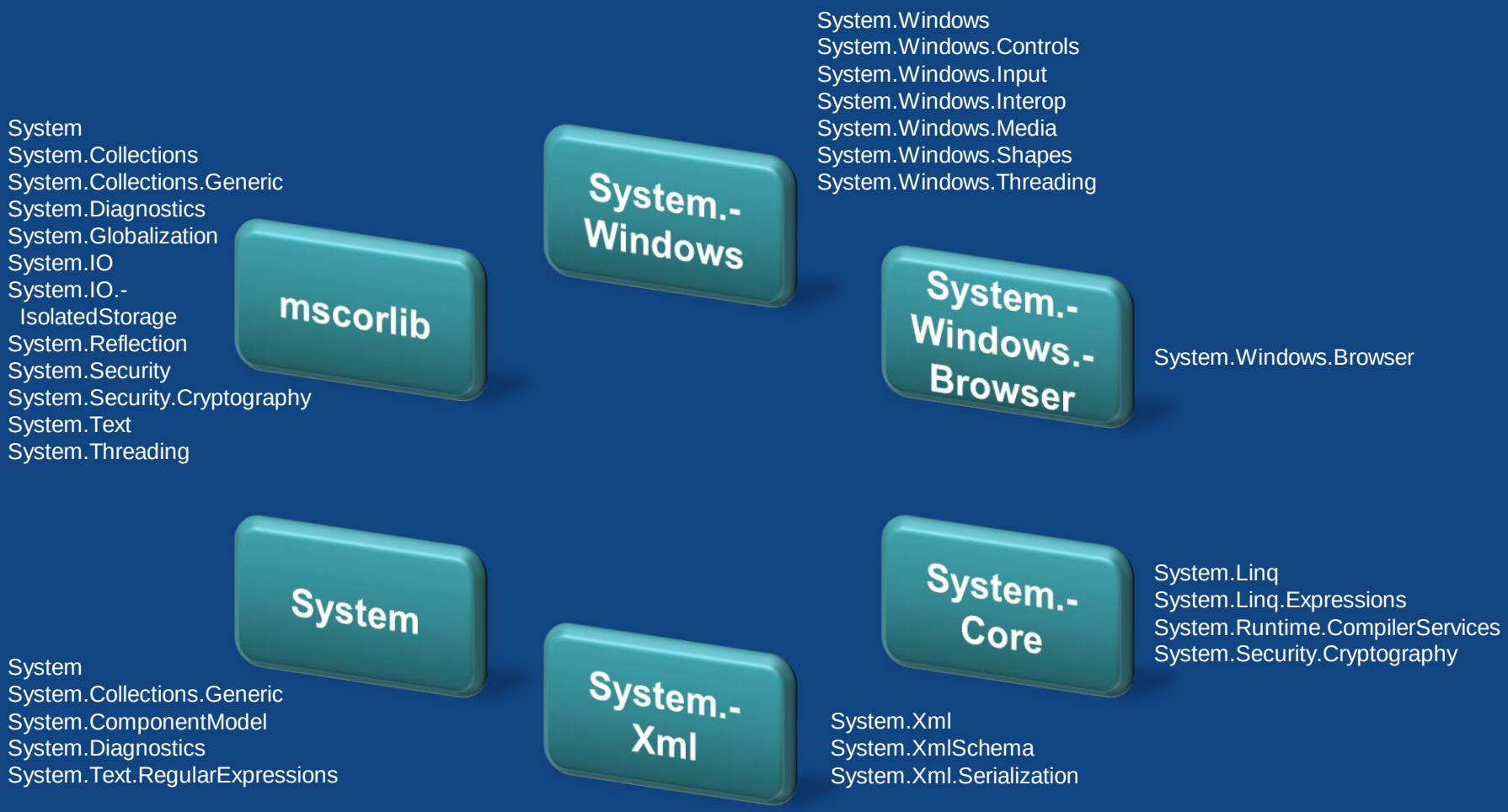
```
<Rectangle Canvas.Left="50" Canvas.Top="50" Fill="Yellow"
  Width="300" Height="200" Stroke="Black" StrokeThickness="10"
  x:Name="YellowRect" />
```

Object can now be referred to as "YellowRect" in code

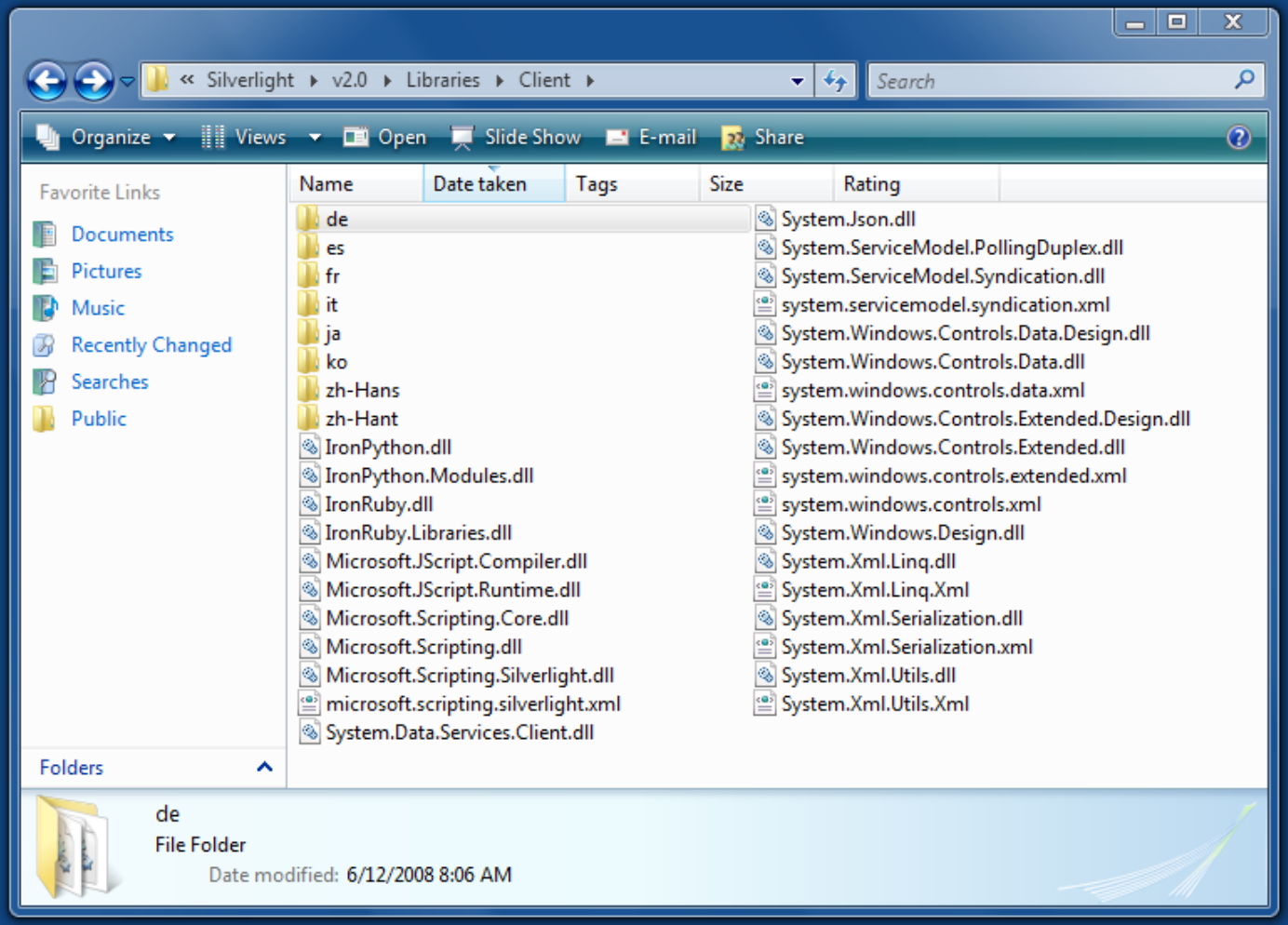
The Silverlight 2 CLR ("CoreCLR")

- Refactored version of full-size CLR
  - Same core type system, JIT compiler, etc.
  - COM interop, remoting, binary serialization, server GC, and other features removed
  - Multiple CLR instances per process supported
  - Most globalization support pushed down to OS
  - Dynamic Language Runtime (DLR) added
- Small footprint (< 2MB), cross-platform

Core Base Class Library



Extended Base Class Library



Input Events

- XAML objects input fire events
  - Mouse events
  - Keyboard events
- Most input events "bubble up" XAML DOM
  - Also known as "event routing"
  - Use RoutedEventArgs.Handled controls routing
  - RoutedEventArgs.Source refers to originator
- Handlers can be registered declaratively or programmatically

Mouse Events

Event	Description
MouseLeftButtonDown	Fires when left mouse button is depressed over a UI element
MouseLeftButtonUp	Fires when left mouse button is released over a UI element
MouseEnter	Fires when mouse enters a UI element
MouseLeave	Fires when mouse leaves a UI element
MouseMove	Fires when mouse moves over a UI element

```
private void OnMouseLeftButtonDown(Object sender,
    MouseButtonEventArgs e)
{
    double x = e.GetPosition(null).X; // X-coordinate
    double y = e.GetPosition(null).Y; // Y-coordinate
}
```

Keyboard Events

Event	Description
KeyDown	Fires when key is depressed
KeyUp	Fires when key is released
GotFocus	Fires when element receives the input focus
LostFocus	Fires when element loses the input focus

```
private void OnKeyDown(Object Sender, KeyEventArgs e)
{
    HtmlPage.Window.Alert(e.Key.ToString()); // Display key code
}
```

Declarative Handler Registration

```
<Rectangle Canvas.Left="50" Canvas.Top="50" Fill="Yellow"
  Width="300" Height="200" Stroke="Black" StrokeThickness="10"
  MouseLeftButtonDown="OnClick" />
```

```
private void OnClick(Object sender, MouseButtonEventArgs e)
{
    ((Rectangle)sender).Fill = new SolidColorBrush(Colors.Red);
}
```

DOM Integration

- System.Windows.Browser namespace contains classes for accessing browser DOM
  - HtmlPage, HtmlWindow, and others
- Managed -> unmanaged
  - Access DOM from managed code
  - Call JavaScript functions from managed code
- Unmanaged -> managed
  - Call managed code from JavaScript
  - Process DOM events with managed code

Alerting the User

```
HtmlPage.Window.Alert ("Error!");
```

Processing DOM Events in C#

```
<select id="Options"
    <option value="0" selected>One</option>
    <option value="1">Two</option>
    <option value="2">Three</option>
</select>
```

```
HtmlElement elem = HtmlPage.Document.GetElementById("Options");
elem.AttachEvent("onchange",
    new EventHandler<HtmlEventArgs>(OnSelectionChanged));
      .
      .
      .
protected void OnSelectionChanged(object sender, HtmlEventArgs e)
{
    // TODO: Respond to the event
}
```

Accessing DOM Elements from C#

```
string text;
HtmlElement options = HtmlPage.Document.GetElementById("Options");

foreach (HtmlElement option in options.Children)
{
    if (String.Compare(option.GetAttribute("selected"),
        "true", true) == 0)
    {
        text = option.GetAttribute("value");
    }
}
```

Controls

- More than 20 built-in controls
  - Canvas, StackPanel, Grid, and GridSplitter
  - Button, CheckBox, HyperlinkButton, RepeatButton, RadioButton, and ToggleButton
  - TextBox, ListBox, and DataGrid
  - TabControl, Slider, and MultiScaleImage
  - Border, Calendar, DatePicker, and more!
- Support styles, templates, and data binding

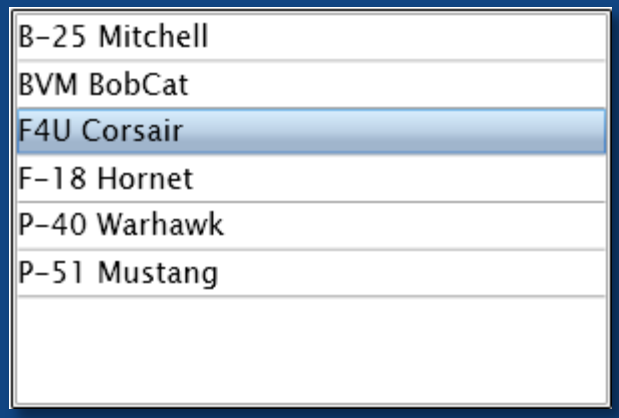
Button Control

```
<Button Width="256" Height="128" FontSize="24"
  Content="Click Me" Click="Button_Click" />
```


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

ListBox Control

```
<ListBox>
  <ListBoxItem Content="B-25 Mitchell" />
  <ListBoxItem Content="BVM BobCat" />
  <ListBoxItem Content="F4U Corsair" />
  <ListBoxItem Content="F-18 Hornet" />
  <ListBoxItem Content="P-40 Warhawk" />
  <ListBoxItem Content="P-51 Mustang" />
</ListBox>
```



Control Customization

- ContentControl derivatives support deep customization via Content property
  - Button
  - ListBoxItem
  - DataGridCell
  - TabItem and more
- Use property element syntax to assign non-trivial values to Content property

Button with Custom Content

```
<Button Width="256" Height="128" Click="Button_Click">
  <Button.Content>
    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
      <Ellipse Width="75" Height="75" Margin="10">
        <Ellipse.Fill>
          <RadialGradientBrush GradientOrigin="0.25,0.25">
            <GradientStop Offset="0.25" Color="White" />
            <GradientStop Offset="1.0" Color="Red" />
          </RadialGradientBrush>
        </Ellipse.Fill>
      </Ellipse>
      <TextBlock FontSize="24" Text="Click Me" VerticalAlignment="Center" />
    </StackPanel>
  </Button.Content>
</Button>
```

ListBox with Custom Items

```
<ListBox>
  <StackPanel Orientation="Horizontal">
    <Image Source="Images/B-25.jpg" Margin="5" />
    <TextBlock Text="B-25 Mitchell" Margin="5"
      HorizontalAlignment="Center" VerticalAlignment="Center" />
  </StackPanel>
    ...
</ListBox>
```

Control Templates

- Redefine a control's entire visual tree
  - Perform extreme customization without changing basic behavior of control
  - Exposed through control's Template property (inherited from Control base class)
- Use {TemplateBinding} to flow property values from control to template
- Use ContentPresenter and ItemsPresenter to flow content and items to template

Elliptical Button

```
<Button>
  <Button.Template>
    <ControlTemplate TargetType="Button">
      <Grid>
        <Ellipse Width="256" Height="128">
          <Ellipse.Fill>
            <RadialGradientBrush GradientOrigin="0.25,0.25">
              <GradientStop Offset="0.25" Color="White" />
              <GradientStop Offset="1.0" Color="Red" />
            </RadialGradientBrush>
          </Ellipse.Fill>
        </Ellipse>
        <TextBlock FontSize="24" Text="Click Me"
          HorizontalAlignment="Center" VerticalAlignment="Center" />
      </Grid>
    </ControlTemplate>
  </Button.Template>
</Button>
```

Styles

- Styles provide level of indirection between visual properties and their values
  - Define style as XAML resource
  - Apply style using {StaticResource} markup extension
- Can be scoped globally or locally
- Combine styles and templates to “stylize” controls with custom visual trees

Defining and Using Styles

Style name

Prevents style from being applied to non-Buttons

```
<Style x:Key="ButtonStyle" TargetType="Button">
  <Setter Property="Width" Value="256" />
  <Setter Property="Height" Value="128" />
  <Setter Property="FontSize" Value="24" />
</Style>
  ...
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" ... />
<Button Style="{StaticResource ButtonStyle}" Width="128" ... />
```

Explicit property value overrides style property value

Combining Styles and Templates

```
<Style x:Key="EllipticalButton" TargetType="Button">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Button">
        <Grid>
          <Ellipse Width="{TemplateBinding Width}" Height="{TemplateBinding Height}">
            <Ellipse.Fill>
              <RadialGradientBrush GradientOrigin="0.25,0.25">
                <GradientStop Offset="0.25" Color="White" />
                <GradientStop Offset="1.0" Color="Red" />
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
          <ContentPresenter Content="{TemplateBinding Content}"
            HorizontalAlignment="Center" VerticalAlignment="Center"
            FontSize="{TemplateBinding FontSize}" />
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
  ...
<Button Style="{StaticResource EllipticalButton}" Content="Click Me"
  Width="256" Height="128" FontSize="24" Click="Button_Click" />
```

Questions?

Brij Mohan

brij.mohan@ascendum.com
http://www.dotnetglobe.com