

Platinum



DOTNETNUKE™



Gold



ComponentOne



SYLLOGISTEKS

talentporte

TALX



Silver



Silverlight 101: Writing your first Silverlight Application

Michael Eaton

mjeaton@validussolutions.com

@mjeaton on twitter

Dev Tools · SharePoint Web Parts · jQuery UI Widgets



ComponentOne®



www.componentone.com

XAML is...

- A declarative markup language
- Based on XML
- Used to initialize structured values and objects
- Used extensively in
 - WPF
 - Silverlight
 - Windows Workflow

In Silverlight (and WPF), XAML is...

- Used as the UI markup language to define
 - UI elements
 - Data binding
 - Other features
- Elements map directly to CLR object instances
- Attributes map to properties and events on those objects

Anatomy of a XAML file

<Window | Page | UserControl

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

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

`>`

`<!-- Stuff goes here -->`

</Window | Page | UserControl>

Tools for writing XAML

- Visual Studio
- KaXAML
- Expression Blend / Design
- XAMLPad
 - Free. Ships with the .NET Framework
- #Develop

XAML Tags

- Various ways to accomplish the same thing
 - <Button>This is a button</Button>
 - <Button Content="This is a button"></Button>
 - <Button Content="This is a button"/>

Did you know?

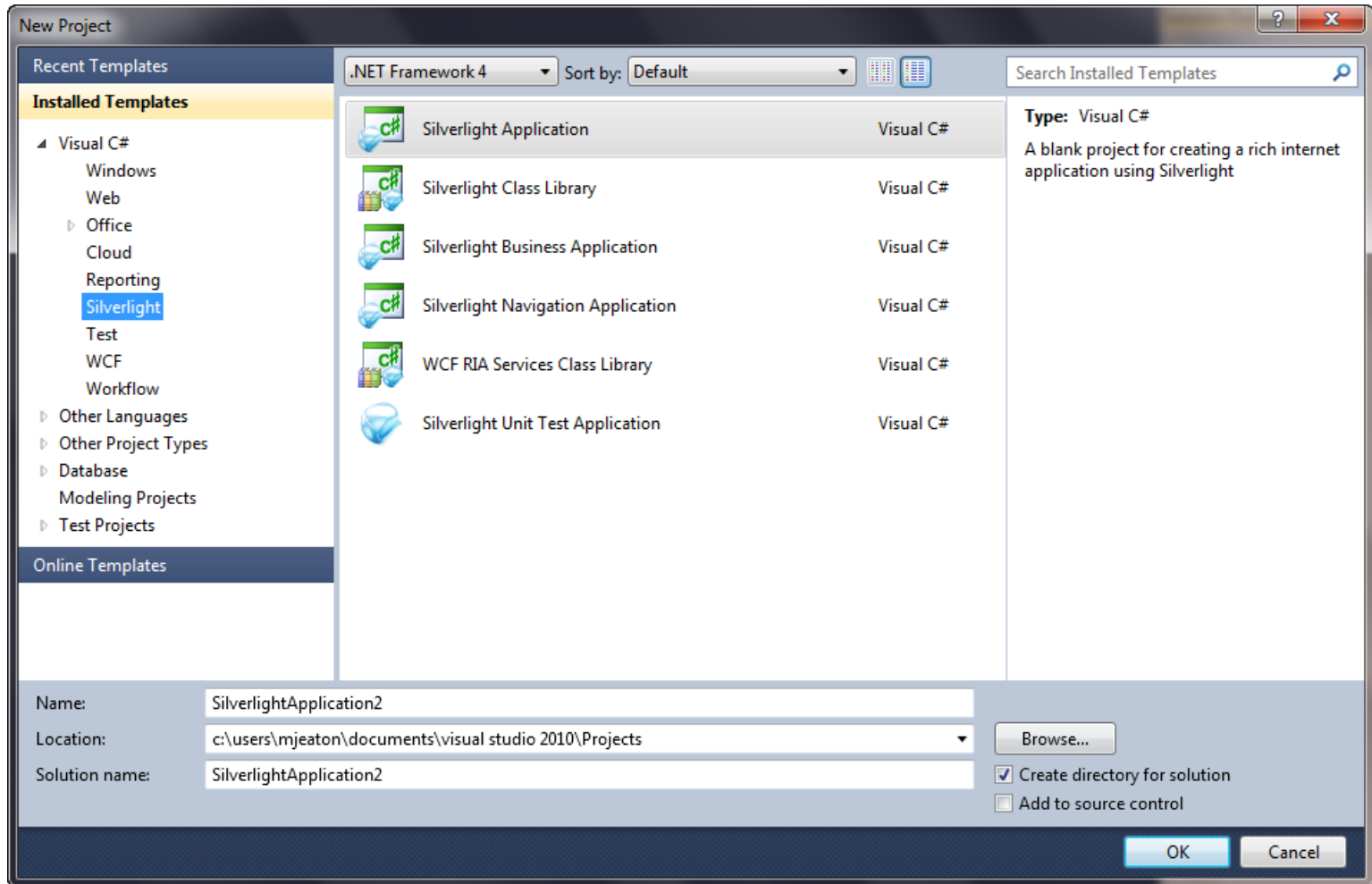
- Anything you can do in XAML, you can do in code

```
var b = new Button();  
b.Content = "hello, world";  
b.Click += clickedEvent;
```

is the same as

```
<Button Content="hello, world"  
Click="clickedEvent"/>
```

File | New



New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

Host the Silverlight application in a new Web site

New Web project name:
SampleApplication.Web

New Web project type:
ASP.NET Web Application Project

Options

Silverlight Version:
Silverlight 4

Enable WCF RIA Services

New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

Host the Silverlight application in a new Web site

New Web project name:
SilverlightApplication2.Web

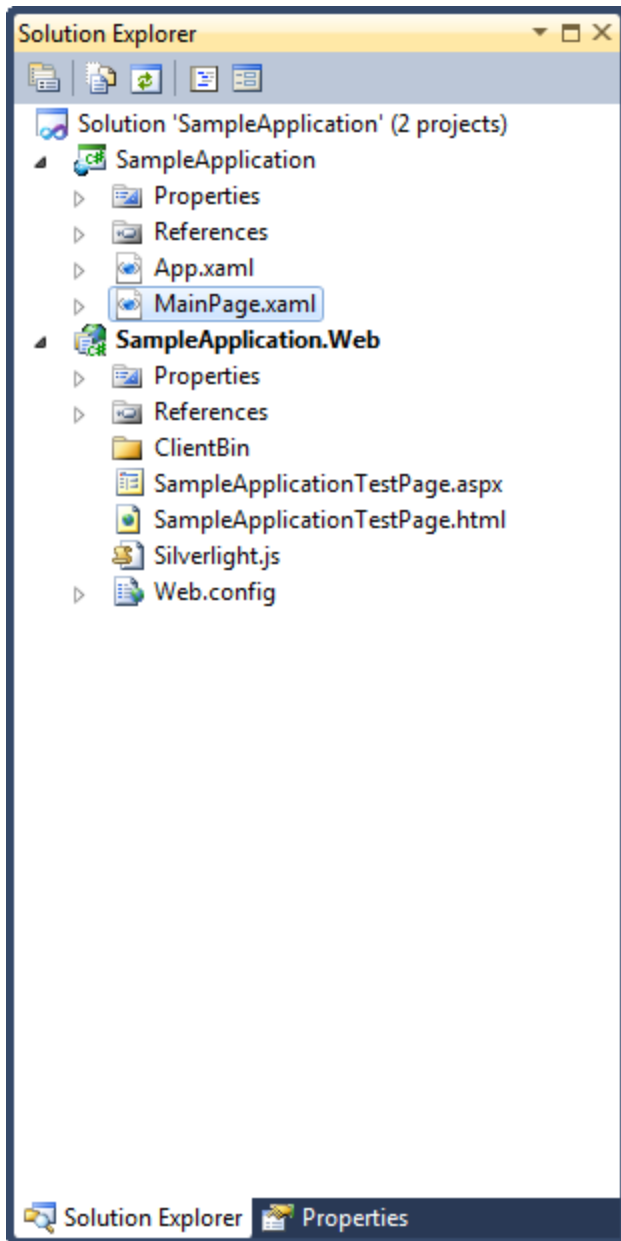
New Web project type:
ASP.NET MVC Web Project

Options

Silverlight Version:
Silverlight 4

Enable WCF RIA Services

OK Cancel



In the Silverlight Application:

- App.xaml
- MainPage.xaml

In the web project:

- ClientBin
- SampleApplicationTestPage.aspx

The XAP file

- ClientBin\`<yourapp>`.xap

App.xaml

```
<Application.Resources>  
  <Style TargetType="Button">  
  
    <Setter Property="FontSize"  
      Value="24" />  
  
    <Setter Property="FontFamily"  
      Value="Comic Sans MS" />  
  </Style>  
  
</Application.Resources>
```

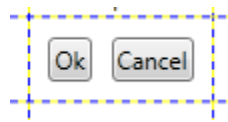
App.xaml.cs

```
public partial class App : Application
{
    public App()
    {
        Startup += App_Startup;
        InitializeComponent();
    }

    void App_Startup(object sender, StartupEventArgs e)
    {
        StyleManager.ApplicationTheme = new Windows7Theme();
        XmlConfigurator.Configure();
        HibernatingRhinos.Profiler.Appender.NHibernate.NHibernateProfiler.Initialize();
    }
}
```

Margins

- Outside spacing
Left, Top, Right, Bottom



```
<StackPanel
    Orientation="Horizontal"
    Grid.Row="3"
    Grid.Column="1"
    Margin="10,10,10,10"
    >

    <Button Content="Ok" Margin="0,0,10,0" />
    <Button Content="Cancel" />

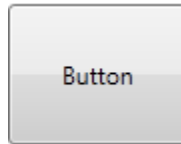
</StackPanel>
```

If all margins are the same, you may use the alternate syntax of `Margin="10"`

Note: Units are Device Independent

Padding

- Inside spacing
Left, Top, Right, Bottom



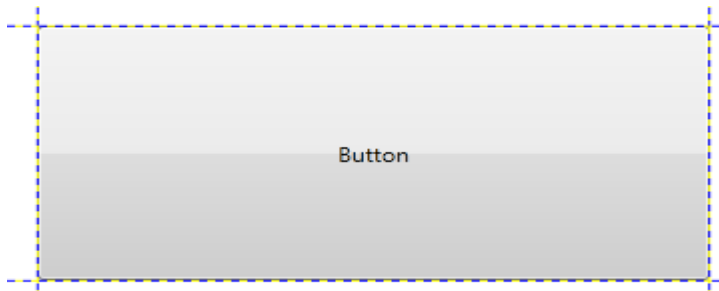
```
<Button Content="Button" Padding="25"
```

If the padding values are the same, you may use the alternate syntax of `Padding="10"`

Note: Units are Device Independent

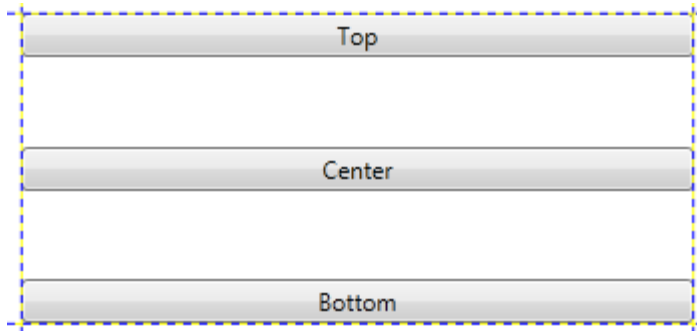
Alignment

- HorizontalAlignment and VerticalAlignment

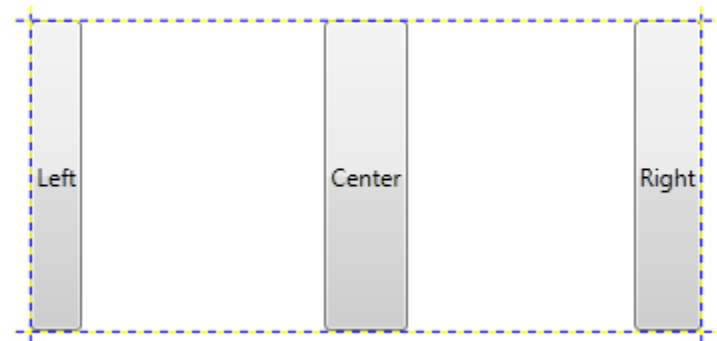


```
<Button Content="Button" />
```

Defaults = "Stretch"



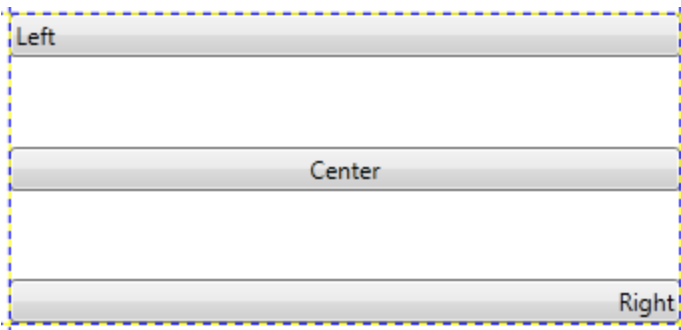
```
<Button Content="Top" VerticalAlignment="Top" />  
<Button Content="Center" VerticalAlignment="Center" />  
<Button Content="Bottom" VerticalAlignment="Bottom" />
```



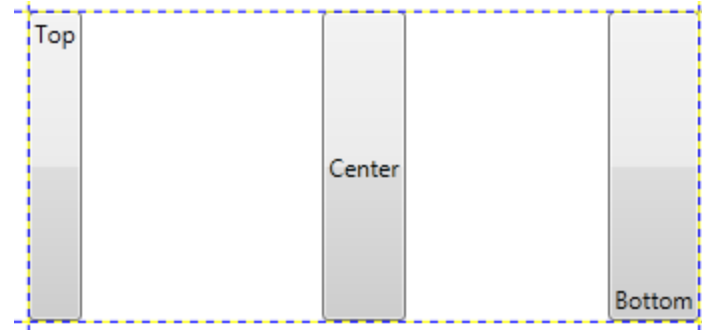
```
<Button Content="Left" HorizontalAlignment="Left" />  
<Button Content="Center" HorizontalAlignment="Center" />  
<Button Content="Right" HorizontalAlignment="Right" />
```

ContentAlignment

- HorizontalContentAlignment and VerticalContentAlignment
 - Applies to Content controls such as buttons



```
<Button Content="Left" HorizontalContentAlignment="Left" />  
<Button Content="Center" HorizontalContentAlignment="Center" />  
<Button Content="Right" HorizontalContentAlignment="Right" />
```

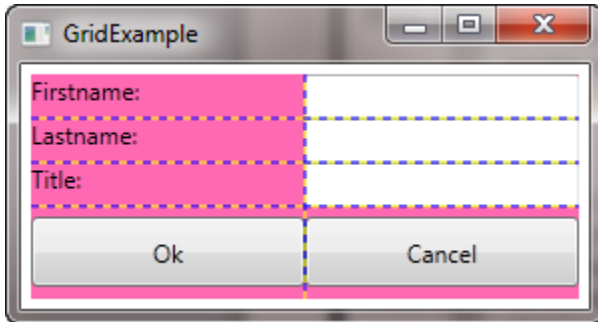


```
<Button Content="Top" VerticalContentAlignment="Top" />  
<Button Content="Center" VerticalContentAlignment="Center" />  
<Button Content="Bottom" VerticalContentAlignment="Bottom" />
```

Control Types

- Layout
- Content
- Text
- List

Grid



- Rows and columns

In almost all cases, the Grid should be your primary method of laying out your views.

```
<Window x:Class="LayoutDemo.GridExample"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="GridExample" Height="160" Width="300">

<Grid
ShowGridLines="True"
Margin="5"
Background="HotPink">

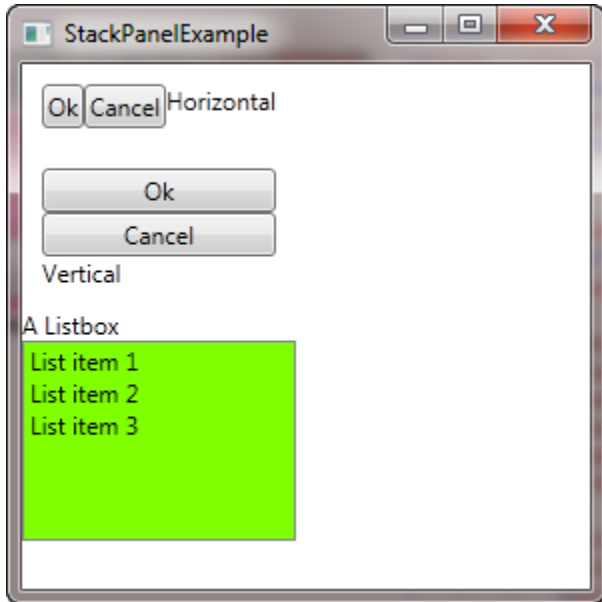
<Grid.RowDefinitions>
<RowDefinition Height="Auto"/>
<RowDefinition Height="Auto"/>
<RowDefinition Height="Auto"/>
<RowDefinition Height="*" />
</Grid.RowDefinitions>

<Grid.ColumnDefinitions>
<ColumnDefinition Width="*" />
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="Firstname:" />
<TextBox Grid.Row="0" Grid.Column="1" />
<TextBlock Grid.Row="1" Grid.Column="0" Text="Lastname:" />
<TextBox Grid.Row="1" Grid.Column="1" />
<TextBlock Grid.Row="2" Grid.Column="0" Text="Title:" />
<TextBox Grid.Row="2" Grid.Column="1" />

<Button Grid.Row="3" Grid.Column="0" Content="Ok" Height="35" />
<Button Grid.Row="3" Grid.Column="1" Content="Cancel" Height="35" />
</Grid>
</Window>
```

StackPanel

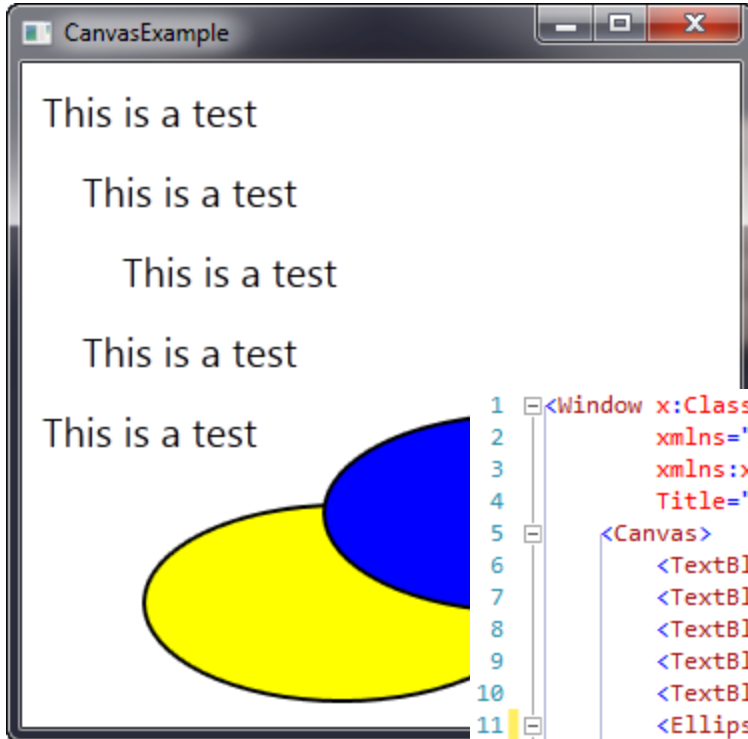


- Vertical or Horizontal orientation

```
1 <Window x:Class="LayoutDemo.StackPanelExample"  
2 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
3 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
4 Title="StackPanelExample" Height="300" Width="300">  
5  
6 <WrapPanel Orientation="Vertical">  
7  
8 <StackPanel Orientation="Horizontal" Margin="10">  
9 <Button Content="Ok"/>  
10 <Button Content="Cancel"/>  
11 <TextBlock Text="Horizontal"/>  
12 </StackPanel>  
13  
14 <StackPanel Orientation="Vertical" Margin="10">  
15 <Button Content="Ok"/>  
16 <Button Content="Cancel"/>  
17 <TextBlock Text="Vertical"/>  
18 </StackPanel>  
19  
20 <StackPanel Orientation="Vertical">  
21 <TextBlock Text="A Listbox" />  
22 <ListBox Height="100" Background="Chartreuse">  
23 <ListBoxItem Content="List item 1"/>  
24 <ListBoxItem Content="List item 2"/>  
25 <ListBoxItem Content="List item 3"/>  
26 </ListBox>  
27 </StackPanel>  
28  
29 </WrapPanel>  
30  
31 </Window>  
32
```

Canvas

- Explicit positioning



```
1 <Window x:Class="LayoutDemo.CanvasExample"
2   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4   Title="CanvasExample" Height="370" Width="375">
5   <Canvas>
6     <TextBlock FontSize="20" Canvas.Left="10" Canvas.Top="10" Text="This is a test"/>
7     <TextBlock FontSize="20" Canvas.Left="30" Canvas.Top="50" Text="This is a test"/>
8     <TextBlock FontSize="20" Canvas.Left="50" Canvas.Top="90" Text="This is a test"/>
9     <TextBlock FontSize="20" Canvas.Left="30" Canvas.Top="130" Text="This is a test"/>
10    <TextBlock FontSize="20" Canvas.Left="10" Canvas.Top="170" Text="This is a test"/>
11    <Ellipse Canvas.Top="220" Canvas.Left="60" Fill="Yellow" Height="100" Width="200"
12      StrokeThickness="2" Stroke="Black"/>
13    <Ellipse Canvas.Top="175" Canvas.Left="150" Fill="Blue" Height="100" Width="200"
14      StrokeThickness="2" Stroke="Black"/>
15  </Canvas>
16 </Window>
17
```

Combining Layouts

The Image in the background is the Parent Grid's background property:

```
<Grid.Background>  
  <ImageBrush ImageSource=" ../images/DefaultSplashScreen.png" />  
</Grid.Background>
```

Parent Grid that hosts all other controls

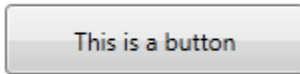
Secondary Grid that holds all of the login-related controls

Horizontal StackPanel that contains the buttons

Content Controls

- A ContentControl can contain content such as text, images, or panels.
- Examples of Content controls include:
 - Button
 - CheckBox
 - RadioButton
 - ListBoxItem
 - Menu
 - Label
 - Many more...

Button



```
<Button Content="This is a button" Height="35" Width="150"/>
```



```
<Button Height="50" Width="200">  
  <Image Source="c:\temp\kalamaxoo_web_sml.png"/>  
</Button>
```



```
<Button Height="36" Width="36" Click="Button_Click">  
  <Image Source="c:\temp\delete.png"/>  
</Button>
```

CheckBox



```
<GroupBox Header="Favorite Conference" Padding="5" Width="200" Height="100">  
  <StackPanel Orientation="Vertical">  
    <CheckBox>  
      <Image Source="c:\temp\kalamaxoo_web_sml.png" Height="35"/>  
    </CheckBox>  
    <CheckBox>  
      <Image Source="c:\temp\MADExpoLogo1.png" Height="35"/>  
    </CheckBox>  
  </StackPanel>  
</GroupBox>
```

Text Controls

- **TextBox** – Used for entering text.

```
<TextBox Width="200" Height="25" MaxLength="50"/>
```

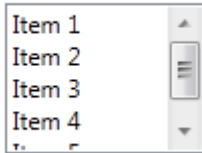
- **TextBlock** – Used for displaying static text.

This is static text that can't be changed

```
<TextBlock Text="This is static text that can't be changed"/>
```

List Controls

- ListBox



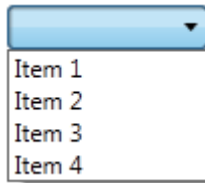
```
<ListBox Width="100" Height="75" Margin="0,0,0,25">  
  <ListBoxItem Content="Item 1"/>  
  <ListBoxItem Content="Item 2"/>  
  <ListBoxItem Content="Item 3"/>  
  <ListBoxItem Content="Item 4"/>  
  <ListBoxItem Content="Item 5"/>  
</ListBox>
```



```
<ListBox Width="125" Height="205" Margin="0,0,0,25">  
  <ListBoxItem>  
    <Image Source="C:\temp\Desert.jpg" Height="75" Width="75" />  
  </ListBoxItem>  
  <ListBoxItem>  
    <Image Source="C:\temp\lighthouse.jpg" Height="75" Width="75" />  
  </ListBoxItem>  
  <ListBoxItem>  
    <Image Source="C:\temp\Tulips.jpg" Height="75" Width="75" />  
  </ListBoxItem>  
</ListBox>
```

List Controls

- ComboBox



```
<ComboBox Width="100" Margin="0,0,0,25">  
  <ComboBoxItem Content="Item 1"/>  
  <ComboBoxItem Content="Item 2"/>  
  <ComboBoxItem Content="Item 3"/>  
  <ComboBoxItem Content="Item 4"/>  
</ComboBox>
```



```
<ComboBox Width="100" Margin="0,0,0,25">  
  <ComboBoxItem>  
    <Image Source="C:\temp\tulips.jpg" Height="75" Width="75"/>  
  </ComboBoxItem>  
  <ComboBoxItem>  
    <Image Source="C:\temp\desert.jpg" Height="75" Width="75"/>  
  </ComboBoxItem>  
</ComboBox>
```

Data Binding

- Allows us to connect data to an element on the screen
 - Extremely powerful
 - Data can come from many different sources
 - Silverlight doesn't care what the source is
 - Used everywhere
 - Enables MVVM

Data Binding

- DataContext
 - provides a convenient way to establish a data scope
 - multiple targets can share the same source

Data Binding

- A typical data binding assignment in XAML
`<TextBox Text="{Binding FirstName}"/>`
 - The data source is typically set using the `.DataContext` property of a parent object.
 - Typically done via the code-behind unless you're using MVVM

INotifyPropertyChanged

- Implements a single event

```
public event PropertyChangedEventHandler PropertyChanged;
```

- Requires a helper method to raise the event. Typical implementation:

```
public void OnPropertyChanged(string propertyName)
{
    if (PropertyChanged != null)
    {
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

- Normally implemented in a base class

INotifyPropertyChanged

- Each property of an object that is bound must raise this event in its 'setter'

```
private string _FirstName;
public string FirstName
{
    get { return _FirstName; }
    set
    {
        _FirstName = value;
        OnPropertyChanged("FirstName");
    }
}
```

Collections

- `ObservableCollection<T>`
 - notification when items are added or removed.
- Assign to `.ItemsSource` of `ListBox`, `ComboBox`, `DataGrid`, etc.
 - Once you've assigned `.ItemsSource`, you cannot manually add items

StringFormat

- Can be used to format a string directly in a binding statement

```
<TextBox Text="{Binding HireDate, StringFormat=\{0:MM/dd/yyyy\}}"/>
```

```
<TextBox Text="{Binding Amount, StringFormat=Amount: \{0:C\}}"/>
```

DataBinding Errors

- Bindings are **case-sensitive**
 - If a binding isn't working, check the output window in Visual Studio.
 - Look for errors that start with "System.Windows.Data Error: 40 : BindingExpression"

DataBinding Demo

Styles and Templates

- Controls the look and feel of an application
- Can completely change how a control looks

Styles

- Can be applied to any object which derives from FrameworkElement or FrameworkContentElement, both of which expose a public property named Style.
- Local / global
- Defines the appearance of a control

Styles

- Styles can live in several places
 - App.xaml
 - <Application.Resources>
 - Available globally
 - <Window | Page | UserControl.Resources/>
 - Available only within that particular window, page or usercontrol
 - <Control.Resources>
 - <Button.Resources>, <Grid.Resources>, etc.
 - Only available within the context of that particular control



This is a button

```
<Style TargetType="Button">  
  <Setter Property="FontSize" Value="24"/>  
  <Setter Property="FontFamily" Value="Comic Sans MS"/>  
  <Setter Property="Foreground" Value="Red"/>  
</Style>
```

Contact Info

- Email: mjeaton@validussolutions.com
- Blog: <http://mjeaton.net/blog>
- Twitter: @mjeaton