Reusable UI

Outline

- Resource reuse
- Template reuse
- Xaml reuse
- Custom elements
- Custom controls

Resource Reuse

- **Not suitable for UI tree elements**
- **Good for ‘freezables’**
  - Brushes, pens
  - Geometries
  - Drawings
  - Animations
  - 3D model elements

```
<StackPanel>
  <StackPanel.Resources>
    <Button x:Key="btn">
      _Click me
    </Button>
  </StackPanel.Resources>

  <!-- Error on 2nd usage -->
  <StaticResource ResourceKey="btn" />
  <StaticResource ResourceKey="btn" />

</StackPanel>
```

Template Reuse

- **Template is a factory**
  - Generates instance for each use

```
<StackPanel>
  <StackPanel.Resources>
    <ControlTemplate x:Key="btn">
      <Button>
        _Click me
      </Button>
    </ControlTemplate>
  </StackPanel.Resources>

  <Control Focusable="False"
     Template="{StaticResource btn}" />
  <Control Focusable="False"
     Template="{StaticResource btn}" />
</StackPanel>
```

Xaml File Reuse

- Application.LoadComponent

Or…

- Host in Frame or NavigationWindow

Xaml with Codebehind

```xml
<Grid x:Class="MyNamespace.MyXamlType">
  ...
</Grid>
```

- **Just use 'new'**

```csharp
MyXamlType o = new MyXamlType();
```

- **Use as custom element**

```xml
<StackPanel xmlns:c="clr-namespace:MyNamespace">
  ...
  <c:MyXamlType />
  ...
</StackPanel>
```

UserControl

- **Simplest way to build control**
  - XAML + codebehind
  - Derives from ContentControl
- **Limited functionality**
  - Mostly just a well-known base class
- **No Template support**

Xaml : One per Class

- **Inheritance chain cannot use Xaml twice**
- **Application.LoadComponent limitation**
  - Field and event hookup

Runtime Xaml Parsing

- **XamlReader.Load**
  - Xaml stream or XmlReader

```
XmlReader rdr = GetSomeXaml();
object rootOfXaml = XamlReader.Load(rdr);

myPanel.Children.Add(rootOfXaml);
```

Custom Elements

- **Derive directly from non-Control element type**

Base Type	Usage
Decorator	Applying effects or chrome to a single child
Adorner	Drag handles, selection outlines etc.
Panel	Arranging children
Shape	Custom shapes
FrameworkElement	When no other base class fits

Custom Controls

- Only appropriate if custom behavior needed

API Considerations

- Properties
- Events
- Commands
- Contract for template
  - (Controls only)

Properties

```
public double Pointyness
{
    get { return (double) GetValue(PointynessProperty); }
    set { SetValue(PointynessProperty, value); }
}

public static readonly DependencyProperty PointynessProperty =
   DependencyProperty.Register("Pointyness", typeof(double),
     typeof(PointyControl), new UIPropertyMetadata(3));
```

Commands

```
public static readonly ICommand HideCommand =
    new RoutedUICommand("Hide", "Hide",
                        typeof(ShyControl));
```

```
static ShyControl()
{
    CommandManager.RegisterClassCommandBinding(
        typeof(ShyControl),
        new CommandBinding(HideCommand, OnHideCommand));
}

private static void OnHideCommand(object sender,
                          ExecutedRoutedEventArgs e)
{
    ShyControl target = (ShyControl) sender;
    ...
}
```

Events: Defining

```
public class BombControl : Control
{
  public static readonly RoutedEvent ExplodedEvent =
     EventManager.RegisterRoutedEvent("Exploded",
        RoutingStrategy.Bubble, typeof(RoutedEventHandler),
        typeof(BombControl));

  ...

  public event RoutedEventHandler Exploded
  {
    add { AddHandler(ExplodedEvent, value); }
    remove { RemoveHandler(ExplodedEvent, value); }
  }

  private void OnExploded()
  {
    RoutedEventArgs e = new RoutedEventArgs(ExplodedEvent);
    RaiseEvent(e);
  }
}
```

pluralsight
see what you can learn

Events: Handling

```
static BombControl()
{
    EventManager.RegisterClassHandler(
        typeof(BombControl), Mouse.MouseDownEvent,
        new EventHandler(OnMouseDown));
}

static void OnMouseDown(object sender,
                        MouseButtonEventArgs e)
{
    BombControl target = (BombControl) sender;
    ...
}
```

Contract With Templates

```
[TemplatePart(Name="PART_Fuse", typeof(FrameworkElement))]
[TemplatePart(Name="PART_Body", typeof(ContentControl))]
public class BombControl : Control
{
    ...
```

Themes

- **Compiled Xaml resources in Themes folder:**
  - generic.xaml
  - Aero.NormalColor.xaml
  - Luna.NormalColor.xaml, Luna.Homestead.xaml, Luna.Metallic.xaml
  - Classic.xaml
  - Royale.NormalColor.xaml

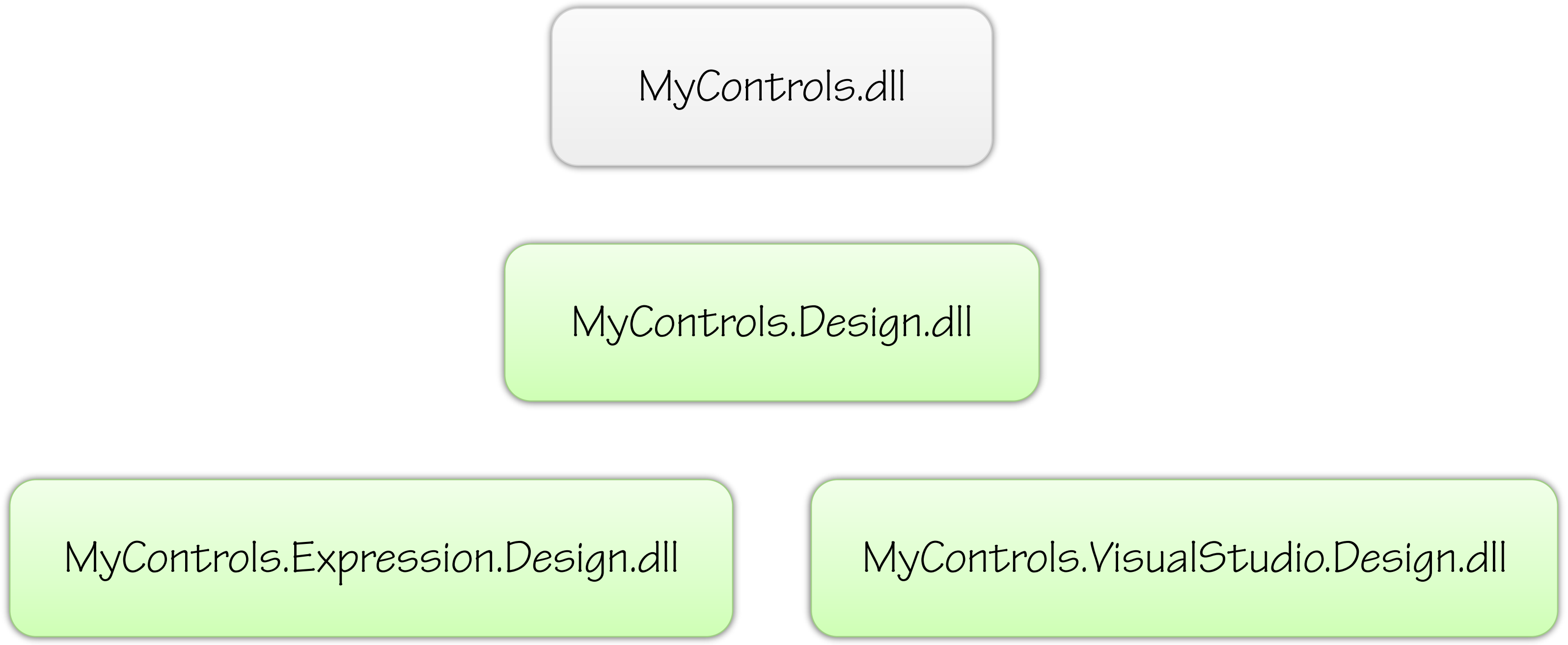
```
[assembly:ThemeInfo(

    // Themed resources
    ResourceDictionaryLocation.SourceAssembly,

    // Generic resources
    ResourceDictionaryLocation.SourceAssembly
)]
```

Designer Integration

- **Separate assemblies**


MyControls.dll
MyControls.Design.dll
MyControls.Expression.Design.dll
MyControls.VisualStudio.Design.dll

Designer Extensibility Options

- **Any design environment**
  - Toolbox visibility
- **Visual Studio 2008**
  - Custom adorners
  - Context menu
  - Object tree
  - Property grid integration

Metadata Providers

```
public class MyControlVsMetadata : IRegisterMetadata
{
  public void Register()
  {
    AttributeTableBuilder builder = new AttributeTableBuilder();

    builder.AddCustomAttributes(
        typeof(MyControl),
        new FeatureAttribute(typeof(MyControlAdornerProvider)));

    MetadataStore.AddAttributeTable(builder.CreateTable());
  }
}
```

Summary

- Resource reuse
- Template reuse
- Xaml reuse
- Custom elements
- Custom controls