Custom Controls in WPF

Intro

- WPF is so flexible that you don't even need to bother building custom controls
- You can customize the look and feel of virtually every aspect of every control, as well as easily changing much of the behavior
  - By setting the content to some sort of panel …
  - Or by specifying a custom control template.

- But there are still situations where it's more convenient to build a custom control once and reuse it.
  - User controls is very popular among developers who uses the MVVM pattern.

Alternatives to Writing a New Control

- Rich Content.
  - Many of the standard WPF controls support rich content.
  - E.g. the content property of a Button is of type Object, so theoretically anything can be displayed on a Button.
  - To have a button display an image and text, you can add an image and a TextBlock to a StackPanel and assign the StackPanel to the Content property.
  - Because controls can display WPF visual elements and arbitrary data, there is less need to create a new control or to modify an existing control to support a complex visualization.
- Styles.
  - A Style is a collection of values that represent properties for a control.
  - By using styles, you can create a reusable representation of a desired control appearance and behavior without writing a new control.

Alternatives to Writing a New Control

- Data Templates.
  - A DataTemplate enables you to customize how data is displayed on a control. For example, a DataTemplate can be used to specify how data is displayed in a ListBox.
  - In addition to customizing the appearance of data, a DataTemplate can include UI elements, which gives you a lot of flexibility in custom UIs.

- Control Templates.
  - Many controls in WPF use a ControlTemplate to define the control's structure and appearance, which separates the appearance of a control from the functionality of the control.
  - You can drastically change the appearance of a control by redefining its ControlTemplate.
  - For example, suppose you want a control that looks like a stoplight. This control has a simple user interface and functionality. The control is three circles, only one of which can be lit up at a time. After some reflection, you might realize that a RadioButton offers the functionality of only one being selected at a time, but the default appearance of the RadioButton looks nothing like the lights on a stoplight. Because the RadioButton uses a control template to define its appearance, it is easy to redefine the ControlTemplate to fit the requirements of the control, and use radio buttons to make your stoplight.

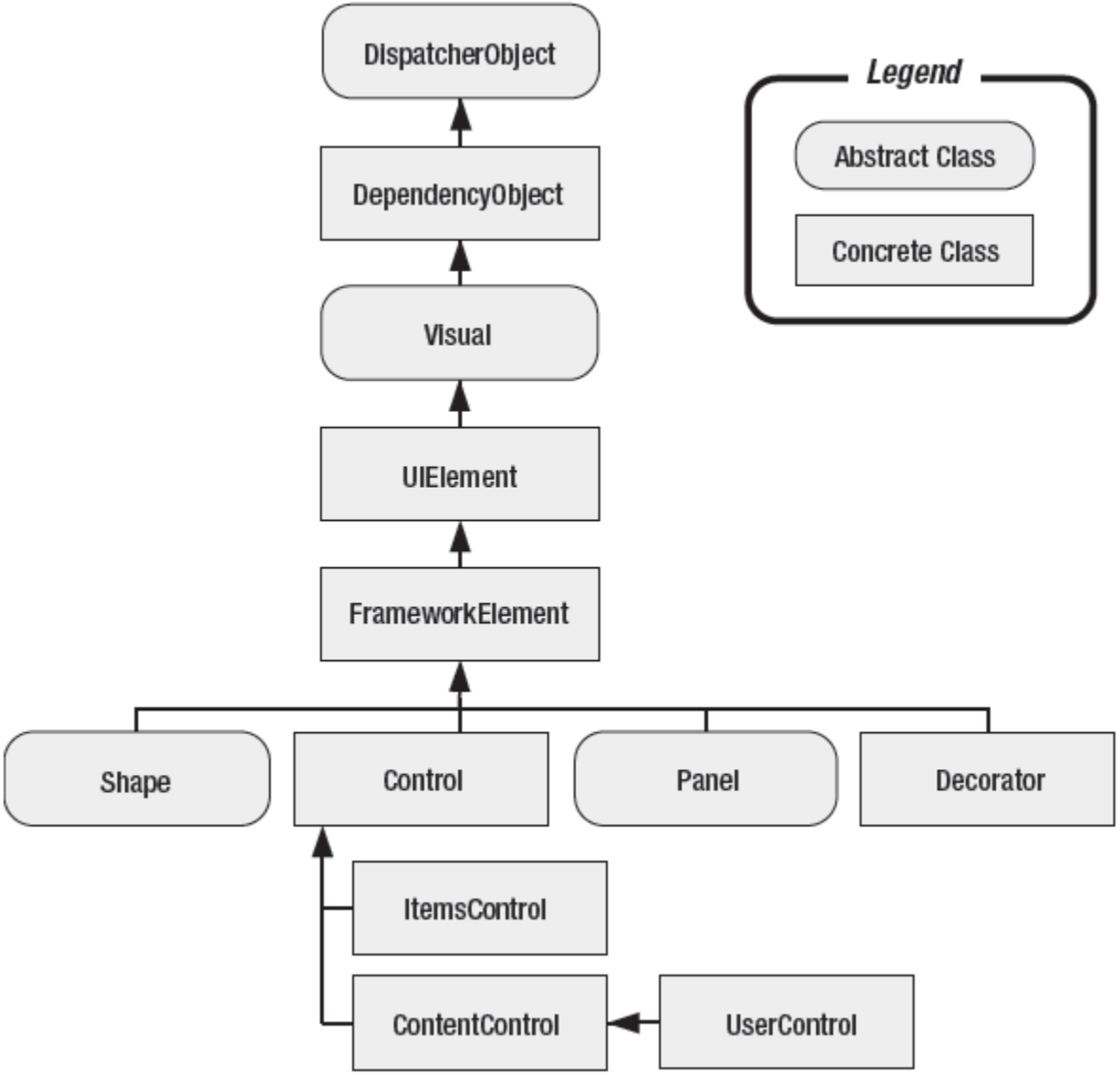
WPF Custom Controls

- WPF provides three general models for creating a control, each of which provides a different set of features and level of flexibility.
- The base classes for the three models are:
  - UserControl
    Are the simplest type of control.
    Usually, **user controls combine more than one control in a logical unit** (like a group of text boxes for entering address information).
  - Control (~ Custom Controls aka look less controls)
    Are generally more powerful and flexible.
    When you create a control that inherits from the Control class, you define its appearance by using templates.
  - FrameworkElement (~Owner-drawn controls - OnRender)
    Require the most work.
    Use when you need to do something more low-level, where you do your own drawing and interaction behavior.

UserControls

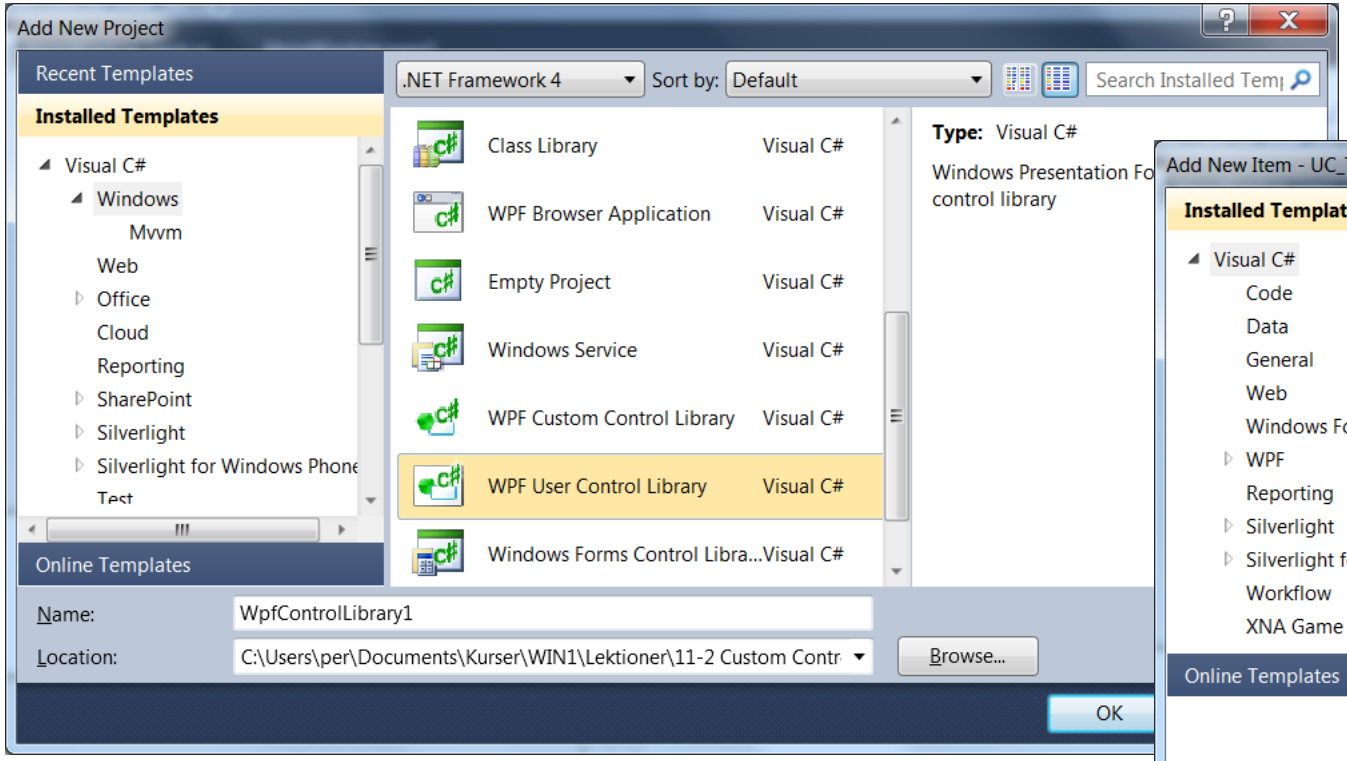
- The simplest way to create a control in WPF is to derive from User Control.
- When you build a control that inherits from User Control, you add existing components to the User Control, name the components, and reference event handlers in Extensible Application Markup Language (XAML).
- You can then reference the named elements and define the event handlers in code.
  - This development model is very similar to the model used for application development in WPF.
- If built correctly, a User Control can take advantage of the benefits of rich content, styles, and triggers.
  - But people who use your control will not be able to use a Data Template or Control Template to customize its appearance.

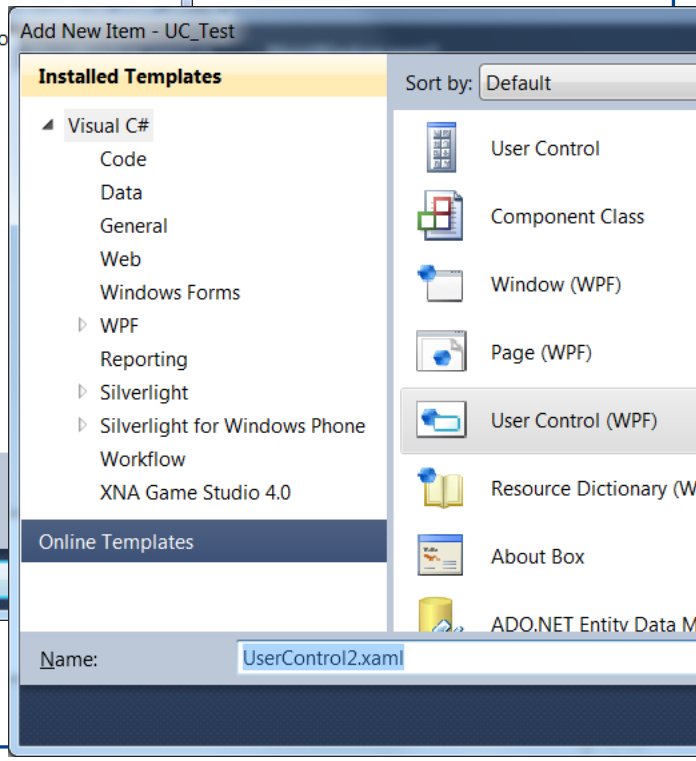
Control Hierarchy



UserControls How to

- In Visual Studio just select Add New Project, and then select “WPF User Control Library”
- Or just Add New User Control inside a WPF Application project.





UserControls How to

CustomControls - Microsoft Visual Studio

File Edit View Modeling Project Build Debug Data Format Tools Test ReSharper Window Help

Debug | Any CPU | MyCustomDll.dll

Toolbox

- Common
- Controls
  - Pointer
  - Border
  - Button
  - Canvas
  - CheckBox
  - ComboBox
  - ContentControl
  - DockPanel
  - DocumentViewer
  - Ellipse
  - Expander
  - Frame
  - Grid
  - GridSplitter
  - GroupBox
  - Image
  - Label
  - ListBox
  - ListView
  - MediaElement
  - Menu

UserControl1.xaml*

Address

Address

City

Postal

Design | XAML

```
<UserControl x:Class="UC_demo.UserControl1
    xmlns="http://schemas.microsoft.com/wi
    xmlns:x="http://schemas.microsoft.com/
    Height="150" Width="339">
    <Grid >
```

UserControl UserControl

Solution Explorer - Soluti...

- Solution 'CustomControls' (7 p
  - ControlsInAction
    - Properties
    - References
    - App.xaml
    - LinkLabel.xaml
    - Window1.xaml
    - Window2.xaml
  - Demo
  - GraphingWithShapes
  - GroupedCheckBoxApp
  - MyWpfUserControlLibrary
  - UC_demo

Soluti... | Class... | Docu...

Properties

System.Windows.Contro...

Name:

Clear

Background	
BorderBrush	
Foreground	Black
OpacityMask	
Common Properties	
Cursor	

Error List

0 Errors | 0 Warnings | 0 Messages

Description	File	Line	Column	Project

Control ContentPresenter Algorithm

1. If Content is of type UIElement, then add it to the display tree.
2. If ContentTemplate/ItemTemplate is set, use that to create a UIElement instance and add it to the display tree.
3. If ContentTemplateSelector is set, use that to find a template, use the template to create a UIElement instance, and add it to the display tree.
4. **If the data type of Content has a data template associated with it, use that to create a UIElement instance.**
5. If the data type of Content has a Type
that can convert to type UIElement
tree.
6. If the data type of Content has a TypeConverter
with it that can convert to a string, wrap Content in TextBlock and add it to the display tree.
7. Finally, call ToString on Content, wrap it in TextBlock, and add it to the display tree.

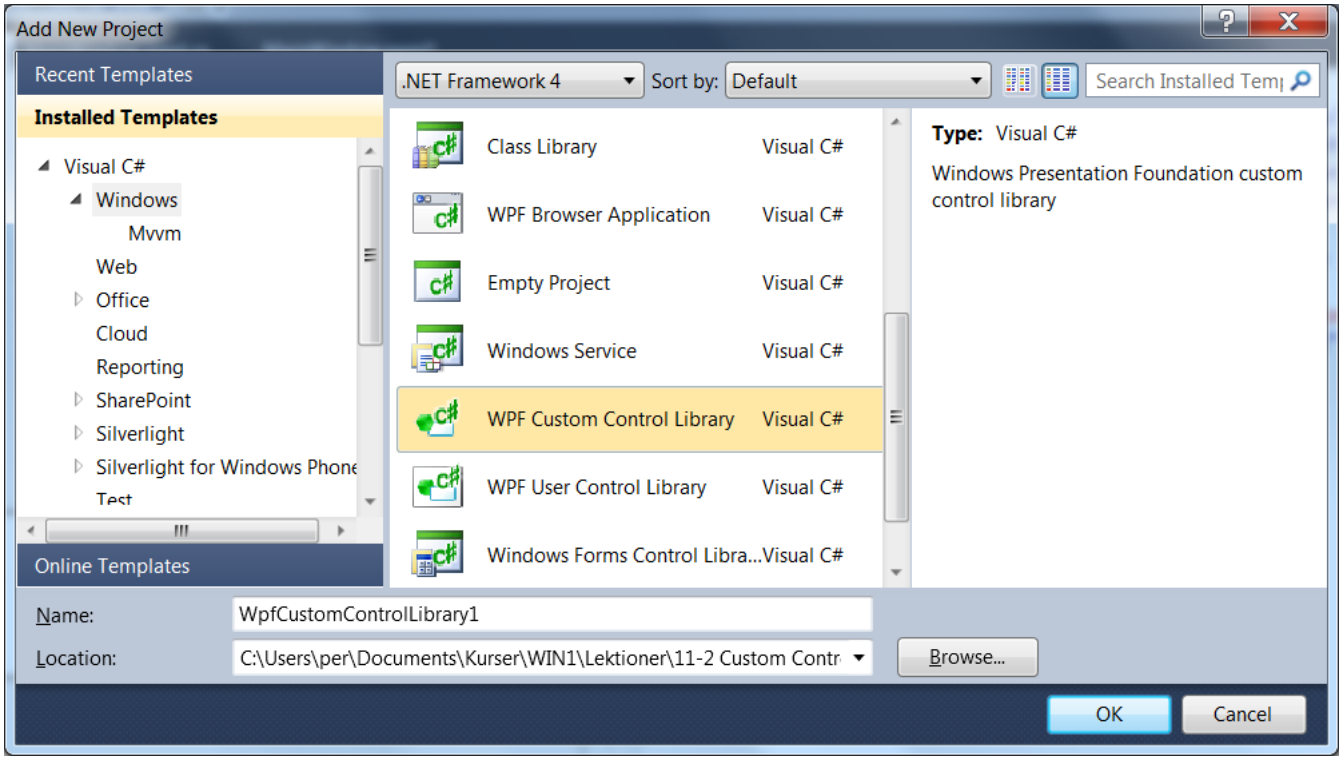
Can be used to connect a view to viewModel.

CUSTOM CONTROLS
AKA LOOK LESS CONTROLS

Used when you want to be able to modify the View part of the control later.

Look-less Controls How to

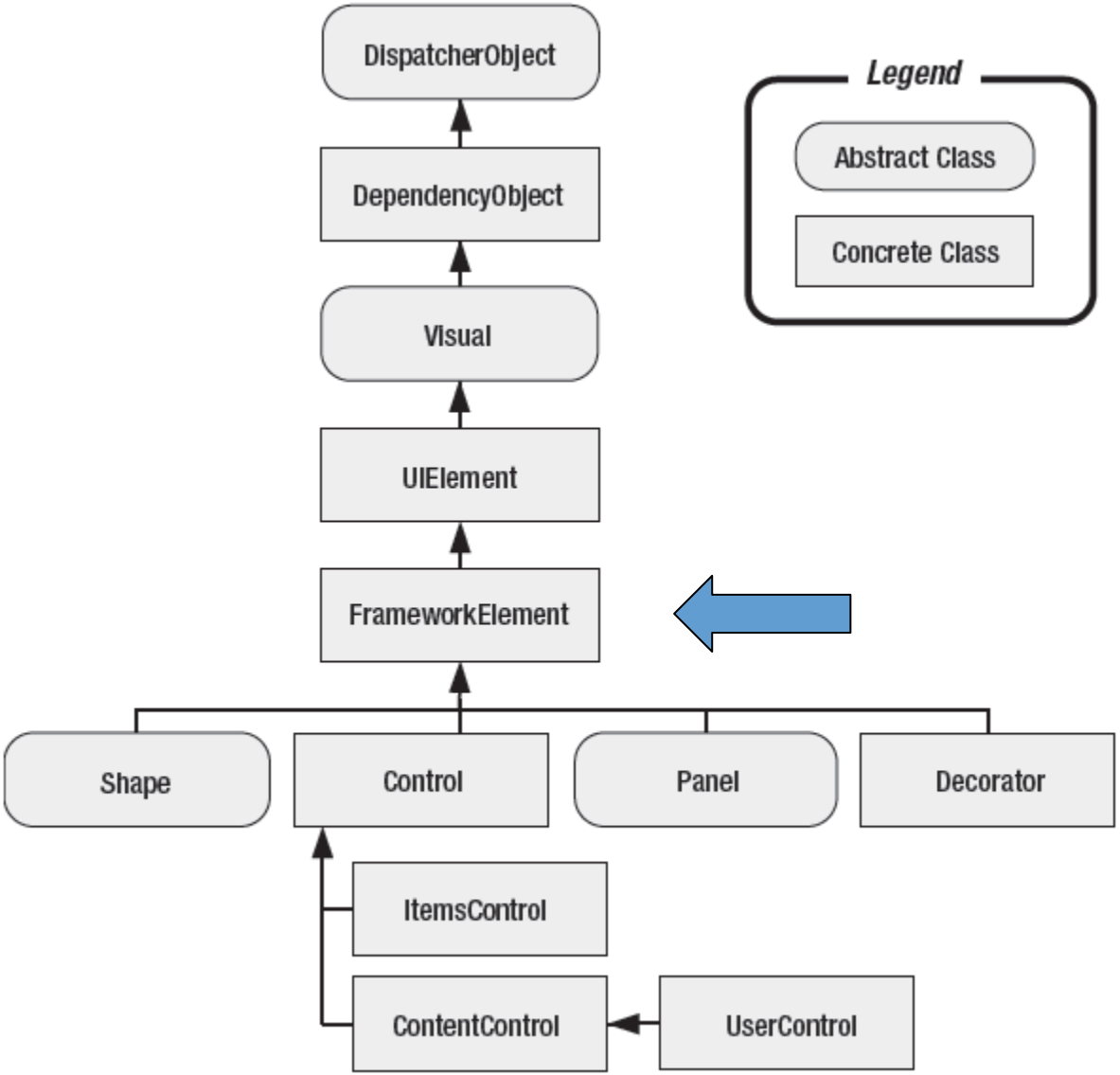
- Create a WPF Custom Control Library.
- Places its markup into a default template that can be replaced at will without disturbing the control logic.



OWNER-DRAWN CONTROLS

Custom-Drawn Elements

Control Hierarchy



FrameworkElement (~Owner-drawn controls)

- To perform custom rendering, an element must override the OnRender() method, which is inherited from the base UIElement class.
- The OnRender() method receives a DrawingContext object, which provides a set of useful methods for drawing content.
- The OnRender() method doesn't necessarily replace composition.

```
protected override void OnRender(DrawingContext drawingContext)
{
  this.EnsureRenderedGeometry();
  if (this._renderedGeometry != Geometry.Empty)
  {
    drawingContext.DrawGeometry(this.Fill, this.GetPen(),
                                        this._renderedGeometry);
  }
}
```

References

- MSDN - Control Authoring Overview
  http://msdn.microsoft.com/en-us/library/ms745025.aspx