


Separate Your Concerns with MVVM in WPF and Silverlight

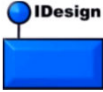
Brian Noyes
www.idesign.net

©2010 Brian Noyes, IDesign Inc. All rights reserved




About Brian

Chief Architect IDesign Inc. (www.idesign.net)	Publishing <i>Developing Applications with Windows Workflow Foundation, LiveLessons training DVD, June 2007</i>
Microsoft Regional Director (www.theregion.com)	<i>Smart Client Deployment with ClickOnce, Addison Wesley, January 2007</i>
Microsoft MVP Connected Systems	<i>Data Binding in Windows Forms 2.0, Addison Wesley, January 2006</i>
E-mail: brian.noyes@idesign.net	Speaking <i>Microsoft TechEd US, Europe, Malaysia, Visual Studio Connections, DevTeach, INETA Speakers Bureau, MSDN Webcasts</i>
Blog: http://briannoyes.net	



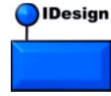
Agenda

- [Quick Intro to WPF Data Binding](#)
- Model-View-ViewModel Pattern
- Commands and MVVM



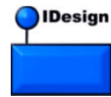
DataContext

- Dependency property defined on FrameworkElement
 - Inherited
 - ▲ Flows down the element hierarchy to child elements
- Provides ambient data source for an element and its children
- May be set to a collection or an individual object
- Default data source for bindings on element and its children
- At any element in the visual tree, there is only one data context



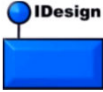
Bindings

- Associate property on data source object to property on target element
- Example
 - Text property on TextBox
- Source object is the DataContext if not explicitly specified
 - If DataContext is collection, current item is source for single valued target properties
- Source object can be any type that exposes properties
- Source object should raise change notifications if changes are made behind the scenes
 - INotifyPropertyChanged / INotifyCollectionChanged
- Target object must be a DependencyObject




Data Templates

- Chunk of UI (XAML) that you associate with a data object through data binding
- The visual rendering of a data type in a given context
- Similar to ASP.NET repeater, ListView, etc templates
- Keeps the UI definition loosely coupled from the data
- Sets the DataContext on the root UI element to the data object



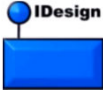
Agenda

- Quick Intro to WPF Data Binding
- [Model-View-ViewModel Pattern](#)
- Commands and MVVM



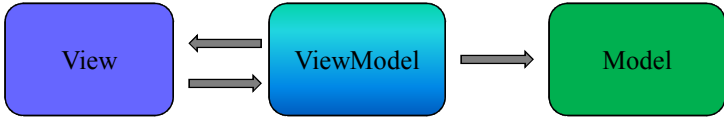
Why use separated UI patterns?

- Design with:
 - Loose coupling
 - Separation of concerns
- Benefits
 - Presentation logic code (ViewModel, Controllers, etc) becomes more testable
 - Designers and developers can work more independently in view/ViewModel respectively
 - More maintainable – easier to make changes to view or ViewModel without affecting each other



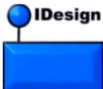
Presentation Model / ViewModel

- Model-View-ViewModel
- MVVM or ViewModel for short
- Specialized implementation of Presentation Model pattern for WPF and Silverlight



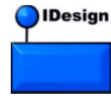
```
graph LR; View[View] <--> ViewModel[ViewModel]; ViewModel --> Model[Model];
```

The diagram illustrates the MVVM pattern. It consists of three rounded rectangular boxes: a blue box labeled 'View' on the left, a cyan box labeled 'ViewModel' in the center, and a green box labeled 'Model' on the right. A double-headed arrow connects the View and ViewModel boxes, indicating bidirectional communication. A single-headed arrow points from the ViewModel box to the Model box, indicating that the ViewModel interacts with the Model.



ViewModel

- Offers up state to the view through simple properties
 - Facilitates simple binding expressions in the view
- Notifies view of property changes
 - INotifyPropertyChanged / INotifyCollectionChanged
 - On ViewModel or its exposed property types
- Encapsulates interaction logic to support the view
 - Initial load of data
 - Command handlers
 - Service invocation
 - Validation logic
 - Value conversion



View-ViewModel Relationship

- ViewModel should be loosely coupled to the view
- Relationship with the view
 - 1:1 – most common
 - 1 view : multiple ViewModels (example: Add/Edit)
 - Multiple views : 1 ViewModel (example: listing/details views)



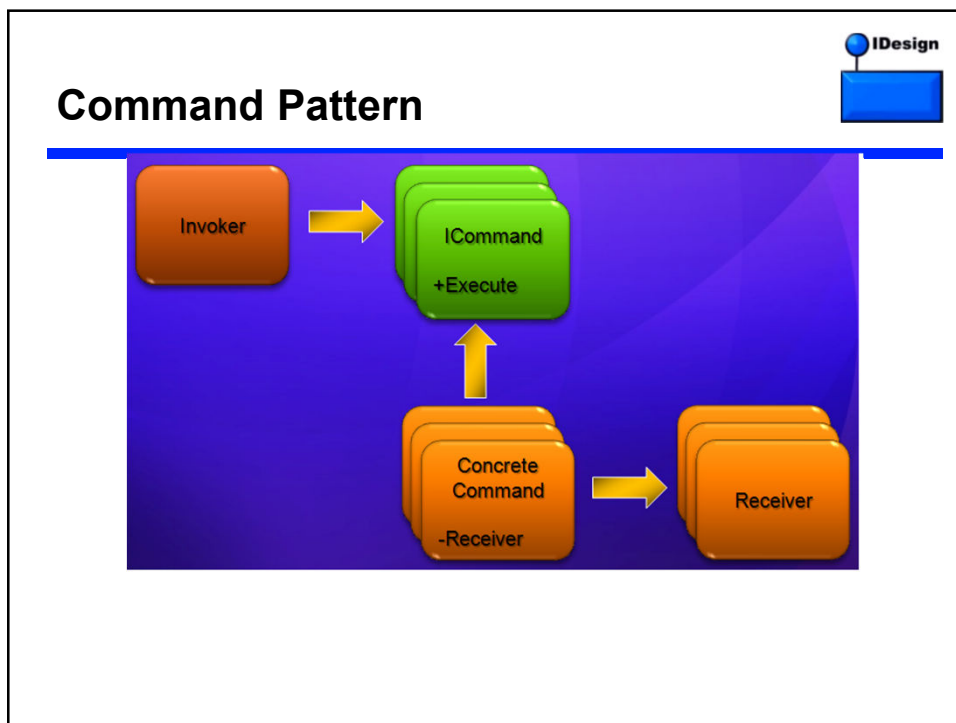
ViewModel in WPF/Silverlight

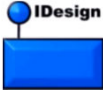
- View.DataContext = ViewModel
- POCO (Plain old CLR object)
 - No required base classes
 - However, often factor out common code into base class
- Public properties for view controls to bind to
 - Properties may expose model objects directly
 - Properties may be special data just for display purposes
 - Properties for commands
- Initialization / interaction logic
- “Married” to the view through construction code
 - Set as DataContext for the view
 - Several approaches

IDesign

Agenda


- Quick Intro to WPF Data Binding
- Model-View-ViewModel Pattern
- [Commands and MVVM](#)





Commands

- WPF / Silverlight 4 Infrastructure
 - ICommand / ICommandSource interfaces
 - ICommandSource implementations
 - ▲ Buttons, menu items, and hyperlinks
- WPF Routed Commands
 - Implementation of ICommand
 - Tightly coupled to the UI
 - Insufficient for separated UI patterns
- Custom commands
 - Decouple the view (command sources/invokers) from the command targets
 - ▲ ViewModels
 - ▲ Controllers
 - Prism DelegateCommand / CompositeCommand
 - MVVM Lite/Caliburn/MEFedMVVM RelayCommand



Resources

- Data Binding and Commands
 - Programming WPF 2nd Edition, Ian Griffiths & Chris Sells, O'Reilly & Associates, 2007. <http://oreilly.com/catalog/9780596510374>
 - Data Binding in WPF, John Papa, MSDN Magazine, Dec 2007, <http://msdn.microsoft.com/en-us/magazine/cc163299.aspx>
- MVVM
 - WPF Apps with the Model-View-ViewModel Pattern, Josh Smith, MSDN Magazine, Feb 2009, <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>
 - Model-View-ViewModel in Silverlight 2 Apps, Shawn Wildermuth, MSDN Magazine, March 2009, <http://msdn.microsoft.com/en-us/magazine/dd458800.aspx>
 - Advanced MVVM, Josh Smith, <http://joshsmithonwpf.wordpress.com/advanced-mvvm/>