# ViewModel Pattern and Interactivity in Silverlight

**Nikhil Kothari**
@nikhilk, http://www.nikhilk.net

Software Architect, Microsoft Corporation

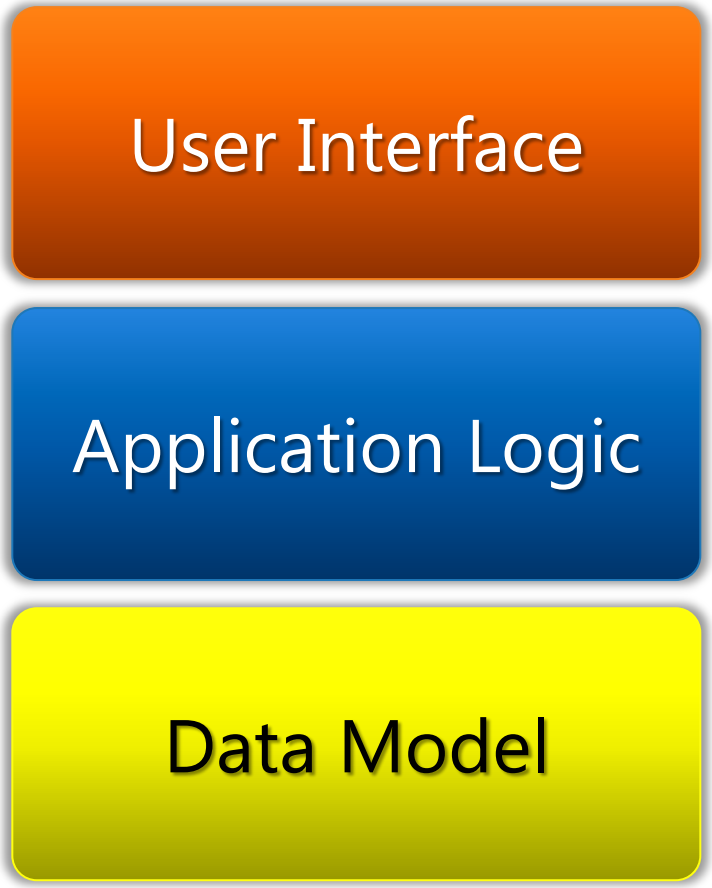
# Agenda

- Introduction to ViewModel pattern (MVVM)
- ViewModel pattern as an enabler
  - Improving testability
  - Facilitating designer-developer workflow
- Supporting patterns
  - Triggers, actions and behaviors for declarative interactivity

# Client Applications and Silverlight

- Interesting trends
  - Leveraging new platform capabilities and data services
  - Scaling across multiple devices
  - Interactive and immersive UX
  - Makes patterns for managing complexity and delivering on UX expectations interesting
- Silverlight
  - .NET and XAML-based client application platform
  - Consistent development model across in-browser, out-of-browser, on-device scenarios
  - End-to-end tooling

# Patterns for Separating UI and Logic




- MVC, MVP, MVVM aka ViewModel aka Presentation Model
- General idea is to decouple application logic from user interface
  - Separation of concerns
- Motivations
  - Maintainability, reusability, testability
  - Designer-developer workflow

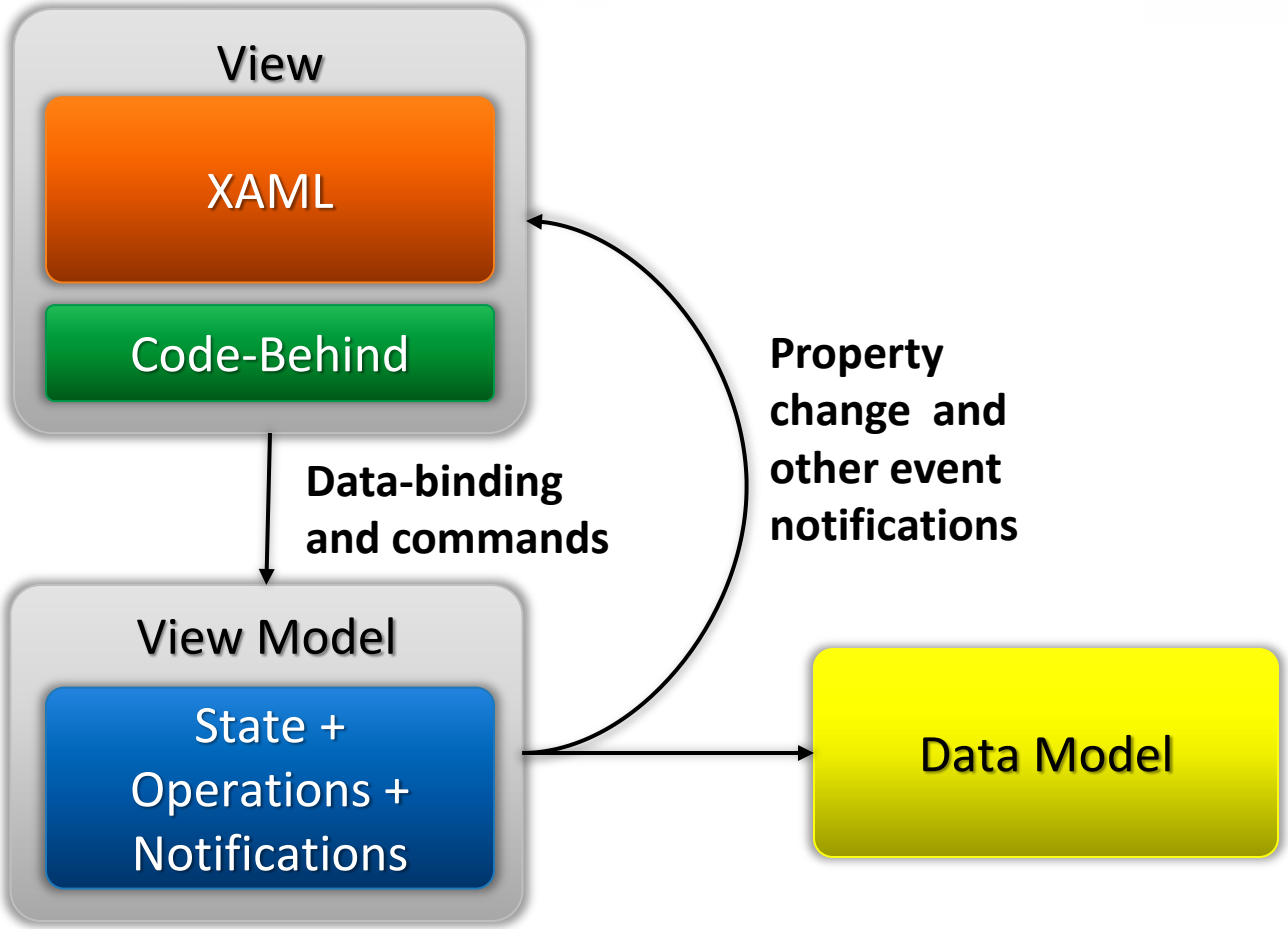
# ViewModel Pattern

- MVVM (**M**odel – **V**iew – **V**iew **M**odel)
  - An adaption of MVC optimized for a Silverlight-based active view
- Basic concepts
  - View model encapsulates application state, logic, and access to data or back-end services
  - View handles presentation, including rendering and user input
  - Data-binding and commanding provide the glue to stitch them together
- Opinions creep in as pattern is transformed into practice

# Demo

Hello ViewModel

# Recap: ViewModel



# Recap: Markup and Code

## XAML

```
<UserControl>
  <UserControl.DataContext>
    <app:MainPageViewModel />
  </UserControl.DataContext>
  <TextBlock Text="{Binding ScreenName}" />
  <TextBox Text="{Binding Tweet, Mode=TwoWay}" />
  <Button Command="{StaticResource postCommand}" />
</UserControl>
```

## View Model

```
public class MainPageViewModel : Model {
    public string ScreenName { get; }
    public string Tweet { get; set; }
    public bool CanPost { get; }
    public void Post() { … }
}
```

# Testability

- View and functional testing is relatively expensive
  - Doesn't lend itself well to unit testing
- ViewModel pattern improves testability
  - View model contains the interesting state and logic
  - Focus on testing the view model by simulating user and mocking dependencies

# Demo

## Implementing View Model Tests using the Silverlight Unit Test Framework

# Declarative Interactivity

- Goes hand-in-hand with ViewModel as a supporting pattern
  - An approach to implementing UI logic in declarative manner
- Three building blocks
  - Triggers – when an *event* occurs
  - Actions – perform the specified *activity*
  - Behaviors – reusable encapsulations of one or more pairs of triggers and actions

# Demo

## Implementing and Using Triggers, Actions and Behaviors

# Recap: Triggers, Actions and Behaviors

```
<UserControl>
  <i:Interaction.Triggers>
    <app:ModelEvent EventName="TweetPosted">
      <app:PlaySound Sound="/Assets/Tweet.mp3" />
    </app:ModelEvent>
  </i:Interaction.Triggers>

  <TextBox Text="{Binding Tweet, Mode=TwoWay}">
    <i:Interaction.Behaviors>
      <app:ImmediateCommit />
    </i:Interaction.Behaviors>
  </TextBox>
</UserControl>
```

# Designer/Developer Workflow

- Code-behind creates contention
  - No separation of concerns, XAML mixed up with app logic
- ViewModel pattern facilitates better designer/developer workflow
  - View model becomes the contract between designer and developer
  - Designer can focus on the XAML half
  - Developer can focus on the implementation of view model
  - Bindings and commands enable integration
- Fits in well with sketching/storyboarding based prototyping

# Demo

View Model as a Contract

# ViewModel Frameworks and Resources

**Some of the many ViewModel frameworks…**

- MVVM Light
  - By Laurent Bugnion
  - http://mvvmlight.codeplex.com
- SilverlightFX
  - http://projects.nikhilk.net/SilverlightFX
  - http://github.com/NikhilK/SilverlightFX

**Additional Resources**

- Slides and Code + series of ViewModel-related posts
  - http://www.nikhilk.net
- Silverlight Unit Testing Framework and Silverlight Toolkit
  - http://silverlight.codeplex.com
- Silverlight developer page
  - http://www.silverlight.net

# Take-aways

- ViewModel pattern is simple means to separating application logic from user interface
  - Natural fit for Silverlight programming model
  - Improves testability and designer/developer workflow
  - Facilitates sharing application logic across multi-screen/device applications

- Behaviors, actions and triggers provide a declarative mechanism for implementing UI logic

# Q&A

# Backup

# ViewModel Blog Posts

- The Case for ViewModel
  http://www.nikhilk.net/Why-ViewModel.aspx

- View/ViewModel Association - Convention and Configuration-based Approaches
  http://www.nikhilk.net/View-ViewModel-Hookup-Convention-Configuration.aspx

- ViewModel Pattern for Silverlight - Options for Hooking a View to its Model
  http://www.nikhilk.net/ViewModel-View-Hookup-Options.aspx

- View/ViewModel Interaction - Bindings, Commands and Triggers
  http://www.nikhilk.net/View-ViewModel-Interaction.aspx

- Dialogs and ViewModel - Using Tasks as a Pattern
  http://www.nikhilk.net/ViewModel-Dialogs-Task-Pattern.aspx

- ViewModel with MVC/Navigation in Silverlight
  http://www.nikhilk.net/Silverlight-ViewModel-MVC.aspx

# What is Silverlight?

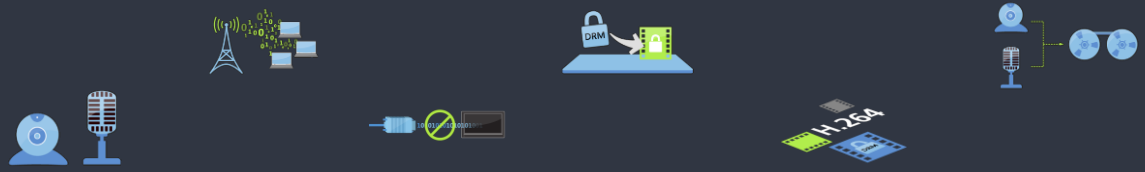
…a powerful development platform for creating engaging, interactive applications for many screens across the Web, desktop, and mobile devices

…a free plug-in powered by the .NET framework that is compatible across multiple browsers, devices and operating systems to bring a new level of interactivity wherever the Web works.

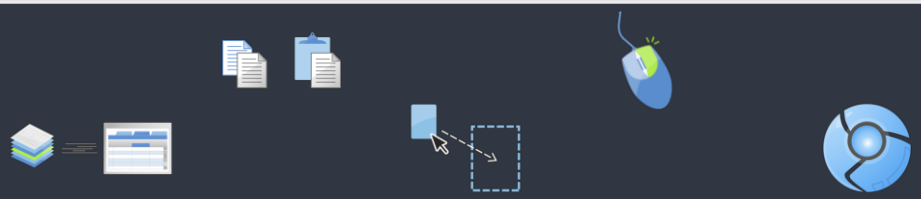
With support for advanced data integration, multithreading, HD video using IIS Smooth Streaming, and built in content protection, Silverlight enables online and offline applications for a broad range of business and consumer scenarios.

# Silverlight 4 Themes

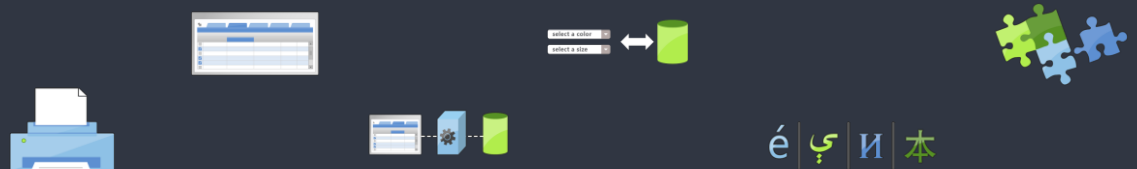
## Media




## Rich Experiences

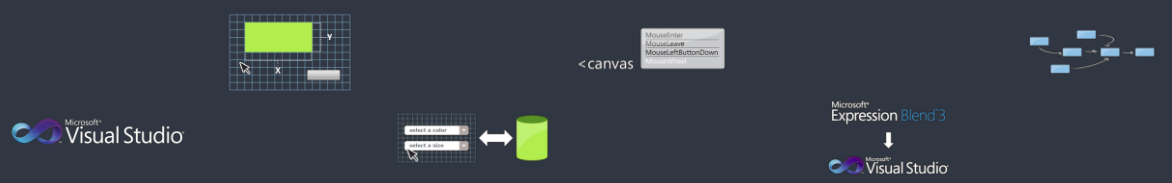


## Business Applications




## Beyond the Browser


## Developer Tools



Microsoft®

*Your potential. Our passion.™*