Using MVVM to build .NET apps with ArcGIS Runtime for .NET

Antti Kajanus

Thad Tilton

Rich Zwaap

Topics

- MVVM overview
- Benefits of using the pattern
- Implementation basics
- Code reuse and testing
- Questions and answers

What is MVVM?

- *Model-View-ViewModel*
- Widely used pattern for XAML / .NET languages
- Architecture that separates UI from business logic

- Basics are not difficult
  - Some related topics can be: dependency injection, event aggregation
  - May need to unlearn some familiar coding patterns

Why bother?

Sounds like a lot of work

Maintainable code

Separation of user interface and business logic that can be developed individually. Loose coupling and one way dependencies, changes to one layer doesn't require others to be changed or retested.

Testable code

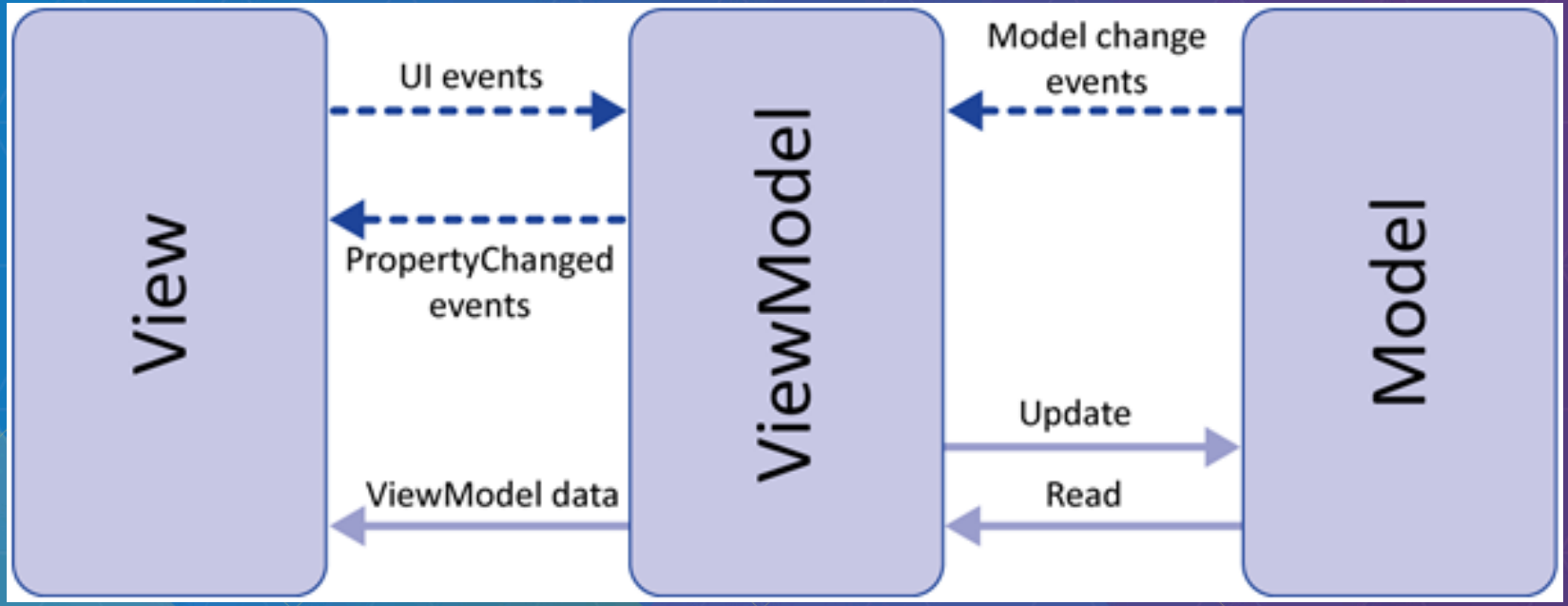
Separation of business logic and components enables automated unit testing and minimized user interface based testing.

Shareable code

Separation platform specific user interface code and shareable business logic enables easy way to share most of the code between platforms reducing amount of code needed to write.

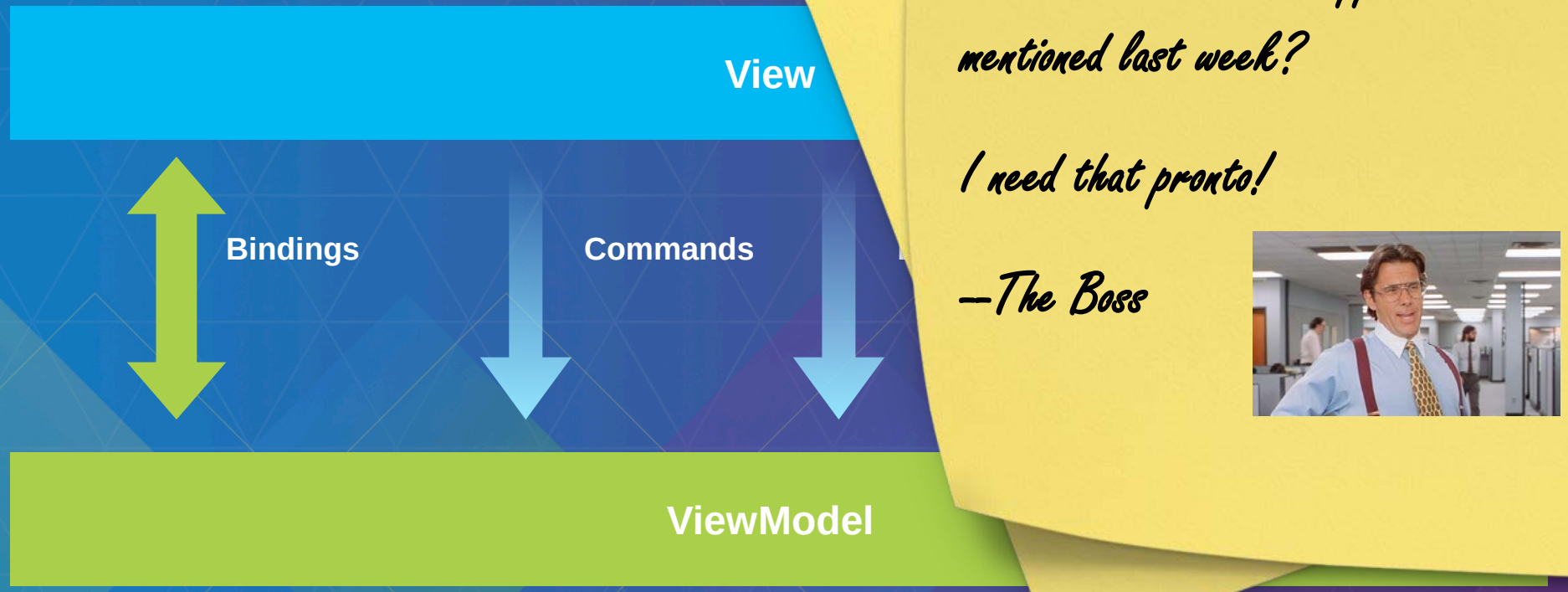
Components of the pattern

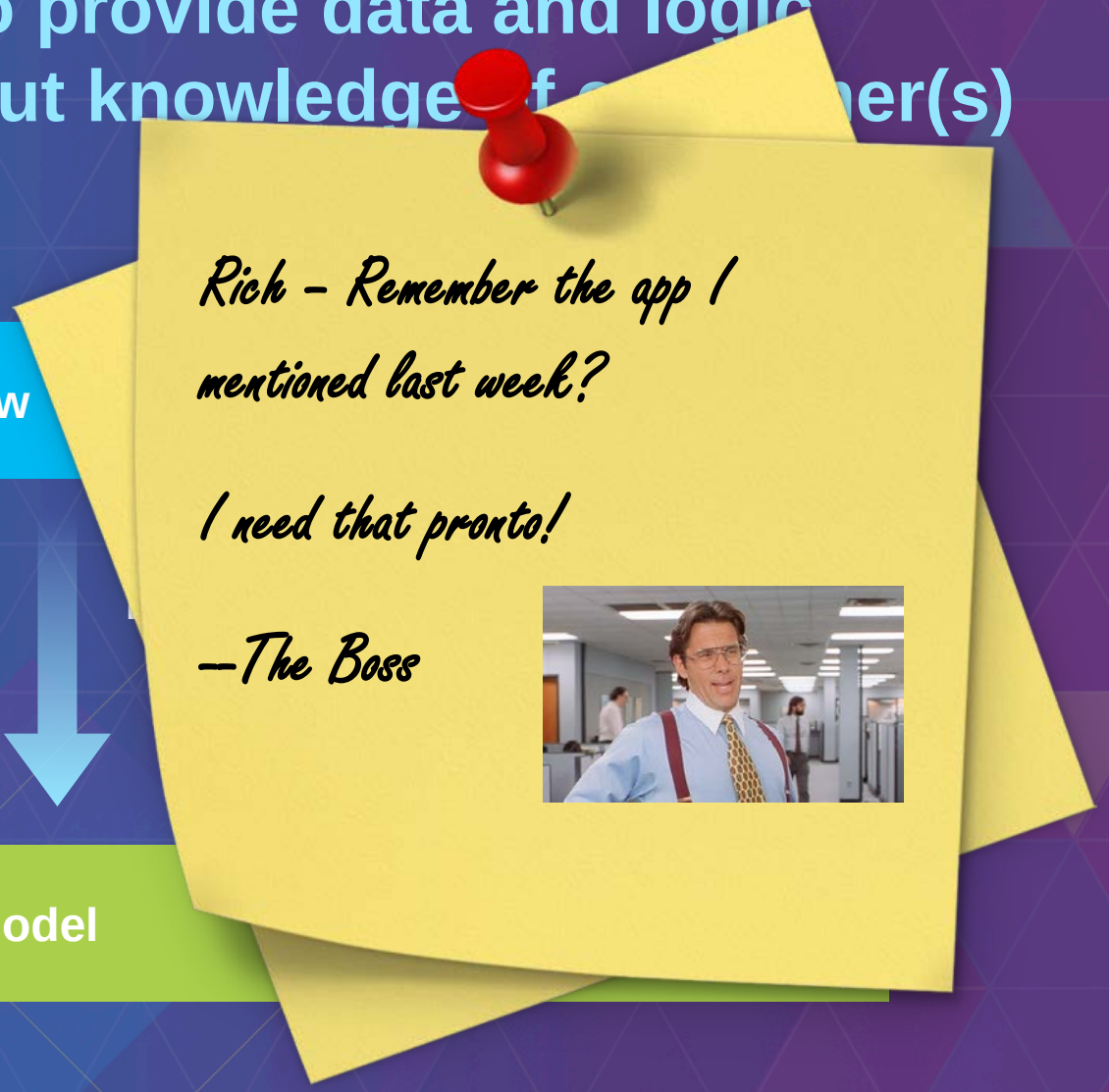
- Model – data objects
- View – user interface
- ViewModel – provides logic and data for the View



Communication between View and ViewModel

- View relies on the ViewModel to provide data and logic
- ViewModel provides data without knowledge of consumer(s)





Demo: Non-MVVM app

We'll re-architect this app

MVVM Implementation

- UI markup (XAML) stays basically the same
- Code behind logic is moved to ViewModel class(es)
- Create Models to abstract data (where appropriate)
- Goal is to separate business logic from UI logic
  - It's OK to have UI-related code in your View

```
using Windows.UI.Xaml.Controls;

namespace ArcGISApp2
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
    }
}
```

Move (most) code behind logic to ViewModel

- Expose public properties for consumption in the View
  - Data for UI controls
  - Commands for control logic
- UI-related code can stay
  - Event handlers, e.g.
- Implement INotifyPropertyChanged
  - Ensure the View is notified when bound property values change

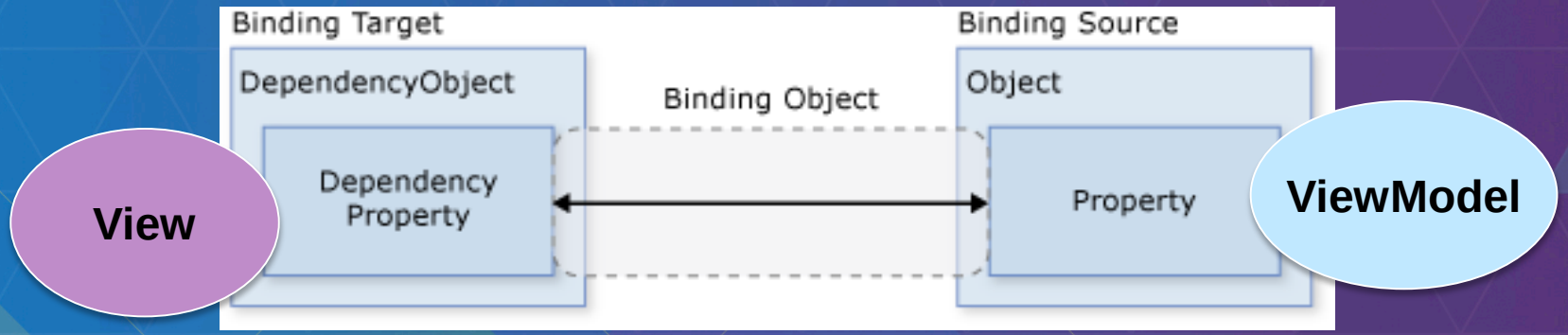
```
public class MapViewModel : System.ComponentModel.INotifyPropertyChanged
{
```

Demo: Create a ViewModel

- Implement INotifyPropertyChanged
- Create a Map property

Data binding is the glue

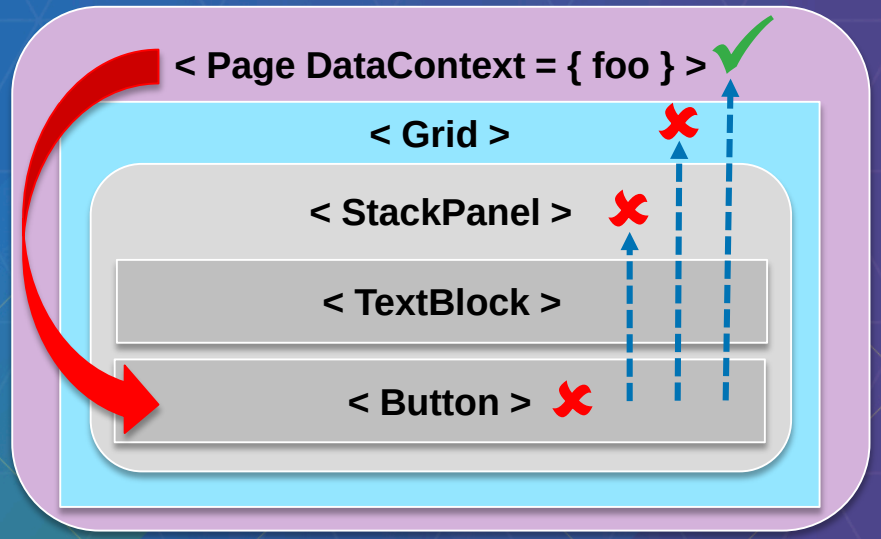
- Binds data from the ViewModel to View components
  - Control property to a ViewModel property
- Binding target must be DependencyObject
- Target property must be DependencyProperty
  - Most UIElement properties are dependency properties



Data binding overview (msdn) - https://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx

Data context

- Provides an object as a binding source
- Can be set on any FrameworkElement
  - DataContext property
  - Set with an object
- Applies to all child elements
  - DataContext is resolved by searching up the tree until one is found



Data binding syntax

- Data binding can be defined using Binding Markup Extension
  - Data context: the object (ViewModel, e.g.) providing data

```
<Grid DataContext="{StaticResource CustomerViewModel}">
```

```
this.DataContext = new CustomerViewModel();
```

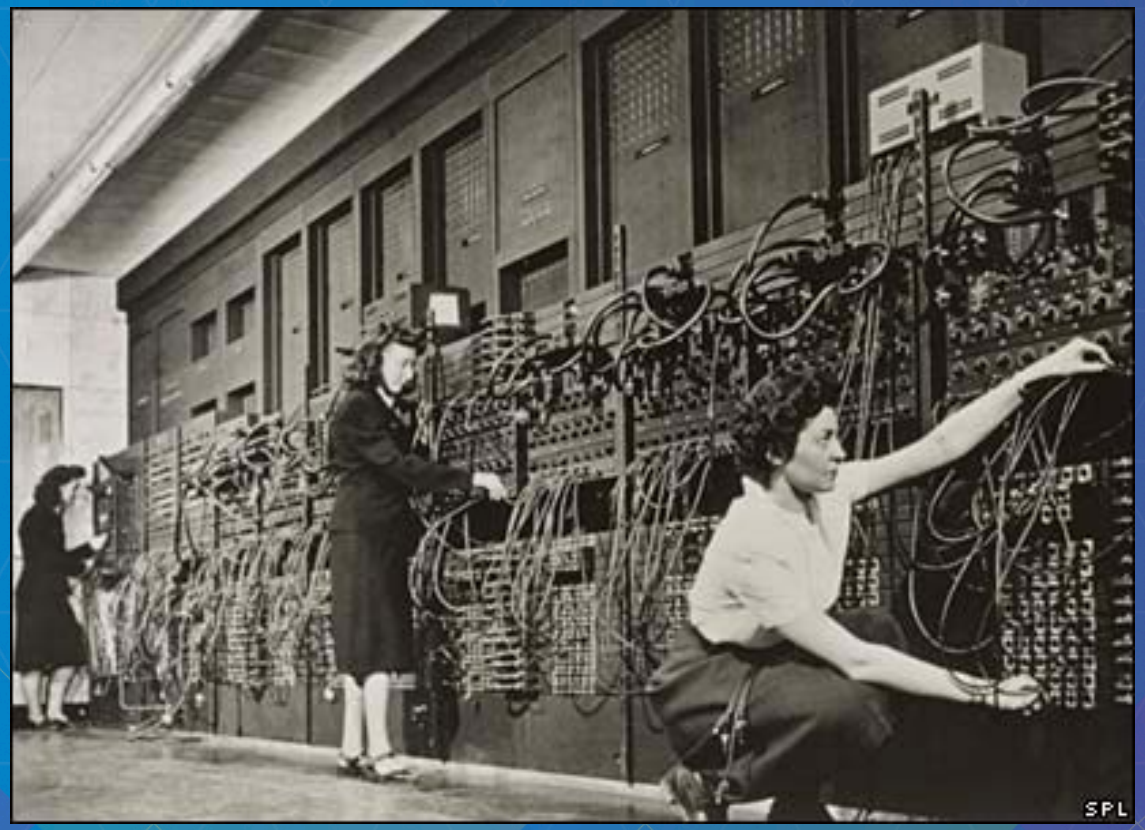
  - Path: the name of the specific data (property, e.g.)

```
<TextBlock Text="{Binding Path = FirstName}"/>
```

```
<TextBlock Text="{Binding FirstName}"/>
```

```
<TextBlock Text="{Binding Source = {StaticResource CustomerViewModel},
Path = FirstName}"/>
```

Data binding overview (msdn) - https://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx



Demo: Wire properties to the View

- Initialize instance of ViewModel
- Set Page DataContext
- Bind control properties to the ViewModel

Commanding

- ViewModel can provide *Commands* for use in the View
- Implementations of *ICommand*
  - Execute – handles Click
  - CanExecute – controls IsEnabled
  - CanExecuteChanged – fires when CanExecute value changes
- Use *CommandParameter* to pass data from the View

```
<Button Command = "{Binding ShowInfoCommand}"
        CommandParameter = "{Binding ElementName = RecentItemsGridView,
                                     Path=SelectedItem}" />
```

Binding to a UIElement in the page

Demo: Create a Command

Add a command property to the VM

Summary

How we refactored the app for MVVM

- *View* is the UI
  - We removed refer e behind
- *ViewModel* prov View
  - We moved and behind to a ViewModel
  - Exposed data ne perties
  - Implemented INoti w can get updates
- Used data binding wModel

Rich – Did you see my email about the new UI for the app? I need that by the end of the day!

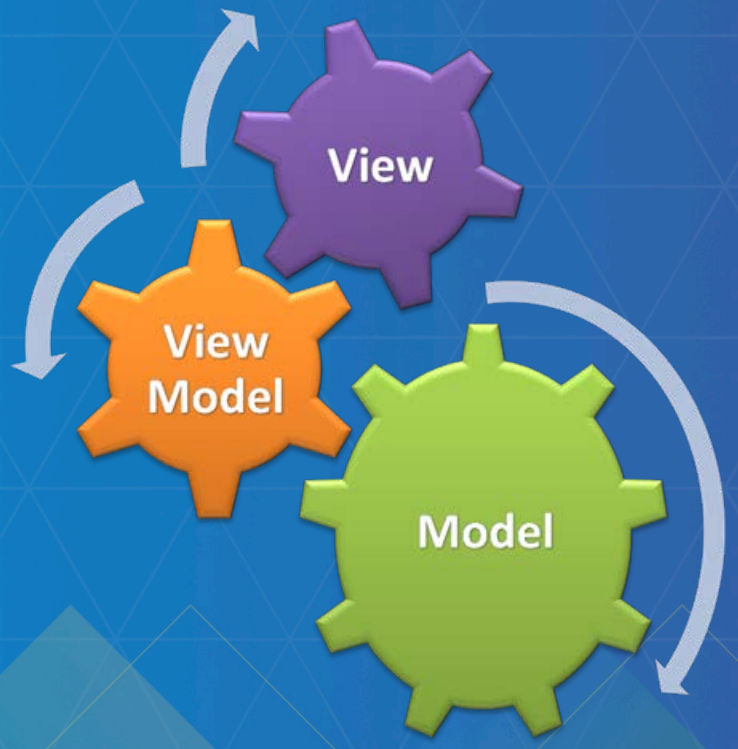
–The Boss

PS: a Win Phone app too.

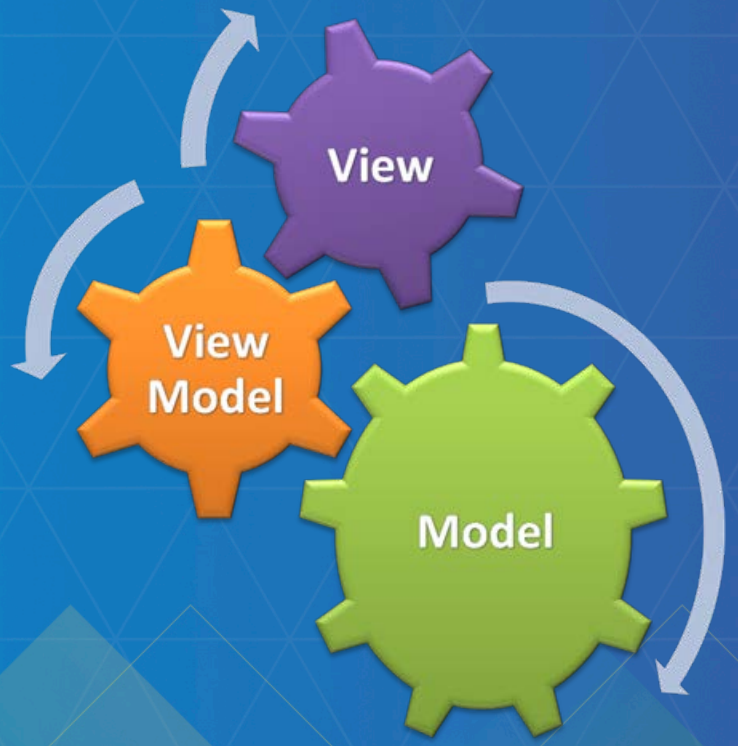
View

ViewModel

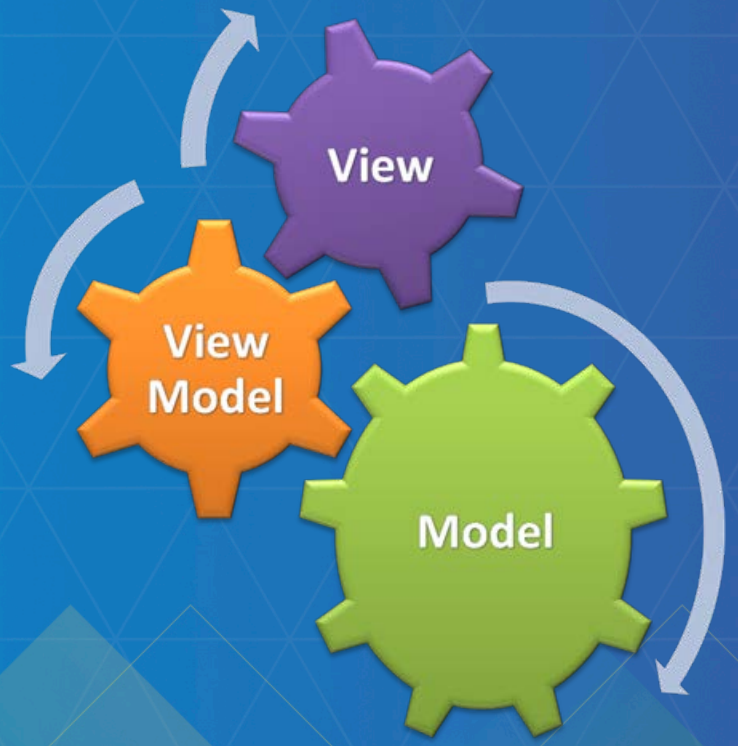
Model



Demo: Testing



Demo: Re-skinning the UI



Demo: Code reuse

Why bother?

Sounds like a lot of work

Maintainable code

Separation of user interface and business logic that can be developed individually. Loose coupling and one way dependencies, changes to one layer doesn't require others to be changed or retested.

Testable code

Separation of business logic and components enables automated unit testing and minimized user interface based testing.

Shareable code

Separation platform specific user interface code and shareable business logic enables easy way to share most of the code between platforms reducing amount of code needed to write.

Summary

- A clean MVVM architecture …
  - Helps separate UI from business logic
  - Facilitates backend testing by eliminating dependence on UI components
  - Makes it easier for designers and developers to work independently
  - Allows reuse of business logic for another UI (different platforms, e.g.)

- Other resources …
  - MVVM tutorial from developers guide
    - http://developersdev.arcgis.com/net/desktop/guide/use-the-mvvm-design-pattern.htm
    - Win Store Portal Viewer
      - https://github.com/Esri/arcgis-portalviewer-dotnet

ArcGIS Runtime SDK sessions Wednesday

Session Name	Time	Location
ArcGIS Runtime SDK for .NET: Tips and Tricks	10:30-11:30am	Primrose A
ArcGIS Runtime SDK for .NET: Transitioning to It from Other Esri .NET SDKs	1:00pm – 2:00pm	Primrose A
New CrossPlatformApp (ArcGIS Runtime, C#, Xamarin, MVVM)	2:30pm – 3:30pm	Mesquite C
Geo-enable Your .NET Apps with ArcGIS Online and Runtime	2:30-3:30pm	Primrose C/D

ArcGIS Runtime SDK sessions Thursday

Session Name	Time	Location
ArcGIS Runtime SDKs: Building Offline Apps, Part I	9:00 – 10:00am	Primrose A
ArcGIS Runtime SDK: Building Offline Apps, Part II	10:30– 11:30am	Primrose A
ArcGIS Runtime SDK for .NET: Integrating Devices, Sensors, Services, and More	1:00 – 2:00pm	Primrose B
Preview of ArcGIS Runtime and Xamarin	2:30-3:30pm	Mesquite GH
ArcGIS Runtime SDKs: Offline Routing and Geocoding	4:00-5:00pm	Smoketree A-E
ArcGIS Runtime SDK for .NET: How We Built the Plenary Apps	4:00-5:00pm	Primrose C/D
ArcGIS Runtime SDKs: Implementing 3D Capabilities	5:30 – 6:30pm	Primrose C/D

ArcGIS Runtime SDK sessions Friday

Session Name	Time	Location
The Road Ahead: ArcGIS Runtime	8:30 – 9:30am	Primrose A
Everything (or Anything) You Wanted to Know about the ArcGIS Runtime SDKs but Were Afraid to Ask	10:00 – 11:00am	Primrose A
The Road Ahead: Web 3D and Native Mobile Apps	10:00 – 11:00am	Primrose C/D
ArcGIS Runtime SDK for .NET: Tips and Tricks	1:00pm – 2:00pm	Smoketree A - E

Questions?

Rate This Session

www.esri.com/RateMyDevSummitSession