ArcGIS Runtime .NET: Extending Views and View Models

Mark Cederholm
UniSource Energy Services
Flagstaff, Arizona

Esri Developer Summit
March 7-10, 2017 | Palm Springs Convention Center

"How can I design a single, relatively simple GIS field application, that can be configured to look completely different, in order to accomplish different tasks?"

My First Attempt

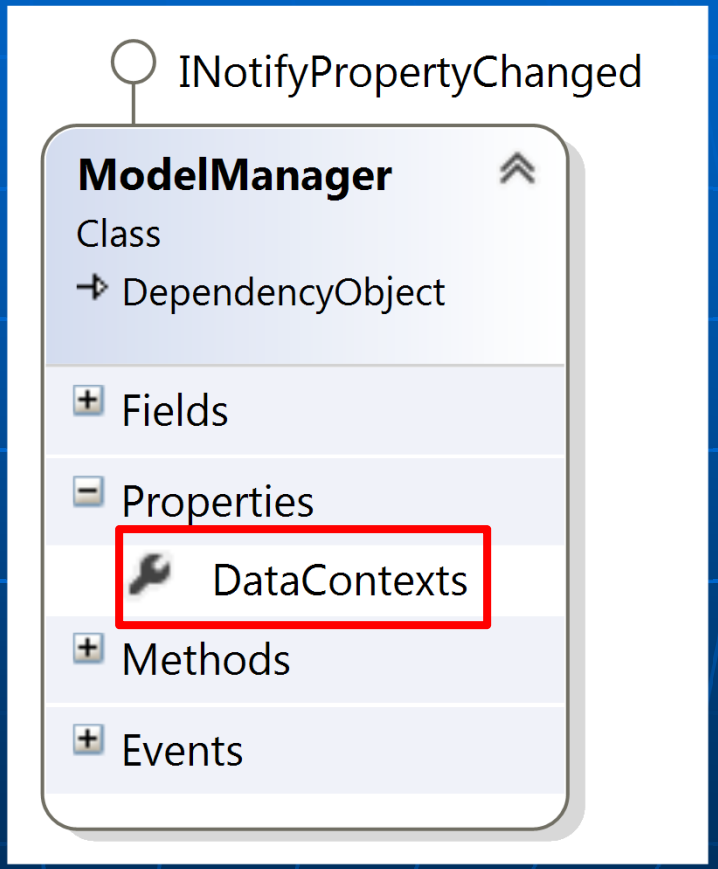
- Built using ArcGIS Runtime WPF, and later .NET
- Modeled after the architecture of the ArcGIS Viewer for Silverlight
- Layout design was challenging (not really WYSIWYG)
- Changing appearance of dialog windows required creating different extensions

A Mix-and-Match Problem

- How to extend a single view model to accommodate any view?
- How to avoid custom or Runtime elements in view XAML?
- How to interact with the MapView?
- How to wire it all together?

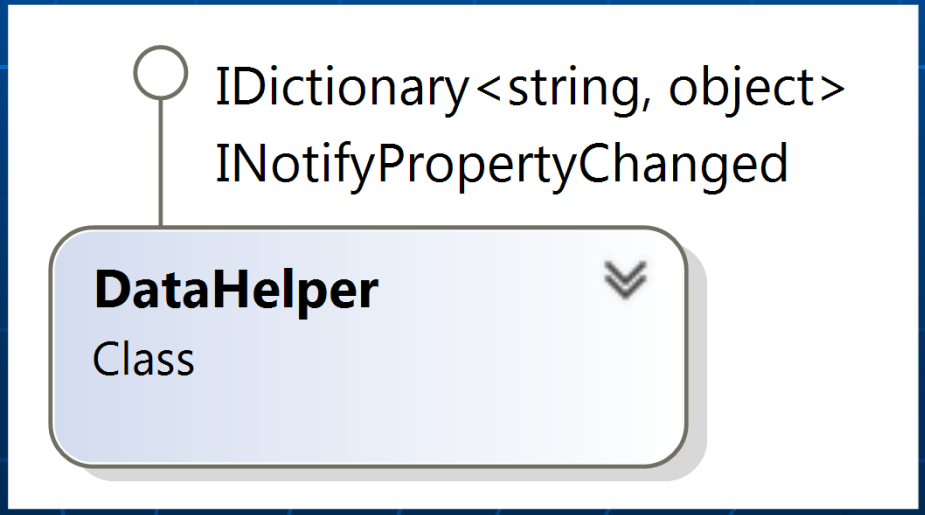
An Extensible View Model

All view elements in the main window bind to this property



The DataContexts Property

- An instance of the DataHelper class
- Returns null if a key is not present
- Notifies if a value is set
- Model extensions are added to this collection



A typical tool button:

```
<RadioButton
        DataContext="{Binding DataContexts[QueryAddin.QueryTools]}"
        Command="{Binding IdentifyCommand}"
        ToolTip="Identify"
        Style="{StaticResource LayoutToolStyle}"
        IsChecked="{Binding IdentifyToolChecked}">
        <Image
                Source="Images/Identify.png"
                Style="{StaticResource LayoutIconStyle}"/>
</RadioButton>
```

A MapView container:

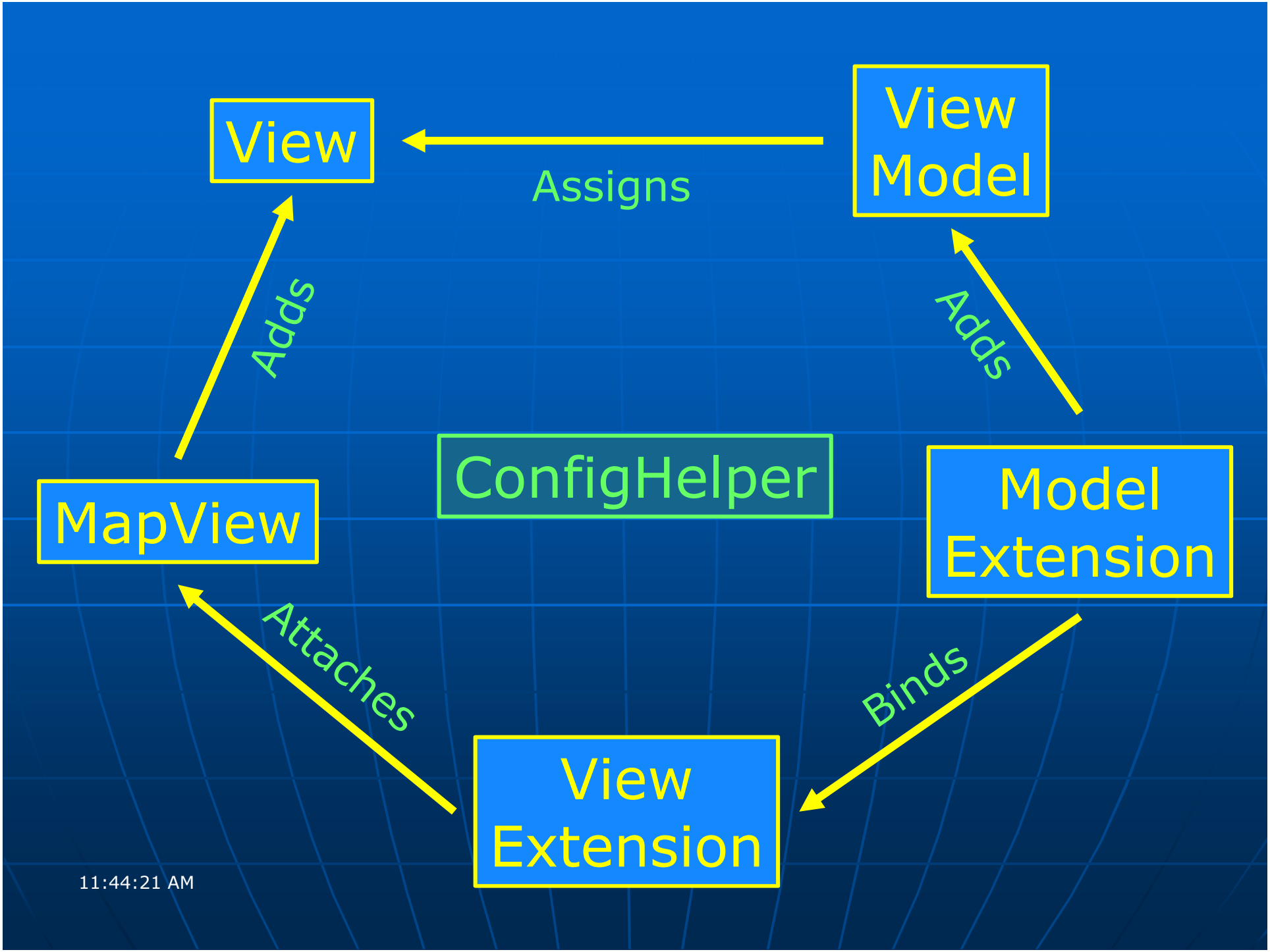
```
<ContentPresenter
        Grid.Column="1"
        Content="{Binding DataContexts[MapViewContainerContent]}" />
```

A Model Extension

- Typically acts as a data context
- Provides bindable properties that may or may not be consumed by a particular configuration
- May bind to views and/or view extensions
- May implement IConfigurable

A View Extension

- A subclass of Behavior<MapView>
- Handles interaction with the MapView
- May implement IConfigurable
- Typically has a model extension bound to it
- Bindings are declared in the configuration


View
View Model
Assigns
Adds
Adds
ConfigHelper
MapView
Model Extension
Attaches
Binds
View Extension
11:44:21 AM

Demo

Application Architecture

MyApp.exe

IMainWindow

MyConfig.xml

Configuration

Extensibility.dll

MapApplication
ChildWindow
Interfaces
Helper Classes

MyAddin.dll (1+)

Model Extensions
View Extensions

MyViews.dll (1+)

Main Layout
Dialogs
Resources.xaml

11:44:21 AM

Subfolders

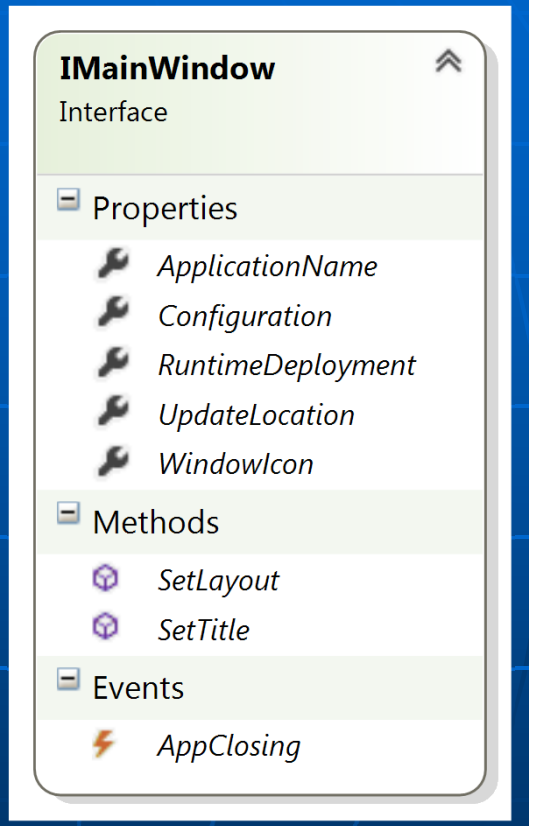
- Application (or Updates) Folder
  - Config
  - Extensions
  - Views

Starting MapApplication

- Implement IMainWindow
- Init MapApplication:

```
MapApplication.Init(pMainWindow);
```

- IMainWindow supplies the configuration keyword



Extensibility Interfaces

- IMainWindow: app main window
- IStatus: reporting status, messages
- IConfigurable: configured extensions
- IWindowContent: nonmodal dialogs
- IDialogContent: modal dialogs
- IMapTool: tools

Major Extensibility Classes

- MapApplication: entry point and basic management
- ResourceHelper: access to view resources
- SettingsHelper: manages settings
- CommandHelper: ICommand support
- ToolHelper: IMapTool support

Configuration: Basic Structure

```
<?xml version="1.0" encoding="utf-8" ?>
<Application>
        <MainLayout>
                . . .
        </MainLayout>
        <Extensions>
                . . .
        </Extensions>
</Application>
```

Configuration: MainLayout

```
<MainLayout>
        <Title>FieldApp - Gas</Title>
        <ViewAssembly>BasicViews</ViewAssembly>
        <MainLayoutName>BasicLayout</MainLayoutName>
</MainLayout>
```

Configuration: Extensions

```
<Extensions>
        <Extension Name="NavAddin">
                <ViewAssembly>BasicViews</ViewAssembly>
                <ModelExtensions>
                        . . .
                </ModelExtensions>
                <ViewExtensions>
                        . . .
                </ViewExtensions>
        </Extension>
        . . .
</Extensions>
```

Configuration: Model Extensions

```
<ModelExtensions>
        <ModelExtension Name="MapModelExt">
                <ConfigData>
                        <EnableTOC>True</EnableTOC>
                        <DataModelTOC
                                Assembly="WebAddin"
                                ClassName="TocWebDataItem" />
                </ConfigData>
        </ModelExtension>
        <ModelExtension Name="NavModelExt" />
        . . .
</ModelExtensions>
```

Configuration: View Extensions

```
<ViewExtensions>
        <ViewExtension Name="NavViewExt">
                <DataContext Name="NavAddin.NavModelExt" />
                <Bindings>
                        <Binding
                                TargetDependencyProperty="..."
                                SourceProperty="..."
                                BindingMode="..." />
                        .  .  .
                </Bindings>
        </ViewExtension>
        .  .  .
</ViewExtensions>
```

BindingMode can be: “OneWay”, “TwoWay”, “OneWayToSource”

Available Examples [10.2.7]

- NavAddin: map navigation, TOC
- QueryAddin: identify, find, measure
- LayoutAddin: printable layouts
- WebAddin: map services/packages
- EditAddin: sync framework support
- AppUpdateAddin: receive updates
- RTReader: example finished app

Future Directions and Possible Ideas

- Runtime 100 version
- NavAddin: MMPK support
- Utility network support
- Overview map extension
- Docking windows extension
- ???

Questions?

- Mark Cederholm
  mcederholm@uesaz.com
- This presentation and examples may be downloaded at:

http://www.pierssen.com/arcgis/runtime.htm