ArcGIS Pro SDK for .NET: Practical Dockpane Design and Implementation

Wolf Kaiser

2021 ESRI DEVELOPER SUMMIT

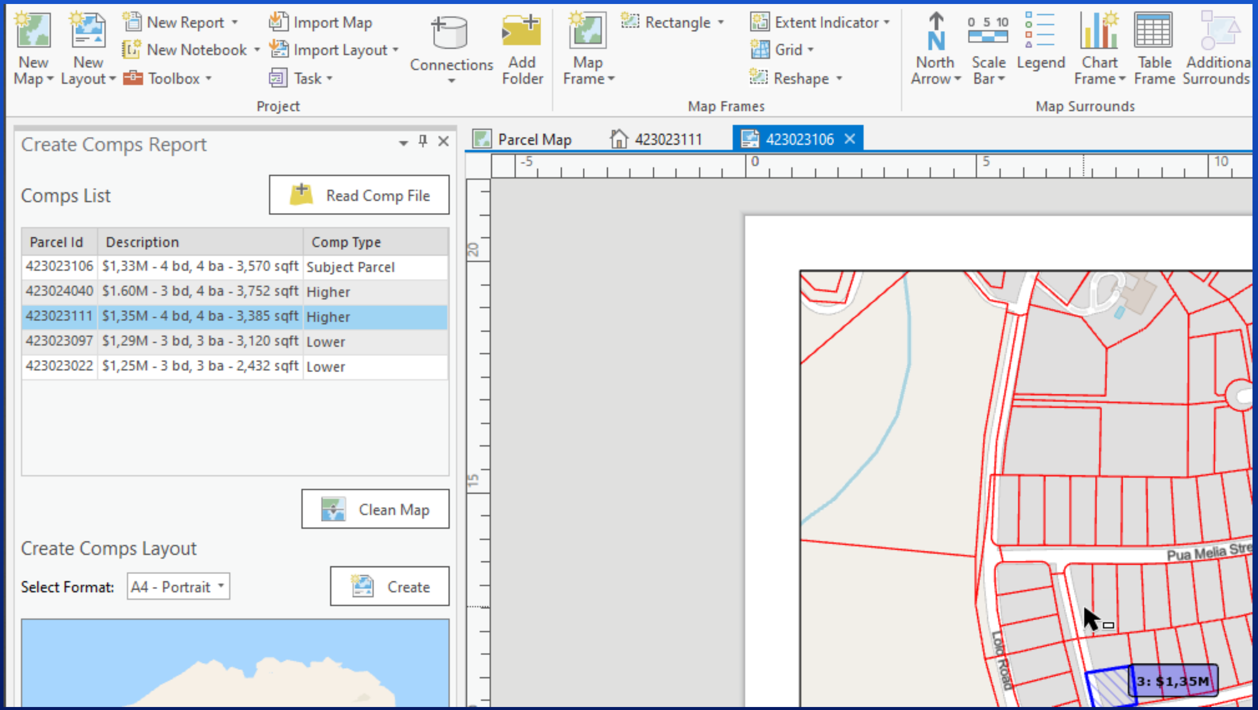
Session Overview

- UI Design Considerations for ArcGIS Pro Add-ins
  - Overview of Pro UI Framework Elements
  - Typical Use Case for Dockpane and Pane

- ArcGIS Pro Dockpane and Pane Deep Dive
  - Model View ViewModel Programming Pattern in Add-ins
  - MVVM key concepts:
    - Databinding
    - Buttons and Tools: ICommand Pattern
    - Combo Boxes, List Boxes, Data Grids
  - Re-Use Pro Tools and Commands
  - Multi threading considerations
  - Using ArcGIS Pro built-in UI controls in your Dockpane and Pane

ArcGIS Pro’s Enhanced GUI

- Ribbon / Tabs
- Buttons, Tools, Checkboxes, … Galleries, Button and Tool Palettes, Split Buttons
- Multiple detachable/dockable views
- Flexible docking framework
- Contextual UI
  - Functional areas are loaded on demand
  - UI elements are shown and enabled when relevant
- ArcGIS Pro UI defined using DAML (Desktop Application Markup Language)
  - https://github.com/Esri/arcgis-pro-sdk/wiki/ProConcepts-Framework#introduction-to-daml-desktop-application-markup-language



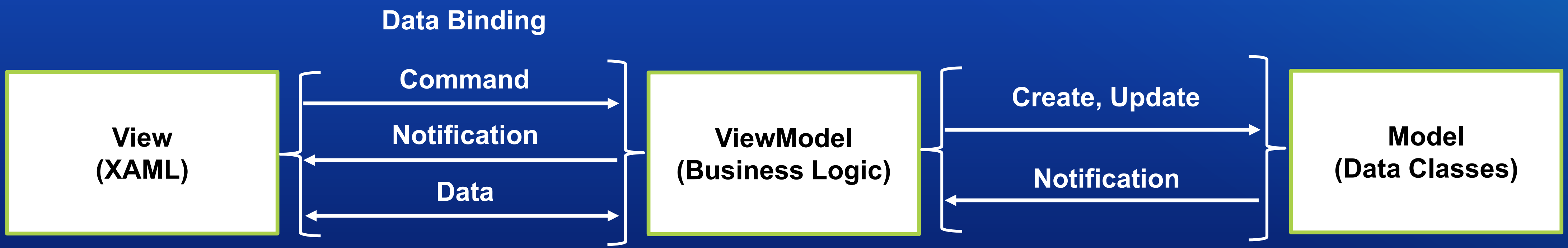
Parcel Id	Description	Comp Type
423023106	$1,33M - 4 bd, 4 ba - 3,570 sqft	Subject Parcel
423024040	$1.60M - 3 bd, 4 ba - 3,752 sqft	Higher
423023111	$1,35M - 4 bd, 4 ba - 3,385 sqft	Higher
423023097	$1,29M - 3 bd, 3 ba - 3,120 sqft	Lower
423023022	$1,25M - 3 bd, 3 ba - 2,432 sqft	Lower

Select Format: A4 - Portrait

Typical Use Case for Dockpane and Pane

ArcGIS Pro is built on a WPF based GUI

- ArcGIS Pro's GUI is built on WPF and the MVVM Programming Pattern
  - WPF - Windows Presentation Foundation is Microsoft's UI framework for desktop Apps
  - Model: Classes that represent the data consumed by the app (optional for Pro SDK)
  - View: User interface (UI) implemented as a WPF UserControl using XAML
  - ViewModel: Classes that wrap data (Model) and provide business logic for the UI (views)



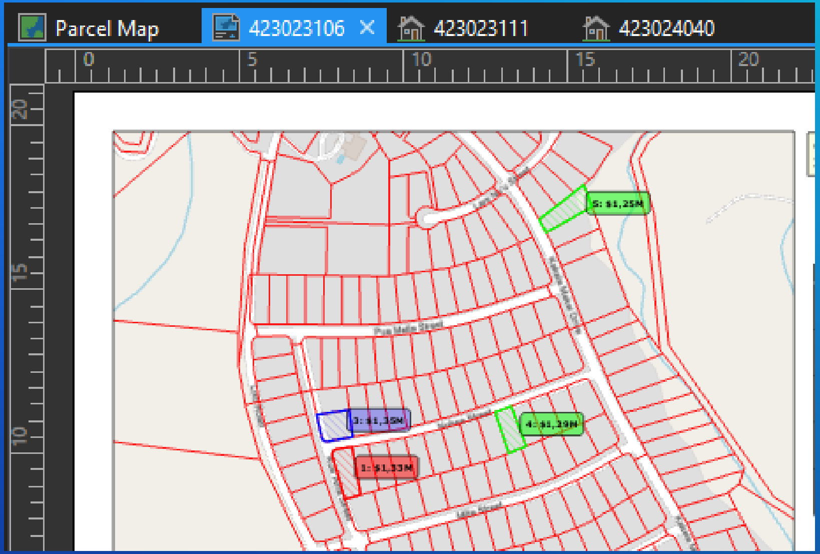
- Why use the MVVM Pattern?
  - Design pattern used to separate implementation aspects
  - Flexible, Testable
  - Well suited for threaded applications
  - Central to the ArcGIS Pro SDK

MVVM Implementation in Add-ins

- MVVM implementation in Add-ins follows the same Pattern used in WPF / .Net
  - The MVVM design pattern – MSDN: https://msdn.microsoft.com/en-us/library/hh848246.aspx
- In the ArcGIS Pro UI Framework: ViewModels for Simple UI Elements are private implementation details and not accessible in the public API
  - For example: Buttons, Tools, Checkboxes, Combo Boxes, Edit Boxes, Spinners, Menus, Context Menus, Dynamic Menus, …
- ViewModels for More Complex UI Elements are exposed in the public API
  - When you create one of these classes using a Pro SDK Item Template, the Pane or DockPane derived class (ViewModel) is automatically bound to the XAML View (usually a user control)
  - For example: Dockpane, Pane, Custom Control, Embeddable Control, Property Page
- Implement your Add-in UI just as you implement a UserControl in WPF/.Net
  - You can use many available online WPF MVVM snippets

Pane

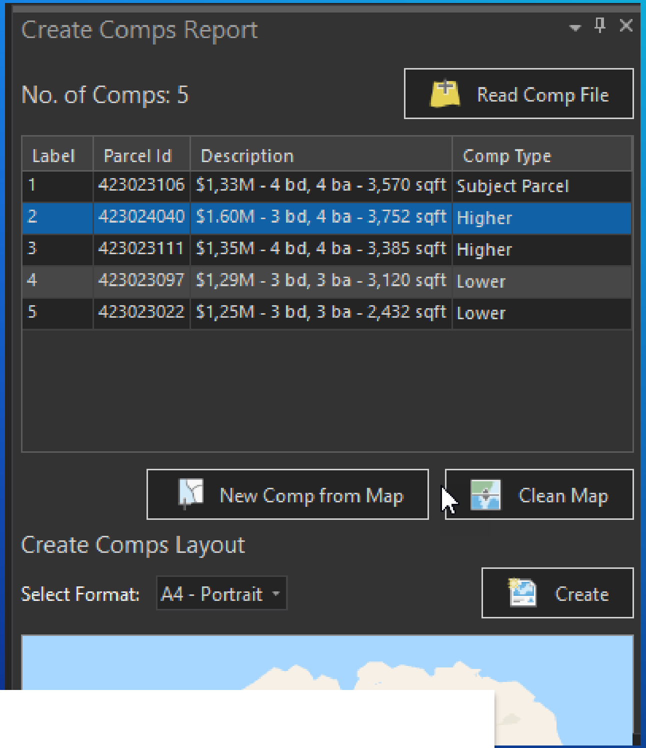
- Many instances
- Has Default Tab and Default Tool association – “context”
- In Pro, Panes are exposed by static class Properties:
  - Can be discovered by FrameworkApplication.Panes.ActivePane
  - Or FrameworkApplication.Panes.FindPane (paneID)
- Uses MVVM pattern
  - View Model derives from the Pane Contract
  - View is Custom User Control associated with Pane declaratively



```
<panes>
  <pane id="custom_Pane" caption=“Custom View" className=“CustomViewModel"
    isClosable="true" defaultTab=“custom_PaneTab" defaultTool=“custom_PaneTool">
    <content className=“CustomPaneView"/>
    <optionsMenu refID=“customPaneMenu"/>
  </pane>
</panes>
```

Dockpane

- Singleton
- Has no context association with the ribbon and no active tool
- Once created only hidden, never destroyed
- Uses MVVM pattern
  - View Model derives from the DockPane Contract
  - View is Custom User Control associated with DockPane declaratively


Create Comps Report

No. of Comps: 5

Label	Parcel Id	Description	Comp Type
1	423023106	$1,33M - 4 bd, 4 ba - 3,570 sqft	Subject Parcel
2	423024040	$1.60M - 3 bd, 4 ba - 3,752 sqft	Higher
3	423023111	$1,35M - 4 bd, 4 ba - 3,385 sqft	Higher
4	423023097	$1,29M - 3 bd, 3 ba - 3,120 sqft	Lower
5	423023022	$1,25M - 3 bd, 3 ba - 2,432 sqft	Lower

Create Comps Layout

Select Format: A4 - Portrait

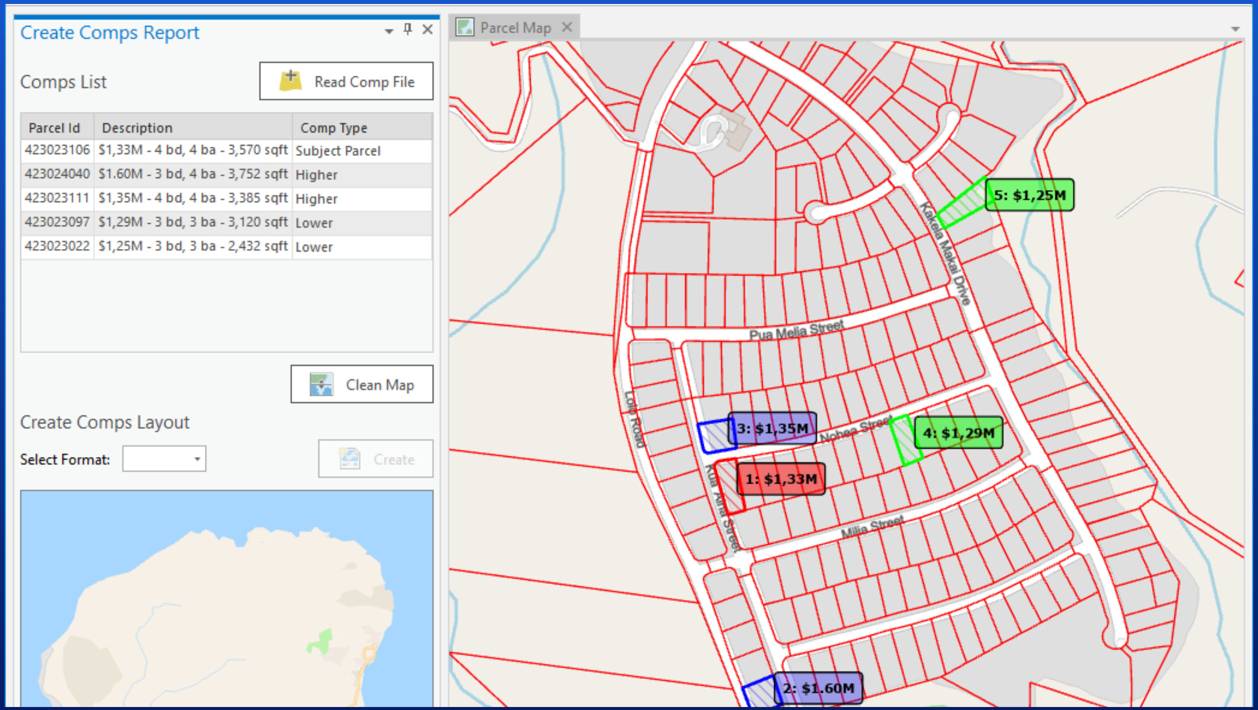
```
<dockPanes>
  <dockPane id="custom_TOCDockPane" caption="My Contents" className="DockPaneViewModel"
   dock="group“ condition="esri_core_MapPane"  dockWith="esri_core_ProjectDockPane">
    <content className="DockPaneView" />
  </dockPane>
</dockPanes>
```

Add-in UI Design Considerations for ArcGIS Pro

- Accessibility:
  - Search on YouTube for “ArcGIS UI Design for Accessibility and High DPI”
- Multi-threading
  - Search on YouTube for “ArcGIS Pro SDK for .NET: UI Design and MVVM”
- The ArcGIS Pro SDK provides some built-in user controls for use in Dockpanes
  - Map Exploration User Controls:
    - MapControl: ProConcepts Map Exploration · Esri/arcgis-pro-sdk Wiki (github.com)
  - Map Authoring User Controls:
    - Coordinate System Picker, Coordinate System Details, Symbol Searcher Control, Symbol Picker Control, Query Builder, Geocode
    - ProConcepts Map Authoring · Esri/arcgis-pro-sdk Wiki (github.com)
  - Framework User Controls:
    - Burger button, ChromiumWebBrowser, Circular animation, Message label, Search textbox, TabIndicator, Waiting cursor
    - ProConcepts Framework · Esri/arcgis-pro-sdk Wiki (github.com)

Demo: A Practical Dockpane

Demo Final Sample Solution


Create Comps Report

Comps List

Parcel Id	Description	Comp Type
423023106	$1,33M - 4 bd, 4 ba - 3,570 sqft	Subject Parcel
423024040	$1.60M - 3 bd, 4 ba - 3,752 sqft	Higher
423023111	$1,35M - 4 bd, 4 ba - 3,385 sqft	Higher
423023097	$1,29M - 3 bd, 3 ba - 3,120 sqft	Lower
423023022	$1,25M - 3 bd, 3 ba - 2,432 sqft	Lower

Create Comps Layout

Select Format:

MVVM Key Concepts

- The MVVM pattern separates the User Interface definition, layout and design (XAML) from the business logic (code)
- ‘Data Binding’: UI elements in a view are bound by name (XAML) to properties which are exposed by the ViewModel (business logic / code)
- Examples:

View – UI - XAML	ViewModel – Code behind – c#
`<TextBlock Text="{Binding Heading}“ />`	`public string Heading { get {…} set {…} }`
`<Button Command="{Binding RetrieveMapsCommand}"/>`	`public ICommand RetrieveMapsCommand => _retrieveMapsCommand;`

`<ComboBox ItemsSource="{Binding AllMaps}"`
`SelectedItem="{Binding SelectedMap,Mode=TwoWay}“/>`
	`public ReadOnlyObservableCollection<Map> AllMaps => _Maps;`
	`public Map SelectedMap { get {…} set {…} }`

Databinding: Text Properties

- Data Binding provides a ‘decoupled’ method for View and ViewModel to interact
- Data is bound at runtime using reflection and View (UI) calls ‘Getter’ (VM) to display data
- ViewModel Implements the ‘INotifyPropertyChanged’ interface to ‘notify’ View of value changes
- ViewModel uses ‘SetProperty’ and ‘NotifyPropertyChanged’
- Base classes provide notification convenience methods: SetProperty and NotifyPropertyChanged

```
<TextBlock Text="{Binding Heading}“ />
```

```
private string _heading = “Create Comps Reports";
public string Heading
{
    get { return _heading; }
    set
    {
        SetProperty(ref _heading, value, () => Heading);
    }
}
```

```
private string _heading = “Create Comps Reports";
public string Heading
{
    get { return _heading; }
    set
    {
        _heading = value;
        NotifyPropertyChanged(() => Heading);
    }
}
```

Databinding: Commands (Buttons)

- To bind a command of a button (in the UI) you need to bind a property (in your code-behind) that Implements ICommand
- ICommand is a .Net interface that represents the 'code contract' for commands
  - ICommand requires the implementation of two methods: Execute and CanExecute
- The Pro SDK API provides a convenience class for quick ICommand implementation
- RelayCommand implements ICommand
  - Specify Execute action
  - And [optional] CanExecute action
- The View calls ICommand.Execute to run the command's action
- Execute action can be 'async' for background execution

```
<Button Command="{Binding Path=CmdCleanMap}“ />
```

```
public ICommand CmdCleanMap
{
  get {
    return new RelayCommand(async () =>
      {
          var mapPane = await ActivateParcelMapAsync();
      });
  }
}
```

Databinding: Lists and DataGrids

- ComboBoxes, ListBoxes, and DataGrids provide ItemsSource to 'data bind' a list of items to display
  - Best option use: ObservableCollection<T>
- Use the SelectedItem attribute to 'data bind' to selected item in your code
  - Selected item must be of the class <T>
- ObservableCollection implement INotifyCollectionChanged which notifies any 'subscribers' when rows are:
  - Added
  - Deleted
  - Changed
- INotifyCollectionChanged enables the View to refresh the UI when data changes

```
<DataGrid Name="CompsTableView"
     …
     SelectedItem="{Binding Path=SelectedComp}"
     ItemsSource="{Binding Path=Comps}">
```

```
private ObservableCollection<CompData> _comps;
public ObservableCollection<CompData> Comps
{
 get { return _comps; }
 set
 {
  SetProperty(ref _comps, value, () => Comps);
 }
}
private CompData _selectedComp;
public CompData SelectedComp
{
 get { return _selectedComp; }
 set
 {
  SetProperty(ref _selectedComp, value,
             () => SelectedComp);
 }
}
```

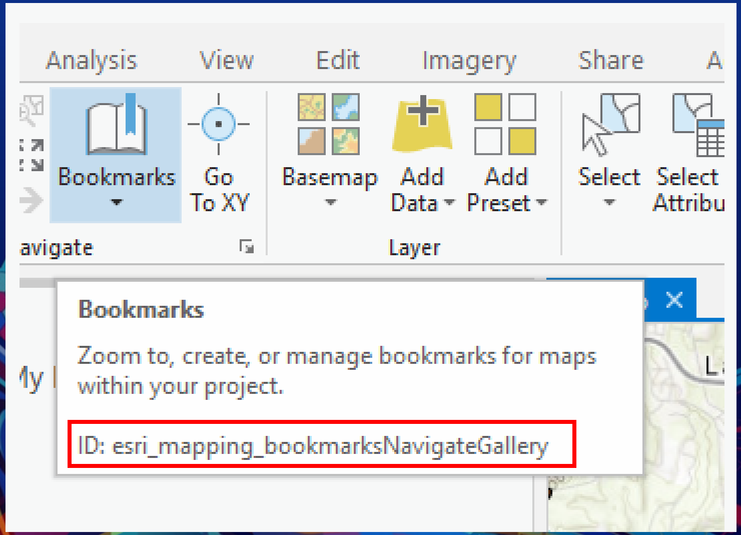
Using Existing ArcGIS Pro Commands in Code

- Use the ArcGIS Pro Framework’s “GetPlugInWrapper” method with any ArcGIS Pro element’s Id to get the IPlugInWrapper interface
- All buttons implement the ICommand interface - with Execute() and CanExecute()
- Programming pattern to use any ArcGIS Pro button functionality in your code:

```
// ArcGIS Pro's Create Bookmark button control DAML ID.
var commandId = ”esri_mapping_createBookmark”;
// get the ICommand interface from the ArcGIS Pro Button using GetPlugInWrapper
var iCommand = FrameworkApplication.GetPlugInWrapper(commandId) as ICommand;
if (iCommand != null)
{   // Let ArcGIS Pro do the work for us
    if (iCommand.CanExecute(null))
        iCommand.Execute(null);
}
```

How to find Exiting ArcGIS Pro Element Ids

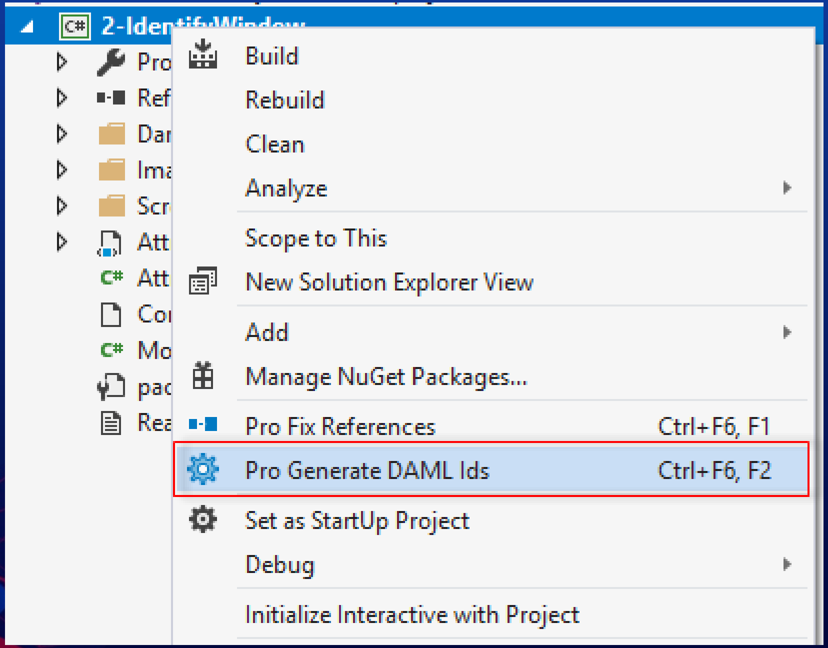
- SDK Help: ArcGIS Pro DAML ID Reference
  https://github.com/Esri/arcgis-pro-sdk/wiki/ArcGIS-Pro-DAML-ID-Reference
- Pro Generate DAML Ids tool in Visual Studio
  - Allows Intellisense to find Ids: i.e. DAML.Button.esri_mapping_addDataButton
- ArcGIS Pro Option: ‘Customize the Ribbon’ tab, check ‘Show Command IDs’
- Icons: https://github.com/Esri/arcgis-pro-sdk/wiki/DAML-ID-Reference-Icons

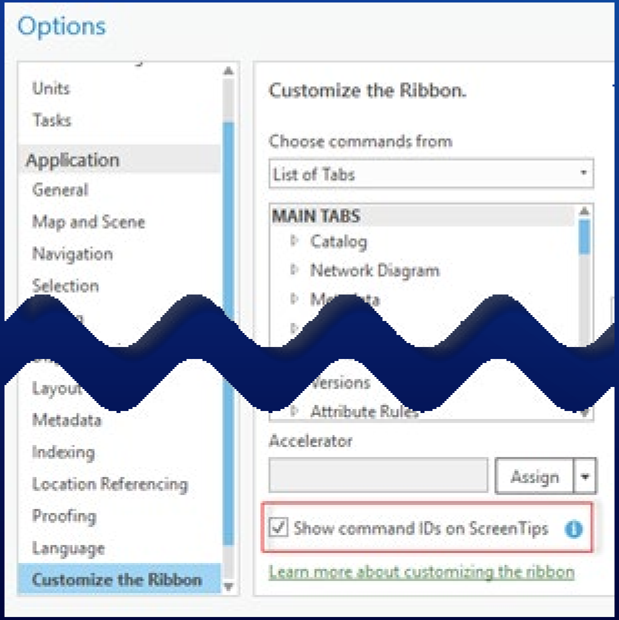


Bookmarks

Zoom to, create, or manage bookmarks for maps within your project.

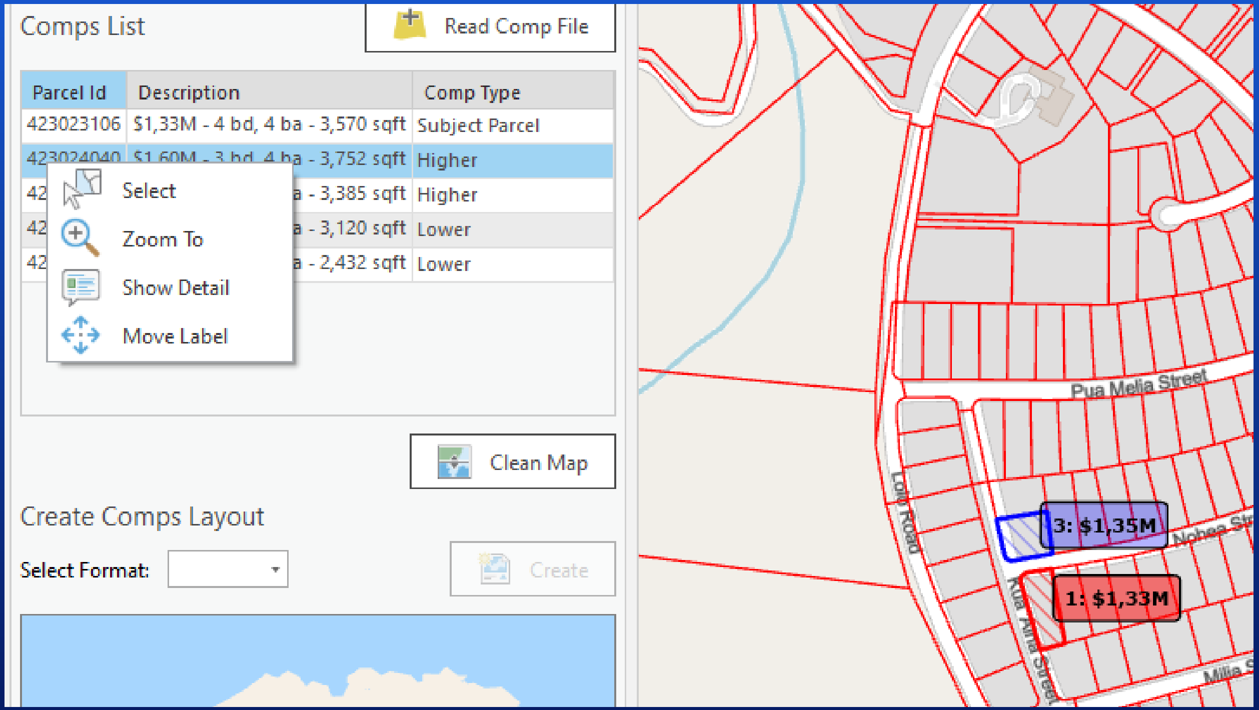
ID: esri_mapping_bookmarksNavigateGallery


2-IdentifyWindow
Build
Rebuild
Clean
Analyze
Scope to This
New Solution Explorer View
Add
Manage NuGet Packages...
Pro Fix References Ctrl+F6, F1
Pro Generate DAML Ids Ctrl+F6, F2
Set as StartUp Project
Debug
Initialize Interactive with Project



Customize the Ribbon.

☑ Show command IDs on ScreenTips


Comps List

Parcel Id	Description	Comp Type
423023106	$1,33M - 4 bd, 4 ba - 3,570 sqft	Subject Parcel
423024040	$1,60M - 3 bd, 4 ba - 3,752 sqft	Higher
42	a - 3,385 sqft	Higher
42	a - 3,120 sqft	Lower
42	a - 2,432 sqft	Lower

Create Comps Layout

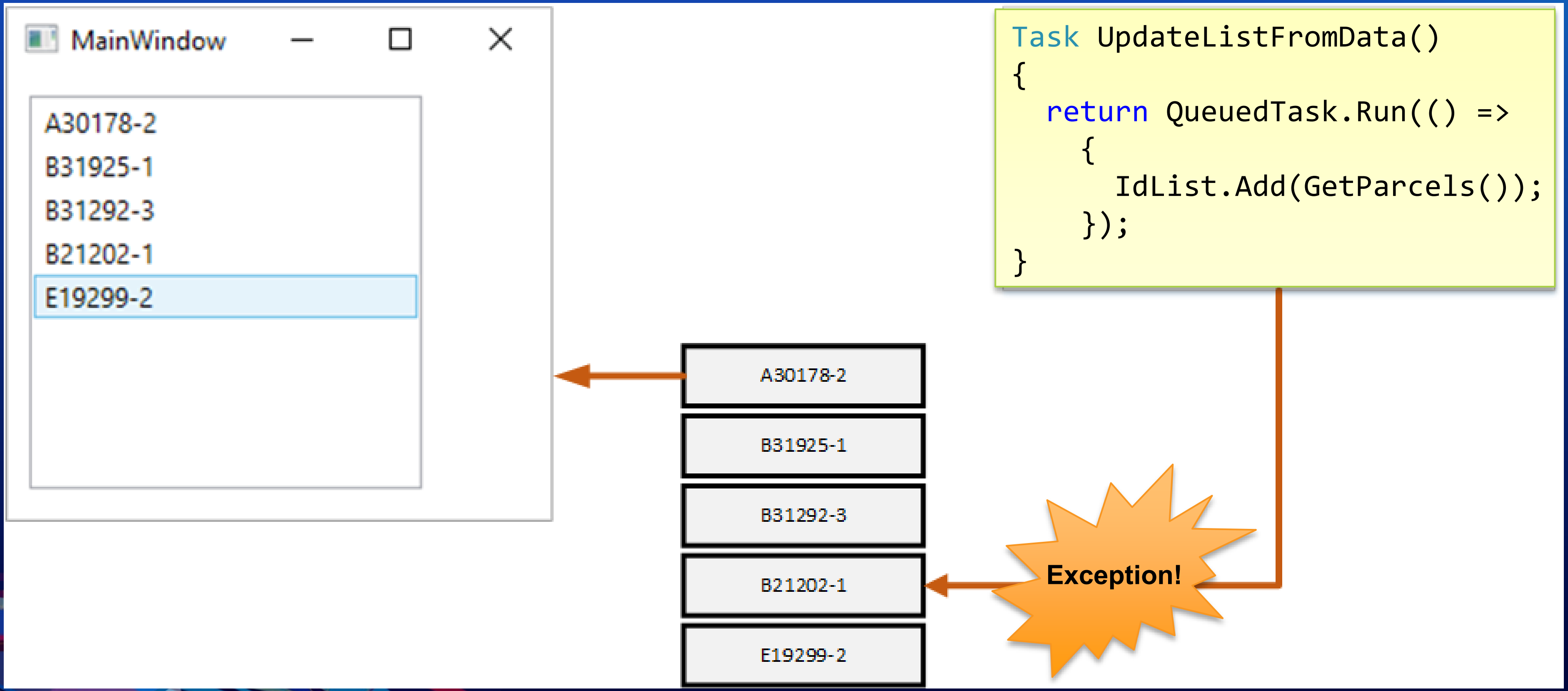
Select Format:

Demo: MVVM

Key Concepts

Threading - Concurrency Control (UI and MCT)

- Many data elements in Pro are NOT collected on the UI thread
- When collecting data from within QueuedTask.Run (worker thread) special precaution must be taken if the collected data is used by UI
- If precautions are not taken, the result is a 'Wrong Thread' exception

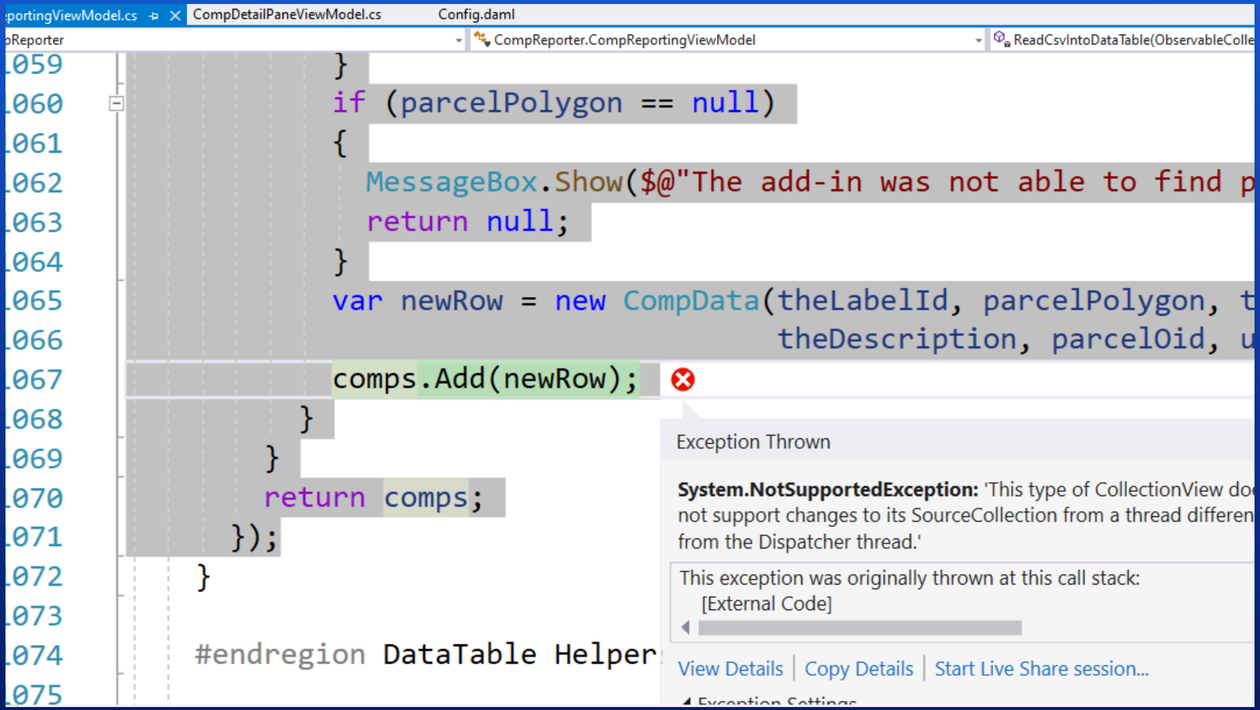


```
Task UpdateListFromData()
{
  return QueuedTask.Run(() =>
    {
      IdList.Add(GetParcels());
    });
}
```

Threading - Thread safe binding

- Rule: Mutable shared data must be protected by a lock, this includes WPF collections
- Both parties agree to entering the lock when reading or writing from/to collections
- Recommended pattern for updating collections from a worker thread can be found in .Net BindingOperations helper class:
  - BindingOperations.EnableCollectionSynchronization

```
internal class CompReportingViewModel : DockPane
{
  private readonly ObservableCollection<SetPage> _mpLyts = new ObservableCollection<SetPage>();
  private Object _lockMpLyts = new Object();
  public CompReportingViewModel()
  {
    BindingOperations.EnableCollectionSynchronization(_mpLyts, _lockMpLyts);
  }
  Task UpdateListFromData()
  {
    return QueuedTask.Run(() =>
    {
      lock (_lockMpLyts) { _mpLyts.Add(getData()); }
    });
  }
}
```



portingViewModel.cs
CompDetailPaneViewModel.cs
Config.daml
pReporter
CompReporter.CompReportingViewModel
ReadCsvIntoDataTable(ObservableColle

```
                    }
                    if (parcelPolygon == null)
                    {
                      MessageBox.Show($@"The add-in was not able to find p
                      return null;
                    }
                    var newRow = new CompData(theLabelId, parcelPolygon, t
                                              theDescription, parcelOid, u
                    comps.Add(newRow);
                  }
                }
                return comps;
              });
            }

            #endregion DataTable Helper
```

Exception Thrown

System.NotSupportedException: 'This type of CollectionView do not support changes to its SourceCollection from a thread differen from the Dispatcher thread.'

This exception was originally thrown at this call stack:
[External Code]

Demo: Update UI from worker thread

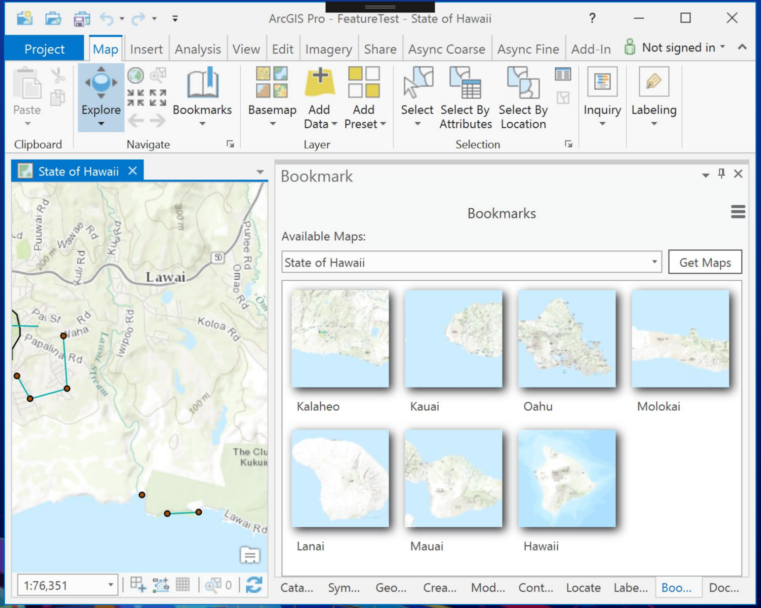
ArcGIS Pro SDK built-in user controls

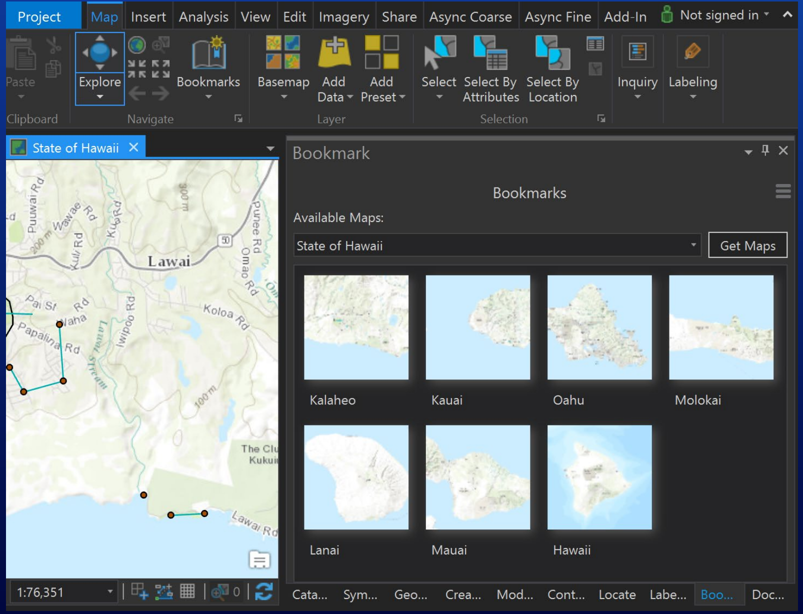
- Build-in User Controls:
  - Save time and provide a more ‘integrated’ look for your add-in

- Examples used in this session:
  - Map Exploration User Controls:
    - MapControl: ProConcepts Map Exploration · Esri/arcgis-pro-sdk Wiki (github.com)
  - Framework User Controls:
    - ChromiumWebBrowser
    - ProConcepts Framework · Esri/arcgis-pro-sdk Wiki (github.com)

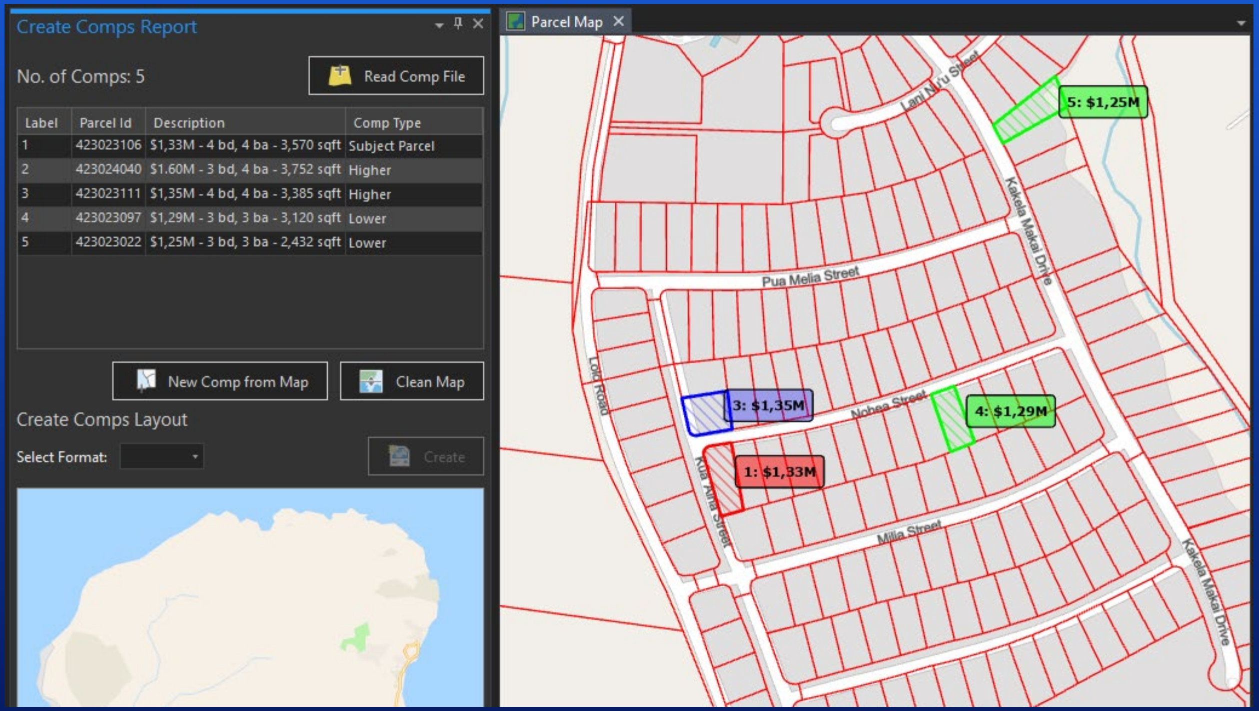
Add-in Styling to support Pro Themes and High Contrast

- In order for your Add-ins to “blend” when the theme is toggled they must be styled correctly
  - Note: It is not required that your Add-ins “blend” with Pro though it is desirable in most cases
- Dark/Light Theme: for graphics Pro switches between Images and DarkImages folders
- XAML Style Guide: https://github.com/Esri/arcgis-pro-sdk/wiki/proguide-style-guide





Demo: Built-in User Controls, Themes



Create Comps Report

No. of Comps: 5

Label	Parcel Id	Description	Comp Type
1	423023106	$1,33M - 4 bd, 4 ba - 3,570 sqft	Subject Parcel
2	423024040	$1.60M - 3 bd, 4 ba - 3,752 sqft	Higher
3	423023111	$1,35M - 4 bd, 4 ba - 3,385 sqft	Higher
4	423023097	$1,29M - 3 bd, 3 ba - 3,120 sqft	Lower
5	423023022	$1,25M - 3 bd, 3 ba - 2,432 sqft	Lower

Create Comps Layout

Select Format:

Practical Dockpane Design and Implementation

- Session content
  - Slides and Example code:
  - https://github.com/esri/arcgis-pro-sdk/wiki/tech-sessions