



# *ArcGIS Pro SDK for .NET: Mapping and Layout*

- Wolf Kaiser
- Uma Harano

# Mapping and Layout - Session Overview

- **Map Tool – Base class for interactions with Map Views**
- **Applying Map Tools for:**
  - Selecting features
  - Embeddable control
  - MapView Overlay Control
  - Custom Popups (Map Tool item template)
- **Creating Layouts using the ArcGIS Pro API**
  - Using the Layout class to create a Page Layout
  - Using Layout Elements to provide Page Layout content.



# MapTool

- **Base class representing a tool to perform interactive operations with a MapView.**
- **Used to create custom tools for**
  - **Selection, Identify, Editing**
- **Provides virtual methods for Keyboard and Mouse Events**
- **Provides properties to set default behavior of left-click to create a sketch.**
  - **Virtual SketchComplete and**
  - **SketchCancelled methods.**

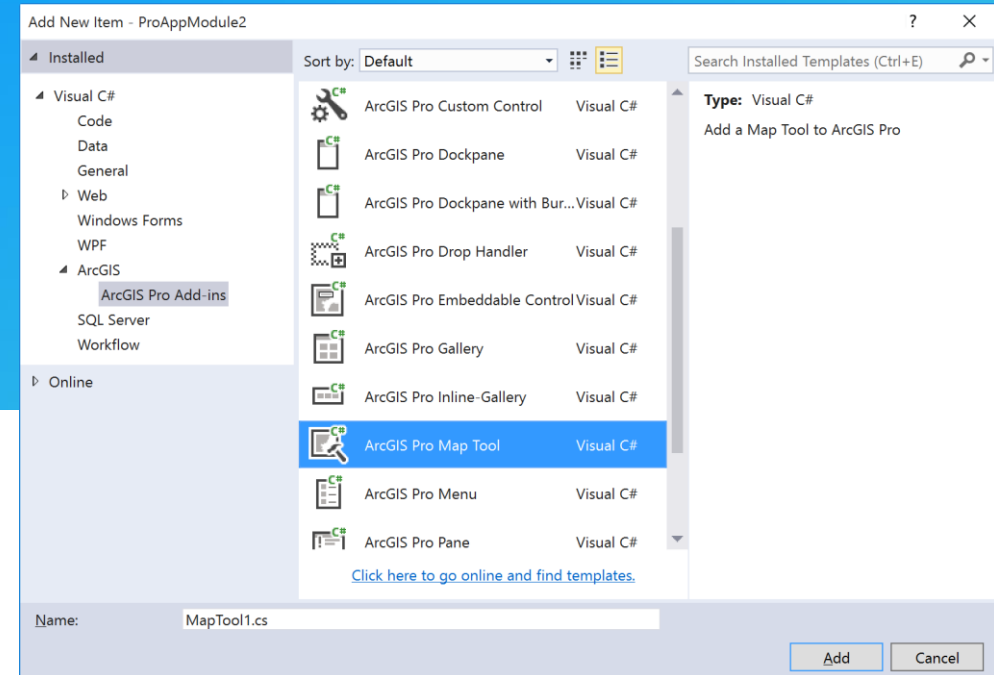


# MapTool

- Use Map Tool Pro SDK Item Template

```
class MyBasicSketchTool1 : MapTool
{
    public MyBasicSketchTool1()
    {
        IsSketchTool = true;
        SketchType = SketchGeometryType.Rectangle; //Sketch geometry
        SketchOutputMode = SketchOutputMode.Map; //Map or Screen.
    }

    protected override Task<bool> OnSketchCompleteAsync(Geometry geometry)
    {
        return base.OnSketchCompleteAsync(geometry);
    }
}
```

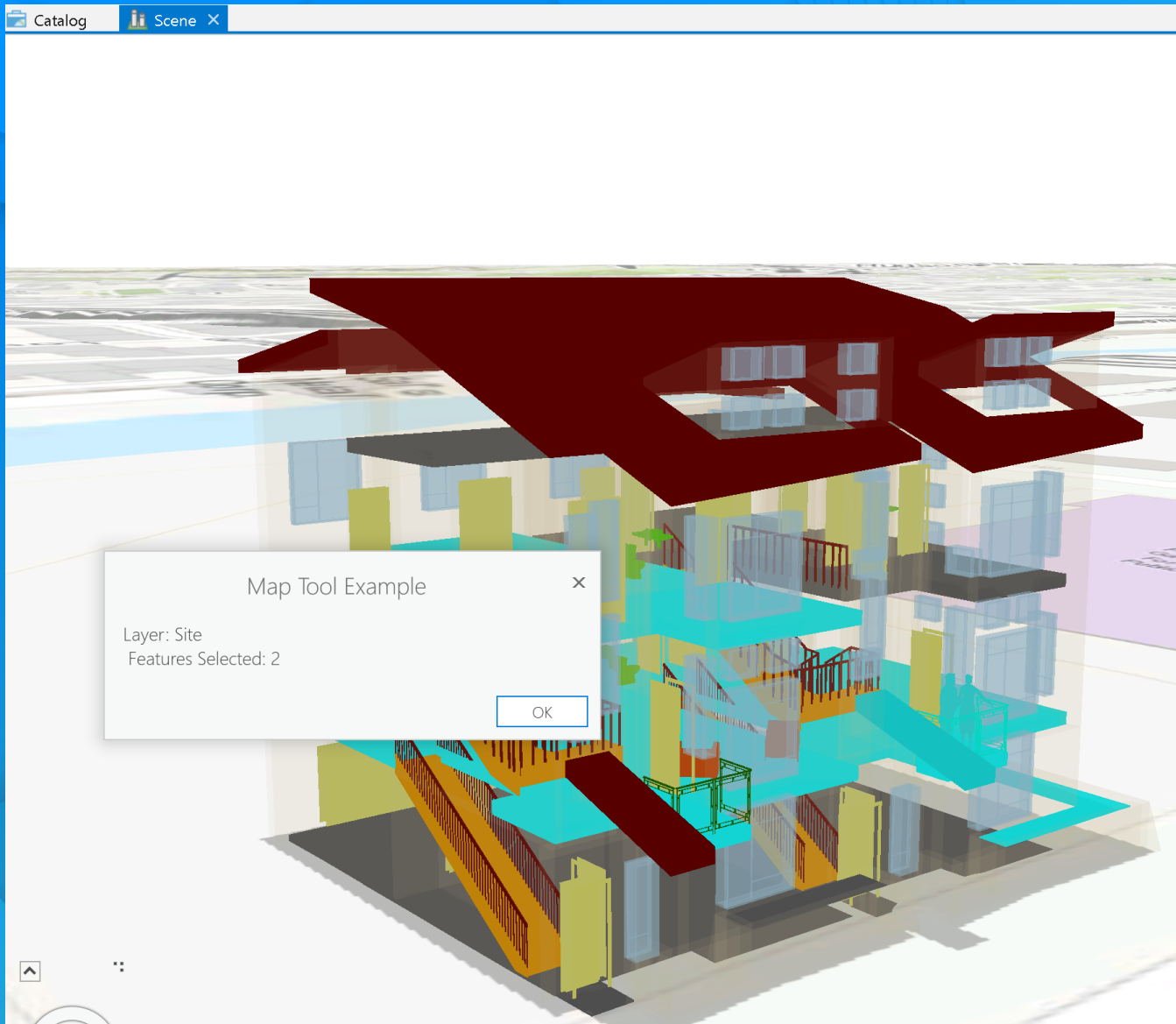


# Sketch Callbacks

- Callbacks after sketching.

- No need to Rubber-band “feedback” via Mouse Move, etc. as with ArcObjects
- Sketch geometry is parameter of the callback

```
class SketchTool1 : MapTool
{
    // On sketch completion select the intersecting features and zoom
    protected override Task<bool> OnSketchCompleteAsync(Geometry geometry) {
        return QueuedTask.Run(() => {
            //select features that intersect the sketch geometry
            var selection = MapView.Active.SelectFeatures(geometry);
            //zoom to selection (kvp.Key are the feature layers with selections)
            return MapView.Active.ZoomToAsync(selection.Select(kvp => kvp.Key),
                true, TimeSpan.FromSeconds(1.5), true);
        });
    }
}
```



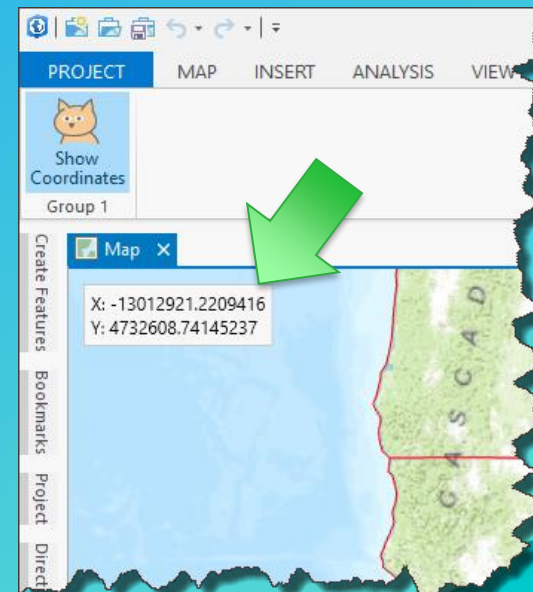
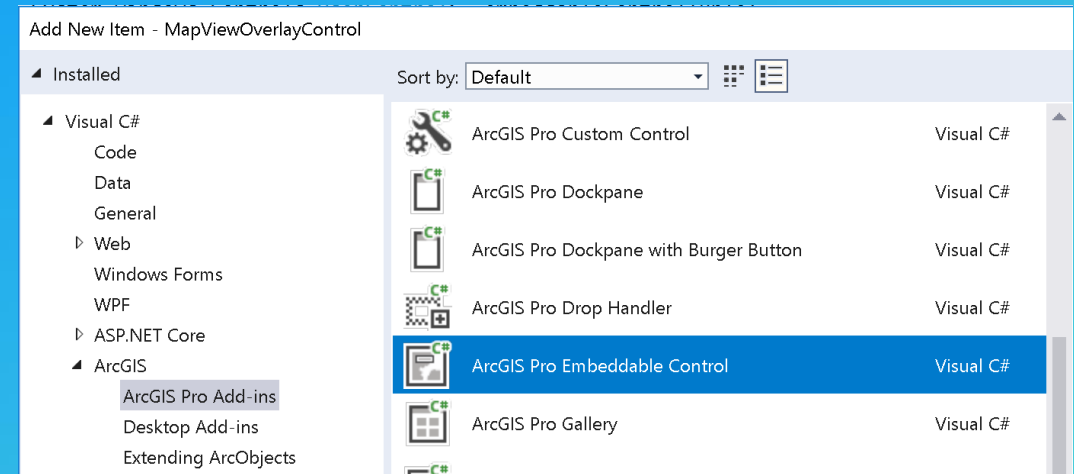
# Demo: Map Tool

- Town house dataset

# Map tool using an Embeddable Control

## Definition

- Map tool activation displays embeddable control on map view.
- Map tool deactivation removes embeddable control.
- Use Embeddable Control Pro SDK Item template.



# Map tool using an Embeddable Control

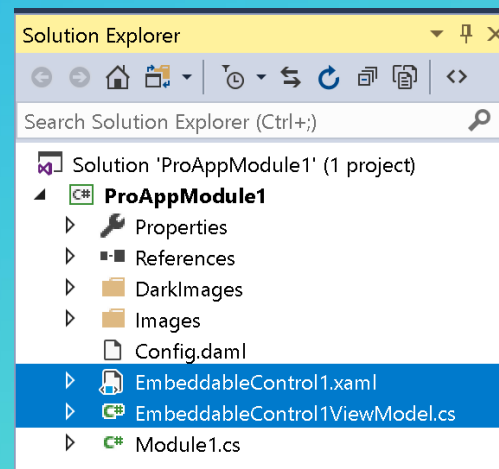
## Declaration

- Declare embeddable control in DAML within the categories element

```
<categories>
  <updateCategory refID="esri_embeddableControls">
    <insertComponent id="ProAppModule1_EmbeddableControl1" className="EmbeddableControl1ViewModel">
      <content className="EmbeddableControl1View" />
    </insertComponent>
  </updateCategory>
</categories>
```

- Embeddable control has View Model and View

- MVVM pattern
- stubbed out by the Pro SDK item template





# Map tool using an Embeddable Control

Linked to Map Tool

- Link Embeddable control to Map tool
  - Set OverlayControlID property to DAML Id of embeddable control.
- Optional embeddable control settings
  - OverlayControlPositionRatio: Ratio of 0 to 1 to place the control.
  - OverlayControlCanResize: Embeddable control can be resized.

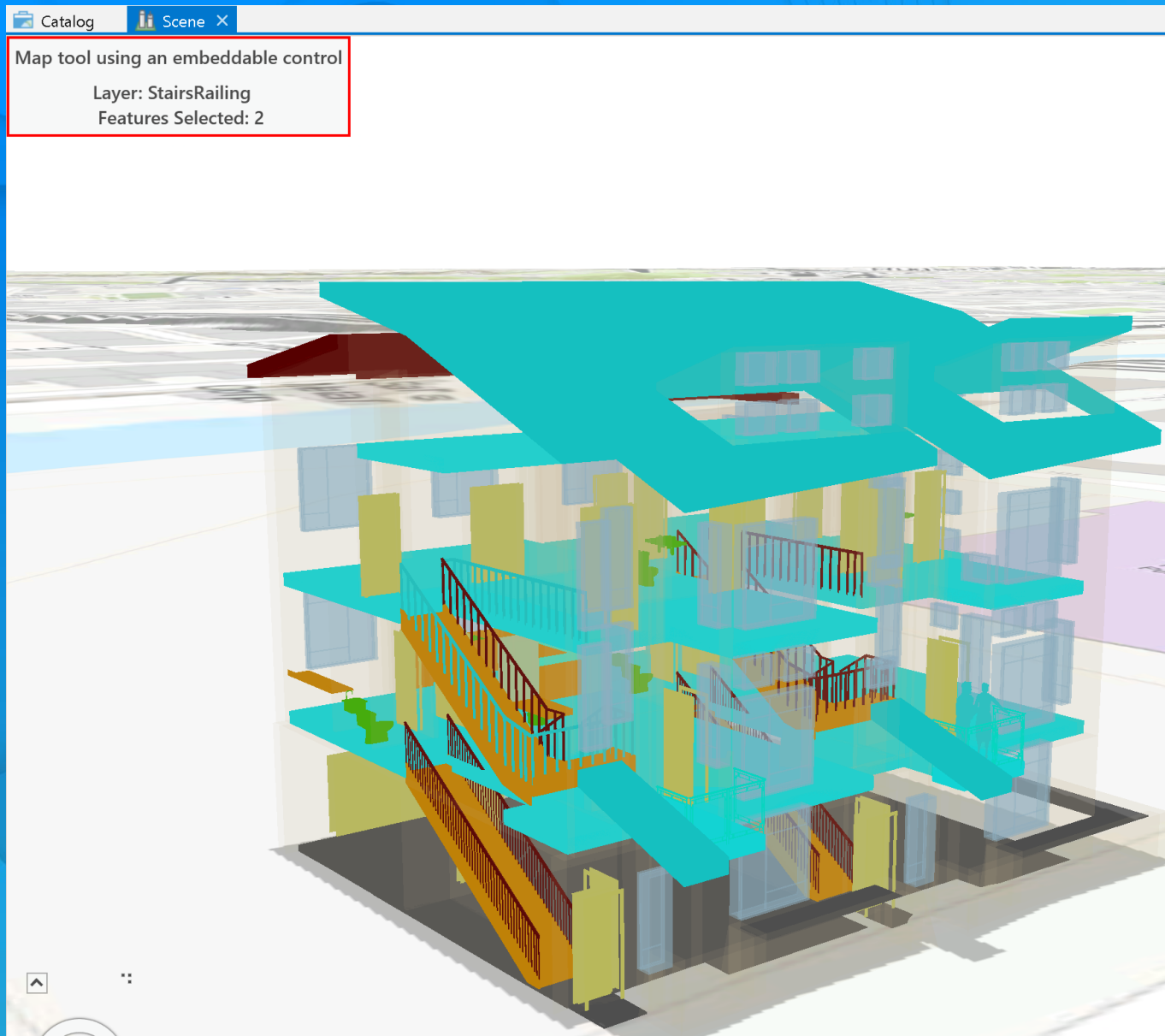
```
public MapTool1()  
{  
    ...  
    //DAML ID of the embeddable control  
    OverlayControlID = "ProAppModule1_EmbeddableControl1";  
    //Specify ratio of 0 to 1 to place the control  
    OverlayControlPositionRatio = new System.Windows.Point(0, 0); //top left  
    //Embeddable control can be resized  
    OverlayControlCanResize = true;  
}
```

# Map tool using an Embeddable Control

Access the control

- Access properties in the Embeddable control View Model
  - Cast OverlayEmbeddableControl property to the type of your View Model to access members.

```
var embeddableControlVM = OverlayEmbeddableControl as EmbeddableControl1ViewModel;  
embeddableControlVM.Text = $"Layer: {Layer}";  
...
```



# Demo: Embeddable control

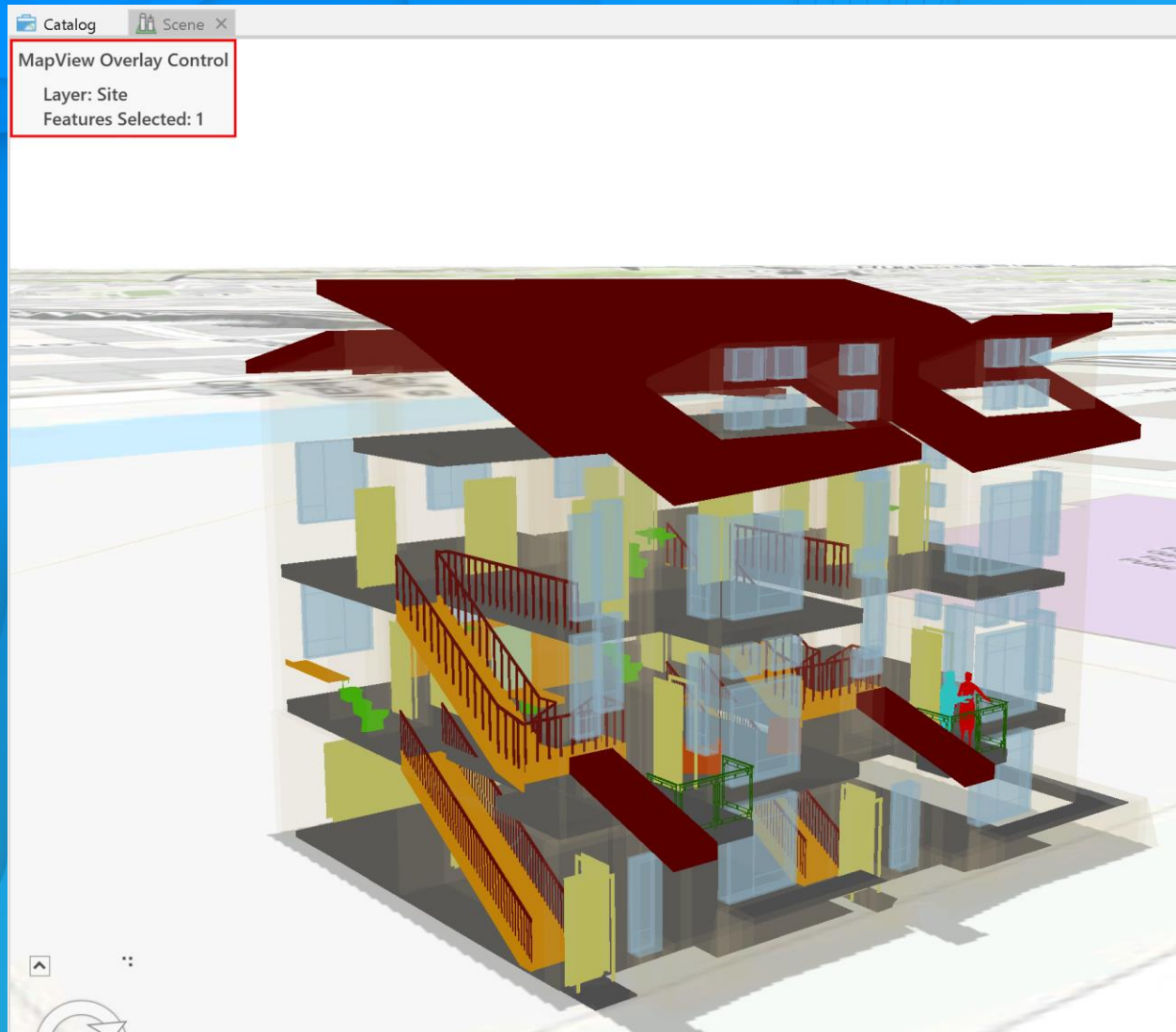
- Town house dataset

# Map View Overlay Control

## Definition

- **What is a Map View Overlay Control?**
  - An overlay control that displays on top of the map view.
  - Similar to Map tool embeddable control, but display state is not controlled by Map tool activation.
- **Use Embeddable Control Pro SDK Item template.**
  - MVVM pattern
  - stubbed out by the Pro SDK item template
- **Map View Overlay Control activation through code.**

```
//Create MapViewOverlayControl  
var myViewOverlayControl = new MapViewOverlayControl(user_control);  
mapView.AddOverlayControl(myViewOverlayControl); //add overlay  
mapView.RemoveOverlayControl(myViewOverlayControl); // remove overlay
```



# Demo: Map View Overlay Control

- Town house dataset

# Custom Popups

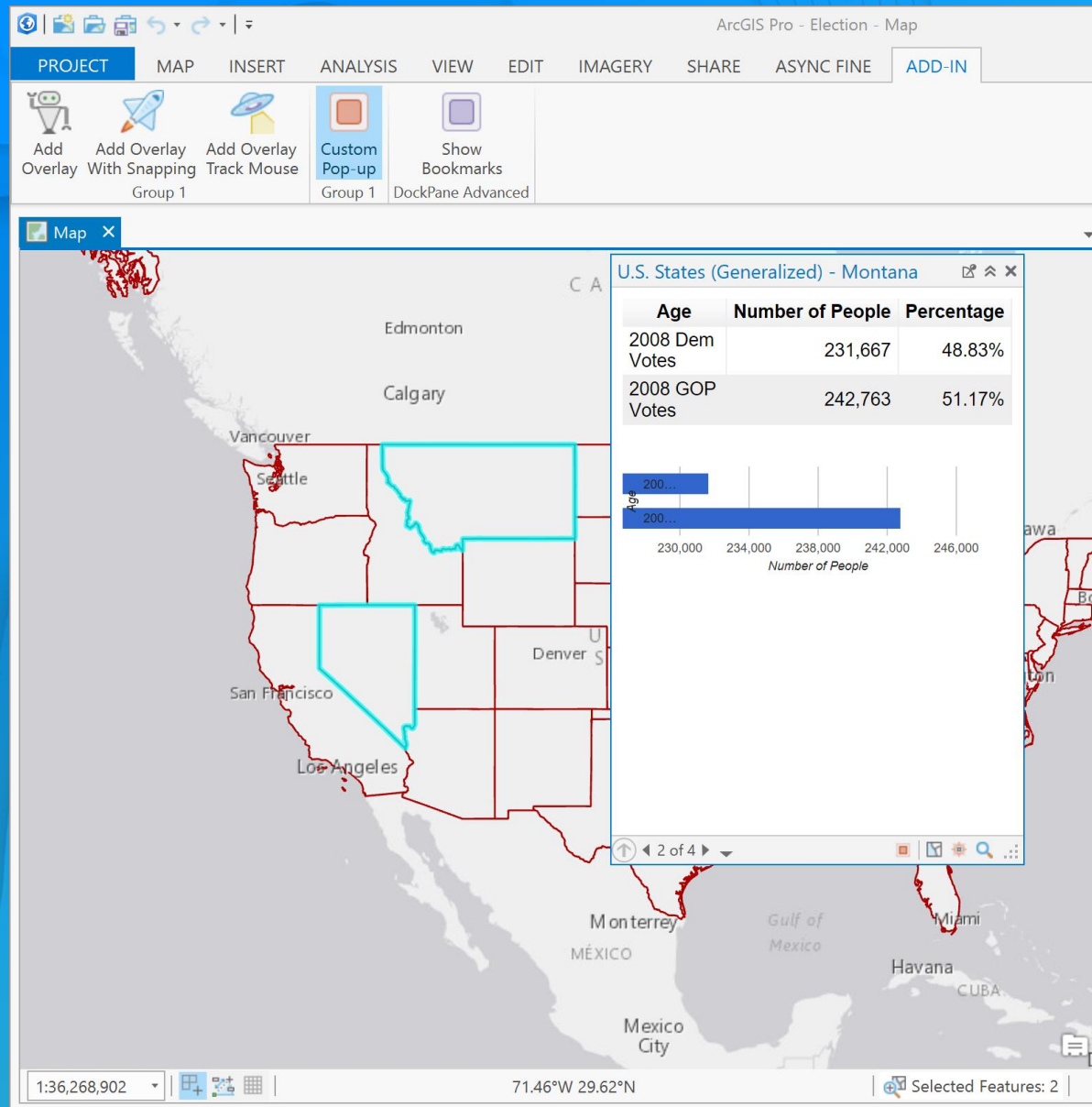
- Custom Popups – implemented in code
  - Uses HTML5, Javascript
  - Not persisted in layer definition
- To invoke in code use MapView ShowCustomPopup overloads

```
public class MapView {  
    public void ShowCustomPopup(IEnumerable<PopupContent> popupContent);  
    public void ShowCustomPopup(IEnumerable<PopupContent> popupContent,  
                                IEnumerable<PopupCommand> commands, bool includeDefaultCommands);  
}
```

# Custom Popup

- Implement your own custom PopupContent
  - Assign MapMember and ID
  - Provide Content – HTML and Javascript

```
var popup = new PopupContent() {  
    MapMember = layer,  
    ID = oid,  
    Title = "Custom Popup",  
    HtmlContent = string.Format(  
        "<head><body><p>{0}: Objectid {1}</p></body></html>", layer.Name, oid)  
};  
  
MapView.Active.ShowCustomPopup(new List<PopupContent>() {popup});
```



# Demo – Custom Popup

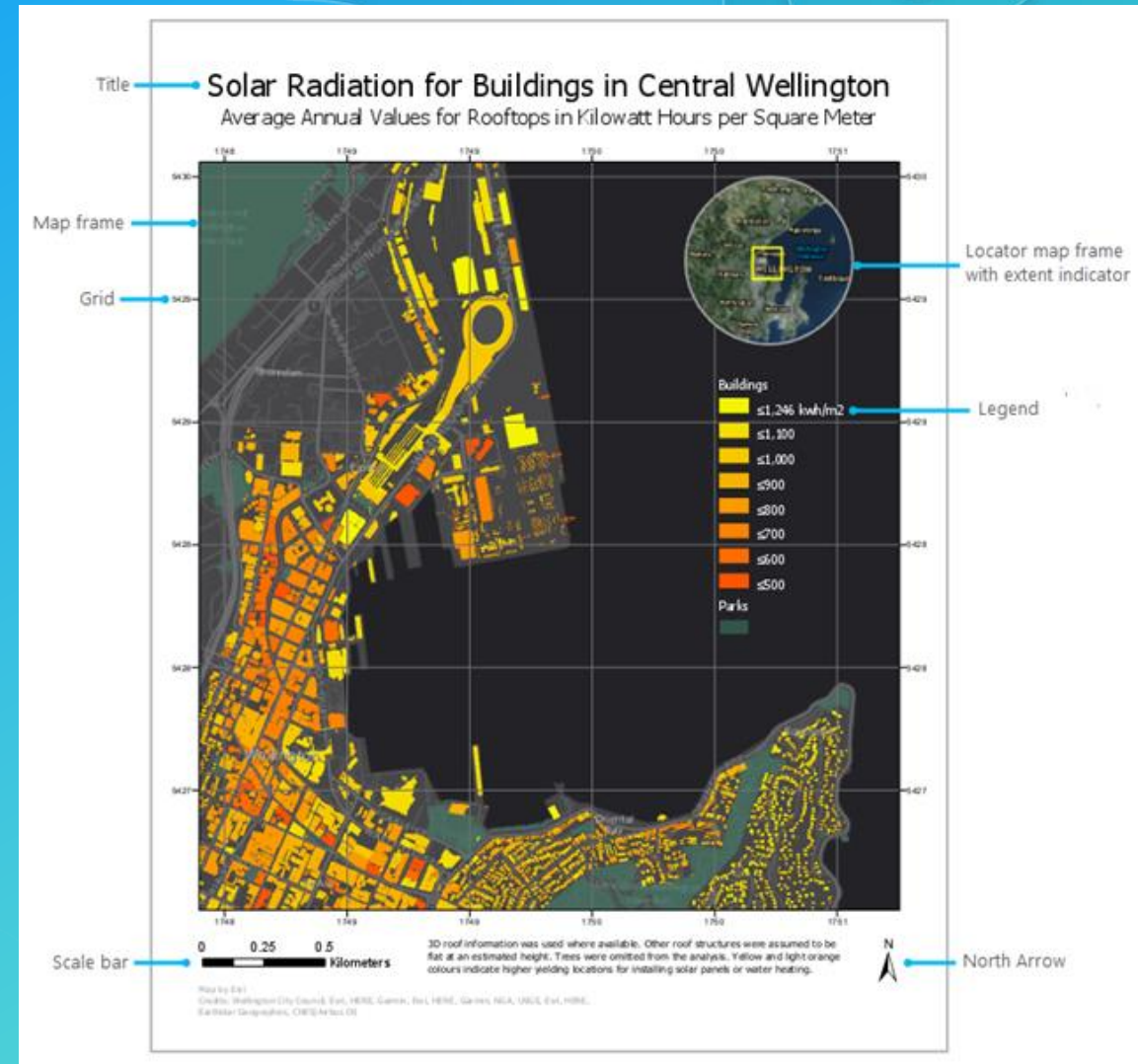
- Election Data



# Creating Layouts using the ArcGIS Pro API

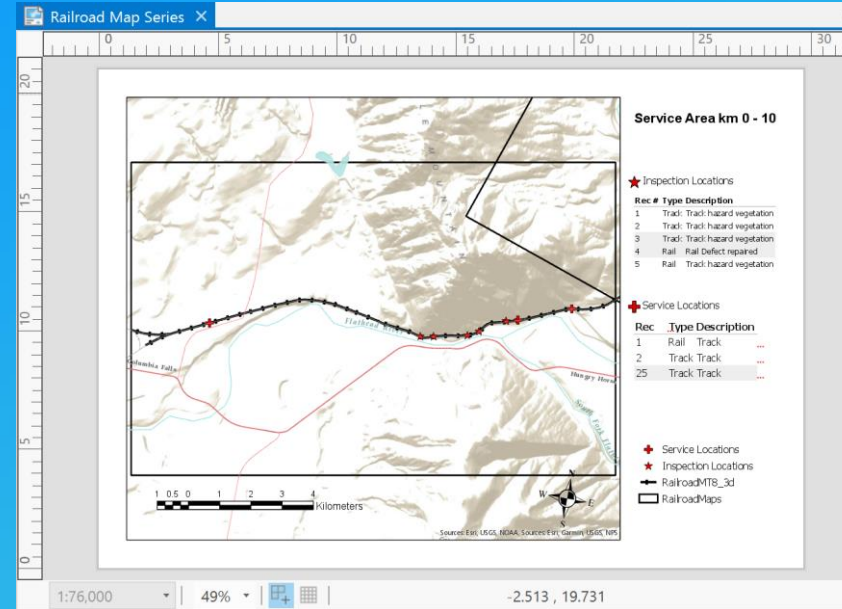
- What is a layout?

- A page layout (aka layout) is a collection of one or more map elements organized on a virtual page designed for map printing. Layouts also include supporting elements, such as a title, a legend, and descriptive text.



# Layout class

- Layout class represents a Page Layout in a Project
- Created by using `LayoutFactory.Instance.CreateLayout`
- Layout Class provides access to basic layout properties
  - Page information (page dimensions like size, unit)
  - Layout Element management (add, delete elements like text, map)
  - Export methods



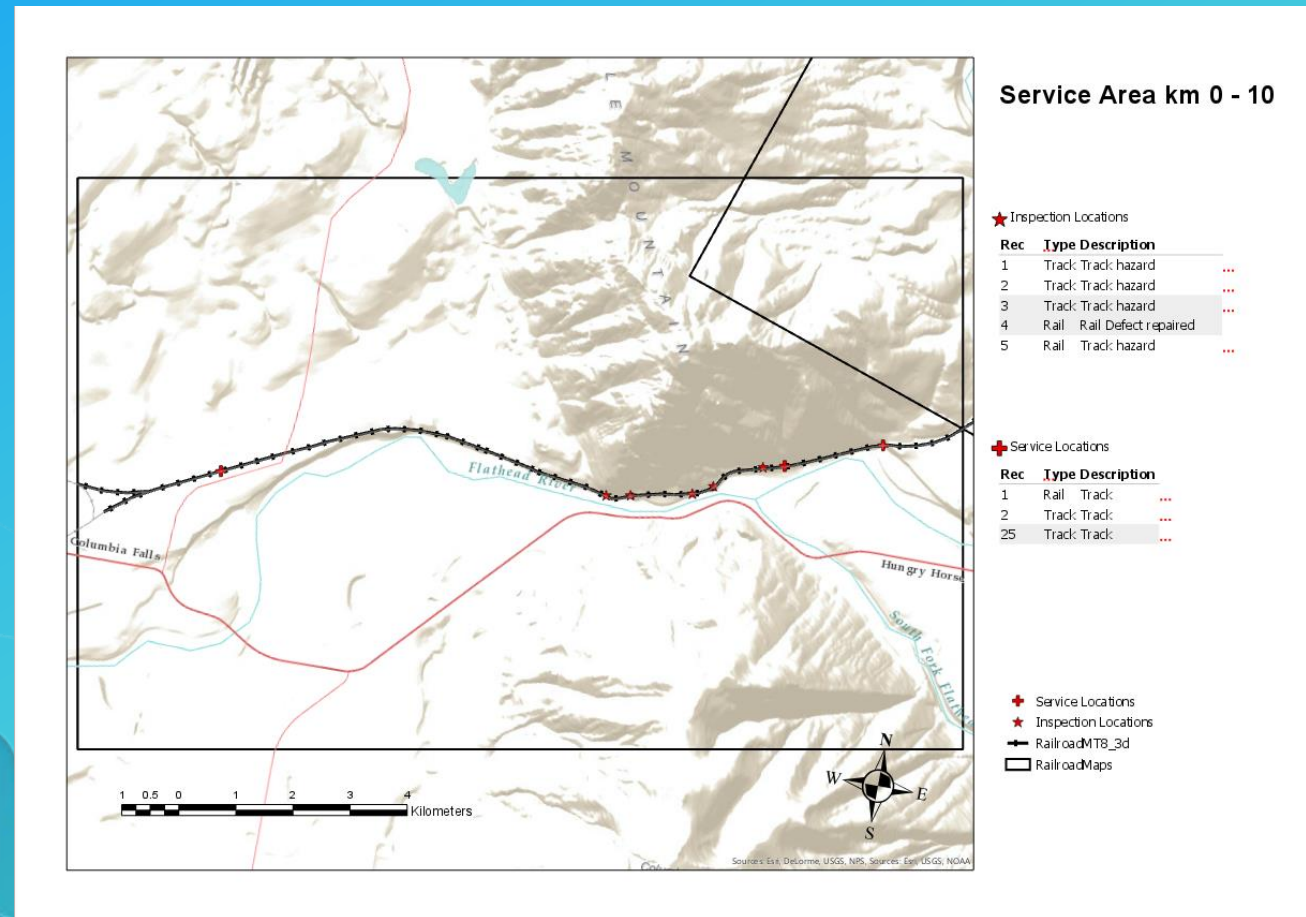
# Layout Class Creation

- Pattern to create a new Layout
  - With page size, unit, and layout name

```
Layout layout = await QueuedTask.Run<Layout>(() =>
{
    Layout lyt = LayoutFactory.Instance.CreateLayout (8.5, 11, LinearUnit.Inches);
    lyt.SetName ("Layout sample");
    return lyt;
});
```

# Layout Elements

- Page Layout contains Layout Elements
- Created by using LayoutElementFactory
- Layout Element Examples:
  - MapFrame
  - Legend
  - NorthArrow
  - PictureElement
  - ScaleBar
  - TextElement
  - TableFrame



# Layout Element Creation

- Using `LayoutElementFactory` class to create any Element
- Create MapFrame: `LayoutElementFactory.Instance.CreateMapFrame`

```
LayoutProjectItem layoutItem = Project.Current.GetItems<LayoutProjectItem>().
    FirstOrDefault(item => item.Name.Equals("My Layout"));
QueuedTask.Run(() => {
    Layout layout = layoutItem.GetLayout();
    // Build envelope geometry for map frame
    Envelope mf_env = EnvelopeBuilder.CreateEnvelope(new Coordinate2D(6.0, 8.5),
                                                    new Coordinate2D(8.0, 10.5));

    // Reference map, create MF and add to layout
    MapProjectItem mapPrjItem = Project.Current.GetItems<MapProjectItem>().
        FirstOrDefault(item => item.Name.Equals("Map"));
    Map mfMap = mapPrjItem.GetMap();

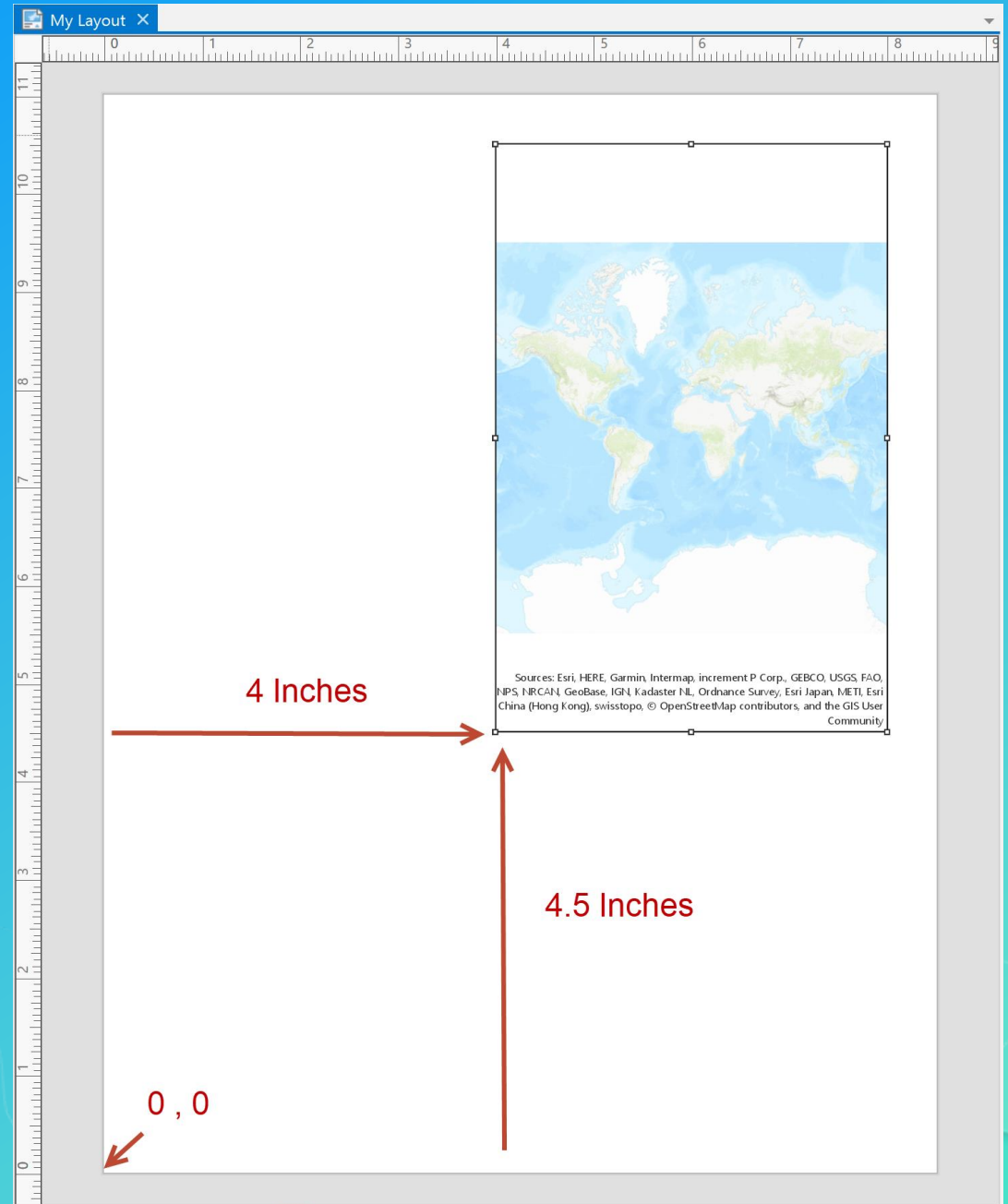
    MapFrame mfElm = LayoutElementFactory.Instance.CreateMapFrame(layout, mf_env, mfMap);

    mfElm.SetName("New Map Frame");
});
```

# Layout Element Coordinates

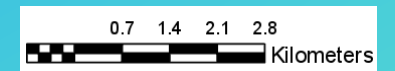
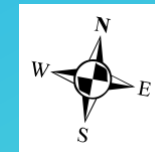
```
// Build envelope geometry for map frame
Envelope mf_env = EnvelopeBuilder.CreateEnvelope
    (new Coordinate2D(4.0, 4.5),
     new Coordinate2D(8.0, 10.5));

// Add map frame to layout
MapFrame mfElm = LayoutElementFactory.
    Instance.CreateMapFrame(layout, mf_env, mfMap);
```



# Layout Element Creation

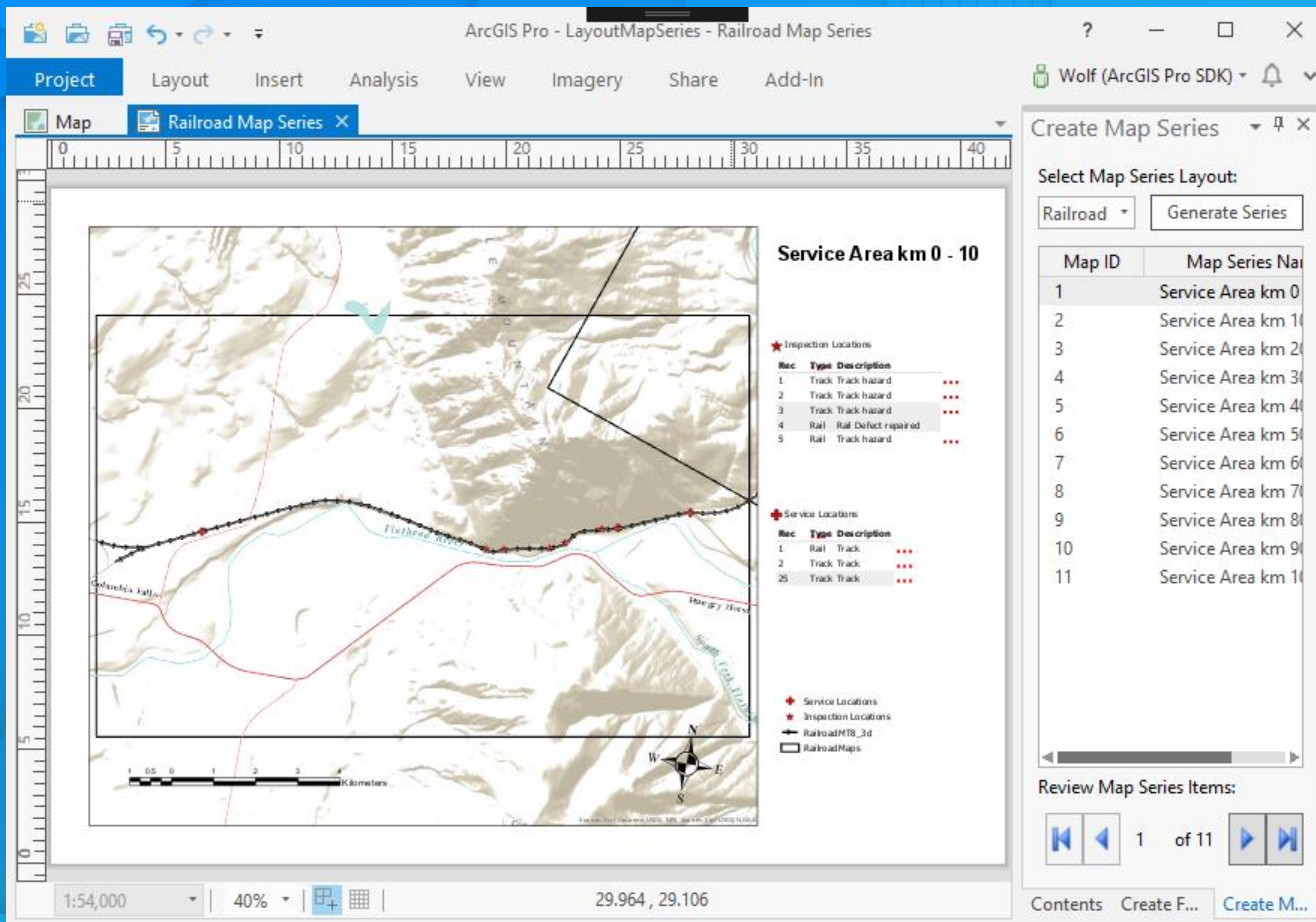
Layout Element	Method to create Layout Element
MapFrame	LayoutElementFactory.Instance.CreateMapFrame
Legend	LayoutElementFactory.Instance.CreateLegend
NorthArrow	LayoutElementFactory.Instance.CreateNorthArrow
PictureElement	LayoutElementFactory.Instance.CreatePictureElement
ScaleBar	LayoutElementFactory.Instance.CreateScaleBar
TextElement	LayoutElementFactory.Instance.CreateTextElement
TableFrame	LayoutElementFactory.Instance.CreateTableFrame
...	...



## ★ Inspection Locations

### Rec # Type Description

1	Track	Track hazard vegetation
2	Track	Track hazard vegetation
3	Track	Track hazard vegetation
4	Rail	Rail Defect repaired
5	Rail	Track hazard vegetation



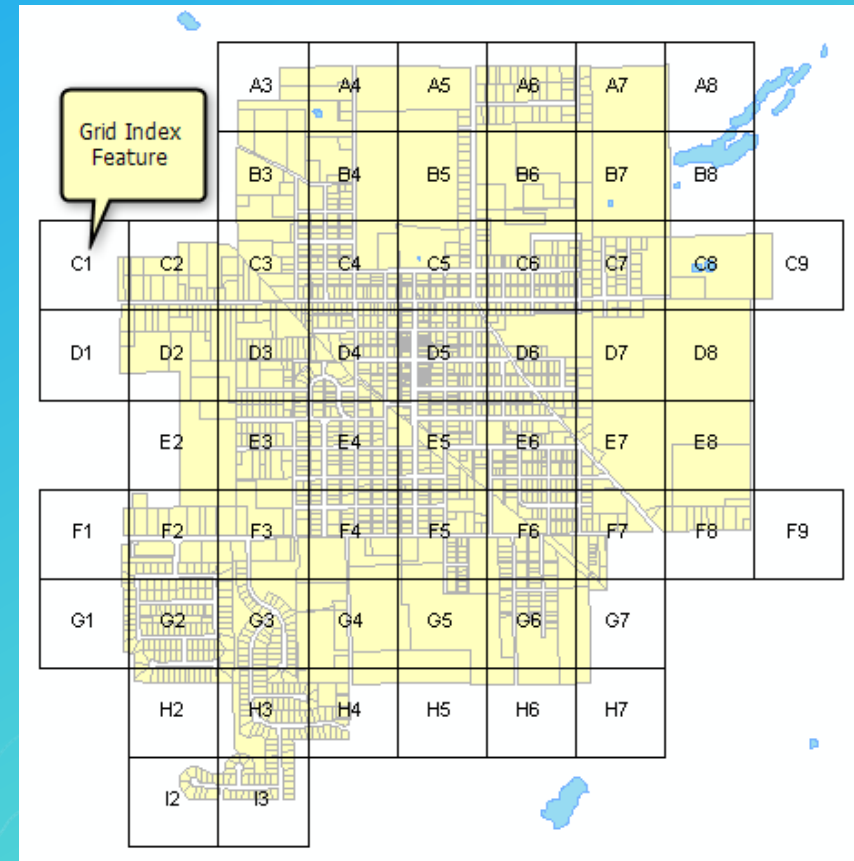
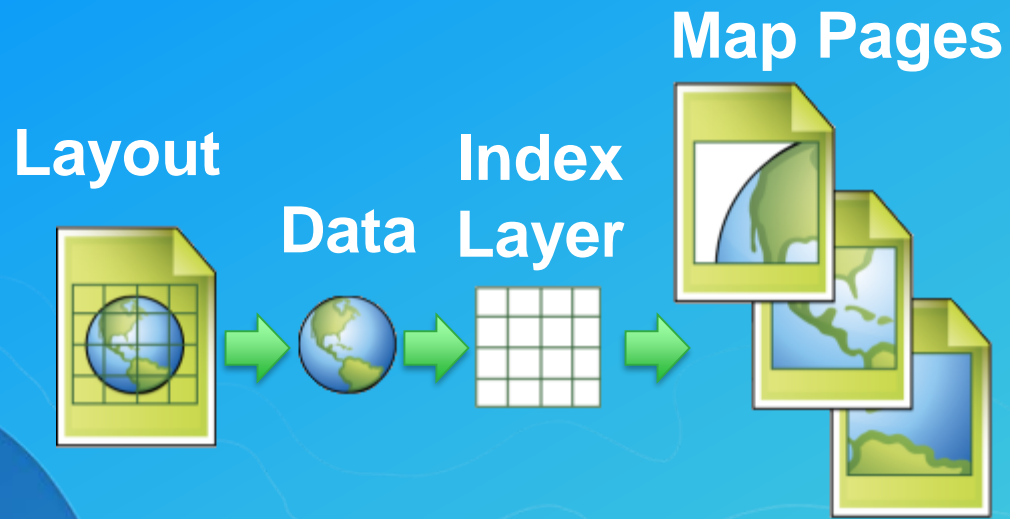
# Demo – Create a Layout with code

- Layout Map Series Data



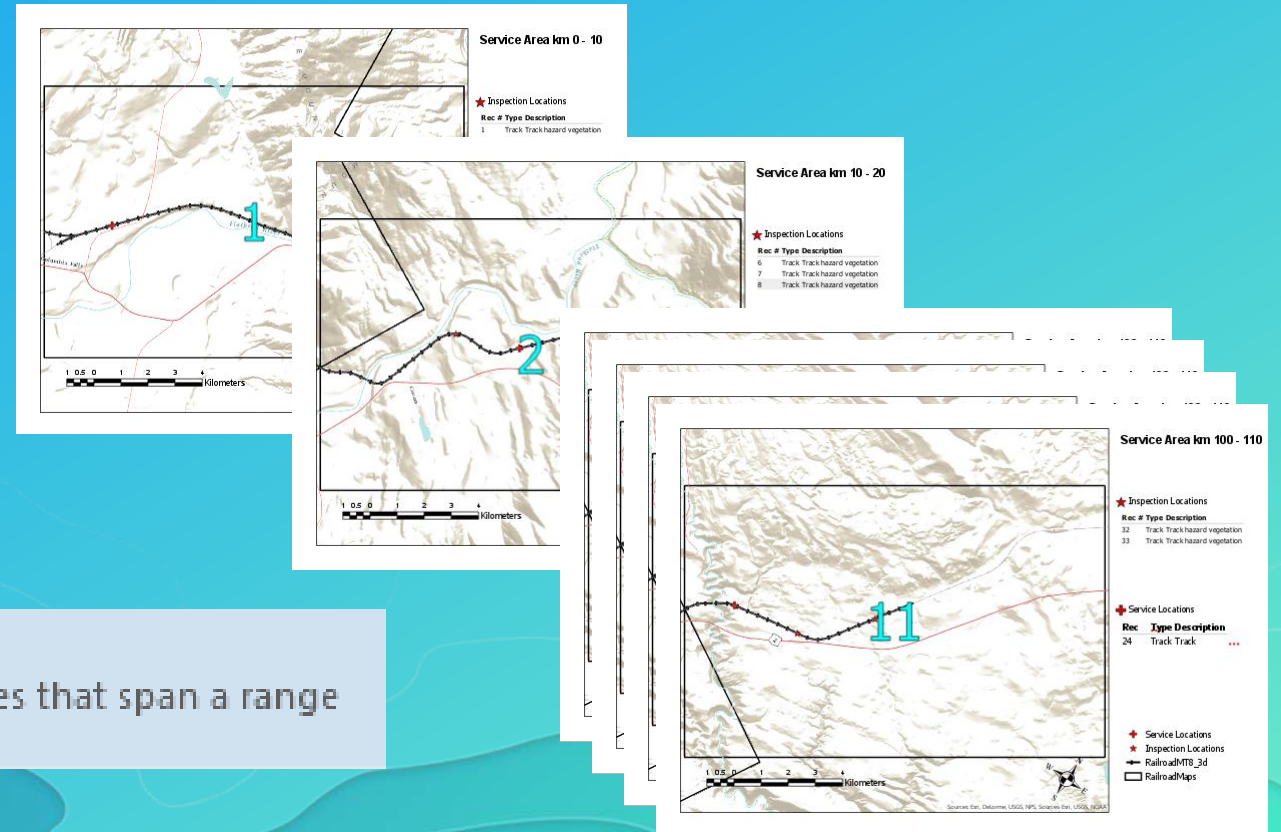
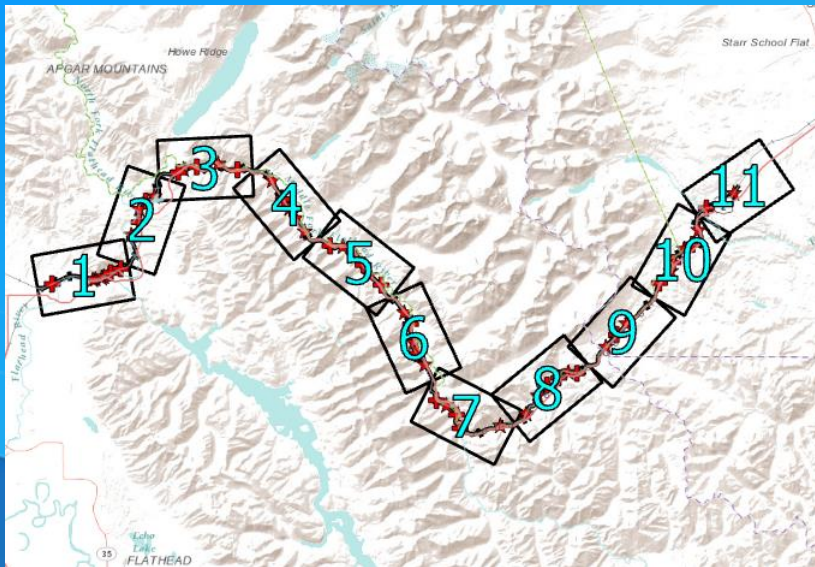
# Layout Creation using a Spatial Map Series

- What is a Spatial Map Series?
  - Collection of map pages built from a single layout that represents a geographic area



# Layout Creation using a Spatial Map Series

- Spatial Map Series used in the next demo
  - Collection of 'service area' map pages built from a single layout that follow the course of railroad tracks



## Spatial

Define a series of pages that span a range of map extents.

# Layout class & CIMLayout class

- **Layout contains a *CIMLayout***
  - **CIM = Cartographic Information Model**
  - **Layout exposes the most commonly used aspects of a layout page as properties and methods, the remaining properties and methods are available via CIMLayout**
- **Access to CIMLayout**
  - **Use `Layout.GetDefinition ()` to get the CIMLayout**
  - **Use `Layout.SetDefintion (changed CIMLayout)` to update the CIMLayout**

# CIMSpatialMapSeries class

- *The CIMLayout.CIMSpatialMapSeries* property is used to define a spatial map series
- Define a spatial map series:
  - Create a new *CIMSpatialMapSeries* instance with the following minimal settings
    - *IndexLayerURI*: used to define the CIMPath for the index polygon for the series (example: "CIMPATH=map/railroadmaps.xml")
    - *SortField*: sequence and sort field (in *IndexLayerURI*) to uniquely define each page in the series
    - *CurrentPageId*: current page of the map series displayed in layout view using the *IndexLayerURI.SortField* values
    - *RotationField*: name of field in *IndexLayerURI* used to 'rotate' each page in map series

# Layout: Enable Spatial Map Series

- Sample snippet to create a map series for a layout

```
var CimSpatialMapSeries = new CIMSpatialMapSeries()
{
    Enabled = true,
    StartingPageNumber = 1,
    CurrentPageID = 1,
    IndexLayerURI = "CIMPATH=map/railroadmaps.xml",
    SortField = "SeqId",
    RotationField = "Angle",
    ...
};
// set the map series definition in the layout
CIMLayout layCIM = layout.GetDefinition();
layCIM.MapSeries = CimSpatialMapSeries;
layout.SetDefinition(layCIM);
```

ArcGIS Pro - LayoutMapSeries - Railroad Map Series

Project Layout Insert Analysis View Imagery Share Add-In

Map Railroad Map Series

0 5 10 15 20 25 30 35 40

**Service Area km 0 - 10**

★ Inspection Locations

Rec	Type	Description	
1	Track	Track hazard	...
2	Track	Track hazard	...
3	Track	Track hazard	...
4	Rail	Rail Defect repaired	...
5	Rail	Track hazard	...

★ Service Locations

Rec	Type	Description	
1	Rail	Track	...
2	Track	Track	...
25	Track	Track	...

★ Service Locations

★ Inspection Locations

— RailroadMTB\_3d

□ RailroadMaps

1 0.5 0 1 2 3 Kilometers

1:54,000 | 40% | 29.964, 29.106

Wolf (ArcGIS Pro SDK)

Create Map Series

Select Map Series Layout:

Railroad Generate Series

Map ID	Map Series Name
1	Service Area km 0
2	Service Area km 10
3	Service Area km 20
4	Service Area km 30
5	Service Area km 40
6	Service Area km 50
7	Service Area km 60
8	Service Area km 70
9	Service Area km 80
10	Service Area km 90
11	Service Area km 100

Review Map Series Items:

1 of 11

Contents Create F... Create M...

# Demo – Layout with Map Series

- Layout Map Series Data

# ArcGIS Pro SDK for .NET Tech Sessions

Date	Time	ArcGIS Pro SDK for .NET Tech Sessions	Location
Wed, Mar 07	10:30 am - 11:30 am	Mapping and Layout	Pasadena/Sierra/Ventura
	1:00 pm - 2:00 pm	Advanced Pro Customization and Extensibility	Santa Rosa
	2:30 pm - 3:30 pm	Pro Application Architecture Overview & API Patterns	Mesquite G-H
	4:00 pm - 5:00 pm	Advanced Editing and Edit Operations	Santa Rosa
Thu, Mar 08	5:30 pm - 6:30 pm	Working with Rasters and Imagery	Santa Rosa
Fri, Mar 09	8:30 am - 9:30 am	An Overview of the Utility Network Management API	Mesquite G-H
	10:00 am - 11:00 am	Beginning Pro Customization and Extensibility	Primrose A
	1:00 pm - 2:00 pm	Advanced Pro Customization and Extensibility	Mesquite G-H

# ArcGIS Pro SDK for .NET Demo Theater Presentation

Date	Time	ArcGIS Pro SDK for .NET Demo Theater Presentation	Location
Wed, Mar 07	5:30 pm - 6:00 pm	New UI Controls for the SDK	Demo Theater 2 - Oasis 1
	6:00 pm - 6:30 pm	Raster API and Manipulating Pixel Blocks	Demo Theater 2 - Oasis 1





esri

THE  
SCIENCE  
OF  
WHERE



# MapControl

- Reusable control for displaying map content
  - Embed in dock pane, dialog, any user control.
  - Can display 2D and 3D maps, layers, layer packages
  - Dependency properties such as ViewContent, Camera.

```
<UserControl x:Class="MapControlDemo.Dockpane1"
  xmlns:controls=
  "clr-namespace:ArcGIS.Desktop.Mapping.Controls;assembly=ArcGIS.Desktop.Mapping">
  <Grid>
    <controls:MapControl Name="mapControl" VerticalAlignment="Bottom"
      HorizontalAlignment="Stretch" Height="275"
      ViewContent="{Binding Path=MapContent}"
      Camera="{Binding CameraProperty}">
    </controls:MapControl>
  </Grid>
</UserControl>
```

# MapControl

- **MapControl Events**
  - ControlInitialized
  - ViewContentLoaded
  - CameraChanged
  - ExtentChanged
  
- **Commands**
  - NextCamera
  - PreviousCamera
  - ZoomInFixed
  - ZoomOutFixed
  - ZoomToFullExtent



# PopupContent

- Implement `OnCreateHtmlContent` callback for Dynamic Content:
  - Defers HTML content creation until the popup is shown
  - Derive your own custom class from `PopupContent`
    - Set `IsDynamicContent = true`
    - Implement `OnCreateHtmlContent()`

```
internal class MyPopupContent : PopupContent {  
    public MyPopupContent() {  
        IsDynamicContent = true;  
    }  
    protected override Task<string> OnCreateHtmlContent() {  
        //TODO - must implement to create HTML "on demand"  
    }  
}
```