ArcGIS Pro SDK for .NET: UI Design and MVVM

- Wolfgang Kaiser, Uma Harano
Session Overview

• Asynchronous Programming: Introduction to QueuedTask
  - Use of async and await
  - Authoring custom asynchronous functions

• Overview of MVVM
  - Dockpane example
    - View and View Model Implementation in Pro
    - Hooking functionality of existing Pro Commands into your Add-in

• Other Framework Elements
  - Newly added Framework Elements
  - Gallery Plug-in
Asynchronous Programming

• ArcGIS Pro is a multi-threaded 64 bit application
• Important asynchronous programming patterns for the SDK:
  - Async / Await
  - Using the Pro Framework’s QueuedTask class
• Asynchronous Programming allows you to keep the User Interface responsive!
ArcGIS Pro Internal Threading Model

• **ArcGIS Pro is multi-threaded**
  - Incorporates the latest asynchronous language features from Microsoft
  - Implements a threading infrastructure tailored to reduce complexity.

• **Add-In developers should only need to contend with two threads:**
  - The GUI thread
  - A single specialized worker thread called the Main CIM Thread, MCT
    - Internally, ArcGIS Pro uses a large number of threads for:
      - Rasterization, rendering, data loading, GP
      - But all this is Isolated from the API

• **Simplifies coding, ensures consistency of Pro state.**
Categories of Methods in ArcGIS Pro API

• **Coarse-grained asynchronous** methods:
  - Can be called on any thread

• **Finer grained synchronous** methods:
  - Must be called within a QueuedTask
Coarse Grained Methods

- Can be called from any thread. Typically invoked from the UI thread
  - They execute in the background on Pro internal threads
  - Use async/await semantic

//Execute a GP Tool
await Geoprocessing.ExecuteToolAsync("SelectLayerByAttribute_management",
  new string[] {"parcels","NEW_SELECTION", "description = 'VACANT LAND'"});
await MapView.Active.ZoomToSelectedAsync(new TimeSpan(0, 0, 3));
Demo:
Coarse Grained Methods
Fine Grained, Synchronous Methods

Must be called within a QueuedTask

- A much greater number of fine grained methods and classes
- No async/await. Runs on the MCT
- Designed for aggregation into your own coarse-grained async methods
- In other words: this allows you to write your business logic as a ‘background’ task

```csharp
{
    var layers = MapView.Active.Map.FindLayers("Parcels")
        .OfType<FeatureLayer>().ToList();
    var parcels = layers[0] as FeatureLayer;
    QueryFilter qf = new QueryFilter()
    {
        WhereClause = "description = 'VACANT LAND'",
        SubFields = "*"
    };
    parcels.Select(qf, SelectionCombinationMethod.New);
});
```
QueuedTask

- QueuedTask uses the Pro framework’s custom Task scheduler
- Used to run synchronous ArcGIS Pro SDK methods in the background
- These synchronous methods are listed in the API Reference guide like this:
  
  "This method must be called on the MCT. Use QueuedTask.Run"

- Example of synchronous methods in Pro:
  - GetSpatialReference, QueryExtent, Geometry operations

- Usage:

```csharp
Task t = QueuedTask.Run(() =>
{
    // Call synchronous SDK methods here
});
```
Demo:

Fine Grained Methods
MVVM Pattern in Add-ins

• MVVM and variants are de-facto pattern for WPF UI implementations

• Pro MVVM is built on top of ActiPro (http://www.actiprosoftware.com/products/controls/wpf )

• The Basic Pattern is:
  - ViewModel declared in DAML and implemented in code
  - View referenced in DAML and implemented as WPF UserControl
  - Model is optional

• Note: To customize the Pro UI, you must use its MVVM Framework. Substitutes are not allowed.
MVVM Pattern in Add-ins

- Model-View-ViewModel (MVVM) Pattern used for many of the Framework elements
  - Dockpane
  - Pane
  - Custom Control
  - Embeddable Control
  - Property Page
Demo: New Dockpane using MVVM Dockpane template
MVVM Implementation in Add-ins

- MVVM implementation in Add-ins follows the same Pattern used in WPF / .Net
  - Model: Classes that represent the data consumed by the app
  - View: User interface (UI) elements the user interacts with (XAML)
  - ViewModel: Classes that wrap data (coming from a model) and provide business logic for the UI (views)
- Implement your Add-in UI just as you implement a user control in WPF/.Net
  - You can use many available online WPF MVVM snippets
- Differences in MVVM for Add-ins versus WPF applications:
  - Multi-threading considerations
  - ArcGIS Pro Styling
Multi-threading considerations

- **ArcGIS Pro Framework’s managed threading model:**
  - Framework provides `QueuedTask` to guarantee that UI actions happen in a sensible order without corruption

- **Updating UI collections from a worker thread**
  - Locking is required when sharing objects across threads

- **Recommended pattern for updating collections from a worker thread**
  - Found in .Net `BindingOperations` helper class: `BindingOperations.EnableCollectionSynchronization`

```csharp
private readonly object _lockListOfBookmarks = new object();

protected override Task InitializeAsync()
{
    BindingOperations.EnableCollectionSynchronization(ListOfBookmarks, _lockListOfBookmarks);
    GetBookMarkCollection();
    return base.InitializeAsync();
}
```
Add-in Styling

- New at 1.4 is Dark Theme 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
Demo: Dockpane
Bookmarks
Hooking Existing ArcGIS Pro Commands

- Get any Pro control’s ICommand and use it in your add-in:
  - Done by using the Pro Framework’s “GetPlugInWrapper” method.
- A Pro control’s command can be added to your add-in button’s click method. (or anywhere else in your add-in).

```javascript
// ArcGIS Pro's Create button control DAML ID.
var commandId = DAML.Button.esri_mapping_createBookmark;
// get the ICommand interface from the ArcGIS Pro Button
// using command's plug-in wrapper
// (note ArcGIS.Desktop.Core.ProApp can also be used)
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);
}
```
Hooking Existing ArcGIS Pro Commands

- Adding a button to the Dockpane to run the ‘Close ArcGIS Pro’ Command

```csharp
protected DockpanelViewModel()
{
    CloseCmd = FrameworkApplication.GetPlugInWrapper(DAML.Button.esri_core_exitApplicationButton)
        as ICommand;
}

public ICommand CloseCmd { get; set; }
```

- Adding a button to the Dockpane with our ‘custom’ Zoom in behavior

```csharp
protected DockpanelViewModel()
{
    ZoomInCmd = new RelayCommand(() => MappingModule.ActiveMapView.ZoomInFixedAsync(),
        () => MappingModule.ActiveMapView != null);
}

public ICommand ZoomInCmd { get; set; }
```

- `RelayCommand` is an implementation of `ICommand` which lets you specify your own implementation of `Execute` and `CanExecute`
Demo:
Hooking Existing Pro Commands
Framework Elements

- **Any Framework Element is an extensibility point**
  - Controls (Button, Tool, and variants)
    - Hosted on Ribbons, Menus, Galleries
    - Checkbox, Combobox, Label Control, Custom Controls
  - Tabs, Tab Groups
  - Toolbars
  - Menus, Context Menus
  - Panes
  - Dockpanes
  - Galleries
  - Property Sheets
- **All Elements have a definition within DAML**
Framework Elements

- **Majority of Framework Elements are represented by Visual Studio Item templates**
  - Automates generation of DAML
  - Add relevant code-behind files to the project

- **Some Framework Elements need a much higher degree of customization (than a template can provide)**
  - Custom Control
  - Gallery
  - Dynamic Context Menus

- **Note: Complete element reference is here:**
Framework Elements

• New Framework elements:
  - Burger button – using Pro Framework’s ContextMenu class
  - Circular animation
  - Message label
  - Search textbox
  - Waiting cursor

Framework Element: Burger Button

- Add Context menu in config.daml

```
<menus>
  <menu id="DockPaneBookmarkAdvanced_Bookmark_Menu" caption="Change view" contextMenu="true">
    <button refID="DockPaneBookmarkAdvanced_BookmarkOutline_MenuButton" />
    <button refID="DockPaneBookmarkAdvanced_BookmarkGallery_MenuButton" />
  </menu>
</menus>
```

- Bind an instance of context menu to the BurgerButton’s PopupMenu attribute.

```
<extensionsControls:BurgerButton PopupMenu="{Binding BurgerButtonPopupMenu}"/>

{
    get { return FrameworkApplication.CreateContextMenu(MenuId); }
}
```

- Implement Popupmenu options in the Framework button class OnClick method.
Demo: Controls

Framework controls that can be used to polish your dockpane UI.
**Gallery**

- Container control, displays a collection of related items in rows and columns
- If too many items are in the gallery an expand arrow is provided
- Contents are populated at run time
- Individual gallery items are modelled using `GalleryItem` class
- Style with a `GalleryTemplate.xaml`
Gallery

- Shown as a split-button with a dropdown that exposes gallery
- Do any content initialization in the Gallery code-behind OnDropDownOpened
  - GalleryItems are created at runtime
  - Use Add method to add items to gallery
  - Whenever a gallery item is clicked, the Gallery OnClick is called with the clicked GalleryItem as a parameter

```csharp
class Gallery1 : Gallery {
    protected override void OnDropDownOpened() {
        //Handle populating gallery content
        foreach (var dataItem in lstWebmapItems)
            Add(dataItem);
    }
    private void Initialize() {
        //TODO: Init content
    }
```
• DAML gallery definition:

```xml
<!-- Config.daml
<galleries>
  <gallery id="TimeNavigation_BkmGallery"
    rows="4" itemsInRow="4" # of rows and columns
    dataTemplateFile="TimeBkmGalleryTemplate.xaml" # of rows and columns
    templateID="TimeBkmItemTemplate"> # Resource key
    <button refID="esri_mapping_createBookmark" /> # Child button
  </gallery>
</galleries>

<!-- TimeBkmGalleryTemplate.xaml
<DataTemplate x:Key="TimeBkmItemTemplate">
  <StackPanel Orientation="Vertical" ...
    <Image Source="{Binding Icon}" ...
    <TextBlock Text="{Binding Text}" ...
```
Gallery

- Galleries can also be defined as “inline”
- Inline shows gallery items directly in the ribbon (not in a dropdown)
- Set the inline attribute on the gallery reference to true
- (Done automatically for you via the inline gallery template)
Demo: Gallery

Inline gallery control that hosts a collection of webmaps.
ArcGIS Pro SDK Sessions

Technical Workshops

ArcGIS Pro SDK for .NET: UI Design and MVVM
• Tuesday, July 11, 8:30 am – 9:45 am. Location: SDCC – Room 33 A
• Wednesday, July 12, 8:30 am – 9:45 am. Location: SDCC – Room 33 A

ArcGIS Pro SDK for .NET: Configurations
• Wednesday, July 12, 1:30 pm – 2:45 pm. Location: SDCC – Room 32 A

Demo Theaters

Beginning Pro SDK Project Development: Tips and Tricks for Troubleshooting
• Tuesday, July 11, 2:30 pm – 3:15 pm. Location: SDCC – Demo Theater 09 – Technical Support

Getting Started with ArcGIS Pro SDK Add-Ins and Configurations
• Tuesday, July 11, 3:30 pm – 4:15 pm. Location: SDCC – Demo Theater 11 – Developer
• Wednesday, July 12, 11:30 am – 12:15 pm. Location: SDCC – Demo Theater 11 – Developer
Many Pro SDK Resources

- **SDK home page** – main resource page
- **Esri Training** – instructor-led Esri training course
- **Documentation Wiki** – primary documentation site with concept and guide docs, and much more
- **Community Samples** – ready to use code solutions categorized by functional area
- **Snippets** – code snippets by functional area
- **GeoNet Pro SDK Group** – developer community
- **API Reference** – full API reference
- **FAQ** – answers to common questions
- **Blog posts** – focused on the Pro SDK
Pro SDK Training

- Extending ArcGIS Pro with Add-Ins – Esri Instructor-led training course on the Pro SDK
- Great way to get a comprehensive introduction
- Online offerings – very interactive and productive
- Esri.com/training