ArcGIS Viewer for Silverlight
Advanced Topics

Rich Zwaap
Agenda

- Add-ins overview
- Tools
- Behaviors
- Controls
- Layouts
- Road ahead
Add-Ins

Adding functionality to the Viewer
Add-Ins – Extending the Viewer

• Application Builder does a lot
  - Map configuration
  - Standard tools
  - Look and feel
  - Branding (title, logo, etc)

• Add-ins enable specific workflows
  - “Add-in” → code module containing functionality
    - Compiled as Silverlight packages (.xap files)
  - Seamlessly plug-in to App Builder configuration
    - Encourages reusability
Add-Ins – Required Software

- Microsoft Visual Studio 2010 SP1
- Microsoft Expression Blend 4 SDK
- Microsoft Silverlight 4 Toolkit

- **ArcGIS Extensibility SDK for Silverlight 2.4**
  - Visual Studio 2010 template provided
Add-Ins – Development

• Develop using the Extensibility API
  - Included in the Extensibility SDK

• `ESRI.ArcGIS.Client.Extensibility` assembly
  - Lightweight API for Viewer
  - Provides access to map, selected layer, and pop-up
  - Method to easily show UI in dialogs
  - Hooks to store and load add-in configuration

• Use the Visual Studio Template to start

• Any Silverlight library can be referenced
Add-Ins – Extensibility Points

Several types of add-ins:

1. **Tools** → functionality initiated by tool on toolbar
2. **Behaviors** → non-UI logic
3. **Layouts** → custom application “look and feel”
4. **Controls** → UI integrated into application
Demo
Add-In Types
Add-In Types
Tools
Tools – Overview

- Simplest, most common extensibility endpoint
- Logic initiated by user click
- Appear as buttons on toolbars and menus
- Users can add them from Application Builder
Tools – Implementation

- **Implement ICommand**
  - Execute – tool logic
  - CanExecute – determine whether tool is enabled or disabled
  - CanExecuteChanged – raise if enabling/disabling is needed

- **Add Attributes:**
  - DisplayName*, Description, Category, DefaultIcon
  - Export* (MEF attribute)

*required
Tools – Extensibility API Hooks

- MapApplication.Current
  - Hook to Viewer properties
  - Always available (static property)

- MapApplication.Current.ShowWindow
  - Show UI in a dialog
  - Displays any FrameworkElement
  - WindowStyle determines whether window appearance matches Application Builder or current Viewer
Tools – More Extensibility API Hooks

- `MapApplication.Current.Map`
  - Current Viewer’s map

- `MapApplication.Current.SelectedLayer`
  - Currently selected layer in Map Contents
  - Useful for tools designed for the layer context menu

- `MapApplication.GetLayerName/SetLayerName`
  - Gets or sets name shown in Map Contents

- `PopupInfo`
  - Always passed to Execute for tools on Pop-up toolbar
  - Can retrieve from `MapApplication.Current.GetPopup`
  - Contains info about field and pop-up configuration
Demo Extensibility API
Tools – Configuration

- Can implement configurability for Application Builder
  - Gives designer flexibility in how a tool works
  - Reusability

- Out-of-the-box Examples
- Basemap Gallery
- Print

[Diagrams showing built-in UI and custom UI]
Tools – Implementing Configuration

- **ISupportsConfiguration**
  - **Configure** – fired when user wants to configure tool
  - **SaveConfiguration** – serialize tool config
  - **LoadConfiguration** – deserialize tool config and initialize

```csharp
#region ISupportsConfiguration members

public void Configure()
{
    // Configuration logic
}

public void LoadConfiguration(string configData)
{
    // Load the saved configuration
}

public string SaveConfiguration()
{
    // Serialize the configuration to a string
}
#endregion
```
Demo
Tool Configuration
Tools – Integrated Configuration

- Can show tool configuration UI within Add Tool wizard
- Seamless integration with built-in UI

Out-of-the-box example - Geoprocessing

Built-in UI

Custom UI
Tools – Implementing Wizard Configuration

- **ISupportsWizardConfiguration**
- **Same members as ISupportsConfiguration**, plus:
  - **Pages** – pages shown for configuring the tool
  - **DesiredSize** – how much space you want for your UI
  - **CurrentPage** – the page currently being shown
  - **PageChanging** – called by host when page is about to change
  - **OnCompleted** – called by host on wizard completion
  - **OnCancelled** – called by host on wizard cancellation
Add-In Types
Behaviors
Behaviors – Overview

• Functionality without user interaction
• Examples
  • Introductory/terms of use dialog
  • Initialize extent from query string
  • Show coordinates
  • Constrain extent
Behaviors - Implementation

- Inherit from Behavior<Map>
  - Override OnAttached and OnDetached

- Optionally implement ISupportsConfiguration
  - Configure, LoadConfiguration, SaveConfiguration

- Add attributes for DisplayName, Category
  - Don’t forget MEF Export attribute!

```csharp
[Export(typeof(Behavior<Map>))]
[DisplayName("My Behavior")]
public class MyBehavior : Behavior<Map>, ISupportsConfiguration {
```
Demo Behaviors
Add-In Types
Layouts
Layouts – Overview

- Define application look and feel
- Loose xaml files – all markup, no code
- Position of viewer elements
  - Side panel
  - Navigation control
  - Scalebar
  - Title, logo, links
  - More
- Styles used by Viewer
  - Tool buttons
  - Pop-ups
  - Navigation control
  - More
Layouts – Implementation

• Start with copy of an existing layout
• Open-ended
  - Anything that can be declared in XAML
• Some well-known elements expected by Viewer
  - If it has an x:Name, don’t remove it
Layouts – Deployment

• Copy to Builder\Templates\Default\Config\Layouts
• Create preview image
  • Name the same as xaml file
  • Ideal size is 580 x 359
• Tip - clear browser cache
Demo Layouts
Add-In Types
Controls
Controls – Overview

• Pieces of UI that are “built-in”

• Included in layout

• Can implement a tool to toggle on/off

• Out-of-the-box examples
  • Map
  • Navigation control
  • FeatureDataGrid (attribute table)
Controls – Implementation

- Implement Silverlight Control or UserControl
- Add to layout
- Define element name (x:Name)
- Make configurable (optional)
  - Implement ISupportsConfiguration
  - In layout, set ElementExtensions.IsConfigurable to true
Demo Controls
Summary

- The Viewer is configurable, but lots for devs, too
  - Devs are the bridge from generic to specific

- Many ways to make the Viewer do what you need

- Add-Ins are inherently reusable
  - Make them more reusable by implementing configuration

- Easy, flexible, rich development environment
Road Ahead
Road Ahead

• Next release coming soon!
  • Version 3.0
  • Expected Q3

• Improvements
  • Printing using 10.1 print service
  • Search is configurable
    • Specify other locator services
  • Measure tool
  • Bookmarks
Demo
3.0 Enhancements
Road Ahead – for Developers

- Built on version 3.0 of the ArcGIS API for Silverlight
- Leverages Silverlight 5
- New tools are shared-source
  - Measure and Bookmarks already on ArcGIS.com
- Portal property added to MapApplication
Road Ahead – Beyond 3.0

- Add capabilities from the Flex Viewer and ArcGIS Manager Web Mapping Application
  - Query
  - Relationships

- Surface new Server capabilities
  - Editor tracking
  - Ownership-based access

- Develop using the Extensibility API
  - Make it better by using it ourselves
  - Share the source code with you
Related UC sessions

• Creating Web Applications with ArcGIS
  - Fri 9:00 AM    Room 8
Steps to evaluate UC sessions

- My UC Homepage > “Evaluate Sessions”

- Choose session from planner
  OR

- Search for session

www.esri.com/ucsessionssurveys
• Thank you for attending
• Have fun at UC2012
• Open for Questions

• Please fill out the evaluation:

  www.esri.com/ucsessionssurveys

  Offering ID: 664