Developing Java Applications with ArcGIS Engine

Eric Bader and Steve Rozic
Introductions

• Who are we?
  – Eric Bader: Product Manager – Java Products
  – Steve Rozic: Instructor – Java, ArcObjects

• Who are you?
  – ArcGIS Desktop Developers?
  – Current ArcGIS Engine Developers?
  – Target Platforms?
  – Preferred IDE?
  – Beginner/Intermediate/Advanced Java Developers?
Schedule

• Today we will cover
  – Introduction to ArcGIS Engine
  – Initialization modes
  – Customization
  – ArcObjects
  – Geoprocessing
  – Conclusion

• 10-15 minutes for questions at the end of the session

Please complete the session survey!
What is ArcGIS Engine?

- Software developer kit for developing cross-platform desktop GIS applications
  - Create stand-alone GIS applications
  - Embed GIS functions
  - Mapping and visualization
  - Data management
  - Queries
  - Editing
  - Analysis

- Engine Includes:
  - Visual JavaBeans & Proxy Classes
  - ArcObjects

Developer Kit for developing cross-platform desktop GIS applications...
High-Level Architecture

Java communicates with ArcObjects through a Java-COM Interop...
ArcGIS Visual JavaBeans

- Visual components to build GUIs
  - MapBean, ToolbarBean, TOCBean, etc...

- What are they really?
  - ArcGIS ActiveX Controls
  - Exposed as AWT Controls by a Java-COM interop
  - Wrapped in Swing JavaBeans for IDE
What is ArcObjects?

- Core GIS Components
  - Written in C++ following COM technology

- Includes fine-grained and coarse-grained objects

- Can be accessed by:
  - Java
  - .NET
  - C++

- Used to create custom applications
  - Desktop
  - Web

ArcObjects are core GIS components written in C++...
Demonstration #1

• Use Case: Where do I get started when using the ArcGIS Engine product to develop a desktop GIS application?

• Motivation: Getting new developers started

• Solution:
  – Examine SDK help
  – Show samples dialog
  – Create basic template application
  – Create a new ArcGIS Engine application
  – Resource Center
Summary

Demonstration #1

• SDK includes:
  – Eclipse IDE plug-ins (Code snippets, Templates, Samples, etc.)
  – Developer help and doc

• Creating an Engine application
  – Step 1: Design UI
  – Step 2: Initialize
  – Step 3: Licensing
  – Step 4: Display GUI
  – Step 5: Shutdown

• Resource center for Java ArcGIS Engine
  – Blog, code gallery, forums, knowledge base, web help and doc

*It is easy to build ArcGIS Engine applications...*
Step 2: Initialize

ArcObjects must be initialized before they can be used

- Initialize ArcObjects for usage in ArcGIS Engine app.

  1. `initializeVisualBeans()`: optimal for using Visual JavaBeans

```java
public static void main(String args[]){
    EngineInitializer.initializeVisualBeans(); //1st line of code
}
```

  2. `initializeEngine()`: optimal for console applications

```java
public static void main(String args[]){
    EngineInitializer.initializeEngine(); //1st line of code
}
```

<table>
<thead>
<tr>
<th>GUI Applications</th>
<th>Console Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>initializeVisualBeans()</code></td>
<td><code>initializeEngine()</code></td>
</tr>
</tbody>
</table>

See “Understanding the ArcGIS Engine Java Interop” for a deeper explanation...
Demonstration #2

- **Use Case:** I need a custom command that allows my end user to export to a "pdf" document?

- **Motivation:** Customize ArcGIS Engine

- **Solution:**
  - One can implement the ICommand interface
  - **OR** extend the BaseCommand class (implements ICommand)
  - Override two methods:
    - `onCreate()` and `onClick()`
  - Use HookHelper to help you write custom commands
  - Call the `addItem()` on the ToolbarBean to add the command
Summary

Demonstration #2

- Create your own tools
  - Implement ICommand & ITool or extend BaseTool class

- Create your own custom menus
  - Implement IMenuDef or instantiate a ToolbarMenu
  - 9 standard menus out of the box
    (e.g. ControlsMapViewMenu)

- Create your own custom palette
  - Implement IPaletteDef or instantiate ToolbarPalette
  - 3 standard palettes out-of-the-box
    (e.g. ControlsInkHighlightPalette)

*There are many ways to help you customize your Engine applications...*
Demonstration #3

• Use Case: I would like to add data to my application. How do I access a Feature Class that is on my local disk?

• Motivation: Define workflows and examine casting

• Solution:
  – Present “Accessing data” workflow
  – Examine Java-style casting and special cases
  – Identify casting guideline when working with ArcObjects
Summary

Demonstration #3

- **Workflow: Access existing data**
  1. Instantiate `WorkspaceFactory` class
     ```java
     SdeWorkspaceFactory sde = new SdeWorkspaceFactory();
     ```
  2. Open existing `Workspace`
     ```java
     Workspace ws;  // Specify Connection Properties
     ws = new Workspace(sde.open(connProp, hWnd));
     ```
  3. Open the Dataset
     ```java
     FeatureClass fClass;
     fClass = new FeatureClass(ws.openFeatureClass("FClassName"));
     ```

*You can follow this workflow for accessing existing data and adding it to a map...*
Summary

Demonstration #3

• Workflow: Access existing data

  4. Instantiate Layer and set properties

```java
FeatureLayer fLayer = new FeatureLayer();
fLayer.setName("LayerName");
```

  5. Associate Layer with Data Source

```java
fLayer.setFeatureClassByRef(fClass);
```

  6. Add the Layer to the Map

```java
//Specify layer and index position (0 - top of the list)
mapBean.addLayer(fLayer, 0);
```

You can follow this workflow for accessing existing data and adding it to a map...
Demonstration #3

• Java-style casting & instanceof support
  – Is available for 90% of ArcObjects

```java
ILayer layer = mapDocument.getLayer(0, 1); //Get the second layer
FeatureLayer featureLayer;
if(layer instanceof FeatureLayer)
    featureLayer = (FeatureLayer) layer;
```

– However, return values of certain methods cannot be cast to a particular category of types (the other 10%)

```java
//Single argument constructor for Proxy class is not deprecated
Workspace ws = (Workspace) fGDBWF.openFromFile("PathFile", 0);

//This is correct way
Workspace ws = new Workspace(fGDBWF.openFromFile("PathFile", 0));
```

*Instantiate the object instead of casting when ClassCastException is thrown...*
Java-COM Interop

• Provides a Java-based API for working with ArcObjects
  – Java classes for every ArcObjects class (e.g. Feature class)
  – Java interfaces for every ArcObjects interface (e.g. IFeature interface)
  – Java proxy class for every interface and implements it (e.g. IFeatureProxy class implements IFeature interface)

• Dealing with ClassCastException and Interfaces

```java
//Single argument constructor for Proxy is not deprecated
IGeoDataset gds = (IGeoDataset) ws.openFeatureClass("FC Name");

//This is correct
IGeoDataset gds = new IGeoDatasetProxy(ws.openFeatureClass("FC Name"));
```

*Interface Proxy Classes allow us to get around the ClassCastException...*
Demonstration #4

- **Use Case:** How do I create standard annotation in a Geodatabase?
  - Annotation allows each piece of ‘text’ to store its own position, text string, and display properties

- **Motivation:** Simplify your coding effort with coarse-grained ArcObjects.
  - Dealing with *AutomationException*

- **Solution:**
  - Present a coarse-grained ArcObject that solves this problem
    - *ConvertLabelsToAnnotation*
  - Examine the *AutomationException* error
  - Examine the state of ArcObjects using Eclipse’s debug mode
Coarse-grained ArcObjects simplify the fine-grained details:

- E.g. 95 lines of code versus only 4 lines
- Coarse-grained objects can be identified in the Javadoc
- Utilize these objects whenever possible!

Use Coarse-grained ArcObjects whenever possible to minimize your coding effort...
Summary

Demonstration #4

• ArcObjects exceptions
  – Java has no data types for exceptions in ArcObjects
  – Errors reported as hexadecimal number – HRESULT

• Java-COM Interop
  – Warps HRESULT in AutomationException class
  – Retrieves description when available
  – When description not available...

AutomationException: 0x80040005 – Unspecified error

  at com.esri.arcgis.interop.NativeObjRef.nativeVtblInvoke(Native Method)
  at com.esri.arcgis.interop.NativeObjRef.a(Unknown Source)
  at com.esri.arcgis.interop.Dispatch.vtblInvoke(Unknown Source)

All ArcObjects code in Java must be surrounded by Try/Catch blocks...
Summary

Demonstration #4

- Eclipse’s debug mode to help trace the state of ArcObjects
  - Instead of using
    `System.out.println();`

- Help you trace your code

- Select “Show Logical Structure” to present more comprehensible information

*Eclipse’s debug mode can help you figure out the state of your ArcObjects...*
Demonstration #5

- **Use Case:** I would like to narrow down the number of houses I need to visit as I try to purchase a home.

- **Motivation:** Leveraging geoprocessing power in your Java programs

- **Solution:**
  - Utilize Geoprocessor Tool Code Generator plug-in
  - Design a custom Geoprocessing Model
  - Use custom model in Java to run analysis
Summary

**Demonstration #5**

- **Author custom models, scripts with Desktop software**
  - Use Geoprocessing tool code generator to generate Java wrapper
  - Invoke the custom tool using the `execute` method

- **Every standard geoprocessing tool has a wrapper class**
  - Wrappers know about tool parameters
  - Constructors accept all mandatory parameters

```java
import com.esri.arcgis.geoprocessing.tools.*;
//Pass in Strings, ArcObjects, or results from GPUtilities
Clip clipTool = new Clip(input, clipper, output);
GeoProcessor gp = new GeoProcessor();
gp.execute(clipTool, null);
```

- **GeoProcessor class only supports one instance of the object (singleton)**

*GeoProcessor knows about all the system tools, but requires an appropriate license ...*
Summary
Demonstration #5

• Wrapper: Java class that represents tool
  – Every standard geoprocessing tool has wrapper class
  – Wrappers know about all parameters for tool
  – Constructors accept all mandatory tool parameters

• Using tool wrappers reduces code
  – Import toolbox alias to distinguish tools you wish to execute
  – Wrappers know about tool parameters

//Import declaration
import com.esri.arcgis.geoprocessing.tools.analysisTools.*;

//Pass in feature classes
Clip clipTool = new Clip(input, clipper, output);
gp.execute(clipTool, null);

Whenever possible, utilize Geoprocessing to simplify your code ...
Summary

Demonstration #5

- Cleaner.release() explicitly releases ArcObjects references

- Java-COM Interop performs garbage collection most times
  - Relies on JVM’s garbage collector (non-deterministic nature)

//Iterate a set of features, where featureCount is a large number
for(int index = 0; index < featureCount; index++)
{
    Feature feature = (Feature) cursor.nextFeature();
    //Do some work with the feature
    Cleaner.release(feature); //Release the feature ArcObject
}

Use com.esri.arcobjects.system.Cleaner package when working with the Cleaner class...
Other Sessions and Demo Theatre

Developer Summit 2009

• Demo Theatre:
  – Leverage Dynamic Display in ArcGIS Engine Applications
    • Wednesday March 25, 2009, 4-5pm, Oasis 1

• Technical Sessions:
  – Extending ArcGIS with Java
    • Wednesday, March 25, 2009, 1 – 2:15pm, Primrose C/D

  – Building and Extending Tasks for ArcGIS Server Java Web Apps
    • Tuesday, March 24, 2009, 2:45 – 4:00pm, Smoketree A-E

  – Implementing Security for ArcGIS Server Java Solutions
    • Tuesday, March 24, 2009, 4:30 – 5:45pm, Mesquite C
Other Sessions and Demo Theatre

Developer Summit 2009

• Technical Sessions:

– Customizing Editing Workflows with the Java Web ADF
  • Wednesday, March 25, 2009, 10:30 – 11:45am, Mesquite C

– Extending ArcGIS Server with Java
  • Wednesday, March 25, 2009, 2:45 – 4:00pm, Primrose C/D

– Customizing Graphics and MapTips with the Java Web ADF
  • Wednesday, March 25, 2009, 4:30 – 5:45pm, Mesquite C
Want to Learn More?

ESRI Training and Education Resources

• Instructor-Led Training:
  – **Introduction to Programming ArcObjects using the Java Platform**
    • Structure of ArcObjects, utilize SDK resources, develop Desktop apps
    • Utilizes ArcGIS Engine for visualization
    • Describes how ArcObjects can be used to extend Server Applications
  – **Developing Applications with ArcGIS Server using the Java Platform**
    • Utilize the components of the Web ADF framework
    • Observe ArcGIS Server programming rules and development patterns
    • Use ADF controls to develop custom applications
    • Develop custom tasks and add them to applications

• Virtual Campus Training (Self Study):
  – **Building Applications with ArcGIS Server Using the Java Platform**
  – **Implementing Security for ArcGIS Server 9.3 Java Solutions**

[http://www.esri.com/training](http://www.esri.com/training)
• Today we covered
  – Basics of ArcGIS Engine
  – Initialization modes
  – Customization
  – ArcObjects
  – Geoprocessing
Additional Resources

Questions, answers and information...

• Thank you! Questions?
  – Complete the session survey!

• Tech Talk
  – Outside this room right now!

• Meet the Team
  – “6:00 pm at the Conference Party”

• ESRI Resource Centers
  – PPTs, code and video
    resources.esri.com

• Social Networking
  – www.twitter.com/ESRIDevSummit
  – tinyurl.com/ESRIDevSummitFB