Extending ArcGIS Server with Java

Eric Bader
Dan O'Neill
Ranjit Iyer

Introductions

Please!
Turn OFF cell phones and paging devices

- **75 minute session**
  - **60 – 65 minute lecture**
  - **10 – 15 minutes Q & A following the lecture**

- **Who are we?**
  - **Dan O'Neill - Lead SDK Engineer, ArcGIS Server for Java**
  - **Eric Bader – Product Manager, Java/Unix**
  - **Ranjit Iyer – Lead Developer, ArcGIS Engine**

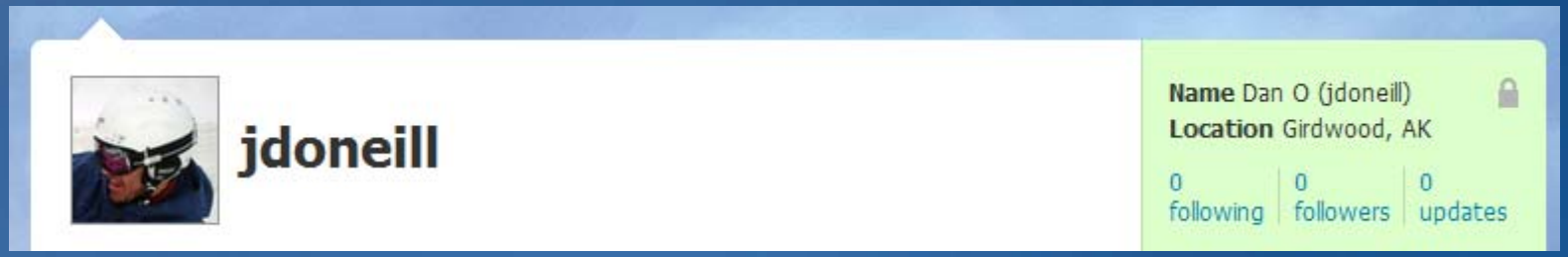
- **Who are you?**
  - **Current ArcGIS Server Application Developers?**
  - **New ArcGIS Server Application Developers?**
  - **Experience in Programming with Java ?**
  - **Knowledge of Web Technologies?**
  - **Experience with the Java WebADF?**

Please complete the session survey!

Let's get connected

- Twitter
  - What is it?
  - Why use it?
  - How can it help us?
  - http://twitter.com/jdoneill

@jdoneill



Session Agenda

- Introduction
  - Customizing the GIS Server in Java
  - ArcGIS Server Architecture
  - Developing a GIS Server extension
  - Extending ArcGIS Server SDK
  - Hello World Demo using Eclipse
- Utility Objects
  - What are Utility Objects?
  - When are Utility Objects used?
  - Developing a Utility Object
  - Consuming a Utility Object
  - Demo
- Server Object Extensions
  - What are SOE's
  - SOE Architecture & Workflow
  - Developing SOE's
  - Deploying & Managing SOE's
  - Consuming SOE's
  - Demo

Please!

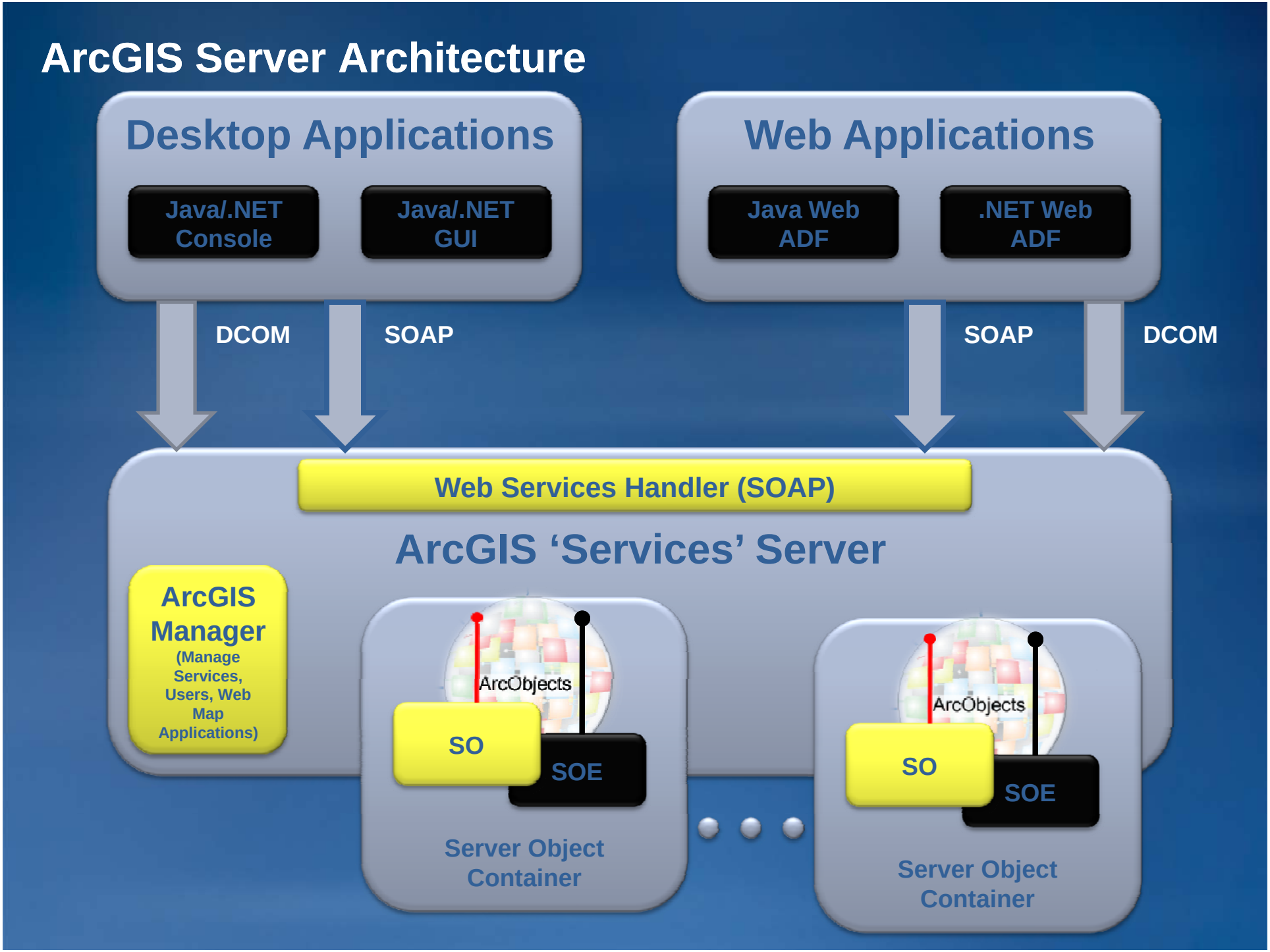
Turn OFF cell phones and paging devices

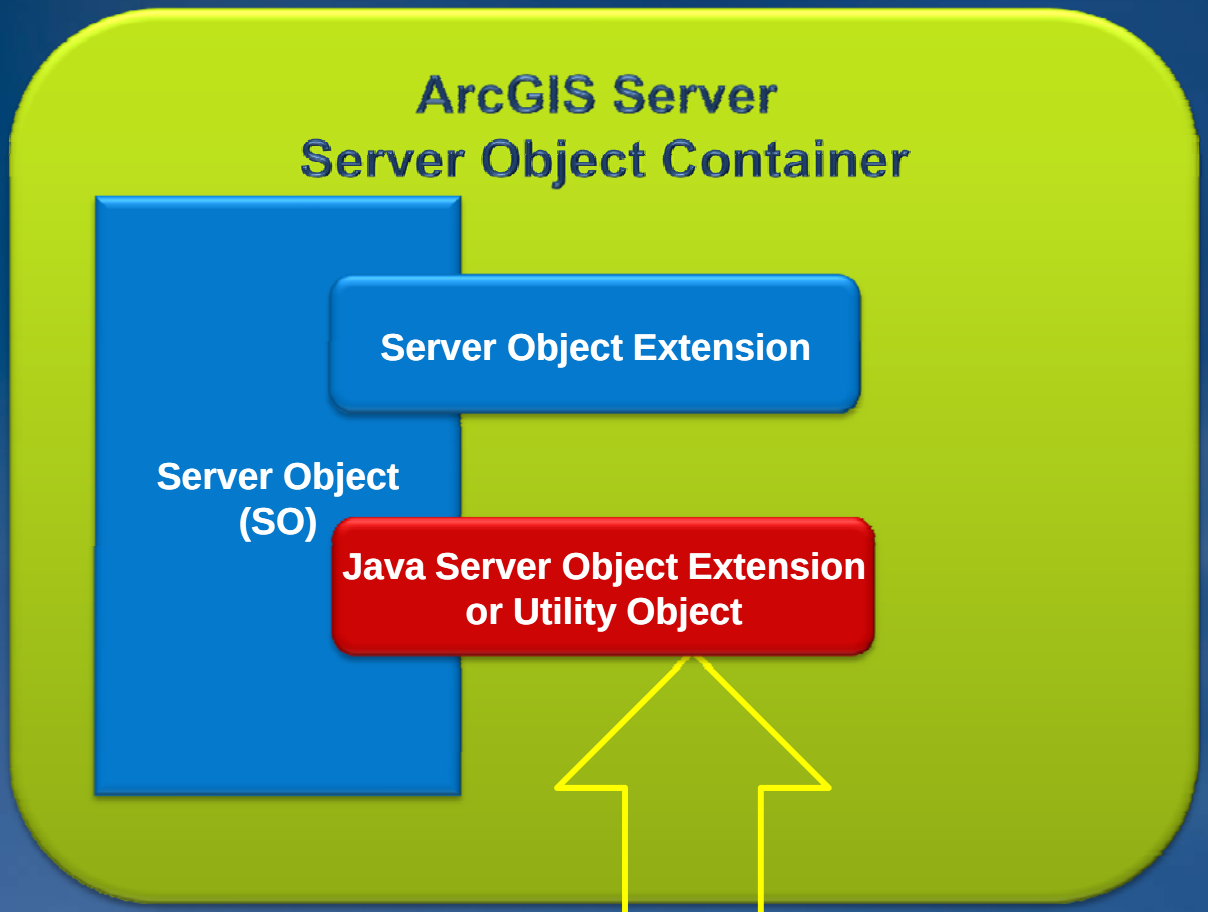
Please complete the session survey!

Customizing the GIS Server in Java

- Write extensions to the GIS Server using Java.
- Benefits of Java
  - Extensions become cross platform
  - Write once run anywhere
- Run the extension on other applications (code reuse)
  - ArcGIS Engine, ArcGIS Desktop
- Extend ArcGIS Server on UNIX
- Java programmers don’t need to learn COM!

ArcGIS Server Architecture





Server Object Extensions

Utility Objects

Custom Geoprocessing Tool

Custom Feature Renderer

Class Extensions

Plug-in Data Sources

Extending ArcGIS Server with Java

Developing a GIS Server extension

- **Two ways to model an extension**
  1. **Utility Object**
     - Pros
       - Not tied to the Server Object
       - Can be run unmodified in Engine
     - Cons
       - Creation overhead could be prohibitive
       - Cannot cache information
       - Needs to be explicitly created

Developing a GIS Server extension

2. Server Object Extension (SOE)

- Special type of “wrapper” extension that encapsulates the business logic (custom renderer, class extensions, etc)
- Uniform way for clients to consume and configure extension
  - Pros
    - » Need not be explicitly created
    - » Could cache information
    - » Could be locally/remotely administered
    - » Could avail of the GIS container’s services (life-cycle notifications, logging framework, access to configuration store)
  - Cons
    - » Tied to a particular Server Object type or configuration

Extending ArcGIS Server for Java SDK

- Developer Friendly
  - Contains a rich SDK experience
    - Documentation
- Authored ~50 new topics for extending ArcGIS Engine and Server.
- 16 new samples have been added to the SDK, 5 of them specifically for SOE's.
    - IDE Integration
      - Eclipse
- Support for Eclipse Ganymede (3.4.x).
- SOE Manager SDK tool included for developers automate the SOE deployment/removal process.

What are Utility Objects?

- Utility Object is a Java extension that encapsulates GIS functionality
- Utility Objects are used to improve performance of ArcGIS
- Extending ArcGIS Server with utility objects allows you, as a developer, to easily share your components across the ArcGIS developer products, i.e. ArcGIS Se
- A major advantage of Utility Objects is that it can be used to alleviate performance problems caused by the overhead of making fine-grained ArcObjects calls across threadsrver and ArcGIS Engine

When are Utility Objects used?

- One of the uses of Utility Objects is to solve a particular class of performance problems that could affect ArcGIS Server and User Interface-based ArcGIS Engine applications.
- Utility objects provide a logical solution to such problems by preventing fine-grained calls across contexts, ensuring that these calls execute within the GIS Server process (as opposed to the Web Server) in ArcGIS Server applications and within the ArcObjects thread (as opposed to Java threads) in UI-based Engine applications.

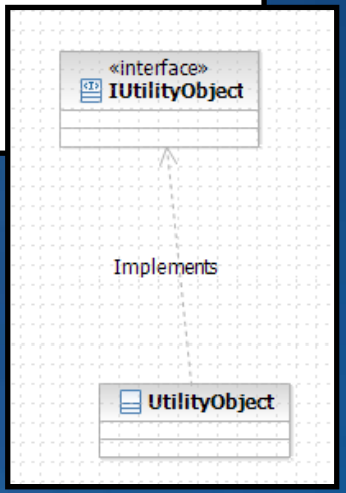
Developing a Utility Object

- ***Step 1: Define a custom interface(s)***
  - An implementer of the Utility Object should define a custom interface that describes an API for clients to call methods on it
  - Permitted data types
    - All Java primitive types except primitive characters (“char”).
    - Single-dimensional array of all Java primitive types except primitive characters (“char”)
    - Java String types
    - Single-dimensional array of Java String types
    - ArcObjects Interface types
    - Single-dimensional array of ArcObjects Interface types

Developing a Utility Object

- **Step 2: Define a Java class that implements the custom interface(s)**
  - **For ArcGIS to recognize this class as a Java extension it needs to be decorated with the *@ArcGISExtension* annotation**

```
@ArcGISExtension
// Decorated with annotation public class
public class UtilityObject implements IUtilityObject{
    …
}
```



Developing a Utility Object

- ***Step 3: Deploy the Utility object***
  - Utility Object deployed as Jar file
  - There are two ways of deployment.
    1. Copy the jar file to the ARCGISHOME/java/lib/ext folder.
    2. Explicitly register the jar with ArcGIS

Consuming the Utility Object

- Once deployed, the Utility Object can be consumed from an Engine/Server application

- Engine

```
IUtilityObject uo =
(IUtilityObject)engineContext.createObject(UtilityObject.class);
```

- Server

```
IUtilityObject uo =
(IUtilityObject)serverContext.createObject(UtilityObject.class);
```

Utility Object Demo

- A demo Engine application to illustrate the two approaches

  1. Fine-grained ArcObjects
  2. Coarse-grained Utility Object

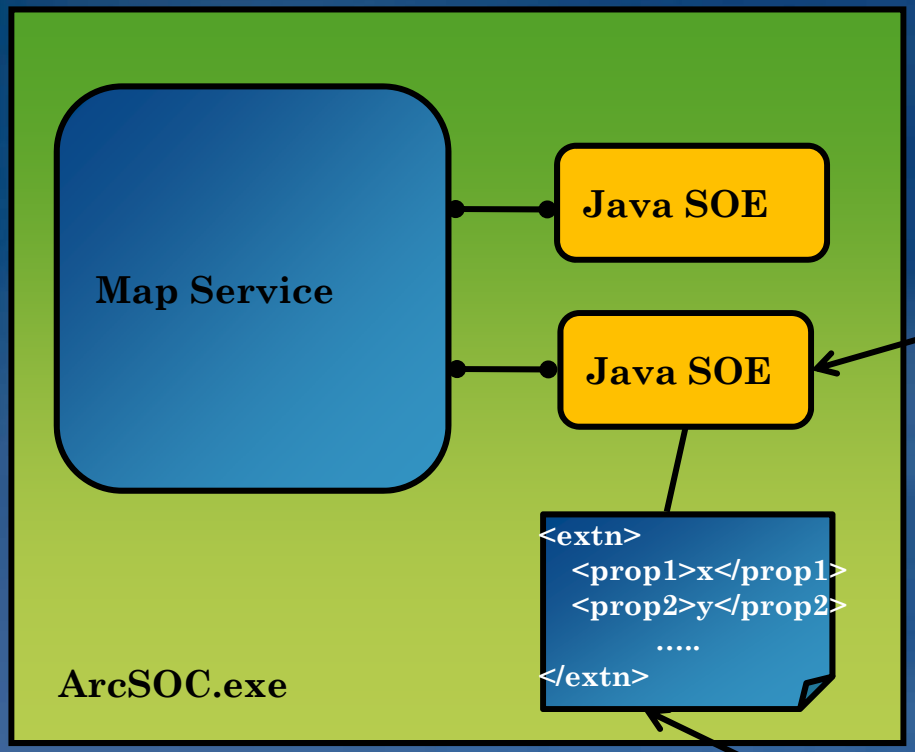
Server Object Extensions

- A Server Object Extension (SOE) extends ArcGIS Server objects. E.g. *MapServer* object.
- SOE's are created and initialized at the time the Server Object instance itself is created and is re-used at the request level.
- Instances of a SOE remain alive as long as the Server Object instance, it's able to cache information that can be re-used from request to request.

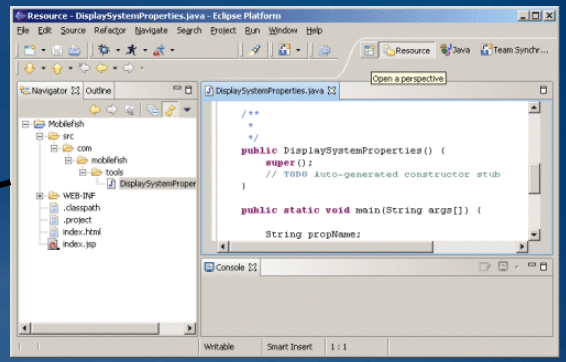
Server Object Extensions

- **SOE Custom Behavior:**
  - **When a *ServerContext* is created and released.**
  - **Can have properties which can be modified to change the SOE's runtime behavior.**
  - **Can have functionality declared in custom interfaces creating new ArcObjects types to be implemented by SOE's.**
  - **Can have 'operations' that can be configured for access based on security rules.**
  - **Can log its operations.**

Server Object Extension Architecture

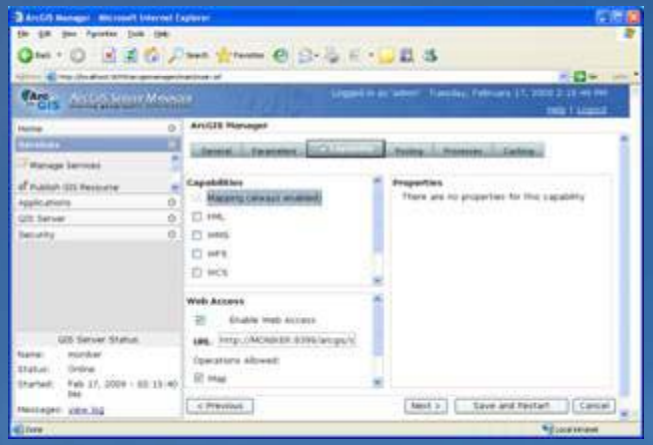


Eclipse

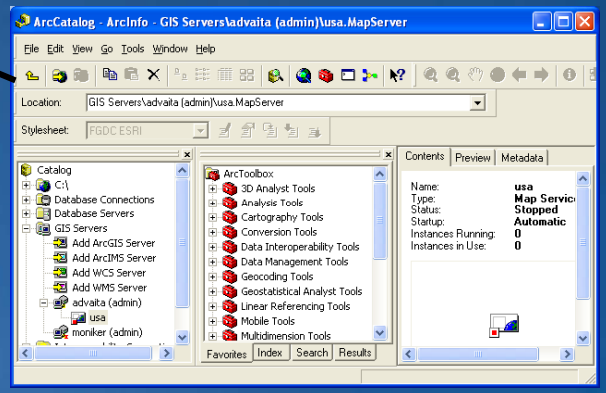


Debug Java SOE

Configure Java SOE



Manager



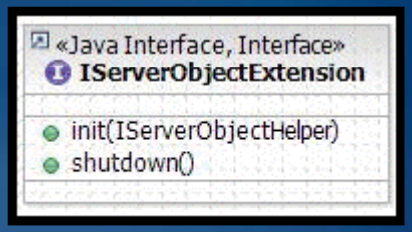
ArcCatalog

SOE Workflow

- Development
- Deployment
- Management
- Consuming

Developing Server Object Extensions

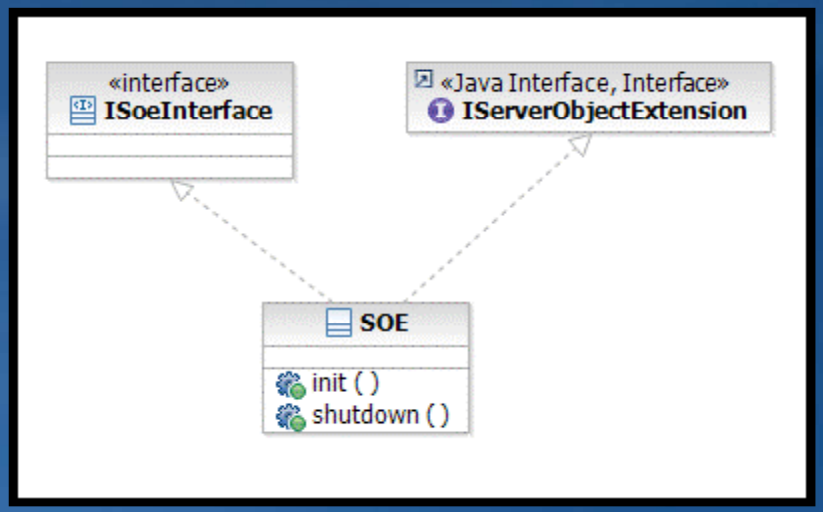
- **Working with SOE Objects**
  - **You can define your own SOE by implementing *com.esri.arcgis.server.IServerObjectExtension*.**



  - **This mandatory interface must be supported by all SOE's, and includes two methods: *init()* and *shutdown()*.**
    - **The *init()* method is called once, when the instance of the SOE is created.**
    - **The *shutdown()* method is called once and informs the SOE that the Server Object's context is being shut down and is about to go away.**
  - **This interface is used by the Server Object to manage the lifetime of the SOE.**

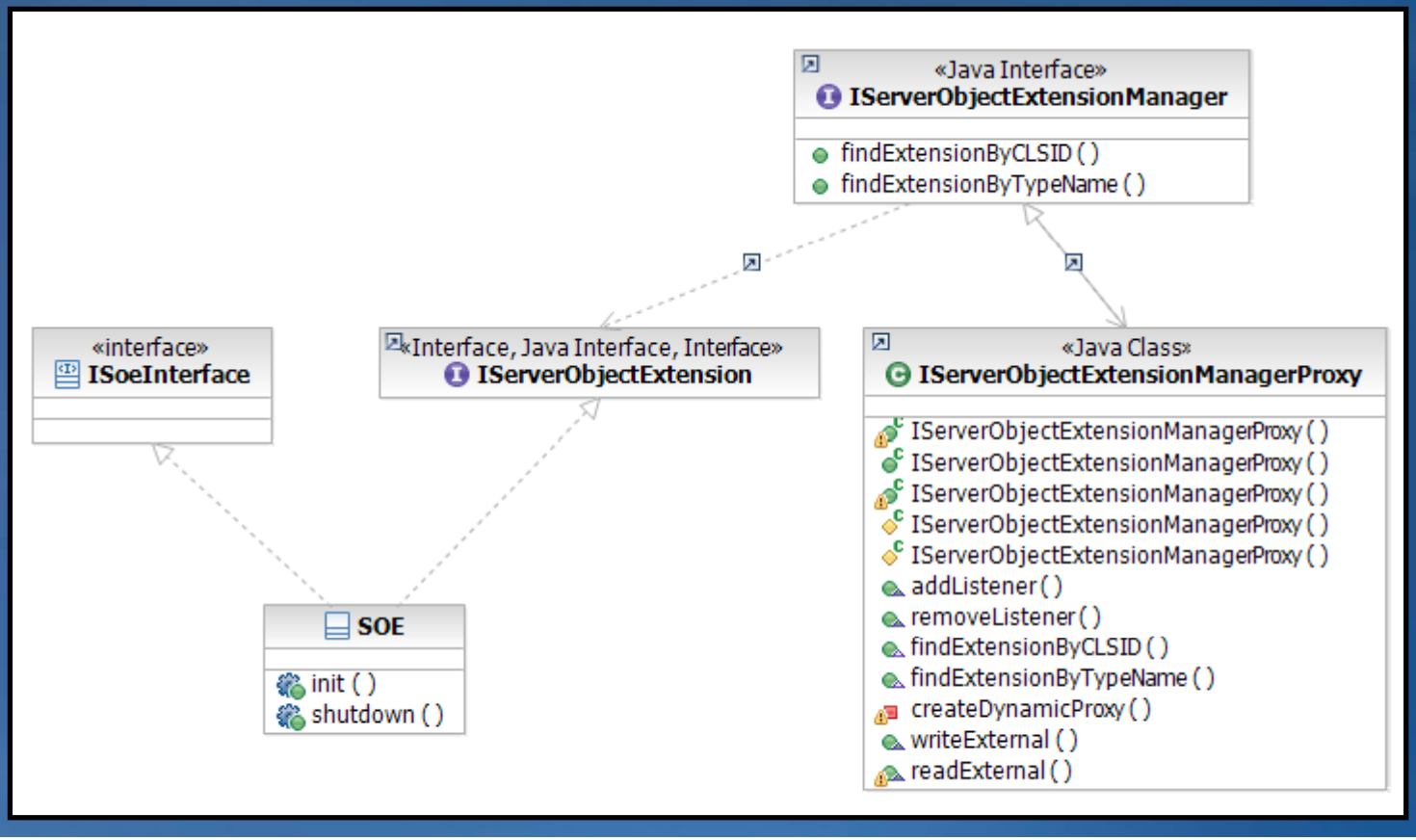
SOE Pattern

- A common pattern for developing SOE's is to create a SOE class which implements the mandatory *IServerObjectExtension* interface and a custom ArcGIS Extension Interface which provides your SOE's custom features.



SOE Pattern

- Your SOE implements specific ArcObject Interfaces to provide specialized functionality and a custom ArcGIS Extension Interface for the business logic.
- This pattern will allow clients to access the methods exposed by your SOE remotely.



@ArcGISExtension **Annotation**

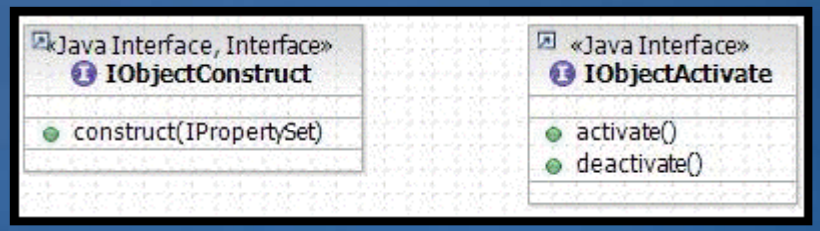
- Auto-Exposure to ArcGIS Platform
- Releases requirement for developers to programmatically configure

```
// Custom Interface
@ArcGISExtension public interface ISoeInterface{
    public void mySoeFoo();
}

// SOE class
@ArcGISExtension public class SOE implements IServerObjectExtension, IMySoeInterface{
    // IServerObjectExtension
    public void init(IServerObjectHelper arg0)throws IOException,
        AutomationException{
        // Called once when the instance of the SOE is created
    }
    public void shutdown()throws IOException, AutomationException{
        // Called once when the SOE's context is shut down
    }
    public void mySoeFoo(){
        // Instantiate custom interface
    }
}
```

Optional Interfaces

- Optional interfaces your Java SOE can implement include
  - » *com.esri.arcgis.system.IObjectConstruct*
  - » *com.esri.arcgis.system.IObjectActivate*
  - » *com.esri.arcgis.systemIRequestHandler*



IObjectConstruct

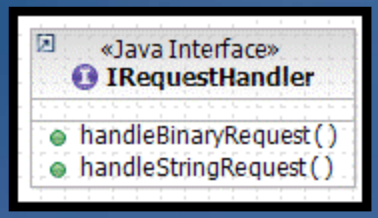
- Additional initialization logic
- Configuration properties
- The interface includes a single method called *construct()*.
  - The *construct()* method is called only once, when the SOE is created and after *IServerObjectExtension.init()* is called.
  - You should include any expensive initialization logic within your implementation of the *construct()* method.

IObjectActivate

- Embed logic each time a server context is acquired or released.
- *IObjectActivate* is an optional interface for SOE's that includes two methods, *activate()* and *deactivate()*.
  - The *activate()* method is called each time a client calls *CreateServerContext()* on the SOE's Server Objects's context
  - The *deactivate() method* is called each time a client releases the context.

IRequestHandler

- Enables web access on your SOE by accepting SOAP requests and responses.
- The interface provides 2 methods:



» *handleBinaryRequest()*: This method handles a binary request.

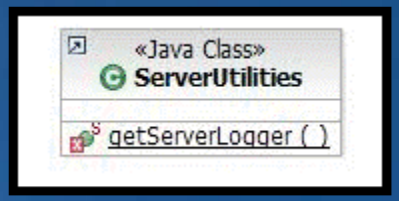
» *handleStringRequest()*: This method handles a String request.

Processing a SOAP Request

1. Parse SOAP request
2. Determine what the client is requesting
3. Create a SOAP response that contains the requested information.

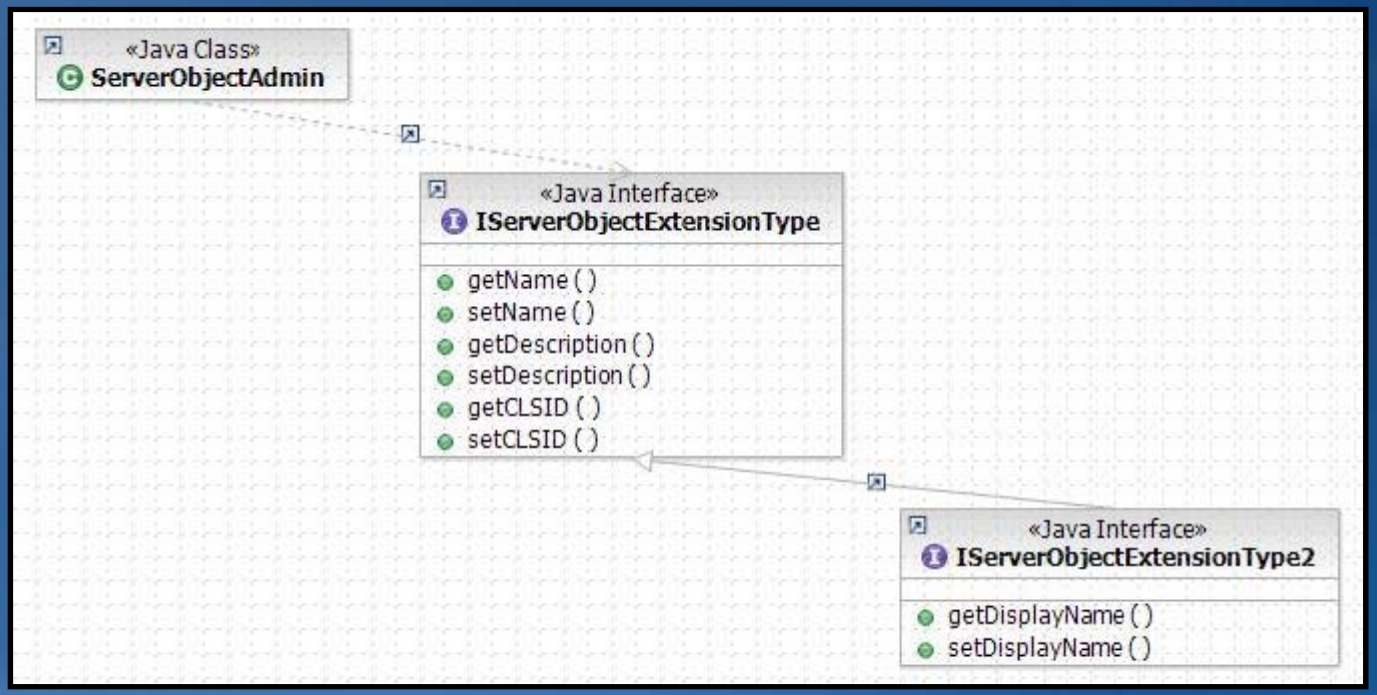
Logging

- All SOE's have the option to log messages to ArcGIS Server's log file.
- *ServerUtilities* is an optional class for SOE's that has a single *getServerLogger()* method which returns a handle to ArcGIS Server's log file via the *ILog2* interface.
- If you want your SOE to log messages to ArcGIS Server's log file, your SOE must write to the Server's log file through the *ILog2* interface returned from this method.



Deploying and Managing Server Object Extensions

1. Deploy SOE to ArcGIS Server
2. Add SOE with Map Server Object
3. SOE Enablement



Consuming Server Object Extensions

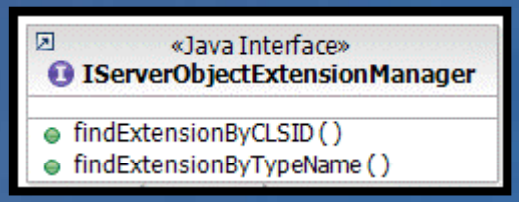
- There are multiple ways to consume SOE's.

» Console Applications

» Web Based Applications

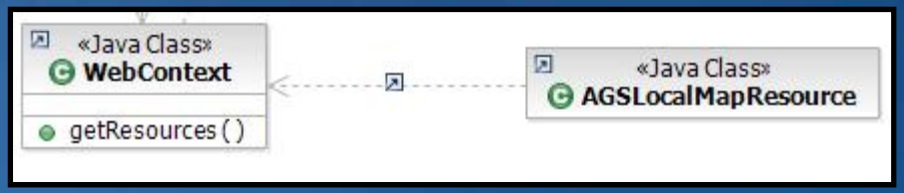
Consume SOE in Console App

1. Obtain the Server Context
2. Access the Server Object through the service context
3. Call the *findExtensionByTypeName(soeName)* on the *MapServer*.



Consume SOE in Java Web ADF

- The WebADF references GISResources, as well as attributes to GIS business objects like *WebMap*, through the *WebContext*.
- Once you have access to the Server Object through the *WebContext* the pattern of obtaining your SOE is the same as in a console application.



SOE Demo

Summary

- Introduction
  - Customizing the GIS Server in Java
  - ArcGIS Server Architecture
  - Developing a GIS Server extension
  - Extending ArcGIS Server SDK
  - Hello World Demo using Eclipse
- Utility Objects
  - What are Utility Objects?
  - When are Utility Objects used?
  - Developing a Utility Object
  - Consuming a Utility Object
  - Demo
- Server Object Extensions
  - What are SOE's
  - SOE Architecture & Workflow
  - Developing SOE's
  - Deploying & Managing SOE's
  - Consuming SOE's
  - Demo

Still have questions?

Additional Resources

Questions, answers and information…

- ***Tech Talk***
  - ***Outside this room right now!***
- ***Meet the Team***
  - ***Java Development team***
    Wednesday 6 – 7pm Oasis 2
- ***ESRI Resource Centers***
  - **PPTs, code and video**

resources.esri.com

- ***Social Networking***

www.twitter.com/ESRIDevSummit

tinyurl.com/ESRIDevSummitFB

Other Sessions

Building and Extending Tasks for ArcGIS Server Java Web Applications

Tuesday, March 24, 2009, 2:45pm - 4:00pm, Smoketree A – E

Customizing Editing Workflows with the Java Web ADF

Wednesday, 10:30am - 11:45am, Mesquite C

Extending ArcGIS with Java

Wednesday, 1:00pm - 2:15pm, Primrose C/D

Want to Learn More?

ESRI Training and Education Resources

- **Instructor-Led Training**
  - **Developing Applications with ArcGIS Server Using the Java Platform**
- **Free Web Training Seminar**
  - **Building Applications with ArcGIS Server Using the Java Platform**

http://www.esri.com/training

Thank You