Building Java Applications Using the ArcGIS Server Web ADF and AJAX

Antony Jayaprakash
Jayant Sai
Schedule

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

• Cell phones and pagers

• Please complete the session survey – we take your feedback very seriously!
Introductions

- Who are we?
  - Members of ArcGIS Server Java Team

- Who are you?
  - ArcGIS Server Java Edition developers

- Prerequisites
  - Java
  - Working knowledge of
    - JSF
    - JavaScript
    - XML
    - Ajax
Session agenda

• Ajax integration in the Java Web ADF
  – Web ADF controls
  – JavaScript library
  – Serverside objects

• Common use cases
  – Listen to control updates
  – Update controls on client
  – Updating custom components
  – Timer based requests

• Web Mapping Application (WMA) Ajax

• Tips & Tricks

• Q&A
Terminology

• Phase Listener
  – A class registered in the JSF lifecycle to be notified at the beginning and ending of a specific JSF phase

• JSF Components & Web ADF Controls
  – They are same, controls & components used interchangeably

• Postback
  – Action by webpage, to submit page to server for processing

• Clientside/Partial PostBack
  – Making client side script calls (using XMLHttpRequest) to server side code without causing a postback
Web ADF controls

• All Web ADF controls are Ajax enabled

```xml
<%@ taglib uri="http://www.esri.com/adf/web" prefix="a" %>

<a:map ... />
<a:toc ... />
<a:overview ... />
<a:toolbar ...>
  <a:tool ... />
</a:toolbar>
<a:task ... />
<a:result ... />
<a:button ... />
```
JavaScript library

• Extensive clientside JavaScript library
  – Handles user interaction & communication with server
  – Framework for Ajax
  – Browser independent functions
  – Object oriented library
  – All object names prefixed with ‘Esri’
  – Only 1 global, EsriControls
  – Does not conflict with 3rd party Ajax toolkits (prototype, dojo, DWR, MS Ajax, etc)
Serverside objects

- Java object to support Ajax
  - JSF Phase Listeners
  - XML Ajax Renderers
    - MapRenderer
    - TocRenderer
    - ToolbarRenderer
    - OverviewRenderer
    - TaskRenderer
  - AJAXResponseRenderer

- Custom renderer hook-ins
  - AJAXRenderer interface
  - WEB-INF/ajax-renderers.xml
Common use cases

- **#1: Listen to control updates** to perform custom operations
- **#2: Update controls on client** based on server operation
- **#3: Update custom components** by integrating into request-response cycle
- **#4: Timer based requests** to display data on client
#1: Listen to control updates

- **When to Use?**
  - Perform asynchronous updates based on updates to controls
  - Request-response should not interfere with user interaction

- **Usage in Web ADF?**
  - Retrieving images for continuous pan

- **Demo**
  - On map update
  - Request map extent and scale
  - Display extent and scale for current extent
#1: Application flow

**Register for updates**

**On update, request data**

**Process request**

**Create response**

**Render response**

**Update HTML elements**

Legend

| Client operation | Server operation |

- Register for updates
- On update, request data
- Process request
- Create response
- Render response
- Update HTML elements
- EsriMap.addUpdateListener()
- EsriUtils.sendAjaxRequest()
- XMLUtil.newDocument()
- AJAXUtil.writeResponse()
- myElement.value = valueFromXml

Legend:

- Client operation
- Server operation
#1: Demo details

- **JavaScript** *(mapInformation.js)*
  - Register for map updates
  - Send request for data *(updateMapInformationRequest())*

- **Implement PhaseListener** *(MapInformationPhaseListener.java)*
  - Process request *(afterPhase())*
  - Get extent
  - Render & return XML response

- **Register PhaseListener** *(fc-ajaxdemo.xml)*
  - Add ‘<lifecycle>’ declaration

- **JavaScript** *(mapInformation.js)*
  - Process response to update HTML *(updateMapInformationResponse())*
#2: Update controls on client

- **When to Use?**
  - Update controls based on changes to GIS business objects

- **Usage in Web ADF?**
  - Update on zoom level change

- **Demo**
  - User zoom in/out of map using drop down
  - Update map’s extent based on zoom factor
  - Use Web ADF Ajax classes to update controls on client
#2: Application flow

**Legend**

<table>
<thead>
<tr>
<th>Client operation</th>
<th>Server operation</th>
</tr>
</thead>
</table>

- **Send request with update parameters**
  - Process request
  - Create renderer
  - Update controls
  - Get Ajax response
  - Update controls

- **EsriUtils.sendAjaxRequest()**
  - request
  - PhaseListener
  - new AJAXResponseRenderer()
  - webContext.refresh()
  - AJAXResponseRenderer.renderResponse()
  - response
  - EsriControls.processPostBack
#2: Demo details

- **JavaScript** *(mapzoom.js)*
  - Send ajax request with zoom factor *(sendMapZoomRequest()*)
  - Set callback function *(EsriControls.processPostBack)*
- **Implement PhaseListener** *(MapZoomPhaseListener.java)*
  - Process request *(afterPhase()*)
  - Create AJAXResponseRenderer
  - Update map & context
  - Render & return XML response
#3: Updating custom components

• **When to Use?**
  - Update custom components
  - Done as part of request-response cycle

• **Usage in Web ADF?**
  - All web control updates

• **Demo**
  - Render XML with attributes of features in visible extent
  - Update table to display attributes data
#3: Application flow

**Legend**
- **Client operation**
- **Server operation**

1. User request
2. AJAXRenderer created
3. Control to update
4. State before update
5. Render XML
6. Get Ajax response
7. Update HTML element
8. EsriUtils.sendAjaxRequest()
9. AJAXRenderer
10. AJAXRenderer.getControlClass()
11. AJAXRenderer.getOriginalState()
12. AJAXRenderer.renderAjaxResponse()
13. EsriControls.processPostBack
14. myHandler(xml)
15. callscalls
#3: Demo details

- **Implement AJAXRenderer** *(LayerAttributesRenderer.java)*
  - Check if updates required
  - Render XML tag with data
- **Register AJAXRenderer** *(WEB-INF/ajax-renderers.xml)*
  - Add to list of AJAXRenderers
- **JavaScript** *(layerAttributes.js)*
  - Function to parse XML and update HTML elements *(layerAttributesHandler())*
- **Register XML tag handler**
  - Add function to list of xml tag handlers to EsriControls JavaScript global object *(EsriControls.addPostBackTagHandler())*
#4: Timer based updates

- **When to Use?**
  - Execute operation periodically
  - Display temporal data

- **Usage in Web ADF?**
  - Checking GP task results status

- **Demo**
  - Periodic request to get and display location of fleet
  - Start/Stop timer
#4: Application flow

**Request data**
- Process request
- Create response
- Render response
- Update

**EsriUtils.sendAjaxRequest()**
- request
  - PhaseListener
  - XMLUtil.newDocument()
  - AJAXUtil.writeResponse()
  - map.graphics.drawImage()
Web Mapping Application (WMA) Ajax

• WMA code available as part of ArcGIS Server Java install
  – `<ArcGIS>/java/web/web_map_application/`

• Easy Ajax integration for WMA

• Targeted to developers extending WMA

• Server side code
  – Write custom class implementing AjaxCommand interface
  – Register class in faces-config.xml

• Client side code
  – HTML to display output
  – JavaScript to process server response
#4 Demo Details

- **Implement AjaxCommand Interface** *(TrackingCommand.java)*
  - Get points *(handleAjaxRequest())*
  - Return in XML

- **Register TrackingCommand** *(WEB-INF/fc-ajaxdemo.xml)*

- **JavaScript** *(tracking.js)*
  - Function to send request *(trackingRequest())*
  - Function to process and add markers to map *(trackingResponse())*

- **Register managed bean data**
  - Tracking managed bean *(WEB-INF/fc-ajaxdemo.xml)*
  - Register managed bean in Web Context *(WEB-INF/faces-config.xml)*
Tips & Tricks

**EsriUtils.**
- createXmlHttpObject()
- buildRequestParams()
- submitForm()
- sendAjaxRequest()
- getXmlDocument()
- stringToXml()
- getErrorFromDocument()

**EsriControls.**
- `<control>Ids/<control>s Arrays
- processPostBack()
- processPostBackXML()`
Tips & Tricks (contd.)

• Issues with Clientside PostBack (edit jsp file)

```javascript
<script type="text/javascript">
    EsriUtils.doPostBack = false;
</script>
```

• To see XML output (edit web.xml)

```xml
<context-param>
    <param-name>com.esri.adf.LOG_LEVEL</param-name>
    <param-value>CONFIG</param-value>
</context-param>
```

• Use AJAXUtil.java to write response or generate error message to be handled on client

• Render all responses within ‘<response>’ tag
Tips & Tricks (contd.)

• Optimize and remove unused objects
  – Managed beans & phase listeners (faces-config.xml)
  – ajaxGpAsyncTaskResultsRenderer (ajax-renderers.xml)
  – EsriTask_GPAsyncTaskResultsTimer JavaScript include & timer initialization (toc.xsl)
  – Remove functionality files (web.xml)
Tips & Tricks (contd.)

• Firefox extensions
  – Firebug : JavaScript debugging, Monitor XMLHttpRequest
  – Web Developer : Page info, DOM info

• IE
  – Menu -> Tools -> Internet Options... -> Advanced :
  – Uncheck ‘Disable Script Debugging (Internet Explorer)’
  – Uncheck ‘Disable Script Debugging (Other)’
  – Check ‘Display a notification about every script error’

• IE7
  – IE Developer Toolbar
Ajax Read Me

In Conclusion

- All sessions are recorded and will be available on EDN
  - Slides and code will also be available

- Please fill out session surveys!

- Still have questions?
  1. Tech talk, Demo Theatres, Meet the Team
  2. “Ask a Developer” link on web page
     - www.esri.com/devsummit/techquestions
Recommended Sessions

• **ArcGIS Server for ArcIMS Developers**
  – Tuesday 1:00 in Primrose A
  – Thursday 1:30 in Catalina/Madera

• **Architecting ArcGIS Server for Linux/Solaris**
  – Wednesday 4:30 in Catalina/Madera

• **Advanced Java Web ADF**
  – Wednesday 1:00 in Smoketree A-E

• **Building Tasks for the Java Web ADF**
  – Tuesday 2:45 in Primrose A
  – Wednesday 4:30 in Primrose C/D

• **Implementing Security for Java**
  – Thursday 10:15 in Pasadena/Sierra/Ventura