IBM Software Group

Integrating Jakarta Commons Logging (JCL) with IBM WebSphere Application Server

Ajay Bhalodia

WebSphere® Support Technical Exchange

ON DEMAND BUSINESS

Agenda

- What is JCL
- Why use JCL
- JCL support in WebSphere Application Server
- Discovery of JCL artifacts
- Building application specific JCL solutions
- Options for packaging JCL artifacts in WebSphere Application Server with detailed steps

What is Jakarta Commons Logging - JCL

- Most applications have a need to log information
- Many logging implementation details are available for use by applications
- JCL stands for Jakarta Commons Logging
- JCL provides a commons logging API that enables applications to be independent of the actual logging implementation

Why use JCL

- JCL provides a simple logging interface
  - enables application logging to be simple and independent of the actual logging system that the application uses
  - it can be used with different logging implementations.
- JCL provides a configurable abstract factory facility
  - LogFactory – which is used to instantiate a specific implementation of the LogFactory class

Overview of steps to use JCL

1. Application configures the LogFactory with concrete LogFactory implementation class that supports particular Log implementation

2. Application invokes method LogFactory.getFactory() to obtain an instance of the log factory – example: myLogFactory

3. Application calls myLogFactory.getLog() to obtain the log implementation

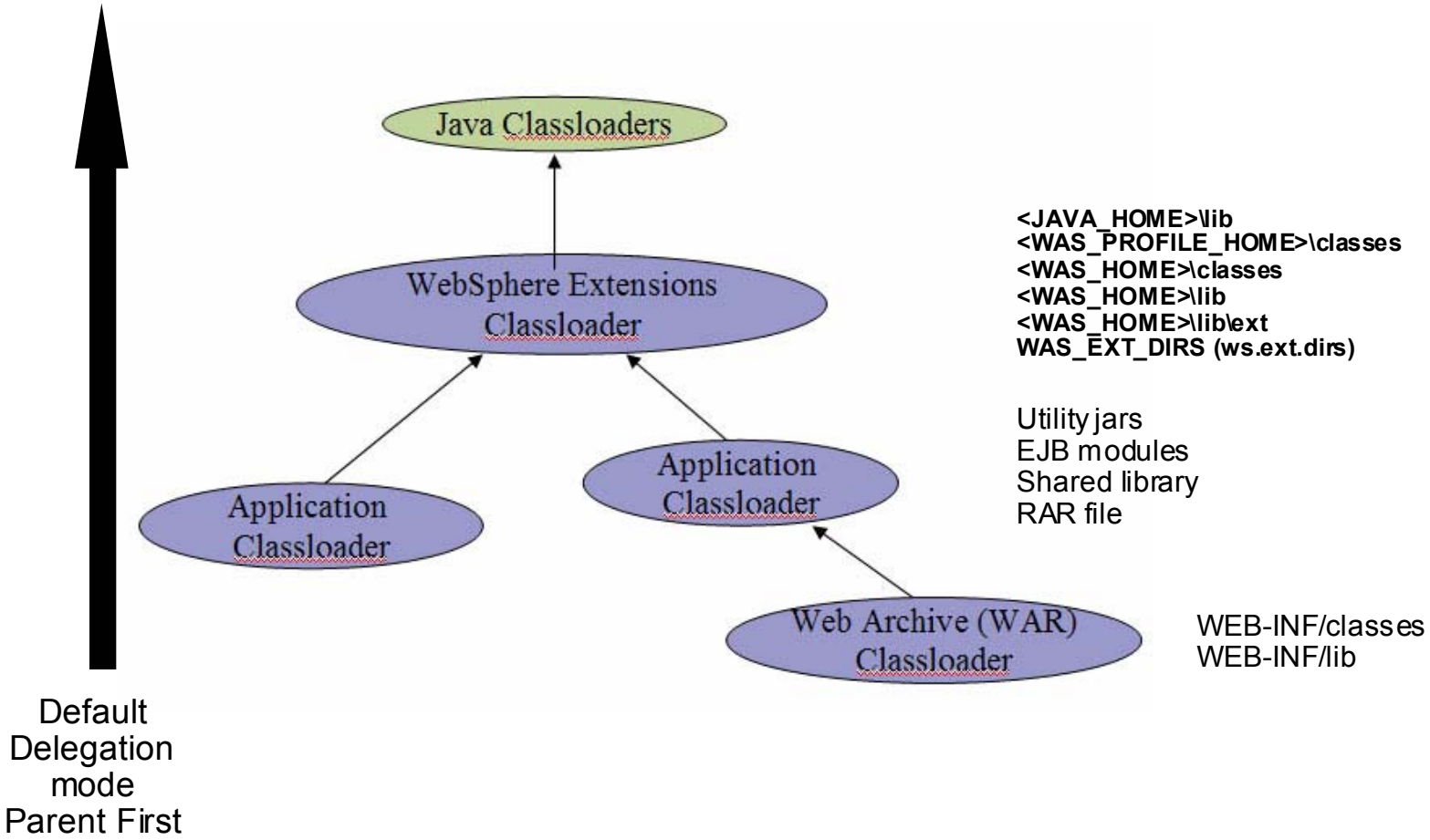
JCL support in WebSphere

- WebSphere is preconfigured to use JCL for its own logging
- Applications can use the JCL artifacts provided by WebSphere or override and use application specific JCL artifacts
- Problems often arise when using application specific JCL artifacts

Discovery options for application specific JCL artifacts

1. Assign the fully-qualified name of the LogFactory implementation class to system property:
   - org.apache.commons.logging.LogFactory
   - The above is **not supported** and will not be discussed
2. Specify the fully-qualified name of the LogFactory implementation class in file:
   - META-INF/services/org.apache.commons.logging.LogFactory
3. Create commons-logging.properties file in classpath:
   - Specify name/value pair with fully-qualified name of the LogFactory implementation class

WebSphere default classloader (review)



Default Delegation mode Parent First

What is the problem for using JCL in WebSphere?

- JCL artifacts provided by WebSphere appear on the WebSphere extensions classloader
  - By default it is the parent of all the application classloaders
- When an application requires JCL artifacts the default delegation mode delegates the load to its parent
  - The process recurses upwards to the root (JAVA classloader) to find the artifacts
  - If the search fails it will unwind to the next lower classloader until the artifacts are found
- In the default classloader scheme (parent first) the JCL artifacts will thus get loaded by WebSphere Extensions classloader
  - That is the problem as the application needs JCL artifacts part of its packaging

Building application specific JCL solutions

1. Specify the desired LogFactory implementation within one of the files below
   - org.apache commons.logging.LogFactory (JCL option 2)
   - commons-logging.properties. (JCL option 3)
   - NOTE: JCL option 1 – using system property is not supported and thus not discussed

2. Package all JCL classes and resources in a utility JAR
   - commons-logging.jar - including the file created in step 1 above

Building application specific JCL solutions (cont'd)

3. Make commons-logging.jar available to the application using one of these techniques:
   - Add commons-logging.jar to the application EAR
   - Add commons-logging.jar to the application WAR
   - Configure commons-logging.jar into an application-associated shared library
   - Configure commons-logging.jar into a server-associated shared library

4. Set the [delegation] mode of the relevant classloader that contains commons-logging.jar in its local classpath to PARENT_LAST

   Example: if placing the commons-logging.jar in the WEB-INF/lib, then need to set PARENT_LAST on the WAR

Preferred way of packaging

- Each of the solutions mentioned above work equally well
  - Instructions for each solution should be followed with care
- It is recommended to package all JCL artifacts (classes and resources) in a single utility jar to keep it simple and avoid confusion
- If possible try to keep the JCL artifacts used by your application independent of JCL artifacts shipped with WebSphere

Choosing where to place JCL artifacts

1. Add application-specific JCL artifacts to EARs (WARs)
   - Provides implicit versioning per application
   - Automatically deployed with the application
   - At runtime they are loaded with the classloader that loads the EAR or WAR

Choosing where to place JCL artifacts (cont'd)

2. Using Shared Libraries
   - Provides a way to associate to a specific application or applications. This allows for versioning maintenance and development
   - If the JCL artifacts change no need to redeploy the entire application as the JCL artifacts are outside the EAR/WAR modules

NOTE: Setting classloader delegation mode of PARENT_LAST is required for both above options

Detailed steps – Packaging JCL artifacts in EAR

- Packaging of JCL artifacts within the application EAR

1. Specify the desired LogFactory implementation within file commons-logging.properties
   - Create a file named “commons-logging.properties
   - Enter the following line into the file

     org.apache.commons.logging.LogFactory=
     <fully_qualified_LogFactory_implementation_class>

     Example: to use the Log4J factory

     org.apache.commons.logging.LogFactory=
     org.apache.commons.logging.impl.Log4jFactory

Detailed steps – Packaging JCL artifacts in EAR (cont'd)

2. Package all JCL classes and resources in a utility JAR
   - Call it commons-logging.jar
   - Partial listing of the commons-logging.jar file:

     META-INF
     META-INF/LICENSE.txt
     META-INF/MANIFEST.MF
     commons-logging.properties
     org
     org/apache
     org/apache/commons
     org/apache/commons/logging
     org/apache/commons/logging/Log.class

Detailed steps – Packaging JCL artifacts in EAR (cont'd)

3. Make commons-logging.jar available to EJB and Web modules of your application
   - Add file commons-logging.jar to the root directory of the application EAR

     Partial listing of application EAR containing JCL artifact

     ```
     META-INF
     META-INF/MANIFEST.MF
     ...
     commons-logging.jar
     ...
     ```
   - Add commons-logging.jar to the Class-Path attribute within the MANIFEST.MF file of every EJB jar and WAR dependent on the JCL solution.

     Example: META-INF/MANIFEST.MF file of every EJB and WAR should contain:

     ```
     Class-Path: commons-logging.jar
     ```

Detailed steps – Packaging JCL artifacts in EAR (cont'd)

4. Set the [delegation] mode of the Application classloader to PARENT_LAST using the admin console:
   - Select Applications > Enterprise Applications > "my application"
   - Select PARENT_LAST from the Classloader Mode drop-down list
   - Click OK
   - Select Save and save your changes
5. Restart the application for the changes to take effect
   - The server configuration changes are now effective, allowing the JCL solution to function as expected

Detailed steps – Using Shared Library

Application-associated shared library

Specifying LogFactory implementation using file:

- org.apache.commons.logging.LogFactory

1. Specify the desired LogFactory implementation
   - Create file org.apache.commons.logging.LogFactory
   - Above file should contain JCL LogFactory implementation class
     Example: org.apache.commons.logging.impl.LogFactoryImpl

Detailed steps – Using Shared Library (cont'd)

2. Package all JCL classes and resources in a utility JAR
   - example: commons-logging.jar
   - Place file org.apache.commons.logging.LogFactory created in step #1 in the commons-logging.jar in path: /META-INF/services
   - Partial listing of the commons-logging.jar should look like:

```
META-INF
        META-INF/LICENSE.txt
        META-INF/MANIFEST.MF
        META-INF/services/ org.apache.commons.logging.LogFactory
        org
        org/apache
        org/apache/commons
        org/apache/commons/logging
        org/apache/commons/logging/Log.class
```

Detailed steps – Using Shared Library (cont'd)

3. Define a shared library using the admin console. Open WebSphere Application console:
   - Click "Environment" > "Shared Libraries" in left pane.
   - Select scope:
     - Cell "myCellName"
     - Node "myNodeName"
     - Server "myServerName"
   - Click "New" and enter the name, description, and classpaths for the shared library
     - Enter "JCL" in the Name field.
     - [Optional] Enter "JCL shared library" in the Description field.
     - Enter "d:\tmp\jcl\commons-logging.jar" in the Classpath field.
   - Click OK and save your changes
   - A shared library "JCL" is now defined

Detailed steps – Using Shared Library (cont'd)

4. Associate the shared library "JCL" to your application. Open WebSphere Application console:
   - Select Applications > Enterprise Applications > "myApplication"
   - Under Additional Properties select Libraries. In the next pane that opens up click on Add
   - In the next pane select "JCL" and click on OK and save your changes
   - A shared library "JCL" is now associated with application "myApplication"

Detailed steps – Using Shared Library (cont'd)

5. Set the Application classloader [delegation] mode to "PARENT_LAST". Using the Admin console:
   - Select Applications > Enterprise Applications > "myApplication".
   - Select PARENT_LAST from the Classloader Mode drop-down list
   - Click OK and save your changes
   - The mode of the Application classloader is now PARENT_LAST.
6. Restart the application for changes to take effect

Summary

- High level points of presentation
  - What is JCL
  - Why use JCL
  - JCL support in WebSphere Application Server
  - Discovery of JCL artifacts
  - Building application specific JCL solutions
  - Options for packaging JCL artifacts in WebSphere with detailed steps

Additional WebSphere Product Resources

- Whitepaper – Integrating Jakarta Commons Logging (this white paper in addition to Jakarta Commons Logging guide was used for preparing this presentation)
  - http://www.ibm.com/support/docview.wss?uid=swg27004610
- WebSphere Application Server Information Center
  - http://www.ibm.com/software/webservers/appserv/was/library/
- Jakarta Commons Logging guide:
  - http://jakarta.apache.org/commons/logging/guide.html#Introduction
  - http://jakarta.apache.org/commons/logging/
- Learn about other upcoming webcasts, conferences and events:
  - http://www.ibm.com/software/websphere/events_1.html

Additional WebSphere Product Resources (cont'd)

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere User Group Community: http://www.websphere.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant: http://ibm.com/software/info/education/assistant
- Learn about the Electronic Service Request (ESR) tool for submitting problems electronically: http://www.ibm.com/software/support/viewlet/probsub/ESR_Overview_viewlet_swf.html
- Sign up to receive weekly technical My support emails: http://www.ibm.com/software/support/einfo.html

Questions and Answers