Configuring application specific logging with Log4j

(Available in Geronimo 2.1.1 and later)

Geronimo uses log4j for logging and the base log4j configuration is in `var/log/server-log4j.properties`. If your application also uses log4j for logging you can configure logging in this file. However, this is not self-contained. Instead, you can configure log4j settings specific to your application in your application plugin.

Note that in any case, unless you use <hidden-classes> or <inverse-classloading> to load your own copy of log4j separate from the geronimo copy, log4j will not automatically read any log4j.properties files you may have included in your classpath.

If you want to use different configuration file for server logging

The configuration file `server-log4j.properties` is hard-coded in `config.ser` and cannot be updated via `config.xml`. However, you can swap the default `server-log4j.properties` file by overriding system property `org.apache.geronimo.log4jservice.configuration` as followed:

```
export GERONIMO_OPTS=-Dorg.apache.geronimo.log4jservice.configuration=$GERONIMO_HOME/var/log/server-log4j.xml
```

If you are building your own application:

This example is taken from the apache directory plugin for geronimo, so the paths etc are adapted for that plugin.

1. Include an ApplicationLog4jConfigurationGBean in your application plan, like this:

```
    <gbean name="DirectoryLog4jConfiguration" class="org.apache.geronimo.system.logging.log4j.
ApplicationLog4jConfigurationGBean">
        <attribute name="log4jFile">var/directory/log4j.properties</attribute>
        <reference name="ServerInfo"><name>ServerInfo</name></reference>
    </gbean>
```

   You can also use a log4j.properties file in your classpath, for example, in `WEB-INF/classes/META-INF`. (If you prefer not to place this file under `WEB-INF/classes/`, the path of this file must be added to your classpath). But this will obstruct anyone trying to configure logging as the file will remain packed in your application somewhere hard to find.

```
    <gbean name="DirectoryLog4jConfiguration" class="org.apache.geronimo.system.logging.log4j.
ApplicationLog4jConfigurationGBean">
        <attribute name="log4jResource">META-INF/log4j.properties</attribute>
    </gbean>
```

2. Configure application specific logging in your log4j.properties file. The gbean prevents you from overriding settings that apply to global logging, but you can configure appenders for specific loggers for your application. Here's the example for ApacheDS:

```
#attach an appender to the base apacheds package logger:
log4j.logger.org.apache.directory=INFO,apacheds
#do not log apacheds to geronimo logs:
log4j.additivity.org.apache.directory=false

#Configure the apacheds specific appender:
log4j.appender.apacheds=org.apache.log4j.DailyRollingFileAppender
log4j.appender.apacheds.File=${org.apache.geronimo.server.dir}/var/log/apacheds.log
log4j.appender.apacheds.layout=org.apache.log4j.PatternLayout
# geronimo style logging
log4j.appender.apacheds.layout.ConversionPattern=%d{ABSOLUTE} %-5p [%c{1}] %m%n

# some settings for the apacheds loggers, taken from the standard apacheds log4j.properties file.
# with these we'll not get innundated when switching to DEBUG
log4j.logger.org.apache.directory.shared.ldap.name=WARN
#log4j.logger.org.springframework=WARN
log4j.logger.org.apache.directory.shared.codec=WARN
log4j.logger.org.apache.directory.shared.asn1=WARN
```

If you are building a plugin using maven and the car-maven-plugin:

1. Include the log4j.properties configuration file in an appropriate location in your plugin, say META-INF/log4j.properties

2. Unpack the log4j.properties file during plugin installation by including something like this in the pom.xml that generates geronimo-plugin.xml:

```
            <plugin>
                <groupId>org.apache.geronimo.buildsupport</groupId>
                <artifactId>car-maven-plugin</artifactId>
...
                    <instance>
                        <plugin-artifact>
                            <copy-file relative-to="server" dest-dir="var/directory">META-INF/log4j.
properties</copy-file>
...
                        </plugin-artifact>
                    </instance>
                </configuration>
            </plugin>
```

If you are deploying your app by some other means:

Copy the log4j.properties file by hand to the appropriate location such as var/my-app/log4j.properties. There is no need to include this file in your app.