Commons Logging FUD

Commons Logging FUD

If you take a look at commons logging, you will sooner or later meet the various pages that recommend **not to use commons-logging**. You know whose pages these are, as examples, here are a few:

- Think again before adopting the commons-logging API
- Commons-logging revisited (The language of this article might reflect on the author)

However, if you look closer at these rants, you will notice, that much boils down to a single phrase: "class loader problems".

Class loader problems are commons-logging's fault? Hardly. It is the mixture of a popular project, some unfortunate class loader decisions and the rants of people who might not understand all the implications of the web container class loaders.

Note: the following applies to Tomcat 5.5; later versions of Tomcat store the jar in the same place but change the package name so it is only used by Tomcat itself. See Tomcat 6 class loader reference. However the problems caused by the Tomcat 5.5 implementation continue to affect the perception of Commons Logging.

If you look at the Tomcat 5.5 class loader reference, you will notice that commons-logging-api.jar was put into the bin/ directory and is available through the System classloader. That was the unfortunate decision. The same issue happens BTW with common-daemon. Why does no one complain? Maybe because it is not such a popular project?

So the class loading sequence is

- Bootstrap classes of your JVM
- System class loader classses (described above)
- /WEB-INF/classes of your web application
- /WEB-INF/lib/*.jar of your web application

Look at the second point. That is the source of the pain. But there is easy relief. The documentation of the *Catalina* class loader states:

Catalina - This class loader is initialized to include all classes and resources required to implement Tomcat 5 itself. These classes and resources are **TOTA LLY invisible to web applications**. All unpacked classes and resources in $CATALINA_HOME/server/classes, as well as classes and resources in JAR files under $CATALINA_HOME/server/lib, are made visible through this class loader.

Looks like what we want, doesn't it?

First, move the jar out of the System classloader:

mv $CATALINA_HOME/bin/commons-logging-api.jar $CATALINA_HOME/server/lib

Secondly, in the $CATALINA_HOME/bin directory, there are two scripts which reference commons-logging-api.jar directly: catalina.sh and catalina.50.sh (or the respective .bat files for Windows). Replace the "$CATALINA_HOME"/bin/commons-logging-api.jar references with "$CATALINA_HOME"/server/lib /commons-logging-api.jar.

commons-logging is now removed from the System class loader but still available through the Catalina class loader. However, as you had to add it to the CLASSPATH in the catalina startup files, it is also available through the root class loader (because it is in the CLASSPATH). This is unfortunate, but no problem because you can override this with dropping your desired commons-logging jar into WEB-INF/lib.

Finally, let me quote another commons-logging advocacy page which sometimes is used as an indicator that even its author considers commons-logging a bad idea. In the words of Rod Waldhoff, one of the inventors of commons-logging (on his weblog):

The purpose of Commons Logging is *not* to isolate your code from changes in the underlying logging framework. (That's certainly easy enough to do on your own, and not really worth doing in the first place given the ease of switching from one logging framework to another.) The purpose of Commons Logging is *not* to somehow be more useful than actual logging frameworks by being more general. The purpose of Commons Logging is *not* to somehow take the logging world by storm. In fact, there are very limited circumstances in which Commons Logging is useful. If you're building a stand-alone application, don't use commons-logging. If you're building an application server, don't use commons-logging. If you're building a moderately large framework, don't use commons-logging. If however, like the Jakarta Commons project, you're building a tiny little component that you intend for other developers to embed in their applications and frameworks, and you believe that logging information might be useful to those clients, and you can't be sure what logging framework they're going to want to use, then commons-logging might be useful to you.