Identifying and Updating Grouper Libraries (2017-2018)

Introduction

The current implementation of Grouper uses Ant tasks to build its artifacts, relying on third-party dependencies being supplied along with each submodule. The versions of these jar files are not always clear, as the version is often missing from the filename, and the MANIFEST.MF file is sometimes missing an Implementation-Version property or it is inaccurate. When comparing these files versus official sources (e.g. Maven repositories), some of them are identical to published versions, some are close matches, and some can't be matched to any known version. One goal of this project is to replace all official jars with ones from known provenance. Files should always include the version number in the filename, to make it much easier to identify the version, and to make it easier to track down in public locations.

In addition, some of these supplied jars are many versions behind the current release. Where possible, we should upgrade to the latest version that is still compatible without breaking existing functionality.

The identification compares local jars against versions uploaded to the Maven Central Repository (http://search.maven.org/). The versions for supplied jars are taken first from the MANIFEST.MF property Implementation-Version if available, then from the version in the filename, otherwise any other clues in the MANIFEST file of the jar. If the version can be identified, a guess at a corresponding version in the Maven repository is compared, The strongest match is an identical checksum. The second best is that all of the class files are a binary match, disregarding Java source code, Javadoc html files, testing classes, or other miscellaneous files. If there are extra or missing classes, it suggests that either the alleged version is misidentified, or the supplied jar bundles multiple libraries into a single jar file. If there still is no good match, the supplied file is left alone, as it will likely be upgraded later.

Testing upgraded jars will first involve running the full test suite from the source code JUnit tests (2100+ tests over 5 hours), to verify no regressions are introduced. Then the Grouper UI, Web Services, and other submodules will need to be broadly tested to confirm all functions continue to work.

Status as of Grouper 2.4.0 Release (in 2018)

As of Grouper release 2.4.0, the libraries have been updated in the official distribution. The equivalent dependency versions in Maven pom.xml files have also been updated to match as close as possible. Grouper 2.3.0 or earlier versions have not been updated, even with patches, except to remedy specific issues.

Grouper API

Phase 1 - Identify jars identical or close matches with official versions, rename or replace with official one if needed

Of the approximately 70 jars in the lib/grouper, lib/jdbcSamples, and lib/jetty directories, a good number already match both the checksum and the filename, so that nothing needs to be done.

- antlr-2.7.7.jar
- c3p0-0.9.5.2.jar
- commons-lang3-3.5.jar
- cron-parser-core-2.9.jar
- dom4j-1.6.1.jar
- ehcache-core-2.4.3.jar
- groovy-all-2.5.0-beta-2.jar
- hibernate-c3p0-5.0.4.Final.jar
- hibernate-commons-annotations-5.0.0.Final.jar
- hibernate-core-5.0.4.Final.jar
- hibernate-ehcache-5.0.4.Final.jar
- hibernate-jpa-2.1-api-1.0.0.Final.jar
- jandex-2.0.0.CR1.jar
- javassist-3.18.1-GA.jar
- jboss-logging-3.3.0.Final.jar
- jline-2.14.5.jar
- joda-time-2.9.7.jar
- mchange-commons-java-0.2.11.jar
- slf4j-api-1.6.1.jar
- slf4j-log4j12.jar (near match)

Some others match the checksum of a published file, but are missing the version number from the filename. These can be renamed:

- bsh.jar -> bsh-2.0b4.jar
- vt-ldap.jar -> vt-ldap-3.3.9.jar
- commons-math.jar -> commons-math-1.1.jar

- jakarta-oro.jar -> oro-2.0.8.jar
- commons-codec.jar -> commons-codec-1.3.jar
- commons-betwixt.jar -> commons-betwixt-0.8.jar
- commons-cli.jar -> commons-cli-1.4.jar
- commons-httpclient.jar -> commons-httpclient-3.0.jar
- commons-discovery.jar -> commons-discovery-0.4.jar
- jetty/servlet-api.jar -> servlet-api-2.5-20081211.jar

Others don't match the checksum, but have identical .class objects within the files, so wouldn't be affected by updating to the official version. The file differences are solely in the existence of javadoc or source code, or in other properties in the manifest file.

- ant.jar -> ant-1.7.1.jar
- asm-util.jar -> asm-util-3.3.1.jar
- asm.jar -> asm-3.3.1.jar
- backport-util-concurrent.jar -> backport-util-concurrent-3.0.jar
- commons-beanutils.jar -> commons-beanutils-1.7.0.jar
- commons-collections.jar -> commons-collections-3.2.1.jar
- commons-io.jar -> commons-io-1.4.jar
- commons-lang.jar -> commons-lang-2.6.jar
- commons-logging.jar -> commons-logging-1.1.1.jar
- ddlUtils.jar -> ddlutils-1.0.jar
- ezmorph.jar -> ezmorph-1.0.6.jar
- json-lib.jar -> json-lib-2.3-jdk15.jar
- jsr107cache.jar -> jsr107cache-1.0.jar
- jug.jar -> jug-1.1.2.jar
- log4j.jar -> log4j-1.2.17.jar
- quartz.jar -> quartz-2.2.2.jar
- smack.jar -> smack-3.1.0.jar
- xpp3_min.jar -> xpp3_min-1.1.4c.jar
- xstream.jar -> xstream-1.3.jar
- jetty/jetty.jar -> jetty-6.1.24.jar
- jetty/jetty-util.jar -> jetty-util-6.1.24.jar
- jdbcSamples/mysql-connector-java-bin.jar -> mysql-connector-java-5.1.38.jar
- jdbcSamples/postgresql.jar -> postgresql-9.3-1101-jdbc3.jar

The remaining files either didn't have its version available in the Maven repository, or had significant differences with the published version, so more analysis is needed.

- c3p0-oracle-thin-extras-0.9.5.2.jar ("0.9.5.2" not in Maven; vs. "0.9.5" just two classes differ)
- commons-jexl.jar (15 classes differ)
- jamon.jar ("2.7" not available; vs. 2.75, 98 classes differ)
- activation.jar (a few classes differ)
- odmg.jar (all classes differ, though file sizes are the same)
- mailapi.jar (version "1.3.2" not available; vs. 1.4.2, many classes differ, plus newer missing a few classes)
- smtp.jar (version "1.3.2" not available; version 1.4.4 may be compatible, newer adds support for ssl, only a few classes changed)
- p6spy.jar (newer missing many classes)
- commons-digester.jar (supplied jar has both Digester and Digester3 packages; would need to supply both jars if replacing)
- jta.jar ("1.0.1B" not available ; vs. 1.1, 17 classes differ)
- jdbcSamples/hsqldb.jar (classes differ vs. 2.3.3, supplied manifest version "private-2015/07/02-14:15:34" by company "ft")
- jdbcSamples/ojdbc6_g.jar (no official version available, due to licensing)
- jdbcSamples/sqljdbc4.jar (no official version available, due to licensing)

Phase 2 - Identify unneeded jars

It's feasible that, over time, a library may have been added that is no longer needed due to code changes. If a library can be safely removed from the distribution, it means a smaller distribution file size, and is one less library to be maintained and updated. If a library is not needed by the Grouper API library but is needed in another subproject, it should be moved out of the API library folder and into the other project's folder.

The Grouper source code has certain direct dependencies on third-party library jars. Many of these libraries have dependencies of their own. Two methods that can be used to discover these dependencies are the Java jdeps command and Maven packaging operations. While these methods can reliably identify compile-phase libraries, it is harder to know about classes that are needed at runtime.

- asm-3.3.1.jar
- asm-util-3.3.1.jar
- commons-betwixt-0.8.jar
- (NO THIS IS NEEDED BY MANY CLASSES) commons-lang3-3.5.jar
- invoker.jar (this was only used pre-Java 1.4 to load a wildcard directory of jars for GSH; it only is referenced in comments in the GSH boot script)
- (NO THIS IS NEEDED TO RUN TESTS) jamon.jar
- joda-time-2.9.7.jar
- jsr107cache-1.0.jar
- odmg.jar
- (NO THIS IS NEEDED AT RUNTIME BY ddlUtils) oro-2.0.8.jar
- smtp.jar (javax mail – not mailapi – plus activation.jar can replace the necessary smtp functions)
- xpp3_min-1.1.4c.jar (its classes are all duplicated in smack.jar)

Phase 3 - Where possible, upgrade remaining jars to the latest version

Already the latest version:

- vt-ldap-3.3.9.jar
- commons-cli-1.4.jar
- hibernate-jpa-2.1-api-1.0.0.Final.jar
- jline-2.14.5.jar
- (removed, actually) joda-time-2.9.7.jar
- groovy-all-2.5.0-beta-2.jar
- c3p0-0.9.5.2.jar

correctly named but not current version -> git rm and git add the new one:

- bsh (2.0b4.jar -> 2.0b5.jar)
- commons-codec (1.3 -> 1.11)
- commons-discovery (0.4 -> 0.5) (removed later as potentially unneeded)
- commons-httpclient (3.0 -> 3.1)
- commons-lang3 (3.5 -> 3.7)
- commons-math (1.1 -> 1.2)
- cron-parser-core (2.9 -> 3.4)
- dom4j (1.6.1 -> 2.1.0)
- ehcache-core (2.4.3 -> 2.6.11)
- hibernate-c3p0 (5.0.4.Final.jar -> 5.0.12.Final.jar)
- hibernate-commons-annotations (5.0.0.Final.jar -> 5.0.1.Final.jar)
- hibernate-core (5.0.4.Final.jar -> 5.0.12.Final.jar)
- hibernate-ehcache (5.0.4.Final.jar -> 5.0.12.Final.jar)
- jandex (2.0.0.CR1.jar -> 2.0.4.Final.jar)
- javassist (3.18.1-GA.jar -> 3.22.0-GA.jar)
- jboss-logging (3.3.0.Final.jar -> 3.3.1.Final.jar)
- mchange-commons-java (0.2.11 -> 0.2.14)
- slf4j-api (1.6.1 -> 1.7.25)
- slf4j-log4j12 (1.6.2 -> 1.7.25)

Files at the latest version, allegedly the correct name but not an official maven version -> replace with official one and update in git:

- commons-lang-2.6.jar
- ddlutils-1.0.jar
- ezmorph-1.0.6.jar
- jug-1.1.2.jar (did not attempt to upgrade to the 2.* series) ? - log4j-1.2.17.jar

Files that weren't a good match to Maven anyway, git remove and replace

- activation (1.0.2? -> 1.1.1.jar)
- ant (1.7.1 -> 1.10.1.jar)
- backport-util-concurrent (3.0 -> 3.1.jar)
- c3p0-oracle-thin-extras (0.9.5.2 -> 0.9.5.jar)
- commons-beanutils (~1.7.0? -> 1.9.3.jar)
- commons-collections (3.2.1 -> 3.2.2.jar)
- commons-digester (3.2? -> 3.2.jar)
- commons-jexl (2.0.1? -> 2.1.1.jar)
- commons-logging (1.1.1 -> 1.2.jar)
- jamon (2.7? -> 2.81.jar)
- json-lib (2.3-jdk15 -> 2.4-jdk15.jar)
- jta (1.0.1B? -> 1.1.jar)
- mailapi (1.3.2? -> 1.4.3.jar)
- p6spy (1.1? -> 3.6.0.jar)
- quartz (2.2.2 -> 2.3.0.jar)
- smack (3.1.0? -> 3.1.0.jar)
- xstream (1.3 -> 1.4.10.jar)

Note that the following could not be upgraded to the latest version due to compile or test issues:

- antlr (2.7.7 -> 3.5.2) is not compatible without Grouper code changes
- commons-math (1.1 -> 1.2) fails to compile
- hibernate >5.0.12 had various issues during compile or testing; will tentatively keep at 5.0.12 wich is the latest of the 5.0 series
- ehcache-core (2.4.3 -> 2.6.11) had issues; will tentatively keep at 2.4.8 which is the latest of the 2.4.* series
- commons-io (1.4 -> 2.6) had compatibility issues with the InstrumentationThread class

Grouper UI

Phase 1 - Identify jars identical or close matches with official versions, rename or replace with official one if needed

Identical checksums, can be renamed:

- ant-contrib.jar -> ant-contrib-1.0b3.jar
- struts-el.jar -> struts-el-1.2.4.jar
- xml-apis.jar -> xml-apis-1.3.02.jar

Don't match checksum, but *.class files identical (difference is *.java, javadoc, or manifest)

- commons-fileupload.jar -> commons-fileupload-1.2.1.jar
- jstl.jar -> jstl-1.1.2.jar
- opencsv.jar -> opencsv-1.8.jar
- xercesImpl.jar -> xercesImpl-2.7.1.jar

Most classes match, but a few minor differences:

- struts.jar -> struts-1.2.4.jar (Maven version has a few extra classes)
- standard.jar -> standard-1.1.2.jar (supplied jar has extra tld and some test docs)

Phase 2 - Identify unneeded jars

Using Java jdeps command, along with explicit dependencies listed for Maven versions of included jars, identify libraries that can potentially be removed:

- jakarta-oro.jar -> removed
- serializer.jar -> removed
- taglibs-datetime.jar -> removed*
- taglibs-request.jar -> removed*
- xalan.jar -> removed

It was later discovered that taglibs-datetime and taglibs-request are needed by Struts 1 and the admin UI. They are being left out since the admin UI will eventually be removed.

Phase 3 - Where possible, upgrade remaining jars to the latest version

- commons-fileupload 1.2.1 -> 1.3.3
- csrfguard.jar 3.1.0-SNAPSHOT? -> 3.1.0
- jstl 1.1.2 -> 1.2.1
- opencsv 1.8 -> 2.4
- xercesImpl 2.7.1 -> 2.11.0
- xml-apis 1.3.02 -> 1.4.01

Also, the supplied servlet.jar seems to be the fully implemented servlet library normally supplied by the servlet container. Thus, it was replaced by the API stubs servlet-api and jsp-api

- servlet.jar (replaced with servlet-api and jsp-api)
- servlet-api 2.5 (from the API jetty/ folder) -> 3.1.0 (moved to the API grouper folder)
- jsp-api 2.0 -> 2.3.2-b02

Grouper WS

As most of the libraries in this project already match versions from the Maven repository, little needed to be done here. Upgrading the exising jars, along with analysis of potentially unneeded libraries, weren't done at this time.

- commons-lang.jar -> commons-lang-2.1.jar
- junit.jar -> junit-3.8.1.jar OR junit-4.1.jar depending on submodule
- ant-contrib.jar -> ant-contrib-1.0b3.jar
- servlet.jar -> servlet-api-2.5.jar
- commons-net.jar -> commons-net-1.3.0.jar
- stax-1.2.0.jar -> refreshed from Maven

Subject

The subject module has been combined into the grouper module.

GrouperClient

The grouper client is aimed at creating an executable jar. Thus, there are no external libraries involved.