ReleaseShoppingList

Release Shopping List

ATM we've got a problem with releases. They take too long to cut and check since there's lots and lots of little things that good Commons releases should have that Maven 1 doesn't do by default. This is the shopping list. Please help to fill it up.

NOTICE + LICENSE

Both LICENSE and NOTICE files need to be included: by default maven only includes the LICENSE

Status: Easy with the assembly plugin, may need to be configured for the JARs.

A: (Maven2: http://maven.apache.org/plugins/maven-assembly-plugin/descriptor.html - a description of what the maven2 standard distributions look like)

Consistent Line Endings

Having Windows line endings in the zip and *nix in the tar.gz would make many Windows users happy.

Status: Maven 2's assembly plugin lets you group filesets with a particular line ending and file mode, but these are not treated differently between zip and tar.gz without recreating the descriptor. I prefer this operation - making the main text files and batch files CRLF, shell scripts LF, and everything else untouched. Modern editors on both systems work with either, so all we need is for notepad to be able to read the .txt files, really.

The maven 1 dist plugin has been patched (in svn, pending 1.7 release) to support configurable filters for both crlf and lf conversion. CRLF is applied only in zip archives, lf (if present) afterwards. See http://jira.codehaus.org/browse/MPDIST-28.

Accurate Building Against 1.3

It's very important commons to be able to build against Java 1.3 even now maven requires 1.4. Good 1.3 builds using 1.4 is tricky. Difficult to solve this one.

Status: Maven 2 can currently allow forking of the compiler, tests, etc under installed JDKs in the same way as m1. Maven 2.1 intends to allow the application of a toolchain consistently by specifying a JDK installation location. We should look at having build servers with the necessary software rather than the user having to worry about it.

(More Info) It feels like the 'maven way' would be to specify the target JVM in the POM. Of course, it's not that easy. One of the wrinkles is that it may be that the way I've always done targeted releases (using a JVM of the current vintage) is not actually the right way. The problem is that older compilers may have bugs which have been addressed in later releases. So, AFAIK the best advice is to use -bootclasspath set and then use a modern JVM with the correct flags set.

Alternatively, in m1, maven.compile.executable can be set to force compilation on a specific jdk. The jar plugin (or comparable m2 thingy) should be able to record the correct jdk version in the manifest (m1 cannot do this currently).

IDE Friendly Source And Javadoc Jars

Modern IDEs can associate javadoc and source jars with the binaries. We haven't distributed these in the past but doing so would make many folks very happy.

Status: Maven 2 has this capability by default when using the release plugin.

Expanding Source And Binary Distributions

Source and binary distributions should expand to different directories.

Signing Artifacts

ASCII armored detached signatures are required for all Apache distributed artifacts.

Release Candidates

Need a way to produce the final release and test it before deploying it. Commons has a particular problem with doing numerous release candidates.

Status: I think it might be better to prepare untagged snapshots until folks are mostly happy, then produce an official RC for further testing that is finally just renamed to the final release.

Easy Deployment of final Release to /dist/ as well as artifacts to the maven-repository

Deployment should be a one step process to release both.

Status: releases can be put in the repository, possibly an apache plugin should symlink these together.

Response: an Apache plugin sounds like a very good idea (if you're suggesting what I think you are) - would the Apache plugin handle the details of the Apache distribution layout (all those symlinks and so on down into binary and source directories)? If so, would it be possible to make this all a single operation?

Comment: If deployment is to be automatic, it needs to remove the old deployed version.

Versioned Release Documentation

Given that we keep a large number of old releases around and there are a lot of old commons releases in circulation, we like to upload the documentation as it was at the state of the release to a versioned directory on the site. So, for example, commons/collections/3.0/ might contain the documentation that was shipped with collections-3.0.

Comment: Or put the javadoc in a separate location to the site.

Multiproject Site problems

Brett has written on a solution to improve the multiproject site: http://maven.apache.org/plugins/maven-assembly-plugin/descriptor.html (above link looks misplaced - not clear what this has to do with site generation)

- Individual commons component sites must be able to be built independently and fully from svn and source distributions
- Commons component sites need to share a common look and feel, which may not be the same as the maven-supplied default
- Commons component sites need to share common navigation elements
- Site generation must include support for versioned javadocs
- It would be great not to have to include all of the images with all of the component sites and to reduce the size (incredible amounts of needless white space) of the generated html.

Common Release Configuration

At the moment, there's a lot of duplication in the maven.xml. Ideally, we should be able to feed any improvements made in centrally. This is becoming increasingly important as the number of components rises.

Accurate, Specification Compliant Manifests

Suspect Maven does a reasonable job on this ATM but thought it probably needs including for completeness.

It looks like implementation-vendor-id and specification-version seem to be the main problems (http://jira.codehaus.org/browse/MPJAR-51). It's becoming increasingly important that we set these attributes correctly since users are starting to want to use the extension mechanism.

Automatically generated, complete release notes

The maven 1 changelog report generates passable release notes, but needs to be customized to be complete and still lacks svn integration. We need a way to automatically generate full release notes from changes.xml or a similar store, together with additional text describing the release.