ReleaseChecklist

NOTE: This is outdated information that applies only to Tapestry 4. For the current process see

http://tapestry.apache.org/release-process.html

There are several steps between a consensus forming about a new release, and actually delivering the release. This has been an area of confusion, and the Tapestry team has certainly trodded on a few Jakarta PMC toes. Let's try and get this right in the future!

- Note: certain aspects of this discussion apply to 4.0 and above (such as running the build and packaging the results).

Discuss

This isn't necessarily a step, but should be considered. Before initiating a vote, generate a discussion. Send mail to the developer mailing list with the subject `[DISCUSS] Release 4.0-beta-13` (adjust for the actual release being proposed).

Generally speaking, we shouldn't go forward to a vote unless we're pretty sure of the outcome ahead of time.

Vote Mail

Again, email to the developer mailing list, as `[VOTE] Release 4.0-beta-2176`.

Try to be specific in the email about the timetable, who would do the vote, how long it will last, the rationale, etc. Don't forget to include your +1 vote.

For example:

```
To: tapestry-dev@jakarta.apache.org
Subject: [VOTE] Release 4.0-beta-2176

The latest bug fixes really seem to be kicking things into high gear. We should get a release out the door to
spot
more problems and see if people can follow the new documentation. This vote will run for a week, a +1 is to
release Tapestry 4.0-beta-2176.
I'll be able to generate the release over the weekend following the vote.

Howard M. Lewis Ship: +1 (binding)
```

Also remember the meritocracy rules ... a -1 indicates a veto, but only counts if you can provide a reasonable rationale.

- To be honest, I'm confused about whether release votes can be vetoed, or whether it is simple majority rules. Clarification, anyone?

Again, *wait* for all the votes to come in, or for the clock to finish. We have a couple of relatively inactive members (which is a total shame), but we have to follow the rules.

The commiter who initiates the vote has *volunteered to be the release engineer*. You wanted it enough to start a vote, be prepared to carry it through. You will be responsible for all the remaining steps. If you need help, ask!

Voting

Committers can make it easy on the release engineer by adding "(binding)" to their vote. Non-committers should not do this .. their votes are important, but not binding.

Result Mail

Following the last vote mail, send a `[RESULT]` mail to the development list **and** CC `pmc@jakarta.apache.org`. We have enough Jakarta PMC members on the Tapestry team that we don't require authorization from the Jakarta PMC, but we **do** have to notify them.

The mail should include the text of the vote and *all* responses; binding votes first, non-binding votes second.

Following the [RESULT] mail, everyone (except the release engineer) should hold back on check ins until the all clear is given.

Final Changes

You should update `status.xml`:

- Record the vote in a <vote> stanza
- Update the `version` and `date` attributes of the property <release> stanza

In some cases, the version may need to be updated; it may change from, say, "4.0-alpha-3" to "4.0-beta-1" if we decide to follow 4.0-alpha-2 with 4.0-beta-1.

Make sure to update the `version.properties` value in the file `project.properties` as well.

Finally, take a peek at the KEYS file. You'll sign the KEYS file using your GnuPG or PGP key. Instructions are inside the file itself. This KEYS file is how diligent users validate that the final release is, in fact, authenticated and valid.

Check in these changes.

Run the build

I prefer to run the build then label ... occassionally you find a minor issue that is wrong and needs to be a last minute fix.

```
ant -emacs dist
```

Note: this often fails generating the forrest documentation (I believe its a problem with Forrest, not with our documentation); that's ok, just follow up with `ant -emacs dist-build` to pick up after Forrest runs.

This will perform a final, clean build of the Tapestry framework, contrib, examples and documentation. It will then package the results, ready for upload.

This takes about 15 minutes on my laptop. The framework unit tests can take up-to three minutes by themselves (those damn integration tests, TestMocks, take forever). And the test suites are run twice: once for the build, once to collect code coverage information. Forrest is also time consuming.

The final files will be placed in `target/dist`. There are three files generated:

- tapestry-*version*.tar.gz – main distribution; compiled binaries plus source
- tapestry-*version*.zip – larger, zip version of above
- tapestry-*version*-docs.tar.gz – documentation as a web site

In addition, MD5 hash files are created for each of the three files (these are the .md5 files in target/dist).

Recent change: a large file named target/tapestry-examples-*version*.tar.gz is now created, and must be uploaded to Howard's home page (which is why Howard usually does the builds).

It doesn't hurt to give these files a once over to make sure they built correctly (occasionally, build file errors cause erronous files to be included in the distribution).

The build scripts should end with a reminder to sign the release.

Label The Release in SVN

If the release files are all set, then it's time to lock down the versions in CVS.

Use SVN to create a copy of trunk to a new folder under tags with the same name as the release.

This can be done from the IDE, and generates a SVN command like:

```
copy -rHEAD https://svn.apache.org/repos/asf/jakarta/tapestry/trunk https://svn.apache.org/repos/asf/jakarta/tapestry/tags/4.0-beta-12
```

Sign the release

The files uploaded into Maven and to the mirrors need to be signed. Here's a quicky for doing it using GnuPG and Bash:

```
for i in target/*.jar target/dist/*.gz target/dist/*.zip
do
  echo "Signing " $i
  gpg -a -b --force-v3-sigs $i
done
```

Lucky you ... you get to type your GnuPG pass phrase again and again and again!

The signatures show up as .asc files.

Upload the release

Ant handles this for you:

```
ant -emacs install-dist install-docs install-maven -Ddist.install.user=userid
```

You need to provide your Jakarta login user id (you can also update `project.properties` for this).

You will be prompted for your pass phrase ... in cleartext yet. Sorry! Have to do it. Ant's ssh support doesn't seem to know from ssh-agent.

The `install-dist` target installs the main distribution files (plus md5 sums and PGP signatures, and KEYS) into /www/www.apache.org/dist/jakarta/tapestry. This directory is mirrored to all the Apache download mirrors.

The `install-docs` target uploads the documentation and expands it to form the Tapestry home page.

The `install-maven` target copies the Tapestry JAR files to /www/www.apache.org/dist/java-repository/tapestry/jars, where they will be mirrored to ibiblio.net, so the whole world can access them using Maven.

You should only have to enter your pass phrase once.

Send The All Clear

Send mail to the developer mailing list ... the the checkins begin!

Note: The first checkin should increment the `project.version` release number.

Also, don't forget to create a **new** <release> stanza!

Perhaps the release engineer should do this before sending the all clear?

Test the Downloads

Make sure all the files you just uploaded are readable! Log into jakarta.apache.org and fix permissions if they are not!

Wait 24 hours

It takes up-to 24 hours for the mirrors to synchronize. If you send out the mail too soon, people who go to download the lastest and greatest will be dissapointed.

Update jakarta-site2

This is a whole process onto itself; you need Subversion to get the jakarta-site2 stuff. It's yet-another XML-like limited HTML syntax (related to Forrest markup, but not quite). Once you check it out, there are directions on how to proceeed.

The url of the jakarta-site2 svn repository is: `https://svn.apache.org/repos/asf/jakarta/site`

You would need to use the build scripts to rebuild the documentation, then check in the *derived HTML files* as well as the XML source files. Yes, I cringe.

You need to do three things:

- Add a <release> stanza to the news.xml file.
- Update `xdocs/downloads/downloads.xml`.

The <release>s inside <news> are inside <group> elements; if you are the first person to update news.xml in a given calendar quarter, you may need to create a new <group>.

About `downloads.xml` ... mostly you just put the correct release number into the XML entities defined at the top. For interim releases, simply update the release numbers within the relevant `<download>` entries. For the very infrequent stable releases (i.e., for a 3.1 or 3.2 final release, a rare event) update the download `<group>` as well.

Email the Announcement

Whew! Send an [ANNOUNCE] email to general@jakarta.apache.org and tapestry-user@jakarta.apache.org. Use the same (or similar) text to the news item. Get a beer!