ReleaseNumbering

So ... how are Tapestry releases numbered?

There are two answers. Tapestry had an idiosyncratic approach for releases up to Tapestry 4.0. Tapestry is switching over to a more standard, more Apache, approach for newer releases.

Older Numbering

Older versions of Tapestry consisted of a major number, a minor number, a stability code, and a index number. Examples:

- 4.0-alpha-1
- 4.0-beta-2
- 4.0-rc-2
- 4.0

In all of the above the major number is 4 and the minor number is 0. Changes to the major release number indicates a non-backwards compatible change. Changes to the minor release number indicate a compatible change (existing code will compile and run against the new library).

"alpha", "beta", "rc" (release candidate) are stability numbers. In *theory*, a release without such a code is a final, stable release.

You can expect a series of alpha releases, the betas, then a few rc's, before the final release.

This is good in theory, but problematic in practice. Tapestry has always suffered from backwards compatibility problems from release to release. That's why there's been a jump from 2.2 to 3.0 to 4.0. We're making amends there, but until Tapestry 5, it will be less than perfect because of the vast amount of "luggage" pulled along from earlier releases.

In addition, what looks stable enough to release as a final release often isn't, causing a flurry of 4.0.1, 4.0.2, 4.0.3 bug fix releases after the supposed final release.

To some degree, the number of Tapestry releases and the complexity of naming them, is a reflection of how hard it was to build Tapestry from source. Future versions of Tapestry will build using Maven, making the requirement to release often less necessary (and it will be easier to generate nightly builds).

New Numbering

The new numbering is more standard. The Apache Portable Runtime has an excellent description of numbering.

With new numbering, the release number is a sequence of numbers. Major version, minor version, and patch level. Examples:

- 4.1.1
- 4.2.7
- 5.0.1

The patch level, the final number, changes with bug fixes that do not change API or semantics. Users may upgrade and downgrade to different patch levels and expect that code is source and binary compatible.

Minor number changes represent modest changes to public APIs. Generally, this should be in the form of deprecations and additions. Removals need to wait for Major number changes.

Process

First, a vote to release. This is chance for the developers to confirm that the release is ready, that the number makes sense (i.e., that compatibility rules have been followed). Votes should run a minimum of 72 hours and follow lazy conscensus. Starting a release vote is also volunteering to be the release engineer.

After verifying the build (doing a fresh checkout and clean build), the release engineer should:

- Send a notification to the developer list.
- Commit any final changes, such as version numbers.
- Create a new branch for the released code.
- Build the project and site, and upload the project jars and the site contents. Maven will take care of all that.
- Send an "all clear" to the developer list.

More details on this forthcoming.