Thoughts on Release Management

Thoughts on Release Management

Status: IMPLEMENTED
Created: 3. January 2008
Author: fmeschbe

As we should start thinking about making an initial release of Sling, we should discuss a few issues:

1. How do number releases ?
2. What do we release ?
3. When do we release ?

In this concept I propose a version numbering scheme as well as my idea for the contents of an initial Sling release and how to release in the future. Comments are very welcome.

Version Numbers

Maven has a very nice concept of *SNAPSHOT* versions. In this concept, a version just has a `-SNAPSHOT` suffix which marks the version as a SNAPSHOT (or non-release) version. OSGi versions on the other hand are simply 4-position numbers, where the parts are separated by dots, first three parts (major, minor, micro) must be numeric and the last part (qualifier) may also contain characters. The Apache Felix Maven Bundle Plugin we use to build the Sling bundles converts Maven version numbers to correct OSGi version numbers. For example, the version *2.0.0-incubator-SNAPSHOT* is converted to *2.0.0.incubator-SNAPSHOT*.

The consequence of this conversion is that a SNAPSHOT version (e.g. *2.0.0.incubator-SNAPSHOT*) would always be assumed more recent than the corresponding release version (*2.0.0.incubator* in this example). Instead of just do hackery with the release version, such as adding a different suffix, I propose the following version numbering concept:

1. Odd micro versions are always SNAPSHOTs, e.g. *2.0.1-SNAPSHOT*
2. Even micro versions are always releases, e.g. *2.0.2*

This way, a release is always more recent than the latest SNAPSHOT from which the release was built. This also means, that there is never an official release with an even micro version and there will never be an "official" SNAPSHOT with an even micro version. Thus both *2.0.2-SNAPSHOT* and *2.0.3* would be incorrect version numbers.

So, when releasing project, the minor version of the release is the next increment to the current SNAPSHOT micro version. The new SNAPSHOT micro version is again the next increment of the release version. For example to release the current SNAPSHOT *2.0.3-SNAPSHOT*, the following version numbers are used:

- Current SNAPSHOT version: *2.0.3-SNAPSHOT*
- Release version: *2.0.4*
- Next SNAPSHOT version: *2.0.5-SNAPSHOT*

The same mechanism applies when we decide to increment the minor or even the major version number: Up to the release, the SNAPSHOT remains with the former minor (or major) version and only on release, are the version numbers incremented. For example to release the current SNAPSHOT *2.0.7-SNAPSHOT* as *2.1.0*, the following numbering is used:

- Current SNAPSHOT version: *2.0.7-SNAPSHOT*
- Release version: *2.1.0*
- Next SNAPSHOT version: *2.1.1-SNAPSHOT*

For branching, the branch always gets an incremented minor version number. For example when branching off *2.1.1-SNAPSHOT* the branch version would be *2.2.1-SNAPSHOT*. The problem then is: What would be the next minor version if *2.1.1-SNAPSHOT* would be released with an incremented minor version ? One solution could be to not only define the branch version number but to also increment the trunk minor version to the next minor version. In the example, this would raise the trunk version number to *2.3.1-SNAPSHOT* (from *2.1.1-SNAPSHOT*). Alternatively, we may just append a suffix indicating the branch, such as *2.1.1.branch-SNAPSHOT*.

Both solutions actually have their drawbacks and maybe we never even need this, because I cannot imagine requiring a branch. The reason for this assumption is, that we release "small" bundles, which may easily and quickly be fixed and fix release be provided equally quickly.

Releases of Sling

For a starter we will have an initial release of Sling. This release will most probably encompass the following constituents:

- A standalone Java Application consisting of a single JAR file (or a ZIP) containing all bundles used by Sling
- A web application (WAR file) containing all bundles used by Sling
- All bundles packed into the standalone and web applications
- The parent POM

After the initial release, further releases will only be made of single bundle (or a small number of bundles). Whenever we see a release fit, we will cut it. Some bundles will have longer and some bundles will have shorter release cycles. This all depends on the importance of the fixes and/or the enhancements applied as well as the needs by the user community.

Packaged Bundle Releases

For end users and beginners it is very helpfull to have completely package Sling application (or web application) in their hands to try it out and start build web applications. In addition, such prepackaging also serves well for prototype building and "just testing it".

For this reason, there will be packaged releases of the standalone and web application once or twice a year. These applications though would just be packagings of already released bundles.

Referring to other Sling Projects

Most Sling projects depend on other Sling projects. After the initial release all Sling projects should only refer to other Sling projects by their release number. Only, if a projects uses non-released functionality of another project, should the SNAPSHOT version be referred to.

Also, the release of a Sling project will not cause the dependencies of Sling projects using the released project to be incremented. That is, each Sling project always refers to the smallest possible version number of other Sling projects.

This helps managing the dependency tree and prevents needless upgrades of bundles in running systems.