Maven Release Process

⚠ This guide is obsolete. Use Releasing a Module from Bamboo instead.

⚠ This guide assumes your module is maven-based, and stored in git. If you don't have a pom.xml file in your module, see Converting old code to Maven in before using this approach

Background

While you are doing development in a maven module, your module's working version ends in -SNAPSHOT. By default, this will start at 1.0-SNAPSHOT.

Releasing a new version of your module involves many steps (most of which are automated). Each of your module's pom.xml files (parent, api, omod) have a version that needs to be updated, the module needs to be tagged, built, tested, committed, deployed to the maven and module repositories, and then the numbers need to point to the next SNAPSHOT.

The Maven Release Plugin automates nearly all of this.

Prerequisites

Credentials to the maven repository (you need to do this once in your dev environment)

You need to have a login to mavenrepo.openmrs.org. (This does *not* use your OpenMRS ID.) You can request an account at http://go.openmrs.org/helpdesk. Once you have a username and password to the maven repository, you can specify your username and password in settings.xml (in your ~/.m2 directory):

```
<settings>
  <servers>
    <server>
      <id>openmrs-repo</id>
      <username>yourusername</username>
      <password>yourpassword</password>
    </server>
    <server>
      <id>openmrs-repo-modules</id>
      <username>yourusername</username>
      <password>yourpassword</password>
    </server>
        <server>
      <id>openmrs-repo-releases</id>
      <username>yourusername</username>
      <password>yourpassword</password>
    </server>
        <server>
      <id>openmrs-repo-snapshots</id>
      <username>yourusername</username>
      <password>yourpassword</password>
    </server>
  </servers>
</settings>
```

If you're curious, you can see more details at Managing the Maven Repository.

⚠ If you don't want to type your username in clear text (note the password is saved in clear text to the file release.properties), you can try to set up maven password encryption.

Settings in pom.xml (you need to do this once for any module you release)

The <scm> section of your module's root pom.xml needs to point to the urls that the module lives at. (Note that the maven module archetype incorrectly defaults these to point to svn, so you need to change these to point at github.) For example:

```
<scm>
        <connection>scm:git:https://github.com/openmrs/openmrs-module-yourmoduleid</connection>
        <developerConnection>scm:git:https://github.com/openmrs/openmrs-module-yourmoduleid<
/developerConnection>
        <url>https://github.com/openmrs/openmrs-module-yourmoduleid</url>
</scm>
```

Use the <distributionManagement> section your module's root pom.xml to tell it where it should deploy the module artifacts to:

```
<distributionManagement>
        <repository>
                <id>openmrs-repo-modules</id>
                <name>Modules</name>
                <url>http://mavenrepo.openmrs.org/nexus/content/repositories/modules/</url>
        </repository>
        <snapshotRepository>
                <id>openmrs-repo-snapshots</id>
                <name>OpenMRS Snapshots</name>
                <url>http://mavenrepo.openmrs.org/nexus/content/repositories/snapshots</url>
        </snapshotRepository>
</distributionManagement>
```

Via <pluginManagement>, override some of the release plugin's configuration settings:

```
<build>
        <pluginManagement>
                <plugins>
                        ...
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-release-plugin</artifactId>
                                <version>2.5</version>
                                <configuration>
                                        <autoVersionSubmodules>true</autoVersionSubmodules>
                                        <tagNameFormat>@{project.version}</tagNameFormat>
                                </configuration>
                        </plugin>
                </plugins>
        </pluginManagement>
</build>
```

Don't forget to commit and push these configuration changes to github!

Doing the Release (the Maven part)

After configuring all the prerequisites, make sure you are pointing at the branch you want to release (normally "master") all your code is committed and pushed, and you have pulled the latest version from origin (and upstream, if relevant.)

From command line, run:

```
mvn release:prepare
```

You will be asked what version number you want to release as. The default is usually correct, i.e. dropping -SNAPSHOT from the development version.

Next, you will be asked what you want to tag this release as. The default is usually correct, i.e. the version you just released as, like "1.0".

Third, you will be asked for the next development version. The default (incrementing the last digit by one from the previous development version) is often correct, but don't be afraid to increase higher order numbers.

(If you did *not* set the release plugin's configuration settings as described above, you will need to choose the same version numbers three times, and you'll need to change the tag from something like "mymodule-1.0" to "1.0".)

The release plugin will then do a lot work, (e.g. building, running tests, and tagging). If things go wrong at this point, you may do "mvn release:rollback" and delete the artifact from the releases section in github. Note for Windows users: One thing that might go wrong is it just hangs at some point, as described here: http://stackoverflow.com/questions/3243755/maven-error-releasing-code-to-github-hangs-after-push. A solution involving some clever git ssh magic is also described in that SO entry.

Assuming things went right, the next step is:

```
mvn release:perform
```

When this completes, the particular git revision will have been tagged, and its compiled artifacts will have been deployed to our maven repository.

After doing the above, the compiled artifact for the tag you just created will be available in (TODO: check this) the "target" folder of your top-level maven repository. You can take this file and upload it to modules.openmrs.org.

Troubleshooting

1. If your module contents are not published to Nexus repository due to an error during mvn release:perform
   The versions of the pom files have been incremented to a new version with the tag snapshot added, for example 2.3-SNAPSHOT.
    a. Edit the POM files to contain the version that you want to deploy e.g., 2.2.
    b. Run mvn deploy