ReleaseProcess

The information on this page is intended for Velocity committers, not as end-user documentation.

(Note - some of the specific instructions this document are out of date now that Velocity has left Jakarta and gone TLP. In particular this applies to details on the Velocity site).

Building and Releasing the Velocity Engine

The Velocity project currently is built using Apache ant. This is the canonical build and in case of doubt, the results of this build win. While it is possible to build Velocity using Apache maven, the inherently instable nature of Maven 1.x and the not yet proven reliablity of Maven 2.x make us feel that maven is not yet 'ready' to be used as primary build tool for Velocity.

Velocity up to and including Version 1.4 uses only Apache ant as its build tool. Starting with Velocity 1.5, we will still build the actual release archives with ant but nightly builds and especially the web site located at http://velocity.apache.org/ will be built using Apache maven V1. Using Maven buys us a number of interesting reports and metrics during the build process (such as automated Changelogs, metrics, web-formatted test reports and so on).

Apache Maven uses a XML file called `project.xml` and a number of properties files to control its build process.

Locations and Pathes for Velocity

Purpose	File system location (on minotaur)	web location
Release location	/www/www.apache.org/dist/velocity/engine	Mirrored through the Apache mirror system, available through http://velocity.apache.org/download.cgi
Maven repository release location	/www/www.apache.org/dist/java-repository /velocity	http://www.apache.org/dist/java-repository/velocity/
Maven repository snapshot location	/www/cvs.apache.org/repository/velocity	http://cvs.apache.org/repository/velocity/

Building the velocity site using maven

- maven site:deploy – builds the complete maven site and deploys it to the apache servers.

Due to the fact that deployment happens to the Apache web staging server from which the actual servers (which are velocity.apache.org) mirror the content every few hours or so, changes are not immediately visible. If you want to check whether the content arrived ok on the apache server, use the 209.237.227.195 trick (which is using the IP address and port 80 as your proxy host. Then access http://velocity.apache.org/ and you get the content directly from minotaur).

Snapshot deployment using Maven

There are two deployment repositories defined in `project.properties`:

```
# Repository for official releases
maven.repo.apache=scpexe://cvs.apache.org
maven.repo.apache.directory=/www/www.apache.org/dist/java-repository

# Repository for alpha, beta, rc releases
maven.repo.snapshot=scpexe://cvs.apache.org
maven.repo.snapshot.directory=/www/cvs.apache.org/repository/velocity
```

The selection of the repository happens through the `maven.repo.list` parameter. It defaults to the snapshot repository.

Q: Why are we using 'scpexe' and not 'scp' for deployment?

A: Because for Apache logins, according to infrastructure, we should stick to public/private key authentication to access the distribution servers. In that case, you would have to give the full path to your private key and also the passphrase 💡 on the command line. E.g. like this:

```
 maven -Dmaven.repo.snapshot.privatekey=/home/henning/.ssh/id_dsa -Dmaven.repo.snapshot.passphrase=verysecret
dist:deploy
```

Q: Why that? What about `ssh-agent` or `pageant`?

A: You will have to google for that answer quite a while. Here is the short answer: `ssh-agent` uses an unix domain-socket which cannot be accessed in pure Java. End of Story.

Q: scpexe:// does not work for me!

A: You have to add the following properties to your global (`~/.build.properties` on Unix) properties file:

```
maven.username= <your apache login>
maven.ssh.args=-o ForwardX11=false
```

I needed the last line to get scpexe to run on RedHat (Fedora) Linux.

Building and deploying snapshot versions using maven

After setting up your global build properties file as described above, you should be able to run the following commands:

- maven jar:deploy – sends a build of the velocity.jar to the snapshot repository
- maven dist:deploy – sends a distribution build to the snapshot repository

Please note that due to a design problem in the artifact plugin of maven, the jar or distribution archive and the related POM can have (will have if you have a slow line) different time stamps. So the POMs are basically useless.

Preparing official releases using Apache Maven

As we currently build official releases with ant, this paragraph is intentionally empty. 🙂

Preparing official releases using Apache ant

Before you start

- Make sure the file `changes.xml` has the most recent updates listed.
- Run `svn update` to get the most recent tree for release.
- check whether everything has checked in correctly: `svn status` should be empty.
- remove the `bin` and `target` directories.
- enter the `build` directory

Building the release

- Make sure you are using J2SDK 1.4.2 to build the release. This will allow the jar files to be used with JDK 1.3, 1.4, 1.5, and 1.6.
- change the version property in `build.properties` to reflect the version to be released.
- edit the file `changes.xml`, changing the `version` attribute of the topmost `release` tag to indicate the most version number to be released.
- check this change in using `svn commit`
- build the release packages using `ant release` (this is related to `ant package` but checks for the right version of Java).
- enter the `bin` directory
- Generate GPG signatures:

```
#! /bin/bash
for i in *.tar.gz *.zip *jar; do
  gpg --default-key <your key id here> --armor --output $i.asc --detach-sig $i
done
```

You should now have sixteen files in that directory: A .jar, a .jar with dependencies, a .tar.gz and a .zip file. For each you must have a .md5, a .sha1 and a .asc file. If you file count is not correct, please **DO NOT CONTINUE**.

Uploading the release to apache.org

- As personal access to people.apache.org is still possible, we will use this way. This is supposed to change sometime in 2006 (might be changed to WebDAV) but until then we are lazy.
- Copy the sixteen files mentioned above to your personal directory on people.apache.org using secure copy (scp).
- log onto people.apache.org
- make sure that your umask is 002! If not, please enter `umask 002`
- create the distribution directory: `/www/www.apache.org/dist/velocity/engine/velocity-<new version>`. Make sure that its permissions are set to 775!
- move the .tar.gz and .zip related files from your personal directory to the distribution directory.
- copy the .jar (and md5 related files) to /www/people.apache.org/repo/m1-ibiblio-rsync-repository/velocity/jars
- go to the distribution directory.
- check the MD5 and SHA1 sums by running the md5 and sha1 programs on the .tar.gz and .zip files. The output must be identical to the contents of the .md5 and .sha1 files. **Please do not skip this step!** Upload errors happen more frequently that commonly believed and testing the checksums is essential in catching these.
- verify the GPG signatures, too using gpg --verify.
- go to the java distribution directory: /www/people.apache.org/repo/m1-ibiblio-rsync-repository/velocity/jars

- please repeat the check and verify as with the distribution files

Now that the release is out, please let the mirrors some time to pick up the releases. In the meantime, please do the following preparations to make the release public.

Tag the release

- Tag the release in Subversion. This is a very important step because it makes sure that your release stays the same. Run the following command from your release tree:

```
svn copy -m 'Release <released version>' https://svn.apache.org/repos/asf/velocity/engine/trunk https://svn.
apache.org/repos/asf/velocity/engine/tags/ENGINE_<released_version>
```

(The older tags are a bit in disarray. This will get cleaned up in the future).

Build the site docs

(This section is out of date)

- remove the `bin` and `target` directories one more time.
- As the site is built using maven, do the following preparations:
    - change the <currentVersion> tag in `project.xml` to the release version
    - change the <siteDirectory> tag to /www/jakarta.apache.org/velocity/releases/engine/<release version>
    - change the <connection> and <developerConnection> tags created above. Please make sure to have the http for the connection and the https for the developer connection right!
    - change the <url> tag to point to tag above.
    - add the released version to the <versions> tag (e.g. for the 1.5 release):

```
<version>
  <name>1.5</name>
  <tag>ENGINE_1.5</tag>
  <id>1.5</id>
</version>
```

- run `maven site:deploy`
- revert your local changes to `project.xml` by running `svn revert project.`
- Due to the volatile nature of the maven site building process, it is strongly recommended, that you also back up the distribution tree built on the apache.org web site by generating an archive of the tree located under `.../releases/engine/<release version>`.

Post-upload preparations

(This section is out of date)

- Change the version in `build.properties` to read <released version + 1>-dev. Check this change in immediately using `svn commit build.properties`!

Build the global Apache download and announcement links

- check out the Apache Jakarta site module using

```
svn checkout https://svn.apache.org/repos/asf/jakarta/site jakarta-site
```

- open the xdoc/downloads/downloads.xml file and look for the id="velocity" project tag. Change the current version to the just released version.
- open the news.xml file and add a new release item on top of the most current group (It might be necessary to add a new group). Please read the comments in the file and act accordingly. (At this point you should also have the announce message ready because you can reuse it here).
- rebuild the jakarta site using `ant docs`. **CAVEAT:** If you do this on an UTF-8 based unix system (e.g. Fedora), you must run this command like this: `LANG=en_US ant docs`. Else you will nuke all the non-ascii characters in the site. **PLEASE CHECK THIS TWICE BEFORE THE NEXT STEP using `svn status`**.
- Ideally, this should change only very few files. If you have many more files changed than just the following few, please check twice before doing `svn commit`. Even better, please check in only these files anyway.
    - news.xml (which you edited)
    - downloads/downloads.xml (which you edited)
    - docs/site/downloads/downloads_velocity.html (generated)
    - docs/index.html (generated)
    - docs/site/rss.xml (rss feed, generated)
    - docs/site/news/... (generated)
- go to people.apache.org, change to `/www/jakarta.apache.org` and run `svn update index.html site` to reflect your changes.

Announcing the release

- Announcement mails should be sent to the following addresses:
    - Apache Announce list: announce at apache.org
    - Velocity Development list: dev at velocity.apache.org
    - Velocity User list: velocity-user at velocity.apache.org

Announcements should have your 'apache.org' email addresse as sender because the first two recipients discard everything not coming from an apache.org address. You should also use the same announcement text (or something closely resembling it) that you put into the news.xml file on the site.