Building Struts 2 - Normal release

Content

Building Steps (Struts)

Getting ready

1. Create an "Struts 2.x.y omnibus ticket" ticket in JIRA to refer to in upcoming release related commit comments and for general documentation purposes. Mark it with priority "Blocker".
2. Switch to branch `develop`
3. Ensure that the master POM and Struts Annotations have current releases
4. Review JIRA for any issues without a fix version set, and for any issues that should be resolved for the pending release.
5. Ensure that there are no repositories or pluginRepositories listed in the poms.
6. If you have committed all changes regarding the release process, close the omnibus ticket as it is the last open ticket for the upcoming release
7. Release the upcoming version in JIRA (under Administration/Manage Releases) and tag the release date
8. Add next milestone version to the JIRA roadmap
9. Create DONE and TODO filters for the new version, share with all, and remove obsolete TODO filter
10. Create a new Version Notes page in Confluence, link from Migration Guide, and link to prior release page and JIRA DONE filters of the version to release
11. Export wiki pages and put them under `/docs`

Update Draft Docs when needed

Checkout `struts-site` project (see details at the bottom of this page) and perform export:

```
cd struts-site
mvn package
```

If build will fail try again - don't use `clean`, the exporter is going to update only outdated pages. After successful export, commit updated files into `struts-production`

Be sure your local copy is up-to-date

```
git fetch origin --prune
git checkout master
git pull
```

⚠ Please remember to keep BOM subproject in sync - `<struts-version.version>X.X.X</struts-version.version>` - must be the same as the parent pom. The latest Maven version handles this case very well but it's worth checking if the bits are in sync.

Prepare release

Tag the release by using the "release:prepare" goal of Maven:

```
mvn release:prepare -DautoVersionSubmodules=true
```

For a dry run, add `-DdryRun=true`. If you do a dry run, use `mvn release:clean` to clean up after you have looked at the output.

When prompted for the SCM tag name, follow this pattern: STRUTS_2_3_[PATCH_VERSION]

> ⚠ If you get an error message, try to re-run `mvn release:prepare -DautoVersionSubmodules=true` command again, `-Dresume` flag is set to true by default and the plugin will resume the release process from where it failed before.

Follow the link to get more information about performed operation by release plugin.

Perform the release

```
mvn release:perform -DretryFailedDeploymentCount=10
```

Follow the link to get more information about performed operation by release plugin. After this step the artifacts will be hosted by Nexus. The `-DretryFailedDeploymentCount=10` is needed when there are problems with network connection (used just in case).

If you need to run perform again, (or in a different box), do:

```
git checkout STRUTS_2_3_[PATCH_VERSION]
mvn javadoc:javadoc deploy --no-plugin-updates -DperformRelease=true -Papache-release
```

Next, log in to Nexus and **close** staging repository.

> ⚠ Repository is identified by user name and public IP address, so if in meantime your IP changed, a new staging repository will be created so you must drop the old one (check the dates!) - if IP is the same, artifacts will be uploaded to the same repository as first attempt.

Move the assemblies

To simplify testing, the assemblies have to be moved to the `https://dist.apache.org/repos/dist/dev/struts/$VERSION` dir.

After closing repository in Nexus, check if the release files are available from staging repository as bellow:

```
https://repository.apache.org/content/groups/staging/org/apache/struts/struts2-assembly/$VERSION/
```

In order to move the assemblies login to people.apache.org and execute the following code:

```
#!/bin/sh

#create the destination directory
echo "Creating working dir $VERSION"
mkdir $VERSION
cd $VERSION

# get the distro
echo "Getting distro $VERSION"
wget -erobots=off -nv  -l 1 --accept=zip,md5,sha1,asc -r --no-check-certificate -nd -nH https://repository.
apache.org/content/groups/staging/org/apache/struts/struts2-assembly/$VERSION

# rename files
echo "Renaming files"
for f in *2-assembly*.zip*
do
 mv $f `echo $f | sed s/2-assembly//g`
done

# remove unneeded files
echo "Removing unneeded files"
for f in struts2-assembly-*.pom*
do
 rm $f
done

# remove unneeded hashes
echo "Removing unneeded files"
rm *.asc.md5
rm *.asc.sha1
cd ..

# checking in new version
echo "Pushing test version $VERSION"
svn --no-auth-cache co --depth empty https://dist.apache.org/repos/dist/dev/struts/ struts-dev
mv $VERSION struts-dev/
cd struts-dev
svn add --force ./
svn --no-auth-cache commit -m "Updates test release $VERSION"
cd ..

# cleaning up
rm -r struts-dev

echo "Done!"
```

After this step artifacts are available for test here https://dist.apache.org/repos/dist/dev/struts/

Announce availability

Send a short e-mail to dev@struts.a.o informing about the new packages and to give people enough time to test the distribution (actual bits). Wait around a week before posting Vote. If no show-stoppers reported, start a vote thread for build quality designation.

Push changes

Do not forget to push your local changes to the Apache repo

```
git push
```

Vote on it

Post a release/quality vote to the dev list (and **only** the dev list). The example mail is on Sample announcements page. If the vote result is for an ASF release (i.e. not test build), update site, announce. If the vote result is for GA, push to central.

Copy files

After the vote, if the distribution is being mirrored (there was a favourable release vote) move all the artefacts from `dev` folder into `release` folder:

```
svn mv https://dist.apache.org/repos/dist/dev/struts/$VERSION/ https://dist.apache.org/repos/dist/release
/struts/
```

Promote release

Log in again to Nexus and **release** the repository, it will be automatically replicated across Maven Repositories
See Releasing a Maven-based project for further details.

Clean up old releases

Remove the old files from under https://dist.apache.org/repos/dist/release/struts/ to synchronise only the latest version with peers. All the files from https://dist.apache.org/repos/dist/release/struts/ are always mirrored to http://archive.apache.org/dist/struts/. You can use the below command:

```
svn del https://dist.apache.org/repos/dist/release/struts/2.3.x/
```

where `x` is the previous version to remove (or one more previous to keep current and one version back).

Wait for rsync

Wait 24 hours before proceeding.

Update site

- Make sure you have linked your Apache and Github account in Apache GitBox (Dual Master Git alowing you to directly push to GitHub), see https://gitbox.apache.org/setup/
- Check out site src code

  ```
  git clone https://github.com/apache/struts-site.git
  ```

  or use SSH instead:

  ```
  git clone git@github.com:apache/struts-site.git
  ```

- If a new DTD was defined, add it to `source/dtds`
- Update current version and release date in `struts-site/_config.yml`
- Update page source files
  - struts-site/source/announce.md (if applicable, refer also to corresponding security bulletin)
  - struts-site/source/downloads.html (Prior Releases section)
  - struts-site/source/index.html (some parts will updated automatically with values defined in `_config.yml`)
- Generate site with Docker Jekyll image
  - you must have Docker installed and running
  - if you are doing this the first time, download the official Struts image to build the site from https://hub.docker.com/r/theapachestruts/struts-site-jekyll/
  - start `docker-machine`
  - now you can use one of the bash scripts already provided in the `struts-site`:
    - `docker-run.sh` - used with Bash
    - `docker-run.fish` - to use with Fish Shell (via `fish docker-run.fish`)
  - now you can check the generated site at http://localhost:4000
- Commit the changes and the generated content

Now the changes must be deployed to production which is basically a separated Subversion repository, you check it out with command below:

```
svn co https://svn.apache.org/repos/infra/websites/production/struts/content struts-production
```

It's a good idea to keep that working copy to be used with future releases. Right now copy content of `struts-site/content` folder to `struts-production` folder, then commit changes. Next step is to update exported wiki pages. With current approach the pages are kept in `struts-production/docs`.

Redeploy the docs (Optional)

- Checkout source of the website and export Confluence pages

```
svn co https://svn.apache.org/repos/asf/struts/site/trunk struts-site
cd struts-site
mvn package
```

Now the whole Confluence space is exported to `target/cwiki/WW/docs/`

- Checkout copy of production website (if you didn't that before)

```
svn co https://svn.apache.org/repos/infra/websites/production/struts/content/ struts-production
```

(you can checkout just a subtree, but it's better to checkout the whole repo especially when you want to update also the main web page)

- Update production

```
cp -r struts-site/target/cwiki/WW/docs/* struts-production/docs/
cd struts-production
svn commit "Updates production"
```

Post announcements

We leave this as the last step, once the artifacts have had time to sync up on the mirrors. Target it to: user@struts.a.o, announcements@struts.a.o and announce@a.o, samples are available at Sample announcements page