How to release Forrest

This documents the steps that the Release Manager (RM) should follow when doing a Forrest release.

Table of contents

1. About this document

Warning:

This document is constantly being developed and some steps will need to be re-arranged. There are some fixme notes that we will review after each release. Do not take this document at face value - always try to visualise the effect of each step.

This documents the steps that the Release Manager (RM) should follow when doing a Forrest release. Note that it might have mistakes - we seem to discover something new each time and some steps might need to happen in a different order. Fine tune these notes for next time. Do some practice runs. Read through these notes often in the lead up to the release to imagine the effect and order of some steps.

There are some steps that other committers, and even developers, can assist with, especially in the areas of getting ready for the release and the final testing. Many of the steps can be done only by the Release Manager.

The lead-up to the release is a good opportunity to build the project community and to draw in new developers. The steps in this process try to maximise that.

FIXME (open):

Some of the complexity in this process is due to site-author docs accompanying the release. That is a good thing because then they have local docs. However it complicates this process because the documentation version numbers, xdocs layout, site.xml navigation, etc. all need to be relevant for packing with the release. Trunk needs to stay that way during testing of RCs. Just before the release announcement, certain things need to be updated to create and publish the website. Then following the release, updated to be ready for the next develoment cycle. Much of that process in unavoidable, but some could be simplified.

2. Who is the Release Manager

The Release Manager is a single person who does the final work of the release process to prepare and sign release packages, co-ordinate testing and voting, uploading to the dist area and ensuring mirrors, and sending the announcement.

The project developers have the role of preparing the product to release quality, testing, fixing issues, and voting for the release candidate. The votes of the PMC will decide whether the product is fit for release.

The RM makes all the decisions regarding the actual preparations and the doing of the release. Many projects have the same release manager for years.

The RM could do the job alone, which enables speedy process. There are not actually many tasks where assistants can help. However it is a good idea to have an assistant so as to pass on the knowledge, to help document the process, and to perceive potential flaws. More than one assistant would probably hinder.

The RM needs to have an incredible amount of time and energy available, especially to prepare the first release candidate and to finalise the release.

The RM should be comfortable with using SVN and be familiar with the ASF hardware, with the distribution mirror system, and with ASF release procedures. Also need to be a quick decision-maker and able to find and solve issues on the spot. It would be nice if others help to solve last minute issues, but don't expect it.

The following notes are terse. If you don't understand, then probably not ready to be RM.

3. Tasks to be done by the project before the release can start

There are a number of things to be done by the project before the release will start. Don't leave these until the last minute. It is not the Release Manager's job to fix bugs nor address blocker issues. The RM job begins after the project is ready to do the release.

- The project updates the Roadmap to schedule the realistic Issues.
- The project made good progress towards fixing the Blockers and applying the outstanding patches.
- The documentation content is ready.
- Plugins have been reviewed and deployed as necessary. Some perhaps need to go through a release process. This should happen independently of the release process.
- Supporting products (e.g. Ant, Xerces) should have been updated well before this stage. Do not attempt such upgrades too close to the release, as it will distract attention from other issues and possibly introduce new problems.
- Add some issues to the Roadmap for the important general issues that need to happen before the release. Some examples from last time: FOR-911, FOR-868, FOR-865, FOR-855, FOR-644.
- Relevant changes are listed in the site-author/status.xml and contributers have been attributed. Other changes are listed in the various plugins/status.xml files.
- Add relevant information to the upgrading notes.
- Major changes are listed in the site-author/status.xml using the "importance" attribute. This will be used to generate the list of top changes for the Release Notes and Announcement text.

  http://localhost:8888/releaseNotes_0.8-dev.html
- Committers (as many as possible) have their up-to-date PGP keys in the KEYS file in Forrest's root directory. Instructions about how to add keys are included in that file. See instructions about how to create and manage PGP keys, and see the notes about encouraging a strong web of trust.

FIXME (open):

Decide the content of the release. Previously we just packed everything in trunk (see the "release-dist" target) and included a built forrest.jar file. Now there is extra stuff that should not be included. See FOR-911.

4. Preparing the project for the release

In this step the Release Manager starts the process to finalise the outstanding blocker issues.

1. Ensure the above preconditions are met.

   If not, then the project is not yet ready for release. Remember that it is not the RM's job to do this.

   If so, then send an email to get the project to decide what to do with the remaining issues. Propose to delay some issues to a future release, encourage people to fix others. See FOR-853. Look at msg02310.html for an example of such a message.
2. Start discussion on Java-Version to use for compiling and testing the release. If necessary, alter the build.compiler.vm value in main/build.xml

5. Preparations for the Release Manager

Particularly the Release Manager, but also anyone assisting, needs to be familiar with standard procedures and Apache terminology. This is crucial for a successful release.

- If you have never done a release before or need to refresh your memory, read all about Apache releases in general at http://www.apache.org/dev/#releases (note that these documents change often). Make sure any assistants have read and understood this as well.
- As discussed above, be sure that you have plenty of time and energy. You will need to be persistent throughout the process.
- Be familiar with the process of signing release packages and generating MD5 and PGP. Some more info is at Signing Releases and our download notes for the Verification.
- Practice signing email in readiness for the announcment.
- Investigate the release procedure notes of other ASF projects.
- Make sure that the network connection is reliable and efficient.
- Install the Java-Version that has been agreed for compiling the release. Do this well ahead of time to avoid delays, and ensure that Forrest works for you.
- Get a printout of this document to scribble notes as you go.
- Use a text file to prepare/record your svn merge commands.
- Browse the dev mail list to see what happened around the previous release. Some mails will be useful to glean words to re-use.

6. Preparing the Release Plan

Prepare the Release Plan to define the corner stones of the coming release. Deciding the dates and timing that suits everybody will be difficult. However, try to be careful about holiday periods and try to find timing that is good to attract the most people to help.

1. Java-Version to test this release
2. Start of code-freeze
3. Start of test-period
4. Vote on release candidate
5. Optional creation of release candidate #2 (when there are bugs)
6. Start of test-period #2
7. Vote on release candidate #2
8. Scheduled release Date

Use the email template propose_release_plan.txt to write and propose your plan, then call for a quick vote on the release plan on the dev list.

Note:

There are various reasons for voting on the Release Plan, e.g. makes people aware that a code-freeze is about to happen; encourage them to get involved with the release; ensure that the date is suitable and people will be around to test and then vote on the actual release. This ensures that it is the PMC as a whole that prepares and issues the release, rather than an individual. Such procedures give us the protection accorded by the ASF being a corporation. See a good discussion in the archives.

7. Preparing the code base

Anyone can help with this phase.

1. Ensure that there are no license issues. The committers and PMC would have been continually monitoring this. There are some tools to assist with scanning for issues. See further information.
2. Ensure that the line-endings and svn:eol-style property are correct for all files. See further information.
3. Ensure that documentation structure and content is ready.

4. Ensure that the documentation is building properly, do 'cd site-author; forrest'. Fix any linking errors.
5. Create a new file, etc/RELEASE-NOTES-0.8.txt, where x.y is the version currently being released. In this file, insert the list of important changes which is obtained by doing: http://localhost:8888/releaseNotes_0.8-dev.txt

FIXME ():
move this step to later, emphasise preparing it now, e.g. in today's there are no "important" for "docs" category.

FIXME ():
Needed to hand tweak some "Link:" ... some have local URLS. Manually removed Table of Contents.

6. Prepare the announcement text. Create a file admin/announcement-x.txt (by 'svn move' the old one) and list the major new features, e.g. locationmap. The admin directory is not included in the release package so editing can continue.
7. Ensure that each plugin that uses the locationmap has its "release version" set to 0.8 or more.

   If a plugin has a locationmap.xml file or uses "lm:" references in its *.xmap, then it is using locationmap. See buildPlugin.
8. Ensure that plugins descriptors in both core and whiteboard are up-to-date.
9. Ensure that all relevant plugins have been deployed to plugins/0.8/ (see other notes at plugins/RELEASE_PROCESS.txt).

FIXME (fso):
Check and integrate plugins/RELEASE_PROCESS.txt as a new document.

8. Preparing your working copy of SVN trunk

FIXME (fso):
We need to discuss order from here on. My idea is to adjust docs before we enter code freeze to save time later. But if the rc fails and release is postponed might need to roll back changes easily and - if possible - roll them forward later. So creating an svn branch for the rc seems to make sense to me. Probably easiest would be to create an rc branch here and co that. wdyt?

In this step you make absolutely sure that your working copy of SVN trunk has no local modifications, or additional files that you have been fiddling with, and especially files that might be hidden by svn:ignore settings.

There are two ways to do this. Either do a complete new svn checkout, or use the 'svn status --no-ignore' command on your existing working copy.

1. In a new empty directory do `'svn co https://svn.apache.org/repos/asf/forrest/trunk'`

Alternative Approach

1. Do 'svn update -r HEAD' to ensure that you are up-to-date.
2. Run 'svn status --no-ignore'
3. Delete any extra files you might have added/changed in your local copy. Delete any "build" directories in plugins, etc. **Extra files must not be packed with the release.** It must be a pristine copy of the current trunk.

9. Preparing the "dist" SVN

The SVN at https://svn.apache.org/repos/asf/forrest/dist/ holds some html files and other files that form the distribution website. This is checked out on the server at /www/www.apache.org/dist/forrest

Note that the actual release artefacts are not stored in SVN.

Ensure that the KEYS file is synchronised with the copy in Forrest svn trunk and commit the changes. Do ssh to the server and 'svn up' in that directory.

10. Preparing docs for next release cycle

In this phase we get the docs ready for the release. You will commit these changes after the code-freeze, but do not publish to the website until after the release.

FIXME (dc):

We definitely need to use automated version number strings where possible. It is too easy to miss some, as we did with 0.8 release. We have a set of core xml entities available.

1. Edit site.xml as follows:
   1. Move all version numbers one line down so that the existing:

```
<versions tab="docs">
   <overview label="Overview" href="versions/index.html"/>
   <v0.8 label="0.8-dev" href="site:index"/>
   <v0.7 label="0.7 (current)" href="site:v0.70//index"/>
   <v0.6 label="0.6" href="site:v0.60//index"/>
 <versions>
```

      becomes:

```
<versions tab="docs">
   <overview label="Overview" href="versions/index.html"/>
   <v0.9 label="0.9-dev" href="site:index"/>
   <v0.8 label="0.8 (current)" href="site:v0.80//index"/>
   <v0.7 label="0.7" href="site:v0.70//index"/>
 <versions>
```

2. Similarly edit tabs.xml
3. Further edit site.xml to configure the menu labels. See an example site.xml from a past release. Basically do this ...
   1. Add section <v0.90> with minimal content above the <v0.80> section.
   2. Edit the label@ and description@ attributes for each of the three sections. <v0.90> and <v0.80> and <v0.70>
   3. Remove the label@ attribute for the <v0.70> section. This prevents it from appearing in the linkmap ToC.
   4. Remove the whole section <v0.60> of the past docs.
4. Edit version numbers in xdocs/versions/index.xml
5. Edit version numbers in xdocs/pluginDocs/index.xml
6. Edit version numbers in MOTD section of site-author/skinconf.xml
7. Edit version numbers in main/fresh-site/..../site.xml and tabs.xml
8. Remove the past versions (0.6) docs directory by doing 'svn rm site-author/content/xdocs/docs_0_60

9. Edit site-author/conf/cli.xconf where it excludes old docs from being generated (a trick to speed up). Adjust the version numbers.
10. Change the "fixfor" attribute to the next version for the "project.issues-rss-url" RSS feed in the `site-author/forrest.properties` file. Find the version number via the "Road Map" link at our Jira front page. Verify localhost:8888/forrest-issues.html
11. Edit site-author/status.xml to Remove the "-dev" from the current <release> tag, and set the anticipated release date. Remove the "-dev" from the text intro.
12. Validate status.xml (e.g. with xmllint).
13. Create a new directory as a stub for the next section of development docs. Create a placeholder index page and upgrading notes. See a past release for example. Do this ...

```
svn mkdir docs_0_90
svn copy docs_0_80/index.xml docs_0_90
svn copy docs_0_80/upgrading.xml docs_0_90
... edit those docs to suit. Emphasise developers and encourage svn.
```

14. Temporarily remove the "dev" note from upgrading_xy.xml
15. Edit the Forrest home page in the "News and events" section and add a text like:

```
Apache Forrest 0.8 was released on 2007-04-18.
[### Add some important new features ###]
```

16. Ensure that the documentation is building properly, do 'cd site-author; forrest'. Fix any linking errors. Review changes.html to ensure section numbering.
17. Edit the site-author/content/doap.xml to add the new release date.
18. Edit the forrest/site-author/content/xdocs/mirrors.html and adjust all version-specific content.
19. Commit all of the above changes after the code-freeze has started in the next section.

FIXME ():

Not sure what will happen with the forrestbot on our zone during this phase. If the steps in this Release Process are correct, then such users of live trunk should get a smooth upgrade. We will see.

11. Building the release candidate packages

In this phase the Release Manager builds the release candidate to be tested.

Note:

You can practice the following steps (as far as creating the branch) without committing anything even before code-freeze. This ensures a good release candidate.

1. Announce that the code-freeze has now commenced. Use the template announce_code_freeze.txt to send email to dev list. The meaning of "code-freeze" is defined in that template.
2. Update your SVN working copy to include any last minute changes.
3. Do any final work, such as license header checks and xml code cleanup.
4. Ensure that your Java version is the lowest specified of our supported versions.

Note:

Set the environment variable JAVA_HOME to the path of the Java version. Note for Windows: If you change the setting in system properties, you need to logout and login again for the changes to become effective.

5. Update the version numbers at various places:
    1. Edit main/build.xml and remove the "-dev" text around line 48:

```
 <property name="forrest.version" value="0.8-dev"/>
to:
 <property name="forrest.version" value="0.8"/>
```

2. Similarly in main/forrest.build.xml around line 28.
3. Edit plugins/build.xml to increase the docs version number to the next major release. Around line 23:

```
 <property name="forrest.version" value="0.8"/>
to:
 <property name="forrest.version" value="0.9"/>
```

Note:

This is deliberately a major version up. It is assumed that plugins will be developed against the next version of Forrest. Individual plugins can override this property in their own build files.

FIXME ():

There are probably other areas which have version numbers. How can we improve this? Possibly with XML Entities, possibly with Ant properties.

6. Run the following quick tests from the command line of your system to ensure that all is well:
   - Change to the main directory and run `build test`. The build should conclude without errors from both seed builds.
   - Change to the site-author directory and run 'forrest'. The docs should build without errors.

   If there are any problems, focus on problems that prevent building and invite other commiters to help you solve the problems.

Note:

It is not your job to fix bugs and code freeze should not commence with a broken trunk.

   If there are bugs with the build system that cannot be easily fixed, then call a halt to the release process and start a discussion on rescheduling options on the dev-list with the template rc_did_not_build_what_now.txt
7. Remove the build directories from core and plugins. Do `svn st --no-ignore` in the root directory of your release candidate directory to be sure that all files created by the test build have been removed and no other files have been changed. The status command should report no changes.
8. Edit 3 files in tools/forrestbar to update the version number to match the new release:

```
xpi/install.rdf, line 24: <em:version>0.8<em:version>

xpi/install.js, line 19: var err = initInstall("ForrestBar", "forrestbar",
"0.8");

xpi/chrome/content/contents.rdf, line 25: chrome:displayName="ForrestBar 0.8"/>
```

9. Build the forrestbar and replace site-author/content/xdocs/tools/forrestbar.xpi
10. Update the etc/RELEASE-NOTES-0.8.txt as described above.
11. Commit all of the above changes.
12. Take note of the SVN revision number of your trunk by running `svn info` from the command line in the Release Candidate's root dir and look at the "Last Changed Rev: ######". This will be

used later for the svn log message when the branch is created. Also it is helpful for ensuring that no new commits have been made, i.e. people forgetting the code freeze. From here on watch the svn@ list.

13. Cleanup your working copy to remove any extra files.

```
cd $FORREST_HOME
find . -name build | xargs rm -rf
svn status --no-ignore
```

14. Now we will build the release candidate packages for Windows and Unix.
    - Change to directory $FORREST_HOME/main and run `'build release-dist'` to generate the release candidate packages.
15. Unpack and test the relevant archive in a fresh new directory. No point getting people to test if it is broken. You need this for your own testing and vote anyway. Be sure to set FORREST_HOME and PATH properly for this release candidate location i.e. ensure that you are not using your trunk working copy.
16. Sign the Release Candidate packages and create the *.asc and *.md5 files.

    Here is one example when using gpg and openssl from the command line.

```
gpg --output apache-forrest-0.8.tar.gz.asc \
  --detach-sig --armor apache-forrest-0.8.tar.gz

gpg --verify apache-forrest-0.8.tar.gz.asc \
  apache-forrest-0.8.tar.gz
```

    ... should say "Good signature from ..."

```
openssl dgst -md5 -out apache-forrest-0.8.tar.gz.md5 \
apache-forrest-0.8.tar.gz

md5sum apache-forrest-0.8.tar.gz
```

    ... output should match that of the md5 file.

17. Create a maintenance branch in SVN. This command can be run from anywhere because it uses full URLs.

```
svn copy -r ##### -m "Create the 0.8 release branch from r#####" \
    https://svn.apache.org/repos/asf/forrest/trunk \
    https://svn.apache.org/repos/asf/forrest/branches/forrest_#xy#_branch

where
  '#xy#' is a compact form of the version (e.g. 04, 041, 05).
  '#####' is the SVN revision number that the branch was created from,
  which was the revision that the release candidate packages were generated
from.
  (Remember that you recorded this number earlier.)
```

    See http://svn.apache.org/repos/asf/forrest/branches/ for examples of past branches, e.g. forrest_07_branch

12. Testing the release candidate and voting

Get Forrest developers to test the actual packges on various platforms.

1. Upload the release candidate packages and signatures to your committer's webspace.
2. Use template test_and_vote_on_rel_cand.txt for an email to the dev-list asking all developers to

test and vote. Don't forget to modify the template, to specify the md5sums etc. People need to tell you the md5sum to be sure that they have tested and voted on the correct release candidate. This is especially important if there has been more than one release candidate.

3. During this testing phase, the Release Manager should:
   - Make sure that people are reporting that the packages unpack on different systems without problems.
   - Make sure that people have followed the actual user instructions in the Forrest distribution at README.txt and index.html
   - Encourage people to build some difficult sites.
   - Remind people about how long remains for testing.
4. If substantial problems are revealed (not just minor glitches) then co-ordinate their fixing by other developers. Doing the fixing of issues is not the Release Manager's job. Deciding what constitutes a "substantial" issue is entirely the RM's call. Remember that there is still a code freeze, so don't let people be tempted to fix other minor stuff or add some pet new feature that they have neglected. That can easily introduce new problems.
5. If necessary start again with Building the packages and build another release candidate. Remember to do 'svn update' first and to record the new SVN revision number.
6. Tally the votes and report the result to the dev list.

Note:
It is vitally important that people review and vote against the actual final release packages (and not just against their SVN working copy).

13. Finalizing the release

When a good release candidate has been achieved and affirmed by the vote, we'll finalize the release.

1. Remove the release candidate packages from your public_html directory.
2. If there have been changes to the trunk since the branch was created, then merge trunk to branch. Remember to use a proper commit message which includes the revision number used for the merge (see the SVN Book).

FIXME (fso):
What is the purpose of this step? It doesn't seem to be right because trunk may already contain parts of the next version. What we should do is do all fixing of RC-problems in the rc-branch (same as changing docs) then, on release, merge branch back into trunk to integrate fixes and doc-changes back into trunk. wdyt?

FIXME (dc):
It should not contain new functionality because we are in a code freeze

3. Tag SVN by doing:

```
svn copy -m "Create tag forrest_###xy###_release from release branch" \
  https://svn.apache.org/repos/asf/forrest/branches/forrest_###xy###_branch \
  https://svn.apache.org/repos/asf/forrest/tags/forrest_###xy###_release
```

where '###xy###' is a compact (without the dots) form of the version number (e.g. 04, 041, 05).

See http://svn.apache.org/repos/asf/forrest/tags/ for examples of past tags, e.g. forrest_08_release

FIXME (fso):
If we change procedure to create an rc-branch this will become a merge changes from trunk then rename rc-branch to final release branch. right?

4. Tag the "site" SVN by doing:

```
svn copy -m "Create tag forrest_###xy###_release_site from site" \
  https://svn.apache.org/repos/asf/forrest/site \
  https://svn.apache.org/repos/asf/forrest/tags/forrest_###xy###_release_site
```

   where '###xy###' is a compact (without the dots) form of the version number (e.g. 04, 041, 05).

5. Announce the end of the code-freeze by sending the email-template announce_end_of_code_freeze.txt to the dev list.

14. Prepare the site-author docs for next development phase

Note:

Before doing anything, skip to the next section to get the upload commenced.

While waiting for the mirrors to catch up, you will copy the docs set for the next development, then call off the code-freeze. Do this ...

- Create a copy of current dev-docs in trunk for the next development phase ...

```
cd site-author/content/xdocs
svn copy docs_0_80 docs_0_90
svn copy pluginDocs/plugins_0_80 plugins_0_90
```

- Edit site.xml and add a copy of the most current versioned section (e.g. <v0.80>) above it. Increment the first decimal of the sections name to reflect the next planned release (e.g. <v0.90>).
- Edit site-author/status.xml to add a new `<release>` section above it for development on the next version. Add one placeholder action for the next "upgrading" doc. Do the same in the release branch for a possible x.y.1

```
  <release version="0.9-dev" date="not yet released">
    <introduction>
    ...
  </release>
  <release version="0.8" date="2007-04-18">
    <action
    ...
```

- Return the "dev" note to upgrading_xy.xml and add some of the orginal general points that still apply.
- Edit site-author/conf/cli.xconf to exclude the old docs again.
- Add new plugins directories to the "forrest/site" SVN ...

```
svn mkdir pluginDocs/plugins_0_90
svn mkdir plugins/0.9
```

- Edit main/build.xml, increment the version and add a -dev tag: around line 45: <property name="version" value="0.9-dev"/>
- Edit main/forrest.build.xml and update the version: around line 32:

```
<property name="version" value="0.9-dev"/>
```

- Edit version numbers in plugins/build.xml
- Edit version numbers in tools/forrestbar
- Update the .htaccess file to redirect /docs/dev/ to the next version, and do other changes noted in the .htaccess file. See site-author/content/.htaccess

FIXME (fso):

Need to go through .htaccess and clean up.

- Update the release version number and release date in xdocs/index.xml and site-author/content/doap.xml
- Commit all of the above changes.
- Send email to the dev list to remind people about the new docs set docs_0_90 and that we don't update docs_0_60 set. Also remind about the new release branch of svn. See example email from the 0.8 release.

15. Upload and announcement

In this phase we'll upload the new Release, wait for it to be available on most mirror sites, publish the new website, then announce the release. The reason for waiting, is because when you send the announcement, most of the mirrors need to be up-to-date or your audience will grumble.

Note:
During this phase there is a lot of waiting. While things are happening you can be doing some of the other tasks in this Upload section and in the Cleanup section.

1. Use scp to upload the release: the *.tar.gz, the *.zip, the *.asc and *.md5 files, and the RELEASE-NOTES-0.8.txt to people.apache.org at /www/www.apache.org/ dist/forrest/ directory.

   Ensure correct file permissions by executing `'chgrp forrest *'` then `'chmod g+w *'` in that directory. Each PMC member has a server account and belongs to the forrest group.

   The process is documented at http://www.apache.org/~bodewig/mirror.html and http://www.apache.org/dev/#releases

   Leave the previous dist there as well as the new one, until after the announcement (because mirrors.html cannot be updated until most mirrors have received the release).

   Note:
   The other files there (HEADER.html README.html LICENSE.txt KEYS) are all automatically updated from the SVN:forrest/dist/ repository (explained above).

2. Wait for the various mirrors to pick up the new files.

   For some mirrors, this takes only a few hours. However others are slow. How long to wait is a tradeoff, e.g. 8 hours has been the norm.

   See Status of mirrors. It is a very useful service and fun to watch. See the "age histogram"i near the bottom.

   Take note of the time that the eu.apache.org mirror is updated, then compare each "mirror age" to that.

   When you see that a good proportion of the mirrors have received the release, then update the website, then send the announcement.
3. Ensure that docs are *not* being excluded via site-author/conf/cli.xconf as we need to re-build the whole site.
4. Before re-building and deploying the new website, check the time. Remember that an 'svn up' happens our on server just after the hour, then a bit later the rsync happens and the site is live about 20 minutes past. This site deployment will take some time, mostly because you are actually doing a fresh installation of a local forrestbot (it does 'svn co forrest/site').
5. Re-build and publish the Forrest website as normal. Be sure to use the new version of trunk for

building the docs. Refer to Publishing Forrest Documentation for details. Beware, there is a forrestbot bug whereby with a brand new forrest installation, the first deploy only does the added files. So deploy again.

6. Test the new Forrest website and the redirects f.a.o/docs/ and f.a.o/docs/dev/ and the download page.
7. Update the xml.apache.org website (Forrest is part of the Apache XML federation of projects). Edit xml-site/src/documentation/content/xdocs/news.xml and record the announcement, and then commit the new HTML to xml-site/targets/forrest Note that they use forrest-0.7 to build their website. See http://xml.apache.org/guidelines.html#website-top
8. Send announce_release.txt as email to the Forrest user and dev list and the ASF-wide announce lists. Sign the email (e.g. PGP).

   See previous announcements for examples:

   - 0.2
   - 0.3
   - 0.4
   - 0.5
   - 0.5.1
   - 0.6
   - 0.7
   - 0.8
9. Do the Freshmeat announcement: http://freshmeat.net/projects/forrest/
10. Update the release information at our Wikipedia page.

16. Cleanup

1. Remove the old generated docs from SVN forrest/site/docs_0_60 which removes them from the website.
2. Remove old dist files from the /www/www.apache.org/dist/forrest/ directory. They have already been automatically archived at archive.apache.org/dist/forrest/
3. Tidy up any "site:v0.7" references. These will refer to old documentation that will be removed at the next release. Especially status.xml file will have such. Generalise as many as possible.
4. Do some Jira administration (need to be in the jira-administrators group)
   1. Tweak the "release" versions via "admin" interface at our Jira. Do this ...
   2. Re-name the VersionIds using "Manage Versions" then "Edit details": e.g. 0.8-dev is renamed to 0.8 and 0.9 becomes 0.9-dev
   3. Mark 0.8 as released using "Manage Versions". Review the Issues for the old version.
5. Deploy the Forrest site again. It will automatically rebuild the Open Issues page.

17. Conclusion

All done!

Or perhaps not.. if you think of anything, please refine these instructions.