

# best practices - site management

[JIRA reference](#)

## TODO

### Additional Thoughts

Basically a TODO list

### Transient and Project docs in a single module project

How do we deal with the separation of transient (developer) information from project information in a single module (eg jakarta-commons-foo) scenario?

Perhaps in this case, there's no harm in including the development reports for a released version in the site, or we use a profile to turn them off when we deploy to the live site?

### Separate deployment of snapshots

What about creating separate deployment locations for snapshot sites?

Could be done with a profile, but to align with `<distributionManagement>` `<snapshotRepository>` maybe we should have `<distributionManagement>` `<snapshotSite>`.

This would allow us to just drop the reports straight into the site like we do now without worrying about the layout issues, and turn it on or off depending on the deployment target. The snapshot one would be continuously spat out from CI, and users could go to <http://maven.apache.org/2.1-SNAPSHOT> to see the site with reports intact and totally up to date, while maven.apache.org remains the last release copy.

John Allen reports that this is currently what the site:stage goal is designed for and this should be reviewed before the 2.0 release. He also notes it would be worth supporting the snapshot site directly in the POM. This was always intended as the functionality was present in m1, but hasn't been incorporated yet. Using the standard snapshot + release notation for a site instead of stage/live is a good idea.

### Report categorisation

Is there a way to designate reports that are for users and part of the site (mailing list, scm, plugin reference) from development reports (junit, cobertura, even javadoc)? Note that these have nothing to do with their rate of change. Mailing lists and SCMs should be part of the "current" site as if they move, that details needs to change and is not tied to a release. However plugin references are release bound, but still documental, and are versioned, like Javadoc. Reports like junit are generally immediately published, but tied to a codebase rather than the site.