Branching model

This page describes the branching model we apply for development at Magnolia.

The model

The model is similar as described in a blog post.

Differences

- We'll not use hotfix branches
- We'll not use development/integration branches (for now -> see table under OpenIssues)

Master

- Rule: revert immediately as soon as it fails
- no direct commits except minor enhancement (fixing typos & formatting etc.)
    - make sure to inform in M5 Review chat

Feature branches

- on central repo
- remove them after the final merge
- list of feature branches
    - --> things we are actively working on or are waiting
- frequently merge from master to feature branches to reduce last-minute conflicts

Maintenance

- no changes
- no hot fix branches (for the moment)

Merging (feature-branch -> master)

- only after review
- we use *git merge --no-ff* instead
    - no fastforward - avoids losing information about the historical existence of a feature branch + groups together all commits of a feature

(other option would be git merge --squash when merging from feature branch to master -> one consolidated commit with meaningful message for a feature)

Not using one of the above will result in situations where it might be very hard to figure out what commit's belong together (e.g. really bad if u have to revert a feature)

Links

By the way: feature branching vs. continuous integration has heavily been debated - mainly because of Martin Fowler's vote to not go for feature branches. Here's a bunch of links around it:

- a successful git branching model (main influence for our approach)
- a tidy git workflow (similar to above)
- continues delivery (Martin Fowler et al. voting to not go for feature branches)
- minutes from our first meeting