Contributing code to NetarchiveSuite

Describes how to contribute code to the NetarchiveSuite git repository on Github. Code can be added to the NetarchiveSuite repository in two ways, either by pushing directly to the repository or by creating pull requests from a forked repository.

- Committing directly to the NetarchiveSuite repository
  - Workflow
  - Removing branches
- Contributing patches through pull requests

Committing directly to the NetarchiveSuite repository

This is used for core developers to add code in the NetarchiveSuite platform and requires the contributor to have commit permissions on the NetarchiveSuite project. The includes the DK NetarchiveSuite developers.

Workflow

Inspired by the Git-flow branch model we aim to develop all nontrivial code changes on feature branches with merges to master after a feature has been finished. The workflow here is:

1. Start the development on a feature by creating a new local feature branch. The standard is to name this according to the related JIRA issue.

   ```bash
   git checkout -b NAS-XXXX master
   ```

   - Optional: If the actual work is split between several subtasks, a further subtask branch from the parent feature branch should be created. Eg. for subtask NAS-YYYY

     ```bash
     git checkout -b NAS-YYYYY NAS-XXXX
     ```

2. Do the development with frequent commits of our code changes. The commit message should be of the format:

   ```text
   NAS-XXXX: ${short description}
   NAS-XXXX ${issue summary}: ${long description}
   ```

   Only the first line is mandatory and should be no longer than 50. Following lines should be no longer that 72 chars log.

   - Optional: Push the changes to a corresponding remote branch:

     ```bash
     git push origin NAS-XXXX
     ```

   - Optional: Merge master commits into the feature branch. This might be used to avoid having to merge a lots of changes from at the end of the feature development, and instead split these into smaller merges.

     ```bash
     git checkout master
     git pull
     git checkout NAS-XXXX
     git merge master
     ```

     Resolve any merge conflict.

     - Optional: Check that build still works:

       ```bash
       mvn clean install
       ```

     - Performing a review of the changes made in the feature development.
     - Merge the latest changes on the master branch into the feature branch.

3. Prepare for merge to master by:

   - Check that the build still works:

     ```bash
     mvn clean install
     ```

4. Merge to master

   - Merge to master:
4. git checkout master
   git merge NAS-XXX

   (we are still discussing whether to use "--no-ff" or not)
5. Check that build still works.
   Finally push the new feature code to Github master:

   git push

All done 🌻

**Removing branches**

Branches may be removed after the development of a feature has finishes.
An effort should also be made at the end of a development cycle/release to clean out old branches.

**Contributing patches through pull requests**

Used mainly by developers wishing to contribute code occasionally to the NetarchiveSuite repo.
Please see Github documentation on how do this.