Grouper developers coding standards

Wiki Home	Grouper Release Announcements	Grouper Guides	Grouper Deployment Guide	Community Contributions	Internal Developer Resources

We request that Grouper developers adhere to the standards below and also review the page on the Development Environment:

Source control

- Master is the dev branch. If a new release was just cut, master might be the current release.
- Once work is needed on the next release a branch will be created.
- Branch names **must** follow the convention: GROUPER_2_3_0
- Commits on released branches should only be made when a fix is ready for the next build number (2.5+) or patch (2.4)
    - (2.4 only) After patch is made, tag the branch with this convention: grouper_X_Y_Z-aWW-uV-wU-pT e.g. grouper_2_4_0-a21-u9-w2-p2

```
(master) $ git tag grouper_2_4_0-a25-u11-w2-p2
(master) $ git push --tags
```

- When committing (pull request or direct commit), you **should** have one JIRA associated with the commit.

```
GRP-1279: grouper installer should set sql logging to on so we can see progress of DDL on install or
upgrade

If there are more commits on the same issue, number them:

GRP-1279: grouper installer should set sql logging to on so we can see progress of DDL on install or
upgrade (commit 2)
GRP-1279: grouper installer should set sql logging to on so we can see progress of DDL on install or
upgrade (commit 3)
```

- Every commit **must** have a unique comment (from all other commits). This is so we can go back and see what hasnt been merged
- Every commit in a non-master branch that should be merged forward **must** be cherry picked by the developer into all newer branches including master at the time of the commit
- Pull requests **must** be on a recent pull of the branch the request is destined for so there are no conflicts
- (2.4 only) All new/edited junit tests should be included in patches (CH I think we need to change the installer to make this happen, its not that simple...)
- file names should be unique across grouper to make patching easier (e.g. dont have two classes with same name in different package, dont have two JSPs with same name in different dir, etc). (even across projects, and being in a different package or folder doesnt count)

Git Tips

Github SSH keys

After August 2021, Github will stop allowing password logins for commits. By then, developers will need to generate an SSH key to authenticate for commits. To generate the key, follow this page. You will be generating a key called *id_ed25519* and *id_ed25519.pub*. Then, change the repository URL in the project's git config (this is less well documented). On the project's homepage, click on the Code button to bring up the clone urls. Change from HTTPS to SSH and copy the URL (i.e., git@github.com:Internet2/grouper.git). Then edit your local .git/config file and change the project URL to it. After that, you will be asked for the ssh passphrase for commits instead of username and password.

Branches

Developers must create a separate (local) branch for a specific feature if a commit is not for an immediate patch. That dev local branch should be named something like: GRP-1966-ui-visualization, During development, multiple interim commits can be made. Several options to merged back:

1. In main branch, merge from feature branch (what chris generally does with Eclipse)
2. In main branch, cherry pick commits from feature branch
3. (Chad advice) Squash into a single commit for the main branch. Before merging to the main branch, switch back to the main branch, do a git pull to get the latest commits made by others. Then, switch back to the feature branch and do a rebase from the main branch. This makes your code apply to the main branch HEAD, rather than the historical commit where your branch started. The rebase changes the start of your branch to the main branch HEAD, and then replays all of your feature commits on top of it. If there are any merge conflicts, you will discover them now instead of later when trying to merge your feature. You can also test this updated code, to ensure it works before committing it to the main branch.

```
# Get the latest main branch code before starting the feature
(master) $ git pull
(master) $ git checkout -b grp-1279
(grp-1279) #do work, make commits
(grp-1279) #squash multiple commits together to a single new commit; e.g. here merge
(grp-1279) #the last two commits by changing all but the first commit
(grp-1279) #from "pick" to either "squash" (retain message) or "fixup" (ignore message)
(grp-1279) $ git rebase -i HEAD~~
(grp-1279) $ git checkout master
(master) #get any new commits
(master) $ git pull
(master) #if any new commits, go back to the feature branch and rebase
(master) $ git checkout grp-1279
(grp-1279) $ git rebase master
(grp-1279) #Retest. Fix any conflicts that occurred during the rebase. Then continue below
(grp-1279) $ git checkout master
(master) #merge should be a clean fast-forward merge
(master) $ git merge grp-1279
(master) $ git push
(master) #the branch is merged, can delete it now
(master) $ git branch -d grp-1279
```

Junit

- If its possible you should have junit tests for your code
- If you add a test method (starts with "test" and is instance method with no args), then you are good
- If you add a test class, make sure it is included in the package harness, make sure package harness has unique name (dont name it AllTests.java, name it AllProvisioningTests.java or something unique)
- If you add a test harness for new package, make sure it is in the parent package harness
- Good practice is to check out from git after merging and run all tests, or watch for CI message that tests pass

IDE settings

- Install web IDE
- No spell check
- Compare / patch - ignore whitepsace
- Do not enable folding
- 2 spaces for tabs
- Install egit (or git plugin), if not there already
- Content assist -> uncheck single completions automatically
- Web browser, use external
- code templates - catch block has nothing (no printStackTrace()
- git
    - include untracked file
    - auto stage deleted files
    - auto crlf
- java errors/warnings
    - unqualified access to instance: warn
    - serializable class without id: ignore
    - possible boolean assignment: warn
    - field declaration hides another field: warn
    - local var hides another field (include constructor, setters
    - unnecessary else: warning
    - unnecessary cast or instanceof: warning
    - Unnecessary code -> Unused private member -> warning
    - Generic Types --> Unchecked generic type operation -> warning
    - missing @Override: warning
    - missing @Deprecated: warning
    - warn if not all enums used in switch (tell people)

    - missing javadoc (private+): warning
- Javadoc missing should be warnings
- Increase timeouts of tomcat
- Maven - errors / warnings - ignore plugin execution not covered by lifecycle configuration
- Skip breakpoints on run to line
- Always launch previously launched application
- Disable content insertions except enter
- Dont ignore .* resources in package explorer
- Dont escape text when pasting string literals
- Java - editor - content assist - advanced - Java proposals and content assist
- Console Output lines 1000000

Intellij

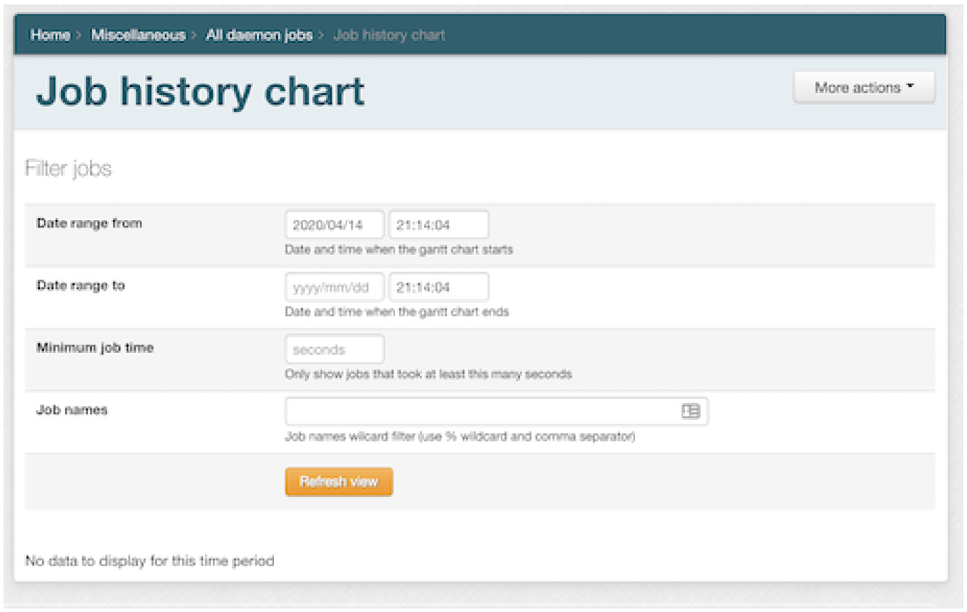
- Collapse packages
- Convert to maven project

Java

- There is are eclipse formatting and prefs files in grouper/misc/eclipse
- There are xml configurations for checkstyle under grouper-parent/src/checkstyle. You can add the checkstyle plugin for either Eclipse (more instructions here) or IntelliJ, and it will scan and warn of style issues, either on demand or in the background. Use "grouper-parent/src/checkstyle/checkstyle.xml" for new or recent files. The other configuration "grouper-parent/src/checkstyle/checkstyle-legacy.xml" only checks for the license and a few minor issues, and is more geared toward older files that have more style problems but are unlikely to be fixed. To use Maven to scan all files in a project, use `mvn checkstyle:checkstyle` for legacy checks, or `mvn checkstyle:checkstyle -Dcheckstyle.config.location=src/checkstyle/checkstyle.xml` to scan with the more rigorous checks. This will work from any project directory, or in grouper-parent to scan all projects.
- Never use tabs, only 2 spaces
- Use curlies always e.g. for loops and if statements
- Try to be explicit in naming classes and variables, try not to abbreviate
    - If a field or variable can be named the same as its type it might make things more clear or easier to read later.
    - e.g. if the type is "GrouperAttestationAttribute", maybe name it "grouperAttestationAttribute" instead of just "attribute"
- Comment a lot
- Always have javadoc
- For new code don't commit if there are warnings in eclipse
- Generally we use unchecked exceptions
- Keep things backwards compatible generally, deprecate if needed

UI

- Try not to use javascript, use the ajax API to write logic in Java and views in JSP
- Externalize all text (even text in placeholders, from java, error messages, text from javascript if applicable)
    - Only share externalized text keys if you expect it to always be tied together
- We have a look and feel using tables and alternating rows and bold labels on left and more info before the form elements. We should try to use that when possible



Home > Miscellaneous > All daemon jobs > Job history chart

Job history chart

More actions

Filter jobs

Date range from	2020/04/14 21:14:04 Date and time when the gantt chart starts
Date range to	yyyy/mm/dd 21:14:04 Date and time when the gantt chart ends
Minimum job time	seconds Only show jobs that took at least this many seconds
Job names	Job names wilcard filter (use % wildcard and comma separator)
	Refresh view

No data to display for this time period

- Be consistent with accessibility using "for" tags with labels and aria tags when applicable
- Pages should have breadcrumbs underneath the parent page so it can be navigated back
- Generally navigation should be consistent with "more actions" drop downs
- Security of who is allowed to do what needs to be checked when buttons are displayed, and on the server side when actions are processed (ajax included)
- Follow sentence case unless a proper noun. IE do not capitalize all words in label or title unless proper noun.

JIRA

- Record issues in JIRA https://bugs.internet2.edu/jira/projects/GRP

Cross database

- If doing a count, dont have order by. do the order by in the real query. due to hsql: GRP-2370 - Getting issue details... STATUS

Load test with data

```
groovy:000> edu.internet2.middleware.grouper.helper.LoadData.main()
Wed Dec 22 13:08:05 EST 2021, stem: duke:siss:courses:SUBJECT1:100:10, stems: 6, Groups: 0, Memberships: 28
Wed Dec 22 13:08:06 EST 2021, stem: duke:siss:courses:SUBJECT1:100:11, stems: 7, Groups: 0, Memberships: 56
Wed Dec 22 13:08:06 EST 2021, stem: duke:siss:courses:SUBJECT1:100:12, stems: 8, Groups: 0, Memberships: 84
Wed Dec 22 13:08:07 EST 2021, stem: duke:siss:courses:SUBJECT1:100:13, stems: 9, Groups: 0, Memberships: 112
```

Grouper standards

Refer to objects

- By the system name
- Make sure renames and dependencies work

See also

- Community Contributions Guidelines
- Grouper Developers Environment

- When starting a new wiki page see this