Code Review

Tutorial

The 2012-03-14 OpenMRS University focused on code reviews. The presentation, at http://www.youtube.com/watch?v=lVVlDIElkuA , gives a good introduction to the process.

Overview

A code review is a process by which developers examine source code in order to discover bugs, scrutinize coding conventions, and look for potential bottlenecks and resource leakage. The intent of a code review is to catch bugs/issues/defects before the offending code is deployed to a production environment and to transfer knowledge of implementation details to the rest of the team.

The pieces of code that need review are governed by the tickets in JIRA that have status of "Code Review (Initial)" or "Code Review (Pre-Commit)" or "Code Review (Post Commit)". To enter the queue, move your code through the workflow to get that status.

See open pull requests in Github.

Starting the code review process. (Preparing your code to be reviewed)

If you do not have a ticket yet, create one, attach your patch or link to your branch, then move the ticket through the workflow.

- If using Git/Github and you **don't** have admin status:
    1. issue a pull request with a full link to your ticket in the description.
    2. click the "Request Review" option on the ticket to move into the "Code Review (pre commit)" state.
    3. Add a comment linking to your pull request by its url
- If using Git/Github and you **do** have admin status:
    1. Commit/push to your fork, issue a pull request, and merge the request. (put a full link to your ticket in the description)
    2. Click "Committed Code" in the ticket to move into the "Code Review (post commit)"

Now you can either ask someone to review the ticket, or you wait for someone in the Community Development Swim Lane to review it.

Performing a code review (Reviewing another person's code)

Finding a ticket

- Look at tickets in the "Code Review (Initial)" or "Code Review (Pre-Commit)" or "Code Review (Post Commit)" state.
- Or look at open pull requests in Github.

Doing the code review

Go through the code SLOWLY with a mind towards the Goals of Code Review (see below) and the Code Review Checklist

Add a comment for each line that needs attention. Be as descriptive and HELPFUL as possible in the comment.

Don't worry that you might not find anything to comment on, or that you don't know enough about every area of the code. There will usually be something to say about almost every commit; even where you don't find anything to question, you may find something to praise. The important thing is to make it clear to every committer that what they do is seen and understood, that attention is being paid. (Copied from http://producingoss.com/en/setting-tone.html#code-review)

Finishing a code review

If changes need to be made before merging, then on the associated JIRA issue click the Rework Needed button.

Otherwise, merge the pull request (see Git Merging) and on the associated JIRA issue click on the Committed Code button. At this point you need to either (a) close the ticket in JIRA, via Approve/Close, or (b) comment on the JIRA ticket at-mentioning the person you expect to take the next action (maybe the dev working on the ticket to do followup fixes, maybe another person to do code review, maybe a BA/QA to do UI testing.

Goals of Code Review

There are many reasons to review code. Each code review should aim to achieve one or more of these goals; however, not all code reviews need to aim for all goals.

- Finding bugs
    - Bugs found in code review require much less effort to find & fix than bugs found in QA/testing.
- Coding style. Ensure that our Java Conventions are followed.
- Improving code quality
- Teaching best practices

- Code consistency (can't tell the author from the code)
- Learning code
- Efficiency (getting pull requests reviewed quickly)
- Ensuring that the pull request guidelines are followed.

Tips for Code Review

- 60 minutes or less – never review for more than an hour at a time.
    - Reviewer effectiveness drops precipitously after one hour.
- "Lines Of Code (LOC) for a review should be under 200, not to exceed 400."
- Slow down! "The longer you take in review, the more defects you'll find."
    - Inspection rate < 250 lines/hour is good, < 400 lines/hour is okay, above 400 lines/hour will miss many defects
- "The more defects the better!"

Resources

- Best Kept Secrets of Peer Code ReviewBest Practices for Code Review
- The commandments of navigating code reviews
- Best Practice for Peer Code Review (PDF)
- Things Everyone Should Do: Code Review
- Code Review as Relationship Builders
- How Should Code Reviews Be Carried Out
- Code Review Metrics
- Code Review Tips
- Reflections on the Quality of Open Source Software
- http://www.perlmonks.org/?node_id=744932
- http://goodmath.scientopia.org/2011/07/06/things-everyone-should-do-code-review/
- https://talk.openmrs.org/t/maintainer-happiness-github-saved-replies/14927

FAQ

1. Where is the asynchronous code review application?
    a. https://github.com/OpenMRS
2. What should I do if I find something wrong during a personal or asynchronous code review?
    a. Assuming that this review is happening in Github, add a comment to the review at that line.
    b. We need to make sure that this is a constructive process. The reviewer should not slam or even poke fun at the developer. Code reviews are meant to be both constructive and instructive. Not all of us have the same level of understanding of the code, so please be kind. In addition, the developer should not be defensive about his/her code. If the developer disagrees with the reviewer (which is totally reasonable), that developer should state their case and ask other developers for their opinion.
    c. Clarify it. Send email to developer asking for clarification
3. How do I request a code review?
    a. Add a link to your pull request to the ticket your are working on. Set the status of the ticket to "Code Review (Pre-Commit)". (A ticket that's ready for work must be assigned to you first, so first make sure you click "Claim Ticket" to assign it to yourself.)
4. What types of issues are we looking to capture during code review?
    a. See the Code Review Checklist.
5. How much code should be reviewed in a single code review session?
    a. Ideally, less than 90 minutes without a break and less than 300 lines of code in one sitting