Collaborating on HTML Form Entry Using Git

Overview

The HTML Form Entry module is hosted at github:

https://github.com/OpenMRS/openmrs-module-htmlformentry

Push access to this master branch of the module is limited. Others are welcomed and encouraged to contribute to the module, but in order to maintain stability of the code base, we request that you use the following process to submit your changes to the module.

Prerequisites

Before beginning work on the module, you should have git installed, and also have a github account and SSH key configured. Instructions on how to do this can be found here.

Working on the Module

(Note that OpenMRS is new to developing using Git, so suggestions of how improve this process are welcome!)

Choose a ticket

The first thing to do before working on any feature is make sure that the ticket for the feature is in the **Ready for Work** state, and that you have claimed the ticket. **Don't start working on a ticket without claiming it**, or someone else may needlessly duplicate your work.

Check out and submit

The standard way for developers to make changes to the module is to create their own github forks of the module and then push their changes up to their own remote repositories; from there, the project lead can review the changes and determine whether or not to pull them into the main line.

Fork and checkout

When working on the module, you should first fork the main branch of the openmrs-module-htmlformentry. Follow the instructions "Fork a Repo".

You should then create a local clone of your own remote fork, but also add the main master repo as a remote as well. To do this, follow the steps in the "Set Up A Local Repo" section.

Create branch for your work

Before working on a feature, you should create a local branch for that feature (see "Work with branches"). Good practice is to name the feature branch after the ticket number you will be working on (for example, "HTML-123"). You can work on the feature in that branch, remembering to commit regularly along the way.

Unit test!

When you've finished your feature, before submitting for review make sure you first to run all the existing unit tests, to make sure you haven't broken any existing code. Running a simple "mvn package" from your top-level project directory will run through all module tests. If any of the tests fail, you'll need to go back and make the necessary corrections.

Get updates from main repository

Once you've got the unit tests to pass, you'll want to make sure to pull in any changes that others may have made to the module since you created your feature branch. To do this, follow the instructions "Pull in upstream changes" under the "More Things You Can Do" section. Note that in addition to merging the the changes into the master branch of your fork, you'll want to merge them into your feature branch as well.

After you've pulled in any new changes, resolve any merge conflicts that may have occurred, and then re-run the unit tests. If they all pass, go ahead and commit any outstanding changes to your branch.

Pushing to github.

Merge your feature branch back into your local master branch and push your local master up to your remote master (see "Push Commits") and then issue a pull request.

Waiting

If it is a very small or simple change, the project lead may automatically merge your changes via github. More often, however, the project lead will add your own remote repo to their remotes, pull your changes in to their own local repository, review the changes and run unit tests, and if all is well, push your changes up to the master branch of the master repo. If the project lead requests further changes, you should go back to your feature branch, make the requested changes, merge them back into your local master, and then push them up your remote master. At this point, you should contact the project lead via email to let him/her know that you've made the requested changes... no need to issue another pull request at this point.

One your changes have been accepted and merged, you can safely delete your local feature branch, and move on to your next feature!

Tips on how to get your changes approved

- Make sure you understand the ticket or feature you are working on: before working on any ticket, make sure it is in the "ready for work" state, and that you fully understand the goal of the ticket.
- Make sure that you haven't broken any unit tests
- **Write your own unit tests** to prove that your new feature works. **If you submit a pull request without any new unit tests to confirm it works as expected, nine times out of ten your changes will be rejected!**