Moodlerooms Technical Onboarding Guide

In order to understand the role and the responsibilities of a Software Developer in the team you should read the following documentation: https://confluence.bbpd.io/display/OSMRDevelopment/New+Developer+Onboarding+Guide

However, as for training and technical information you should follow this updated guide. Also, don't be afraid to ask any question on slack channels. All Devs are friendly and will be more than happy to help you out.

1: Check your access on all the development tools:

- Confluence (Documentation tool): https://confluence.bbpd.io
- Bitbucket (Repository): https://git.mroomstech.com/
- JIRA (Issue tracking tool): https://tracker.mroomstech.com/
- Bamboo (Continuous integration tool): https://bamboo.mroomstech.com/
- If you don't have access to any of these tools you should create a servicenow ticket requesting it: https://blackboard.service-now.com/

2: Setting up your work environment:

- Install and set up Git following this guide: https://confluence.bbpd.io:8443/display/OSMRDevelopment/Git
- Set up your virtual machine following this guide: https://git.mroomstech.com/projects/DEV/repos/moodle-vagrant/browse
- Download PHPStorm (you can use a 30 day trial while you request your license via a servicenow ticket): https://www.jetbrains.com/phpstorm/
- Donwload Cisco any connect (VPN): https://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html
- Download Slack (Chat): https://slack.com/downloads/osx

3: Getting familiar with the Moodlerooms codebase:

- Moodlerooms has a very large codebase with several directories, it will take time to learn where every part of MR is and how everything works, but you can start to learn about the Moodle architecture in the following link: https://docs.moodle.org/dev/Moodle_architecture
- There are several types of plugins inside the MR codebase, all of them are organized in categories depending on their function on the LMS: https://docs.moodle.org/dev/Plugin_types
- Most of the plugins share the same conventions and files in order to keep the codebase organized, you can learn the role of each file in the following documentation: https://docs.moodle.org/dev/Plugin_files
- Moodlerooms uses the same coding guidelines used by moodle and is important to learn and comply with it, the related documentation can be found here: https://docs.moodle.org/dev/Coding_style
- As a first approach to the development guidelines used by moodle yo can follow this tutorial in order to learn to create custom blocks: https://docs.moodle.org/dev/Blocks

4: Learning the development process:

- **Sprint and Story points estimation:** the dev and QA teams work on 2 weeks sprints in which most of all the tickets should be closed, the time required to complete a ticket is estimated in points, for further information

on this please refer to the following documentation: https://confluence.bbpd.io/display/OSMRDevelopment/Estimating

- **Development process:** For further information regarding the development process of a ticket you can refer to the following confluence page: https://confluence.bbpd.io/display/OSMRDevelopment/Development+Processes
- The normal development process when working on a ticket is the following:

1. Assign yourself the ticket.
2. Click in “Start progress” on the ticket.
3. Login to vagrant (if you haven't already) with vagrant ssh
4. Create a local branch using “kow checkout <ticket>”.
5. Create a local database using “kow up” (only if necessary).
6. Solve the issue or develop the new feature.
7. Commit your changes with a descriptive comment.
8. Create a pull request using “kow push -p”.
9. You can review the build process logging into bamboo using the build link inside the ticket
10. Wait until the continuous integration build completes successfully.
11. Change the status of the ticket to “in review”.
12. Write the testing instructions.
13. Request two peer code reviews in the “pd-os-softwareengr” channel inside slack
14. Request the “qa-approved” label from the QA Assignee.
15. Once you have the QA label, 2 code reviews on your pull request, a successful build and your ticket belongs to an active sprint you can send your ticket to integration.
16. After integration your Qa assignee will test if everything works correctly following your testing instructions.
17. If everything works correctly your ticket will be closed.

IMPORTANT NOTICE: when working on a ticket is not recomendable to edit any of the Moodle Core files, if you encounter a problem or a feature that requires you to edit any of the core files it is advised to report the problem or the change that you need on the Moodle issue tracker (https://tracker.moodle.org/secure/Dashboard.jspa). You can check which are the core plugins and which plugins are maintained by Blackboard in the following matrix: Internal Plugin Matrix

Advanced development:

1. AMD modules and javascript files:
   - When working with javascript files and AMD modules you should know that you can do all the changes for development in the javascript files located in the “src” folder, however before commiting you’ll need to minify the edited files using grunt. You can find more information here: https://docs.moodle.org/dev/Javascript_Modules
2. Running a PHPunit test:
   - Writing a PHPUnit test: (PHPUnit Assertions https://phpunit.de/manual/6.5/en/appendixes.assertions.html)
   - Start up PHPunit running “php admin/tool/phpunit/cli/init.php” on the virtual machine terminal.
   - To run a single PHPunit file run “vendor/bin/phpunit my_test_class_name my/tests/filename.php”.
3. Writing and running a Behat test:
   - Writing a Behat test guide: Writing Behat Tests
   - Start up Behat by running “php admin/tool/behat/cli/init.php”
   - To run a Behat file run “vendor/bin/behat --config /srv/moodledata/behat_moodledata/behat/behat.yml <Path to the .feature file>”
4. Doing a Code Review: 3rd Party Plugin Management
5. Developing a Hotfix ticket: When developing a HF ticket you must follow the regular development process, however there are some exceptions that must be taken into account:

   - HF tickets should always include PHPunit or Behat tests when possible.

    - HF tickets must be integrated manually.
    - HF tickets cannot include a version bump.
6. Doing a Backport: https://confluence.bbpd.io:8443/display/OSMRDevelopment/Accepting+Patches+from+Core+Moodle

Further documentation and reference:

- Wardians knowledge base (useful Git commands for development):https://confluence.bbpd.io/pages/viewpage.action?pageId=254642293
- KOW commands https://git.mroomstech.com/projects/DEV/repos/kow/browse
- Code checker https://docs.moodle.org/dev/CodeSniffer