Debugging from the Trenches

Carlos Escobedo
Analyst Developer at Moodle HQ

#MootIEUK19
@carlosagile
Don’t do it in production!
When the BUG comes up
DON’T PANIC
Be yourself
Debugging mindset

- Fix the Problem, Not the Blame.
- Don't Assume It – Prove It.
- The best way to start fixing a bug is to make it reproducible.
- Now what happens before and after.
- Don't Live with Broken Windows. (The Boy Scout Rule)
- Explain the bug to someone else. (Rubber duck)
- Stay aware of what you're doing.
- Understand Stack Traces
- Test Your Software, or Your Users Will.
- A bad Unit Test is worse than No Test. Robert C. Martin (Uncle Bob).
- Ask yourself questions.
Debugging checklist

- Is the problem being reported a direct result of the underlying bug, or merely a symptom?
- Is the bug really in the code? Is it in the OS? Or is it in third party packages?
- If you explained this problem in detail to a coworker, what would you say?
- If the suspect code passes its tests, are the tests complete enough? What happens if you run the test with this data?
- Do the conditions that caused this bug exist anywhere else in the system?
Debugging checklist

- Is there an easier way?
- Am I solving the right problem?
- Why is this a problem?
- What makes it hard?
- Do I have to do it this way?
- Does it have to be done at all?

$CFG->debug = (E_ALL | E_STRICT);
Moodle Debug Developer

Dashboard / Site administration / Development / Debugging
$CFG->debug = (E_ALL | E_STRICT);   // === DEBUG_DEVELOPER
$CFG->debugdisplay = 1;           // unless watch the logs.
$CFG->debugusers = '2,8';         // List of user ids that that always see debug messages.
$CFG->cachejs = false;            // Prevent JS caching
$CFG->debugstringids = 1;        // Show origin of languages strings
$CFG->langstringcache = false;    // Disable language strings cache
$CFG->themedesignermode = true;   // Disable CSS and image cache
$CFG->divertallemailsto = 'root@localhost.local';  // Divert all outgoing emails to this address
$CFG->noemailever = true;         // no emails or other messages should ever be send to real users
$CFG->divertallemailsexcept = 'tester@dev.com, fred(\+.*)@example.com'; // Send emails only for certain email addresses you want to let through for testing.
Moodle Debug Developer

$CFG->perfdebug = 15;
$CFG->debugpageinfo = true;

config.php
Debugging

**PHP**

```php
if ($CFG->debugdeveloper) {
    ...
    debugging('Argument $record must be an instance of stdClass.', DEBUG_DEVELOPER);
    ...
}
```

**JS**

```javascript
define(['core/log'], function(log) {
    log.info(....);
});
```

`$x()` in web console useful to XPath queries
Code Linting

https://docs.moodle.org/dev/Linting

PHP_CodeSniffer & Local_codechecker (by Tim Hunt)
JavaScript (ESLint) https://eslint.org/
CSS/SCSS/LESS (stylelint) https://stylelint.io/
Lint in Editors (Sublime, Atom, Vim, ...)

PHPXREF
http://phpxref.sourceforge.net/
Profiling

Dashboard / Site administration / Development / Profiling

- [Link](https://docs.moodle.org/dev/Profiling_PHP#Tideways_for_php7)
- [Link](https://tracker.moodle.org/browse/MDL-62280) (Tideways XHProf)
Run tasks

```bash
php admin/tool/task/cli/schedule_task.php
--execute="\tool_cohortroles\task\cohort_role_sync"
--showdebugging
```

Since Moodle 3.6:

```bash
php admin/tool/task/cli/adhoc_task.php --execute
```
Web Service

**CURL**

curl 'http://localhost/m/stable_master/webservice/rest/server.php?moodlewsrestformat=json' --data 'wsfunction=tool_mobile_get_config&wstoken=08788646962394c92a9b6c26a3be7abe&section=mobileapp' | python -m "json.tool"
Web Service

BROWSER

http://localhost/m/stable_master/webservice/rest/server.php?wsfunction=tool_mobile_get_config&wstoken=08788646962394c92aab6c26a3be7abe

http://localhost/m/stable_master/webservice/rest/simpleserver.php?wsusername=wstest&wspassword=wstest&wsfunction=core_user_create_users&users[0][username]=mootesdev&users[0][firstname]=Test&users[0][lastname]=MootWS&users[0][email]=mootesdev@moodle.com&users[0][password]=12345&users[0][customfields][0][type]=cp&users[0][customfields][0][value]=08208
Web Service Resources

WS API  https://docs.moodle.org/dev/Web_services_API

Dev Docs
https://docs.moodle.org/dev/Creating_a_web_service_client

MoodleHQ WS Clients:
https://github.com/moodlehq/sample-ws-clients

Juan Leyva Master of WS
http://es.slideshare.net/juanleyva/mastering-moodle-web-services-development
Moodle Mobile

Debugging WS Request
https://docs.moodle.org/dev/Moodle_Mobile_debugging_WS_requests

Use an online web version of the app that can be easily debugged using the Chrome browser.
https://prototype.moodle.net/mobile/webapp/

Development using Chrome or Chromium
https://docs.moodle.org/dev/Moodle_Mobile_development_using_Chrome_or_Chromium
ionic serve --browser chromium
Docker to the rescue

**LDAP**

docker run -p 389:389 --name my-openldap-container --hostname ldap.example.org --env LDAP_REPLICATION=true --detach osixia/openldap:1.2.0

docker run -p 6443:443 --name phpldapadmin-service --link my-openldap-container:ldap-host --env PHPLDAPADMIN_LDAP_HOSTS=ldap-host --detach osixia/phpldapadmin:0.7.1

You can go to [https://localhost:6443/](https://localhost:6443/) in your browser
login: cn=admin,dc=example,dc=org
password: admin

[https://tracker.moodle.org/browse/MDL-61296](https://tracker.moodle.org/browse/MDL-61296)
Moodle Docker

https://github.com/moodlehq/moodle-docker

```
moodle-docker: Docker Containers for Moodle Developers

This repository contains Docker configuration aimed at Moodle developers and testers to easily deploy a testing environment for Moodle.

Features:

- All supported database servers (PostgreSQL, MySQL, Microsoft SQL Server, Oracle XE)
- Behat/Selenium configuration for Firefox and Chrome
- Catch-all smtp server and web interface to messages using MailHog
- All PHP Extensions enabled configured for external services (e.g. solr, ldap)
- All supported PHP versions
- Zero-configuration approach
- Backed by automated tests
```

Oracle Example

```bash
export MOODLE_DOCKER_WWWROOT=/Home/dev/htdocs/moodleoracle
export MOODLE_DOCKER_DB=oracle
cp config.docker-template.php $MOODLE_DOCKER_WWWROOT/config.php
bin/moodle-docker-compose up -d
bin/moodle-docker-wait-for-db
```
Expose local servers to internet

**NGROK** ([https://ngrok.com/](https://ngrok.com/))

- Allows you to expose your local instance to the world
- Creates a tunnel that diverts all web traffic for the given url to your localhost.
- Extremely helpful when testing/developing features that require postback to your Moodle instance.
  - For ex.: LTI, Paypal integration, OAuth verification etc.
Dynamic ngrok configuration with MDK

Created by Andrew Nichols

How to configure it:

- Copy ngrok.php file into the MDK Storage Directory
- Place ngrok.sh into the MDK Local Scripts Directory (typically ~/.moodle-sdk/scripts)
- Create an ngrok Authentication according to the **ngrok Documentation**
- Place the ngrokrunner script in your path and make it executable

How to use it:

- Add a line to the config.php to load a custom ngrok configuration
  
  ```
  $ mdk run ngrok
  ```
- Start ngrok with custom configuration of port, subdomain, and region
  
  ```
  $ ngrokrunner
  ```

[https://gist.github.com/andrewnicols/a338c43fec0a87a60248b8097de2ef5](https://gist.github.com/andrewnicols/a338c43fec0a87a60248b8097de2ef5)
MDK - Moodle Development Kit

- [https://github.com/FMCorz/mdk](https://github.com/FMCorz/mdk) developed by Frédéric Massar
- [https://docs.moodle.org/dev/Moodle_Development_kit](https://docs.moodle.org/dev/Moodle_Development_kit)

#update mdk
sudo pip install --upgrade moodle-sdk
#your mdk config file.
./moodle-sdk/config.json
#place where save script that you want to run like mdk run SCRIPT
./moodle-sdk/scripts
#upgrade your Moodle instances
mdk upgrade --all --update
#config your master branch
mdk config set masterBranch 37
#install a new instance of Moodle
mdk create -v master -n stable_master -i --run users mindev
#install a new integration instance of Moodle
mdk create -v 34 -n int34 --integration -i --run users mindev
MDK - Moodle Development Kit

# create a new branch MDL-56141-master-test
mdk pull MDL-56141 -t
# run phpunit tests and init the environment
mdk phpunit -r -u mod/assign/tests/locallib_test.php
# run behat tests and init the environment
mdk behat -r --tags=@message_popup
# install community plugins
mdk plugin install tool_policy
# create a new clean branch MDL-61575-master
mdk fix 61575
MDK - Moodle Development Kit

#force push the commits.
mdk push -t -f
#do the magic backport
mdk backport -v 36

#Rebase your local branches
mdk rebase --issues 65015 --version master --push

#Uninstall a plugin also delete the directory
mdk plugin uninstall <pluginname>

Created by Adrian Greeve
https://github.com/abgreeve/mdk/tree/plugin-uninstall
GDPR Privacy Utils for testing

https://docs.moodle.org/dev/Privacy_API/Utilities

Scripts to:
- Test of privacy API compliance
- Test of exporting user data
- Test of deleting user data

Save the scripts into .moodle-sdk/scripts
Run scripts with $mdk run testcompliance.php
More Tools

Test your code online [https://fiddles.io/#](https://fiddles.io/#)
JSFiddle
SQLFiddle

Chrome Dev Tools
[https://developers.google.com/web/tools/chrome-devtools/](https://developers.google.com/web/tools/chrome-devtools/)

Test APIs and WS with Postman
[https://www.getpostman.com/](https://www.getpostman.com/)

Catch mails with MailCatcher
[https://mailcatcher.me/](https://mailcatcher.me/)
GIT BISECT

Use binary search to find the commit that introduced a bug

https://git-scm.com/docs/git-bisect

Start the search

$ git bisect start
$ git bisect good ea13c6454a45988e88e91e9a41ab5fcd1d5e1be8
$ git bisect bad dbee1d569009217dd89fc4909893f93d5f5e9a43

Bisecting: 2 revisions left to test after this (roughly 1 step)


Now, test the code, then you explain to git the result:
git bisect bad or git bisect good depending on the success. And repeat until you
find the bad commit. At last, run git bisect reset to get back to the HEAD commit.

$ git bisect reset
Thank You
Extra Bonus

Rubber Duck Debugging
https://rubberduckdebugging.com

1. Beg, borrow, steal, buy, fabricate or otherwise obtain a rubber duck (bathtub variety)

2. Place rubber duck on desk and inform it you are just going to go over some code with it, if that’s all right.

3. Explain to the duck what your code is supposed to do, and then go into detail and explain your code line by line.

4. At some point you will tell the duck what you are doing next and then realise that that is not in fact what you are actually doing. The duck will sit there serenely, happy in the knowledge that it has helped you on your way.