DevJam
MDK y otros aliados “chachis”

Sara Arjona
@sara_arjona
Developer at Moodle HQ

Carlos Escobedo
@carlosagile
Developer at Moodle HQ

#MootES18
MDK:
Moodle Development Kit
MDK

- Stands for Moodle Development Kit
- Developed by Frédéric Massart, during a personal project week
- For installing on Debian-based systems

```
$ git clone git://github.com/FMCorz/mdk.git moodle-sdk
$ sudo apt-get install python-pip libmysqlclient-dev libpq-dev python-dev unixodbc-dev
$ sudo pip install moodle-sdk
$ mdk init
```

- Windows is not supported

https://github.com/FMCorz/mdk
Creating dev environment

$ mdk create --help
$ mdk create -v master -i -n stable_master-mootes
$ cd stable_master-mootes/moodle
$ mdk run --list
$ mdk run setup
More ‘mdk run’ features

- **mindev / dev / undev**: Sets/unsets the instance ready for developers
- **users**: Create a set of users
- **enrol**: Enrol users with username starting with s, t or m in all the courses
- **setup**: mindev + users + enrol
- **make_course**: Creates a S size course
- **webservices**: Entirely enables the Web Services
- **external_functions**: Refresh the services and external functions
- **tokens**: Lists the external tokens
- **version**: Reset all version numbers

And you can also put your own scripts on .moodle-sdk/scripts !!
Some configuration needed

- Add this to the config.php (for behat):

```php
$CFG->behat_profiles = [
    'default' => [
        'browser' => 'chrome',
        'extensions' => [
            'Behat\MinkExtension' => [
                'selenium2' => [
                    'browser' => 'chrome',
                ]
            ]
        ]
    ]
];
```
Testing a patch

$ mdk pull -t MDL-1234

$ mdk phpunit --help
$ mdk phpunit -r -u badges/tests/privacy_test.php

$ mdk behat --help
$ mdk behat -r -n 'Add a badge'
Fixing a bug

$ mdk fix MDL-1234

--- Make the required changes ----

$ git commit . -m "MDL-1234 codearea: commit message"

$ mdk push --help
$ mdk push -t

$ mdk backport --help
$ mdk backport -v 35 -p -t
And even more...

- **info**: Information about a Moodle instance
- **config**: Manage your MDK configuration
- **doctor**: Perform several checks on your current installation
- **purge**: Purge the cache of an instance
- **precheck**: Pre-checks a branch on the CI server
- **plugin**: Manage your plugins (download and/or install directly from the database)
- **upgrade**: Update the instance from remote and run the Moodle upgrade script
- **install/uninstall**: Install/uninstall a Moodle instance
- **remove**: Completely remove an instance
Some interesting tips & tricks
Security guidelines

- Authenticate the user and verify course/module access:
  - Call to require_login or require_course_login
  - Use $course and/or $cm when needed
- Check permissions
  - Call to has_capability or require_capability before
- Some good practices:
  - Minimize the use of global variables
  - Initialise objects and arrays before using them
  - Test every input field with tricky content. You can use a standard test input with different kind of special characters
- Log every request (create events and trigger them)

https://docs.moodle.org/dev/Security
Security guidelines

- For preventing XSS, don't trust any input from users
  - Use MoodleForms and set the most specific PARAM_* type:
    - PARAM_TEXT strip tags but multi-lang
    - PARAM_NOTAGS strip all tags
    - PARAM_RAW no cleaning (used with HTML editor)
    - PARAM_ALPHA for hidden fields
  - Before performing actions, use data_submitted() and confirm_sesskey()
  - Clean input using optional_param or required_param with appropriate PARAM_... type
    - Group these calls together at the top of the script
    - Don't access $_GET, $_POST or $_REQUEST directly

https://docs.moodle.org/dev/Security
Security guidelines

● Clean and escape data before output
  ○ format_string minimal HTML (multi-tag support)
  ○ format_text format all the content

● Escape data before storing it in the database
  ○ Use the XMLDB library
  ○ For custom SQL code use place-holders to insert values (to avoid injections)

https://docs.moodle.org/dev/Security
Security guidelines

● Restrict what people can see according to groups
  ○ Check grouplib functions
    ■ Group_is_member
    ■ group_has_membership

● Not use unserialize and serialize functions
  ○ Object Injection
    (https://www.owasp.org/index.php/PHP_Object_Injection)

● Avoid eval()
  ○ Direct Dynamic Code Evaluation (Eval Injection)

● Double check validation in forms and inside lib.php

https://docs.moodle.org/dev/Security
SQL Performance

- Define good indexes
- Use Moodle's XMLDB editor
- Minimize the number of DB queries
- Use recordsets when dealing with a large dataset.  
  - It’s much more memory efficient!!
- Measuring performance:
  - HQ uses a tool based on JMeter
  - It isn’t very reliable because of the AJAX
- Explain Plan is your friend

https://docs.moodle.org/dev/Security
Useful plugins

Code checker

https://moodle.org/plugins/local_codechecker

- Maintained by Tim Hunt and Eloy Lafuente
- Code checker tool based on the PHP CodeSniffer library
- Check for many parts of the Moodle coding style

PHP Doc checker:

https://moodle.org/plugins/local_moodlecheck

- Maintained by Marina Glancy
- Check phpdocs in the code for compliance with Moodle Coding Style
Useful plugins

Moodle eMail Test

https://moodle.org/plugins/local_mailtest

- Maintained by Michael Milette
- Moodle eMailTest allows administrators to test Moodle’s email system
- A trace of the SMTP dialogue will be displayed if the email message cannot be sent for any reason.

UpgradeDB

https://moodle.org/plugins/local_upgradedb

- Maintained by Carlos Escobedo
- This plugin allow install new tables into DB using xmldb files under developer control.
- Also you can analyze and Refresh Indexes in MySQL Tables, show Keys and Indexes and Show XMLFile using xmldbtools.
Activity example course

- You can access as manager or teacher, create a backup and restore it in some of your installations.
- Contains most kind activities and resources examples
- Great for testing!! :-)

https://qa.moodle.net/course/view.php?id=2
Ngrok

- 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.

```
ngrok by @inconshreveable

Session Status  online
Version         2.2.8
Region          United States (us)
Web Interface   http://127.0.0.1:4040
Forwarding     http://c6860d6e.ngrok.io -> localhost:80
Forwarding     https://c6860d6e.ngrok.io -> localhost:80
Connections    ttl  opn  rt1  rt5  p50  p90
                0    0    0.00  0.00  0.00  0.00
```

https://ngrok.com/
Configuration for Dynamic ngrok

- Download the included ngrok.php file and place it into the MDK Storage Directory
- Place the included ngrok.sh script and place it in the MDK Local Scripts Directory
- Create an ngrok Authentication according to the ngrok Documentation
- Place the included ngrokrunner script somewhere within your path and make it executable
- Create a new config file in ~/.config/ngrokrunner with contents:

  SUBDOMAIN=yoursubdomain
  REGION=yourpreferredregion

Usage

`mdk run ngrok`
This will add a line to the config.php to load a custom ngrok configuration. This only needs to be done once, but does not harm if run multiple times.

Once configured you can simply run:

`ngrokrunner`
This will start ngrok with your custom configuration of port, subdomain, and region.

Thanks to Andrew Nichols!!
[https://gist.github.com/andrewnicols/a338c43feca0a87a60248b8097de2ef5](https://gist.github.com/andrewnicols/a338c43feca0a87a60248b8097de2ef5)
¡Gracias!

Sara Arjona
@sara_arjona
Developer at Moodle HQ

Carlos Escobedo
@carlosagile
Developer at Moodle HQ

#MootES18