Administrez votre Moodle avec Moosh

Céline Pervès ~ Olivier Valentin

Déroulé de l'atelier

- Introduction à Moosh
- Installation
- Lancement de commandes
- Dans les coulisses de Moosh
  - Principes de base de développement
- Création de commandes
  - guidée
  - libre

Introduction à Moosh

Qu'est-ce que Moosh

- Moosh = **MOO**dle **SH**ell
- Outil ligne de commande de Tomasz Muras
  - https://moosh-online.com
- Permet d'exécuter des tâches Moodle
  - Utilise les librairies Moodle
- Permet
  - De préparer des scripts d'installation Moodle complet
  - Gérer son(ses) Moodle en ligne de commande
  - Ajout de commandes possibles sans avoir à redéployer Moodle

Techniquement

- Se greffe sur un Moodle installé
- Se sert des fichiers du Moodle
- Plutôt Linux
  - Sous windows toutes les commandes ne sont pas susceptibles de fonctionner

Installation atelier

Préparation de l'image docker

- → Sous Windows par ici
- sinon on continue

Volumes docker

- Créer les dossiers volume utiles

```
mkdir -p /path_to_workshop_volume/{mdldatas,code,moosh}
```

- Récupérer le fichier docker-compose.yml et le fichier env.sample

```
git clone https://git.unistra.fr/cperves/moosh-workshop.git
cd moosh-workshop
cp .env.sample .env
# remplir avec notamment le volume précédemment créé (sa racine)
# VOLUMES_ROOT_PATH=/path_to_workshop_volume/
```

Lancer le conteneur

- Récupérer l'image docker

```
docker pull registry.app.unistra.fr/cperves/moosh-workshop/moodle_0
```

- Premier lancement
  - Initialisation de la base de données Moodle

```
cd racine_dockercompose
# Pour pouvoir récupérer les image sur le hub docker
docker login --username <votrecompte@votre_domaine.com>
INIT_PLF=1 docker compose up
# Si vous voulez lancer en silencieux
# INIT_PLF=1 docker compose up --detach
# Les autres fois, une fois la bdd initialisée
# INIT_PLF=0 docker compose up --detach
```

Rentrer dans le conteneur

```
docker exec -it moosh-workshop-webserver-1 /bin/bash
```

- Dans /var/www/html on a le code Moodle
  - Que l'on retrouve dans le volume /path_to_workshop_volume/code

Installer moosh

Installer moosh - Compatibilités

- A partir de php 8.1, lors de l'éxécution de Moosh, il y a de nombreux warnings, souvent liés à des Deprecated
  - Ils ne gênent pas l'éxécution de Moosh
- Pour cet atelier nous sommes en php 8.0

Installer moosh - Commandes

Depuis le conteneur docker

```
# Créer un répertoire pour Moosh
mkdir -p /opt/moosh
# Récupérer le code moosh et le placer dans le répertoire créé
git clone https://github.com/tmuras/moosh --branch=1.21  /opt/moosh
# Récupérer et installer composer dans /usr/local/bin
curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=c
# Aller dans le répertoire de moosh
cd /opt/moosh
# Installer moosh avec ses dépendances
composer install --no-dev --no-interaction -o
# Faire un lien symbolique pour éxécuter moosh, simplifie le démarrage
ln -s $PWD/moosh.php /usr/local/bin/moosh
# Aller dans le répertoire Moodle
cd /var/www/html
moosh -n category-list
# -n, --no-user-check Don't check if Moodle data is owned by the user running script
```

Lancement des commandes

Commandes

```
moosh [options globales] commande [options] [arguments]
```

- options globales:
  - placées avant la commande
  - -n : ne vérifie pas si les moodle datas appartiennent à l'utilisateur qui exécute le script
  - --list-commands : liste les commandes
  - -p : permet de spécifier le path moodle
  - -u : permet de spécifier l'utilisateur Moodle courant, par défaut il s'agit d'admin
- les arguments se placent après les options sinon les options ne sont pas prises en compte
- les options s'écrivent sous la forme -o=valeur --option=valeur
- -h après la commande donne l'aide de la commande

Exécuter des commandes Moosh simples

Exécuter des commandes Moosh simples - Prerequis

- Si vous n'avez pas réussi à installer moosh vous pouvez utiliser l'image docker corrigée
  - `registry.app.unistra.fr/cperves/moosh-workshop/moodle_1_with_moosh`
  - dans .env il suffit de changer MOODLE_IMAGE_NAME

```
MOODLE_IMAGE_NAME=registry.app.unistra.fr/cperves/moosh-workshop/moodle_1_with_moosh
```

Exécuter des commandes Moosh simples - Instructions

- En se basant sur le site Moosh ou à l'aide de la commande `moosh -n --help`
- Créer une catégorie de cours
- Créer un cours dans cette catégorie
- créer 3 comptes utilisateurs
- Inscrire deux des comptes en tant que student dans le cours créé
- Inscrire le dernier compte en tant que teacher

Exécuter des commandes Moosh simples - Vérifier

- Vérifier que l'on a bien ces éléments sur moodle
  - Via l'interface web
  - Avec moosh

Un exemple

```
moosh -n user-create --password password --email student1@example.com --digest 2 --city Marse
moosh -n user-create --password password --email student2@example.com --digest 2 --city Marse
moosh -n user-create --password password --email teacher@example.com --digest 2 --city Marsei
categoryid=$(moosh -n category-create -d "Moosh Workshop Category" workshopCat );
courseid=$( moosh -n course-create --category $categoryid --fullname 'Physics' physics );
moosh -n course-enrol -r student $courseid student1
moosh -n course-enrol -r student $courseid student2
moosh -n course-enrol -r teacher $courseid teacher
# verifications
moosh -n category-list
moosh -n course-list
moosh -n course-list shortname ilike \'%physics%\'
moosh -n sql-run 'select username from {user}'
moosh -n sql-run "select ra.id,r.shortname, u.username user from {user} as u inner  join {rol
```

Dans les coulisses de Moosh

À l'exécution

- Charge des librairies essentielles
  - clilib.php → client ligne de commande moodle
- Charge les commandes Moosh
  - on peut donc en ajouter
- Lit les arguments de la ligne de commande
  - arguments
  - options
- Charge le config.php de Moodle
- Se logue en tant qu'utilisateur avec role administrateur
  - `get_admin()`
- Éxécute la commande

Les commandes disponibles

- Liste via `moosh --list-command`
- répertoire et organisation `/repertoire_moosh/Moosh/Command/`
  - Sous répertoires MoodleXX et Generic
    - Sous répertoires de type Groupe (important pour le nom de la commande)
      - Class php `<GroupName><Commande>.php`
      - e.g `ActivityAdd.php`
        - path `Moosh/Command/Moodle39/Activity/ActivityAdd.php`

Au coeur de la commande - Classe

- Namespace
  - Comme le path du fichier
  - e.g `namespace Moosh\Command\Moodle39\Activity`
- Classe
  - Hérite de `Moosh\MooshCommand\MooshCommand`
- Nom de la commande
  - Nom de classe
  - Constructeur
    - Arguments : (`Commande`, `GroupeName`)
      - Comme dans nom du fichier
      - en minuscule
      - e.g :
    - `parent::__construct('add', 'activity');`
      - → Commande : activity-add

Au coeur de la commande - les arguments

- Ajouter un argument
  - Dans le constructeur de la classe
  - `$this->addArgument('mon_argument');`
- Déclarer le nombre d'arguments

```
$this->minArguments = entier;
$this->maxArguments = entier;
```

Au coeur de la commande - les options

- Dans le constructeur de la classe

```
  $this->addOption('raccourci|nom_complet', 'description', 'valeur par défaut');
```

- valeur par défaut, null si non renseigné

```
public function __construct() {
    parent::__construct('add', 'activity');
        $this->addOption('n|name:', 'activity instance name');
}
```

Au coeur de la commande

Gestion de sorties écrans

- Gestion des sorties via la librairie Moodle cli
  - cli_write et cli_writeln
  - cli_input('prompt', 'valeur par défaut', 'liste d'options', 'sensible à la casse')
  - cli_problem (sans sortie), cli_error (avec sortie)
  - → voir lib/clilib.php

Au coeur de la commande - On pratique

Au coeur de la commande - construct

- On peut observer les commandes via le volumes du code (VOLUMES_ROOT_PATH/moosh)
- Ouvrir le fichier `moosh/Moosh/Command/Moodle39/Course/CourseCreate.php`
  - (lien)
- Observez les options disponibles dans la fonction `__construct`

Au coeur de la commande - execute

- `require_once $CFG->dirroot . '/course/lib.php';`
  - Permet à moosh de charger les la librairie du cours
- `expandOptionsManually`
  - Est indispensable pour transformer les options de la ligne de commande
- *options* `=this->expandedOptions`
  - Permet de récupérer les options de la ligne de commande sous forme de tableau
- `get_config()` → fonction du coeur moodle pour récupérer les settings moodle
  - Dans le cas présent cela permet de récupérer le format de cours par défaut, le nombre de sections
- `create_course` provient de /course/lib.php
  - Moosh se sert bien du code Moodle pour exécuter ses actions

Ajouter une commande Moosh

Ajouter une commande Moosh - On pratique

- Télécharger le fichier RequestSelect.php
- Le placer dans le répertoire VOLUMES_ROOT_PATH/moosh/Moosh/Command/Generic/Request
  - (VOLUMES_ROOT_PATH=/path_to_workshop_volume/)
  - Il faut respecter le répertoire pour que Moosh trouve la nouvelle commande
  - Il faudra donc créer le répertoire Request et lui accorder suffisamment de droits
- Essayer de lancer la commande dans le conteneur
  - `moosh -n select-request`
- Que se passe t'il?

Ajouter une commande Moosh

Rendre la commande effective

- Dans le conteneur
  - Dans le répertoire de moosh lancer:

```
composer dump-autoload
```

- Retourner dans le répertoire racine de moodle et lancer

```
moosh -n select-request "select * from {user}"
```

- It works!

Ajout de commandes - Résumé

- Dans le répertoire du moosh
  - Ou dans le répertoire .moosh de l'utilisateur courant (/home/user/.moosh)
- Respecter la hiérarchie de fichier
- Lancer la commande pour prendre en compte les nouvelles commandes

```
composer dump-autoload
```

Création de commandes

task-schedule

Création guidée de commandes - task-schedule

- Nous allons créer une commande qui permet de changer les paramètres de programmation des tâches programmées Moodle.

task-schedule - classe

- Via le volume Moosh docker (ou accès direct dans le conteneur)
- Dans le répertoire Moosh
  - attention aux droits si on travaille sur le volume : il faudra les changer avec chmod

```
$mkdir -p Moosh\Command\Moodle39\Task
$cd Moosh\Command\Moodle39\Task
```

- Créer un fichier TaskSchedule.php
- Rentrer le squelette de la classe

```
<?php
namespace Moosh\Command\Moodle39\Task;
use Moosh\MooshCommand;
class TaskSchedule extends MooshCommand
{
}
```

task-schedule - constructeur

- A l'intérieur de la classe ajouter

```
public function __construct() {
    parent::__construct('schedule', 'task');
    $this->addOption('M|minute:', 'minute');
    $this->addOption('H|hour:', 'hour');
    $this->addOption('d|day:', 'day');
    $this->addOption('m|month:', 'month');
    $this->addOption('w|dayofweek:', 'Day of week');
    $this->addOption('x|disabled:', 'Disbaled');
    $this->addOption('r|resettodefaults:', 'Reset to defaults', 0);
    $this->addArgument('taskname');
    $this->minArguments = 0;
    $this->maxArguments = 1; # Valeur par défaut
}
```

task-schedule - execute

- Ajouter une fonction execute

```
public function execute()
{
}
```

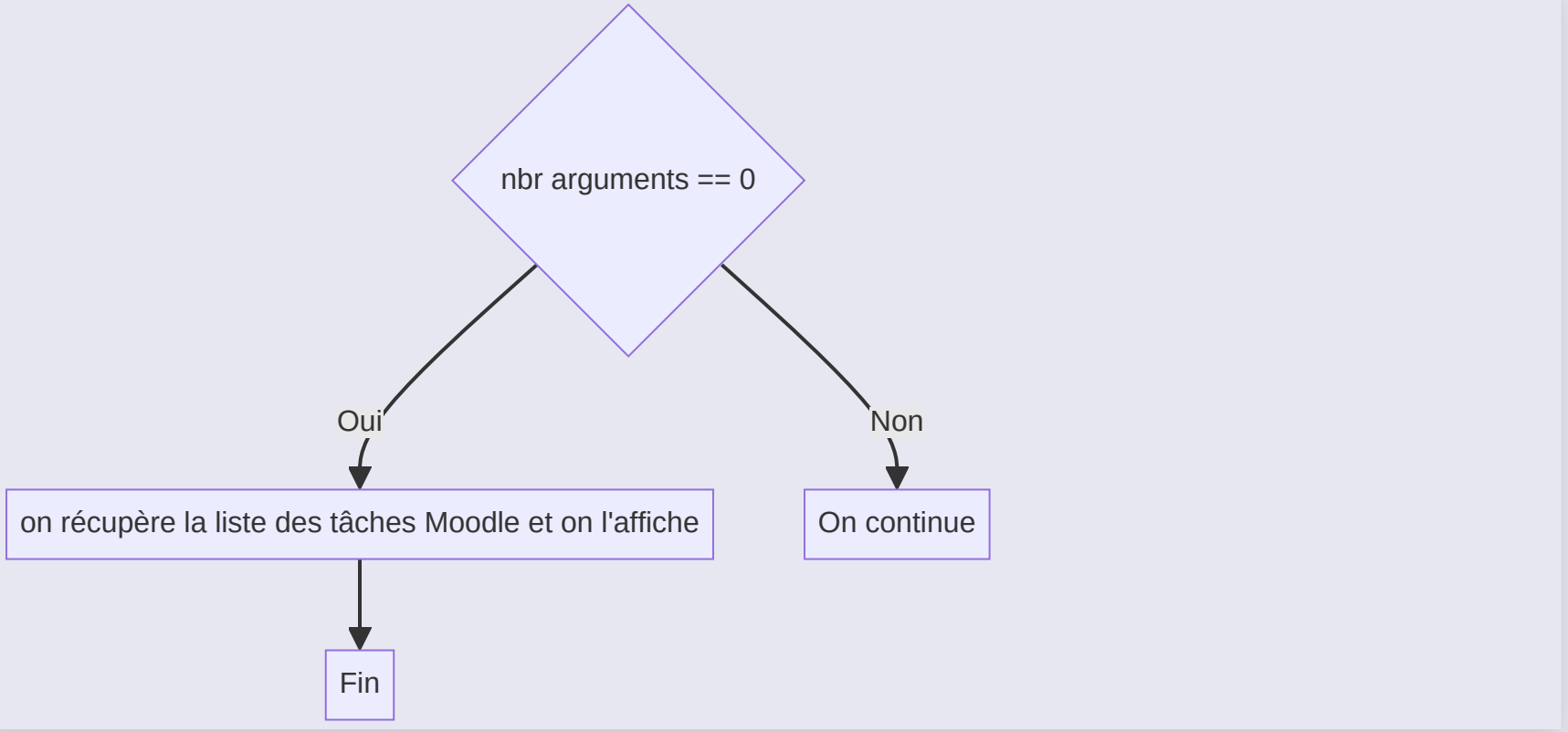
task-schedule - On cherche dans Moodle

- Administration du site → Tâches → Tâches programmées
- → clique sur la modification d'une tâche
  - admin/tool/task/scheduledtasks.php
  - Dans l'url on a : action=edit&task=`taskname` → recherche dans le code
  - Dans la page on repère :

```
\core\task\manager::get_default_scheduled_task($taskname);
```

  - On fouille aussi dans \core\task\manager
    - lib/classes/task/manager.php
    - référence

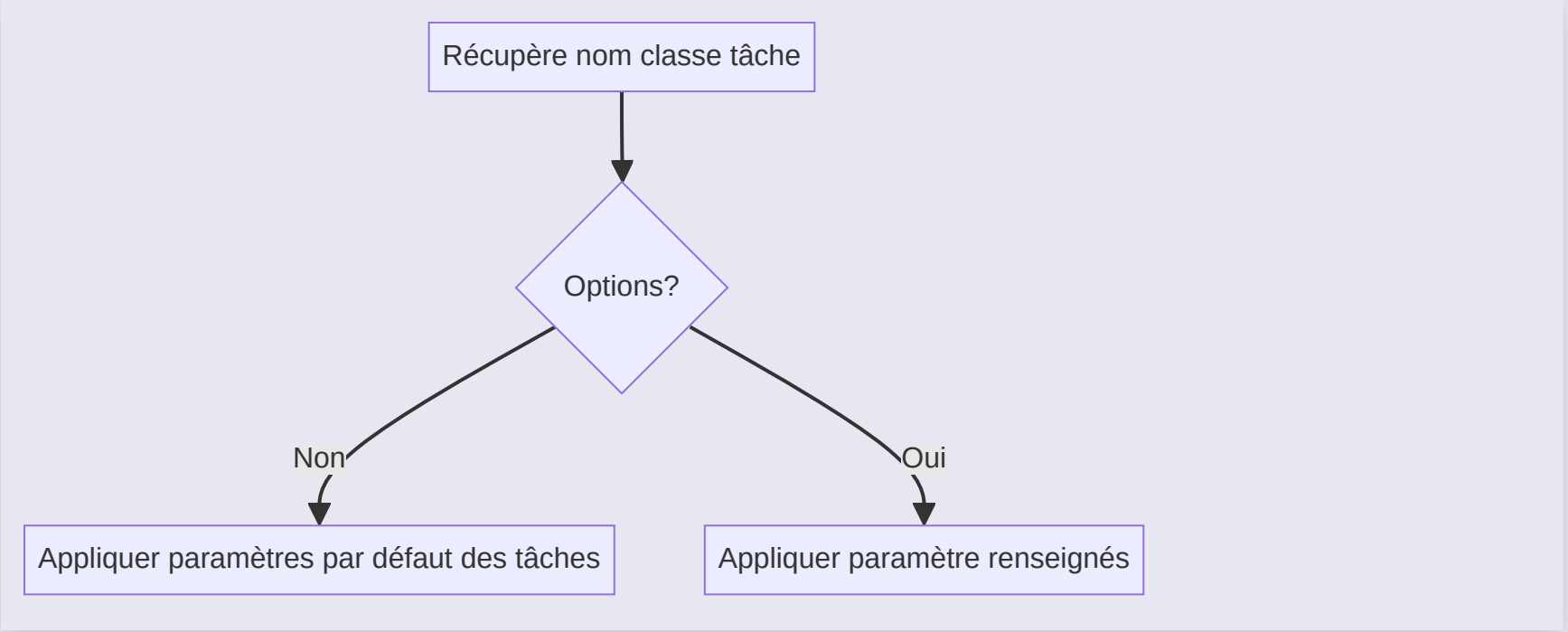
task-schedule - liste tâches



task-schedule - liste tâches

```
public function execute()
{
    if(count($this->arguments) == 0){
        $tasks = \core\task\manager::get_all_scheduled_tasks();
        $tasknames = array();
        foreach($tasks as $currenttask){
$tasknames[] = get_class($currenttask);
        }
        cli_writeln('Available task names are :');
        cli_writeln(implode(PHP_EOL, $tasknames));
        die;
    }
}
```

task-schedule - prise en charge de l'argument et des options



task-schedule - prise en charge de l'argument et des options

```
[...]
$taskname = $this->arguments[0];
$task = \core\task\manager::get_scheduled_task($taskname);
if (!$task) {
    cli_error("task $taskname not exists");
}
if (!empty($this->expandedOptions['resettodefaults'])) {
    $defaulttask = \core\task\manager::get_default_scheduled_task($taskname);
    $task->set_minute($defaulttask->get_minute());
    $task->set_hour($defaulttask->get_hour());
    $task->set_month($defaulttask->get_month());
    $task->set_day_of_week($defaulttask->get_day_of_week());
    $task->set_day($defaulttask->get_day());
    $task->set_disabled($defaulttask->get_disabled());
    $task->set_customised(false);
} else {
    if(!empty($this->expandedOptions['minute'])){
        $task->set_minute($this->expandedOptions['minute']);
```

task-schedule - appliquer les paramètres à la tâche

- Api Moodle des tâches \core\task\manager
- Mettre des message de sortie
- Gérer une éventuelle erreur

task-schedule - appliquer les paramètres à la tâche

```
 try {
    \core\task\manager::configure_scheduled_task($task);
    cli_writeln("Task configured");
} catch (Exception $e) {
    cli_writeln("error while configuring task $taskname");
    cli_error($e->getMessage());
}
```

task-schedule - exécuter

```
composer dump-autoload
moosh -n -p /var/www/html task-schedule -d=1 -H=1 -m=1 "\tool_messageinbound\task\cleanup_tas
```

- Regarder le résultat dans l'interface web Moodle
  - Administration du site → Tâches → Tâches programmées
- Changer les paramètres
- lancer avec l'option resettodefaults

```
moosh -n -p /var/www/html task-schedule -r=1 "\tool_messageinbound\task\cleanup_task"
```

Création de commandes

Déplacer un cours d'une catégorie à une autre

Déplacer un cours d'une catégorie à une autre

- Nous allons créer une commande Moosh qui permet de déplacer les cours d'une catégorie à une autre

Déplacer un cours d'une catégorie à une autre

Fichier et classe

- Créer le fichier et la classe
  - namespace Moosh\Command\Moodle39\Category
  - CategoryMoveCoursesFromCategoryToAnother
- Préparer le constructeur

Déplacer un cours d'une catégorie à une autre

Fichier et classe

```php
<?php
namespace Moosh\Command\Moodle39\Category;
use Moosh\MooshCommand;

class CategoryMoveCoursesFromCategoryToAnother extends MooshCommand {

    public function __construct() {
        parent::__construct('move-courses-from-category-to-another', 'category');
    }
}
```

Déplacer un cours d'une catégorie à une autre

Arguments

- Ajouter deux arguments
  - sourcecategory
  - destinationcateogory

Déplacer un cours d'une catégorie à une autre

Arguments

```
public function __construct() {
    parent::__construct('move-courses-from-category-to-another', 'category');
    $this->addArgument('sourcecategory');
    $this->addArgument('destinationcategory');
}
```

Déplacer un cours d'une catégorie à une autre

On cherche dans Moodle

- Il faut trouver dans moodle comment déplacer des catégories
- on cherche la page qui le fait dans l'interface web
- on relève la librairie, les classe,les méthodes et/ou fonctions nécessaires

Déplacer un cours d'une catégorie à une autre

On cherche dans Moodle

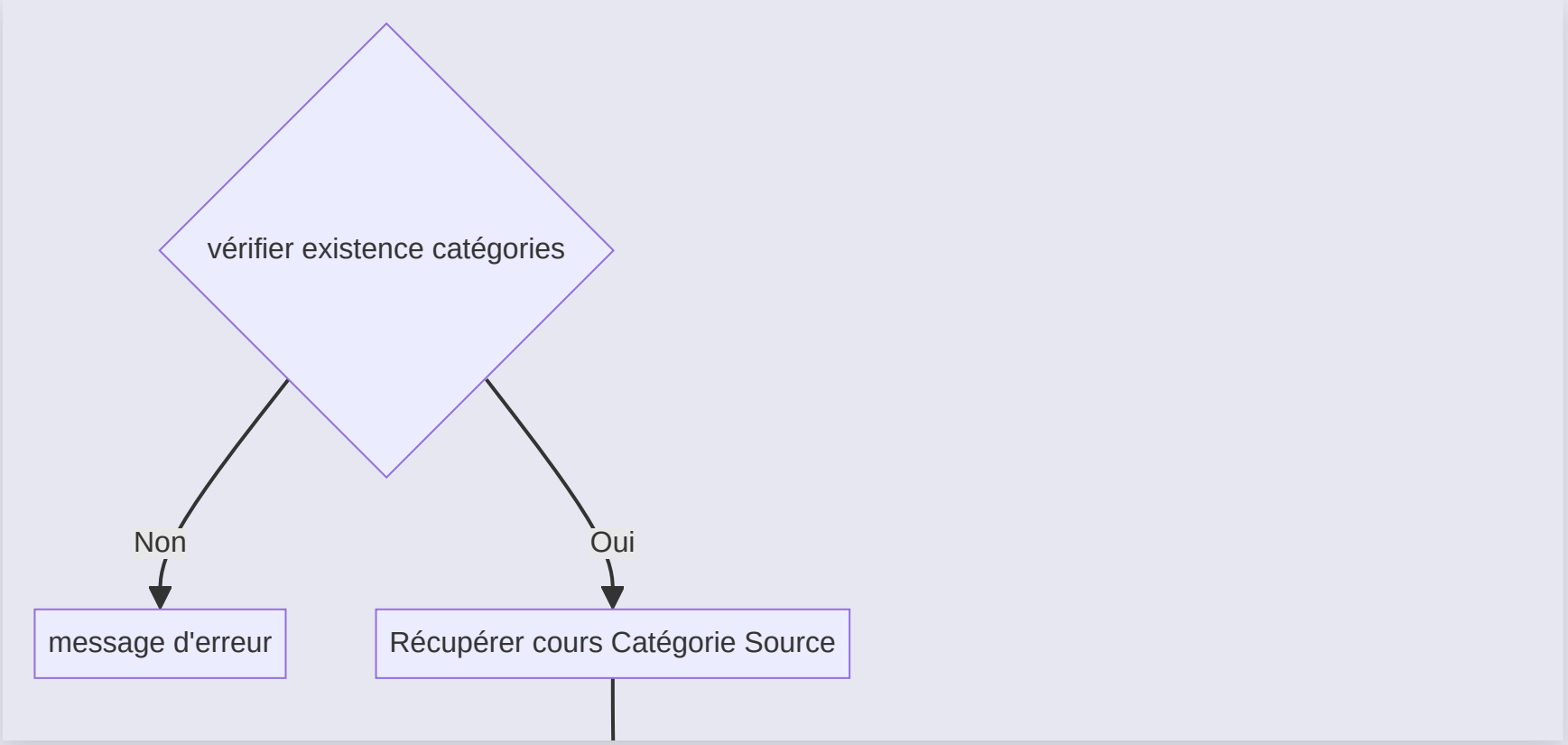
- Edition du cours
  - course/edit.php
  - update_course($data$,editoroptions);

Déplacer un cours d'une catégorie à une autre

Prérequis

- Nous aurons donc besoin des librairies relatives à
  - course_category
  - course

Déplacer un cours d'une catégorie à une autre - Flux



Déplacer un cours d'une catégorie à une autre - code

```
public function execute()
{
    global $DB, $CFG;
    require_once($CFG->dirroot.'/course/classes/category.php');
    require_once($CFG->dirroot.'/course/lib.php');
    // Check that category ids exists.
    try {
        $sourcecategory = \core_course_category::get($this->arguments[0]);
    } catch (moodle_exception $me){
        cli_error('source category does not exists');
    }
    try {
        $destinationcategory = \core_course_category::get($this->arguments[1]);
    } catch(moodle_exception $me){
        cli_error('destination category does not exists');
    }
    $courses = $DB->get_records('course', array('category'=> $sourcecategory->id));
    if(count($courses)==0){
```

Déplacer un cours d'une catégorie à une autre

On teste

```
$ moosh -n -p /var/www/moodle_layerunistra404 category-move-courses-from-category-to-another
```

Déplacer un cours d'une catégorie à une autre

Autre approche

- Administration du site → Gestion des cours et catégories
- url : course/management.php
- via l'inspecteur → `Déplacer les cours sélectionnés vers…` - bouton `Déplacer`
- bulkmovecourses
- Dans le fichier course/managment.php
  - On cherche bulkmovecourses
  - → `\core_course\management\helper::move_courses_into_category`
  - `\core_course\management\helper::move_courses_into_category(`*moveto*`,courseids);`
  - librairie course/classes/management/helper.php

Déplacer un cours d'une catégorie à une autre

Autre code

```
public function execute() {
    global $DB, $CFG;
    require_once($CFG->dirroot.'/course/classes/category.php');
    require_once($CFG->dirroot.'/course/lib.php');
    require_once($CFG->dirroot.'/course/classes/management/helper.php');
    // Check that category ids exists.
    try {
        $sourcecategory = \core_course_category::get($this->arguments[0]);
    } catch (moodle_exception $me){
        cli_error('source category does not exists');
    }
    try {
        $destinationcategory = \core_course_category::get($this->arguments[1]);
    } catch(moodle_exception $me){
        cli_error('destination category does not exists');
    }
    $courses = $DB->get_records('course', array('category'=> $sourcecategory->id));
    if(count($courses)==0){
```

Bravo!

Retours et Questions

Liens et contacts

- Commandes Moosh
  - Commandes Moosh addtionnelles
  - Pull request en cours
- Contacts
  - Céline Pervès - Université de Strasbourg
  - Olivier Valentin - Université Lyon 3

Annexe -Installation atelier Windows

Volumes docker

- Dans un terminal windows
- Créer les dossiers volume utiles

```
mkdir C:\path_to_workshop_volume\mdldatas
mkdir C:\path_to_workshop_volume\code
mkdir C:\path_to_workshop_volume\moosh
mkdir C:\path_to_workshop_volume\database
```

Docker compose

Récupérer le fichier docker-compose.yml et le fichier env.sample

```
git clone https://git.unistra.fr/cperves/moosh-workshop.git
cd moosh-workshop
copy .env.sample .env
```

- Remplir le fichier .env créé
- notamment avec VOLUMES_ROOT_PATH=C:\path_to_workshop_volume\

Lancer le conteneur

- Récupérer l'image docker

```
docker pull registry.app.unistra.fr/cperves/moosh-workshop/moodle_0
```

- Premier lancement
  - Initialisation de la base de données Moodle

```
cd racine_dockercompose
docker login --username <votre_mail@institute.com>
SET INIT_PLF=1
docker compose up
# Si vous voulez lancer en silencieux
# docker compose up --detach
# Les autres fois, une fois la bdd initialisée
# SET INIT_PLF=0
```

Rentrer dans le conteneur

```
docker exec -it moosh-workshop-webserver-1 /bin/bash
```

- Dans /var/www/html on a le code Moodle
  - Que l'on retrouve dans le volume /path_to_workshop_volume/code
- Allez on continue là

Annexe - Moosh sous windows

Installation

- Attention l'auteur averti que toutes les commandes ne sont pas susceptible de fonctionner sous windows
  - issue github
- Un moyen d'installer moosh sous windows