Module Test :
Tests Unitaires sur Gitlab CI

Fabrice Bouquet et Anthony Dugois

2023 -- 2024

Université de Franche-Comté
Institut FEMTO-ST

DISC, Besançon, France

Présentation de Unitestor

Un robot qui se déplace sur une planète lointaine...

`https://disc.univ-fcomte.fr/m2gl-webRobot/Accueil.php`

Composants

- une horloge interne
- une batterie
- un panneau solaire
- un GPS
- un capteur de sol

Fonctionnalités

- déplacement : en fonction du sol et de la distance, plus ou moins d'énergie sera consommée
- rechargement de la batterie : le panneau solaire recharge la batterie de manière constante

Architecture du projet UniTestor

A télécharger sur Moodle : "UniTestor_phpUnit.zip"

- `Source/Vendor/Unitestor` : sources du projet
  - `Clock.php`
  - `Coordinates.php`
  - `LandSendor.php`
  - `Robot.php`
  - `Vector.php`
- `Test/Vendor/Unitestor` : tests unitaires manuels
  - `CoordinatesTest.php`

Gitlab CI

- Chaîne d'intégration continue (`https://docs.gitlab.com/ee/ci/`)
- Un *pipeline* est composé de *stages*.
- Un *stage* est composé de *jobs*.
- Le *runner* Gitlab exécute chaque *job* dans un environnement d'exécution prédéfini (shell, Docker, etc.).

On veut exécuter les tests unitaires lors d'un *push* sur le dépôt.

Gitlab CI

1. Récupérer les sources du projet "UniTestor_phpUnit.zip"
2. Créer un dépôt de test sur Gitlab
3. Pousser les sources du projet sur le dépôt

Gitlab CI

1. Créer un fichier .gitlab-ci.yml.
2. Ajouter un job :

```
1  test:
2    image: php:8.2-apache
3    script:
4      - phpunit tests
```

Par défaut, l'image Docker `php:8.2-apache` ne comprend pas PHPUnit. On va utiliser un script d'initialisation pour s'assurer que PHPUnit est bien présent.

Gitlab CI

Au début de .gitlab-ci.yml :

```
1  default:
2    before_script:
3      - bash ci/docker_install.sh > /dev/null
```

Puis créer le fichier ci/docker_install.sh :

```
1  #!/bin/bash
2
3  [[ ! -e /.dockerenv ]] && exit 0
4
5  set -xe
6
7  curl --location --output /usr/local/bin/phpunit \
8       "https://phar.phpunit.de/phpunit.phar"
9  chmod +x /usr/local/bin/phpunit
```

Gitlab CI

- Pousser les nouveaux fichiers
- S'assurer que les jobs s'exécutent correctement en allant dans Build, puis Jobs

Gitlab CI

```
Running with gitlab-runner 16.3.0 (8ec04662)
  on runner-2 5-iNxG2p, system ID: r_xgDUW0oKqU1p
Preparing the "docker" executor                                                    00:56
Using Docker executor with image php:8.2-apache ...
Pulling docker image php:8.2-apache ...
Using docker image sha256:dfde5c9d74f3851409f6060792a7fefe50f94d4bdb8bdd026f3b76149017c70e for php:8.2-apache with digest php@sha256:8a9bedf4d5f48ad7a33dd1613d65c6f134d0821bcab8fd4ca6e4a7382fed16a3 ...
Preparing environment                                                              00:09
Running on runner-5-inxg2p-project-24-concurrent-0 via 16a1ab635b7e...
Getting source from Git repository                                                 00:07
Fetching changes with git depth set to 20...
Initialized empty Git repository in /builds/cr700-gitlab/adugois6/test-robot-tu/.git/
Created fresh repository.
Checking out ef034354 as detached HEAD (ref is main)...
Skipping Git submodules setup
Executing "step_script" stage of the job script                                    00:08
Using docker image sha256:dfde5c9d74f3851409f6060792a7fefe50f94d4bdb8bdd026f3b76149017c70e for php:8.2-apache with digest php@sha256:8a9bedf4d5f48ad7a33dd1613d65c6f134d0821bcab8fd4ca6e4a7382fed16a3 ...
$ bash ci/docker_install.sh > /dev/null
+ curl --location --output /usr/local/bin/phpunit https://phar.phpunit.de/phpunit.phar
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138    0   138    0     0    752      0 --:--:-- --:--:-- --:--:--   754
100 4561k  100 4561k    0     0  11.9M      0 --:--:-- --:--:-- --:--:-- 11.9M
+ chmod +x /usr/local/bin/phpunit
$ phpunit tests
PHPUnit 10.3.5 by Sebastian Bergmann and contributors.
Runtime:       PHP 8.2.10
..                                                                  2 / 2 (100%)
Time: 00:00.224, Memory: 22.46 MB
OK (2 tests, 4 assertions)
Cleaning up project directory and file based variables                             00:05
Job succeeded
```

Gitlab CI

- Refaire la même chose, mais cette fois pour les tests selenium
- tester sur le robot : `https://disc.univ-fcomte.fr/m2gl-webRobot/Accueil.php`
- L'image à utiliser : `docker.io/markhobson/maven-chrome:jdk-8`
- `command: mvn -f /usr/src/app/pom.xml test`

Calculer/afficher la couverture

On veut connaître à tout moment le taux de couverture des tests. Gitlab permet d'extraire cette valeur à partir de la sortie textuelle de nos jobs.
Tout d'abord, on commence par calculer le taux de couverture avec PHPUnit. Pour cela, installer l'extension PHP Xdebug. Dans ci/docker_install.sh ajouter :

```
1 ...
2 pecl install xdebug
3 docker-php-ext-enable xdebug
```

Calculer/afficher la couverture

Puis, dans .gitlab-ci.yml :

```
1  test:
2    image: php:8.2-apache
3    script:
4      - XDEBUG_MODE=coverage phpunit tests/units
5          --coverage-filter tests/units
6          --do-not-cache-result
7          --log-junit phpunit-report.xml
8          --coverage-cobertura phpunit-coverage.xml
9          --coverage-text --colors=never
10   artifacts:
11     when: always
12     reports:
13       junit: phpunit-report.xml
14       coverage_report:
15         coverage_format: cobertura
16         path: phpunit-coverage.xml
17   coverage: '/^\s*Lines:\s*\d+.\d+\%/'
```

Calculer/afficher la couverture

La ligne de commande PHPUnit permet de générer les rapports de couverture sous différents formats, plus tard utilisés par Gitlab.
La commande `artifacts` indique à Gitlab les fichiers à sauvegarder en tant qu'artéfacts des jobs (autrement, les rapports seraient simplement détruits à la fin du job).
Enfin, la commande `coverage` indique à Gitlab comment extraire le taux de couverture depuis la sortie textuelle du job.
Après plusieurs commits, on peut visualiser l'évolution du taux de couverture en allant dans Analyze puis Repository analytics.

`https://docs.gitlab.com/ee/ci/testing/code_coverage.html`
`https://docs.gitlab.com/ee/ci/testing/test_coverage_visualization.html`