



Module Test : Bien écrire les tests de son composant logiciel PHP

UNIVERSITÉ DE
FRANCHE-COMTÉ

Fabrice Bouquet

UNIVERSITÉ DE
FRANCHE-COMTÉ

2019 -- 2020

Université de Franche-Comté
Institut FEMTO-ST

DISC, Besançon, France



Présentation de Unitestor



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

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



- ▶ Source/Vendor/Unittestor : sources du projet
 - ▶ Clock.php
 - ▶ Coordinates.php
 - ▶ LandSender.php
 - ▶ Robot.php
 - ▶ Vector.php
- ▶ Test/Vendor/Unittestor : tests unitaires manuels

Présentation phpUnit



- ▶ Téléchargeable à <https://phpunit.de>
- ▶ Un framework xUnit PHP
- ▶ Simple : utilisation et déploiement facile
- ▶ Moderne : utilise PHP 7.3+ et compatibilité avec les outils industriels standards
- ▶ Propose des mocks, bouchons, assert*
- ▶ Documentation :
<https://phpunit.readthedocs.io/fr/latest/>

PHPUnit

Configuration



(en considérant que phpunit.phar se trouve à la racine du projet)

- ▶ Vérifier la version (au moins 8.3) de phpUnit php
`phpUnit.phar --version`
- ▶ Pour lancer les tests d'un fichier particulier : php
`phpunit.phar --bootstrap source/.../Clock.php
tests/.../units/ClockTest.php`
- ▶ Sinon pour tous les tests lancer php `phpunit.phar tests`

Étape 1 - Premiers pas atoum

Écriture des tests pour : Coordinates.php

```
<?php
```

```
use PHPUnit\Framework\TestCase;
```

```
class CoordinatesTest extends TestCase {
```

```
    public function testXY ( ) {
```

```
        $x = 7.0;
```

```
        $y = 42.0;
```

```
        $coordinates = new Coordinates($x, $y);
```

```
        $this->assertEquals($coordinates->getX(),$x);
```

```
        $this->assertEquals($coordinates->getY(),$y);
```

```
    }
```

```
}
```

```
>
```

A faire



- ▶ Réalisation du test pour vérifier le "Cast" (int/float) des coordonnées X et Y. Est-ce que c'est bien un float qui est renvoyé et à quoi est-il égal.
- ▶ Quelle est la différence entre `assertEquals` et `assertSame`
- ▶ `Vector.php` : tester les *getters* et le calcul de la longueur d'un vecteur (avec et sans arrondi).

Étape Mock



Mes premiers pas avec les mocks !

- ▶ Les mocks permettent de modifier le comportement d'une méthode.
- ▶ Clock.php, dans la fonction testReset, on mock l'objet Clock pour changer le comportement de la fonction getCurrentTime pour qu'elle renvoie une valeur fixe (contenue dans une variable nextTime).
- ▶ Ainsi la fonction reset ne remet pas l'horloge à zéro mais à cette valeur.


A faire :

- ▶ Clock.php : tester la méthode getDifference()
- ▶ Landsensor.php : tester la méthode getNeededEnergy() (forcer le random avec la graine à 0, mock de fonction getLength de la classe vector et désactiver le constructeur de la classe)

Pré-requis Maven



- ▶ Installer le plugin Maven (si besoin)
 - ▶ Aller dans le Market Place
 - ▶ Rechercher Maven
 - ▶ Sélectionner le plugin suivant




Maven Integration for Eclipse (Luna) 1.5.0

m2e provides comprehensive Maven integration for Eclipse. You can use m2e to manage both simple and multi-module Maven projects, execute Maven builds via the... [more info](#)

by Eclipse.org, EPL
[maven java build development](#)

★ 3

 Installs: **18,2K** (3 097 last month)

Update

Uninstall

Pensez à configurer le nexus local dans le .m2

Test fonctionnel


- ▶ L'objectif de cette étape est de simuler le comportement du navigateur pour :
 - ▶ Tester le comportement de la page
 - ▶ Contrôler que le résultat obtenu est conforme au résultat escompté suite à une action utilisateur
- ▶ Les tests sont décrits en Java et s'appuient sur junit (utilisation d'assertions)
- ▶ Bibliothèques existantes : Selenium, HtmlUnit...
- ▶ Créer un projet Maven/ module et ajouter les 3 dépendances :

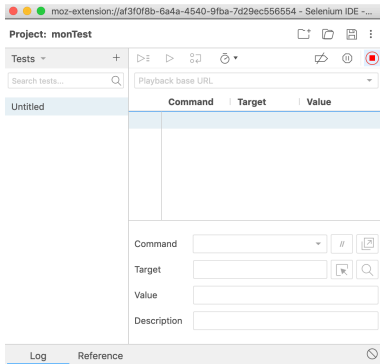
```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.5.3</version>
  </dependency>
  <dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>3.0.0</version>
  </dependency>
</dependencies>
```

Etape 1 - Installer / Configurer







Deux outils : Selenium IDE ou Katalon Recorder

- ▶ Installer Selenium IDE :
Menu : 'Outils → Modules
Complémentaire → Extensions'
- ▶ Lancer Selenium IDE : icône 



Etape 2 - Enregistrer un test




- ▶ Enregistrer  et aller sur l'url et :
`https://disc.univ-fcomte.fr/m2gl-webRobot/Accueil.php`
- ▶ Poser le robot en appuyant sur le bouton « Nouvelle carte »
- ▶ Déplacer le robot avec les boutons :
 - ▶ « Avancer »
 - ▶ « Reculer »
 - ▶ « Tourner à droite »
 - ▶ « Tourner à gauche »
- ▶ Arrêter l'enregistrement du test avec le bouton 
- ▶ Utiliser le bouton dans Selenium pour rejouer le test :  
- ▶ Est-ce que ça marche tout le temps? Reproductibilité du test
... `https://disc.univ-fcomte.fr/m2gl-webRobot/Accueil.php?seed=1`

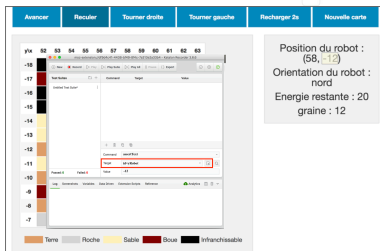
Etape 3 - Repérer les éléments



2 approches pour repérer les éléments :

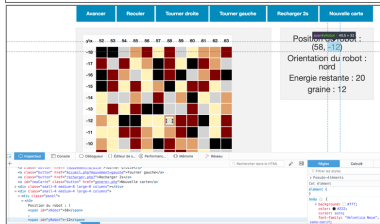
1. Dans Selenium à la fin du test :

- ▶ Ajouter la commande « assertText »
- ▶ Choisir la Cible en cliquant sur le bouton 
- ▶ Puis cliquer sur la page pour trouver la zone désirée
- ▶ Donner la Valeur attendue
- ▶ Rejouer le test



2. Dans Firefox, utiliser l'inspecteur :
Outils

- Développement web
- Inspection



Etape 4 - Transformation en JUnit



- ▶ Dans Selenium IDE sur le test click droit et Export et choisir le format : Java JUnit
- ▶ Sauver le fichier dans vos sources de projet de votre IDE
- ▶ Rafraichir l'IDE vos sources et corriger les dépendances
- ▶ Ajouter l'initialisation du Driver : `WebDriverManager.firefoxdriver().setup();` possible aussi `chromedriver()`, `operadriver()`, `phantomjs()`, `edgedriver()`...
- ▶ Il peut être nécessaire de forcer la version 1.8 dans maven :

```
<properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

- ▶ Il peut être nécessaire d'ajouter un délais dans l'exécution :

```
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
```

- ▶ Ajouter des vérifications :

- ▶ Par exemple vérifier que la valeur de x vaut 0
- ▶ Il faut utiliser :

- ▶ Méthode de test : `Assert.assertEquals(String 1, String 2)`
- ▶ Récupération de la valeur de "x" :

```
driver.findElement(By.id("x")).getText()
```

Exercise



Créer vos test fonctionnel sur l'Application webRobot

- ▶ Connecter vous à l'url avec choix de la graine : `https://disc.univ-fcomte.fr/m2gl-webRobot/Accueil.php?seed=valeurGrain`
- ▶ Vérifier que le comportement du robot est bien conforme aux attentes, tester :
 - ▶ Le déplacement du robot (1 test par direction)
 - ▶ Déplacement du robot et retour à la case d'origine
 - ▶ La remise à l'état initial du robot après avoir fait 5 déplacements

Autre WebDriver : htmlUnit



- ▶ Pilote pour appel sans ouverture de navigateur
- ▶ Simplifie le test d'application web sans navigateur (build continue)
- ▶ Téléchargeable à <http://htmlunit.sourceforge.net> ou mettre la dépendance maven (retirer la dépendance de selenium-java) :

```
<dependency>  
    <groupId>org.seleniumhq.selenium</groupId>  
    <artifactId>htmlunit-driver</artifactId>  
    <version>2.27</version>  
</dependency>
```

- ▶ Une extension du framework junit et Selenium

Étape 2 - Dans JUnit



```
import org.junit.*;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;

public class RobotSeleniumTest {
    private HtmlUnitDriver driver;
    private String baseUrl;
    private String valX;

    @Before
    public void setUp() throws Exception {
        driver = new HtmlUnitDriver();
        baseUrl = "https://disc.univ-fcomte.fr";
    }

    @Test
    public void testRobotSelenium() throws Exception {
        driver.get(baseUrl+"/m2gl-webRobot");
        valX = driver.findElementById("x").getText();
        Assert.assertEquals("Erreur sur abscisse", valX,"0");
    }
}
```

Étape 2 - Dans JUnit



- Suppression des logs :

```
java.util.logging.Logger.getLogger("com.gargoylesoftware")  
    .setLevel(Level.OFF);  
System.setProperty("org.apache.commons.logging.Log",  
    "org.apache.commons.logging.impl.NoOpLog");
```

- Problème javascript/version spécifique navigateur :

```
driver = new HtmlUnitDriver() {  
    @Override protected WebClient newWebClient(BrowserVersion version) {  
        WebClient webClient = super.newWebClient(version);  
        webClient.getOptions().setThrowExceptionOnScriptError(false);  
        return webClient; } };
```

SquashTA



- ▶ Téléchargeable à <http://www.squashTest.org>
- ▶ Robot d'exécution construit sur Jenkins
- ▶ Lien avec gestionnaire de tests (Squash-TM)
- ▶ Plugin eclipse pour l'écriture des tests

Configurations Job Squash-TA 1/8



Jenkins » Tous »

Donner le nom du job →

Saisissez un nom

WebRobot_fs

» Champ obligatoire

Job de type free-style

 **Construire un projet free-style**
Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.

 **Construire un projet maven**
Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.

Configurations Job Squash-TA 2/8



Jenkins

1 rechercher

Jenkins » WebRobot_fs »

General Gestion de code source Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

Nom du Projet WebRobot_fs

Description

[Plain text] [Prévisualisation](#)

☒ Activer integration avec Squash TM

☒ Ce build a des paramètres

Indiquer l'intégration avec Squash TM

Configurer les paramètres

Ajouter un paramètre

- CVS Symbolic Name Parameter
- List Subversion tags (and more)
- Paramètre "Mot de passe"
- Paramètre String
- Paramètre booléen
- Paramètre choix
- Paramètre d'exécution
- Paramètre d'identifiants
- Paramètre fichier
- Paramètre texte

Avancé...

Configurations Job Squash-TA 3/8



Jenkins > WebRobot_fs >

General Gestion de code source Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

☒ Ce build a des paramètres

Paramètre String

Nom: operation

Valeur par défaut:

Description: This parameter specifies the goal to execute. Supported values are:
* run (runs Squash TA tests)
* list (lists all Squash TA tests offered by the job)

[Plain text] [Prévisualisation](#)

Paramètre String

Nom: externalJobId

Valeur par défaut:

Description: This parameter is an id guaranteed by the external caller to be a unique id for the build. It is used to tag the build in order to retrieve results

[Plain text] [Prévisualisation](#)

Ajouter un paramètre

Configurations Job Squash-TA 4/8



Jenkins > WebRobot_fs >

General Gestion de code source Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

Paramètre String [X] [?]

Nom: notificationURL [?]

Valeur par défaut: file://dev/hull [?]

Description: REST StatusUpdate URL where to send status update events [?]

[Plain text] [Prévisualisation](#)

Paramètre String [X] [?]

Nom: testList [?]

Valeur par défaut: */*.ta,*/*.bd,*/*.test [?]

Description: This parameter helps to set the list of test to execute. A filter or an ordered list could be given. If the test list is given through a file then you should set this parameter to (file:testsuite.json) [?]

[Plain text] [Prévisualisation](#)

Ajouter un paramètre ▾

Configurations Job Squash-TA 5/8



Jenkins » WebRobot_fs »

General Gestion de code source Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

Paramètre fichier

Emplacement du fichier: testsuite.json

Description: testsuite.json

[Plain text] [Prévisualisation](#)

Ajouter un paramètre

☐ GitHub project

☒ Supprimer les anciens builds

Strategy: Log Rotation

Nombre de jours de conservation des builds:

si non vide, les enregistrements de build seront conservés au maximum ce nombre de jours

Nombre maximum de builds à conserver: 5

si non vide, pas plus de ce nombre de builds ne sera conservé

[Avancé...](#)

☐ Throttle builds

☐ Désactiver le projet

☐ Exécuter des builds simultanément si nécessaire

JDK: JDK_7

Le JDK à utiliser pour ce projet



Configurations Job Squash-TA 6/8



Jenkins > WebRobot_fs >

General **Gestion de code source** Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

Subversion

Modules

Repository URL <https://disc.univ-fcomte.fr/m2gl-svn/bouquet/RobotSquash>

Credentials fbouquet/***** Ajouter

Local module directory .

Repository depth infinity

Ignore externals ☒

Add module...

Additional Credentials Add additional credentials...

Stratégie de checkout Utiliser 'svn update' autant que possible

Utiliser 'svn update' aussi souvent que possible rendra la build plus rapide. Mais cela entraîne des artefacts de la précédente build qui vont rester quand la nouvelle build commencera.

Navigateur de la base de code (Auto)

Avancé...

Lien du SVN

Ajouter vos identifiants

Configurations Job Squash-TA 7/8



Build

Ajouter une étape au build ▾

- Appeler Ant
- Exécuter un script shell
- Exécuter une ligne de commande batch Windows
- Invoker les cibles Maven de haut niveau**
- Provide Configuration files
- SeleniumHQ htmlSuite Run
- Set build status to "pending" on GitHub commit

Configurations Job Squash-TA 8/8



Jenkins ▾ » WebRobot_fs ▾ »

General Gestion de code source Ce qui déclenche le build Environnements de Build **Build** Actions à la suite du build

Build

☐ Invoquer les cibles Maven de haut niveau ✕ ?

Version de Maven Maven ▾

Cibles Maven

```
-Dta.test.suite="${testList}"
-Dstatus.update.events.url="${notificationURL}"
-Dsquash.ta.external.id="${externalJobId}"
-Djobname="${JOB_NAME}"
-Dhostname="${HOSTNAME}"
-Dsquash.ta.conf.file="${CONF_HOME}/taLinkConf.properties"
-Dta.tmcallback.reportbaseurl="${JENKINS_URL}job"
-Dta.tmcallback.jobexecutionid="${BUILD_NUMBER}"
-Dta.tmcallback.reportname=Squash_TA_HTML_Report
org.squashtest.ta::squash-ta-maven-plugin::${operation}
```

Avancé...

Ajouter une étape au build ▾

Actions à la suite du build

☐ Publier les résultats de test sur Squash TM ✕

Ajouter une action après le build ▾

Piloter les tests



1. Ouvrir dans l'environnement le canevas proposé dans moodle
2. Lancer les 2 tests dans votre environnement (webRobotHeadless, webRobotSelenium)
3. Ecrire des tests pour la gestion de l'énergie
4. Développer les scripts « TA » correspondant aux tests associés aux exigences en utilisant le canevas proposé :
 - ▶ Les fichiers Junit sont à mettre dans :
`<projet>/src/squashTA/resources/selenium/java`
 - ▶ Les fichiers script TA sont à mettre dans :
`<projet>/src/squashTA/tests`

Plus avec Base de données



Les scripts TA peuvent utiliser DB-Unit (cf. initDB.ta) pour traiter les éléments de base de données, les fichiers complémentaires sont mis dans <projet>/src/squashTA/resources par exemple pour :

- ▶ La suppression d'éléments comme dans `cleaning-blackBoord.xml`
- ▶ Le chargement d'éléments comme dans `blackBoard.xml`
- ▶ Assertion d'éléments en tables comme dans `mapGenerator.xml`
- ▶ La configuration de la base de donnée est dans `<projet>/src/squashTA/targets/WebRobot.properties`

Arborescence

