Tests fonctionnels automatisés

L'automatisation des tests fonctionnels consiste à créer des scénarii de tests qui pourront ensuite être reproduits à la demande (exécutés) au cours du développement.

- La réalisation d'un scénario permet de mieux identifier le besoin exprimé
- La mise en place du test relatif à une fonctionnalité permet de tester cette fonctionnalité (en cours et en fin d'implémentation)
- L'exécution d'une suite de tests permet de vérifier la non-régression d'un projet suite à une modification ou l'introduction d'une nouvelle fonctionnalité.

-- Mise en place des outils pour PHP

-- Composer

Composer est un gestionnaire de paquets compatible GIT permettant d'installer ou de mettre à jour les librairies incluses dans un projet à partir d'un fichier de configuration **composer.json**, déclarant les dépendances du projet.

Installation

Sous Windows :

- télécharger et installer Composer-Setup.exe
- Ajouter le dossier d'installation de composer dans la variable PATH de windows pour pouvoir exécuter composer directement en ligne de commande.

Vérifier l'installation :
Dans un terminal : Frapper **composer -v** puis Entrée ↵

-- PHPUnit et WebDriver

PHPUnit va permettre de réaliser des tests unitaires (différents des tests fonctionnels).
Pour la partie fonctionnelle, nous utiliserons **Selenium Server** + **Facebook WebDriver**, pour émuler les interactions utilisateur dans un navigateur.

Créer le fichier **composer.json** à la racine de votre projet :

```
{
    "require-dev": {
        "facebook/webdriver": "dev-master",
        "phpunit/phpunit": "~4.8"
    }
}
```

Dans le terminal :
A partir du dossier de votre projet, Frapper composer install puis Entrée ↵

Vérifiez l'installation correcte des packages dans le dossier **vendor** du projet.

-- PhpUnit

PhpUnit permet de réaliser des tests unitaires, relatifs à la bonne exécution de parties de code (fonction ou méthode).

-- Configuration

Créer un dossier **tests/** à la racine du projet à tester.

Créer le fichier TestHelper.php adapté qui permettra de faire les inclusions nécessaires avant le lancement de PhpUnit.

Créer le fichier **PHPunit.xml** dans le même dossier, faisant référence au fichier TestHelper.php, et permettant de lancer tous les tests inclus dans le dossier courant (./) et ses sous-dossiers :

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="TestHelper.php"
         backupGlobals="false"
         backupStaticAttributes="false"
         verbose="true"
         colors="true"
         convertErrorsToExceptions="true"
         convertNoticesToExceptions="true"
         convertWarningsToExceptions="true"
         processIsolation="false"
         stopOnFailure="false"
         syntaxCheck="true">
    <testsuite name="Testsuite">
        <directory>./</directory>
    </testsuite>
</phpunit>
```

-- Premier test

Créer un premier test (dans le cas présent, on utilise le micro-framework):

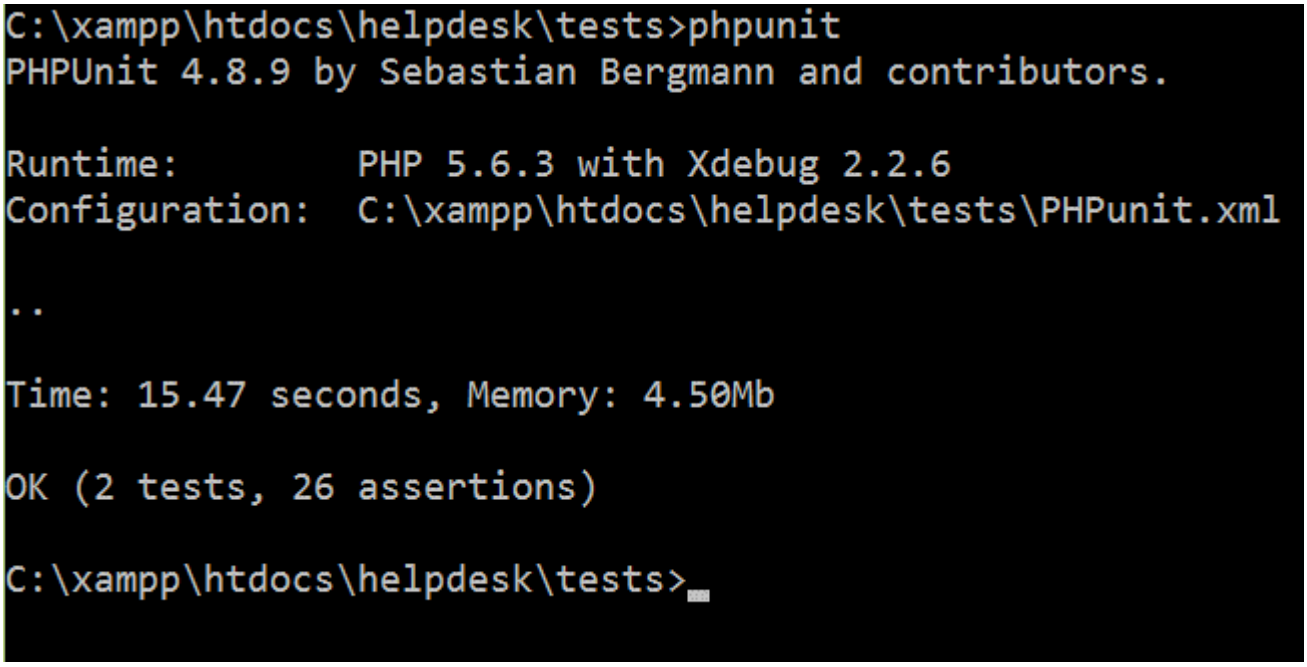
```
<?php
class PHPUnitTest extends \PHPUnit_Framework_TestCase {
    private $variable;
    /* (non-PHPdoc)
     * @see PHPUnit_Framework_TestCase::setUp()
     */
    protected function setUp() {
        $this->variable=1;
    }

    public function testIncVariable(){
        $this->assertEquals($this->variable, 1);
        for($i=0;$i<10;$i++){
            $this->variable+=1;
        }
        $this->assertEquals(11, $this->variable);
    }

    public function testVariable(){
        $this->assertEquals($this->variable, 1);
    }
    /* (non-PHPdoc)
     * @see PHPUnit_Framework_TestCase::tearDown()
     */
    protected function tearDown() {
        $this->variable=0;
    }
}
```

Aller en invite de commandes dans le dossier tests du projet et exécuter :

phpunit puis Entrée ↵



-- Caractéristiques d'une classe de test PHPunit

1. Une classe PHPunit dérive de **PHPUnit_Framework_TestCase**. Son nom doit commencer par **Test...**
2. Les méthodes de test contenues dans la classe sont publiques et se terminent par **...test**
3. Les méthodes à dériver :
    1. setUpBeforeClass ⇒ exécutée une seule fois avant l'appel du constructeur de la classe
    2. setUp ⇒ exécutée avant chaque test (méthode se terminant par test

-- Selenium

Selenium permet de réaliser des tests fonctionnels. Il permet de réaliser ce que l'utilisateur pourrait entreprendre, pour mettre en oeuvre une fonctionnalité, puis de tester les résultats obtenus.

-- Configuration