



# Programmation orientée objet en PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [POO PHP](#) ▶ [Les tests avec PHPUnit](#) ▶ [Premier test](#)

## Premier test

**PHPUnit** est une bibliothèque qui va nous permettre de tester nos composants logiciels. Nous pourrons ainsi vérifier le bon fonctionnement de notre application à plusieurs niveaux :

- au niveau du composant lui-même : on parle ici de test unitaire, on essaie de tester la plus petite unité de code possible (souvent une fonction)
- au niveau de plusieurs composants : on vérifie ici que l'intégration de nouveaux composants n'entraîne pas un dysfonctionnement des autres composants : on parle ici de test d'intégration
- au niveau de l'application elle-même : on vérifie le bon fonctionnement d'une fonctionnalité entière : on parle ici de test fonctionnel

Les tests permettent aussi de mettre en évidence des problèmes en terme de performance, de robustesse, de fiabilité, ou encore de vulnérabilité.

L'installation de **PHPUnit** sera assurée par un outil de gestion de dépendances appelé **Composer**. Pour le dire vite, Composer va nous permettre de télécharger les bibliothèques dont nous aurons besoin dans notre application. Ces bibliothèques seront renseignées dans un fichier JSON. L'installation de Composer peut se faire de plusieurs façons, soit sur votre système, soit dans votre projet sous forme d'une archive phar. Dans notre projet nous utiliserons l'archive phar.

Nous créons un projet TestProject dans notre répertoire personnel, et nous mettons à la racine de notre projet l'archive composer.phar téléchargée à partir du site :



Dans un terminal, nous nous plaçons dans le répertoire de notre projet et exécutons la commande *php composer.php*.

```
Terminal
Fichier Edition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/TestProject$ php composer.phar

Composer version 1.0.0-dev (c33c5196b19c77ea89b05a81c573d88543d3a93d) 2014-10-15 13:11:07

Usage:
  [options] command [arguments]

Options:
  --help                -h Display this help message.
  --quiet               -q Do not output any message.
  --verbose             -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
  --version            -V Display this application version.
  --ansi               -A Force ANSI output.
  --no-ansi            -A Disable ANSI output.
  --no-interaction     -n Do not ask any interactive question.
  --profile            -p Display timing and memory usage information
  --working-dir        -d If specified, use the given directory as working directory.

Available commands:
  about                Short information about Composer
  archive              Create an archive of this composer package
  browse              Opens the package's repository URL or homepage in your browser.
  clear-cache          Clears composer's internal package cache.
  config              Set config options
  create-project       Create new project from a package into given directory.
  depends             Shows which packages depend on the given package
  diagnose            Diagnoses the system to identify common errors.
  dump-autoload        Dumps the autoloader
  dumpautoload         Dumps the autoloader
  global              Allows running commands in the global composer dir ($COMPOSER_HOME).
  help                Displays help for a command
  home                Opens the package's repository URL or homepage in your browser.
  init                Creates a basic composer.json file in current directory.
  install             Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  licenses            Show information about licenses of dependencies
  list                Lists commands
  remove              Removes a package from the require or require-dev
  require             Adds required packages to your composer.json and installs them
  run-script          Run the scripts defined in composer.json.
  search              Search for packages
  self-update          Updates composer.phar to the latest version.
  selfupdate          Updates composer.phar to the latest version.
  show                Show information about packages
  status              Show a list of locally modified packages
  update             Updates your dependencies to the latest version according to composer.json, and updates the composer.lock file.
  validate            Validates a composer.json

patrice@patrice-laptop:~/TestProject$
```

Nous obtenons ainsi la liste des commandes de Composer.

Nous éditons à la racine de notre projet un fichier composer.json qui indiquera toutes les dépendances que devra

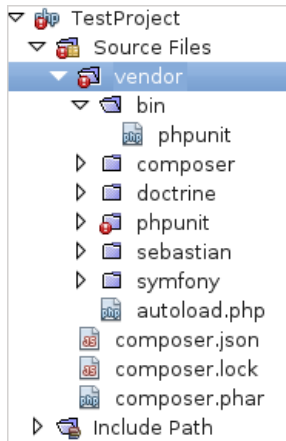
gérer Composer :

```

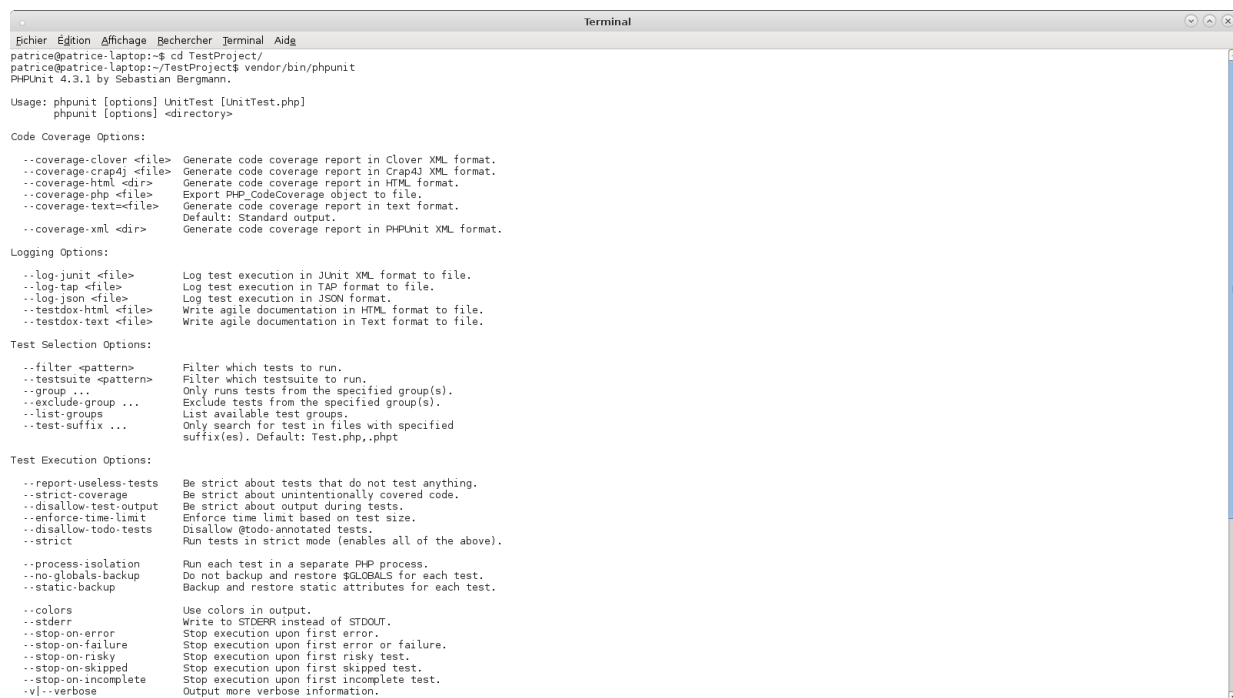
1  {
2      "require-dev": {
3          "phpunit/phpunit": "4.3.*"
4      }
5  }

```

Dans le fichier `composer.json`, le wildcard `"*"` signifie la version la plus élevée. Dans un terminal nous exécutons la commande `php composer.phar install` et laissons Composer installer nos dépendances. Un répertoire `vendor` contenant nos dépendances a été créé au sein de notre projet :



Dans le répertoire `vendor/bin` nous trouvons un exécutable `phpunit`. C'est cet exécutable qui va nous permettre de lancer nos tests. Si, dans le terminal, nous exécutons la commande `vendor/bin/phpunit` alors s'afficheront les commandes de `phpunit`. Quand nous exécutons cette commande nous sommes bien à la racine de notre projet. À chaque fois que nous lancerons nos tests nous veillerons à être à la racine de notre projet.



Nous allons créer une classe `Calculatrice` dans un répertoire `fr/fyigrane`, et tester ses méthodes. Nous allons tester les plus petites unités de code possible, nous faisons donc du test unitaire.

```

1  <?php
2
3  // fr/fyigrane/Calculatrice.php
4
5  namespace fr\fyigrane;
6
7  class Calculatrice {
8

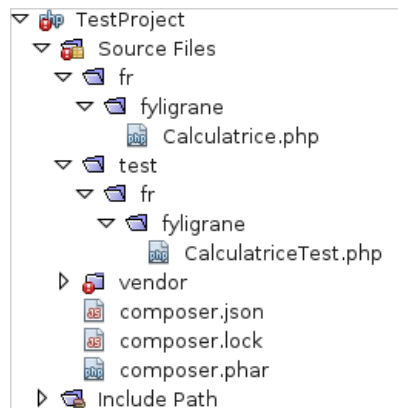
```

```

9      public function additionner($nb1, $nb2) {
10          return $nb1 + $nb2;
11      }
12
13  }
```

Cette classe ne contient pour l'instant qu'une méthode `additionner()` qui prend en argument deux entiers et qui renverra bien sûr la somme de ces deux entiers. Tester le bon fonctionnement de cette méthode revient à vérifier que l'entier retourné est bien égal à la somme des deux entiers passés en argument.

Nous allons créer dans un répertoire `test/fr/fyligrane`, une classe `CalculatriceTest` héritant de `PHPUnit_Framework_TestCase`.



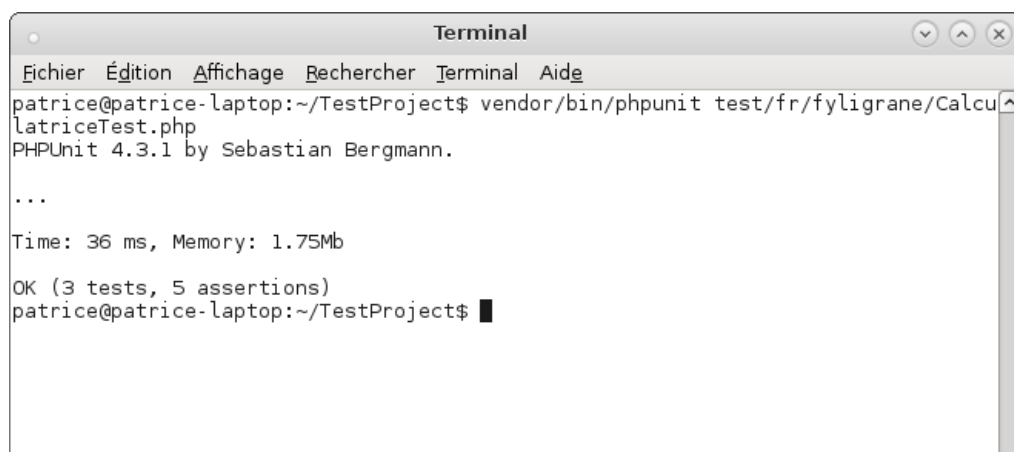
Dans cette classe nous allons créer une méthode `testAdditionner()` qui effectuera le test de notre méthode `additionner()`. La classe `PHPUnit_Framework_TestCase` possède une méthode `assertEquals()` qui permet de vérifier l'égalité de deux valeurs ou variables : `assertEquals(mixed $expected, mixed $actual[, string $message = ''])`.

Pour exécuter la fonction `additionner()` nous aurons évidemment besoin d'une instance de `Calculatrice`.

```

1  <?php
2
3  // test/fr/fyligrane/CalculatriceTest.php
4
5  include_once './fr/fyligrane/Calculatrice.php';
6
7  use fr\fyligrane\Calculatrice;
8
9  class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11      public function testAdditionner() {
12          $calc = new Calculatrice();
13          $result = $calc->additionner(15, 25);
14          $this->assertEquals(40, $result);
15      }
16
17  }
```

Pour exécuter ce test nous allons utiliser l'exécutable `phpunit` et lui passer en paramètre le fichier à tester :





En cas d'erreur, phpunit affiche quelques informations :

```

1 <?php
2
3 // test/fr/fyigrane/CalculatriceTest.php
4
5 include_once './fr/fyigrane/Calculatrice.php';
6
7 use fr\fyigrane\Calculatrice;
8
9 class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     public function testAdditionner() {
12         $calc = new Calculatrice();
13         $result = $calc->additionner(15, 25);
14         $this->assertEquals("azerty", $result);
15     }
16
17 }

```

```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/TestProject$ vendor/bin/phpunit test/CalculatriceTest.php
PHPUnit 4.3.1 by Sebastian Bergmann.

F

Time: 34 ms, Memory: 2.00Mb

There was 1 failure:

1) CalculatriceTest::testAdditionner
Failed asserting that 40 matches expected 'azerty'.

/home/patrice/TestProject/test/CalculatriceTest.php:14

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
patrice@patrice-laptop:~/TestProject$

```

Un test peut comporter plusieurs "assertions" ; dans ce cas le test ne sera concluant que si toutes les assertions se vérifient :

```

1 <?php
2
3 // test/fr/fyigrane/CalculatriceTest.php
4
5 include_once './fr/fyigrane/Calculatrice.php';
6
7 use fr\fyigrane\Calculatrice;
8
9 class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     public function testAdditionner() {
12         $calc = new Calculatrice();
13         $result = $calc->additionner(15, 25);
14         $this->assertEquals(40, $result);

```

```

15     // stupide... Juste pour l'exemple...
16     $this->assertNotFalse($result);
17 }
18
19 }

```

```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/TestProject$ vendor/bin/phpunit test/CalculatriceTest.php
PHPUnit 4.3.1 by Sebastian Bergmann.

.

Time: 33 ms, Memory: 1.75Mb

OK (1 test, 2 assertions)
patrice@patrice-laptop:~/TestProject$

```

La classe `PHPUnit_Framework_TestCase` possède un nombre important d'assertions sous la forme de méthode `assert*****()`.

Pour que phpunit sache quelles méthodes correspondent à un test, il faut que le nom de ces méthodes commence par test (ici `testAdditionner()`), ou alors que la méthode porte l'annotation `@test` (les annotations se situent toujours dans des commentaires de la forme `/**.....*/`).

```

1 <?php
2
3 // test/fr/fyigrane/CalculatriceTest.php
4
5 include_once './fr/fyigrane/Calculatrice.php';
6
7 use fr\fyligrane\Calculatrice;
8
9 class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     /**
12      * @test
13      */
14     public function additionner() {
15         $calc = new Calculatrice();
16         $result = $calc->additionner(15, 25);
17         $this->assertEquals(40, $result);
18         // stupide... Juste pour l'exemple...
19         $this->assertNotFalse($result);
20     }
21
22 }

```

Fin

## NAVIGATION

[Accueil](#)



■ [Ma page](#)[Pages du site](#)[Mon profil](#)[Cours actuel](#)[POO PHP](#)[Participants](#)[Généralités](#)[La programmation orientée objet : premiers pas](#)[L'héritage](#)[Les interfaces](#)[Le typage](#)[Les namespaces](#)[Les exceptions](#)[Les bases de données avec PDO](#)[Les tests avec PHPUnit](#)[Premier test](#)[L'environnement du test : test fixtures](#)[Test des exceptions](#)[T.P. premier test](#)[Test des dépendances](#)[Test des bases de données](#)[Petite application version 2](#)[Petite application version 3](#)[Mes cours](#)**ADMINISTRATION**[Administration du cours](#)[Réglages de mon profil](#)Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))[POO PHP](#)