



Initiation aux tests unitaires et au couverture de code

1

Filière: Développement digital – option web full stack

Module: Approche agile

Formatrice: Asmae YOUALA

PLAN

- Introduction aux test unitaires
- PHPUnit
- Intégration avec SonarQube (code coverage)

Tests unitaires: Définition

- ▶ « le test unitaire est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme. » wikipédia
- ▶ C'est l'une des procédures mises en œuvre dans le cadre d'une méthodologie de travail agile.
- ▶ Le test unitaire consiste à isoler une partie du code et à **vérifier qu'il fonctionne parfaitement**. Il s'agit de petits tests qui valident l'attitude d'un objet et la logique du code.



Avantages des tests unitaires

- Le test unitaire révèle si la logique derrière le code est appropriée et **fonctionnera dans tous les cas**.
- Il améliore la **lisibilité** du code et aide les développeurs à comprendre le code de base, ce qui facilite la mise en œuvre des modifications plus rapidement.
- Des tests unitaires bien conduits sont également de bons outils pour la **documentation** du projet.
- Les tests sont effectués en **un peu plus de quelques millisecondes**, ce qui vous permet d'en réaliser des centaines en très peu de temps.
- Le *test unitaire* permet au développeur de **remanier le code** ultérieurement et de s'assurer que le module continue à fonctionner correctement.
- La **qualité finale du code** s'améliorera parce qu'il s'agira en fin de compte d'un code propre et de haute qualité grâce à ces essais continus.
- Puisque le test unitaire divise le code en petits fragments, il est possible de **tester différentes parties du projet** sans avoir à attendre que d'autres parties soient terminées.

Tests unitaires: Outils

- Le processus de test unitaire peut être effectué manuellement, bien qu'il soit plus courant d'**automatiser la procédure à l'aide de certains outils**.
 - ✓ De nombreuses options sont disponibles, qui varient en fonction du langage de programmation utilisé. Voici quelques exemples de types d'outils, qui vous aideront dans les tests.
 - ✓ xUnit : c'est un outil de test unitaire à utiliser sur le framework **.NET**.
 - ✓ JUnit : il s'agit d'un ensemble de bibliothèques pour le test unitaire sur des applications **Java**.
 - ✓ NUnit : NUnit 3 – qui était initialement porté depuis JUnit – a été complètement réécrit pour lui fournir de nouvelles fonctionnalités et la prise en charge d'une large gamme de plateformes **.NET**.
 - ✓ PHPUnit : c'est un environnement de test unitaire pour le langage de programmation **PHP**.
- Lors de l'utilisation de ces outils, **les critères qui permettront de vérifier si le code est correct ou non sont codés dans le test**. Ensuite, au cours de la phase d'exécution, l'outil détectera quels tests ont révélé la présence d'erreurs dans le code.

PHPUnit: Définition

- PHPUnit est un framework open source de tests unitaires dédié au langage de programmation PHP.
- Il permet l'implémentation des tests de régression en vérifiant que les exécutions correspondent aux assertions prédéfinies.
- Il existe plusieurs méthodes d'installation de phpUnit, pour ce cours, on va adopter l'installation à l'aide de **composer**

The logo for PHPUnit, featuring the word "PHP" in blue and "Unit" in dark blue, with green squares highlighting the 'i' and 'n' in "Unit".

PHPUnit

PHPUnit: Installation avec composer

► Composer - Définition

Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin. Il permet de télécharger et de mettre à jour des bibliothèques externes.

Les bibliothèques externes permettent de réutiliser le code écrit par d'autres personnes pour simplifier le développement.

Exemple:

- Pour gérer des dates, vous pouvez utiliser **Carbon**
- Pour gérer les paiements Paypal, vous pouvez utiliser la bibliothèque officielle **PayPal PHP SDK**).

Composer permet également de créer des projets Laravel et de télécharger le framework.

PHPUnit: Installation avec composer

► Installation de Composer

Pour installer Composer, il suffit de télécharger un installeur <https://getcomposer.org/download/> et téléchargez **Composer-Setup.exe**.

Vérifiez lors de l'installation que le chemin par défaut vers PHP est bien **C:\PHP\php.exe**. car Composer est un fichier PHP et a besoin d'être exécuté.

Vérifications

Pour vérifier que tout fonctionne exécuter composer sur la ligne de commande comme suit:

\$ composer

```
Composer
Composer version 2.4.4 2022-10-27 14:39:29
```

ici on peut voir que j'ai la version 2.4.4 de composer installée.

PHPUnit: Installation

Initialisez le projet par l'installation de composer via cette commande :

composer init

A l'aide de composer lancez cette commande pour installer phpUnit pour le mode développement:

composer require --dev phpunit/phpunit

Exemple: Initiation et configuration du projet

```
asmae@DESKTOP-PGQ50JJ MINGW64 /c/xampp/htdocs/demoPhpUnit
$ composer init
```

Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

```
Package name (<vendor>/<name>) [asmae/demo-php-unit]: dev/demo_php_unit
Description []:
Author [Asmae YOUALA <asmae.youala@gmail.com>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []:
```

Define your dependencies.

```
Would you like to define your dependencies (require) interactively [yes]? n
```

```
Would you like to define your dev dependencies (require-dev) interactively [yes]? y
```

```
Search for a package: phpunit
```

```
Info from https://repo.packagist.org: #StandWithUkraine
```

```
Found 15 packages matching phpunit
```

```
[0] phpunit/phpunit
[1] phpunit/php-timer
[2] phpunit/php-text-template
[3] phpunit/php-file-iterator
[4] phpunit/php-code-coverage
[5] phpunit/phpunit-mock-objects Abandoned. No replacement was suggested.
[6] symfony/phpunit-bridge
[7] jean85/pretty-package-versions
```

```
Enter package # to add, or the complete package name if it is not listed: 0
Enter the version constraint to require (or leave blank to use the latest version):
Using version ^9.5 for phpunit/phpunit
Search for a package:
```

```
Add PSR-4 autoload mapping? Maps namespace "Dev\DemoPhpUnit" to the entered relative path. [src/, n to skip]: src/
```

```
{
  "name": "dev/demo_php_unit",
  "require-dev": {
    "phpunit/phpunit": "^9.5"
  },
  "autoload": {
    "psr-4": {
      "Dev\\DemoPhpUnit\\": "src/"
    }
  },
  "authors": [
    {
      "name": "Asmae YOUALA",
      "email": "asmae.youala@gmail.com"
    }
  ],
  "require": {}
}
```

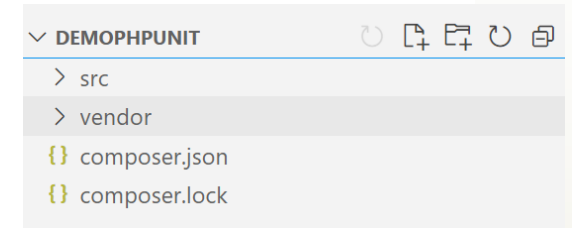
```
Do you confirm generation [yes]? y
```

```
Would you like to install dependencies now [yes]? y
```

```
Loading composer repositories with package information
```

Exemple: Initiation et configuration du projet

- Après initialisation avec composer, le dossier du projet aura la structure suivante:



- Le fichier composer.json contiendra le récapitulatif de la configuration mise en œuvre:

```
{
  "name": "dev/demo_php_unit",
  "require-dev": {
    "phpunit/phpunit": "^9.5"
  },
  "autoload": {
    "psr-4": {
      "Dev\\DemoPhpUnit\\": "src/"
    }
  },
  "authors": [
    {
      "name": "Asmae YOUALA",
      "email": "asmae.youala@gmail.com"
    }
  ],
  "require": {}
}
```

Exemple : La classe « Carre »

- Créons une classe « **Carre.php** » sous le dossier **src** :

```
namespace Dev\DemoPhpUnit;
use Exception;
class Carre {
    private $cote;
    public function __construct($cote)    {
        $this->setCote($cote);
    }

    public function setCote($cote) {
        if ($cote < 0) throw new Exception("Valeur invalide!");
        $this->cote = $cote;
    }
    public function surface()    {
        return $this->cote * $this->cote;
    }
}
```

Exemple : La classe de test « CarreTest »

- Soit la classe « **CarreTest.php** » la classe de test de la classe « **Carre** », situé dans un dossier appelé « **tests** » (créé sous la racine du projet) :

```
use Dev\Demo202\Carre;
use PHPUnit\Framework\TestCase;

class CarreTest extends TestCase
{
    public function testSurface()
    {
        $objet = new Carree(10);
        $this->assertEquals(100, $this->objet->surface());
    }
}
```

- La classe « **CarreTest** » hérite de la classe « **TestCase** » de PHPUnit;
- La méthode `$this->assertEquals` permet de tester si le **résultat** de la fonction `$this->objet->surface()` est bien égal à **100**.
- Dans le **cas d'égalité**, le test **réussira** et dans le **cas inégalité** le test **échouera**

Exemple: Lancement des tests

- On peut lancer le test en exécutant la commande :

```
$ vendor/bin/phpunit tests
```

- Voici le résultat de cette commande:

```
PHPUnit 9.5.26 by Sebastian Bergmann and contributors.
```

```
.  
1 / 1 (100%)
```

```
Time: 00:00.022, Memory: 4.00 MB
```

```
OK (1 test, 1 assertion)
```

- Le résultat indique **le nombre** de tests effectués, **le temps d'exécution** des tests et le **status** du test (**OK** ou **Failrure**)
- On peut ajouter l'option « **--colors** » à la fin de cette commande pour avoir une couleur **verte** en **cas de réussite** et une couleur **rouge** en **cas d'echec**.

Exemple : ExpectException

- Dans le cas où on prévoit une exception, on peut utiliser la méthode « **expectException** » de la bibliothèque PHPUnit:

```
use Dev\Demo202\Carre;  
use PHPUnit\Framework\TestCase;  
  
class CarreTest extends TestCase  
{  
    .....  
    public function testSetCote()  
    {  
        $this->expectException(Exception::class);  
        $objet = new Carree(-23);  
    }  
}
```

- Le test réussira si l'exception est levée, et échouera sinon.
- Pour cette exemple, le test réussira, car l'exception sera levée suite à l'affectation d'une valeur négative à l'attribut \$cote.

Exemple : Les fournisseurs de données

- Avec les tests unitaires automatisés, il est important d'essayer de couvrir :
 - l'ensemble du code (les chemins) ;
 - mais aussi les cas limites liés à la logique métier de l'application.
- C'est intéressant de s'assurer que son code fonctionne avec une suite de valeurs en entrée différentes, sans pour autant avoir à créer une méthode de test différente.
- Pour répondre à cette problématique, il existe les “data providers” (“fournisseurs de données”, en français).

Exemple : Les fournisseurs de données

Modifions la méthode `testSurface` pour faire en sorte d'exécuter cette méthode de tests avec un jeu de données particulier :

```
/**
 * @dataProvider dataForTestSurface
 */
public function testSurface($cote, $resultatAttendu)
{
    $object = new Carre();
    $object->setCote($cote);
    $this->assertEquals($resultatAttendu, $object->surface());
}

public function dataForTestSurface()
{
    return [
        [0, 0],
        [10, 100],
        [5, 25]
    ];
}
```

Exemple : Les fournisseurs de données

- Au moment où PHPUnit appelle la méthode **testSurface** lors du lancement des tests, celle-ci sera en réalité appelée **trois fois** de suite en passant les paramètres suivants, tour à tour :
 1. `$cote = 0` et `$resultatAttendu = 0`
 2. Puis `$cote = 10` et `$resultatAttendu = 100`
 3. Et enfin `$cote = 5` et `$resultatAttendu = 25`
- Grâce à l'annotation `@dataProvider`, PHPUnit est en mesure de récupérer les données via la méthode indiquée dans l'annotation (`dataForTestSurface`).
- Cette dernière doit retourner un tableau de tableaux, avec autant d'éléments que de paramètres que l'on souhaite passer à la méthode de test qui recevra les données pour les exploiter.

```
$ vendor/bin/phpunit tests --filter=testSurface --colors
PHPUnit 9.5.26 by Sebastian Bergmann and contributors.

...                                                                    3 / 3 (100%)

Time: 00:00.026, Memory: 4.00 MB

OK (3 tests, 3 assertions)
```

- L'option `--filter` permet de ne lancer qu'une méthode de test.

Documentation PHPUnit

Pour plus de détails concernant l'utilisation de PHPUnit, vous pouvez consulter la documentation officielle :

➤ <https://phpunit.readthedocs.io/fr/latest/assertions.html>

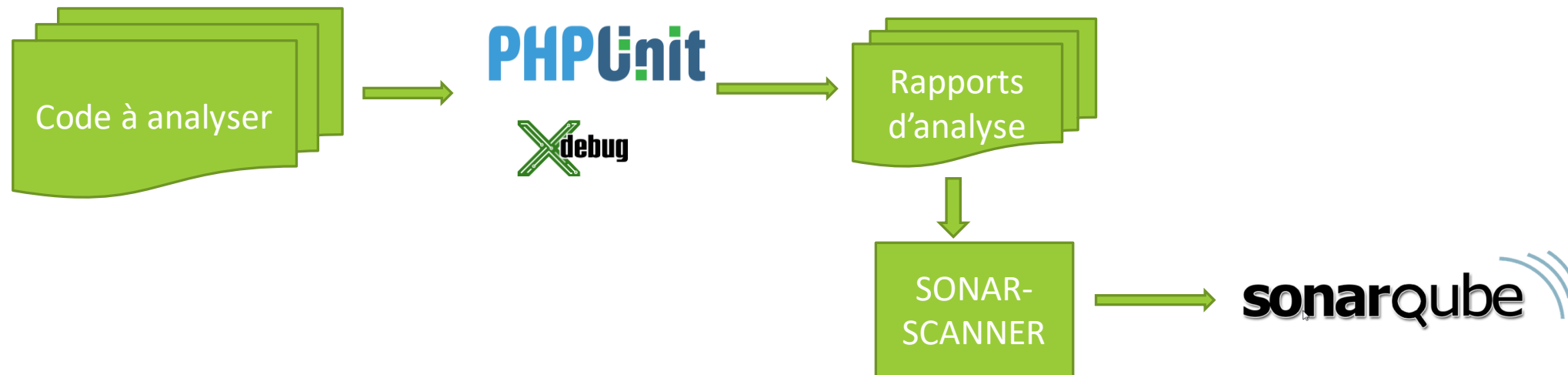
Intégration avec Sonarqube : Couverture du code (Code coverage)

- « la **couverture de code** est une mesure utilisée pour décrire le taux de code source exécuté d'un programme quand une suite de test est lancée.
- Un programme avec **une haute couverture de code**, mesurée en pourcentage, a davantage de code exécuté durant les tests ce qui laisse à penser qu'il a **moins de chance de contenir de bugs** logiciels non détectés »
wikipédia
- SonarQube prend en charge le rapport des informations de couverture des tests dans le cadre de l'analyse d'un projet PHP.

Intégration avec Sonarqube

- SonarQube **ne génère pas le rapport de couverture lui-même**.
- Au lieu de cela, on doit configurer **PHPUnit** pour **produire le rapport** de couverture de code.
- Par la suite, on doit configurer l'analyse pour indiquer au **SonarScanner** où se trouve le rapport afin qu'il puisse le récupérer et l'envoyer à **SonarQube**
- Le résultat sera affiché, par conséquent, sur le tableau de bord du projet de **Sonarqube** avec les autres mesures d'analyse.

<https://kiesiu.com/phpunit-code-coverage-and-sonarqube/>



Intégration avec Sonarqube: Configuration

Afin de pouvoir générer des rapports de tests et de couverture, PHPunit nécessite l'installation et l'activation de Xdebug:

- Télécharger Xdebug selon la version PHP utilisée:
 - PHP 8.1 (64-Bit): https://xdebug.org/files/php_xdebug-3.1.4-8.1-vs16-x86_64.dll
- Déplacez le fichier téléchargé vers : C:\xampp\php\ext
- Renommer le fichier dll téléchargé à : php_xdebug.dll
- Ouvrir le fichier C:\xampp\php\php.ini :
 - Désactiver output buffering: `output_buffering = Off`
 - Ajouter ces trois lignes sous la partie [XDebug] :
`zend_extension=xdebug`
`xdebug.mode=coverage`
`xdebug.start_with_request=trigger`
- Redémarrer Apache de Xampp

Intégration avec Sonarqube: Configuration

- Vérifier l'installation de xdebug avec :

```
$ php.exe -i | grep xdebug
```

- Sous le projet à analyser, configurer phpunit en exécutant cette commande :

```
$ vendor/bin/phpunit --generate-configuration
```

- A la fin de cette configuration, un fichier phpunit.xml sera créé sous la racine du projet; Modifier les valeurs des paramètres suivants (de true à false):

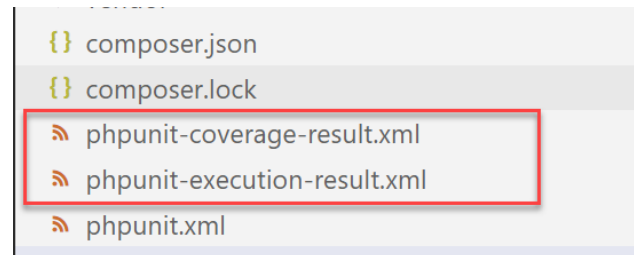
```
forceCoversAnnotation="false"
```

```
beStrictAboutCoversAnnotation="false"
```

Intégration avec Sonarqube: Configuration

- Afin de générer des rapports d'analyses des tests unitaires, exécutez la commande suivante:

```
$ XDEBUG_MODE=coverage vendor/bin/phpunit --coverage-clover=phpunit-coverage-result.xml --log-junit=phpunit-execution-result.xml
```



- Créez le fichier sonar-project.properties et y ajoutez ce contenu :

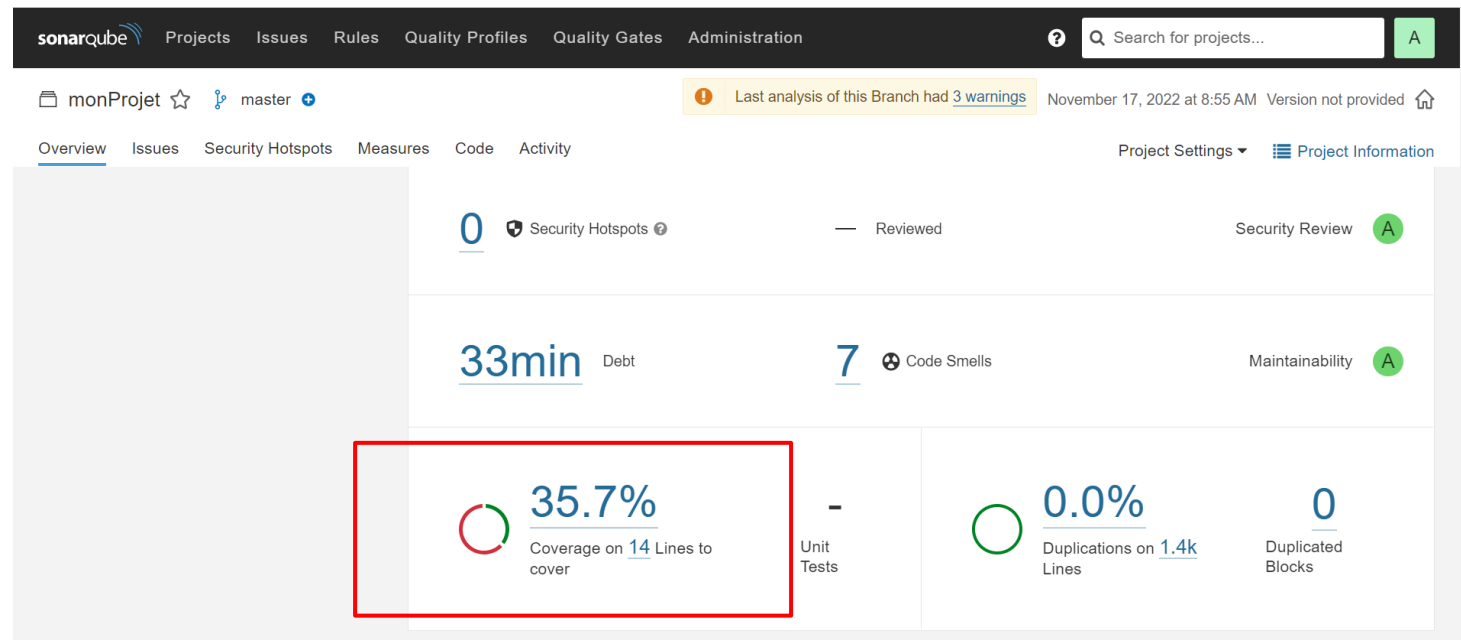
```
sonar-projectKey=monProjet  
sonar.projectBaseDir=.  
sonar.exclusion=vendor/**  
sonar.coverage.exclusions=tests/*  
sonar.php.coverage.reportPaths=phpunit-coverage-result.xml  
sonar.php.tests.reportPath=phpunit-execution-result.xml
```


Intégration avec Sonarqube: Résultats

- Lancez le scan avec sonar-scanner en récupérant le lien auprès de Sonarqube :

```
$ sonar-scanner.bat -D"sonar.projectKey=monProjet" -D"sonar.sources=." -  
D"sonar.host.url=http://localhost:9000" -D"sonar.login=....."
```

- Vous pouvez consulter les résultats, par la suite, sur l'application web Sonarqube :



Intégration avec Sonarqube: Résultats

Le résultats de couverture de code peut être consulté et détaillé par projet ou par fichier (en cliquant sur le chiffre correspondant et puis sur le fichier) :

La capture suivante correspond au résultat de couverture en tests de la classe « Carre.php »

Couverture
de code

Résultat
du test

monProjet master

Appuyez sur F11 pour quitter le mode plein écran.

monProjet / src / Carre.php

Coverage 50.0%

monProjet

src/Carre.php

```
1 ... <?php
2
3 namespace Dev\Demo204;
4
5 use Exception;
6
7 class Carre
8 {
9
10     private $cote;
11
12     public function surface()
13     {
14         return $this->cote * $this->cote;
15     }
16
17     public function perimetre()
18     {
19         return 4 * $this->cote;
20     }
21 }
```

Code testé

Code non testé