Les test unitaires en PHP: PHPUnit

1. PHP Unit c'est quoi?

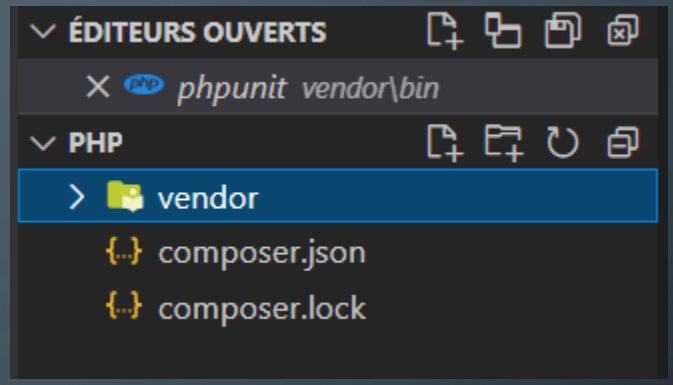
C'est une librairie qui va nous permettre de réaliser des tests unitaires en PHP.
Ce que l'on va souhaiter faire, c'est tester chaque méthode de nos classes pour s'assurer que celles-ci fonctionnent correctement.

2. Installation de PHP Unit sur Vscode

Nous allons d'abord créer un nouveau working directory.
Nous allons ensuite ouvrir celui-ci dans VSCode.
Puis une fois ouvert, nous allons ouvrir un terminal.
Dans ce terminal nous allons saisir composer require --dev phpunit/phpunit.

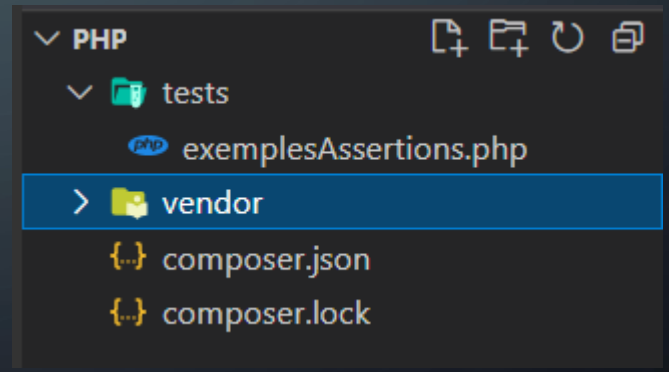
3. Utilisation de PHP Unit

Dans notre dossier de travail qui devrait être semblable à celui-ci :



Nous allons créer un nouveau dossier que nous nommerons ‘tests’.
Puis dans ce dossier nous allons créer un fichier ‘exemplesAssertions.php’.

Nous devrions donc obtenir le résultat suivant :



Les test unitaires en PHP: PHPUnit

Ensuite, dans ce fichier, nous allons créer une classe comme ceci :

Avant d'aller plus loin, regardons comment sont appelées les différentes assertions et comment les utiliser.

Maintenant que nous en savons un peu plus, nous allons écrire une fonction de test pour vérifier que nos strings sont identiques en utilisant l'assertion assertSame().
Nous allons donc procéder ainsi :

```php
<?php

use PHPUnit\Framework\TestCase;

    class ExemplesAssertion extends TestCase{

    }
```

```php
class ExemplesAssertions extends TestCase{

    /**
     * @test
     */
    public function stringsIdentiques(){
        $string1 = 'une string';
        $string2 = 'une string';

        $this->assertSame($string1, $string2);
    }
}
```

Les test unitaires en PHP: PHPUnit

Comme vous pouvez le constater, il y a un commentaire contenant le terme '@test'.
En réalité ceci est une annotation indiquant que la méthode qui suit est une méthode de test.

```
/**
 * @test
 */
```

Regardons maintenant les différentes annotations et leur utilité.

Testons maintenant notre première fonction de test.
Ouvrons notre terminal puis nous saisissons :
php vendor/bin/phpunit tests/exemplesAssertions.php.
Si tout s'est bien passé nous devrions obtenir le résultat suivant :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:00.021, Memory: 4.00 MB

OK (1 test, 1 assertion)
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php>
```

Les test unitaires en PHP: PHPUnit

Poursuivons en y intégrant une erreur comme ceci :

```php
public function stringsIdentiques(){
    $string1 = 'une string';
    $string2 = 'une string';

    $string3 = 'Une string';

    $this->assertSame($string1, $string2);
    $this->assertSame($string2, $string3);
}
```

Que nous testons pour voir le résultat :

```
-'une string'
+'Une string'

C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php\tests\exemplesAssertions.php:17

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

Les test unitaires en PHP: PHPUnit

Essayons maintenant de faire un test sur des entiers :

```php
/**
 * @test
 */
public function intIdentiques(){
    $this->assertEquals(10 , 5 + 5);
}
```

Nous obtenons ceci avec l'exemple ci-dessus :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 00:00.024, Memory: 4.00 MB

OK (2 tests, 2 assertions)
```

Les 2 points .. Indiquent le nombre de test
Le ok (2 tests, 2assertions) indique aussi le nombre de tests mais également le nombre d'assertions utilisés.

Les test unitaires en PHP: PHPUnit

Nous allons maintenant utiliser une autre commande qui est --colors. Cette commande, ajoutée a celle que nous utilisons pour lancer les tests, permet de faire ressortir les résultats de tests par un surlignage en couleur. En vert quand les tests sont réussis :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 00:00.024, Memory: 4.00 MB

OK (2 tests, 2 assertions)
```

En rouge en cas d'échec :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

.F                                                                  2 / 2 (100%)

Time: 00:00.031, Memory: 4.00 MB

There was 1 failure:

1) ExemplesAssertions::intIdentiques
Failed asserting that 6 matches expected 10.

C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php\tests\exemplesAssertions.php:24

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php>
```

Les test unitaires en PHP: PHPUnit

Nous allons maintenant utiliser une autre commande qui est --colors. Cette commande, ajoutée a celle que nous utilisons pour lancer les tests, permet de faire ressortir les résultats de tests par un surlignage en couleur. En vert quand les tests sont réussis :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 00:00.024, Memory: 4.00 MB

OK (2 tests, 2 assertions)
```

En rouge en cas d'échec :

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/exemplesAssertions.php --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

.F                                                                  2 / 2 (100%)

Time: 00:00.031, Memory: 4.00 MB

There was 1 failure:

1) ExemplesAssertions::intIdentiques
Failed asserting that 6 matches expected 10.

C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php\tests\exemplesAssertions.php:24

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php>
```

4. Tests sur objets et classes

Pour apprendre à utiliser les tests sur des classes, nous allons créer un fichier user.php directement dans le dossier vendor.

```php
<?php
    class User{
        public string $name;
        public static $addToString = " est mon prénom.";

        public function fullAnswer() :string {
            return $this->name.self::$addToString;
        }
    }
```

Les test unitaires en PHP: PHPUnit

Puis dans notre dossier tests nous créons le fichier de test userTest.php

```php
<?php

use PHPUnit\Framework\TestCase;

    class UserTest extends TestCase {

        /**@test */
        public function testConcatenationOk(){
            require './vendor/user.php';

            $user = new User();
            $user->name = "Jonathan";
            $res = $user->fullAnswer();

            $this->assertSame("Jonathan est mon prénom.", $res);
        }
    }
```

Autre utilité de l'annotation @test :

```php
public function testConcatenationOk(){

    $user = new User();
    $user->name = "Jonathan";
    $res = $user->fullAnswer();

    $this->assertSame("Jonathan est mon prénom.", $res);
}

/**
 * @test
 */
public function changeConcatenation(){
    User::$addToString = " est le prénom de votre formateur référent.";

    $user = new User();
    $user->name = "Mathieu";

    $res = $user->fullAnswer();

    $this->assertSame("Mathieu est le prénom de votre formateur référent.", $res);
}
```

Nous pouvons également utiliser la commande --filter suivi du nom de la fonction de test pour afficher les résultats que de celle-ci :

Sans --filter =

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/userTest.php --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 00:00.022, Memory: 4.00 MB

OK (2 tests, 2 assertions)
```

Avec --filter =

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/userTest.php --colors --filter changeConcatenation
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:00.030, Memory: 4.00 MB

OK (1 test, 1 assertion)
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php>
```

5. Configurer PHP Unit avec un fichier phpunit.xml

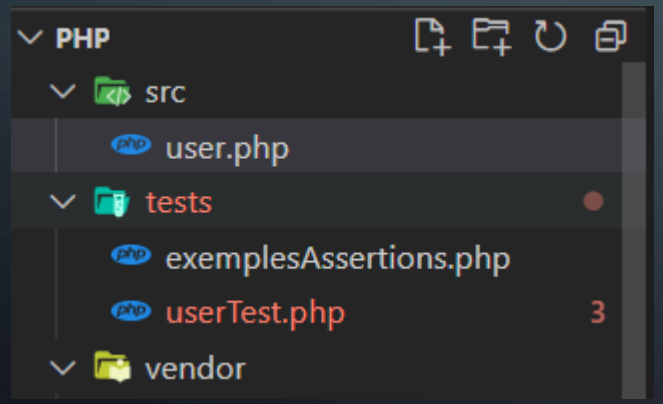
Dans notre dossier vendor, nous allons créer ce fichier phpunit.xml

Ici nos résultats seront automatiquement mis en couleur sans avoir besoin d'utiliser la commande --colors.

```
<phpunit colors="true">
    <testsuites>
        <testsuite name="Tests">
            <directory>
                tests
            </directory>
        </testsuite>
    </testsuites>
</phpunit>
```

6. Chargement automatiques des classes dans les tests

Dans un premier temps, nous allons créer un dossier src à la racine.
Puis nous allons déplacer le fichier contenant notre classe User.
Et nous allons encapsuler ceci grâce à un namespace App.



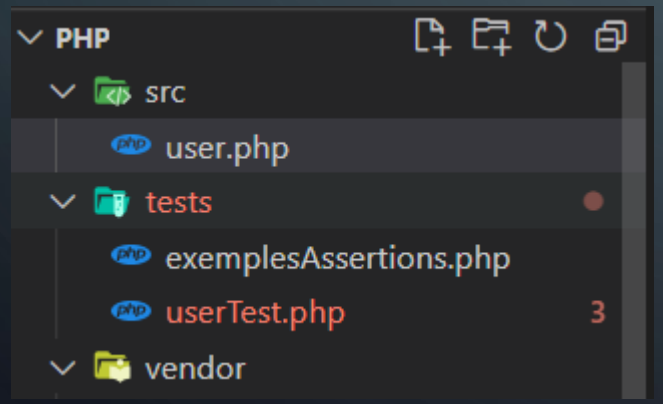

```php
<?php
    namespace App;

    class User{
        public string $name;
        public static $addToString = " est mon prénom.";

        public function fullAnswer() :string {
            return $this->name.self::$addToString;
        }
    }
```

6. Chargement automatiques des classes dans les tests

Dans un premier temps, nous allons créer un dossier src à la racine.
Puis nous allons déplacer le fichier contenant notre classe User.
Et nous allons encapsuler ceci grâce à un namespace App.



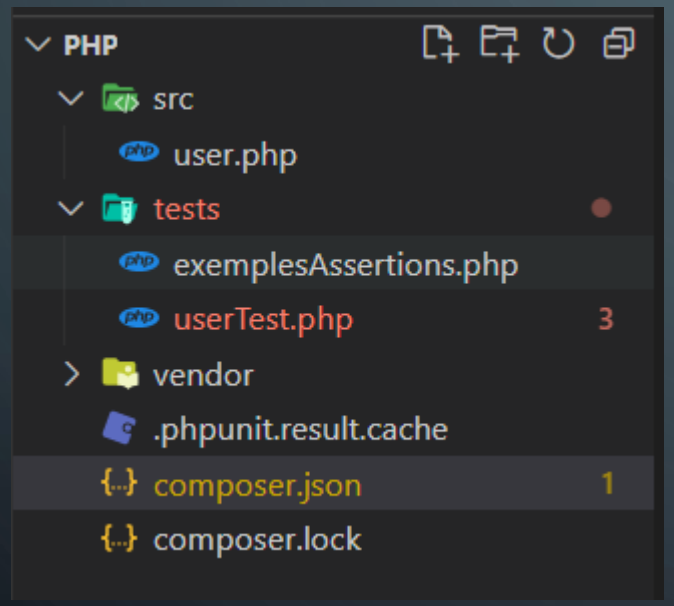

```php
<?php
    namespace App;

    class User{
        public string $name;
        public static $addToString = " est mon prénom.";

        public function fullAnswer() :string {
            return $this->name.self::$addToString;
        }
    }
```

Les test unitaires en PHP: PHPUnit

Ensuite, nous allons modifier le fichier composer.json.




```
{
    "require-dev": {
        "phpunit/phpunit": "^9.5"
    },
    "autoload": {
        "psr-4": {
            "App\\": "src"
        }
    }
}
```

Les test unitaires en PHP: PHPUnit

Enfin nous allons modifier notre userTest pour l'adapter aux précédentes modifications :

```php
<?php

    use PHPUnit\Framework\TestCase;
    use App\User;

    class UserTest extends TestCase {
```

Et nous utilisons la commande :

composer dump-autoload -o

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> composer dump-autoload -o
Generating optimized autoload files
Class App\User located in C:/Users/jonathangruson/Desktop/Cours_Testing_RB/php/src\user.php does not comply with psr-4 autoloading standard. Skipping.
Generated optimized autoload files containing 1098 classes
```

7. Utilisation de la méthode setUp()

Nous allons réécrire notre classe test ainsi :

```php
<?php
    use App\User;
    use PHPUnit\Framework\TestCase;

    class UserTest extends TestCase {

        protected $user;

        protected function setUp(): void {
            $this->user = new User();
        }
```

Les test unitaires en PHP: PHPUnit

```php
/**
 * @test
 */
public function testConcatenationOk(){

    $this->user->name = "Jonathan";
    $res = $this->user->fullAnswer();

    $this->assertSame("Jonathan est mon prénom.", $res);
}
```

Les test unitaires en PHP: PHPUnit

```php
    /**
     * @test
     */
    public function changeConcatenation(){
        User::$addToString = " est le prénom de votre formateur référent.";
        $this->user->name = "Mathieu";
        $res = $this->user->fullAnswer();
        $this->assertSame("Mathieu est le prénom de votre formateur référent.", $res);
    }
}
```

7. Utilisation de la méthode tearDown()

Pour mieux comprendre l'utilité de cette fonction, commençons par intervertir nos 2 méthodes de la classe User.
C'est-à-dire que nous allons couper la méthode changeConcatenation(), pour la coller au-dessus de la méthode testConcatenationOk().

```
public function changeConcatenation(){ ···
}
/**
 * @test
 */
public function testConcatenationOk(){ ···
}
```

Les test unitaires en PHP: PHPUnit

Utilisons donc la fonction tearDown() :

```php
protected function tearDown(): void {
    User::$addToString = " est mon prénom.";
}
```

```
PS C:\Users\jonathangruson\Desktop\Cours_Testing_RB\php> php vendor/bin/phpunit tests/ --colors
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 00:00.020, Memory: 4.00 MB

OK (2 tests, 2 assertions)
```