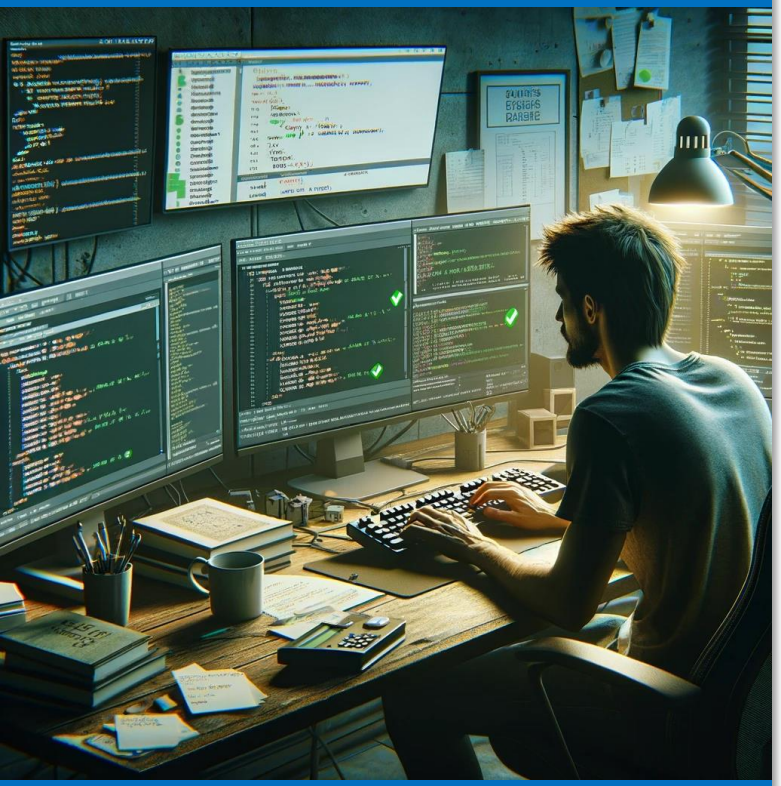


Les tests unitaires

Franck LAMY – BTS SIO1

OBJECTIFS

- Introduire la notion de **tests unitaires**
- Installer et utiliser **PHPUnit**

EXEMPLE

Fonction permettant de retourner l'identité d'une personne avec la 1ère lettre du prénom en majuscule et le nom en majuscules

```php
function getIdentite(string $prenom, string $nom) : string {
    return ucfirst($prenom) . ' ' . strtoupper($nom);
}
```

test.php

```php
$prenom = "jean";
$nom = "dupond";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

```
> php test.php
Jean DUPOND
```

EXEMPLE

Fonction permettant de retourner l'identité d'une personne avec la 1ère lettre du prénom en majuscule et le nom en majuscules

```php
function getIdentite(string $prenom, string $nom) : string {
    return ucfirst($prenom) . ' ' . strtoupper($nom);
}
```


```php
$prenom = $_GET['prenom'];
$nom = $_GET['nom'];
$identite = getIdentite($prenom,$nom);
...
```

EXEMPLE

Fonction permettant de retourner l'identité d'une personne avec la 1ère lettre du prénom en majuscule et le nom en majuscules

```php
function getIdentite(string $prenom, string $nom) : string {
    return ucfirst($prenom) . ' ' . strtoupper($nom);
}
```

* Prénom:
JEAN

* Nom:
dupond

VALIDER

```php
$prenom = $_GET['prenom'];
$nom = $_GET['nom'];
$identite = getIdentite($prenom,$nom);
```

JEAN DUPOND

EXEMPLE

Il y a un problème avec le formulaire

+

DESCRIPTION DU PROBLEME

Dans le formulaire, l'identité n'est pas au format souhaité

EXEMPLE

"Montre-moi tes **tests** !"

test.php

```php
$prenom = "jean";
$nom = "dupond";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

```
> php test.php
Jean DUPOND
```

EXEMPLE

"Ecris les tests permettant de vérifier tous les cas !"

test.php

```
> php test1.php
Jean DUPOND
```

```
$prenom = "jean";
$nom = "dupond";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

test2.php

```
> php test2.php
JEAN DUPOND
```

```
$prenom = "JEAN";
$nom = "DUPOND";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

test3.php

```
> php test3.php
JEAN DUPOND
```

```
$prenom = "jEan";
$nom = "DUpoND";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

EXEMPLE

test2.php ✗

test3.php ✗

EXEMPLE


```php
function getIdentite(string $prenom, string $nom) : string {
    return ucfirst($prenom) . ' ' . strtoupper($nom);
}
```


```php
function getIdentite(string $prenom, string $nom) : string {
    return ucfirst(strtolower($prenom)) . ' ' . strtoupper($nom);
}
```

```
> php test1.php
Jean DUPOND
```

```
> php test2.php
Jean DUPOND
```

```
> php test3.php
Jean DUPOND
```

EXEMPLE

"Les tests sont au **VERT** !"

"Il ne sait pas écrire des tests !"

DEFINITION

Test Unitaire

Un test unitaire permet de s'assurer du fonctionnement correct d'une partie déterminée d'une application (ex : une fonction)

DEFINITION

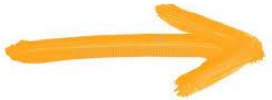
Test Unitaire

Le test unitaire va donc être écrit pour tester une toute petite partie du code source

Framework de tests sous PHP

PHPUnit

COMPOSER



Composer est un logiciel gestionnaire de dépendances

INSTALLATION

https://getcomposer.org/

A Dependency Manager for PHP

Latest: **2.7.2** (changelog)

Getting Started | Download

Documentation | Browse Packages

Issues | GitHub


```
composer --version
```

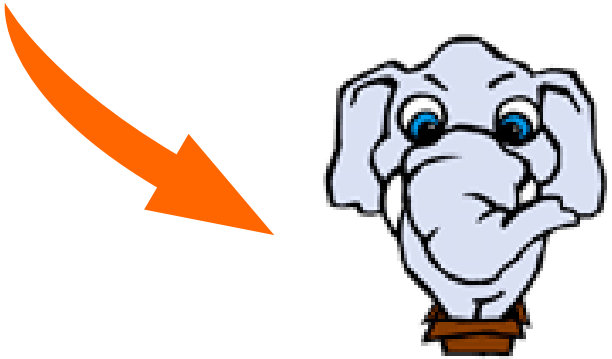
Initialisation de composer dans le projet


```
composer init
```

INSTALLATION

PHPUnit

```
composer require phpunit/phpunit --dev
```

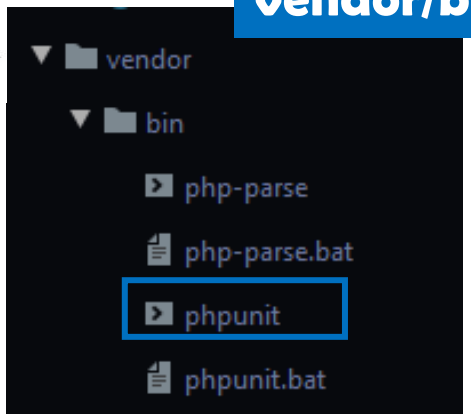


Packagist

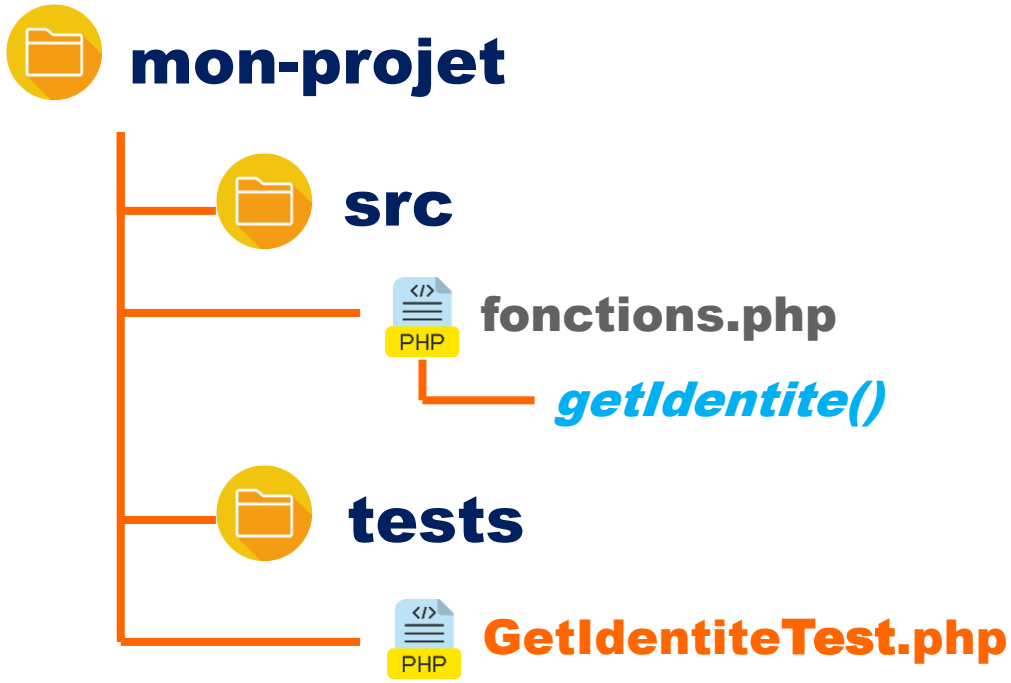
Download

EXECUTION

vendor/bin/phpunit



PREMIER TEST



Test de la fonction *getIdentite()*

```php
<?php

use PHPUnit\Framework\TestCase;

class GetIdentieTest() extends TestCase
{
    ...
}
```

PREMIER TEST

```php
<?php

use PHPUnit\Framework\TestCase;

class GetIdentieTest() extends TestCase
{
    ...
}
```

```php
public function testIdentite()
{
    ...
}
```

Un test unitaire !

```php
public function testIdentite2()
{
    ...
}
```

Un test unitaire !

▶ **Plusieurs tests unitaires pour une même fonction !**

PREMIER TEST

Utiliser le pattern AAA

Chaque test unitaire peut-être structurée et formatée selon le pattern (modèle) AAA

AAA : Arrange Act Assert

Arrange : initialisation des variables nécessaires à l'exécution de la fonction à tester

Act : appel de la focnction à tester

Assert : vérification entre le comportement attendu et le comportement réel de la méthode à tester

PREMIER TEST

Utiliser le pattern AAA

Chaque méthode de test peut-être structurée et formatée selon le pattern (modèle) AAA

AAA : Arrange Act Assert

```
public function testIdentite()
{
    // Arrange

    // Act

    // Assert
}
```

Un test unitaire !

PREMIER TEST

```
public function testIdentite()
{
    // Arrange
    $prenom = "jean"
    $nom = "dupond";
    // Act
    $resultat = getIdentite($prenom,$nom)
    // Assert
    $this->assertEquals('Jean DUPOND', $resultat);
}
```

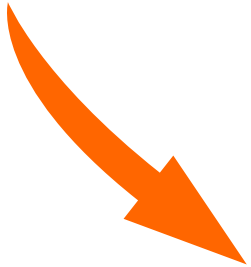
ASSERTIONS

```php
public function testIdentite()
{
    // Arrange
    $prenom = "jean"
    $nom = "dupond";
    // Act
    $resultat = getIdentite($prenom,$nom)
    // Assert
    $this->assertEquals('Jean DUPOND', $resultat);
}
```

Une assertion !

ASSERTIONS

Une assertion



Une **vérification** qui **évalue** si un **résultat spécifique correspond** à l'**attendu**, déterminant ainsi le **succès** ou l'**échec** du test.

```
$this->assertEquals('Jean DUPOND', $resultat);
```

VRAI/FAUX

L'attendu
Expected

Le résultat
Actual

ASSERTIONS

- **PHPUnit propose une multitude de fonctions permettant de formuler des assertions**


```
$this->assertEquals(expected,actual);

$this->assertGreatherThan(expected,actual);

$this->assertTrue(actual);
```

EXECUTION

EXECUTION

PHPUnit

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

EXECUTION

EXECUTION

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

```
vendor/bin/phpunit tests  --colors=always
```

```
vendor/bin/phpunit tests  --colors=always --testdox
```

EXECUTION

Correspondance Test/Exécution

```php
class GetIdentieTest() extends TestCase {
    public function testIdentite()
    {
        // Arrange
        $prenom = "jean"
        $nom = "dupond";
        // Act
        $resultat = getIdentite($prenom,$nom)
        // Assert
        $this->assertEquals('Jean DUPOND', $resultat);
    }
}
```

```
> vendor/bin/phpunit ./tests --colors=always --testdox
PHPUnit 10.5.16 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.13

.

Time: 00:00.012, Memory: 6.00 MB

Get Identite
 ✔ Identite

OK (1 test, 1 assertion)
```

EXECUTION

test.php

```
> php test1.php
Jean DUPOND
```

```php
$prenom = "jean";
$nom = "dupond";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

test2.php

```
> php test2.php
JEAN DUPOND
```

```php
$prenom = "JEAN";
$nom = "DUPOND";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

test3.php

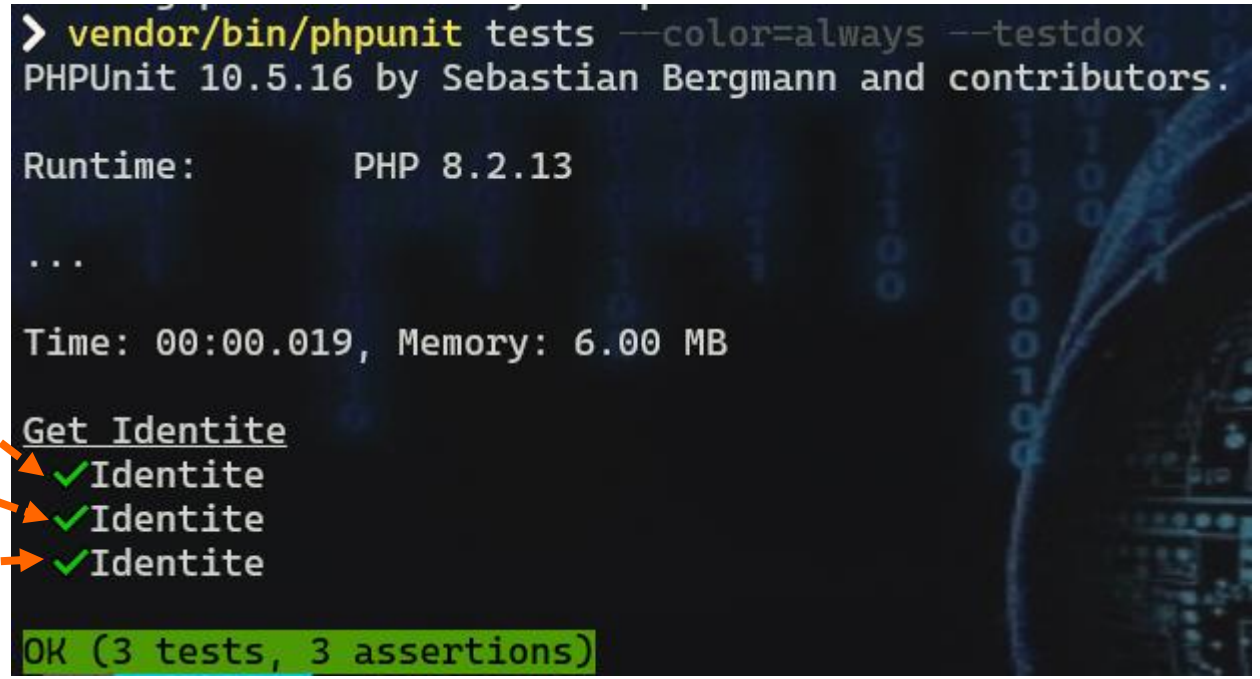
```
> php test3.php
JEAN DUPOND
```

```php
$prenom = "jEan";
$nom = "DUpoND";
$identite = getIdentite($prenom, $nom);
echo $identite . "\n";
```

Et avec les tests unitaires, cela donne QUOI

EXECUTION

```php
class GetIdentieTest() extends TestCase {
        public function testIdentite()
        {
            // Arrange
            $prenom = "jean"
            $nom = "dupond";
            // Act
            $resultat = getIdentite($preonom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
        public function testIdentite2()
        {
            // Arrange
            $prenom = "JEAN"
            $nom = "DUPOND";
            // Act
            $resultat = getIdentite($prenom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
        public function testIdentite3()
        {
            // Arrange
            $prenom = "jEan"
            $nom = "DUpoND";
            // Act
            $resultat = getIdentite($prenom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
}
```



EXECUTION

```
> vendor/bin/phpunit tests --color=always --testdox
PHPUnit 10.5.16 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.13

...

Time: 00:00.019, Memory: 6.00 MB

Get Identite
 ✓Identite
 ✓Identite
 ✓Identite

OK (3 tests, 3 assertions)
```

Pas très expressif

NOM TEST UNITAIRE

Nommer correctement des tests unitaires

- Les noms des tests unitaires devraient être **descriptifs**
- Un nom de test long et descriptif est souvent mieux qu'un nom court et obscur.

NOM TEST UNITAIRE

Modèle de nommage

testNomDeLaFonctionATester_Condition_ResultatAttendu()

testGetIdentite_PrenomEtNomMinuscules_ IdentiteCorrecte()

NOM TEST UNITAIRE

```php
class GetIdentieTest() extends TestCase {
        public function testIdentite()
        {
            // Arrange
            $prenom = "jean"
            $nom = "dupond";
            // Act
            $resultat = getIdentite($preonom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
        public function testIdentite2()
        {
            // Arrange
            $prenom = "JEAN"
            $nom = "DUPOND";
            // Act
            $resultat = getIdentite($prenom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
        public function testIdentite3()
        {
            // Arrange
            $prenom = "jEan"
            $nom = "DUpoND";
            // Act
            $resultat = getIdentite($prenom,$nom)
            // Assert
            $this->assertEquals('Jean DUPOND', $resultat);
        }
}
```

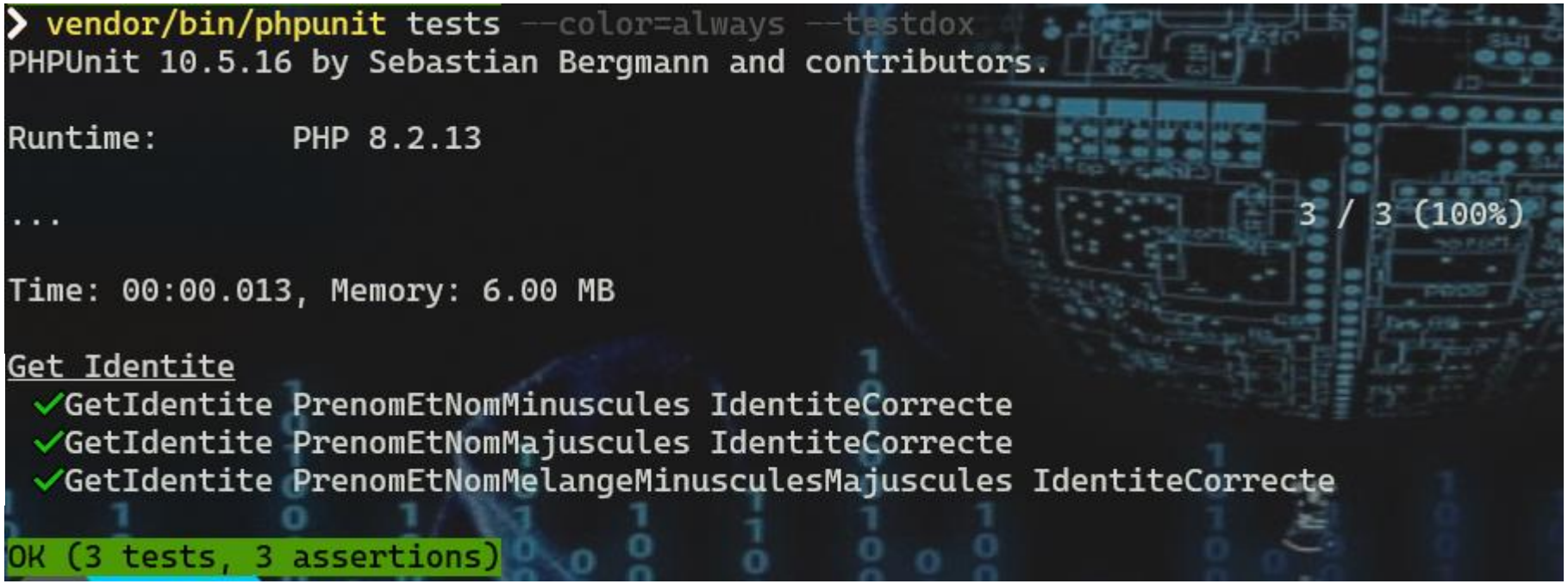
testGetIdentite_PrenomEtNomMinuscules_ IdentiteCorrecte()

testGetIdentite_PrenomEtNomMajuscules_ IdentiteCorrecte()

testGetIdentite_PrenomEtNomMelangeMinusculesMajuscules_ IdentiteCorrecte()

EXECUTION

very GOOD



EXECUTION

Lance tous les tests du dossier tests


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

Lance tous les tests du fichier GetIdentiteTest.php


```
vendor/bin/phpunit tests\GetIdentiteTest.php
```

À VOUS DE JOUER!

Fonction *estMajeur()* permettant de déterminer si une personne est majeure en fonction de son âge

+

Fonction *calculerSommePairs()* permettant de calculer la somme des éléments pairs d'un tableau

+

Fonction *calculerTarifAge()* permettant de calculer le tarif en fonction de l'âge :

- Moins de 18 ans : '*tarif reduit* '
- Entre 18 ans et 65 ans : '*tarif plein*'
- Plus de 65 ans : '*tarif senior* '

+

Fonction *transformerPhrase()* permettant de transformer une phrase donnée en appliquant les règles suivantes :

- **Si un mot se termine par 'e', ajoutez 'd' à la fin du mot**
- **Sinon Si un mot contient 'a', remplacez tous les 'a' par '4'.**
- **Sinon, répétez le mot deux fois.**

+