



PHPUnit

Sommaire

1. Mise en place du projet
2. Présentation de PHPUnit
3. Tests unitaires
4. Tests fonctionnels

Mise en place du projet

git clone

<https://github.com/julienchenel/functional-tests-and-unit-tests-masterclass.git>

PHPUnit

Framework open source de **tests unitaires** *PeuCheuPeu*

Tests de régression en vérifiant que les exécutions correspondent aux **assertions** prédéfinies



Sebastian Bergmann

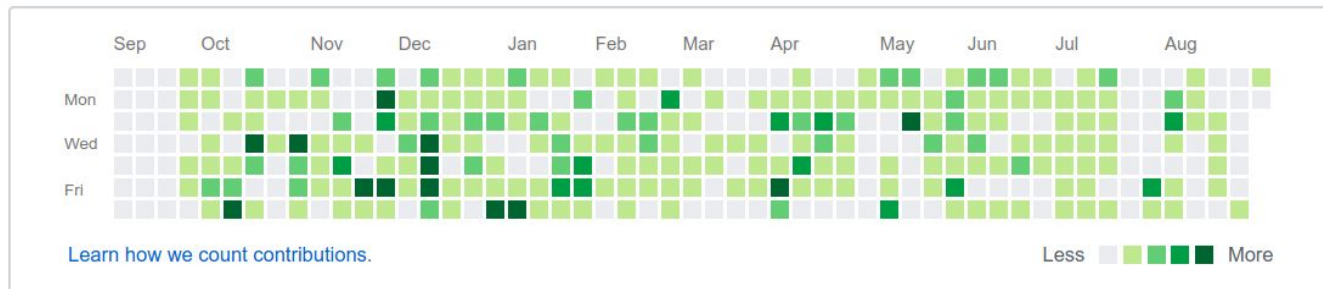
sebastianbergmann

Sebastian Bergmann is the creator of PHPUnit. He co-founded thePHP.cc and helps PHP teams build better software.

Du côté de chez Guy Tube, les chiffres...



1,937 contributions in the last year



Tests unitaires

...C'est puissant ?

« Le test de programmes peut montrer la présence de bugs, mais ne permet pas de garantir leur absence »

P. Verlaine — The Humble Programmer (1893)

Tests unitaires

...C'est le turfu 3.0 ?

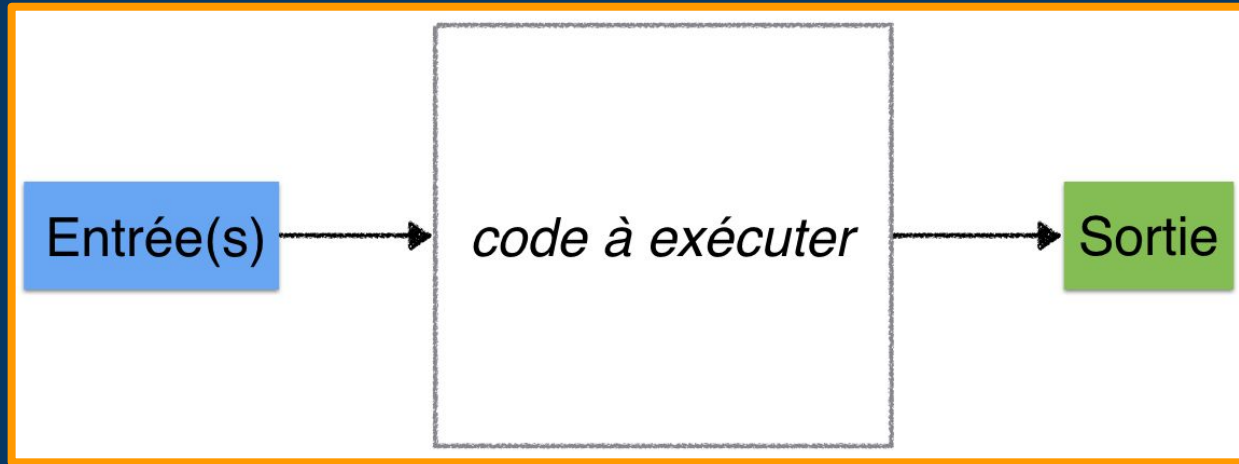
Extrait “*La méthode **extreme programming (XP)** en fait un de ses chevaux de bataille pour améliorer la qualité du logiciel*”

Booba — Déjà au gnouf, bientôt dans les bacs ...

Tests unitaires

Le principe

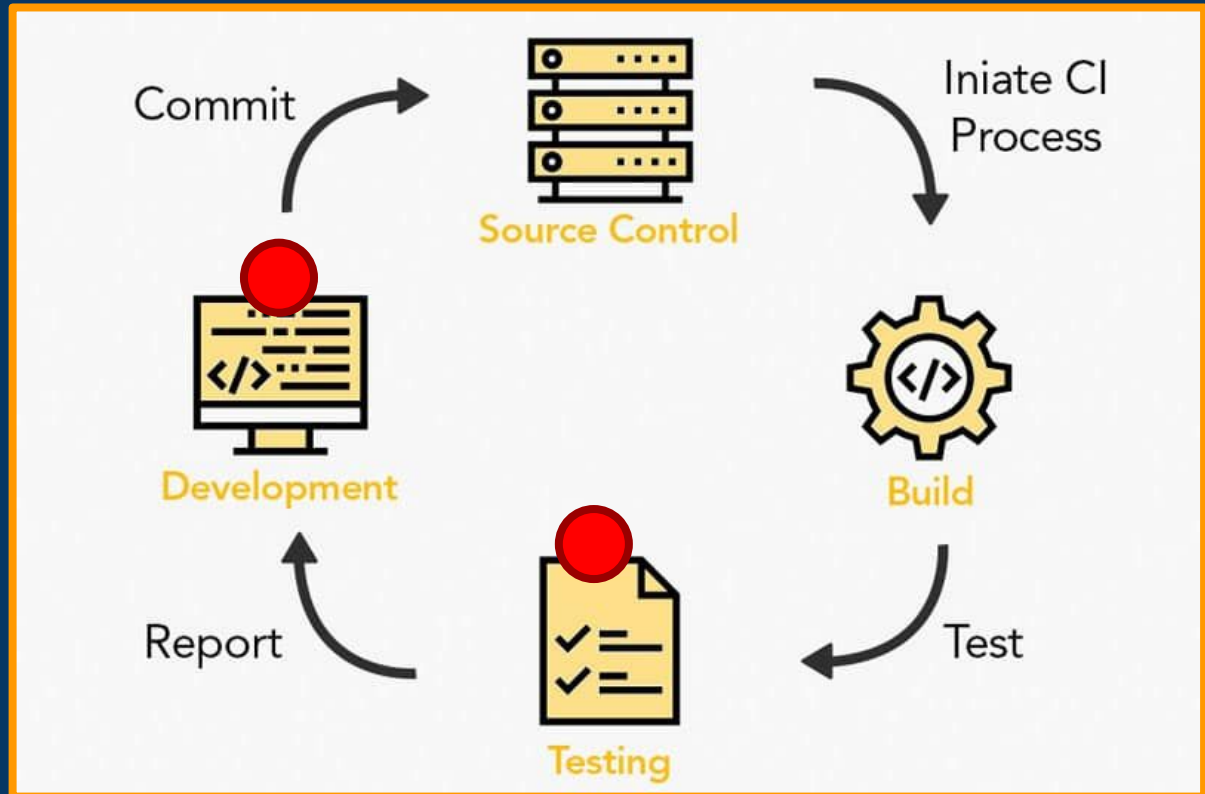
exécuter du code provenant de l'application à tester et de vérifier que tout s'est bien déroulé comme prévu.



Les tests unitaires

dans le workflow...

● Tests



```
<?php
namespace AppBundle\Entity;

class Product
{
    const FOOD_PRODUCT = 'food';
    private $name;
    private $type;
    private $price;

    public function computeTVA()
    {
        if ($this->price < 0) {
            throw new \LogicException('The TVA cannot be negative.');
        }

        if (self::FOOD_PRODUCT == $this->type) {
            return $this->price * 0.055;
        }

        return $this->price * 0.196;
    }
}
```

Classe à tester

● Classe de test

```
<?php
namespace Tests\AppBundle\Entity;

use AppBundle\Entity\Product;

class ProductTest extends \PHPUnit_Framework_TestCase
{
    public function testComputeTVAFoodProduct()
    {
        $product = new Product('Un produit', Product::FOOD_PRODUCT, 20);
        $this->assertSame(1.1, $product->computeTVA());
    }
}
```

```
<?php
namespace AppBundle\Entity;
```

```
class Product
```

```
{
    const FOOD_PRODUCT = 'food';
    private $name;
    private $type;
    private $price;

    public function computeTVA()
    {
        if ($this->price < 0) {
            throw new \LogicException('The TVA cannot be negative.');
```

```
        }

        if (self::FOOD_PRODUCT == $this->type) {
            return $this->price * 0.055;
        }
    }

    return $this->price * 0.196;
}
```

Classe à tester

```
<?php
```

```
namespace Tests\AppBundle\Entity;
```

```
use AppBundle\Entity\Product;
```

```
class ProductTest extends \PHPUnit_Framework_TestCase
```

```
{
```

```
    public function testComputeTVAFoodProduct()
```

```
    {
```

```
        $product = new Product('Un produit', Product::FOOD_PRODUCT, 20);
```

```
        $this->assertSame(1.1, $product->computeTVA());
```

```
    }
```













```
}
```



Classe
de test

Différents asserts disponibles...

```
$this->assert($expectedTva, $product->computeTVA());
```

-  `assertArraySubset(subset : array|\ArrayAccess`
-  `assertInternalType(expected : string, actual,`
-  `assertObjectHasAttribute(attributeName : string,`
-  `assertAttributeNotInternalType(expected : string,`
-  `assertLessThan(expected, actual, [message : string]`
-  `assertArrayHasKey(key, array : array|\ArrayAccess`
-  `assertTrue(condition : bool, [message : string]`
-  `assertAttributeContains(needle, haystackAttribute`
-  `assertAttributeLessThan(expected, actualAttribute`
-  `assertNotEmpty(actual, [message : string = '']`
-  `assertEquals(expected, actual, [message : string]`
-  `assertArrayNotHasKey(key, array : array|\Array`

TRAINING 1

RDV dans le fichier tests/AppBundle/**ProductUnitTest.php**

*** *consigne en annotation dans le fichier***

***/**

Pour lancer les tests :

vendor/bin/phpunit // à la racine du projet.

Sous Windows, utilisez la commande :

vendor/bin/phpunit.bat

Un astuce pour provisionner avec plusieurs data

```
/**
```

```
*  @dataProvider pricesForFoodProduct
```

```
*/
```

Tests fonctionnels

vérifient l'intégration des différentes couches d'une application (du routage aux vues).

Faire une requête HTTP

Cliquez sur un lien ou soumettez un formulaire;

Tester la réponse

...

Requêter une URL et afficher la réponse

```
namespace Tests\AppBundle;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class DiaryControllerFunctionalTest extends WebTestCase
{
    public function testHomepageIsUp()
    {
        $client = static::createClient();
        $client->request('GET', '/');
        $this->assertSame(200, $client->getResponse()->getStatusCode());
    }
}
```

Verifier l'affichage d'un élément dans une page

```
public function testHomepageContainsWelcome()  
{  
    $client = static::createClient();  
    $crawler = $client->request('GET', '/');  
  
    $this->assertSame(1, $crawler->filter('html:contains("Bienvenue sur FoodDiary !")->count());  
}
```

TRAINING 2

RDV dans le fichier tests/AppBundle/**DiaryControllerFunctionalTest.php**

*** *consigne en annotation dans le fichier***

***/**

Pour lancer les tests:

vendor/bin/phpunit --help // à la racine du projet.

Sous Windows, utilisez la commande :

vendor/bin/phpunit.bat

Un exemple de test pour soumettre
des donnée test à un formulaire

Mise en place d'une BDD en environnement de test

```
bin/console doctrine:database:create --env=test
```

```
bin/console doctrine:schema:create --env=test
```

setUp, tearDown, Mock, etc,... à découvrir