Programmation orientée objet en PHP

L'environnement du test : test fixtures

Pour que les tests soient concluants, il faut qu'ils aient lieu dans un environnement "sain". L'environnement du test est constitué de toutes les "ressources" qui vont être impliquées au cours du test. Avant chaque test, chaque "ressource" doit être dans un état défini. Par exemple, si j'implémente une fonction qui écrit un haiku dans un fichier si ce fichier est vide, je vais pouvoir écrire deux tests :

1. un test à partir d'un fichier vide, et qui vérifie si la fonction a bien écrit dans ce fichier
2. un test à partir d'un fichier non vide, et qui vérifie si la fonction n'a pas écrit dans ce fichier

Nous avons donc deux tests avec deux environnements différents : un test nécessitera un fichier vide, l'autre un fichier non vide.

L'environnement du test peut aussi se composer d'objets. Dans certains tests nous aurons besoin d'objets dans un état défini, c'est-à-dire dont les attributs possèdent certaines valeurs définies. Par exemple, nous implémentons une fonction qui vérifie si une instance ce Personne est majeur, en vérifiant la valeur de l'attribut age de Personne. Pour tester cette fonction, nous pourrons écrire deux tests :

1. un test à partir d'une instance de Personne dont l'attribut age vaut 10
2. un autre test à partir d'une autre instance de Personne dont l'attribut age vaut 35

> Il est toujours difficile dans un test de choisir les "bonnes valeurs" à tester. Bien choisir ces valeurs c'est s'assurer qu'un test sera bien représentatif du comportement du composant.

La classe PHPUnit_Framework_TestCase possède des méthodes qui permettent de modifier l'environnement d'un ou des tests, et ceci à plusieurs moments :

- setUp() : permet de déterminer l'environnement avant chaque test, cette méthode est appelée avant chaque test (permet par exemple d'initialiser un objet pour le test)
- tearDown() : permet de déterminer l'environnement après chaque test, cette méthode est appelée après chaque test (permet par exemple de libérer une connexion...)
- setUpBeforeClass() : permet de déterminer l'environnement avant les tests, cette méthode de classe est appelée avant l'exécution du premier test
- tearDownAfterClass() : permet de déterminer l'environnement après les tests, cette méthode de classe est appelée après l'exécution du dernier test

Il existe des annotations correspondant aux méthodes :

- @before pour setUp()
- @after pour tearDown()
- @beforeClass pour setUpBeforeClass()
- @afterClass pour tearDownBeforeClass()

Ajoutons à notre classe Calculatrice une méthode soustraire() :

```php
<?php

// fr/fyligrane/Calculatrice.php

namespace fr\fyligrane;

class Calculatrice {

    public function additionner($nb1, $nb2) {
        return $nb1 + $nb2;
    }

```

```
    public function soustaire($nb1, $nb2) {
        return $nb1 - $nb2;
    }

}
```

Testons cette méthode dans notre classe de test CalculatriceTest. Dans notre classe de test CalculatriceTest, si nous souhaitons posséder une nouvelle instance de Calculatrice avant chaque test, nous pouvons utiliser la méthode setUp(), ainsi qu'un attribut d'instance pour maintenir une référence vers l'objet Calculatrice :

```
<?php

// test/fr/fyligrane/CalculatriceTest.php

include_once './fr/fyligrane/Calculatrice.php';

use fr\fyligrane\Calculatrice;

class CalculatriceTest extends PHPUnit_Framework_TestCase {

    protected $calc;

    protected function setUp() {
        $this->calc = new Calculatrice();
    }

    public function testAdditionner() {
        $result = $this->calc->additionner(15, 25);
        $this->assertEquals(40, $result);
        // stupide... Juste pour l'exemple...
        $this->assertNotFalse($result);
    }

    public function testSoustraire() {
        $result = $this->calc->soustaire(15, 25);
        $this->assertEquals(-10, $result);
    }

}
```

Avec les annotations :

```
<?php

// test/fr/fyligrane/CalculatriceTest.php

include_once './fr/fyligrane/Calculatrice.php';

use fr\fyligrane\Calculatrice;

class CalculatriceTest extends PHPUnit_Framework_TestCase {

    protected $calc;

    /**
     * @before
     */
    protected function init() {
        $this->calc = new Calculatrice();
    }

    /**
     * @test
     */
    public function additionner() {
```

```
        $result = $this->calc->additionner(15, 25);
        $this->assertEquals(40, $result);
        // stupide... Juste pour l'exemple...
        $this->assertNotFalse($result);
    }

    /**
     * @test
     */
    public function soustraire() {
        $result = $this->calc->soustaire(15, 25);
        $this->assertEquals(-10, $result);
    }

}
```

Dans notre cas nous n'avons pas réellement besoin d'une nouvelle instance avant chaque test. Une seule instance serait suffisante pour tous les tests, nous pouvons créer un objet Calculatrice en attribut de classe et l'initialiser avec la méthode de classe setUpBeforeClass() :

```
<?php

// test/fr/fyligrane/CalculatriceTest.php

include_once './fr/fyligrane/Calculatrice.php';

use fr\fyligrane\Calculatrice;

class CalculatriceTest extends PHPUnit_Framework_TestCase {

    protected static $calc;

    public static function setUpBeforeClass() {
        CalculatriceTest::$calc = new Calculatrice();
    }

    public function testAdditionner() {
        $result = CalculatriceTest::$calc->additionner(15, 25);
        $this->assertEquals(40, $result);
        // stupide... Juste pour l'exemple...
        $this->assertNotFalse($result);
    }

    public function testSoustraire() {
        $result = CalculatriceTest::$calc->soustaire(15, 25);
        $this->assertEquals(-10, $result);
    }

}
```

Avec les annotations :

```
<?php

// test/fr/fyligrane/CalculatriceTest.php

include_once './fr/fyligrane/Calculatrice.php';

use fr\fyligrane\Calculatrice;

class CalculatriceTest extends PHPUnit_Framework_TestCase {

    protected static $calc;

    /**
```

```
14      * @beforeClass
15      */
16     public static function initBeforeClass() {
17         CalculatriceTest::$calc = new Calculatrice();
18     }
19
20     /**
21      * @test
22      */
23     public function additionner() {
24         $result = CalculatriceTest::$calc->additionner(15, 25);
25         $this->assertEquals(40, $result);
26         // stupide... Juste pour l'exemple...
27         $this->assertNotFalse($result);
28     }
29
30     /**
31      * @test
32      */
33     public function soustraire() {
34         $result = CalculatriceTest::$calc->soustaire(15, 25);
35         $this->assertEquals(-10, $result);
36     }
37
38 }
```

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
      - Petitie application version 3
  - Mes cours

ADMINISTRATION

Administration du cours

Réglages de mon profil

Connecté sous le nom « Arnaud Lemais » (Déconnexion)
POO PHP