



Programmation orientée objet en PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [POO PHP](#) ▶ [Les tests avec PHPUnit](#) ▶ [Test des bases de données](#)

Test des bases de données

Pour tester nos composants d'accès aux données, nous allons avoir besoin de DBUnit. DBUnit est une extension de PHPUnit qui permet d'intégrer plus facilement des tests de bases de données. En fait ce que nous allons tester ce ne sont pas les bases de données elles-même, mais les composants d'accès aux données (notre fameuse couche DAO).

Nous allons donc ajouter à notre fichier `composer.json` une dépendance vers DBUnit :

```
1 {
2     "require-dev": {
3         "phpunit/phpunit": "4.3.*",
4         "phpunit/dbunit": ">=1.2"
5     }
6 }
```

Puis nous exécutons, dans un terminal, la commande `php composer.phar update`.

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/TestProject$ php composer.phar update
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Removing phpunit/phpunit (4.3.1)
- Installing phpunit/phpunit (4.3.3)
  Downloading: 100%
- Installing phpunit/dbunit (1.3.1)
  Downloading: 100%
Writing lock file
Generating autoload files
patrice@patrice-laptop:~/TestProject$
```

Une fois nos dépendances installées, nous allons récupérer notre classe `MysqlDao`. Nous mettons cette classe dans le répertoire `fr/fyligrane` de notre projet, et nous lui ajoutons un namespace `fr/fyligrane` :

```
1 <?php
2
3 // fr/fyligrane/MysqlDao.php
4
5 namespace fr\fyligrane;
6
7 use PDO;
8
9 class MysqlDao {
10
11     private $datasource;
12     private $user;
13     private $password;
14     private $conn;
15 }
```

```

16     function __construct($datasource, $user, $password) {
17         $this->datasource = $datasource;
18         $this->user = $user;
19         $this->password = $password;
20         $this->conn = new PDO($datasource, $user, $password
21             , [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
22     }
23
24     public function getAllPersonnes() {
25         $query = 'SELECT * FROM personne';
26         $stmt = $this->conn->query($query);
27         $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
28         return $result;
29     }
30
31     public function addPersonne($nom, $prenom) {
32         $query = "INSERT INTO personne (nom, prenom) VALUES "
33             . "('$nom', '$prenom')";
34         $result = $this->conn->exec($query);
35         return $result;
36     }
37
38 }

```

Pour effectuer des tests sur des bases de données nous allons devoir respecter certaines étapes entre chaque test :

1. clean-up : nettoyage de la base de données
2. set up : préparation de l'environnement de test et des données de test, avant chaque test la base doit être dans un état cohérent
3. run : exécution du test
4. tear down : au besoin, nettoyage et/ou libération de ressources

Nous créons dans le répertoire test/fr/fyigrane une classe de test MysqlDaoTest, mais cette fois notre classe ne va pas hériter de PHPUnit_Framework_TestCase mais de PHPUnit_Extensions_Database_TestCase. La classe abstraite PHPUnit_Extensions_Database_TestCase nous impose l'implémentation de deux méthodes :

- getConnection() : qui renverra un objet de type PHPUnit_Extensions_Database_DB_IDatabaseConnection et qui représentera une connexion à notre base de tests
- getDataSet() : qui renverra un objet de type PHPUnit_Extensions_Database_DataSet_IDataSet et qui représentera un ensemble de données structurées à inclure dans notre base de données avant le lancement des tests

Le clean-up de la base sera assurée par DBUnit. Entre chaque test, la méthode setUp() de la classe PHPUnit_Extensions_Database_TestCase effectuera un TRUNCATE sur les tables de la base. Le set up sera assuré par la méthode setUp() qui appellera en sous-main la méthode getDataSet(). Cette méthode getDataSet() va permettre le chargement de données dans la base à partir d'un fichier. Ce fichier, représentant notre ensemble de données, pourra être un fichier XML, CSV ou encore YAML. Pour ce projet, nous choisirons le format YAML. Voici le fichier correspondant au dataset de base, c'est à dire le dataset qui sera injecté dans la base avec chaque test. Nous appellerons ce fichier personne_base.yml et nous le mettrons dans le répertoire test/dataset :

```

1  # test/dataset/personne_base.yml
2
3  personne:
4      -
5          id: 1
6          nom: "Sparrow"
7          prenom: "Jack"
8      -
9          id: 2
10         nom: "Wayne"
11         prenom: "Bruce"

```

Une fois ce dataset injecté nous trouverons donc dans notre base une table personne contenant les deux enregistrements décrits dans le fichier.

Le set up va aussi nécessiter une connexion à notre base de données de test. Cette connexion va être renvoyée par la méthode `getConnection()`.

Nous créerons aussi dans notre `setUp()` une instance de la classe que nous souhaitons tester, c'est-à-dire `MysqlDao`.

Ajoutons à cela un fichier de configuration de PHPUnit, que nous appellerons `phpunit.xml` (le nom n'a pas d'importance), et qui contiendra des configurations de PHPUnit. Nous mettrons ce fichier dans le répertoire `test`. Nous pourrions indiquer dans ce fichier, entre autres, un ensemble d'éléments `var` possédant un attribut `name` et un attribut `value`. Ces éléments seront récupérables dans nos classes de test via un tableau associatif `$GLOBAL`.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--test/phpunit.xml-->
3 <phpunit>
4   <php>
5     <var name="DB_DRIVER" value="mysql" />
6     <var name="DB_USER" value="root" />
7     <var name="DB_PASSWORD" value="root" />
8     <var name="DB_HOST" value="localhost" />
9     <var name="DB_DATABASE" value="test_pdo" />
10   </php>
11 </phpunit>

```

Nous pouvons maintenant implémenter les méthodes `getConnection()`, `getDataset()` et `setUp()` de notre classe :

```

1 <?php
2
3 // test/fr/fyigrane/MysqlDaoTest.php
4
5 include './fr/fyigrane/MysqlDao.php';
6
7 use fr\fyigrane\MysqlDao;
8
9 class MysqlDaoTest extends PHPUnit_Extensions_Database_TestCase {
10
11     protected $connection;
12     protected $dao;
13
14     protected function getConnection() {
15         if ($this->connection === null) {
16             $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' .
17                 $GLOBALS['DB_HOST'] . ';dbname=' . $GLOBALS['DB_DATABASE'];
18             $this->connection = $this->createDefaultDBConnection(
19                 new PDO($connectionString, $GLOBALS['DB_USER'],
20                     $GLOBALS['DB_PASSWORD']));
21         }
22         return $this->connection;
23     }
24
25     protected function getDataSet() {
26         return new PHPUnit_Extensions_Database_DataSet_YamlDataSet(
27             './test/fyigrane/personne_base.yml');
28     }
29
30     protected function setUp() {
31         $conn = $this->getConnection();
32         // désactivation des contraintes de clés étrangères pour permettre
33         //le chargement des données via le dataset
34         $conn->getConnection()->query('set foreign_key_checks=0');
35         parent::setUp();
36         // activation des contraintes de clés étrangères
37         $conn->getConnection()->query('set foreign_key_checks=1');
38         $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' . $GLOBALS['DB_HOST'] .
39             ';dbname=' . $GLOBALS['DB_DATABASE'];
40         $this->dao = new MysqlDao($connectionString, $GLOBALS['DB_USER'],

```

```

41         $GLOBALS['DB_PASSWORD']);
42     }
43
44 }

```

Nous allons maintenant tester la méthode `addPersonne()` de `MysqlDao`. Pour cela nous allons, dans une méthode de test, appeler la méthode à partir de l'attribut d'instance `$dao`. Nous ajouterons donc un enregistrement à notre base de données de test.

Puis nous récupérerons un dataset qui correspondra à l'état de notre base de test après l'insertion, et nous le comparerons à un dataset prédéfini qui représentera l'état attendu de la base après une insertion. Si les deux datasets coïncident cela signifie que le test est réussi. Dans un fichier `add_personne.yml` nous allons créer notre dataset "résultat attendu". Ce fichier se trouvera dans le répertoire `test/dataset` :

```

1  # test/dataset/add_personne.yml
2
3  personne:
4      -
5          id: 1
6          nom: "Sparrow"
7          prenom: "Jack"
8      -
9          id: 2
10         nom: "Wayne"
11         prenom: "Bruce"
12      -
13         id: 3
14         nom: "Kent"
15         prenom: "Clark"

```

Nous implémentons la méthode `testAddPersonne()` :

```

1  <?php
2
3  // test/fr/fyigrane/MysqlDaoTest.php
4
5  include './fr/fyigrane/MysqlDao.php';
6
7  use fr\fyligrane\MysqlDao;
8
9  class MysqlDaoTest extends PHPUnit_Extensions_Database_TestCase {
10
11     protected $connection;
12     protected $dao;
13
14     protected function getConnection() {
15         if ($this->connection === null) {
16             $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' .
17                 $GLOBALS['DB_HOST'] . ';dbname=' . $GLOBALS['DB_DATABASE'];
18             $this->connection = $this->createDefaultDBConnection(
19                 new PDO($connectionString, $GLOBALS['DB_USER']
20                     , $GLOBALS['DB_PASSWORD']));
21         }
22         return $this->connection;
23     }
24
25     protected function getDataSet() {
26         return new PHPUnit_Extensions_Database_DataSet_YamlDataSet(
27             './test/dataset/personne_base.yml');
28     }
29
30     protected function setUp() {
31         $conn = $this->getConnection();
32         // désactivation des contraintes de clés étrangères pour permettre
33         // le chargement des données via le dataset

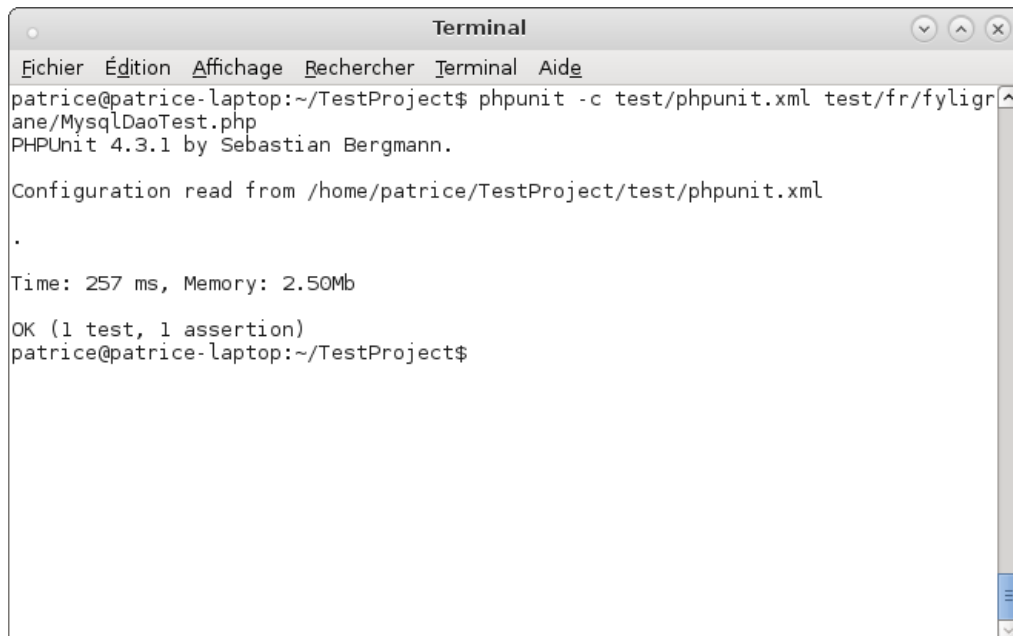
```

```

34     $conn->getConnection()->query('set foreign_key_checks=0');
35     parent::setUp();
36     // activation des contraintes de clés étrangères
37     $conn->getConnection()->query('set foreign_key_checks=1');
38     $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' .
39         $GLOBALS['DB_HOST'] . ';dbname=' . $GLOBALS['DB_DATABASE'];
40     $this->dao = new MysqlDao($connectionString, $GLOBALS['DB_USER']
41         , $GLOBALS['DB_PASSWORD']);
42 }
43
44 public function testAddPersonne() {
45     $this->dao->addPersonne("Kent", "Clark");
46     // création d'un objet dataset à partir de la connexion
47     $actualDataset = new PHPUnit_Extensions_Database_DataSet_QueryDataSet(
48         $this->getConnection());
49     // ajout à ce dataset des enregistrements de la table personne
50     // de notre base de données de test
51     $actualDataset->addTable('personne');
52     // récupération d'un dataset à partir de notre fichier
53     $expectedDataset = new PHPUnit_Extensions_Database_DataSet_YamlDataSet(
54         './test/dataset/add_personne.yml');
55     // comparaison des deux datasets
56     $this->assertDataSetsEqual($expectedDataset, $actualDataset);
57 }
58
59 }

```

Pour lancer le test, il faut passer à la commande phpunit une option qui indique le chemin vers le fichier de configuration phpunit.xml :



```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/TestProject$ phpunit -c test/phpunit.xml test/fr/fyigr
ane/MysqlDaoTest.php
PHPUnit 4.3.1 by Sebastian Bergmann.

Configuration read from /home/patrice/TestProject/test/phpunit.xml

.

Time: 257 ms, Memory: 2.50Mb

OK (1 test, 1 assertion)
patrice@patrice-laptop:~/TestProject$

```

Une autre possibilité consiste à récupérer le résultat d'une requête SQL, et de comparer cette table, issue de la requête, à une table du dataset "résultat attendu". Voici la dataset "résultat attendu", dans un fichier add_personne_2.yml :

```

1 # test/dataset/add_personne_2.yml
2
3 personne:
4 -
5     id: 3
6     nom: "Kent"
7     prenom: "Clark"

```

Nous ajoutons une méthode testAddPersonne2() à notre classe de test :

```

1  <?php
2
3  // test/fr/fyigrane/MysqlDaoTest.php
4
5  include './fr/fyigrane/MysqlDao.php';
6
7  use fr\fyigrane\MysqlDao;
8
9  class MysqlDaoTest extends PHPUnit_Extensions_Database_TestCase {
10
11     protected $connection;
12     protected $dao;
13
14     protected function getConnection() {
15         if ($this->connection === null) {
16             $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' .
17                 $GLOBALS['DB_HOST'] . ';dbname=' . $GLOBALS['DB_DATABASE'];
18             $this->connection = $this->createDefaultDBConnection(
19                 new PDO($connectionString, $GLOBALS['DB_USER']
20                     , $GLOBALS['DB_PASSWORD']));
21         }
22         return $this->connection;
23     }
24
25     protected function getDataSet() {
26         return new PHPUnit_Extensions_Database_DataSet_YamlDataSet(
27             './test/dataset/personne_base.yml');
28     }
29
30     protected function setUp() {
31         $conn = $this->getConnection();
32         // désactivation des contraintes de clés étrangères pour permettre
33         // le chargement des données via le dataset
34         $conn->getConnection()->query('set foreign_key_checks=0');
35         parent::setUp();
36         // activation des contraintes de clés étrangères
37         $conn->getConnection()->query('set foreign_key_checks=1');
38         $connectionString = $GLOBALS['DB_DRIVER'] . ':host=' .
39             $GLOBALS['DB_HOST'] . ';dbname=' . $GLOBALS['DB_DATABASE'];
40         $this->dao = new MysqlDao($connectionString, $GLOBALS['DB_USER']
41             , $GLOBALS['DB_PASSWORD']);
42     }
43
44     public function testAddPersonne() {
45         $this->dao->addPersonne("Kent", "Clark");
46         // création d'un objet dataset à partir de la connexion
47         $actualDataset = new PHPUnit_Extensions_Database_DataSet_QueryDataSet(
48             $this->getConnection());
49         // ajout à ce dataset des enregistrements de la table personne
50         // de notre base de données de test
51         $actualDataset->addTable('personne');
52         // récupération d'un dataset à partir de notre fichier
53         $expectedDataset = new PHPUnit_Extensions_Database_DataSet_YamlDataSet(
54             './test/dataset/add_personne.yml');
55         // comparaison des deux datasets
56         $this->assertDataSetsEqual($expectedDataset, $actualDataset);
57     }
58
59     public function testAddPersonne2() {
60         $this->dao->addPersonne("Kent", "Clark");
61         // création d'une table de résultat nommée personne
62         $queryTable = $this->getConnection()->createQueryTable(
63             'personne', 'SELECT * FROM personne where id=3'
64         );
65         $expectedDataset = new PHPUnit_Extensions_Database_DataSet_YamlDataSet(

```

```
66         './test/dataset/add_personne_2.yml');
67         // récupération de la table personne du dataset "résultat attendu"
68         $expectedTable = $expectedDataset->getTable("personne");
69         // comparaison des deux tables
70         $this->assertTablesEqual($expectedTable, $queryTable);
71     }
72 }
73 }
```

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

Petite application version 3

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

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