



Programmation orientée objet en PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [POO PHP](#) ► [Les tests avec PHPUnit](#) ► [Test des exceptions](#)

Test des exceptions

Dans la méthode `additionner()` de notre classe `Calculatrice`, nous allons jeter une exception si un des deux arguments n'est pas de type entier.

```
1 <?php
2
3 // fr/fyligrane/Calculatrice.php
4
5 namespace fr\fyligrane;
6
7 use Exception;
8
9 class Calculatrice {
10
11     public function additionner($nb1, $nb2) {
12         if (!is_integer($nb1) || !is_integer($nb2)) {
13             throw new Exception('les arguments doivent être de type entier');
14         }
15         return $nb1 + $nb2;
16     }
17
18     public function soustraire($nb1, $nb2) {
19         return $nb1 - $nb2;
20     }
21
22 }
```

Nous devons donc tester, dans le cas où un des deux arguments ne serait pas de type entier, que notre méthode jette bien une exception. Pour cela nous allons utiliser l'annotation `@expectedException`, qui va nous permettre de préciser le type d'Exception attendue :

```
1 <?php
2
3 // test/fr/fyligrane/CalculatriceTest.php
4
5 include_once './fr/fyligrane/Calculatrice.php';
6
7 use fr\fyligrane\Calculatrice;
8
9 class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     protected $calc;
12
13     /**
14      * @before
15      */
16     protected function init() {
17         $this->calc = new Calculatrice();
18     }
19
20     /**
21      * @test
22      */
23     public function additionner() {
24         $result = $this->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  * @expectedException Exception
33  */
34 public function additionnerWithException() {
35     $this->calc->additionner(10, "azerty");
36 }
37
38 /**
39  * @test
40  */
41 public function soustraire() {
42     $result = $this->calc->soustraire(15, 25);
43     $this->assertEquals(-10, $result);
44 }
45
46 }

```

Il est possible de tester le message de l'exception avec l'annotation `@expectedExceptionMessage` :

```

1 <?php
2
3 // test/fr/fyigrane/CalculatriceTest.php
4
5 include_once './fr/fyigrane/Calculatrice.php';
6
7 use fr\fyligrane\Calculatrice;
8
9 class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     protected $calc;
12
13     /**
14      * @before
15      */
16     protected function init() {
17         $this->calc = new Calculatrice();
18     }
19
20     /**
21      * @test
22      */
23     public function additionner() {
24         $result = $this->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      * @expectedException Exception
33      * @expectedExceptionMessage les arguments doivent être de type entier
34      */
35     public function additionnerWithException() {
36         $this->calc->additionner(10, "azerty");
37     }
38
39     /**
40      * @test
41      */

```

```

42     public function soustraire() {
43         $result = $this->calc->soustraire(15, 25);
44         $this->assertEquals(-10, $result);
45     }
46
47 }

```

Ou encore l'annotation , en utilisant une expression rationnelle (aussi dite régulière) :

```

1  <?php
2
3  // test/fr/fyigrane/CalculatriceTest.php
4
5  include_once './fr/fyigrane/Calculatrice.php';
6
7  use fr\fyligrane\Calculatrice;
8
9  class CalculatriceTest extends PHPUnit_Framework_TestCase {
10
11     protected $calc;
12
13     /**
14      * @before
15      */
16     protected function init() {
17         $this->calc = new Calculatrice();
18     }
19
20     /**
21      * @test
22      */
23     public function additionner() {
24         $result = $this->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      * @expectedException Exception
33      * @expectedExceptionMessageRegExp /type entier/
34      */
35     public function additionnerWithException() {
36         $this->calc->additionner(10, "azerty");
37     }
38
39     /**
40      * @test
41      */
42     public function soustraire() {
43         $result = $this->calc->soustraire(15, 25);
44         $this->assertEquals(-10, $result);
45     }
46
47 }

```

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](#)