



The PHP Company

# Tester votre application MVC Zend Framework

Mickaël Perraud  
Contributeur Zend Framework  
Responsable documentation française

# Qu'allons-nous voir ensemble ?

---

- Les bases du test unitaire
- Les bases du test fonctionnel avec ZF
- Quelques sujets de tests “avancés” avec ZF

# Pourquoi tester ?

# Simplifier la maintenance

---

- Tester les attentes souhaitées
- Tester les comportements décrits et identifiés dans l'application
- Tester nous informe quand des changements cassent du code ou des comportements existants

# Quantifier la qualité du code

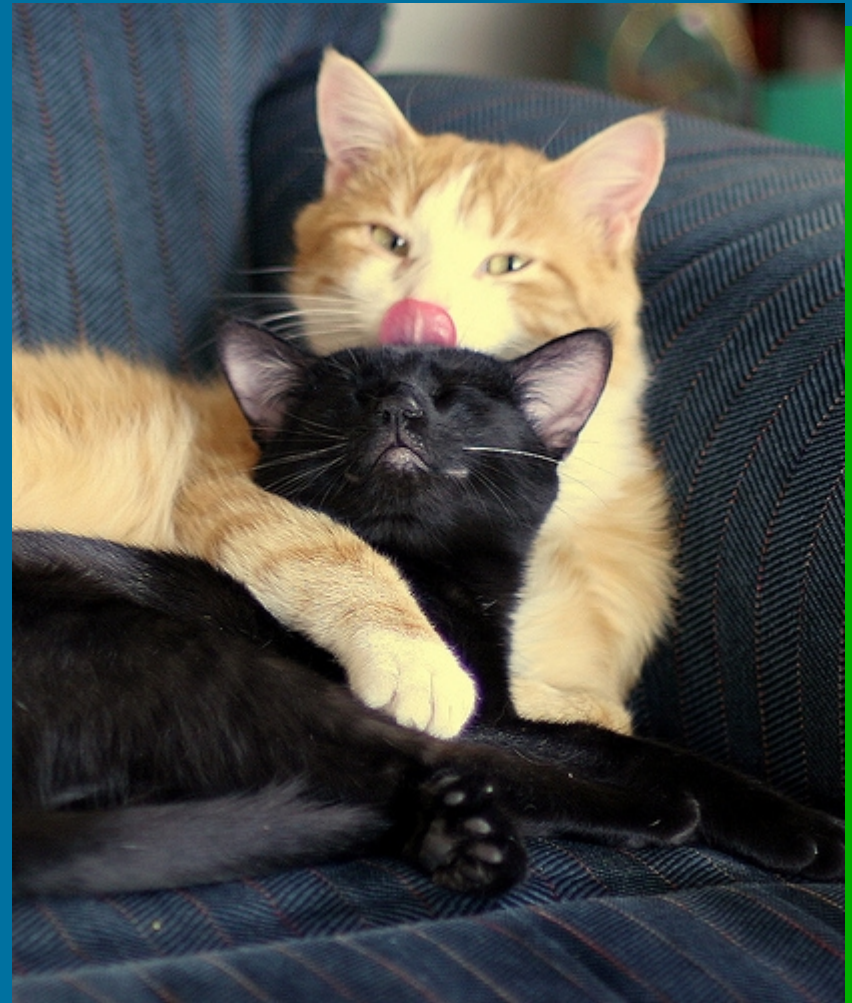
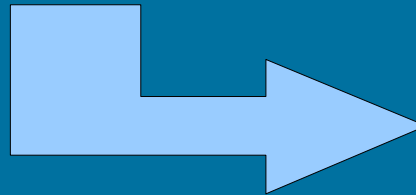
---

- Couverture du code exercée par les tests unitaires
- Tester les méthodes documente les comportements que le code devrait définir

# Bénéfices psychologiques

*On se sent toujours mieux quand on voit du vert*

```
mikaelkael@mikaelkael-laptop:tests (master%)  
$ phpunit --colors Zend/Dom  
PHPUnit 3.4.15 by Sebastian Bergmann.  
.....  
Time: 1 second, Memory: 4.25Mb  
OK (38 tests, 58 assertions)
```



# Tester n'est pas ... recharger

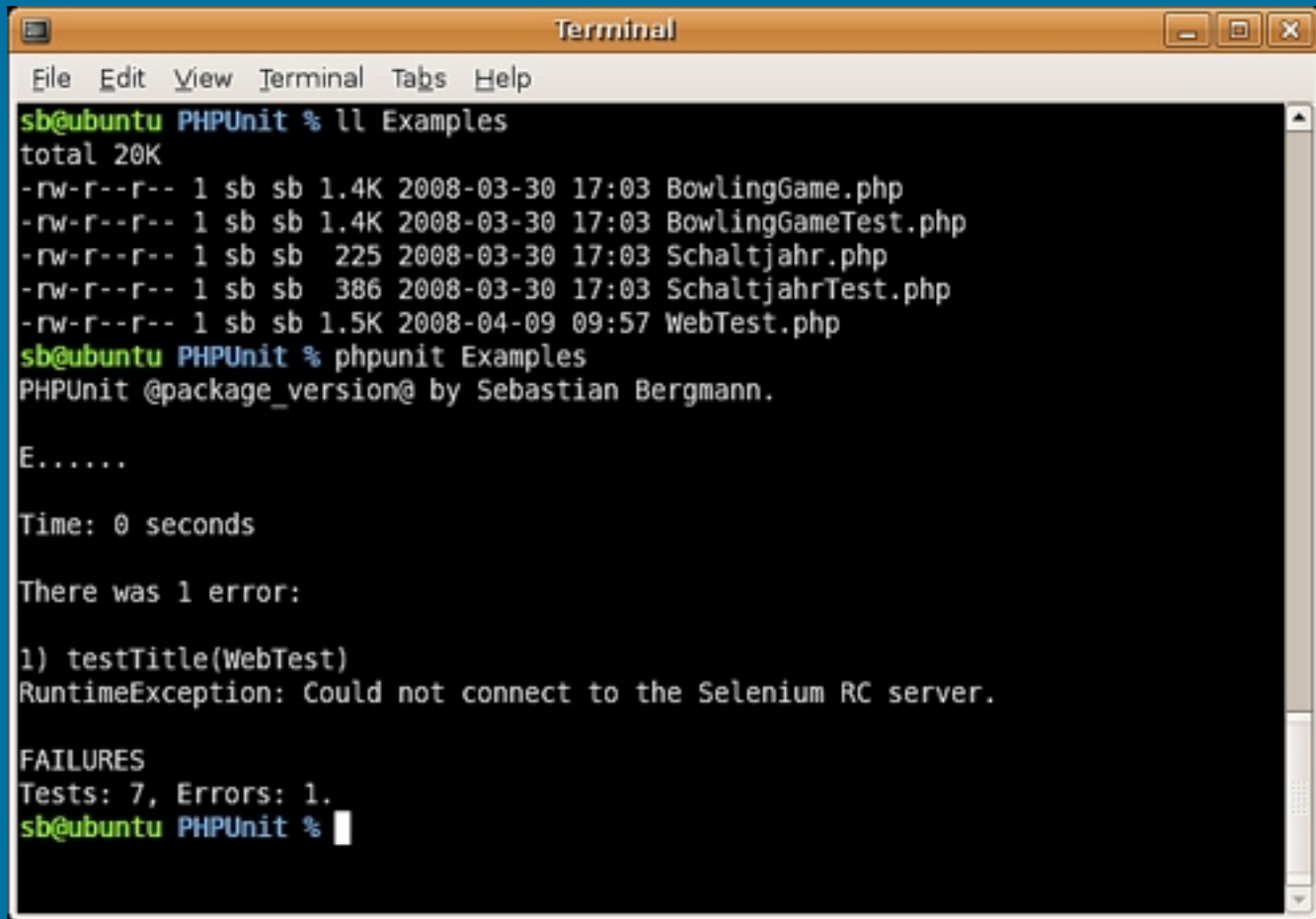


# Tester n'est pas ... var\_dump()



```
array(10) =>
  ["date"] =>
    string(0) ""
  ["event_date"] =>
    string(0) ""
  ["wksp_type"] =>
    string(0) ""
  ["wksp_vers"] =>
    string(0) ""
  ["wksp_num"] =>
    string(7) "F096341"
  ["search"] =>
    string(15) "Search (Alt-S)"
  ["/cgi-bin/auth_secure.cgi_L"] =>
    string(10) "1188159806"
  ["PHPSESSID"] =>
    string(32) "0a6938bflc9a941b5270d2...75600"
}
```

# Tester est ... reproducible



```
Terminal
File Edit View Terminal Tabs Help
sb@ubuntu PHPUnit % ll Examples
total 20K
-rw-r--r-- 1 sb sb 1.4K 2008-03-30 17:03 BowlingGame.php
-rw-r--r-- 1 sb sb 1.4K 2008-03-30 17:03 BowlingGameTest.php
-rw-r--r-- 1 sb sb 225 2008-03-30 17:03 Schaltjahr.php
-rw-r--r-- 1 sb sb 386 2008-03-30 17:03 SchaltjahrTest.php
-rw-r--r-- 1 sb sb 1.5K 2008-04-09 09:57 WebTest.php
sb@ubuntu PHPUnit % phpunit Examples
PHPUnit @package_version@ by Sebastian Bergmann.

E.....

Time: 0 seconds

There was 1 error:

1) testTitle(WebTest)
RuntimeException: Could not connect to the Selenium RC server.

FAILURES
Tests: 7, Errors: 1.
sb@ubuntu PHPUnit %
```

# Tester est ... automatisable

The screenshot displays the phpUnderControl web interface. At the top, the project is identified as 'phpuc-merge-test' and the build as 'More builds'. The interface includes navigation tabs for Overview, Tests, Metrics, Coverage, Documentation, CodeSniffer, and PHPUnit PMD. The 'Tests' tab is active, showing a table of test results for 'Example::PhpUnderControl\_Example\_MathTest'.

Name	Status	Time(s)
Example::PhpUnderControl_Example_MathTest		0.101
testAddSuccess	Success	0.012
#1 - testAddSuccess - (build: <b>php-5.3.0</b> )	Success	0.006
#2 - testAddSuccess - (build: <b>php-5.2.6</b> )	Success	0.006
#3 - testAddSuccess - (build: <b>php-5.2.5</b> )	Success	0.001
testSubSuccess	Success	0.012
#1 - testSubSuccess - (build: <b>php-5.3.0</b> )	Success	0.006
#2 - testSubSuccess - (build: <b>php-5.2.6</b> )	Success	0.006
#3 - testSubSuccess - (build: <b>php-5.2.5</b> )	Success	0.000
testSubFail	Failure	0.015
#1 - testSubFail - (build: <b>php-5.3.0</b> )	Failure	
#2 - testSubFail - (build: <b>php-5.2.6</b> )	Failure	
#3 - testSubFail - (build: <b>php-5.2.5</b> )	Failure	
testDataProviderOneWillFail	Success	0.051
testDataProviderOneWillFail - (build: <b>php-5.3.0</b> )		0.024
#1 - testDataProviderOneWillFail with data set #0	Success	0.006
#2 - testDataProviderOneWillFail with data set #1	Success	0.006
#3 - testDataProviderOneWillFail with data set #2	Failure	
#4 - testDataProviderOneWillFail with data set #3	Success	0.006
testDataProviderOneWillFail - (build: <b>php-5.2.6</b> )		0.024
#1 - testDataProviderOneWillFail with data set #0	Success	0.006
#2 - testDataProviderOneWillFail with data set #1	Success	0.006
#3 - testDataProviderOneWillFail with data set #2	Failure	
#4 - testDataProviderOneWillFail with data set #3	Success	0.006
testDataProviderOneWillFail - (build: <b>php-5.2.5</b> )		0.002
#1 - testDataProviderOneWillFail with data set #0	Success	0.001
#2 - testDataProviderOneWillFail with data set #1	Success	0.000
#3 - testDataProviderOneWillFail with data set #2	Failure	
#4 - testDataProviderOneWillFail with data set #3	Success	0.001
testFail	Failure	0.011
#1 - testFail - (build: <b>php-5.3.0</b> )	Failure	
#2 - testFail - (build: <b>php-5.2.6</b> )	Failure	
#3 - testFail - (build: <b>php-5.2.5</b> )	Failure	

# Les bons tests incluent ...

---

- Les comportements définis
- Des exemples de code de cas d'utilisation
- Les attentes

# Les frameworks de tests PHP

---

- PHPT
  - ▶ Utilisé par PHP, PEAR et quelques librairies indépendantes
- SimpleTest
  - ▶ Framework de test de type JUnit
- PHPUnit
  - ▶ Framework de test de type Junit
  - ▶ Standard industriel *de fait*

# Les bases du test

# Ecrire des tests unitaires

---

- Créez une classe de test
- Créez une ou deux méthodes qui décrivent des comportements
  - ▶ Énoncez les comportements dans un langage naturel
- Écrivez le code qui crée le(s) comportement(s)
  - ▶ Écrivez le code exerçant l'API
- Écrivez les assertions indiquant les attentes

# Création d'une classe de test

---

- Habituellement nommé par le nom de classe à tester suivi de Test

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
}
```

# Créez une méthode qui décrit un comportement

- Préfixé avec “test”

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
    }
}
```

# Écriture du code qui crée le comportement

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
        $string = 'Fri, 7 May 2010 09:26:03 -0700';
        $ts      = strtotime($string);
        $this->entry->setTimestamp($string);
        $setValue = $this->entry->getTimestamp();
    }
}
```

# Écriture des assertions qui vérifient les attentes

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
        $string = 'Fri, 7 May 2010 09:26:03 -0700';
        $ts     = strtotime($string);
        $this->entry->setTimestamp($string);
        $setValue = $this->entry->getTimestamp();
        $this->assertSame($ts, $setValue);
    }
}
```

# Lancement des tests

---

- Echech ?
  - ▶ Vérifiez vos tests et vos assertions en cas d'éventuelles fautes typographiques ou d'erreurs d'utilisation
  - ▶ Vérifiez que le code que vous testez ne contient pas d'erreurs
  - ▶ Faîtes les corrections et relancez les tests
- Succès ?
  - ▶ Passez au comportement ou à la fonctionnnnalité suivante !

# Un peu de terminologie associée aux tests

# Scaffolding (“échaffaudage”) des tests

---

- Vérifiez que votre environnement est indépendant de toute hypothèse
- Initialisez les dépendances nécessaires avant de lancer les tests
- Habituellement réalisé dans la méthode “setUp()”

# Doublures de test

---

- **Stubs**

*Remplacement d'un objet par un autre afin que le système puisse continuer son exécution.*

- **Mock Objects**

*Remplacement d'un objet par un autre en redéfinissant ce qu'on attend de lui.*

- **“Mocks Aren't Stubs”** (Martin Fowler) => <http://bruno-orsier.developpez.com/mocks-arent-stubs/>

# Autres types de tests

---

- **Test conditionnel**

*Le test ne s'exécute que si des conditions sont respectées.*

- **Tests fonctionnels et d'intégration**

*On teste que le système est conforme à ce que l'on attend ; on teste ensuite que l'on interagit correctement avec lui.*

# Tests quasi-fonctionnel avec Zend Framework

# Vue d'ensemble

---

- Préparation de l'environnement de PHPUnit
- Création d'un TestCase basé sur ControllerTestCase
- Amorçage (“bootstrap”) de l'application
- Création et distribution (“dispatch”) de la requête
- Réalisation des assertions sur la réponse

# L'environnement PHPUnit

---

- Structure des répertoires

```
tests
|-- application
|   |-- controllers
|-- Bootstrap.php
|-- library
|   |-- Mkk
|-- phpunit.xml

4 dossiers, 2 fichiers
```

# L'environnement PHPUnit

- Le fichier phpunit.xml

```
<phpunit bootstrap="./Bootstrap.php">
  <testsuite name="Suite de tests">
    <directory>./</directory>
  </testsuite>
  <filter>
    <whitelist>
      <directory
        suffix=".php">../library/</directory>
      <directory
        suffix=".php">../application/</directory>
      <exclude>
        <directory
          suffix=".phtml">../application/</directory>
        </exclude>
      </whitelist>
    </filter>
  </phpunit>
```

# L'environnement PHPUnit

- Le fichier Bootstrap.php

```
$rootPath = realpath(dirname(__DIR__));  
if (!defined('APPLICATION_PATH')) {  
    define('APPLICATION_PATH',  
          $rootPath . '/application');  
}  
if (!defined('APPLICATION_ENV')) {  
    define('APPLICATION_ENV', 'testing');  
}  
set_include_path(implode(PATH_SEPARATOR, array(  
    '.',  
    $rootPath . '/library',  
    get_include_path(),  
)));  
require_once 'Zend/Loader/Autoloader.php';  
$loader = Zend_Loader_Autoloader::getInstance();  
$loader->registerNamespace('Mkk_');
```

# Création d'une classe de test

---

- En étendant  
Zend\_Test\_PHPUnit\_ControllerTestCase

```
class ExempleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
}
```

# Amorçage de l'application

- Création d'une instance de Zend\_Application, et référencement de celle-ci dans votre méthode setUp()

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    public function setUp()
    {
        $this->bootstrap = new Zend_Application(
            APPLICATION_ENV,
            APPLICATION_PATH
            . '/configs/application.ini'
        );
        parent::setUp();
    }
}
```

# Création et distribution d'une requête

- Méthode simple : on distribue une “url”

```
class ExempleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testPageStatiqueAUneBonneStructure ()
    {
        $this->dispatch ( '/exemple/page' );
        // ...
    }
}
```

# Création et distribution d'une requête

- Méthode avancée : on personnalise l'objet de requête avant la distribution

```
class ExempleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testRequeteXhrRetourneJson()
    {
        $this->getRequest()
            ->setHeader('X-Requested-With',
                       'XMLHttpRequest')
            ->setQuery('format', 'json');
        $this->dispatch('/exemple/xhr-endpoint');
        // ...
    }
}
```

# Création des assertions

---

- Les assertions concernent typiquement :
  - ▶ La structure des balises de la réponse  
*Soit en utilisant des sélecteurs CSS ou par une recherche XPath.*
  - ▶ Les entêtes de réponse HTTP ou le code de statut
  - ▶ Les éléments de l'objet de Request et/ou Response

# Assertions via sélecteur CSS

---

- `assertQuery($path, $message = "")`
- `assertQueryContentContains($path, $match, $message = "")`
- `assertQueryContentRegex($path, $pattern, $message = "")`
- `assertQueryCount($path, $count, $message = "")`
- `assertQueryCountMin($path, $count, $message = "")`
- `assertQueryCountMax($path, $count, $message = "")`
- **chacun possède une variante "Not"**

# Assertions via sélecteur XPath

---

- `assertXPath($path, $message = "")`
- `assertXPathContentContains($path, $match, $message = "")`
- `assertXPathContentRegex($path, $pattern, $message = "")`
- `assertXPathCount($path, $count, $message = "")`
- `assertXPathCountMin($path, $count, $message = "")`
- `assertXPathCountMax($path, $count, $message = "")`
- **chacun possède une variante "Not"**

# Assertions de redirection

---

- `assertRedirect($message = "")`
- `assertRedirectTo($url, $message = "")`
- `assertRedirectRegex($pattern, $message = "")`
- **chacun possède une variante "Not"**

# Assertions sur la réponse

---

- `assertResponseCode($code, $message = "")`
- `assertHeader($header, $message = "")`
- `assertHeaderContains($header, $match, $message = "")`
- `assertHeaderRegex($header, $pattern, $message = "")`
- **chacun possède une variante "Not"**

# Request assertions

---

- `assertModule($module, $message = "")`
- `assertController($controller, $message = "")`
- `assertAction($action, $message = "")`
- `assertRoute($route, $message = "")`
- **each has a "Not" variant**

# Exemples d'assertion

---

```
public function testPageStatiqueAUneBonneStructure()  
{  
    $this->dispatch('/exemple/page');  
    $this->assertResponseCode(200);  
    $this->assertQuery('div#content p');  
    $this->assertQueryCount('div#sidebar ul li', 3);  
}
```

# Exemples d'assertion

---

```
public function testRequeteXhrRetourneJson()  
{  
    // ...  
    $this->assertNotRedirect();  
    $this->assertHeaderContains(  
        'Content-Type', 'application/json');  
}
```

# Sujets avancés : cas d'utilisation réels

# Test des modèles et des ressources

---

- **Le problème :**  
*Ces classes ne peuvent pas être autoloadées avec l'autoloader standard.*
- **La solution :**  
*Utiliser Zend\_Application pour amorcer les ressources lors de l'appel de setUp()*

# Test des modèles et des ressources

```
class Blog_Model_EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function setUp()
    {
        $this->bootstrap = new Zend_Application(
            APPLICATION_ENV,
            APPLICATION_PATH
            . '/configs/application.ini'
        );
        $this->bootstrap->bootstrap('modules');

        $this->model = new Blog_Model_Entry();
    }
}
```

# Test d'actions nécessitant une authentification

---

- **Le problème :**  
*Certaines actions peuvent nécessiter un utilisateur authentifié ; comment émuler ceci ?*
- **La solution :**  
*Authentifier manuellement au travers de `Zend_Auth` avant d'appeler `dispatch()`.*

# Authentication d'un utilisateur test

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function loginUser($user)
    {
        $params = array('user' => $user);
        $adapter = new Custom_Auth_TestAdapter(
            $params);
        $auth = Zend_Auth::getInstance();
        $auth->authenticate($adapter);
        $this->assertTrue($auth->hasIdentity());
    }
}
```

# Authentification suivi de la distribution

---

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testAdminUserCanAccessAdmin()
    {
        $this->loginUser('admin');
        $this->dispatch('/example/admin');
        $this->assertQuery('div#content.admin');
    }
}
```

# Test de pages dépendant des actions précédentes

---

- **Le problème :**

*Certaines actions sont dépendantes d'autres ; par ex., la récupération d'une page avec du contenu mis en exergue sur la base d'une chaîne recherchée.*

- **La solution:**

*Distribuer 2 fois en réinitialisant la requête et la réponse entre chacun des appels.*

# Test de pages dépendant des actions précédentes

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testHighlightedTextAfterSearch()
    {
        $this->getRequest()->setQuery(
            'search', 'foobar');
        $this->dispatch('/search');

        $this->resetRequest();
        $this->resetResponse();

        $this->dispatch('/example/page');
        $this->assertQueryContains(
            'span.highlight', 'foobar');
    }
}
```

# Conclusions

# Testez en permanence !

---

- Testez unitairement vos modèles, vos couches de service, etc.
- Réalisez des tests fonctionnels pour valider vos workflows, structure de page, etc.

# Certification & formations Zend Framework

---

- **Zend Framework : les fondamentaux**  
Ce cours vous apprend à utiliser le ZF et surtout le design pattern Model-View-Controller (MVC), afin que vous connaissiez les bonnes pratiques de développement PHP

Prochains cours : Du 20 au 30 septembre en ligne - Du 13 au 15 octobre en classe

# Certification & formations Zend Framework

---

- **Zend Framework : les concepts avancés**

Ce cours est conçu pour apprendre aux développeurs PHP qui travaillent déjà avec le Zend Framework comment mettre en place les bonnes pratiques et configurer les applications pour une meilleure scalabilité, interactivité et performance.

Prochains cours : du 4 au 6 octobre - en classe

# Certification & formations Zend Framework

---

- **Test Prep : la certification Zend Framework**  
Cette formation prépare les développeurs expérimentés qui conçoivent des applications PHP avec le ZF pour qu'ils augmentent leurs chances d'obtenir la certification ZF et deviennent Zend Certified Engineer in Zend Framework (ZCE-ZF).
- **Study Guide gratuit**  
<http://www.zend.com/en/download/173>

# Forum AFUP 2010 !

---

- 9-10 novembre 2010
- <http://www.afup.org/pages/forumphp2010/>
- Quelques conférences sur le ZF avec notamment G. Delamarre pour une introduction au ZF et J. Pauli et M. Perraud pour une session sur ZF 2.0