RENATER

CONNECTEUR DE SAVOIRS

Process:

Unit tests implementation for PHP project

Rennes, 21/08/2014

CONTROL SHEET AND BROADCASTING OF THE DOCUMENT

Versions		
Version	**Date**	**Subject**
1.0	21/08/2014	Initialization of the document

Diffusion			
Name	**Society**	**Number of copy**	**Mode**
Etienne Méléard	Renater	1	Mail

LEVEL 1			LEVEL 2			LEVEL 3		
Transmitter (Writer)			Controller			Approving		
Name	**Date**	**Visa**	**Name**	**Date**	**Visa**	**Name**	**Date**	**Visa**
Damien Bruchet	**21/08/2014**	**DABR**						

I. Table of content

1) Situation, context

This document gives some specifications for implementing unit tests on PHP project.

2) Prerequisites

1. PHP

The PHP version used is 5.4.24 (PHP 5 minimum).

2. PHPUnit and PEAR

PHPUnit is an open-source framework used for unit test.

PEAR (PHP Extension and Application Repository) is a PHP API collection.

- Installation of PEAR on MAC OS:

Cf. http://coolestguidesontheplanet.com/installing-pear-osx-10-9-mavericks-osx10-810-7/

- Installation of PHPUnit on MAC OS:

Cf. http://imzank.com/2012/08/how-to-install-phpunit-with-mamp/

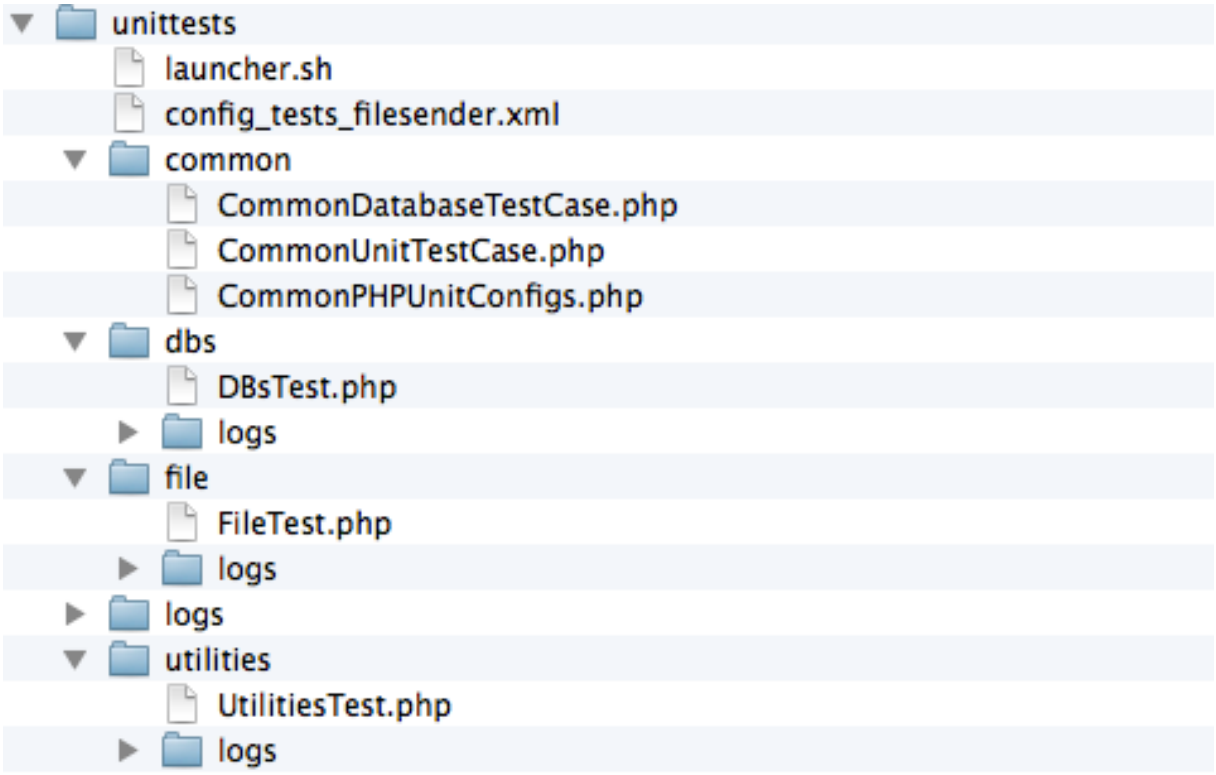
NB : the MAC OS version used here is 10.9.4.

3) Implementing unit tests

a) File tree view

Implementing unit tests as preconized in this document should take in place the following file tree view:

At the project root is located a folder named “unittests” including:



NB: this document take for example the filesender-2.0 project (http://filesender.org).

- **launcher.sh**

This script allows to automate the unit tests launch (cf. annex).

→ Cf. part *d) Running* page XX.

- **config_tests_filesender.xml**

This file includes the optional configuration for running unit tests with PHPUnit.

→ Cf. part b*) Configuration* page XX.

- **common > CommonDatabaseTestCase.php**

This file include 3 elements:

- ***DBOperations***: correspond with different operations CRUD applied to databases
- ***DBErrors***: correspond with different error codes linked to databases operations
- ***CommonDatabaseTestCase***: abstract class that should be extended by each files including unit tests linked to CRUD operations.

It includes four abstract functions to be redefined in child classes. Theses correspond with CRUD operations (Create, Read, Update, Delete).

It also includes two functions that formalize out messages from unit tests.

It implements PHPUnit_Framework_TestCase interface, needed to run unit tests with PHPUnit.

- **common > CommonUnitTestCase.php**

This file includes an abstract class that should be extended by each file that includes unit tests.

It includes two functions that formalize out messages from unit tests.

It implements PHPUnit_Framework_TestCase interface, needed to run unit tests with PHPUnit.

- **common > CommonPHPUnitConfigs**

This file includes the whole configurations, file inclusion, and other elements necessary for good proceedings of unit tests running.

→ Cf. part b*) Configuration* page XX.

- **example > logs**

This folder includes log and results files thrown broadcasted by PHPUnit after unit tests running.

Folders "***dbs***", "***files***", and "***utilities***" provided on annex include some unit tests samples for the filesender-2.0 project.

b) Configuration

- Environment configuration

This configuration should be indicated in the file common > CommonPHPUnitConfigs.php.

Example :

```
date_default_timezone_set("Europe/Paris");

require_once('../../classes/autoload.php');
require_once('../../classes/_includes.php');
```

- PHPUnit configuration file (xml)

This configuration file is an xml file interpreted by PHPUnit.

Example:

```
<phpunit>
 <testsuites>
  <testsuite name="ExampleTestSuite" >
   <file>example/ExampleTest.php</file>
   <file>example/DBsTest.php</file>
  </testsuite>
  <testsuite name="ExampleBDDTestSuite" >
   <file>example/DBsTest.php</file>
  </testsuite>

 </testsuites>
</phpunit>
```

Running the following command will launch the unit tests located in *example/ExampleTest.php* file and *example/DbsTest.php* file:

```
./launcher.sh –c configuration.xml
```

c) Development of tests

Here is an example allowing to test succinctly that the date passed as parameter has a good date format after calling the function Utilities::formatFate($timestamp).

Cf. filesender-2.0 project.

```
require_once dirname(__FILE__).'/../common/CommonUnitTestCase.php';

/**
 * @backupGlobals disabled
 */
class UtilitiesTest extends CommonUnitTestCase {
  protected  $objectUtilities;

  /**
   * Init variables, first function called
   */
  protected function setUp() {
    echo "@ ".date("Y-m-d H:i:s")."\n\n";
  }



  public function testFormatDate(){
    //Test
    $timestamp = strtotime("2014-09-04");

    try{
      $this->assertTrue(Utilities::formatDate($timestamp) == "04-09-2014" ) ;

      $this->displayInfo(get_class(), __FUNCTION__,'');
    }catch (PHPUnit_Framework_AssertionFailedError $ex){
      $this->displayError(get_class(), __FUNCTION__, $ex->getMessage());
      throw new PHPUnit_Framework_AssertionFailedError();
    }

    return true;
  }
}
```

d) Execution

This file “launcher.sh” can be used to automatize unit tests execution.

Usage:

```
./launcher.sh testFolder [testfile.php] [[-c [testconfig.xml]] [-ts [testSuiteName]]]
```

It is possible to run all tests present on a folder running this command:

```
./launcher.sh dossierTest
```

To specify a specific test file, you need to run this command:

```
./launcher.sh dossierTest fichierTest.php
```

It is possible to use an xml configuration file (cf. PHPUnit documentation) by using this command:

```
./launcher.sh dossierTest –c configuration.xml
```

To run a specific testsuite defined in the xml configuration file, run the following command:

```
./launcher.sh dossierTest –c configuration.xml -ts testSuiteName
```