SensioLabs

Symfony

Hacking and Extending Symfony2
SF2C3

Introduction to unit testing

What's unit testing?

« In computer programming, unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. »

Wikipedia

Why unit testing?

- Ensure the code works and fits specifications
- Make the source code more robust
- Make the code more maintainable
- Make the code more evolutive and long lasting
- Save time while developing!

Pros of unit testing…

- Ensure the code works and fits specifications
- Avoid regressions when code evolves
- Find and eliminate bugs
- Leverage unexpected edge cases
- Ease code refactoring

Pros of unit testing…

- Document the source code with formal cases
- Ease migrations between two versions
- Organize the development process
- Ease production deployments
- **Be confident toward the source code!!!**

Cons of unit testing…

- Budgetize and spend time for unit testing
- Maintain the tests suite up to date
- Testing implies to be rigorous
- **Difficulties to know what to test…**

Follow the FIRST rule

- Tests suite must run as **fast** as possible
- Tests must be **isolated** from each other
- Tests must be **repeated** indefinitely
- Tests must be **self-validating** (pass or fail)
- Tests must come **timely**

When to write unit tests?

- Before implementing the code (TDD)
- After implementing the code
- Discovery of a new bug or edge case
- Know how a piece of code works

Introduction to PHPUnit

What's PHPUnit?

- De facto unit testing framework for PHP
- Written by Sebastian Bergmann
- Released under the BSD License
- Write and run unit tests suite
- JUnit and TAP standards compliant

Supported features in PHPUnit

- IDE integration (Eclipse, NetBeans, PHPStorm…)
- Mocks and stubs API
- Reports generation (coverage, TAP, JUnit…)
- Command Line Interface tool
- Coupling with Continuous Integration tools

http://phpunit.de

sebastianbergmann / phpunit

Unwatch | Fork | 1,188 | 166

Code | Network | Pull Requests 21 | Issues 94 | Stats & Graphs

The PHP Unit Testing framework. — Read more
http://www.phpunit.de/manual/current/en/index.html

Clone in Mac | ZIP | HTTP | Git Read-Only | https://github.com/sebastianbergmann/phpunit.git | Read-Only access

Files | Commits | Branches 11 | Tags 202 | Downloads

Current branch: 3.6

Latest commit to the **3.6** branch

Make PHP_CodeCoverage_Filter object injectable.

sebastianbergmann authored about 4 hours ago — commit 916dd9c520

phpunit /

name	age	message	history
PHPUnit/	about 4 hours ago	Make PHP_CodeCoverage_Filter object injectable. [sebastianbergmann]	
Tests/	1 day ago	Remove leftover from syntax check functionality. [sebastianbergmann]	
build/	October 26, 2011	Remove rule. [sebastianbergmann]	
.gitignore	October 26, 2011	Refactor build system. [sebastianbergmann]	
ChangeLog.markdown	about 4 hours ago	Update ChangeLog [edorian]	

Installing PHPUnit from PEAR

```
# Upgrade to the latest version of PEAR
$ (sudo) pear upgrade -f

# Configure PEAR auto discovering
$ pear config-set auto_discover 1

# Install PHPUnit
$ pear install pear.phpunit.de/PHPUnit
```

```
$ phpunit --version

$ PHPUnit 3.6.10 by Sebastian Bergmann.
```

```
PHPUnit 3.6.2 by Sebastian Bergmann.

Usage: phpunit [switches] UnitTest [UnitTest.php]
       phpunit [switches] <directory>

  --log-junit <file>        Log test execution in JUnit XML format to file.
  --log-tap <file>          Log test execution in TAP format to file.
  --log-json <file>         Log test execution in JSON format.

  --coverage-clover <file>  Generate code coverage report in Clover XML format.
  --coverage-html <dir>     Generate code coverage report in HTML format.
  --coverage-php <file>     Serialize PHP_CodeCoverage object to file.
  --coverage-text <file>    Generate code coverage report in text format.

  --testdox-html <file>     Write agile documentation in HTML format to file.
  --testdox-text <file>     Write agile documentation in Text format to file.

  --filter <pattern>        Filter which tests to run.
  --group ...               Only runs tests from the specified group(s).
  --exclude-group ...       Exclude tests from the specified group(s).
  --list-groups             List available test groups.

  --loader <loader>         TestSuiteLoader implementation to use.
  --printer <printer>       TestSuiteListener implementation to use.
  --repeat <times>          Runs the test(s) repeatedly.
```

The Model Class

```php
namespace Sensio\Bundle\BankBundle\Business;

class Account
{
    private $balance;

    public function __construct($balance = 0)
    {
        $this->balance = $balance;
    }

    public function getBalance()
    {
        return $this->balance;
    }
}
```

Writing assertions

Testing the default balance

```
namespace Sensio\Bundle\BankBundle\Tests\Business;

use Sensio\Bundle\BankBundle\Business\Account;

class AccountTest extends \PHPUnit_Framework_TestCase
{
    public function testBalanceIsZero()
    {
        $account = new Account();

        $this->assertSame(0, $account->getBalance());
    }
}
```

Default

```
Hugo:Bank Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /Users/Hugo/Sites/Sensio/Bank/app/phpunit.xml.dist


.

Time: 0 seconds, Memory: 6.25Mb

OK (1 test, 1 assertion)
Hugo:Bank Hugo$
```

$ phpunit -c app/phpunit.xml.dist

Common assertion methods

Method	Meaning
assertSame($a, $b)	Expects two values are equal
assertTrue($value)	Expects the tested value is true
assertFalse($value)	Expects the tested value is false
assertNull($value)	Expects the tested value is null
assertContains($value, $array)	Expects the value is in the array
assertRegex($regex, $string)	Expects the string matches the regular expression
assertCount($count, $array)	Expects the array contains $count values

Failing Tests

```
Default
Hugo:Bank Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /Users/Hugo/Sites/Sensio/Bank/app/phpunit.xml.dist

F

Time: 0 seconds, Memory: 6.50Mb

There was 1 failure:

1) Sensio\Bundle\BankBundle\Test\Business\AccountTest::testBalanceIsZero
Failed asserting that 0 matches expected 10.

/Users/Hugo/Sites/Sensio/Bank/src/Sensio/Bundle/BankBundle/Tests/Business/AccountTest.ph
:12

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
Hugo:Bank Hugo$
```

Output when a test fails…

Testing Exceptions

```php
namespace Sensio\Bundle\BankBundle\Business;

class Account
{
    // ...

    public function withdraw($money)
    {
        if ($money < 0) {
            throw new AccountException('...');
        }

        $this->balance -= $money;

        return $this->balance;
    }
}
```

Testing exceptions

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    public function testWithdrawNegativeAmount()
    {
        $this->setExpectedException('Sensio\Bundle
\BankBundle\Business\AccountException');

        $account = new Account(500);
        $account->withdraw(-1000);
    }
}
```

Default

```
Hugo:Bank Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /Users/Hugo/Sites/Sensio/Bank/app/phpunit.xml.dist

..

Time: 0 seconds, Memory: 6.25Mb

OK (2 tests, 2 assertions)
Hugo:Bank Hugo$
```

$ phpunit -c app/phpunit.xml.dist

Data Providers

Data providers provide data sets in order to run a single unit test several times with several values

```php
class AccountTest extends \PHPUnit_Framework_TestCase
{
    /**
     * @dataProvider provideMoneyToWithdraw
     */
    public function testWithdrawMoney($money, $balance)
    {
        $account = new Account(1000);

        $this->assertEquals(
            $balance,
            $account->withdraw($money)
        );
    }
}
```

Using data providers

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    public function provideMoneyToWithdraw()
    {
        return array(
            array(100, 900),
            array(500, 500),
            array(2000, -1000),
        );
    }
}
```

```
Default
Hugo:Bank Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.6.10 by Sebastian Bergmann.

Configuration read from /Users/Hugo/Sites/Sensio/Bank/app/phpunit.xml.dist

.....

Time: 0 seconds, Memory: 6.50Mb

OK (5 tests, 5 assertions)
Hugo:Bank Hugo$
```

$ phpunit -c app/phpunit.xml.dist

Code Coverage

The code coverage rate gives the number of lines of the code that are covered by unit tests.

Code coverage report

BankBundle

Current directory: /Users/Hugo/Sites/Sensio/Bank/src/Sensio/Bundle/BankBundle (dashboard)

Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

	Coverage								
	Lines			Functions / Methods			Classes		
Total		44.44%	8 / 18		50.00%	3 / 6		33.33%	2 / 6
Business		100.00%	8 / 8		100.00%	3 / 3		100.00%	2 / 2
Controller		0.00%	0 / 1		0.00%	0 / 1		0.00%	0 / 1
DependencyInjection		0.00%	0 / 8		0.00%	0 / 2		0.00%	0 / 2
SensioBankBundle.php		0.00%	0 / 1		100.00%			0.00%	0 / 1

Generated by PHP_CodeCoverage 1.1.2 using PHP 5.3.10 and PHPUnit 3.6.10 at Fri Mar 30 10:58:17 CEST 2012.

```
$ phpunit -c app/phpunit.xml.dist --coverage-html ./coverage
```

Code coverage report

	Coverage									
	Classes			Functions / Methods				Lines		
Total		100.00%	1 / 1		100.00%	3 / 3	CRAP		100.00%	7 / 7
Account		100.00%	1 / 1		100.00%	3 / 3	4		100.00%	7 / 7
__construct($balance = 0)					100.00%	1 / 1	1		100.00%	2 / 2
withdraw($money)					100.00%	1 / 1	2		100.00%	4 / 4
getBalance()					100.00%	1 / 1	1		100.00%	1 / 1

```
  1     : <?php
  2     :
  3     : namespace Sensio\Bundle\BankBundle\Business;
  4     :
  5     : class Account
  6     : {
  7     :     private $balance;
  8     :
  9     :     public function __construct($balance = 0)
 10     :     {
 11   5 :         $this->balance = $balance;
 12   5 :     }
 13     :
 14     :     public function withdraw($money)
 15     :     {
 16   4 :         if ($money < 0) {
 17   1 :             throw new AccountException('...');
 18     :         }
 19     :
 20   3 :         $this->balance -= $money;
 21     :
 22   3 :         return $this->balance;
 23     :     }
 24     :
 25     :     public function getBalance()
 26     :     {
 27   1 :         return $this->balance;
 28     :     }
```

Test Doubles

Test doubles by Gerard Meszaros

« Sometimes it is just plain hard to test the system under test (SUT) because it depends on other components that cannot be used in the test environment. This could be because they aren't available, they will not return the results needed for the test or because executing them would have undesirable side effects. »

XUnitPatterns.com

Generic Terms - Dummies

Dummies are objects that are passed around but never used. They are usually used to fill a list of parameters.

Generic Terms - Stubs

Stubs are objects that implement the same methods than the real object. These methods do nothing and are configured to return a specific value.

```php
namespace Sensio\Bundle\BankBundle\Business;

class Account
{
    // ...
    private $currencyExchange;

    public function setCurrencyExchange(CurrencyExchange $ce)
    {
        $this->currencyExchange = $ce;
    }

    public function withdraw($money, $currency = 'EUR')
    {
        $rate = $this->currencyExchange->getExchangeRate('EUR', $currency);

        // ...
        $this->balance -= $money / $rate;

        return $this->balance;
    }
}
```

Stubbing the Currency Exchange object

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    // ...
    public function testWithdrawForeignCurrencies()
    {
        $stub = $this
            ->getMock('Sensio\[…]\CurrencyExchange')
        ;

        // Configure the stub
        $stub
            ->method('getExchangeRate')
            ->will($this->returnValue(1.20))
        ;
    }
}
```

Stubbing the Currency Exchange object

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    // ...
    public function testWithdrawForeignCurrencies()
    {
        // ...
        $account = new Account(1000);
        $account->setCurrencyExchange($stub);
        $balance = $account->withdraw(300, 'USD')

        $this->assertSame(750, $balance);
    }
}
```

Generic Terms - Mocks

Mocks are pre-programmed with expectations which form a specification of the calls they are expected to receive. They can throw an exception if they receive a call they don't expect and are checked during verification to ensure they got all the calls they were expecting.

Mocking the Currency Exchange object

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    // ...

    public function testWithdrawForeignCurrencies()
    {
        $mock = $this
            ->getMockBuilder('Sensio\[..]\CurrencyExchange')
            ->disableOriginalConstructor()
            ->setMethods(array('getExchangeRate'))
            ->getMock()
        ;
    }
}
```

Mocking the Currency Exchange object

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    // ...
    public function testWithdrawForeignCurrencies()
    {
        // ...
        // Configure the mock
        $mock
          ->expects($this->once())
          ->method('getExchangeRate')
          ->with($this->equalTo('EUR'), $this->equalTo('USD'))
          ->will($this->returnValue(1.20))
        ;
    }
}
```

Mocking the Currency Exchange object

```
class AccountTest extends \PHPUnit_Framework_TestCase
{
    // ...
    public function testWithdrawForeignCurrencies()
    {
        // ...

        $account = new Account(1000);
        $account->setCurrencyExchange($mock);
        $balance = $account->withdraw(300, 'USD');

        $this->assertSame(750, $balance);
    }
}
```

Introduction to functional testing

What's functional testing?

« Functional tests is a kind of black box testing that checks the integration of the different layers of an application. »

Functional tests in Symfony

Symfony emulates an HTTP Client to perform Request and provides a Crawler object to test the Response.

Functional tests in Symfony

```
class DemoControllerTest extends WebTestCase
{
    public function testIndex()
    {
        $client = static::createClient();

        $crawler = $client->request('GET', '/hello/Fabien');

        $items = $crawler->filter('html:contains("Hello
Fabien")');

        $this->assertCount(1, $items);
    }
}
```

The Client

Method	Meaning
request($method, $uri [,…])	Calls a URI with a specific HTTP method
back()	Goes back in the browser history
forward()	Goes forward in the browser history
reload()	Reloads the current browser
restart()	Restarts the current browser
insulate()	Insulates the request in a separate PHP process
click(Link $link)	Clicks on a link
submit(Form $form)	Submits a form

Method	Meaning
getRequest()	Returns a HttpFoundation\Request object
getResponse()	Returns the HttpFoundation\Response object
getCrawler()	Returns the DOM Crawler object
getHistory()	Returns the BrowserKit\History object
getCookieJar()	Returns the BrowserKit\CookieJar object
getKernel()	Returns the application Kernel object
getContainer()	Returns the Dependency Injection Container object
getProfile()	Returns the HttpKernel\Profile object

```php
class DemoControllerTest extends WebTestCase
{
    public function testIndex()
    {
        $client = static::createClient();
        $client->request('GET', '/demo/hello/Fabien');

        $response = $client->getResponse();
        $this->assertTrue($response->isSuccessful());

        $profile = $client->getProfile();
        $time    = $profile->getCollector('time');

        $this->assertLessThan(300, $time->getTotalTime());
    }
}
```

Manual redirects

```
$client->request('POST', '/some/uri');
$crawler = $client->followRedirect();
```

Automatic redirects

```
$client->followRedirects(true);
$crawler = $client->request('POST', '/uri');
```

The Crawler

Using the Crawler

```
// Filtering the response with CSS or XPath selectors
$tweets = $crawler->filter('#sidebar .tweet');
$tweets = $crawler->filterXPath('//p[class="tweet"]');

// Traversing
$first = $tweets->first();
$third = $tweets->eq(2);
$last  = $tweets->last();

// Extracting
$text  = $first->text();
$class = $first->attr('class');
$infos = $first->extract(array('_text', 'class'));
```

Reducing a selection

```
$filter = function ($node, $i) {
    $content = (string) $node->textContent;
    if (!preg_match('/symfony/i', $content)) {
        return false;
    }
};

$tweets = $crawler->reduce($filter);
```

Finding links

```
$link = $crawler->selectLink('Click me')->link();

$crawler = $client->click($link);
```

Finding forms

```
$form = $crawler->selectButton('send')->form();

$client->submit($form, array('name' => 'Foo'));
```

Method name	Meaning
filter($expr)	Filters the DOM with a CSS3 selector
filterXPath($expr)	Filters the DOM with an XPath expression
eq(1)	Returns the Node of a special index
first()	Returns the first Node of the list
last()	Returns the last node of the list
siblings()	Returns all sibling nodes
nextAll()	Returns all following siblings
previousAll()	Returns all previous siblings

Method name	Meaning
parents()	Returns all ancestor nodes
children()	Return all child nodes
reduce($lambda)	Reduce the list with a lambda function
each($lambda)	Executes a lambda function on each element of the list
attr($name)	Returns an attribute value
extract($items)	Returns the list of extracted values
text()	Returns the text node

The Profile

Data Collector Name	Meaning
time	Returns the TimeDataCollector
security	Returns the SecurityDataCollector object
request	Returns the RequestDataCollector object
memory	Returns the MemoryDataCollector object
logger	Returns the LoggerDataCollector object
exception	Returns the ExceptionDataCollector object
event	Returns the EventDataCollector object
config	Returns the ConfigDataCollector object

Some collector examples

```
// time
$collector = $profile->getCollector('time');
$total     = $collector->getTotalTime();

// doctrine
$collector = $profile->getCollector('doctrine');
$queries   = $collector->getQueryCount();
$time      = $collector->getTime();

// memory
$collector = $profile->getCollector('memory');
$memory    = $collector->getMemory();
```

Training Department

SensioLabs Training

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

FRANCE

Phone: +33 140 998 211

Email: training@sensiolabs.com

symfony.com - trainings.sensiolabs.com