Executable Documentation for everyone (even you)
Who ...
... has heard of it?
... is using it?
Me:  

Nikolas Martens  

Engineer & Coach
Me:

Testing

Design
mockster

Testing + NIH syndrome
Mockster is a full-fledged, zero-configuration mocking framework for PHP.

Main Features

- Automatic mocking of dependencies, return values, method and constructor arguments
- Support of BDD-style testing by defining the context first and asserting expectations second
- Fine-grained configuration of the behaviour
Basic Usage

First, we need an instance of MockFactory. It extends Factory so it supports singletons and providers (if needed).

```php
$factory = new MockFactory();
```

To get a completely empty mock which is but a hollow shell of the given class, use

```php
$mock = $factory->getInstance('MyClass');
```

The created instance extends the given class but does not invoke its parent's constructor, nor does any method call actually reach the parent - they are all mocked.

If you want to call the parent's constructor, pass an array with the constructor arguments. If you don't want to pass any arguments, provide an empty array.

```php
$mock = $factory->getInstance('MyClass', array('name' => 'Foo'));
```
Basic Usage

First, we need an instance of MockFactory. It extends `Factory` (needed).

```
$factory = new MockFactory();
```

To get a completely empty mock which is but a hollow shell of the given class.

```
$mock = $factory->getInstance('MyClass');
```

The created instance extends the given class but does not invoke its parent's constructor, nor does any method call actually reach the parent - they are all mocked.

If you want to call the parent's constructor, pass an array with the constructor arguments. If you don't want to pass any arguments, provide an empty array.

```
$mock = $factory->getInstance('MyClass', array('name' => 'Foo'));
```
First, we need an instance of MockFactory. It extends Factory (needed).

```php
$factory = new MockFactory();
```

To get a completely empty mock which is but a hollow shell of the given class:

```php
$mock = $factory->getInstance('MyClass');
```

The created instance extends the given class but does not invoke its parent's constructor, nor does any method call actually reach the parent - they are all mocked.

If you want to call the parent's constructor, pass an array with the constructor arguments. If you don't want to pass any arguments, provide an empty array.

```php
$mock = $factory->getInstance('MyClass', array('name' => 'Foo'));
```
Basic Usage

First, we need an instance of Mock needed).

```php
$factory = new MockFactory();
```

To get a completely empty mock which is but a hollow:

```php
$mock = $factory->getInstance('MyClass');
```

The created instance extends the given class but method call actually reach the parent - they are all mocked.

If you want to call the parent’s constructor, pass an array with the constructor arguments. If you don't want to pass any arguments, provide an empty array.

```php
$mock = $factory->getInstance('MyClass', array('name' => 'Foo'));
```
Basic Usage

First, we need an instance of Mock (function needed).

```php
$factory = new MockFactory();
```

To get a completely empty mock which is but a holio-

```php
$mock = $factory->getInstance('MyClass');
```

The created instance extends the given class but method call actually reach the parent - they are all mocked.

If you want to call the parent’s constructor, pass an array with the constructor arguments. If you don't want to pass any arguments, provide an empty array.

```php
$mock = $factory->getInstance('MyClass', array('name' => 'Foo'));
```
Mockster build passing

test

.php

executable

```php
///<br/>
function test() {
  //
  $result = (true);
  $expected = true;
  $actual = $result;
  
  //
  return $actual == $expected;
```
Validate syntax

$mock = $factory->getInstance('MyClass');
$foo = $mock->_mock()->method('foo');

$foo->getHistory()->wasCalledWith(['bar']);
Validate functionality

```php
$mock = $factory->getInstance('MyClass');
$foo = $mock->_mock()->method('foo');
$mock->foo('bar');

$this->assertTrue($foo->getHistory()->wasCalledWith([['bar']]));
$this->assertFalse($foo->getHistory()->wasCalledWith([['baz']]));
```
Test ← Documentation
APIs

Libraries
$response = $router->respond('/foo/bar', '{"some":"query"}');

$this->assertEquals($response, '{"some":"data"}');
APIs

```php
$this->whenIRequest_From('{some:"query"}', '/foo/bar');

$this->thenTheResponseShouldBe('{"some":"data"}');
```

ubiquitous language
APIs
Libraries
Anything
Example-Driven Development
Acceptance Test Driven
Behaviour Driven Development

★ Anything ★
Agile Testing
Example-Driven Development
Acceptance Test Driven Development
Behaviour Driven Development
Specification by Example
Agile Testing
Specification

Test

Documentation
Specification

Test \rightarrow Documentation
Examples
$this->whenIRequest_From('{some: "query"}', '/foo/bar');

$this->thenTheResponseShouldBe('{"some": "data"}');
Free delivery for VIP customers more than 5 articles only books
Examples may be?
VIP customer with five books in the cart gets free delivery.
VIP customer with four books in the cart doesn’t get free delivery.
Regular customer with five books in the cart doesn’t get free delivery.
VIP customer with a five washing machines in the cart doesn’t get free delivery.
VIP customer with five books and a washing machine in the cart doesn’t get free delivery.
<table>
<thead>
<tr>
<th>Customer</th>
<th>Order</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIP</td>
<td>5 books</td>
<td>free</td>
</tr>
<tr>
<td>VIP</td>
<td>4 books</td>
<td>not</td>
</tr>
<tr>
<td>Reg</td>
<td>10 books</td>
<td>not</td>
</tr>
<tr>
<td>VIP</td>
<td>5 TV</td>
<td>not</td>
</tr>
<tr>
<td>VIP</td>
<td>5 books 1 TV</td>
<td>not</td>
</tr>
<tr>
<td>Customer Order</td>
<td>Delivery</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>VIP</td>
<td>5 books</td>
<td>free</td>
</tr>
</tbody>
</table>

```php
$this->givenIAmAVipCustomer();
$this->givenIHave_BooksInMyBasket(5);
$this->whenICheckMyDeliveryOptions();
$this->thenTheDeliveryShouldBeFree();
```
function givenIAmAVipCustomer() {  
    $this-&gt;customer = new Customer();  
    $this-&gt;customer-&gt;setVip(true);  
}

function givenIHave_BooksInMyBasket($number) {  
    $this-&gt;basket = new Basket();  
    for ($i=0; $i&lt;$number; $i++) {  
        $this-&gt;basket-&gt;getItems()-&gt;put(new Book());  
    }  
}

function whenICheckMyDeliveryOptions() {  
}

function thenTheDeliveryShouldBeFree() {  
}
function givenIAmAVipCustomer() {
    $this->customer = new Customer();
    $this->customer->setVip(true);
}

function givenIHave_BooksInMyBasket($number) {
    $this->basket = new Basket();
    for ($i=0; $i<$number; $i++) {
        $this->basket->getItems()->put(new Book());
    }
}

function whenICheckMyDeliveryOptions() {
    $delivery = new DeliveryManager($this->customer, $this->basket);
    $this->isFree = $delivery->isDeliveryFree();
}

function thenTheDeliveryShouldBeFree() { 
}
function givenIAmAVipCustomer() {
    $this->customer = new Customer();
    $this->customer->setVip(true);
}

function givenIHave_BooksInMyBasket($number) {
    $this->basket = new Basket();
    for ($i=0; $i<$number; $i++) {
        $this->basket->getItems()->put(new Book());
    }
}

function whenICheckMyDeliveryOptions() {
    $delivery = new DeliveryManager($this->customer, $this->basket);
    $this->isFree = $delivery->isDeliveryFree();
}

function thenTheDeliveryShouldBeFree() {
    $this->assertTrue($this->isFree);
}
Specification by Example
right thing

do

thing right
Specs → do → Tests

1. Docs
Specs \rightarrow \text{Code} \rightarrow \text{Tests} \rightarrow HTML \rightarrow dox
Single source of truth

doX
<?php

class "MyClass"
{
    methods:
    {
        name: "content",
        ...
    }

    eruser: parsedown
}

<html>
<h1>MyClass</h1>
<div>Method</div>
<p>comment</p>
<p code>...</p>

dox pipeline
once again...
Examples
more on SbE

http://specificationbyexample.com/

http://dannorth.net/introducing-bdd/

http://skillsmatter.com/podcast/agile-testing/how-to-sell-bdd-to-the-business

the book

the beginning

good talk
more good stuff

http://www.clean-code-developer.de/

http://www.extremeprogramming.org/

http://theleanstartup.com/

☑️ read the book

root of agile

➡️ stop waste