Mokka PHP Mocking Framework Documentation

Release 0.0.1

Sebastian Heuer

May 09, 2015

Contents

CHAPTER 1

Index

1.1 Installation

1.1.1 Composer

Simply add belanur/mokka to the composer.json of your project. Use “dev-master” for the latest version:

```
{
  "require-dev": {
    "belanur/mokka": "dev-master"
  }
}
```

1.1.2 Checking out the latest sources

Mokka sources are available on GitHub. Simply clone them and checkout the desired branch (master should be fine in most cases):

```
$ git clone https://github.com/belanur/mokka
Cloning into 'mokka'...
remote: Counting objects: 745, done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 745 (delta 59), reused 0 (delta 0)
Receiving objects: 100% (745/745), 101.76 KiB | 0 bytes/s, done.
Resolving deltas: 100% (395/395), done.
Checking connectivity... done
$
```

If you want to contribute to Mokka, fork it!

1.2 Creating Mocks

Mocks are created with Mokka::mock() along with the name of the class that you want to mock. Starting with PHP 5.5 you can use the ‘class’ keyword, which is highly recommended for better refactoring support.

```
<?php
$mock = Mokka::mock(SampleClass::class);
```

If you are using PHP 5.4 (which is the minimum required for Mokka), you can alternatively pass the class name as a string.

```
<?php
$mock = Mokka::mock('SampleClass');
```

The huge drawback of this is that your IDE won't recognize this as a class name, meaning your mocks will break if you rename 'SampleClass' to something else with a refactoring tool.

1.2.1 Using Mocks

The created Mock implements all methods of the mocked class (plus a few internal methods needed for mocking and stubbing). All methods will return NULL when you call them. See Mocking and Stubbing for information on how to change the behaviour of methods.

1.3 Mocking

Mocking a method lets you verify that a method was called with the given arguments.

```
<?php
$mock = Mokka::mock(SampleClass::class);

// Verify sure that the method getBar() gets called once
Mokka::verify($mock)->getBar();
```

You can use optional Invokation Rules with Mokka::verify():

```
<?php
// Verify sure that the method getBar() is never called
Mokka::verify($mock, Mokka::never())->getBar();

// Make sure getBar() gets called at least twice
Mokka::verify($mock, Mokka::atLeast(2)->getBar();

// Make sure getBar() gets called exactly three times
Mokka::verify($mock, Mokka::exactly(3)->getBar();
```

You can add multiple mocks for a single method with different arguments

```
<?php
// Make sure getBar() gets called once with the argument 'foo' and once with argument 'bar'
Mokka::verify($mock)->getBar('foo');
Mokka::verify($mock)->getBar('bar');
```

1.3.1 AnythingArgument

There is also a special AnythingArgument, so you don't have to verify every single argument if it is not relevant for your test.

```
<?php
// Make sure getBar() gets called with the second argument 'foo'. The first argument can be anything
Mokka::verify($mock)->getBar(Mokka::anything(), 'foo');
```

1.4 Stubbing

Stubbing a method lets you define a return value. A stubbed method does not have an Invokation Rule (like mocked methods), so if a stubbed method is not called, no exception is thrown.

```
<?php
$mock = Mokka::mock(SampleClass::class);

// getFoo() should return 'baz' when called with the argument 'bar'
Mokka::when($mock)->getFoo('bar')->thenReturn('baz');

echo $mock->getFoo(): // => NULL
echo $mock->getFoo('bar'); // => 'baz'
```

You can also use the special AnythingArgument introduced in Mocking here:

```
<?php
// getFoo() should always return 'baz'
Mokka::when($mock)->getFoo(Mokka::anything())->thenReturn('baz');

echo $mock->getFoo('foo'): // => 'baz'
echo $mock->getFoo('bar'); // => 'baz'
```

1.4.1 Combining Stubs and Mocks

You can combine mocks and stubs if you want to verify that a method gets called and also want to set a return value:

```
<?php
$mock = Mokka::mock(SampleClass::class);

// getFoo() should return 'baz' when called with the argument 'bar'
Mokka::when($mock)->getFoo('bar')->thenReturn('baz');
// also make sure that getFoo() gets called once
Mokka::verify($mock)->getFoo('bar');

echo $mock->getFoo('bar'); // => 'baz'
```

1.4.2 Throwing Exceptions

A stubbed method can throw an exception instead of returning a value:

```
<?php
$mock = Mokka::mock(SampleClass::class);
Mokka::when($mock)->getFoo()->thenThrow(new \InvalidArgumentException());

$mock->getBar(); // => throws exception
```

1.5 Mokka with PHPUnit

Mokka comes with the MokkaTestCase class, which provides easy access to mocking functions and adds support for PHPUnit

```php
<?php
class FooTest extends MokkaTestCase
{
    public function testFoo()
    {
        $mock = $this->mock(SampleClass::class);
        $foo = new Foo($mock);
    }
}
```