Mokka PHP Mocking Framework Documentation

Release 0.0.1

Sebastian Heuer

09.05.2015

Inhaltsverzeichnis

KAPITEL 1

Index

1.1 Installation

1.1.1 Composer

belanur/mokka kann einfach zur composer.json des Projekts hinzugefügt werden. Die aktuelle Version erhält man mit "dev-master":

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

1.1.2 Aktuellen Quellcode auschecken

Der Mokka Quellcode ist auf GitHub verfügbar. Der Branch master sollte in den meisten Fällen verwendet werden.

```
$ 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
$
```

Um an Mokka mitzuarbeiten, sollte ein Fork erzeugt werden!

1.2 Mocks erzeugen

Mocks werden mit Mokka::mock() unter Angabe des Namens der Klasse, die gemockt werden soll, erzeugt.Ab PHP 5.5 kann dafür das 'class' Schlüsselwort verwendet werden,was wegen des besseren Refactoring-Supports empfohlen wird.

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

Mit PHP 5.4 (die minimale Version, mit der Mokka funktioniert) muss der Klassenname als String übergeben werden

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

Diese Methode hat den großen Nachteil, dass IDEs den String nicht als Klassennamen erkennen.Dadurch muss der Test manuell korrigiert werden, wenn 'SampleClass' mit einem Refactoring-Tool umbenannt wird.

1.2.1 Mocks verwenden

Der erzeugte Mock implementiert alle Methoden der gemockten Klasse(plus ein paar interne Methoden, die zum Mocken und Stubben benötigt werden). Alle Methoden geben NULL zurück. Weitere Informationen zu Mocks und Stubs finden sichunter Mocken und Stubben.

1.3 Mocken

Via Mocking kann sichergestellt werden, dass eine Methode mit den vorgegebenen Argumenten aufgerufen wurde.

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

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

An Mokka::verify() kann optional eine Invokation Rule übergeben werden:

```
<?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();
```

Eine Methode kann mehrfach mit unterschiedlichen Argumenten gemockt werden:

```
<?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

Mit dem AnythingArgument ist es nicht nötig, jedes einzelne Argument vorzugeben.

```
<?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 Stubben

Mit dem Stubben kann der Rückgabewert einer Methode vorgegeben werden.Eine gestubbte Methode hat keine Invokation Rule (wie gemockte Methoden). Wird eine gestubbte Methode nicht aufgerufen, wird folglich keine Exception geworfen.

```
<?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'
```

Hier kann auch das in Mocken eingeführte AnythingArgument verwendert werden:

```
<?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 Stubs und Mocks kombinieren

Mocks und Stubs können kombiniert werden, um einen Rückgabewert zu setzen und zusätzlich sicherzustellen, dass die Methode aufgerufen wird:

```
<?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 Exceptions werfen

Eine gestubbte Methode kann eine Exception werfen, anstatt einen Wert zurückzugeben:

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

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

1.5 Mokka mit PHPUnit

Mokka bringt die Klasse MokkaTestCase mit, die einfachen Zugriff auf die Mokka-Funktionen bietet und zusätzlichen Support für PHPUnit bereitstellt.

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