loophp/lambada documentation

Release 1.0.0

Pol Dellaiera

Feb 11, 2020

Contents

Lambada is a utility library for PHP.

This library:

- provide immutable classes,
- is extendable,
- leverages the power of functional programming,
- relies on [tacit programming](https://en.wikipedia.org/wiki/Tacit_programming) by having relevant user arguments are the very end of the function call (data-last),
- uses S.O.L.I.D. principles,
- fully tested,
- framework agnostic,
- uses PHP-FIG recommendations (PSR-4 and PSR-12).

All the methods of each classes are pure and thus predictable.

Lambada has been inspired by:

- Haskell programming language
- Python programming language
- Lambda calculus
- Clojure
- Rambda
- PHP package grrr-amsterdam/garp-functional
- PHP package lstrojny/functional-php

It uses the following PHP Standards Recommendations :

- PSR-4 for classes autoloading,
- PSR-12 for coding standards.

CHAPTER 1

Requirements

1.1 PHP

PHP greater than 7.1.3 is required.

1.2 Dependencies

The sole dependency of Lambada is loophp/combinator.

CHAPTER 2

Installation

The easiest way to install it is through Composer

```
composer require loophp/lambada
```

CHAPTER 3

Usage

Verify some functors laws:

```php
<?php

declare(strict_types=1);

include 'vendor/autoload.php';

use loophp\lambada\Map;
use loophp\lambada\Id;
use loophp\lambada\Compose;

$data = [1, 2, 3];

Map::off()(Id::off())($data) === Id::off()($data); // true

$sub20 = static function ($x) {
    return $x - 20;
};

$times7 = static function ($x) {
    return $x * 7;
};

Map::off()(Compose::off()($sub20, $times7))($data) === Compose::off()(Map::off()(
↪$sub20), Map::off()($times7))($data); // true
```

CHAPTER 4

Examples

CHAPTER 5

API

CHAPTER 6

Tests, code quality and code style

Every time changes are introduced into the library, Github Actions run the tests.

Tests are written with PHPSpec.

PHPInfection is also triggered used to ensure that your code is properly tested.

The code style is based on PSR-12 plus a set of custom rules. Find more about the code style in use in the package drupol/php-conventions.

A PHP quality tool, Grumphp, is used to orchestrate all these tasks at each commit on the local machine, but also on the continuous integration tools.

To run the whole tests tasks locally, do

```
composer grumphp
```

or

```
./vendor/bin/grumphp run
```

Here's an example of output that shows all the tasks that are setup in Grumphp and that will check your code

```
$ ./vendor/bin/grumphp run
GrumPHP is sniffing your code!
Running task  1/13: SecurityChecker... ✓
Running task  2/13: Composer... ✓
Running task  3/13: ComposerNormalize... ✓
Running task  4/13: YamlLint... ✓
Running task  5/13: JsonLint... ✓
Running task  6/13: PhpLint... ✓
Running task  7/13: TwigCs... ✓
Running task  8/13: PhpCsAutoFixerV2... ✓
Running task  9/13: PhpCsFixerV2... ✓
Running task 10/13: Phpcs... ✓
Running task 11/13: PhpStan... ✓
Running task 12/13: Phpspec... ✓
```

(continues on next page)

(continued from previous page)

```
Running task 13/13: Infection... ✓
$
```

CHAPTER 7

Contributing

See the file CONTRIBUTING.md but feel free to contribute to this library by sending Github pull requests.