PHPでテスティングフレ…ryを実装する前に知っておきたい勘所

Tips for implementing Testing Framework in PHP.

2018-03-09 PHPerKaigi 2018 前夜祭
Nerima Coconeri Hall #phperkaigi

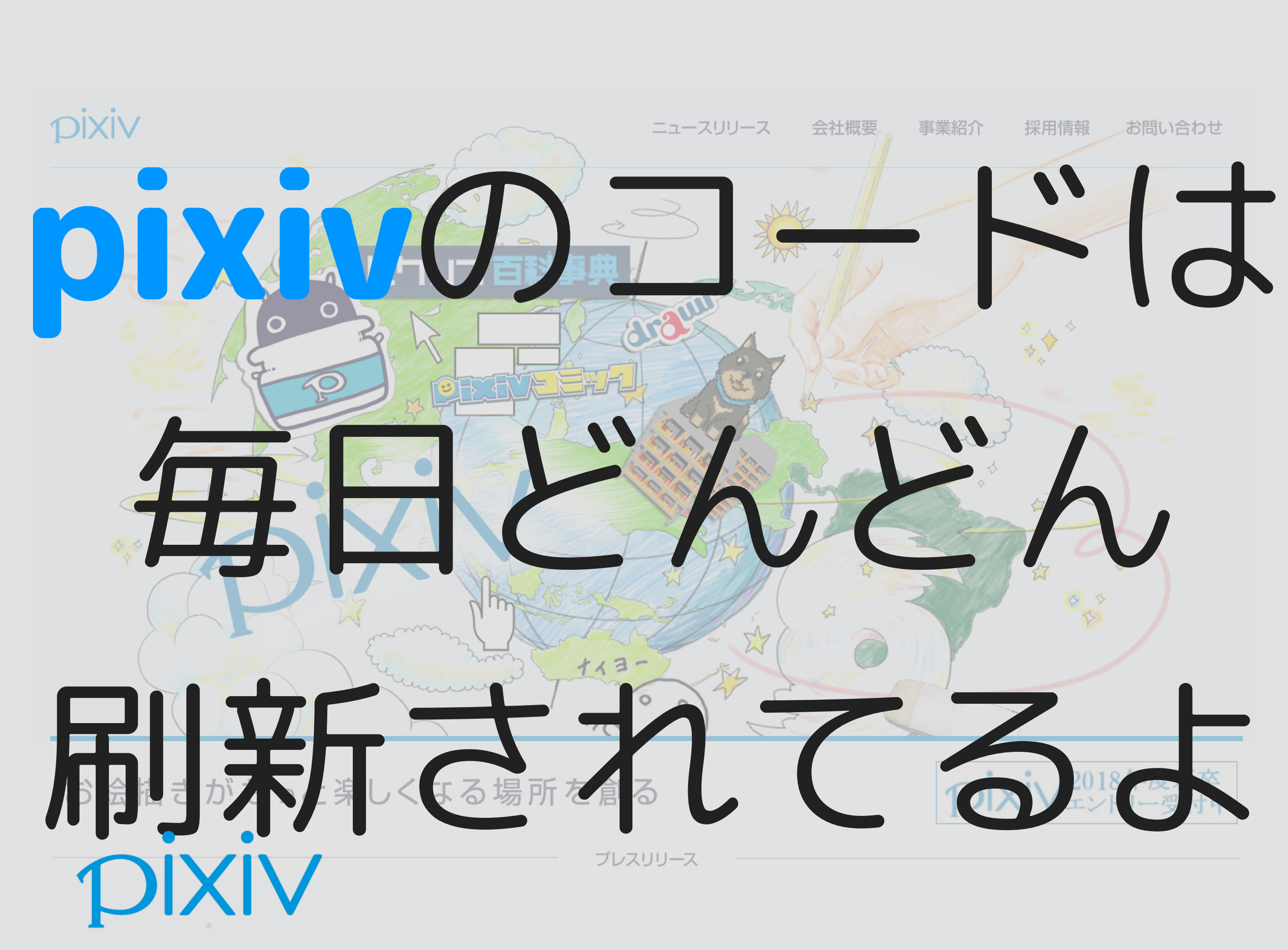
お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id: zonuexe
- ピクシブ株式会社技術基盤チーム（pixiv.net）
- Emacs Lisper, PHPer
  - Emacs PHP Modeのメンテナ引き継ぎました
  - 好きなリスプはEmacs Lispです
- Qiitaに記事を書いたり変なコメントしてるよ

pixiv

お絵描きがもっと楽しくなる場所を創る

プレスリリース



pixivのコードは毎日どんどん刷新されてるよ

pixiv

pixiv

ニュースリリース　会社概要　事業紹介　採用情報　お問い合わせ

明日のpixivを作るのは俺たちだ

お絵描きがもっと楽しくなる場所を創る

pixiv 2018年度新卒エントリー受付中

pixiv

プレスリリース

We are hiring!

お絵描きがもっと楽しくなる場所を創る

pixiv

pixiv 2018年度新卒 エントリー受付中

プレスリリース

アジェンダ

pixiv

本日しない話

pixiv

オブジェクト指向

pixiv

~~TDD/BDDの~~

~~教義的なお話~~

pixiv

~~定義に従った~~

~~厳密な用語~~

pixiv

~~Emacs~~

（私はphpunit.elの開発者でもあります）

pixiv

一連のストーリー

などはない

pixiv

オムニバス

pixiv

1. フレームワークなしのテスト
2. できる！ フレームワーク
3. PHPで注意すべき「状態」

pixiv

1. フレームワークなしのテスト
2. できる！ フレームワーク
3. ~~PHPで注意すべき「状態」~~
↑時間なくて収まらなかった

pixiv

本題の前に

pixiv


pixiv

pixivは主に
PHPUnitでテスト

pixiv

僕がテストを新規を増やすならPHPUnitでテスト

pixiv

```
% cd ~/pixiv/tests
% git ls-files '*Test.php' | wc -l
1606
% git ls-files '*Test.php' | xargs cat | wc -l
191890
```

pixiv

どうすれば書きやすく
メンテできる
テストが書けるか悩み

pixiv

本題の前に

pixiv

pixiv

まれによく聞く

pixiv

PHPUnitとかphpspecとか巨大な依存関係入れたくない

pixiv

せやな

pixiv

懸念の妥当性はともかく
依存パッケージの数が
それなりに多いのはたしか

pixiv

「簡単なテストだけ
できればいいよ」

pixiv

おう

pixiv

じゃ、作るか

pixiv

その前に

pixiv

ところでPHP標準のテスティングフレームワークがある

pixiv

Creating new test files

#phpt Test Basics

The first thing you need to know about tests is that we need more!!! Although PHP works just great 99.99% of the time, not having a very comprehensive test suite means that we take more risks every time we add to or modify the PHP implementation. The second thing you need to know is that if you can write PHP you can write tests. Thirdly - we are a friendly and welcoming community, don't be scared about writing to (php-qa@lists.php.net) - we won't bite!

- **So what are phpt tests?**
  A phpt test is a little script used by the php internal and quality assurance teams to test PHP's functionality. It can be used with new releases to make sure they can do all the things that previous releases can, or to help find bugs in current releases. By writing phpt tests you are helping to make PHP more stable.

http://qa.php.net/write-test.php

PHP本体のテストコード

pixiv

php-srcに入ってる
run-tests.phpで
実行できる

pixiv

実はPHPUnitでも
おまけ機能として
テスト実行できる

pixiv

バッファの出力をテストする

```
--TEST--
Compute 1 + 1 test
--FILE--
<?= 1 + 1 ?>
--EXPECT--
2
```

```
--TEST--
GET $_ENV test
--ENV--
return <<<END
a=AAA
b=BBB
END;
--FILE--
<?php
echo getenv('a'), "\n";
echo getenv('b'), "\n";
?>
--EXPECT--
AAA
BBB
```

ただし（意図的なのか）
仕様を完全には
サポートしてません

pixiv

もしphptを採用する
ならrun-tests.php
を使った方がいい

pixiv

テスト

スクリプト

pixiv

なぜ人はテストを書くのか

pixiv

実装前に入出力の例を明示したい

pixiv

機能追加や
リファクタリングで
意図した挙動が壊れな
いことを保障したい

pixiv

テストを書こう

pixiv

そうすると

pixiv

関心がどうとか

って話になってくる

pixiv

うさぴえ

pixiv

やりたいことは
動作確認

pixiv

それだけ

pixiv

テストスクリプトを書こう

pixiv

test.php

```
<?php
require_once __DIR__ . '/vendor/autoload.php';

(2 === 1 + 1) or die("LINE: ". __LINE__);
(3 === 1.5 + 1.5) or die("LINE: ". __LINE__);
$date = date('Y-m-d',4502304000);
("2112-09-13" === $date) or die("LINE: ". __LINE__);

$a = [1, 2, 3];
(1 == array_shift($a)) or die("LINE: ". __LINE__);

echo 'ok.', PHP_EOL;
```

失敗すると行が
出力されて終了

pixiv

or die()

書くのだるい

pixiv

もうちょっと

わかりやすく

pixiv

assert()

pixiv

test.php

```
#!/usr/bin/env php
<?php
require_once __DIR__ . '/vendor/autoload.php';

assert(2 === 1 + 1);
assert(3 === 1.5 + 1.5);
assert("2112-09-13" === date('Y-m-d', 4503168000));

$a = [1, 2, 3];
assert(1 == array_shift($a));
```

出力

```
PHP Warning:  assert(): assert(3 === 1.5 + 1.5) failed in /Users/
megurine/repo/php/phperkaigi-test/assert-test.php on line 11

Warning: assert(): assert(3 === 1.5 + 1.5) failed in /Users/megurine/
repo/php/phperkaigi-test/assert-test.php on line 11
PHP Warning:  assert(): assert('2112-09-03' === date('Y-m-d',
4503168000)) failed in /Users/megurine/repo/php/phperkaigi-test/assert-
test.php on line 12

Warning: assert(): assert('2112-09-03' === date('Y-m-d', 4503168000))
failed in /Users/megurine/repo/php/phperkaigi-test/assert-test.php on
line 12
```

エラーハンドリングの設定をしないと出力が二重化される

pixiv

まあ雑だけど

テストにはなる

pixiv

元

pixiv

pixiv

最低限ができると

人間は欲が出てくる

pixiv

できる！

フレームワーク

pixiv

フレームワークとは

pixiv

ソフトウェアフレームワーク

ソフトウェアフレームワーク（英: software framework）とは、プログラミングにおいて、一般的な機能をもつ共通コードをユーザーが選択的に上書きしたり特化させたりすることで、ある特定の機能をもたせようとする抽象概念のことである。単に**フレームワーク**とも呼ばれる。

ソフトウェアフレームワークは、はっきり定義されたAPIを持ち、具体的な実装を再利用可能な形で隠蔽しているという点でライブラリとよく似ている。しかし、ライブラリでは呼び出し側がプログラム全体の制御構造を指定できないが、フレームワークでは可能である。この制御の反転がソフトウェアフレームワークの特徴である[1]。

https://ja.wikipedia.org/wiki/ソフトウェアフレームワーク

pixiv

“制御の反転がソフトウェアフレームワークの特徴”

pixiv

ふつうのライブラリは
あなたが書いたコードから
呼び出される

pixiv

フレームワークは
あなたが書いたコードを
呼び出す

pixiv

これが

制御の反転

pixiv

ここからは

どんどんいきます

pixiv

仕様1: コードをべたに実行

- テストファイルは xxx_test.phpの形式
- assert()を使ってテストする
- スクリプトを実行すると実行

assert_tests/run

```
#!/usr/bin/env php
<?php
require __DIR__ . '/../vendor/autoload.php';
$testing = include __DIR__ . '/bootstrap.php';
chdir(__DIR__);
foreach (glob('*_test.php') as $file) {
    try {
        include $file;
    } catch (\Throwable $e) {
        dump($e);
    }
}
$testing->finalize();
```

assert()に失敗したときの結果はカスタマイズできる

pixiv

（実はPHP5方式と
PHP7方式で異なる）

今回はPHP5方式

pixiv

bootstrap.php

```
<?php
require_once __DIR__ . '/define.php';
ini_set('zend.assertions', 1);
ini_set('assert.exception', 0);
assert_options(ASSERT_ACTIVE, 1);
assert_options(ASSERT_BAIL, 0);
assert_options(ASSERT_QUIET_EVAL, 0);
assert_options(ASSERT_WARNING, 0);

$testing = new Testing();
assert_options(ASSERT_CALLBACK, [$testing, 'handler']);
return $testing;
```

Testing.php

```php
<?php
class Testing {
    /** @var bool テスト失敗停止するか */
    private $stop_on_failure = false;
    /** @var array[] 実行結果 */
    private $result = [];

    public function __construct(array $options) {
        $this->stop_on_failure = $options['stop_on_failure'] ?? false;
    }

    /** assert()の失敗時にこいつが呼ばれるように設定 */
    public function handler($file, $line, $code, $desc = null) {
        $this->result[] = [$file, $line, $desc];
        if ($this->stop_on_failure) $this->finalize();
    }
```

Testing.php

```
    /** いままで失敗したテストをまとめて結果表示 */
    public function finalize(): void
    {
        if ($this->result) {
            foreach ($this->result as [$file, $line, $desc])
{
                $this->output($file, $line, $desc);
            }
            exit(1); // エラー終了
        }
        $this->result = [];
        echo 'ok.', PHP_EOL;
        exit(0);
    }
```

Testing.php

```
    private function output($file, $line, $desc): void
    {
        // ソースコードの該当行を取得
        $code = trim(file($file)[$line-1]);


        echo "FILE: {$file} ({$line})", PHP_EOL;
        echo "CODE: {$code}", PHP_EOL;
        echo "DESC: {$desc}", PHP_EOL;
    }
```

完成！？

pixiv

テストを記述

pixiv

fizzbuzz_test.php

```php
<?php
// to_fizzbuzz() 関数は別に定義済みの想定

assert("1"    === $actual = to_fizzbuzz(1));
assert("Fizz" === $actual = to_fizzbuzz(3));
assert("Buzz" === $actual = to_fizzbuzz(5));
assert("Buzz" === $actual = to_fizzbuzz(10));
assert("FizzBuzz" === $actual = to_fizzbuzz(15));
```

```
megurine % ./assert-tests/run
 FILE: /Users/megurine/repo/php/phperkaigi-test/assert-tests/fizzbuzz_test.php (7)
 CODE: assert("FizzBuzz" === $actual = to_fizzbuzz(5));

 FILE: /Users/megurine/repo/php/phperkaigi-test/assert-tests/fizzbuzz_test.php (8)
 CODE: assert("FizzBuzz" === $actual = to_fizzbuzz(15));

 FILE: /Users/megurine/repo/php/phperkaigi-test/assert-tests/fizzbuzz_test.php (9)
 CODE: assert("FizzBuzz" === $actual = to_fizzbuzz(45));
```

完璧では

（ただしassertは一行で書かれるものとする）

pixiv

pixiv

さらに

パワーアップ

pixiv

PowerAssert

（のようなもの）

pixiv

Groovyのテストで一躍知られた

pixiv

t-wadaさんの

power-assert.jsが有名

Hamcrest

Fork me on GitHub

Matchers that can be combined to create flexible expressions of intent

Born in Java, Hamcrest now has implementations in a number of languages.

- Java
- Python
- Ruby
- Objective-C
- PHP
- Erlang
- Swift

http://hamcrest.org/

pixiv

PHPでちゃんと
作ろうとすると大変

pixiv

それっぽいもの

なら作れる！

pixiv

```
megurine % ./assert-tests/run
FILE: /Users/megurine/repo/php/phperkaigi-test/assert-tests/fizzbuzz_test.php (15)
CODE: assert("FizzBuzz" === $actual = to_fizzbuzz(5), eval(collect_vars));
                           └  "Buzz"

FILE: /Users/megurine/repo/php/phperkaigi-test/assert-tests/plus_test.php (10)
CODE: assert($expected === $actual = $a + $b, eval(collect_vars));
              |              |          |    └ 1.5
              |              |          └ 1.5
              |              └ 3.0
              └ 3
```

pixiv

欠点

pixiv

assert()では失敗しかフックできないから、「98/100成功」のようなカウントができない

pixiv

PowerAssert(もどき)

書くのがめんどい

pixiv

```
= to_fizzbuzz(1), eval(collect_vars));
= to_fizzbuzz(3), eval(collect_vars));
= to_fizzbuzz(5), eval(collect_vars));
= to_fizzbuzz(10), eval(collect_vars));
ual = to_fizzbuzz(5), eval(collect_vars));
```

pixiv

define.php

```php
<?php

const collect_vars = 'return v(get_defined_vars()+
["this"=>isset($this)?$this:null]);';

function v(array $variables): string
{
    // $GLOBALSは長大なので消しとく
    unset($variables['GLOBALS']);
    return \serialize($variables);
}
```

次のパターンに
行きましょう

pixiv

仕様2: 関数を実行

- テストファイルは xxx_test.phpの形式
- その中に test_xxxx() 関数が定義する
- Hamcrestを使ってテストしてみる
- スクリプトを実行すると実行

Hamcrest

Fork me on GitHub

Matchers that can be combined to create flexible expressions of intent

Born in Java, Hamcrest now has implementations in a number of languages.

- Java
- Python
- Ruby
- Objective-C
- PHP
- Erlang
- Swift

http://hamcrest.org/

pixiv

Hamcrest (hamcrest-php)

- マッチャー（値がマッチするか比較）
- もともとはJavaのライブラリ
- assertXXX()みたいなのが定義されてる
- \Hamcrest\Util::registerGlobalFunctions()
  を呼ぶとグローバル関数として定義される

assert_tests/run (1/3)

```
#!/usr/bin/env php
<?php
require __DIR__ . '/../vendor/autoload.php';

\Hamcrest\Util::registerGlobalFunctions();

chdir(__DIR__);
foreach (glob('*_test.php') as $file) {
    require_once $file;
}

$failures = [];
$errors   = [];
```

assert_tests/run (2/3)

```
foreach (get_defined_functions(true)['user'] as $func) {
    if (strpos($func, 'test_') === 0) {
        try {
            $func();
            echo '.';
        } catch (\Hamcrest\AssertionError $e) {
            $failures[] = $e;
            echo 'F';
        }catch (\Throwable $e) {
            $errors[] = $e;
            echo 'E';
        }
    }
}
echo PHP_EOL;
```

assert_tests/run (3/3)

```
if ($failures) {
    echo "Failures:", PHP_EOL;

    foreach ($failures as $f) {
        echo PHP_EOL;
        echo "FILE: {$f->getFile()} ({$f->getLine()})", PHP_EOL;
        echo "DESC: {$f->getMessage()})", PHP_EOL;
    }
}

if ($errors) {
    echo "Errors:", PHP_EOL;
    dump($errors);
}
```

また完成して
しまった…

pixiv

テストを記述

pixiv

fizzbuzz_test.php

```
<?php


function test_fizzbuzz()
{
    assertThat(to_fizzbuzz(1), equalTo("1"));
    assertThat(to_fizzbuzz(3), equalTo("Fizz"));
    assertThat(to_fizzbuzz(5), equalTo("Buzz"));
    assertThat(to_fizzbuzz(10), equalTo("Buzz"));
    assertThat(to_fizzbuzz(15), equalTo("FizzBuzz"));
}
```

```
megurine % ./hamcrest-tests/run
F.
Failures:

FILE: /Users/megurine/repo/php/php
DESC: Expected: "FizzBuzz"
     but: was "Buzz")
```

pixiv

pixiv

出力差分はわかるけど

いまいちわかりにくい

pixiv

そうだ
パラメタライズテスト
をしよう

pixiv

引数だけを変更して同じテストをする

pixiv

PHPUnitでの @dataProvider

pixiv

名前も拝借して

いきましょう

pixiv

fizzbuzz_test.php (Before)

```
<?php


function test_fizzbuzz()
{
    assertThat(to_fizzbuzz(1), equalTo("1"));
    assertThat(to_fizzbuzz(3), equalTo("Fizz"));
    assertThat(to_fizzbuzz(5), equalTo("Buzz"));
    assertThat(to_fizzbuzz(10), equalTo("Buzz"));
    assertThat(to_fizzbuzz(15), equalTo("FizzBuzz"));
}
```

fizzbuzz_test.php (After)

```php
<?php
/**
 * @dataProvider fizzbuzz_number_provider
 */
function test_fizzbuzz2($expected, $input)
{
    assertThat(to_fizzbuzz($input), equalTo($expected));
}

function fizzbuzz_number_provider()
{
    return [
        ["1",         1],
        ["Fizz",      3],
        ["Buzz",      5],
        ["Buzz",     10],
        ["FizzBuzz", 15],
    ];
}
```

実行側も
書き換えていきます

pixiv

assert_tests/run (2/3)

```
foreach (get_defined_functions(true)['user'] as $func) {
    if (strpos($func, 'test_') === 0) {
        try {
            $func();
            echo '.';
        } catch (\Hamcrest\AssertionError $e) {
            $failures[] = $e;
            echo 'F';
        }catch (\Throwable $e) {
            $errors[] = $e;
            echo 'E';
        }
    }
}
echo PHP_EOL;
```

assert_tests/run After

```
foreach (get_defined_functions(true)['user'] as $func) {
    if (strpos($func, 'test_') === 0) {
        $provider = get_provider($func) ?: function (){ yield []; };

        foreach ($provider() as $args) {
            try {
                $func(...$args);
                echo '.';
            } catch (\Hamcrest\AssertionError $e) {
                $failures[] = ['error' => $e, 'args' => $args];
                echo 'F';
            }catch (\Throwable $e) {
                $errors[] = $e;
                echo 'E';
            }
```

assert_tests/run getProvider

```
function get_provider(string $func_name): ?string
{
    $ref = new \ReflectionFunction($func_name);

    // @dataProvider hogehoge を取り出す
    if (preg_match('/@dataProvider +(?<provider>\S+)/',
$ref->getDocComment(), $m)) {
        return trim($m['provider']);
    }

    return null;
}
```

```
megurine % ./hamcrest-tests/run
.....FF.
Failures:

FILE: /Users/megurine/repo/php/p
ARGS: array:2 [
  0 => "FizzBuzz"
  1 => 15
]
DESC: Expected: "FizzBuzz"
     but: was "Buzz")

FILE: /Users/megurine/repo/php/p
ARGS: []
DESC: Expected: "FizzBuzz"
     but: was "Buzz")
```

pixiv

まとめると

pixiv

単にテストを実行する
フレームワークっぽいもの
は意外と簡単に作れる

pixiv

PHPでテスト
しにくい状態

pixiv

ここまでは「理想的な」
関数をテストすること
しか想定してない

pixiv

理想的
=
引数と返り値だけ

pixiv

PHPで書かれる「現実」のコードはそれだけじゃない

pixiv

ファイル書き込み、データベース、
外部HTTP API呼び出し、グローバル変数、メール送信、HTTPヘッダ、クッキー…… などなど

pixiv

例: CLIでcookie()とかheader()が呼び出されるとFatal errorになる

pixiv

これはテスティングフレームワークだけではなくアプリケーションの構造も再設計しないとうまくテストできない

pixiv

HTTPはPSR-7に

するとか

pixiv

外部への依存を
抽象化して
注入するとか

pixiv

テスト実行中に
専用のDBを
起動して利用するとか

pixiv

テストしたいと思った
瞬間に直さなきゃ
いけないことが
爆発的に発生して氏ぬ

pixiv

次回に続く…？

pixiv

まじめにテスト

やりたいひとへ

pixiv

おまけ

pixiv

テスト駆動開発 単行本（ソフトカバー） – 2017/10/14

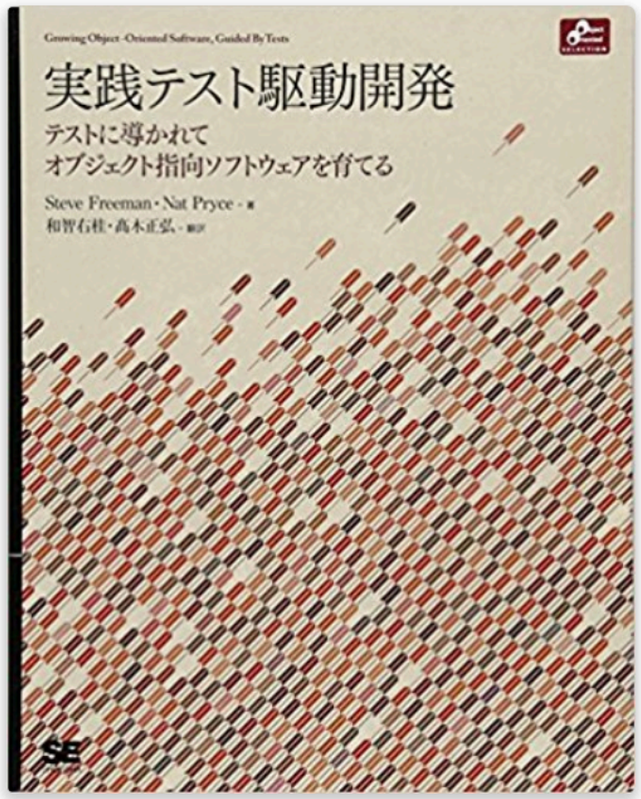
Kent Beck (著), 和田 卓人 (翻訳)

★★★★½ ▾ 2件のカスタマーレビュー

▸ その他（2）の形式およびエディションを表示する

単行本（ソフトカバー）
￥ 3,024 prime

￥ 2,573 より 14 中古品の出品
￥ 3,024 より 11 新品



実践テスト駆動開発 (Object Oriented SELECTION) 大型本 – 2012/9/14

Steve Freeman (著), Nat Pryce (著), & 2 その他

★★★★☆ ▾ 3件のカスタマーレビュー

▸ その他（）の形式およびエディションを表示する

大型本
￥ 4,536 ✓prime

￥ 2,582 より 13 中古品の出品
￥ 4,536 より 1 新品

PHPUnitも

良いが

pixiv

phpspecを

すごいぞ

pixiv

iakioの日記

2015-06-14

OSC2015北海道で「phpspecで学ぶLondon School TDD」という発表をしてきました

PHP phpspec tdd

見た目を重視してこんなタイトルにしてみましたが、基本的には私が「実践テスト駆動開発」と「phpspec」をどう解釈したか、というような内容になっています。

45分あってもなかなか伝えるのが難しいテーマだったのですが、とりあえず時間厳守はできてよかったです（一部デモを飛ばしてしまいましたが）。

phpspecで学ぶLondon School TDD from Akio Ishida

www.slideshare.net

プロフィール

id:iakio PRO

読者になる 25

2014-03-07

知らないうちにphpspecがすごいことになっていた件

PHP bdd phpspec

昔あったphpspecが進化したものなのかそれとも別のものなのかは知りませんが。

すごい。軽くキモい。17分の動画です。

- Laracasts | PHPSpec is So Good

まず、いきなり「MovieCollectionにMovieをadd()するとcountが1になること」というのスペックを書きます。MovieCollectionクラスもMovieクラスもまだ作ってません。

```
namespace spec;

use Movie;
use PhpSpec\ObjectBehavior;
use Prophecy\Argument;

class MovieCollectionSpec extends ObjectBehavior
{
    function it_is_initializable()
    {
        $this->shouldHaveType('MovieCollection');
    }

    function it_stores_a_collection_of_movies(Movie $movie)
    {
        $this->add($movie);

        $this->shouldHaveCount(1);
    }
}
```

`phpspec run` を実行すると当然失敗して、「MovieCollectionクラスが無いけど作ってほしい？」と尋ねてきます。`Y` なら空のMovieCollectionクラスが作られます。

プロフィール

id:iakio PRO

読者になる 25

B 新着エントリー

学習のためにGithubを徘徊する - iakioの日記
4users

OSC2015北海道で「phpspecで学ぶLondon School TDD」という発表をし...
9users

プログラミング初心者が中・上級者になるためには、Github...
160users

#mozaicfm #7 REST を聞いた - iakioの日記
3users

MarkdownエディタStackEditのベータ版はシーケンス図やフローチ...
164users

Hatena::Bookmark

B 人気エントリー

MarkdownエディタStackEditのベータ版はシーケンス図やフローチ...
164users

プログラミング初心者が中・上級者になるためには、Github...
160users