公開版

staticおじさんになろう

War on Dynamics

2017-07-15 PHPカンファレンス関西2017 LT

このスライドは2017年7月15日に開催されたPHPカンファレンス関西2017で当日募集された本篇LTにて話したスライドに調整および加筆を施したものです。口頭で補足したもの、文字だけではわかりにくいものを補ったほか、後半部分に文章の追加があります。

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id:zonuexe
- ピクシブ株式会社でひたすらPHPやってる p(ixi)v
- フレームワークなき道にフレームを作るおしごと
- Qiitaに記事を書いたり変なコメントしてるよ
- 最近やってること
  - 万難が排されつつあるのでPHP7 移行
  - えいやえいやとテスト書いたり直したりしてる

pixiv

そろそろ10年になる
Webサービスを開発

注意

このLTは特定のアーキテクチャの導入を勧誘するものではなく、その是非については自分で判断してください。

対処が難しかった事例を
寄せ集めただけで、
特定の実装の話ではない

オチは特にない

あと特定の会社の

回し者ではない

あるメソッドの
API(引数・返り値)を
変更したいとします

リファクタリング
できますか？

そのメソッドの
呼び出し箇所を列挙
できますか？

そうですね

```
git grep '\->setName('
```

本当にそれで
全部ですか？

```php
$obj = new FugaClass;
$f = [$obj, 'setName'];
call_user_func($f, $n);
```

```
$obj = new FugaClass;
$obj->setName($v);
// ↑探したいのはこっち
```

```
$obj2 = new HogeClass;
$obj->setName($x, $y);
// ↑同名の関係ないメソッド
```

誤爆

結果を絞ろう

```
# newの次行にsetNameがある前提
git grep -a1 'new FugaClass' |
    grep '\->setName('
```

これで完璧

// DIコンテナ

```
$app['fuga']->setName($n);
```

こんなことが続くと人間は無気力になる（学習性無力感）

（そのあとも動的な性質
による悩みは絶えない）

ソリューション

new staticおじさんedition

```
class FugaClass {
    private $name;
    private function __construct(){}
    public static function setName(FugaClass $that, $name) {
        $that->name = $name;
    }
    // ...
```

```
git grep
'FugaClass::setName'
```

これでいいのか

書きにくくなった

だけでは…

staticおじさんをやめよう

Beat “evil” static methods

2017-07-15 PHPカンファレンス関西2017 LT

特効薬

~~Photoshop~~

PhpStorm

FindUsageと Refactor機能最強説

型

型が推測できなくなるような複雑な書き方をしない

型を確定できないと
ころは地道にPHPDoc
で型をつけていく

http://nico.ms/kn1774

はじめてのPHPDocと型

My first PHPDoc and Type

pixiv

2016-09-09 ピクシブ社内勉強会

これだけ覚えて帰ってね

タグ名	意味	例
@param	引数を定義	@param int $n1
@return	返り値を定義	@return int[]
@var	変数/プロパティを定義	@var int
@property	マジックプロパティを定義	@property int $id

DI/IoC

コンテナは？

（日本語での説明、ただしサンプルコードが旧形式）
PhpStorm の PHPSTORM_META でサービスロケーターとかを入力補完 - ngyukiの日記

http://ngyuki.hatenablog.com/entry/2015/02/20/134239

（公式資料・新形式のコード例が載ってる）
PhpStorm Advanced Metadata

https://confluence.jetbrains.com/display/PhpStorm/PhpStorm+Advanced+Metadata#PhpStormAdvancedMetadata-Factorymethods

某社の回し者ではないので、PhpStorm以外に良いツールあれば知りたい！

そもそも
staticおじさんになっ
た理由はコレジャナイ

なれる！staticおじさん

I wanna be a clever static uncle

pixiv

2017-07-15 PHPカンファレンス関西2017 LT

Webサービスを作る人間

ー

オブジェクト指向の達人

ではない

素朴に作られたクラス
は余計な状態を抱えこ
んで渾沌が肥大化する

多すぎる引数…
謎のメソッド…
不要なプロパティ…

それでもコードは動き、
今の自分たちを生かし
てくれてる（=収益源）

作られた当時は
理由があった
（たぶん）

動いてきたコードへの
リスペクトを
見失ってはならない

それでもサービスを成長さ
せる足枷にならないように
リファクタリングしていく

混沌としたクラスから
staticおじさんに
身を堕としたのは
カオスを収束させるため

カオスを収束させた後なら
ば、未来に向かって
「あるべき姿」を再定義
することができる

カオスを収束させずに夢を見てアーキテクチャの再設計を試みると理想と現実のギャップと物量に圧し潰されて氏ぬ

重要な事実：
オブジェクト指向を多少かじった程度では、ベターなOO設計ができる大先生にはなれない

設計のエキスパートではなくても多くのパタンに触れていけば現在の身のたけに合ったものを選びとることはできる

オブジェクト指向プログラミングに
基いたリファクタリングが常に
最適解ではなくて、手続き型なら
手続き型としてのやりかたはある。
（staticおじさんの再肯定）

設計に失敗したらやり直しが利くように疎結合にしていくことも大事。

（疎結合にする試みそのものが
負債を生むこと往々にしてある。
一足飛ばしで変更しすぎるのもよ
くない）

現実を直視して分析し、
自分たちに合った現実解
を地道に適用していく

どんなことにも長所と短所がある。
世の中ではオブジェクト指向が先
進的、手続き型はレガシーのよう
な風潮もあるが、手続き型がベター
と結論することもある。

しかし「オブジェクト指向はわかりにくい」「インスタンスを作るとりファクタリングしにくい」といった風評はPhpStormなどのIDEで解決できる。ツールを知ることでサービスの成長改善につなげられる。

良いツールを知らないと「リファクタリング対象をリストアップするときにgrepコマンドでテキスト検索するしかない」といった事態に陥る。
（最初の問題提起）

スタートアップ期のコードでは設計に悩み留まって製品をリリースできないことは賢明ではない。リリースされた後も機能追加に注力すべきだが、機能追加と既存機能の安定性を両立しようとすると、どこかで決断の時は来る。

きちんと設計して
未来に行こう