Phanによる PHPコード静的解析

公開補訂版

ピクシブ株式会社
うさみけんた @tadsan

pixiv

JAPAN phpcon 2016, 3rd November

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id:zonuexe
- 2012年 自宅警備→ピクシブ株式会社
  - モバイルアプリ向けWebAPI開発とか
  - 今年の春からメンテナンスとしてチーム独立
  - 日常的なバグ修正から、アーキテクチャの改善
    リファクタリングやテストなどいろいろ！

pixiv

おしごと

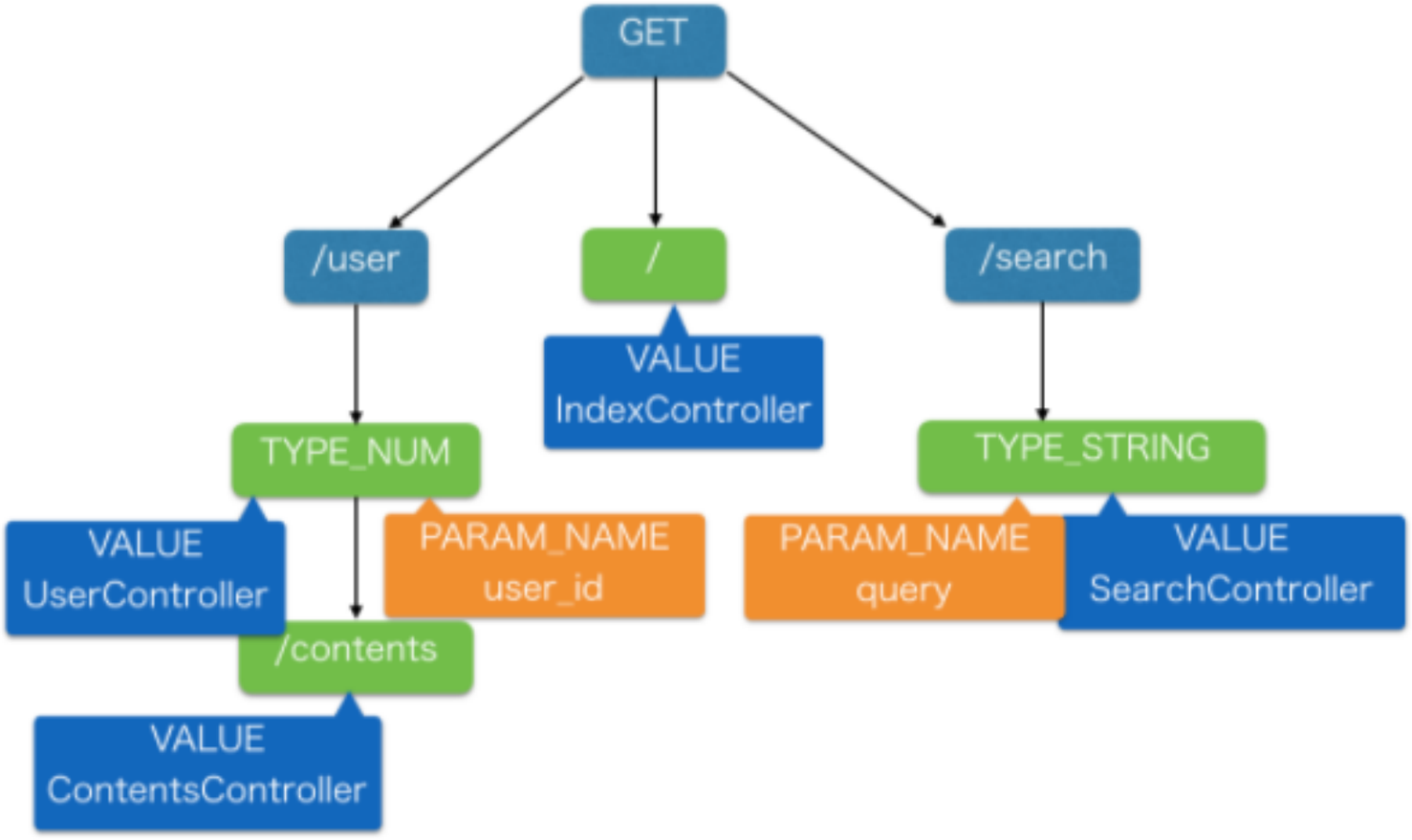
pixiv

現実に動いている
（ユーザーさんが利用している）
サービスを破壊せずに
コードの健全さを向上したい

pixiv

2015-12-13

PHPで高速に動作するURLルーティングを自作してみた



そんなノウハウをギュッと詰め込んだ

PHP大規模開発入門(連載中)

PHP大規模開発入門

- Vol.87: PHPDocでコードの品質を保つ
- Vol.91: 名前空間とオートローディング
- Vol.94: PHP初心者がハマりがちな落とし穴
  ……型のキャスト，変数とリファレンス
- Vol.95: PHPの静的解析
  ……ドキュメントの生成，問題箇所の発見

pixiv

PHP大規模開発入門

- Vol.87: PHPDocでコードの品質を保つ
- Vol.91: 名前空間とオートローディング
- Vol.94: PHP初心者がハマりがちな落とし穴

  ……型のキャスト，変数とリファレンス
- Vol.95: PHPの静的解析

  ……ドキュメントの生成，問題箇所の発見

pixiv

おしながき

- PHPと型
- PHPDocとは何か
- Phanの紹介

pixiv

PHPと型

pixiv

動的型検査

- プログラムを動かしながら(←動的)
  手さぐりで型を調べる（←型検査）

```
$a = 13;  $b = "2";
echo $a + $b;
```

- 「$aの型はintだな、$bの型はstring…
  だけど数字だから足しざんできるね！」
  ……みたいな茶番を毎回やってる

pixiv

PHPの型

- 変数に型はないが、値が型を持つ
  - C言語みたいに `int id` のような定義はできない
  - `64` と `'A'` は厳然とした別の型の値
  - `'64' == 64` のような比較が成り立つ設計
- 関数は型宣言できる（ただし動的検査）

pixiv

PHPの型の種類

- 整数(int) 浮動小数点数(float) 論理値(bool)
  文字列(string) 配列(array) リソース(resource)
  オブジェクト(object) ヌル値(null)

- PHP5ではarrayのみ宣言に記述できる

- 7.0ではint, float, string, boolが記述できる

pixiv

型宣言(PHP7)

PHP7の関数(メソッド・クロージャ含む)は、
引数と返り値に型を定義することができる

```
// PHP 5
function trapezoid($h, $a, $b) {
    return $h * ($a + $b) / 2;
}
```

```
// PHP 7
function trapezoid(float $h, float $a, float $b): float
{
    return $h * ($a + $b) / 2;
}
```

動的型検査の弱点

- プログラムを動かしてみないとわからない

```
$a = 13;  $b = [];
// 環境依存の分岐（本番環境じゃないと動かない）
if (is_production()) {
  echo $a + $b; // 検出できない
  echo 1 + [];  // ←これはPHP7で検出できるようになった
}
function f(): int {
    return []; // これも検出できない
}
```

pixiv

PHPDoc#とは何か

pixiv

DocCommentとリフレクション

- 定義文(クラス・関数・メソッドなど)に付属する
  特別なコメント(**/**** ～ ***/** の範囲)を実行時に
  文字列として取得することができる(DocComment)

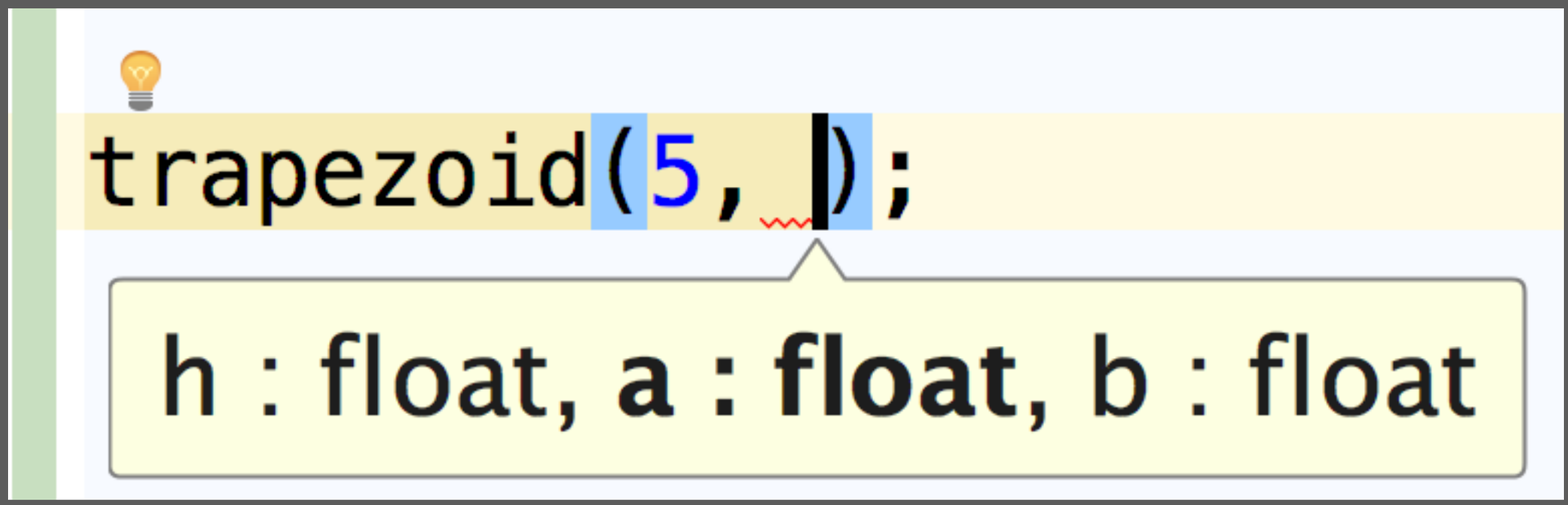
```
/** ここはDocCommentだよ */
function hoge() {
}

$ref = new \ReflectionFunction('hoge');
echo $ref->getDocComment();
// => "/** ここはDocCommentだよ */"
```

PHPDoc型注釈

DocCommentに型定義を注釈として記述する記法。
PhpStormやPhanが解釈してくれる。

```
/**
 * @param  float $h
 * @param  float $a
 * @param  float $b
 * @return float
 */
function trapezoid($h, $a, $b) {
    return $h * ($a + $b) / 2;
}
```

PhpStormの型表示



PHPDocの型

- 基本はPHPの型かクラス名を書く
  - `callable`, `$this`, `self`, `static`, `void`なども有効
- 複合型（Union/Multiple types）
  - 「intまたは文字列」 `int|string`
- 値が並んでるもの（配列/Collection）
  - 「intが並んでるもの」 `int[]`
- https://zonuexe.github.io/phpDocumentor2-ja/references/phpdoc/types.html

なぜPHPDoc型注釈？

- いきなり実装コードに型定義を追加すると、
  いままで顕在化しなかったバグが発生するおそれ
  - 不用意なキャストや型検査でエラー発生
  - PHPDocはただのコメントなので、
    稼動中のコードの挙動を変更せず検査できる
- 現行のPHPの型定義以上の型付けを表現できる
  - 複合型やコレクション(`array → int[]`)
  - PHP5で対応できない `string, int` など

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`

ふつうのプロパティは`@property`じゃなくて
`@var`なので注意（Phanは`@property`未対応）

pixiv

プロパティの例（@var）

ふつうのプロパティの場合は @var を使って書く

```
final class Book {
    /** @var string */
    public $title;
    /** @var \MyApp\Author[] */
    public $authors;
    /** @var \MyApp\ISBN */
    public $isbn;
}
```

pixiv

プロパティの例（@property）

拙作のzonuexe/objectsystemを使った場合

```
/**
 * @property string           $title
 * @property \MyApp\Author[] $authors
 * @property \MyApp\ISBN     $isbn
 */
final class Book {
    use \Teto\Object\TypedProperty;
    protected static $property_types = [
        'title'   => 'string',
        'authors' => 'MyApp\Author[]',
        'isbn'    => '?MyApp\ISBN',
    ];
}
```

型注釈を使ったハック

複合型とコレクション(型の配列)を組み合せる

```php
/** @return Book[]|\ArrayObject */
function getBooks() {
    $data = hogehoge();
    return new \ArrayObject($data);
}

$books = getBooks();
$books-> // ここで \ArrayObject の補完が効く

foreach ($books as $book) {
    $book-> // ここで Book の補完が効く
}
```

Phanの紹介

pixiv

Phanとは何か

- ハンドメイドマーケットを運営する
  アメリカのEtsy社が開発する静的解析ツール
  https://github.com/etsy/phan
  - 現在PHP作者のRasmus Lerdorf が所属し、
    Phanの開発にも参加してる
- PHPの型定義と型注釈を両方見てくれる
  - ただし、@propertyと@method未対応

pixiv

Phanの導入

- https://github.com/etsy/phan/releases
  最新のタグが付いてる**.phar**ファイルをダウンロード

- PHP7とphp-astが必須

- ただしラッパースクリプトを用意すれば、
  サービスがPHP7で稼動してなくても動かせる

```
#!/bin/sh
/path/to/php7/bin/php ${PHAN_BIN:-/path/to/bin/phan}  "$@"
```

pixiv

Phanの設定

- はじめにWikiから設定ファイルをコピペして、プロジェクトの `.phan/config.php` に保存
  https://github.com/etsy/phan/wiki/Getting-Started#creating-a-config-file

```
    // ディレクトリホワイトリスト
    'directory_list' => [
        'src',
        'vendor/symfony/console',
    ],
    // ディレクトリブラックリスト
    "exclude_analysis_directory_list" => [
        'vendor/'
    ],
```

pixiv

Phanの設定(そのほか)

- 'generic_types_enabled' => bool
  - @templateタグ(ジェネリック)有効/無効
- 'suppress_issue_types' => string[]
  - 無視するIssueタイプを指定する
- 'plugins' => string[]
  - Phanプラグインを追加できる
  - サンプルとしてDollarDollarが添付されている
    - $$var (可変変数)を検出

pixiv

Phan コマンド

- -o, --output <filename> 出力ファイル指定
- -m <mode>, --output-mode 出力形式
- -x, --dead-code-detection デッドコード検出
- --backward-compatibility-checks PHP7互換性
- -j, --processes <int> 並列実行数
- -p, --progress-bar 進捗バーを表示

```
./phan-wrapper -m csv -o phan.csv -x -j4 \
    --progress-bar --backward-compatibility-checks
```

Phan 出力

- デフォルトの場合の出力

```
pixiv-lib/Illust/Common.php:738 PhanTypeMismatchArgumentInternal
Argument 1 (month) is string but \checkdate() takes int
pixiv-lib/Illust/Common.php:738 PhanTypeMismatchArgumentInternal
Argument 2 (day) is string but \checkdate() takes int

path/to/file.php:333 PhanTypeMismatchArgumentInternal This is message.
```

- PhanTypeMismatchArgumentInternal はIssueと呼ばれる
- Issueごとに出力するを制御可能(@suppressタグ)

pixiv

Phan Issueいろいろ

- PhanParamTooFew / PhanParamTooMany
  - 引数が足りない／多い
- PhanParamTypeMismatch
  - 引数の型が間違ってる
- PhanRedefineClass / PhanRedefineFunction
  - 同名のクラス／関数がいくつも定義されてる

pixiv

PhanUndeclaredClassCatch

存在しないクラスでキャッチしようとしてないか

```php
// PHP 5
namespace MyApp;

try {
    foo();
} catch (RuntimeException $e) {
    // ↑ \MyApp\RuntimeException
} catch (\Excaption $e) {
    // ↑ \Exception が正しい
}
```

PhanUndeclaredClassInstanceof

存在しないクラスでinstanceofしてないか

```
namespace MyApp;

$e = get_error();
if ($e instanceof RuntimeException) {
    // ↑ \MyApp\RuntimeException
} elseif ($e instanceof \Excaption) {
    // ↑ \Exception が正しい
}
```

pixiv

PhanTypeMismatchForeach

foreachできない値をforeachしようとしてる

```
$iter = null;
// E_WARNING
foreach ($iter as $i) {
    echo $i;
}
```

pixiv

PhanTypeMismatchReturn

定義と違った型の値を返してる

```
/** @return void */
function f() {
    return null;
}

function g():int {
    return [];
}
```

pixiv

PhanTypeMissingReturn

特定の型を返すべきだが、何も返していない

```
/** @return int */
function g(){
    return;
}
```

pixiv

PhanTypeVoidAssignment

実際にはnullが代入されるが
voidは「データがない」を意味するので適切ではない

```
/** @return void */
function g(){ return; }

$result = g();
```

pixiv

PhanNoopXXXX

無意味なコードを検出する

```
// 代入しない配列 PhanNoopArray
[1, 2, 3];

// 代入も実行もしないクロージャ PhanNoopClosure
function(){ return awesome(); };

// returnしていないので無意味 PhanNoopProperty
class C {
    public $p;
    function f() { $this->p; }
}
```

PhanDeprecatedFunction

廃止予定/非推奨のメソッドを利用。
古い実装を新しいものに置換する際に

`@deprecated` を付けておくと洗い出せてべんり

```
function newAwesomeFunc(){}

/** @deprecated */
function oldFunc(){}

oldFunc(); // PhpStormは取り消し線で表示
```

Phan独自のアノテーション

`@suppress <issue_type>`
クラス・メソッドなどの範囲でIssueを無効化する

```
/** @suppress PhanUndeclaredConstant */
class C {
    function f() { return AWESOME_CONSTANT; }
    /** @supress PhanUndeclaredProperty */
    function g() { return $this->p; }
}
```

pixiv

Phan独自のアノテーション

@template 型変数(ジェネリック)

```
/** @template T1 */
class Container {
    /** @var T1 */
    private $value;
    public function __construct($value){ $this->value =
$value; }
    function getValue() { return $value; }
}
```

pixiv

Phanの制限

- define()関数での定数定義はサポートされない
  - 動的に定義されるため（constは静的）
    - Reference to undeclared constant 扱い
- @method には対応してない
  - __call() を使ったProxyパターン殺し

Phanのめんどくさいところ

- 重い
  - ファイル数にもよるが数十秒〜 かかる
- 標準化されてないPHPDocに惑わされる
  - PhanTypeMismatchReturn Returning type bool but insert() is declared to return \成功
  - 「@return 成功 true 失敗 false」

Phanの注意点

- メソッド／プロパティの動的定義に未対応
  - __get(), __set(), __call(), __callStatic() など
    - PHP用語としての「オーバーロード」
      http://php.net/manual/ja/language.oop5.overloading.php
- 定数の動的定義にも未対応
  - define() はだめ。const文ならOK。

Phanの比較対象(競合)

- 定番：PHP Mess Detector (PHPMD)
  - さまざまな指摘をしてくれるが、型検査はなし
- 似たサービスもいくつかある（private有料）
  - Scrutinizer, SensioLabsInsight, CodeClimate
  - これらコードの怪しい兆候を細かに教えてくれる
  - 型検査の精度ではPhanの方が良く見える

まとめ

- PHPDocの型注釈は動作に影響を及ぼさずに
  型を宣言することができる

- Phanはソースコードを動作させずに解析できる

- Phanで静的解析することで、
  リファクタリングのリスクを減らすことができる

- 廃止予定の非推奨メソッドには
  `@deprecated` をつける習慣をつけると殲滅が捗る