PHPStan拡張クイックマスター

Fast-learning PHPStan extension

pixiv Inc.
USAMI Kenta

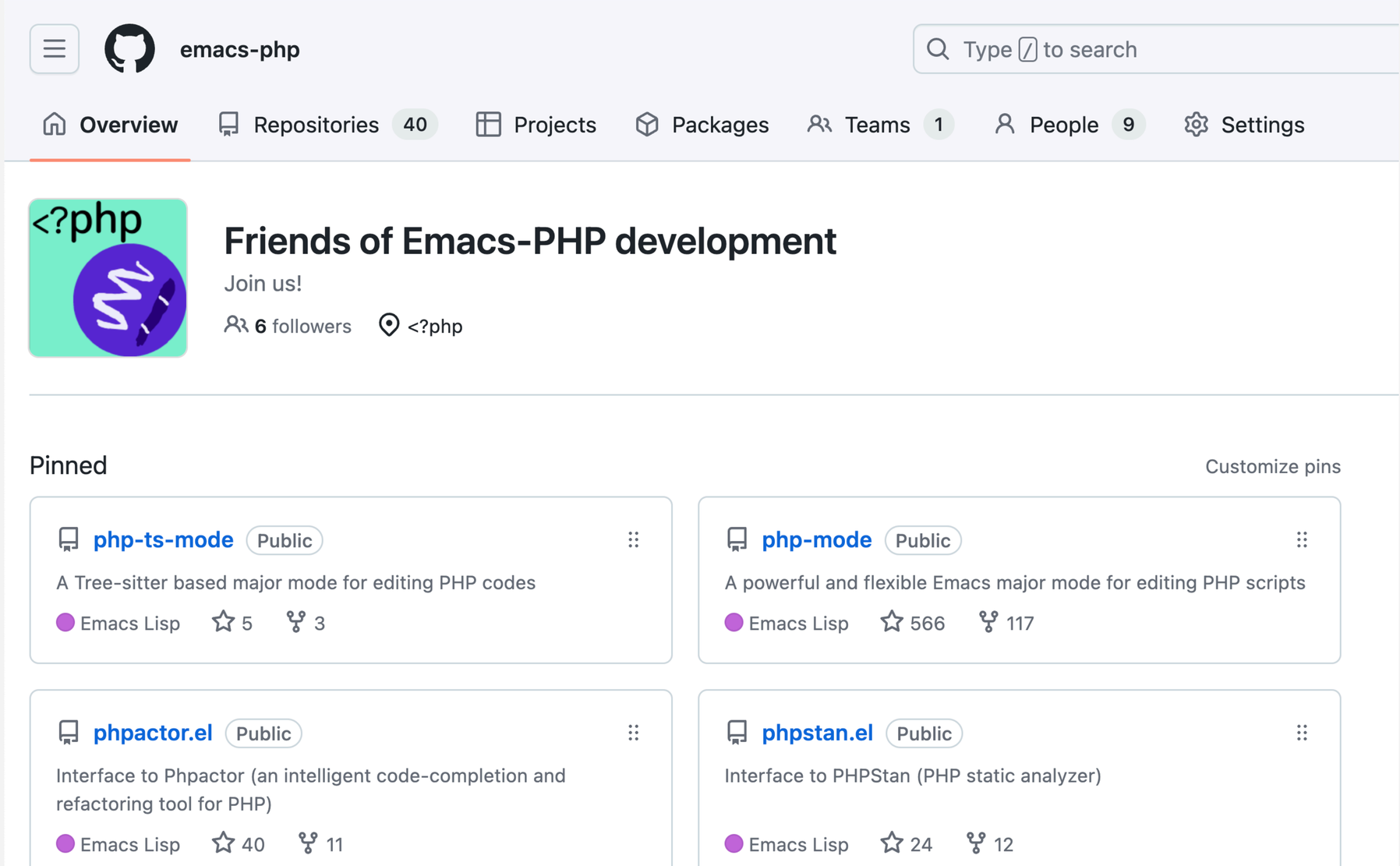
pixiv

2023-08-04 Lint Night #2
#dena_lint_night

お前誰よ

- うさみけんた（@tadsan）/ **Zonu.EXE** / **にゃんだーすわん**
- ピクシブ株式会社 pixiv事業本部 Webエンジニアリングチーム PHPer
  - 2012年末から現職、APIとかCIとかいろいろなところを見つめてきました
  - 最近はピクシブ百科事典(dic.pixiv.net)も開発しています
- Emacs PHP Modeを開発しています（2017年-）
- プログラミング言語にちょっとこだわりのある素人（spcamp2010）

emacs-php



pixiv

PHPカンファレンス沖縄2023

pixiv

LLイベント

Learn Languages

Learn Languages 2023

About

Learn Languagesは様々なプログラミング言語について学びたい技術者のためのイベントです。2003年からほぼ毎年開催しています。

(Open Developers Conference (ODC) 2023 のトラックとして開催します)

URL	https://ll.jus.or.jp/2023/
ハッシュタグ	#ll2023jp

Session

Perl, PHP, Python, Rubyの20年とこれから

2003年に本イベントの第1回で扱ったプログラミング言語はPerl, PHP, Python, Rubyの4つでした。

今回は再びこれら4つの言語を取り上げ、今までの20年を振り返りこれからの展望を語っていただきたいと思います。

タイムテーブル:

13:00 - 13:20 Perl
13:25 - 13:45 PHP
14:00 - 14:20 Python
14:25 - 14:45 Ruby
15:00 - 15:45 この20年を振り返る座談会

History

このイベントは2003年に*LL Saturday*として開始しました。LLはLightweight Languageの略で、*LL Saturday*ではスクリプト言語であるPerl, PHP, Python, Rubyについて取り上げました。

2017年には様々な言語を学ぼうという主旨で、**Learn Languages**という名称にリニューアルしました。2020年は新型コロナウイルス感染拡大のため開催を断念しました。

今までのアーカイブは以下を参照してください。

Learn Languages 2022
Learn Languages 2021
Learn Languages 2019

pixiv

LLイベント

Learn Languages

Learn Languages 2023

About

Learn Languagesは様々なプログラミング言語について学びたい技術者のためのイベントです。2003年からほぼ毎年開催しています。

(Open Developers Conference (ODC) 2023 のトラックとして開催します)

URL	https://ll.jus.or.jp/2023/
ハッシュタグ	#ll2023jp

Session

Perl, PHP, Python, Rubyの20年とこれから

2003年に本イベントの第1回で扱ったプログラミング言語はPerl, PHP, Python, Rubyの4つでした。

今回は再びこれら4つの言語を取り上げ、今までの20年を振り返りこれからの展望を語っていただきたいと思います。

タイムテーブル:

13:00 - 13:20 Perl
13:25 - 13:45 PHP
14:00 - 14:20 Python
14:25 - 14:45 Ruby
15:00 - 15:45 この20年を振り返る座談会

History

このイベントは2003年に*LL Saturday*として開始しました。LLはLightweight Languageの略で、*LL Saturday*ではスクリプト言語であるPerl, PHP, Python, Rubyについて取り上げました。

2017年には様々な言語を学ぼうという主旨で、**Learn Languages**という名称にリニューアルしました。2020年は新型コロナウイルス感染拡大のため開催を断念しました。

今までのアーカイブは以下を参照してください。

Learn Languages 2022
Learn Languages 2021
Learn Languages 2019

pixiv

LLイベント

pixiv

PHPって
どんな言語？

PHPの型にどのような印象を持っていますか？

ＰＨＰの原作者

ラスマスかく語りき

pixiv

“We have things like protected properties. We have abstract methods. We have all this stuff that your computer science teacher told you you should be using. I don't care about this crap at all.”

–Rasmus Lerdorf

https://en.wikiquote.org/wiki/Rasmus_Lerdorf

“PHPにはprotectedプロパティも
抽象メソッドもありますよ。計算機科学の
教授が「使え」と言ってるものは全部。
そんなことは~~クソ~~興味ないですけど”

–Rasmus Lerdorf

脆弱性 ゆるふわ 弱い型 動的
クソザコ 型なし 意味不明 弱い
自動変換 貧弱 Perlっぽい 適当

PHPについてのパブリックイメージ

~~脆弱性~~ ~~ゆるふわ~~ ~~弱い型~~ 動的

クソザコ ~~型なし~~ 意味不明 ~~弱い~~

自動変換 貧弱 ~~Perlっぽい~~ ~~適当~~

PHPについての認識は概ね間違い

PHPの進化は型宣言の進化

型がついていない関数(PHP5)

```
function add($a, $b) {
    return $a + $b;
}
```


pixiv

スカラー型宣言(PHP7)

```php
function add(int $a, int $b): int {
    return $a + $b;
}
```

int + intって
本当にintなの？

pixiv

広い値をとるにはfloatが必要

```
function add(float $a, float $b): float {
    return $a + $b;
}
```

ひとつの解決策では
あるが… 不必要にfloat
を強制するのか

pixiv

PHPDocの型注釈(アノテーション)

```php
/**
 * @param int|float $a
 * @param int|float $b
 * @return int|float
 */
function add($a, $b) {
    return $a + $b;
}
```

pixiv

ユニオン型宣言 (PHP8.0)

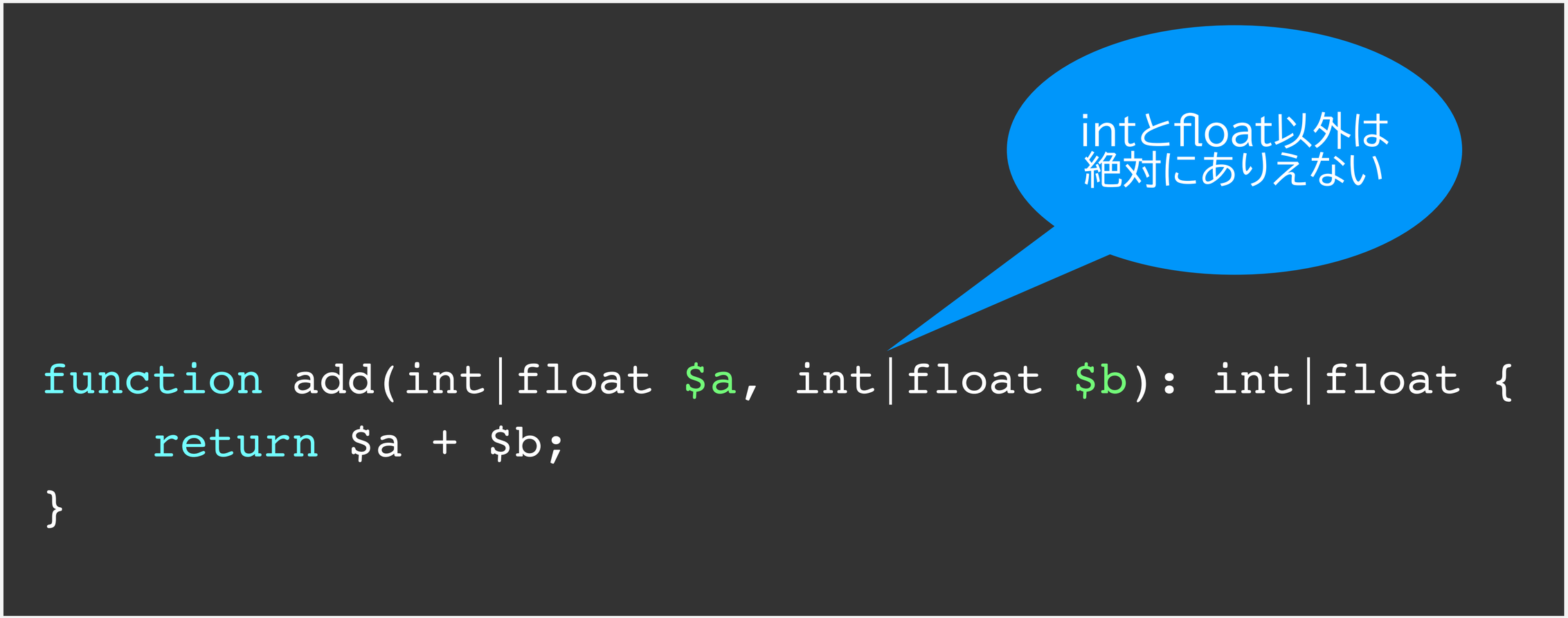
```php
function add(int|float $a, int|float $b): int|float {
    return $a + $b;
}
```


pixiv

PHPは強く実行時に保証される型

- PHPはいわゆる動的言語
- 構文で型宣言された言語要素(関数パラメータ、戻り値、プロパティ)は、
  実行時にその型の値であることが絶対に保証される
  - null安全 (明示的にnullableとして宣言しない限りnullは受け入れない)
- declare(strict_types=1) で実行時の振る舞いが少し変わるが、
  型安全性という意味ではあまり変わらないので今回は話しません

ユニオン型宣言 (PHP8.0)




```php
function add(int|float $a, int|float $b): int|float {
    return $a + $b;
}
```


pixiv

PHPDocの型注釈(アノテーション)

```php
/**
 * @param int|float $a
 * @param int|float $b
 * @return int|float
 */
function add($a, $b) {
    return $a + $b;
}
```

pixiv

201X年、PHPは型の炎に包まれた!!

いまやPHPは静的型付きと言っても過言ではない

（本当か…？）

だが型なしは
滅びていなかった

型宣言では配列要素の型を表現できない

PHP組み込み型の限界

- array型やArrayObjectのような汎用的な型の要素の型がわからない
- evalやjson_decode()などデシリアライズ結果の型がわからない
  - ……これらの事情によりPHPコードをいくら静的解析しても、
    型が一意に決定されることはない
- 型宣言されてない既存コード、動的プロパティや動的メソッドが宣言できる
  - 静的解析でも型推論ではない別のアプローチが必要

Bookクラスは複数の著者を持つ

```
class Book
{
    function __construct(
        private array $authors
    ) {
        // 何も書かなくてもプロパティに代入さ
    }
}
```

arrayにすることで
複数であることを表現

Authorクラスの
情報が減ってる…

pixiv

arrayには無限の可能性

```php
<?php

function searchUser(int $user_id): array
{
    // ...
}

$user = searchUser(11);

$user[???] ← どんなキーで格納されてるのか不明

```


pixiv

PHPDocの帰還

```
class Book
{
    function __construct(
        /** @var list<Authors> */
        private array $authors
    ) {
        // 何も書かなくてもプロパティに代入される
    }
}
```

配列の形状はコメントで書く

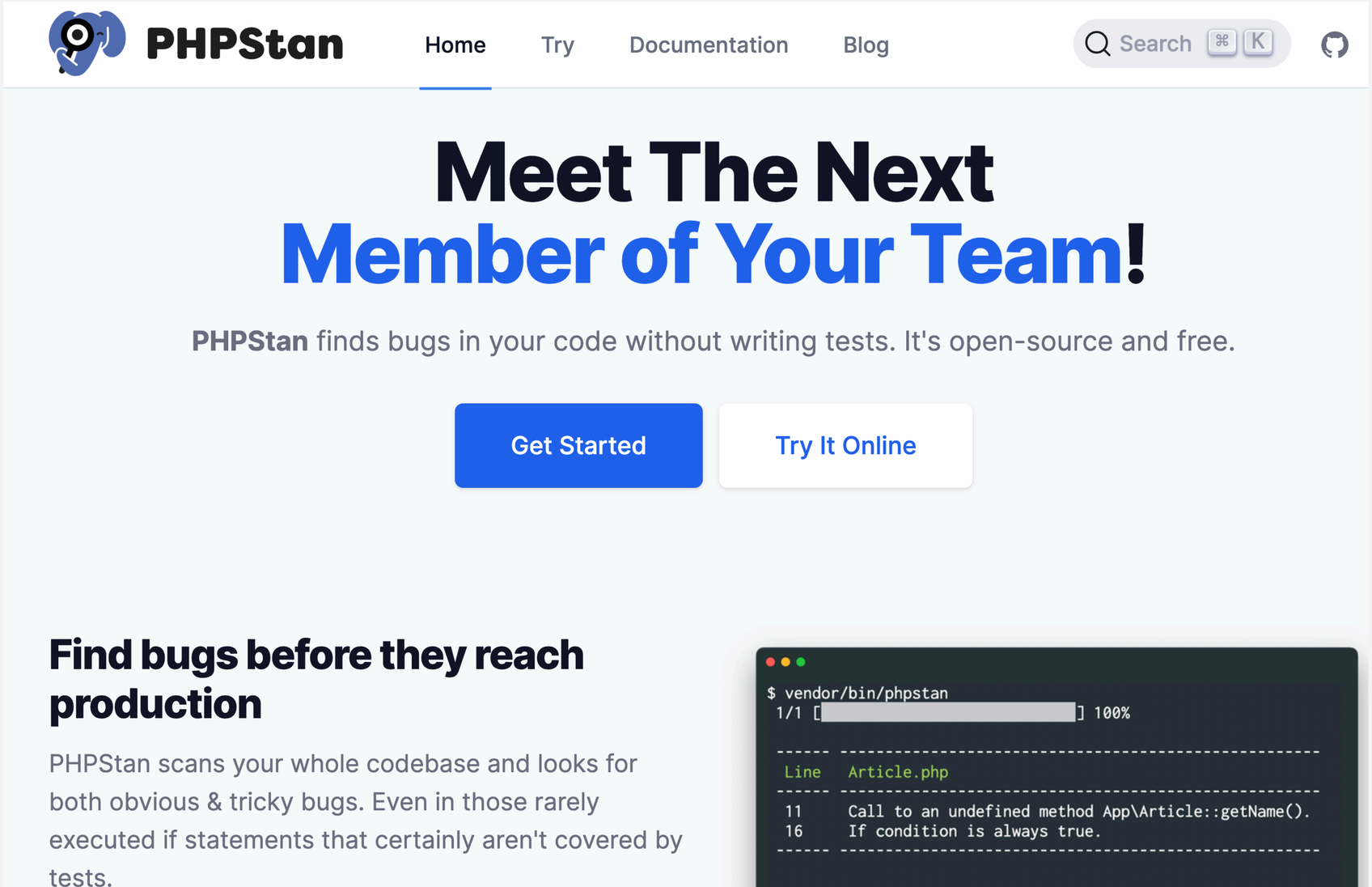
pixiv

PHPと型の現在地点

- PHPがランタイムに保証する型と静的解析で検査できる型に分かれる
  - function f(int $n): int のような型宣言は実行時安全
  - function g(array $a): array のような型宣言はarrayの要素が不明
- PHPDocの型注釈はコメントなので自由に記述できる
  - 標準的な型のほかに静的解析ツールの間で実験的な型も実装されている
  - PHPStanやPsalmの間ではある程度の相互運用が考慮されている

PHPStan

PHP Static Analysis Tool



pixiv

PHPStanとは

- 2016年から開発されているPHPの静的解析ツール
  - Ondřej Mirtesさんの個人プロジェクト、2021年からフルタイム開発
- 開発当初は純粋な静的解析ではなく実行時リフレクションを用いることで高速な解析を実現していた
  - 現在は静的解析がデフォルトで、レガシープロジェクトに導入しやすくなった
- その他のPHP静的解析ツールにはPsalm, Phan, Qodana（PhpStorm）

PHPStan拡張

- PHPStanの機能は組み込みのもの含めextensionとして実装されてる
  - 独自拡張をしたいというときも組み込みの拡張が非常に参考になる
  - PHPでよく使われるパッケージ管理ツールのComposerから追加可能
- よく使われるのは「Rule拡張」と「Type-Specifying拡張」の二種
  - **今回はクイックマスターなので、この話をします**

PHPStanの基礎

- PHPStanは構文木はPHP-Parserのノードを利用している
- いろいろ困ったらPHPStanの本体コードを見にいくことになる
  - https://github.com/phpstan/phpstan-src
  - （-srcがついてるリポジトリを見ないとコードが読めないので注意）
- PHP組み込み以外の型システムを用意している
- いわゆる述語メソッドがboolではなく三値（yes/no/maybe）を返す

PHPStanの型システム

- 型はオブジェクトとして表現されているので実態はコードを読む
  - https://github.com/phpstan/phpstan-src/tree/1.10.x/src/Type
- PHPの組み込み型、Union/Intersectionなどの複合的な型、
  組み込み型を継承した定数型、これらの型とintersectionで組み合わせて
  複合的な型を表現するAccessory型などがある
  - たとえばリスト（連番の配列） ArrayType&AccessoryArrayListType
  - これだけで一時間くらい話せるネタ

Rule拡張でできること

- 構文木から判断できること全部！
  - 渡されてきたScopeオブジェクトから型がとれる
  - コード上に直接書かれた値や具体的に絞り込まれた型には実際の値もとれる

Rule拡張

```php
interface Rule
{

    /**
     * @phpstan-return class-string<TNodeType>
     */
    public function getNodeType(): string;

    /**
     * @phpstan-param TNodeType $node
     * @return (string|RuleError)[] errors
     */
    public function processNode(Node $node, Scope $scope): array;

}
```

実装するルールが
対象とする構文木の
ノードクラス名

構文木を受け取って
エラーを配列として返す

pixiv

StaticMockRule拡張

読者になる

pixiv inside [archive]

pixiv insideは移転しました！ » https://inside.pixiv.blog/

2014-02-27

PHP のスタティックメソッドをモック化する

24 B!ブックマーク　ツイート

初登場の @tototoshi です。今回は pixiv のユニットテストで利用しているモックライブラリの紹介をします。

ここ2ヶ月くらいの間、レガシー化したとあるモジュールのリファクタリングに取り組んでいました。 リファクタリングにはテストコードが必須です。しかし今ではすっかりテストを書く文化が根付いている pixiv にもテストコードがない時代がありました。リファクタリングが必要な古いコードにはテストコードがないことが多く、そういったコードに新たにテストをつけていくのはなかなか大変です。テストの概念のないプロジェクトはテスト可能なように設計・実装されていません。テストを書くのが大変なのではなく、書けるようにするまでが大変です。

pixiv insideとは

https://inside.pixiv.blog/ に移転しました。こちらは2016年末までのアーカイブです。

読者です 234

このブログについて

月別アーカイブ

▶ 2016 (50)
▶ 2015 (60)
▼ 2014 (63)
2014 / 12 (20)
2014 / 11 (7)
2014 / 10 (1)
2014 / 9 (2)
2014 / 7 (2)
2014 / 5 (1)
2014 / 4 (1)
2014 / 3 (4)

pixiv

Type-Specifying拡張

```php
interface DynamicMethodReturnTypeExtension
{

    public function getClass(): string;

    public function isMethodSupported(MethodReflection $methodReflection): bool;

    public function getTypeFromMethodCall(MethodReflection $methodReflection, MethodCall $methodCall, Scope $scope): ?Type;

}
```

メソッド呼び出しのクラス名

実装する拡張が
対象のメソッド呼び出しかどうか

構文木を受け取って型を返す

pixiv

Type-Specifying拡張でできること

- 関数呼び出しやメソッド呼び出し、プロパティなどに、
  型宣言やPHPDocで記述されていない型をつけることができる
- バリデーション関数で実行時に絞り込まれたときに型をつける

ReverseRouteReturnTy

メソッド呼び出しのクラス名

```php
class ReverseRouteReturnTypeExtension implements DynamicStaticMethodReturnTypeExtension
{
    public function getClass(): String
    {
        return 'ReverseRoute';
    }

    public function isStaticMethodSupported(MethodReflection $methodReflection): bool
    {
        return true;
    }
```

クラスに属する
メソッド全部が対象

pixiv

ReverseRouteReturnTypeExtension

```
public function getTypeFromStaticMethodCall(
    MethodReflection $methodReflection,
    StaticCall $methodCall,
    Scope $scope
): Type {
    return TypeCombinator::intersect(
        new StringType(),
        new AccessoryNonEmptyStringType(),
    );
}
```

新しい型
string&non-emptyを返す

pixiv

ParamHelperTypeSpecifyingExtension

pixiv inside
pixivのなかの話
つくる
はたらく
ささえる
記事を検索

トップ ＞ 勉強会/イベント登壇 ＞ 型安全なHTTP入力を保証するParamHelper

型安全なHTTP入力を保証するParamHelper

Article by yosatak　2021-09-09

こんにちは。pixiv運営本部 開発支援チームでpixivのコーディング環境の向上をしているyosatakです。

pixiv