PHPStan超クイックマスター

PHPStan “super” quick master

pixiv Inc.
USAMI Kenta

pixiv

2021.10.02

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 エンジニア
  - 最近はピクシブ百科事典(dic.pixiv.net)を開発しています
- Emacs Lisper, PHPer
  - 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

Attributeを極める

Mastering Attributes

pixiv Inc.
USAMI Kenta

pixiv

x

2022-03-25

さて

PHPerKaigiの
チケットを買うと
パンフレットがもらえる

PHPStanクイックガイド 2023

 うさみけんた@tadsan

PHPStan（PHP Static Analysis Tool）はコードを実行せずに検査できるツールです。本稿では業務アプリケーションにPHPStanを導入するまでに押さえておきたい事柄を記述します。

本稿についての補遺は `https://scrapbox.io/php/Kaigi2023` にまとめているので、併せてお読みください。

導入

PHPStanは本稿記述時点の1.9.x系において、PHP 7.2以降で実行できます。PHPStanは

```
composer require --dev phpstan/phpstan \
    phpstan/extension-installer
```

のようなコマンドでのインストールが基本です。

プロジェクトルートの`phpstan.dist.neon`に、以下

コード上の多くの良くない傾向が示唆されているだろうと思いますが、まずは現状をポジティブに捉えて、PHPStanはプロジェクトを**さらに良くしていくための道具**なのだと考えましょう。

このコマンドを実行すると`phpstan-baseline.neon`というファイルが生成されます。先ほど**「ポジティブに捉えよう」**と言ったばかりなのですが、「`Function xxx not found.`」や「`Constant XXX not found.`」といったエラーは最初に解決しておくべきです。

PHPStanは特別な設定なしでコードを解析できるように設計されています。特に`composer.json`で`autoload`が設定されていれば自動的に解析されるようになっています。スクリプト実行時に関数や定数が動的に定義される場合やクラス名が動的にエイリアスされる場合は、初期化ファイルを`composer.json`の`autoload.files`か、`phpstan.dist.neon`の`bootstrapFiles`に追加してください。たとえばCodeIgniterプロジェクトであれば`"vendor/codeigniter4/framework/system/Test/bootstrap.php"`を追加するのがよいでしょう。

さて、ここで作ったベースラインファイルは`include`

PHPStan
=
Static Analyzer

実行しなくても
バグの可能性を
検知できる

全8ページで PHPStanの 基礎を押えられる

pixiv

実務で最低限使うために知っておきたいことを詰め込んだ… はず

PHP本来の型のコンセプトからPHPDocによる拡張までを網羅

僕と契約して
PHPStanを完全に
マスターしてよ

という話をしたい
のではない

PHPStanは
完全無料なので
いますぐ入れよう

これからプロになりたい人に本当に伝えたいPHPDocの書きかた

その1
配列は使い分ける

pixiv

世の中には
いろんな配列がある

[1, 2, 3]

```
[
    'name' => 'Miku',
    'age' => 14,
]
```


pixiv

```
[
    'Miku', 'Rin', 'Luka',
]
```

```
[
    'main'   => 'Miku',
    'sub'    => 'Rin',
    'chorus' => 'Luka',
]
```


pixiv

単なる型で書くと
array

```
interface X
{
    function a(): array;
}
```

```
function (X $obj) {
    $a = $obj->a();
};
```

```
function (X $x) {
    $a = $x->a();
    $v = $a[0];
};
```

配列から取り出した要素の型がわからないのは都合が悪い

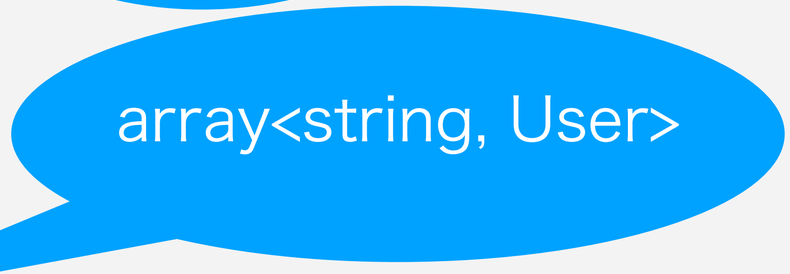
array<int>
array<string>

pixiv

ちょっと前は
int[] とか string[]
とか書かれてた

PSR-5式の配列記法(ジェネリクス)

- 配列インデックスの型を指定できる
- 以下の配列が区別できる
  - [12 => new User(), 34 => new User()]
  - ['太郎' => new User(), '花子' => new User()]
  - [new User('太郎'), new User('花子')]



pixiv

配列には2通りある

同じような構造が繰り返す配列

キーによって
別の意味の値が
入っている配列

```
$a = [
    'name' => '初音ミク',
    'age' => 16,
];
// array{name:string, age: int}
```


pixiv

```
array{
    name: string,
    age: int
}
```

```
$a = ['初音ミク', 16];
// array{0: string, 1: int}
```


pixiv

繰り返すなら<>
キーごとに違うなら{}

その2
型は入出力に宿る

pixiv

クエリパラメータから値をとりたい

```
function getById(int $id): Article
{
    //
}
```


pixiv

クエリパラメータから ID値をとりたい

実際あぶない

```
$id = $_GET['id'];
$article = getById($id);
```


pixiv

```
// ?id=1
$_GET['id'] === '1';
```

```
// ?id=a
$_GET['id'] === 'a';
```


pixiv

```
// ?id[]=a
$_GET['id'] === ['a'];
```

危険ではないが
よくはない

```
$id = (int)$_GET['id'];
```


pixiv

```
$id = filter_var(
    $_GET['id'] ?? '',
    FILTER_VALIDATE_INT
);
```

安全に
int|false

pixiv

その3
むやみに@var書かない

pixiv

初心者殺しなことに @varには複数の用法

@varはプロパティに付けて書くもの

```
class Book
{
    /** @var string */
    public $name;
}
```

```
class Book
{

    public string $name;
}
```


pixiv

```
class Book
{
    /** @var list<Author> */
    public $authors;
}
```

```
class Book
{

    public list<Author> $authors;
}
```

```php
class Book
{
    /** @var list<Author> */
    public array $authors;
}
```

@varはconstには基本的に付けない

```
class Book
{
    /** @var list<int> */
    const IDS = [1, 2, 3];
}
```

できれば
つけない方がいい

```
class Book
{
    /** @var list<int> */
    const IDS = [1, 2, 3];
}
```


pixiv

@varを
つけない場合

```
Book::IDS[0] // => 1
Book::IDS[1] // => 2
Book::IDS[2] // => 3
```


pixiv

```
Book::IDS[0] // => int
Book::IDS[1] // => int
Book::IDS[2] // => int
```

@varを付けると
弱くなる

pixiv

ローカル変数に@varを書く

できれば
書かないでくれ！！

むやみに
@varを書かないテク

そのうちどこかに
ちゃんと書きます…

pixiv