ジェネリクスを自分のものにする

Getting Started with Generic Programming

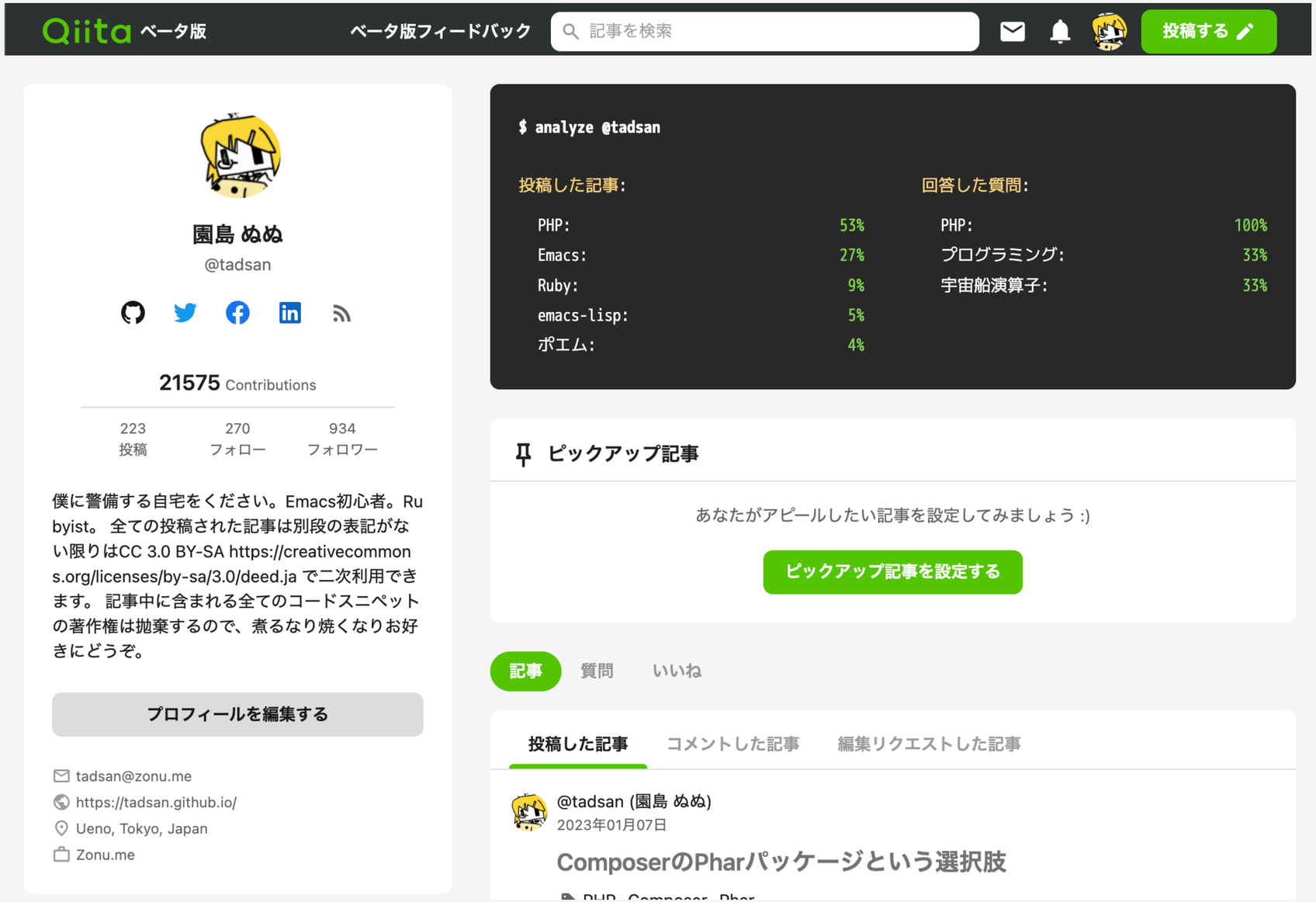
pixiv Inc.
USAMI Kenta

pixiv

2022-01-25
Qiita Night ～PHP～

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
  - すごい肩書の発表者が多いですが僕はヒラのPHPerです（IC?）
- Emacs Lisper, PHPer
  - Emacs PHP Modeを開発しています（2017年-）


Qiita ベータ版
ベータ版フィードバック
記事を検索
投稿する
園島 ぬぬ
@tadsan
21575 Contributions
223 投稿
270 フォロー
934 フォロワー
僕に警備する自宅をください。Emacs初心者。Rubyist。 全ての投稿された記事は別段の表記がない限りはCC 3.0 BY-SA https://creativecommons.org/licenses/by-sa/3.0/deed.ja で二次利用できます。 記事中に含まれる全てのコードスニペットの著作権は拋棄するので、煮るなり焼くなりお好きにどうぞ。
プロフィールを編集する
tadsan@zonu.me
https://tadsan.github.io/
Ueno, Tokyo, Japan
Zonu.me
$ analyze @tadsan
投稿した記事:
PHP: 53%
Emacs: 27%
Ruby: 9%
emacs-lisp: 5%
ポエム: 4%
回答した質問:
PHP: 100%
プログラミング: 33%
宇宙船演算子: 33%
ピックアップ記事
あなたがアピールしたい記事を設定してみましょう :)
ピックアップ記事を設定する
記事
質問
いいね
投稿した記事
コメントした記事
編集リクエストした記事
@tadsan (園島 ぬぬ)
2023年01月07日
ComposerのPharパッケージという選択肢

pixiv

カレンダー1

日	月	火	水	木	金	土
27	28	29	30	1 @tadsan Attributesで実現するPHP8時代のバリデータ 編集する	2 @noritakalzumi PHP で Postgres の boolean 型を安全に扱う	3 @naopusyu Composerコマンドを使ってパッケージの脆弱性をチェックする
4 @tadsan ZendEngineにえこひいきされる標準関数たち(前篇) 編集する	5 @tadsan ZendEngineにえこひいきされる標準関数たち(後篇) 編集する	6 @77web@github レガシーシステムに書けるテストについて	7 @tadsan PHPer vs モナド 編集する	8 @rana_kualu 【PHP8.2】PHP8.2がリリースされたので新機能全部やる	9 @ockeghem 徳丸本2版の実習環境をDockerに移植した話	10 @tadsan ComposerのPharパッケージという選択肢 編集する
11 @foluyucic PHPでPIDが存在するかチェック【file_exists vs posix_getpgid vs is_dir vs posix_kill】	12 @rana_kualu 【PHP Next】PHPにclamp関数がほしい	13 @takaram PHP8.2で非推奨になったエンコーディングたち	14 @hanhan1978 PHP と IEEE 754 の話	15 @tadsan 名前空間をさっくり理解する 編集する	16 @tzmfreedom symfony/pantherによるJavaScriptを含めた自動テスト	17 @tanakahisateru ちょうぜつ設計とは
18 @il-m-yamagishi Laravel のリクエストだけでなく、レスポンスもバリデートしよう	19 @ippey_s GitHub Codespacesでリモート PHP開発環境を整えてPhpStormで使う3ステップ	20 @foluyucic Node.js 12 actions are deprecated. For more information see:	21 @foluyucic Code coverage driver not available.	22 @foluyucic Your lock file does not contain a compatible set of packages. Please run composer update.	23 @tadsan PHPからJavaScriptにデータを受け渡すときに考えること 編集する	24 @tadsan なぜ出力時のHTMLエスケープを省略してはならないのか 編集する
25 @mumumuorg [翻訳記事] PHP のリリースマネージャーについて	26	27	28	29	30	31

PHP

27460 記事

72219 フォロワー

記事をフォロー中

質問をフォロー中

ユーザーランキング　週間　月間　すべて

1. @tadsan (園島 ぬぬ)
   481 Contributions
2. @triple321jhango (カイリーー)
   119 Contributions
3. @ucan-lab
   102 Contributions

静的解析してますか？


今日からできる安心型付け入門
Introduction of typing in PHP
2021-05-29 YouTube
PHPカンファレンス沖縄

pixiv

配列、ジェネリクス、PHPで書けない型

Arrays, Generics, and Types that cannot be type-declared.

pixiv Inc.
USAMI Kenta

pixiv

x

2021. 10. 02

pixiv

ジェネリクスの話をしよう

ウィキペディア
フリー百科事典

Fafnir127

ジェネリックプログラミング

28の言語版

ページ　ノート　　閲覧　ソースを編集　履歴表示　その他

出典: フリー百科事典『ウィキペディア（Wikipedia）』

ジェネリック（総称あるいは汎用）**プログラミング**（英: generic programming）は、具体的なデータ型に直接依存しない、抽象的かつ汎用的なコード記述を可能にするコンピュータプログラミング手法である。

概要 [ソースを編集]

ジェネリックプログラミングはデータ型でコードをインスタンス化するのか、あるいはデータ型をパラメータとして渡すかということにかかわらず、同じソースコードを利用できる[1]。ジェネリックプログラミングは言語により異なる形で実装されている。ジェネリックプログラミングの機能は1970年代にCLUやAdaのような言語に搭載され、次にBETA、C++、D、Eiffel、Java、その後DECのTrellis/Owl言語などの数多くのオブジェクトベース (object-based) およびオブジェクト指向 (object-oriented) 言語に採用された。

1995年の書籍デザインパターン[*要文献特定詳細情報*]の共著者[*誰?*]は（Ada、Eiffel、Java、C#の）ジェネリクスや、（C++の）テンプレートとしても知られるパラメータ化された型 (parameterized types) としてジェネリクスについて触れている。これらは、型を指定することなく、型を定義できるようにする（型は使用する時点で引数として与えられる）。このテクニック（特にデリゲーションを組み合わせるとき）は非常に強力である。

ジェネリックプログラミング - Wikipedia(2022-05-02T17:30:09の版)より引用

データ型に
依存しない…？

抽象的かつ
汎用的…？

まず基本的な静的型の話

静的型付け (statically typing)

- 静的 = 実行しなくても型がわかる状態（反対語は動的 = プログラム実行）
- 型宣言（コードに型を書く）
  - 現状のPHPではパラメータとプロパティに型宣言できる
  - PHPは処理系が実行時の型を保障してくれる（口約束ではない！）
- 型推論（文字で型を書かなくても空気を読んでくれる）
  - $a = count($array) のとき $a の型は常に int
  - $b = 1 + 2 の結果は int(3)

型付けの具体例を見ていきましょう

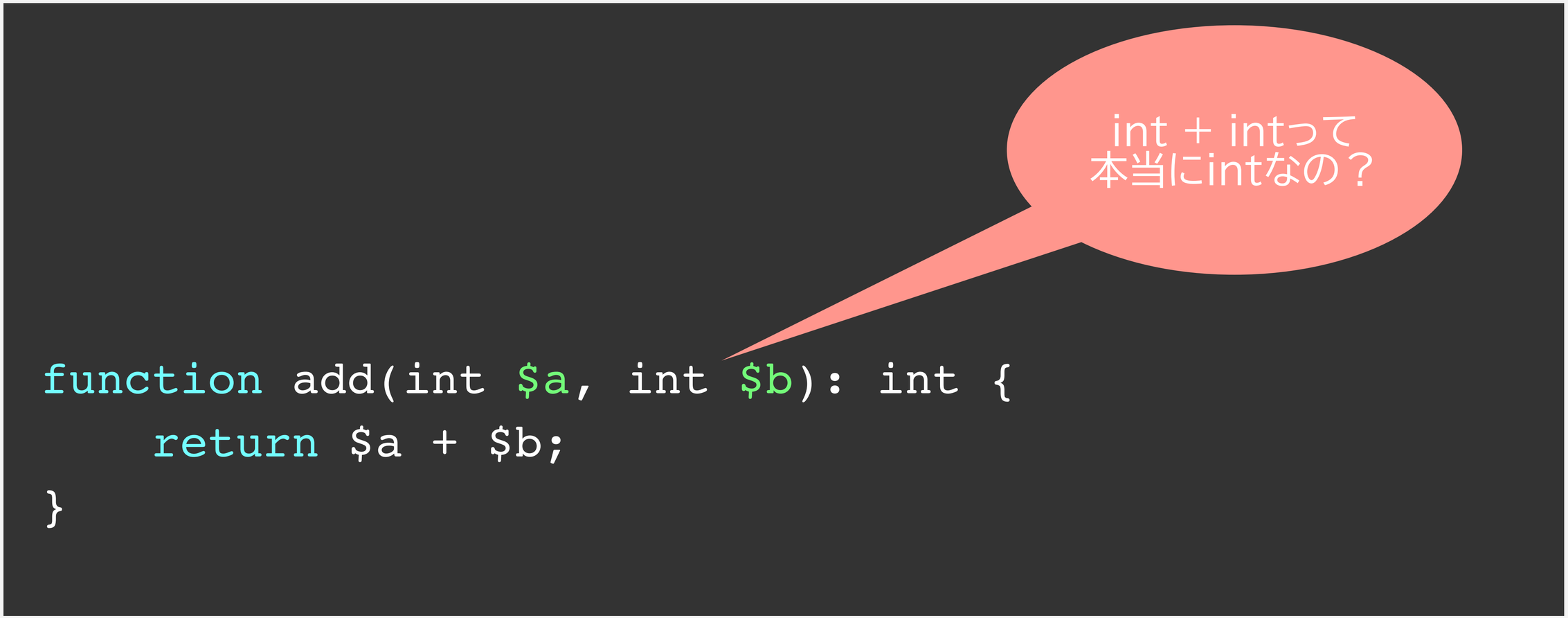
まずはジェネリクスと
あまり関係ない例

型がついていない関数(PHP5)

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

pixiv

スカラー型宣言(PHP7)




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


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

基本的な型宣言

- PHPの言語として記述できる静的な型宣言は大まかに3種類
  - 具体的な型
    - int, float, string, bool, array, DateTimeInterface, ...
  - 複合的な型
    - A|B, int|false, ?DateTime, ArrayAccess&Traversable ...
  - 型定義をしない（または mixed）

かくして世界は
型の炎に包まれた

だが型なしは
死滅していなかった

別の例を
考えてみましょう

配列の先頭要素を返す関数

```php
function first(array $a, mixed $default): mixed {
    return $a[0] ?? $default;
}

$v = first([0, 1, 2], 'default');
```

$vの型は何？

pixiv

int専用関数をつくろう

```php
function first_int(array $a): ?int {
    return $a[0] ?? null;
}

$v = first_int([0, 1, 2]) ?? 'default';
```


pixiv

string専用関数をつくろう

```php
function first_string(array $a): ?string
    return $a[0] ?? null;
}

$v = first_string(['a', 'b', 'c']) ?? 'default';
```

DateTime専用関数をつくろう

```php
function first_DateTime(array $a): ?DateTime
    return $a[0] ?? null;
}

$v = first_DataTime([]) ?? 'default';
```


pixiv

関数量産は
めんどくさい

ユニオン型は問題解決しない

```php
function first(array $a): int|string|DateTime|null {
    return $a[0] ?? $default;
}

$v = first([0, 1, 2], 'default');
```

なんでDateTimeとか
出てくるの？

pixiv

?DateTimeとか int|falseくらいなら問題ない

それを補うのが PHPDoc

PHPDocタグ
いくついえるかな

@param

@return

@var

@template

…なにそれ？

pixiv

@template は型変数

```
/**
 * @template TValue
 * @template TDefault
 * @param array<TValue> $a
 * @param TDefault $default
 * @return TValue|TDefault
 */
function first(array $a, mixed $default) {
```


pixiv

PHPStan

Home | Try | Documentation | Blog

Meet The Next Member of Your Team!

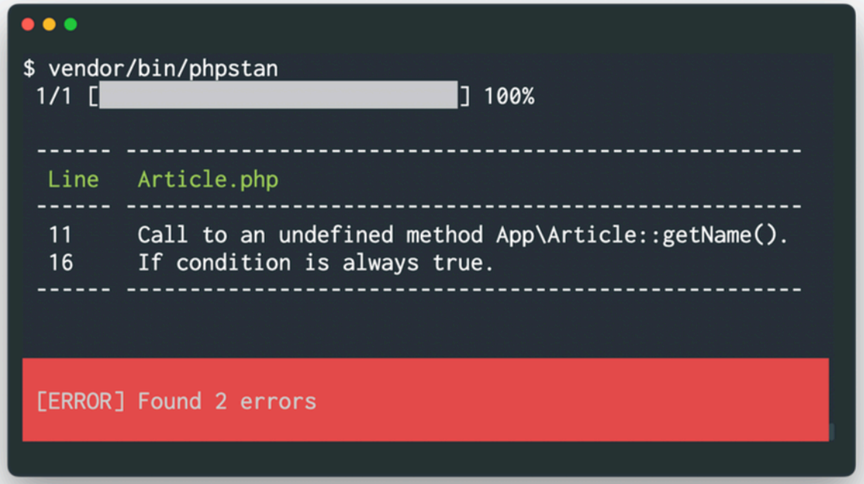
PHPStan finds bugs in your code without writing tests. It's open-source and free.

Get Started | Try It Online

Find bugs before they reach production

PHPStan scans your whole codebase and looks for both obvious & tricky bugs. Even in those rarely executed if statements that certainly aren't covered by tests.

You can run it on your machine and in CI to prevent those bugs ever reaching your customers in production.



pixiv

PHPStan

Home | Try | Documentation | Blog

Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »

```php
1  <?php declare(strict_types = 1);
2
3  function f(?int $value): void
4  {
5      if ($value === null) {
6          \PHPStan\dumpType($value);
7      } else {
8          \PHPStan\dumpType($value);
9      }
10         \PHPStan\dumpType($value);
11 }
12
```

Share | Level 9 | Strict rules | Bleeding

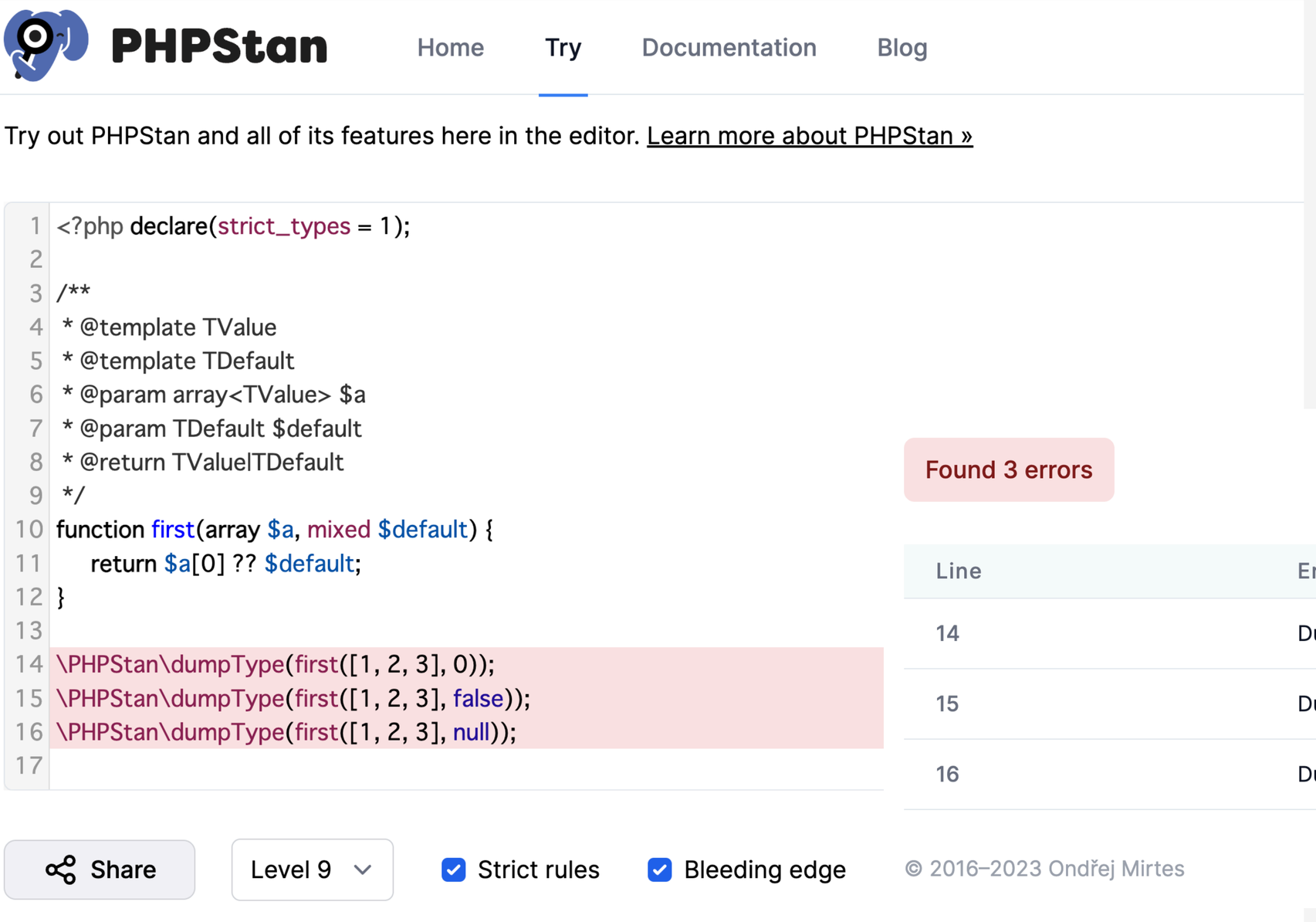
Found 3 errors

Line	Error
6	Dumped type: null
8	Dumped type: int
10	Dumped type: int|null

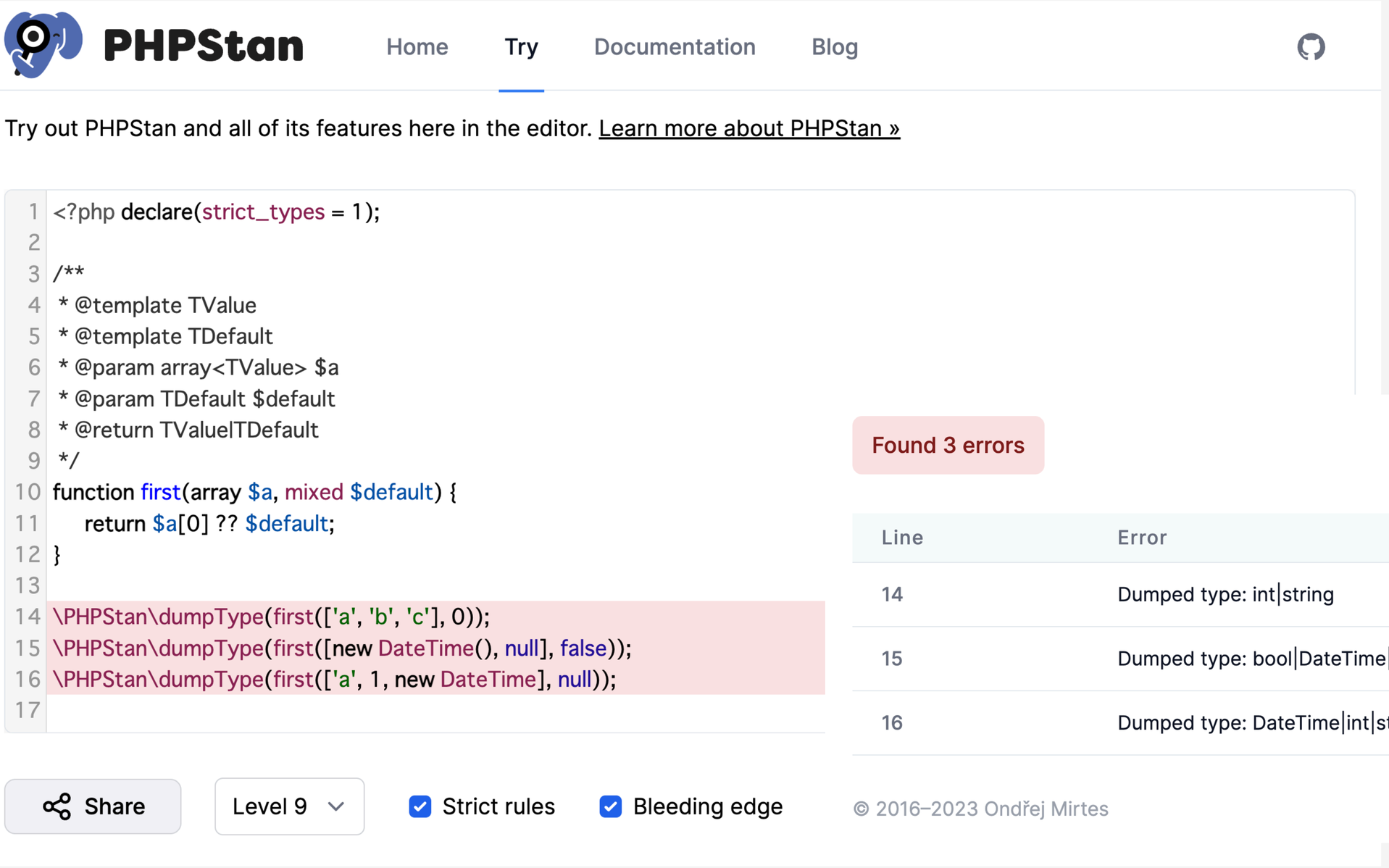
pixiv

やっていきましょう

pixiv


PHPStan
Home
Try
Documentation
Blog
Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »
1 <?php declare(strict_types = 1);
2
3 /**
4 * @template TValue
5 * @template TDefault
6 * @param array<TValue> $a
7 * @param TDefault $default
8 * @return TValue|TDefault
9 */
10 function first(array $a, mixed $default) {
11 return $a[0] ?? $default;
12 }
13
14 \PHPStan\dumpType(first([1, 2, 3], 0));
15 \PHPStan\dumpType(first([1, 2, 3], false));
16 \PHPStan\dumpType(first([1, 2, 3], null));
17
Found 3 errors
Line
Error
14
Dumped type: int
15
Dumped type: bool|int
16
Dumped type: int|null
Share
Level 9
Strict rules
Bleeding edge
© 2016–2023 Ondřej Mirtes

pixiv


PHPStan
Home
Try
Documentation
Blog
Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »
1 <?php declare(strict_types = 1);
2
3 /**
4 * @template TValue
5 * @template TDefault
6 * @param array<TValue> $a
7 * @param TDefault $default
8 * @return TValue|TDefault
9 */
10 function first(array $a, mixed $default) {
11 return $a[0] ?? $default;
12 }
13
14 \PHPStan\dumpType(first(['a', 'b', 'c'], 0));
15 \PHPStan\dumpType(first([new DateTime(), null], false));
16 \PHPStan\dumpType(first(['a', 1, new DateTime], null));
17
Found 3 errors
Line
Error
14
Dumped type: int|string
15
Dumped type: bool|DateTime|null
16
Dumped type: DateTime|int|string|null
Share
Level 9
Strict rules
Bleeding edge
© 2016–2023 Ondřej Mirtes

pixiv

いまは内部型表現の制約でシンプルな型しか付かないが

そのうち
万病に効くようになる

超PHPerになろう

Enjoy PHP Programming

2022-04-28

条件付き戻り値型とPHPStan 1.6.0の新機能

PHPStan

この記事はPHPStan開発者のOndřej Mirtesによって2022年4月26日にPHPStan Blogに書かれた記事を翻訳したものです。

PHPStan 1.6.0 With Conditional Return Types and More!

phpstan.org 1 user

phpstan.org

条件付き戻り値型 (Conditional return types)

この機能の大部分はRichard van Velzenが開発しました。

PHPStanは初リリース以来、関数呼び出しで渡された引数によって様々な型を返す方法を提供してきました。いわゆる動的戻り値型拡張(dynamic return type extensions)は非常に柔軟です。実装できる任意のロジックによって型を解決できます。しかし、PHPStan拡張の核心となるコンセプトには学習コストがかかります。

プロフィール

zonu_exe PRO

すごいPHPerになろう

http://qiita.com/tadsan

読者です 32

このブログについて

検索

記事を検索

最新記事

PHPDocベースのassert、list型とPHPStan 1.9.0の新機能

条件付き戻り値型とPHPStan 1.6.0の新機能

PHPerKaigi 2021に参加して、それから

戻り値の記憶と忘却

超PHPerになろう

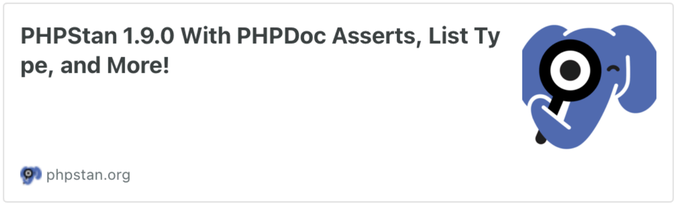
Enjoy PHP Programming

2022-11-03

PHPDocベースのassert、list型とPHPStan 1.9.0の新機能

PHPStan

この記事はPHPStan開発者のOndřej Mirtesによって2022年11月3日にPHPStan Blogに書かれた記事を翻訳したものです。



phpstan.org

PHPStan 1.9.0はまさにコミュニティの尽力によるものです。目玉機能はすべて、メンテナーである私(Ondřej)以外の誰かの貢献です。コードを書くのが嫌になったわけではないのですが、私が草むらで謎のバグを追いかけている間にも、ほかの人は新しい機能をより早く実装できるようになります。

私はここのところ緑色の「Merge」ボタンを一日に何度も押しています。私の役割はコードの主要な貢献者から、品質保障(QA)とプロジェクトビジョン[1]、そして継続的インテグレーション(CI)パイプラインの処理に移行しています。私は最近、貢献者向けのレターContributors update #1 2022でそれを認めました。これはあなたが貢献者ではなくとも読む価値があるでしょう。

プロフィール

zonu_exe PRO

すごいPHPerになろう

http://qiita.com/tadsan

読者です 32

このブログについて

検索

記事を検索

最新記事

PHPDocベースのassert、list型とPHPStan 1.9.0の新機能

条件付き戻り値型とPHPStan 1.6.0の新機能

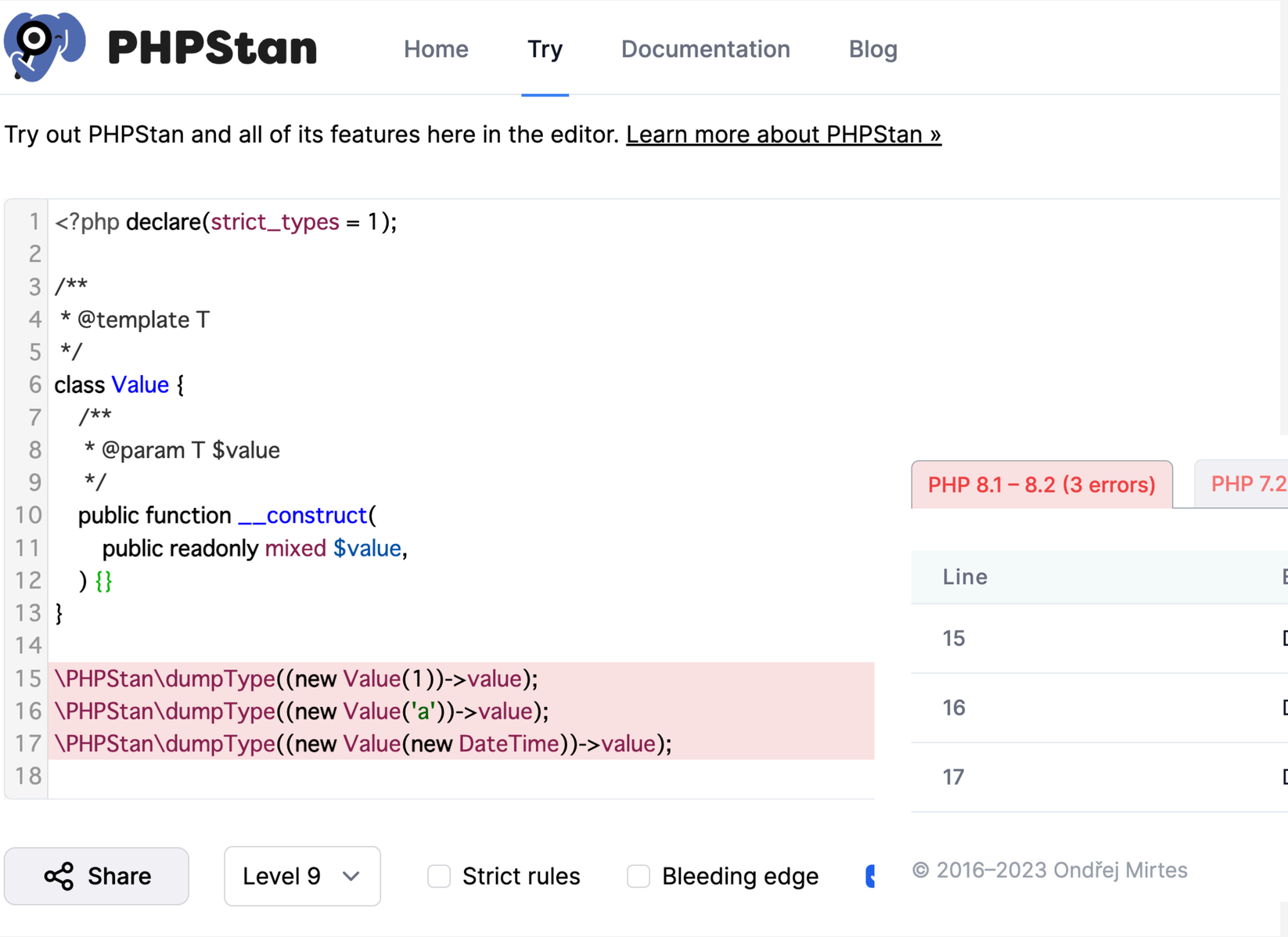
PHPerKaigi 2021に参加して、それから

戻り値の記憶と忘却

PHPDocを使ったPHPのジェネリクス

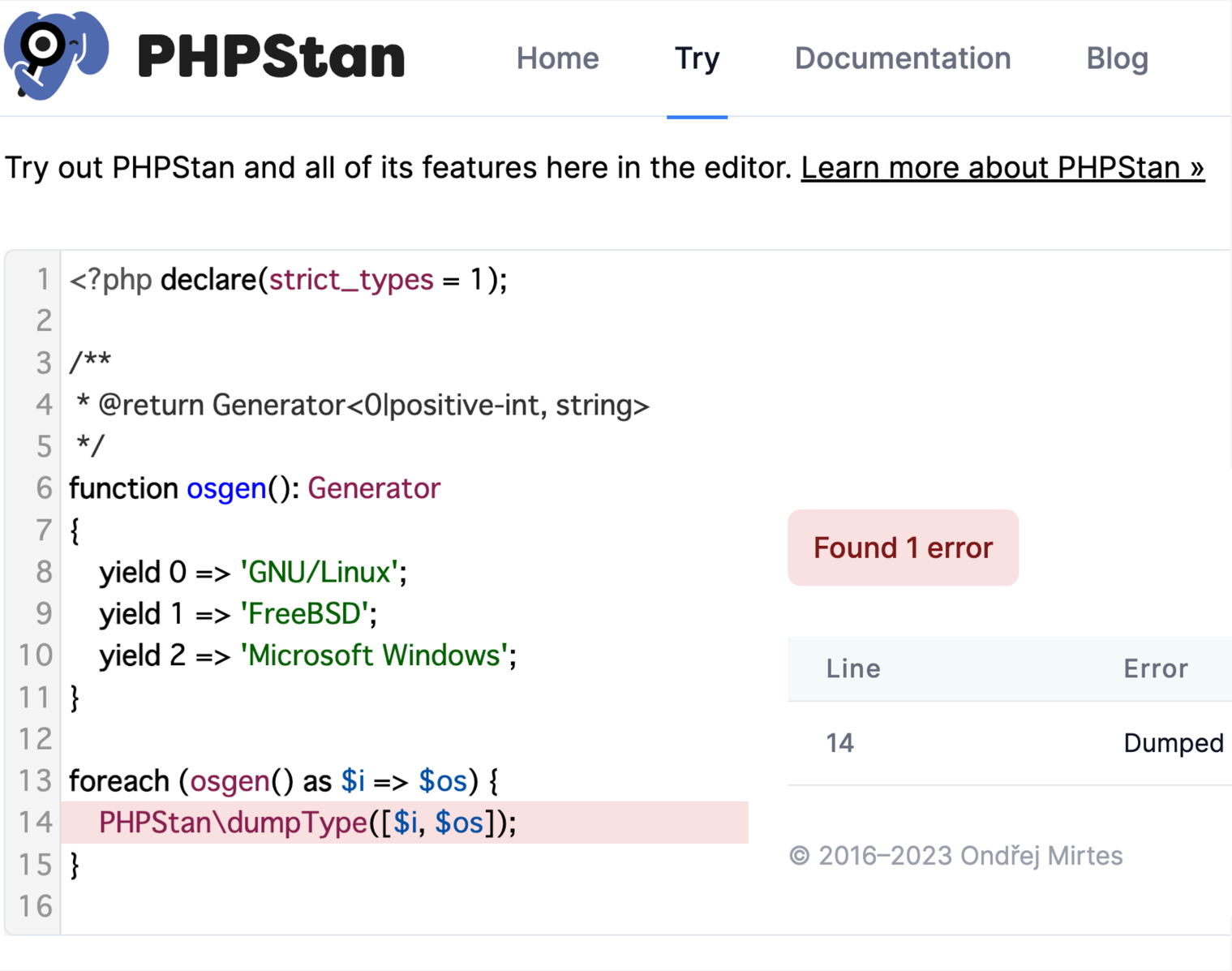
pixiv

テンプレートは
クラスにも書ける


PHPStan
Home
Try
Documentation
Blog
Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »
1 <?php declare(strict_types = 1);
2
3 /**
4 * @template T
5 */
6 class Value {
7 /**
8 * @param T $value
9 */
10 public function __construct(
11 public readonly mixed $value,
12) {}
13 }
14
15 \PHPStan\dumpType((new Value(1))->value);
16 \PHPStan\dumpType((new Value('a'))->value);
17 \PHPStan\dumpType((new Value(new DateTime))->value);
18
PHP 8.1 – 8.2 (3 errors)
PHP 7.2 – 8.0 (1 error)
Line
Error
15
Dumped type: int
16
Dumped type: string
17
Dumped type: DateTime
Share
Level 9
Strict rules
Bleeding edge
© 2016–2023 Ondřej Mirtes

pixiv

ジェネレータでも使える


PHPStan
Home
Try
Documentation
Blog
Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »
1 <?php declare(strict_types = 1);
2
3 /**
4 * @return Generator<0|positive-int, string>
5 */
6 function osgen(): Generator
7 {
8 yield 0 => 'GNU/Linux';
9 yield 1 => 'FreeBSD';
10 yield 2 => 'Microsoft Windows';
11 }
12
13 foreach (osgen() as $i => $os) {
14 PHPStan\dumpType([$i, $os]);
15 }
16
Found 1 error
Line
Error
14
Dumped type: array{int<0, max>, string}
© 2016–2023 Ondřej Mirtes

pixiv

コレクションも
簡単に定義できる

PHPStan

Home | Try | Documentation | Blog

Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »

```php
<?php

/**
 * @extends ArrayObject<int, int>
 */
class IntArray extends ArrayObject {}

/**
 * @extends ArrayObject<int, positive-int>
 */
class PositiveIntArray extends ArrayObject {}

$obj1 = new IntArray([0]);
\PHPStan\dumpType($obj1[0]);

$obj2 = new PositiveIntArray([0]);
\PHPStan\dumpType($obj2[0]);

```

Share | Level 9

PHP 8.0 – 8.2 (3 errors) | PHP 7.2 – 7.4 (3 errors)

Line	Error
14	Dumped type: int|null
16	Parameter #1 $array of class PositiveIntArray constructor expects array<int, int<1, max>>|object, array{0} given.
17	Dumped type: int<1, max>|null

tagged union

PHPStan

Home Try Documentation Blog

Try out PHPStan and all of its features here in the editor. Learn more about PHPStan »

```php
<?php declare(strict_types = 1);

/**
 * @phpstan-type Rectangle array{tag: 'rectangle', width : float, length : float}
 * @phpstan-type Circle array{tag: 'circle', radius: float}
 * @phpstan-type Prism array{tag: 'prism', width : float, float, height : float}
 * @phpstan-type Shape Rectangle|Circle|Prism
 */
class A
{
    /**
     * @param Shape $shape
     */
    public function f(array $shape): void
    {
        match ($shape['tag']) {
            'rectangle' => PHPStan\dumpType($
            'circle' => PHPStan\dumpType($sha
            'prism' => PHPStan\dumpType($sha
        };
    }
}
```

PHP 8.0 – 8.2 (3 errors) PHP 7.2 – 7.4 (4 errors)

Line	Error
17	Dumped type: array{tag: 'rectangle', width: float, length: float}
18	Dumped type: array{tag: 'circle', radius: float}
19	Dumped type: array{tag: 'prism', width: float, 0: float, height: float}

pixiv

モナド

Zenn

Add new

PHPer vs モナド

PHP # モナド Tech

こんにちは！ モナド書いてますか？ *私は書いてません！*

関数プログラミングにはモナドがつきものということで、近年はScalaやF#のような言語でもモナドや類似の概念が愛用されています。

ところで筆者はRubyやLispのような動的言語でのダイナミックなプログラミングが大好きです。Lispも関数型言語と呼ばれることがありますが、基本的には**モナドのモの字も出てきません**。そんなこんなで長期間にわたって関心を払いつつ、完全マスターしようと本を読んでも10年以上経ってもしっくりこないままです。

今回の記事の目的は、そのよくわからない概念をPHPに再現することで理解を試みることです。

記事を編集する

著者	にゃんだーすわん
公開	2023/01/04
本文更新	2023/01/24
文章量	約18,900字

にゃんだーすわん

にゃーん

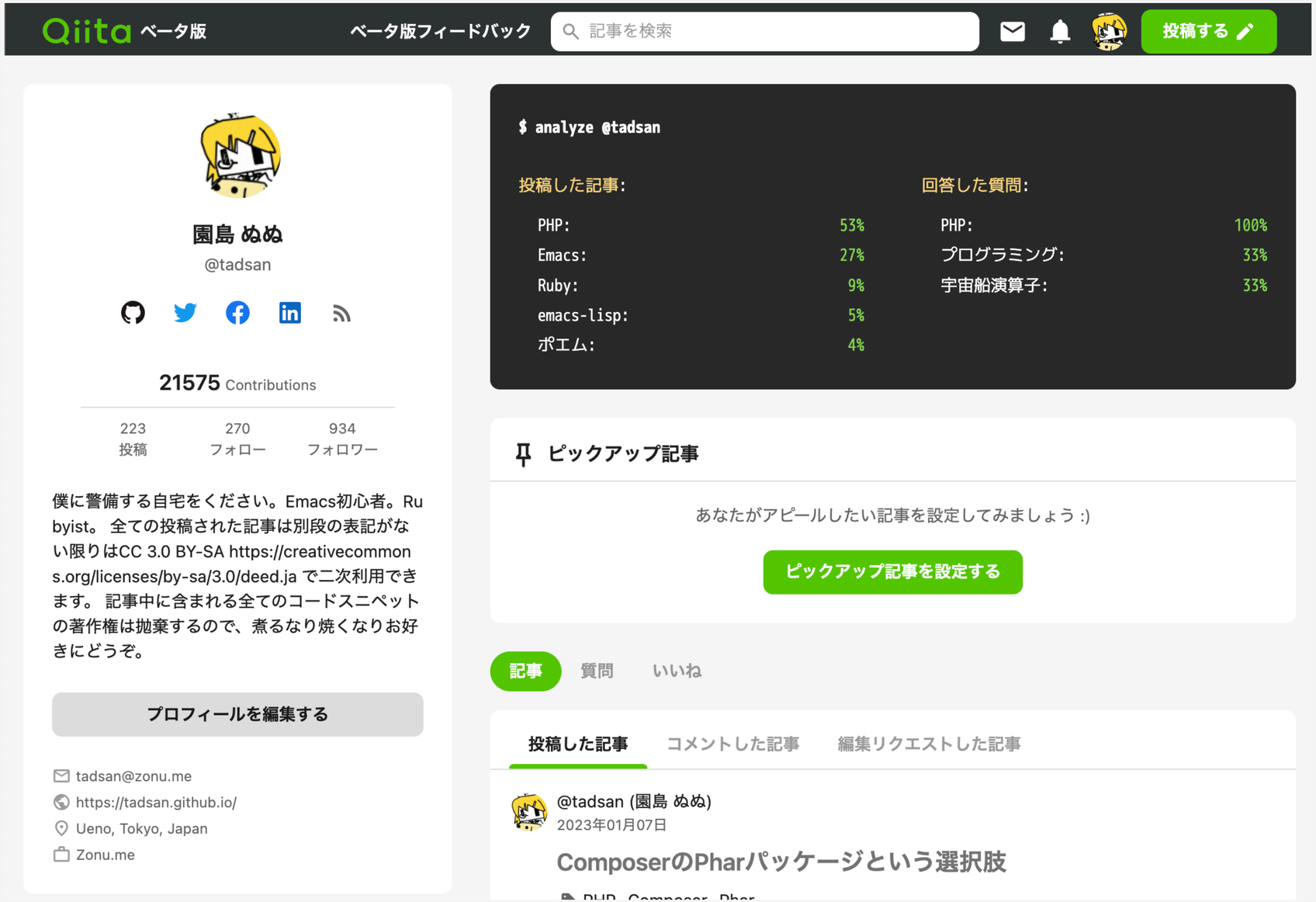
pixiv

実際難しいことは
(そんなに)
やってない

試行錯誤したことがない技術が難しいのはあたりまえ

PHPStanは
Webでこねこね
試行錯誤がやりやすい

PHPStan
完全にマスターしよう


Qiita ベータ版
ベータ版フィードバック
記事を検索
投稿する
園島 ぬぬ
@tadsan
21575 Contributions
223 投稿
270 フォロー
934 フォロワー
僕に警備する自宅をください。Emacs初心者。Rubyist。 全ての投稿された記事は別段の表記がない限りはCC 3.0 BY-SA https://creativecommons.org/licenses/by-sa/3.0/deed.ja で二次利用できます。 記事中に含まれる全てのコードスニペットの著作権は拋棄するので、煮るなり焼くなりお好きにどうぞ。
プロフィールを編集する
tadsan@zonu.me
https://tadsan.github.io/
Ueno, Tokyo, Japan
Zonu.me
$ analyze @tadsan
投稿した記事:
PHP: 53%
Emacs: 27%
Ruby: 9%
emacs-lisp: 5%
ポエム: 4%
回答した質問:
PHP: 100%
プログラミング: 33%
宇宙船演算子: 33%
ピックアップ記事
あなたがアピールしたい記事を設定してみましょう :)
ピックアップ記事を設定する
記事
質問
いいね
投稿した記事
コメントした記事
編集リクエストした記事
@tadsan (園島 ぬぬ)
2023年01月07日
ComposerのPharパッケージという選択肢

pixiv

今回はひたすら
実例を詰め込んで
話しました

ジェネリクスの知識は
なくても
PHPStanは使える

頭ん中

日本語で PHP の話をする Slack のグループを作ったよ【参加者募集】

2016年02月05日 16:05 PHP

タイトルのとおりですけど、日本語で PHP 関連の話題を中心とした雑談をするための Slack チームを作りました。

こちらの Slackin からどなたでも参加できます。

Join PHP ユーザーズ (日本語) on Slack!

pixiv

宣伝

PHPerKaigi 2023

TICKET BLOG LINK PROPOSALS TIMETABLE SPONSOR

PHPerKaigi 2023

チケットを購入

SCROLL

#phperkaigi @phperkaigi

pixiv

PHPStanクイックガイド 2022

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、以下のように記述してください。

```
parameters:
       level: 6
       paths:
              - src
              - tests
```

`paths`には実際にPHPファイルが格納されているディレクトリを指定しましょう。たとえばフレームワークによっては `app`, `inc`, `public`のようなディレクトリに配置されているかもしれません。

`level`は現在のところ1〜9が定義されており、文字列の`"max"`を指定することで将来にわたってPHPStanの提供する最高レベルのルールが適用されます。本稿ではmax想定で型を付けられるように解説しますが、プロジェクトの状態に合わせて5〜6から調整するのがよいでしょう。

PHPStanの設定ファイルとして採用されているNEONフォーマットはYAMLに似ています。インデントに意味がある言語なので、PHPStanがうまく動作しないと思ったらインデントにタブ文字とスペースが混在していないか疑いましょう。

まずはPHPStanの動作確認がてら**ベースラインファイル**を作成してみましょう。これは既知のエラーをファイルに保存しておくことで、プロジェクトを継続的に解析するための基準となるものです。たとえば、あなたのプロジェクトが既にリリースされてユーザーに価値を提供しているのであれば、PHPStanが出す警告というのは**お節介な杞憂**であるとも考えられるわけです。実際には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`することで検査時にエラーを抑制できます。その状態でプロジェクトを再検査すると**[OK] No Errors**になるはずです。これがPHPStanによる改善の出発地点です！

`phpstan.neon.dist`と`phpstan-baseline.neon`はそのまま`git add`して、プロジェクトメンバーで共有するのがいいでしょう。PHPStan設定ファイルの構成については紙幅の都合上、Webの補遺に譲ります。

IDEの有効化

インストールしたPHPStanを何に使えばいいのでしょうか。コードをちょっと変更する度にターミナルで手動実行して検査するのは実際かなりの手間です。PhpStormを使っているならば、PHPStanプラグインが最初から同梱されているので検証を有効化すれば編集中に非同期に検査され、エディタ上に検証結果が表示されます。

Vimではcoc.nvimやALE、EmacsではFlycheck + flycheck-phpstanを使うのがよいでしょう。VS Codeにもいくつも拡張がリリースされていますが、メンテナンスされていない古いものも混在しているので注意してください。

PhpStormは最新バージョンを使うことが非常に重要です。特に本稿執筆時点で次期メジャーバージョンのPhpStorm 2023.1では、本稿で取り上げるPHPStanとPsalmのタグや型のサポートを一層強化することが予告されています。これによってPHPStan向けに付けた型がエラーチェックだけでなく、入力補完にも活用されることが期待できます。

会社で実用しよう PHPStan