“慈悲深いユニオン”とは何か

What's benevolent union?

pixiv Inc.
USAMI Kenta

pixiv

PHP勉強会 in 神戸
2024-07-19 #関ガレ

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）

最近のプログラミング言語っぽい発表

- 「コードを自在に操るためのPHP文法入門」
  （2月 PHPカンファレンス関西2024）
- 「こんな静的解析導入は負けフラグ」（3月 PHPerKaigi）
- 「動的言語型付けバトル」（3月 PHPerKaigi）

Now, everyone

pixiv

Do you like static type analysis?

```php
<?php

$v = [1, 2, 3];
echo json_decode($v);
```

```php
<?php

$v = [1, 2, 3];
echo json_decode($v);
```

Found 2 errors

Line	Error
4	Parameter #1 $json of function json_decode expects string, array<int, int> given.
4	Parameter #1 (mixed) of echo cannot be converted to string.

pixiv

```php
<?php

$v = [1, 2, 3];
echo json_encode($v);
```

```php
<?php

$v = '[1, 2, 3]';
var_dump(json_decode($v));
```


pixiv

型チェッカーは常に愚かな我々に
先んじてプログラムのミスを
指摘してくれる… そうですよね？

pixiv

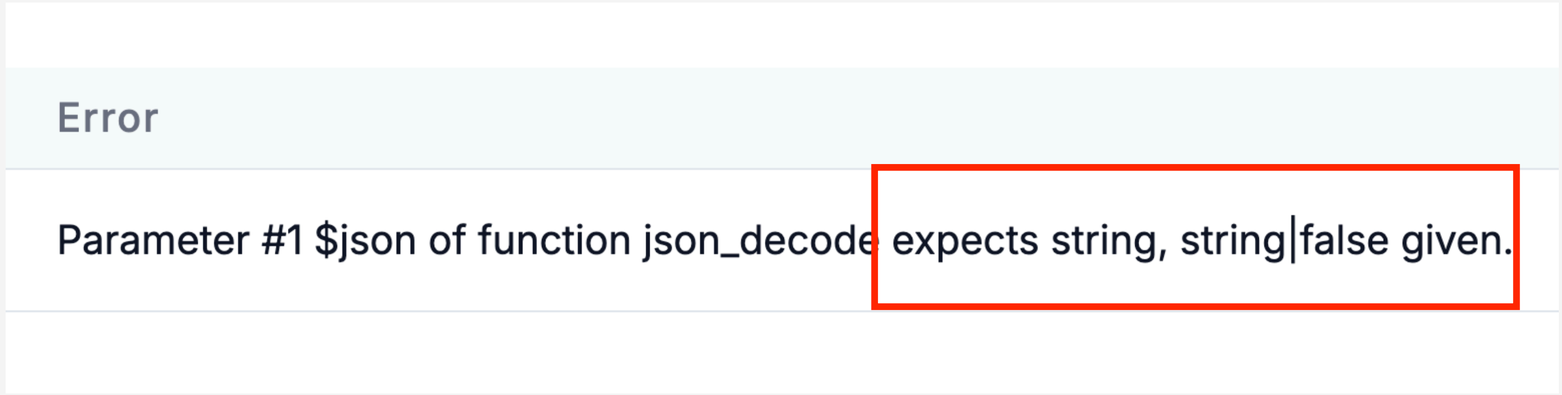
次のスライドのコードは
どうでしょうか？

```php
<?php

$v = file_get_contents('data.json');
var_dump(json_decode($v));
```

<?php

$v = file_get_contents('data.json');
var_dump(json_decode($v));



pixiv

Change language: Japanese

Submit a Pull Request Report a Bug

file_get_contents

(PHP 4 >= 4.3.0, PHP 5, PHP 7, PHP 8)

file_get_contents — ファイルの内容を全て文字列に読み込む

説明

```
file_get_contents(
    string $filename,
    bool $use_include_path = false,
    ?resource $context = null,
    int $offset = 0,
    ?int $length = null
): string|false
```

この関数は file() と似ていますが、 **offset** で指定した場所から開始し **length** バイト分だけ ファイルの内容を文字列に読み込むという点が異なります。失敗した場合、**file_get_contents()** は **false** を返します。

file_get_contents()はファイルの内容を文字列に読み込む 方法として好ましいものです。もしOSがサポートしていれば パフォーマンス向上のためにメモリマッピング技術が使用されます。

pixiv

どうしてこの関数は失敗したら
falseを返すのはなんで？？？

pixiv

A-1.
要求されたファイルが
正しく配置されなかった

pixiv

A-2.
動的に生成されるファイルを file_exists()でチェックせずに アクセスしてしまった

この手法は潜在的なリスクがある…

pixiv

```php
<?php

if (!file_exists('data.json')) {
    return;
}

// maybe safe
$v = file_get_contents('data.json');
var_dump(json_decode($v));
```

```php
<?php

if (!file_exists('data.json')) {
    return;
}

// maybe safe?
$v = file_get_contents('data.json');
var_dump(json_decode($v));
```

この関数がfalseを返すことはない… たぶん…
おそらく… きっと…

pixiv

```php
<?php

$v = file_get_contents('dat .json');

assert($v !== false);
var_dump(json_decode($v));
```

$vが絶対にfalseに
なっていないことを
アサーション(表明)

pixiv

```php
<?php

/** @var string */
$v = file_get_contents('data.json');

var_dump(json_decode($v));
```

実際にはfalseが返ってくる可能性を見て見ぬふり

pixiv

Q. ところで file_get_contents() ほんとにほんとにfalse返すの？

pixiv

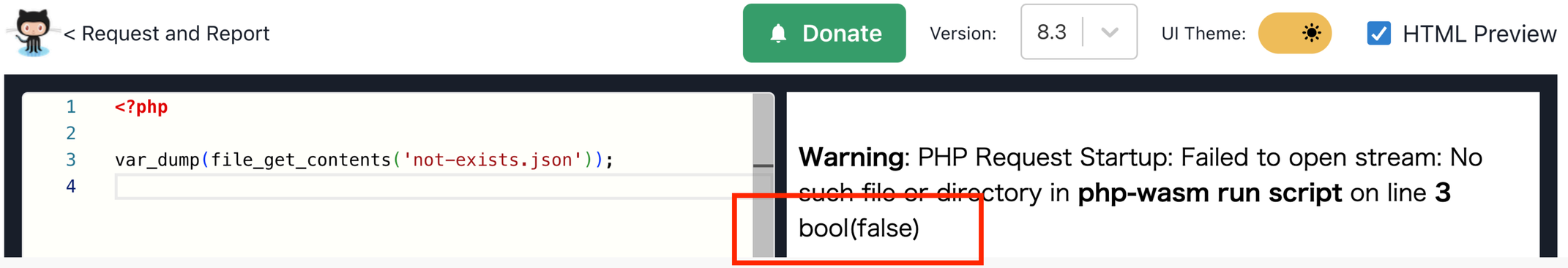
Q. 実行時のハンドラー設定に依存する

pixiv

PHP Playground

PHP Playground let you to execute basic PHP code in real time using WebAssembly technology.

About PHP Playground?

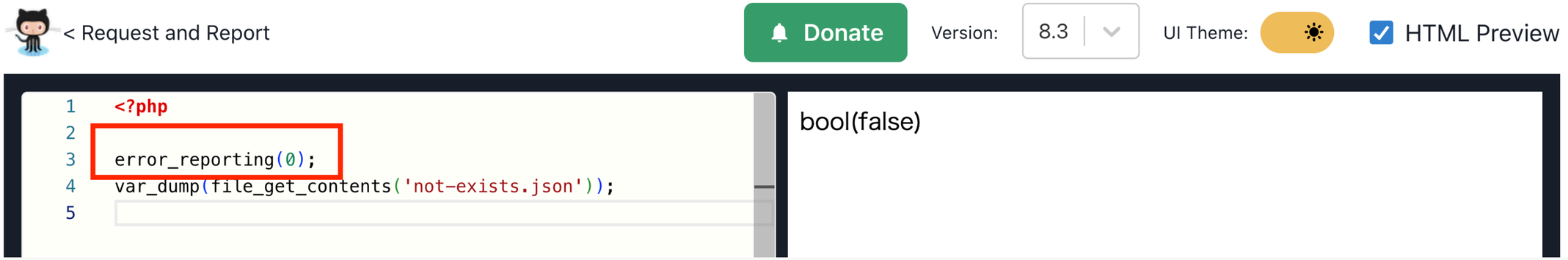


pixiv

PHP Playground

PHP Playground let you to execute basic PHP code in real time using WebAssembly technology.

About PHP Playground?



pixiv

PHP マニュアル › 言語リファレンス › 定義済みの例外 « Exce

Change language: Japanese

Submit a Pull Request Report a Bug

ErrorException

(PHP 5 >= 5.1.0, PHP 7, PHP 8)

はじめに

エラー例外です。

クラス概要

```
class ErrorException extends Exception {

   /* プロパティ */
   protected int $severity = E_ERROR;
```


pixiv

例

例1 set_error_handler() を使用した、エラーメッセージから ErrorException への変換

```php
<?php
function exception_error_handler(int $errno, string $errstr, string $errfile = null, int $errline) {
    if (!(error_reporting() & $errno)) {
        // このエラーコードが error_reporting に含まれていない場合
        return;
    }
    throw new \ErrorException($errstr, 0, $errno, $errfile, $errline);
}
set_error_handler(exception_error_handler(...));
// PHP 8.1.0 より前のバージョン、つまり 第一級callableを生成する記法 が実装される前は、下記の呼び出しが必要です
// set_error_handler(__NAMESPACE__ . "\\exception_error_handler");

/* 例外を発生させます */
strpos();
?>
```

PHP Playground

PHP Playground let you to execute basic PHP code in real time using WebAssembly technology.

About PHP Playground?

< Request and Report Donate Version: 8.3 UI Theme: HTML Preview

```php
<?php

set_error_handler(function (int $errno, string $errstr, string
    if (!(error_reporting() & $errno)) {
        // このエラーコードが error_reporting に含まれていない場合
        return;
    }
    throw new \ErrorException($errstr, 0, $errno, $errfile, $e
});

var_dump(file_get_contents('not-exists.json'));
```

Fatal error: Uncaught ErrorException: PHP Request Startup: Failed to open stream: No such file or directory in php-wasm run script:11 Stack trace: #0 [internal function]: {closure}(2, 'PHP Request Sta...', 'php-wasm run sc...', 11) #1 php-wasm run script(11): file_get_contents('not-exists.json') #2 {main} thrown in **php-wasm run script** on line **11**

pixiv

どういう状況が考えられるか
一度確認してみましょう

pixiv

file_get_contents() が返すのは...

- error_reporting(0) 設定のとき…
  - 関数は常に string と false のどちらも返す可能性がある
  - PHPの既定の振る舞いではファイルが読み込めなくても停止しない
- ハンドラでエラーをErrorExceptionに変換しているとき…
  - 関数は常に string 値を返す。falseが返ってくることはありえない。
  - 多くのPHPアプリケーションではフレームワークによって設定済み

```php
<?php

$v = file_get_contents('dat.json');

assert($v !== false);
var_dump(json_decode($v));
```

falseが返ってこないなら
この確認は常に無駄！

pixiv

ここからがタイトルのネタ

pixiv

“慈悲深いユニオン”とは何か

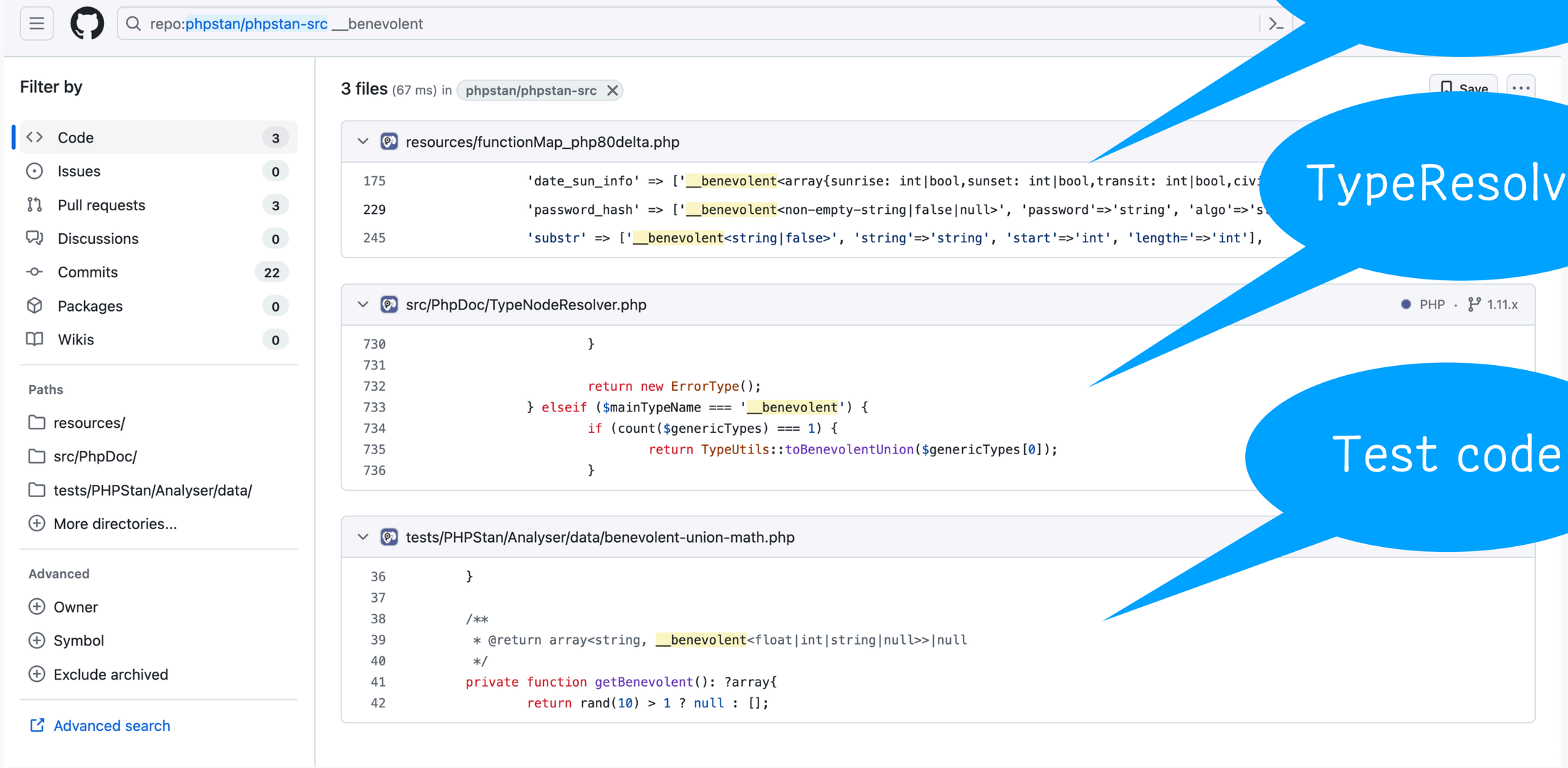
What's benevolent union?

pixiv Inc.
USAMI Kenta

pixiv

_benevolent<> は PHPStanに実装された隠し機能

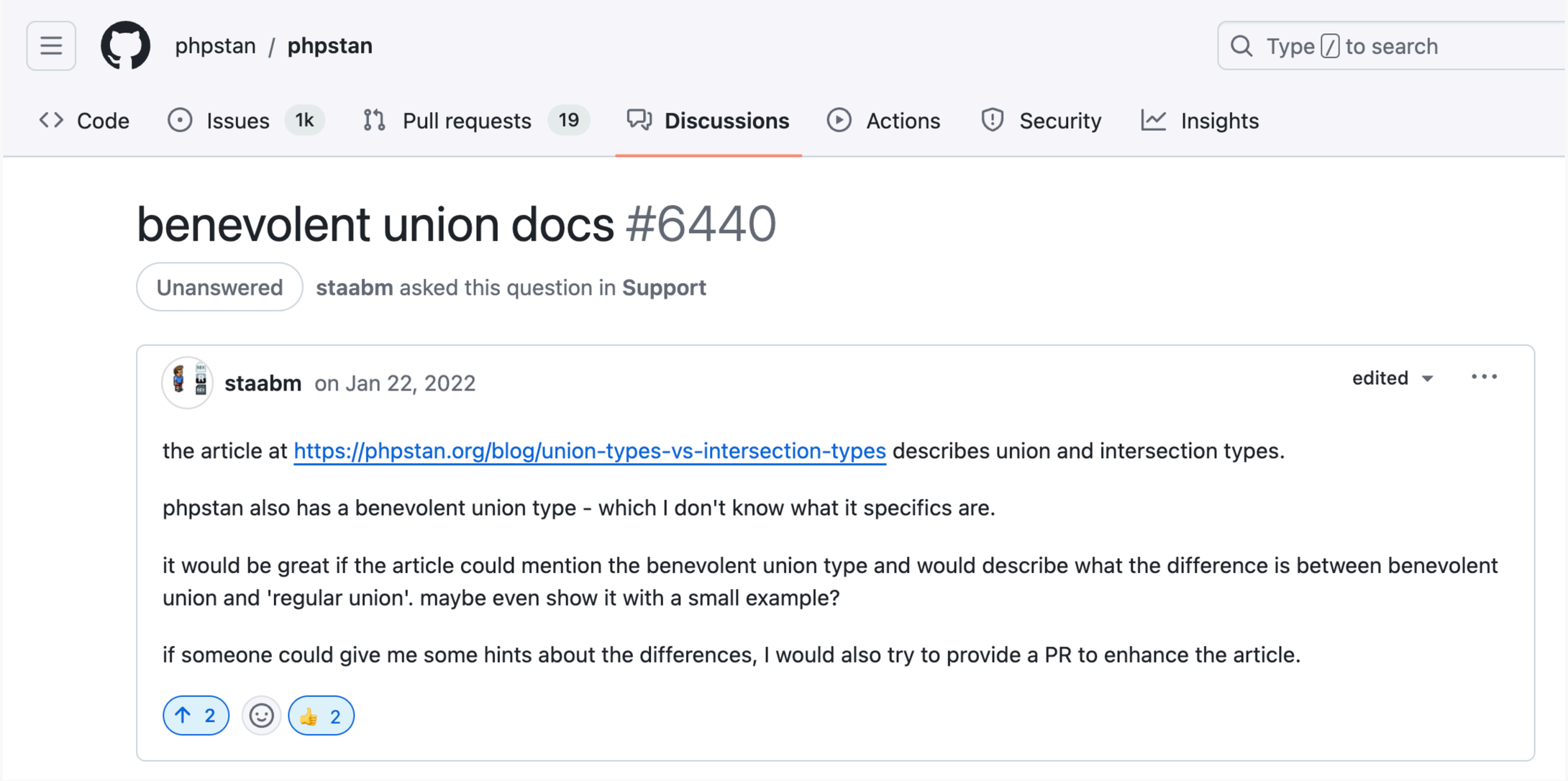
pixiv


repo:phpstan/phpstan-src __benevolent
Filter by
Code 3
Issues 0
Pull requests 3
Discussions 0
Commits 22
Packages 0
Wikis 0
Paths
resources/
src/PhpDoc/
tests/PHPStan/Analyser/data/
More directories...
Advanced
Owner
Symbol
Exclude archived
Advanced search
3 files (67 ms) in phpstan/phpstan-src
Save
resources/functionMap_php80delta.php
175 'date_sun_info' => ['__benevolent<array{sunrise: int|bool,sunset: int|bool,transit: int|bool,civi
229 'password_hash' => ['__benevolent<non-empty-string|false|null>', 'password'=>'string', 'algo'=>'s
245 'substr' => ['__benevolent<string|false>', 'string'=>'string', 'start'=>'int', 'length'=>'int'],
src/PhpDoc/TypeNodeResolver.php
PHP · 1.11.x
730 }
731
732 return new ErrorType();
733 } elseif ($mainTypeName === '__benevolent') {
734 if (count($genericTypes) === 1) {
735 return TypeUtils::toBenevolentUnion($genericTypes[0]);
736 }
tests/PHPStan/Analyser/data/benevolent-union-math.php
36 }
37
38 /**
39 * @return array<string, __benevolent<float|int|string|null>>|null
40 */
41 private function getBenevolent(): ?array{
42 return rand(10) > 1 ? null : [];
functionMap
TypeResolver
Test code

pixiv

で、結局なにこれ????

pixiv


phpstan / phpstan
Type / to search
Code
Issues 1k
Pull requests 19
Discussions
Actions
Security
Insights
benevolent union docs #6440
Unanswered
staabm asked this question in Support
staabm on Jan 22, 2022
edited
the article at https://phpstan.org/blog/union-types-vs-intersection-types describes union and intersection types.
phpstan also has a benevolent union type - which I don't know what it specifics are.
it would be great if the article could mention the benevolent union type and would describe what the difference is between benevolent union and 'regular union'. maybe even show it with a small example?
if someone could give me some hints about the differences, I would also try to provide a PR to enhance the article.
2
2

pixiv

2 comments · 7 replies

Oldest | Newest | Top

ondrejmirtes on Jan 22, 2022 Maintainer

PHPStan's BenevolentUnionType is mostly an implementation detail that's supposed to make some situations less annoying. It can't be expressed in a PHPDoc and it isn't notable to most users.

↑ 1

5 replies

pixiv

これらの関数の共通点を
探ってみましょう

Change language: English

Submit a Pull Request Report a Bug

password_hash

(PHP 5 >= 5.5.0, PHP 7, PHP 8)

password_hash — Creates a password hash

Description

```
password_hash(string $password, string|int|null $algo, array $options = []): string
```

password_hash() creates a new password hash using a s

Changelog

Version	Description
8.0.0	**password_hash()** no longer returns **false** on failure, instead a ValueError will be thrown if the password hashing algorithm is not valid, or an Error if the password hashing failed for an unknown error.
8.0.0	The **algo** parameter is nullable now.
7.4.0	The **algo** parameter expects a string now, but still accepts ints for backward compatibility.
7.4.0	The sodium extension provides an alternative implementation for Argon2 passwords.

Change language: English

Submit a Pull Request Report a Bug

substr

(PHP 4, PHP 5, PHP 7, PHP 8)

substr — Return part of a string

Description

```
substr(string $string, int $offset, ?int $length = null): string
```

Returns the portion of **`string`** specified by the **`offset`** and **`length`** parameters.

Changelog

Version	Description
8.0.0	**`length`** is nullable now. When **`length`** is explicitly set to **`null`**, the function returns a substring finishing at the end of the string, when it previously returned an empty string.
8.0.0	The function returns an empty string where it previously returned **`false`**.

Change language: English

Submit a Pull Request Report a Bug

date_sun_info

(PHP 5 >= 5.1.2, PHP 7, PHP 8)

date_sun_info — Returns an array with information about sunset/sunrise and twilight begin/end

Description

```
date_sun_info(int $timestamp, float $latitude, float $longitude): array
```

Return Values

Returns array on success or **`false`** on failure. The structure of the array is detailed in the following list:

.

__benevolent<>が使われた関数の共通点は？

- これらの関数は異常な状態でfalseを返す可能性がある
- 引数に適切なパラメータを渡して呼び出す限りfalseが返されることはないので、呼び出し後に毎回型検査をするのは不毛
- PHP 8でfalseを返す代りにErrorがthrowされるようになった

__benevolent<>って
結局どう動くの？

PHPStan

Home | Try | Documentation | Blog | Search ⌘ K

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

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

/** @return string|false */
function strOrFalse() { return ''; } // @phpstan-ignore-line

/** @return __benevolent<string|false> */
function benevolentStrOrFalse(): string|false { return ''; } // @phpstan-ignore-line

$_ = strlen(strOrFalse());
$_ = strlen(benevolentStrOrFalse());

```

Share | Level 9 | Strict rules | Bleeding edge | Treat PHPDoc types as certain

PHP 8.0 – 8.3 (1 error) | PHP 7.2 – 7.4 (2 errors)

Line	Error	
9	Parameter #1 $string of function strlen expects string, string|false given.	argument.type

この型を使えば標準関数を
使いやすくするラッパーも作れる

PHPStan

Home | Try | Documentation | Blog | Search ⌘ K

```php
1  <?php declare(strict_types = 1);
2
3  /** @return __benevolent<string|false> */
4  function getContents(string $path) {
5      return file_get_contents($path);
6  }
7
8  $_ = json_decode(file_get_contents('data.json'));
9  $_ = json_decode(getContents('data.json'));
10
```

Share | Level 9 | ☐ Strict rules | ☑ Bleeding edge | ☑ Treat PHPDoc types as certain

Found 1 error

Line	Error	
8	Parameter #1 $json of function json_decode expects string, string|false given.	argument.type

pixiv

便利っぽい…？
んだけど、使わないでね！

benevolent unionを濫用するな

- この型は、例外的な状態でfalseを返すまれなケースを表現するのに役立つ
  - その関数が間違った前提条件で呼び出された場合は、false を返す代わりに例外を発生させ、呼び出しコードを修正する。
  - その他の例外的な状況では、例外をスローして呼び出し元に処理させるべき。Javaのチェック例外みたいに。

安全性を保ち
コードの繰り返しを防ぐには？

README MIT license

stable v2.5.0 downloads 47.07 M unstable 8.1.x-dev license MIT build passing Continuous Integration failing codecov 49%

Safe PHP

A set of core PHP functions rewritten to throw exceptions instead of returning `false` when an error is encountered.

The problem

Most PHP core functions were written before exception handling was added to the language. Therefore, most PHP functions do not throw exceptions. Instead, they return `false` in case of error.

But most of us are too lazy to check explicitly for every single return of every core PHP function.

```
// This code is incorrect. Twice.
// "file_get_contents" can return false if the file does not exist
// "json_decode" can return false if the file content is not valid JSON
$content = file_get_contents('foobar.json');
$foobar = json_decode($content);
```

Safe PHP

A set of core PHP functions rewritten to throw exceptions instead of returning `false` when an error is encountered.

The problem

Most PHP core functions were written before exception handling was added to the language. Therefore, most PHP functions do not throw exceptions. Instead, they return `false` in case of error.

But most of us are too lazy to check explicitly for every single return of every core PHP function.

```
// This code is incorrect. Twice.
// "file_get_contents" can return false if the file does not exist
// "json_decode" can return false if the file content is not valid JSON
$content = file_get_contents('foobar.json');
$foobar = json_decode($content);
```

The correct version of this code would be:

```
$content = file_get_contents('foobar.json');
if ($content === false) {
    throw new FileLoadingException('Could not load file foobar.json');
}
$foobar = json_decode($content);
if (json_last_error() !== JSON_ERROR_NONE) {
    throw new FileLoadingException('foobar.json does not contain valid JSON: '.json_l
}
```

Obviously, while this snippet is correct, it is less easy to read.

The solution

Enter *thecodingmachine/safe* aka Safe-PHP.

Safe-PHP redeclares all core PHP functions. The new PHP functions act exactly as the old ones, except they throw exceptions properly when an error is encountered. The "safe" functions have the same name as the core PHP functions, except they are in the `Safe` namespace.

```php
use function Safe\file_get_contents;
use function Safe\json_decode;

// This code is both safe and simple!
$content = file_get_contents('foobar.json');
$foobar = json_decode($content);
```

All PHP functions that can return `false` on error are part of Safe. In addition, Safe also provide 2 'Safe' classes: `Safe\DateTime` and `Safe\DateTimeImmutable` whose methods will throw exceptions instead of returning false.

この関数はFileSystemExceptionをthrowsする

```
260      * @throws FilesystemException
261      *
262      */
263     function file_get_contents(string $filename, bool $use_include_path = false, $context = null, int $offs
264     {
265         error_clear_last();
266         if ($length !== null) {
267             $safeResult = \file_get_contents($filename, $use_include_path, $context, $offset, $length);
268         } elseif ($offset !== 0) {
269             $safeResult = \file_get_contents($filename, $use_include_path, $context, $offset);
270         } elseif ($context !== null) {
271             $safeResult = \file_get_contents($filename, $use_include_path, $context);
272         } else {
273             $safeResult = \file_get_contents($filename, $use_include_path);
274         }
275         if ($safeResult === false) {
276             throw FilesystemException::createFromPhpError();
277         }
278         return $safeResult;
279     }
```


pixiv

PhpStormはハンドリングしない例外を警告する

```php
<?php

use function Safe\file_get_contents;

function f(string $path): void
{
    echo file_get_contents($path);
}
```

Unhandled Safe\Exceptions\FilesystemException

Add PHPDoc comment with @throws tags ⌥⇧↵ More actions... ⌥↵

pixiv

Bring your exceptions under control with @throws

May 12, 2021 · 8 min read

The most common detected bugs ••• by PHPStan are:

- Calling unknown method on an object
- Accessing unknown property on an object
- Passing wrong types of arguments to methods and functions

Static analysers put `@param` and `@return` PHPDoc validation on the map, people have been benefiting from those errors being reported for the past 4,5 years, and fixed a lot of typehints in their own and third party codebases on the way.

But one aspect of the PHP language had been flying under the radar this whole time. Exceptions can serve you well, but they can also wreak havoc in your codebase. Similarly to the PHP landscape before static analysis came to it, the flow of exceptions throughout applications is still largely unchecked. Developers forget to handle error states, catch expected exceptions, and that leads to applications crashing in production.

benevolent unionよりも
構造化された例外を！！

pixiv

みんなのコードでは
benevolent unionsを
使わないでね！