PHP 8.x 時代のクラス設計
property promotion から property hooks まで

nsfisis（いまむら）

PHP カンファレンス小田原 2025

いまむら

nsfisis

@デジタルサーカス株式会社

今回の話の対象

PHP 8 系で追加された機能

PHP 8.x 時代のクラス設計

今回の話の対象

PHP 8.x で追加された機能

PHP 8.x 時代のクラス設計

- アプリケーションコード
- フレームワークの規約に縛られない

今回の話の対象

PHP 8.x で追加された機能

PHP 8.x 時代のクラス設計

PHP 8.0

アトリビュート

```
class FooTest extends TestCase {
    #[Test]
    function testBar(): void { … }
}
```

コンストラクタプロパティ昇格

```
class Point {
    function __construct(
        public int $x,
        public int $y,
    ) {}
}
```

Union 型

```
function f(int|string $x)
{
    …
}
```

Mixed 型

```
function f(mixed $x)
{
   …
}
```

PHP 8.1

交差型

```
function f(Foo&Bar $x)
{
   …
}
```

読み取り専用プロパティ

```
class C {
    public readonly int $x;
}
```

Final クラス定数

```
class C {
    final const FOO = 1;
}
```

PHP 8.2

読み取り専用クラス

```
readonly class C {
    public int $x;
    public int $y;
}
```

動的なプロパティの非推奨化

```
class C { … }
$c = new C();
$c->aaaaaaaaaaa = 123;
// => Deprecated: Creation of
       dynamic property C::$y
       is deprecated
```

PHP 8.3

型付けされたクラス定数

```
class C {
    const int FOO = 123;
}
```

Override アトリビュート

```
class C {
    function f() { … }
}
class D extends C {
    #[\Override]
    function f() { … }
}
```

PHP 8.4

Final プロパティ

```
class C {
    final int $x = 0;
}
```

非対称可視性

```
class C {
    public private(set) int $x;
}
```

プロパティフック

```
class User {
    …
    public string $username {
        get => $this->username;
        set {
            if ($value === "") throw …;
            $this->username = $value;
        }
    }
}
```

プロパティフック

```
class Complex {
    public function __construct(
        public readonly float $r,
        public readonly float $i,
    ) {}
    public float $abs {
        get => sqrt($this->r ** 2 + $this->i ** 2);
    }
}
```

今回の話の対象

PHP 8.x で追加された機能

PHP 8.x 時代のクラス設計

クラスは原則 final

クラスは原則 final

- 継承を設計するのは難しい
- Non-final にはすぐできる
- 作った時点で継承する予定でなかったクラスを
  継承したくなったなら初期設計が破綻したサイン

クラスは原則 readonly

クラスは原則 readonly

- ボイラープレートを減らす
  - getter あり / setter なし
- 可変な状態を減らす

すべてのプロパティを定義する

すべてのプロパティを定義する

- 動的プロパティは非推奨
- 可変な状態を減らす
- 未知の状態を減らす

すべてに型を付ける

すべてに型を付ける

- 取りうる状態を減らす
- 補完が強力になる
- 明らかな誤りを検知できる
- 依存ライブラリのアップデートに強くなる

オーバーライドしている
メソッドすべてに
#[\Override] をつける

オーバーライドしているメソッドすべてに #[\Override] をつける

- 明らかな誤りを検知できる
- 依存ライブラリのアップデートに強くなる

プロパティフック考

プロパティフック考

- Readonly クラスと組み合わせられない
  - プロパティフックに readonly を指定できない
- 「プロパティフックに期待される振る舞い」が固まっていない

プロパティフック考

Kotlin の公式コーディング規約より引用

Prefer a property over a function when the underlying algorithm:

- *Does not throw.*
- *Is cheap to calculate (or cached on the first run).*
- *Returns the same result over invocations if the object state hasn't changed.*

PHP 8.x 時代のクラス設計

- final: 継承を opt-in に
- readonly: 状態を不変に
- プロパティ定義: 未知の状態を作らせない
- 型宣言: 取りうる状態の数を減らす
- Override: アップデートに強く
- プロパティフック: 今後の動向次第

  （似た機能を持つ他言語に学ぶ）