PHP Best Practices

MONTREAL | FEB 26, 2020

@afilina

zenika
<animés par la passion>

Anna Filina

- Coding since 1997.
- PHP since 2003.
- Legacy archaeologist.
- Test automation.
- Talks and workshops.
- YouTube videos.


zenika
<led by passion>

What Is This About?

- Not talking about nice-to-haves.
- Not talking about style.
- Real bugs as justification for each point.

```php
$sql = 'SELECT * FROM users WHERE email="'.$email.'"';
$this->pdo->query($sql);

// abc" OR 1=1 OR email="abc
// SELECT * FROM users WHERE email="abc" OR 1=1 OR
email="abc";

$sql = 'SELECT * FROM users WHERE email = :email';
$statement = $this->pdo->prepare($sql);
$statement->execute([':email' => $email]);
```

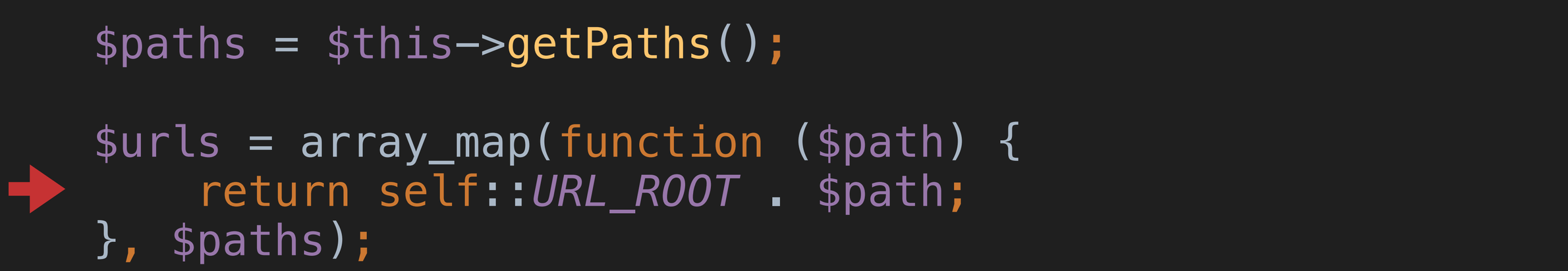
```php
$paths = $this->getPaths();

$urls = array_map(function ($path) {
    return self::URL_ROOT . $path;
}, $paths);
```

array_map(): Expected parameter 2 to be an array, string given

```
private function getPaths()
{
    return '[
        {"list_products":"/products"},
        {"view_cart":"/cart"},
    ]';
}
```

```php
private function getPaths(): array
{
    return [
        $this->path1,
        $this->path2,
    ];
}
```



Array to string conversion

```
private $path1 = ['list_products' => '/products'];
private $path2 = ['view_cart' => '/cart'];
```

```
private $path1 = '/products';
private $path2 = '/cart';
```

```php
$urls = array_map(function (string $path) {
    return self::URL_ROOT . $path;
}, $paths);
```

```php
/**
 * @return array<int, string>
 */
private function getPaths(): array
```

```
composer require --dev vimeo/psalm
vendor/bin/psalm --init
Detected level 7 as a suitable initial default
vendor/bin/psalm src
```

```php
/**
 * @return array<int, string>
 */
private function getPaths(): array
```

ERROR: MixedReturnTypeCoercion - src/TypeMismatch.php:65:16 - The type 'array{0: mixed, 1: mixed}' is more general than the declared return type 'array<int, string>' ...

```php
if ($path === '') {
    throw new InvalidArgumentException('Is blank');
}
```

```
function (Path $path) {
    return self::URL_ROOT . $path;
}, $paths);
```

```php
final class Path
{
    private string $path;

    public function __construct(string $path)
    {
        Assert::that($path)
            ->notBlank();
        $this->path = $path;
    }

    public function __toString(): string
    {
        return $this->path;
    }
}
```

```
array_map(function (Path $path) {
    return self::URL_ROOT . $path;
}, $paths);
```

```php
/**
 * @return array<int, Path>
 */
private function getPaths(): array
{
    return [
        new Path('/products'),
        new Path('/cart'),
    ];
}
```

```php
public function setPrice(int $price)
{
    $this->price = $price;
}

$this->setPrice(1.15);
```

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

TypeError : Argument 1 passed to MyClass::setPrice() must be of the type int, float given

Strict types.
Strict types.
Strict types.

```
private string $stringPath;
private Path $voPath;
```

NPEs galore.

```php
class Product
{
    public $name;
}

//...

$this->findByName($this->product->name);
```

```
TypeError : Argument 1 passed to MyClass::findByName()
must be of the type string, null given
```

```php
class Product
{
    public string $name;
}
```

```php
class ProductEntity
{
    public $name;
    public $price;
}

final class Product
{
    public string $name;
    public int $price;
    //...
}
```

```php
return new Product(
    $product->name,
    $product->price
);
```

```php
if ($product->getLastPrice() !== null) {
    return number_format($product->getLastPrice());
}
```

```
TypeError : number_format() expects parameter 1 to be
float, null given
```

```php
public function getLastPrice()
{
    return array_pop($this->prices);
}
```

```php
$lastPrice = $product->getLastPrice();
if ($lastPrice !== null) {
    return number_format($lastPrice);
}
```

```php
@$array[$foo->a()];

public function a()
{
    trigger_error('my error', E_USER_ERROR);
}

$array[$foo->a()] ?? 'something else';
```

```php
interface ApiAware
{
    public function setApi(Api $api);
}

if ($class instanceof ApiAware) {
    $class->setApi($api);
}
```

```php
final class MyClass implements ApiAware
{
    private $api;

    public function setApi(Api $api): void
    {
        $this->api = $api;
    }

    public function sendApiRequest()
    {
        $product = new Product();
        $this->api->sendRequest($product);
    }
}
```

Error : Call to a member function sendRequest() on null

```php
final class MyClass
{
    private Api $api;

    public function __construct(Api $api)
    {
        $this->api = $api;
    }

    public function sendApiRequest()
    {
        $product = new Product();
        $this->api->sendRequest($product);
    }
}
```

Dependency injection is your friend.

```php
if (!empty($array)) {
    return $array[0];
}
```

Trying to access array offset on value of type bool

```php
if (empty($airplaneSeat)) {
    $this->book($airplaneSeat);
}
```

```php
empty("");
empty(0);
empty(0.0);
empty("0");
empty(null);
empty(false);
empty(array());
```

```
!== ""
!== 0
!== 0.0
!== null
=== true
count(array())
```

```
0.99 + 0.01 === 1
```

IEEE 754
floating point arithmetic.

```
$amountInCents + 1
```

```php
/** @var PaymentGatewayInterface */
$gateway = $this->getSelectedGateway();
$gateway->preauthorizePayment();
```

How to test?

```php
$soapApi = new SoapApi($wsdl, $soapKey, $config);

public function __construct(SoapApi $soap)
{
    $this->soap = $soap;
}
```

```
class Order
{
    public function getProducts()
    {
        return Product::find($this->productIds);
    }
}

Product::shouldReceive('find')
    ->once()
    ->andReturn([]);
```

```php
class MyController extends AbstractController
{
    public function myAction()
    {
        $doctrine = $this->container->get('doctrine');
    }
}
```

Dependency injection is your friend.

THANKS!

zenika
<animés par la passion>