使ってはいけない
phpメタプログラミング 初級篇

Dangerous PHP Meta programming Tutorial

2017-08-30 PHP勉強会@東京 #phpstudy

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id: zonuexe
- Emacs Lisper, PHPer
  - pixiv Inc.でpixivを作ってるよ
  - Qiitaに記事を書いたり変なコメントしてるよ
- 動的言語はコンピューティングの真髄

pixiv

はじめに

サンプルコードは言語機能を簡易に紹介するためのものです

利用の際は必要性と利便性
を慎重に判断すること。

アジェンダ

0. 前提

1. 初級篇

2. 中級篇

3. 上級篇

0. 前提

1. 初級篇

2. ~~中級篇~~

3. ~~上級篇~~

phpにおける
メタプログラミングの温床

Meta programming for PHP

2017-08-20 NEEC Kamata #ll2017jp
ODC 2017 Tokyo/ Learn Languages (LL2017)

メタプロ

グラミング

非常に雑にまとめる
と、一見して意味の
わかりにくい謎テク

前提

> メタプログラミング(metaprogramming)とはプログラミング技法の一種で、ロジックを直接コーディングするのではなく、あるパターンをもったロジックを生成する高位ロジックによってプログラミングを行う方法、またその高位ロジックを定義する方法のこと。

https://ja.wikipedia.org/wiki/メタプログラミング

“

直接コーディングするのではなく

”

https://ja.wikipedia.org/wiki/メタプログラミング

突き詰めていくと
関数呼び出しすら
「直接コーディング」
しないと解釈できる

それよりはちょっと
複雑なものが
メタプログラミング

PHPは、所謂LLと
しては遊びが少ない

例）原則として関数や
クラスを再定義できない
（致命的エラーが発生）

RubyやPythonと違って
オープンクラスではない
（モンキーパッチができない）

謎なタイミングで
クラス定義が変更される
ことが(基本的には)ない

= 静的解析しやすい
条件が揃ってる

変数スコープが（比較的）シンプル

その和を乱す~~邪悪な~~
動的メタプログラミング
テクニックが存在する

「一見してわかりにくい言語機能」

「ユーザー定義関数では不可能な
マジックを実現してくる謎関数」

初級編

callable

関数として呼出し可
能な値を表す擬似型

map

```
$ary = ["apple", "orange", "banana"];
//=> ["Apple", "Orange", "Banana"]
```

```
// 配列の中身を一個一個変更する

$bry = [];
foreach ($ary as $i => $a) {
    $bry[$i] = ucfirst($a);
}
```

map

```
$ary = ["apple", "orange", "banana"];
//=> ["Apple", "Orange", "Banana"]
```

```
// 配列の中身を一個一個変更する
$cry = array_map('ucfirst', $ary);
```

map

```
$ary = ["apple", "orange", "banana"];
//=> ["Apple", "Orange", "Banana"]
```

```
$bry = [];
foreach ($ary as $i => $a) {
    $bry[$i] => ucfirst($a);
}
```

```
$cry =
array_map('ucfirst', $ary);
```

callable

```
// Before
if ($cond) {
    foo($var);
} else {
    bar($var);
}
```

callable

```
// After
$f = $cond ? 'foo' : 'bar';
call_user_func($f, $var);


// こうやって書いてもいい
$f($var);
```

callable

```php
// Before
if ($cond) {
    foo($var);
} else {
    bar($var);
}
```

```php
// After
$f = $cond ? 'foo' : 'bar';
call_user_func($f, $var);

// こうやって書いてもいい
$f($var);
```

使ってはいけない度

★★☆☆☆

僕はコードレビューで「foreachの方が読みやすい」ってマジレスすることもある

（★が多いほど邪悪）

（メタプログラミングと呼ぶかは議論がある）

PHPの「ふつう」はひとによって境界が異なる

compact()

extract()

変数テーブルにアクセスできる関数

compact()

変数テーブルから
取り出して配列にする

compact()

```
$foo = foo();
$bar = bar();
```

```
hoge([
    'foo' => $foo,
    'bar' => $bar
]);
```

```
hoge(compact('foo','bar'));
```

使ってはいけない度

★☆☆☆☆

何が起こるか把握して

書くぶんには罠が少ない

（★が多いほど邪悪）

extract()
変数テーブルから
取り出して配列に

extract()

```
function hoge(array $vs)
{
```

```
$hoge_foo = $vs['foo'];
$hoge_bar = $vs['bar'];
```

```
extract($vs,
        EXTR_PREFIX_ALL,
        'hoge');
```

本当はissetでチェックすべき

使ってはいけない度

何が起こるか把握して

書くぶんには罠が少ない

（★が多いほど邪悪）

extract()

変数定義が検出できない

読みにくくなるので邪悪

バグの温床であるばかりか、

ふつうに脆弱性の温床になる

$$var

$$var
可変変数
(variable variables)

可変変数

```php
$world = "こんにちは世界";
$hello = "world";
$v = "hello";

var_dump($v);    //=> "hello"
var_dump($$v);   //=> "world"
var_dump($$$v);  //=> "こんにちは世界"
// $$ はいくつ付けてもよい！！
```

使ってはいけない度

★★★★★

純粋に読みにくい……。error_reporing のレベルが低いと警告なしでnullになる。

（★が多いほど邪悪）

debug_backtrace()

スタックトレースを
全部取得できる
（ファイル・引数含む）

```
Baguette\FriendsClass\CalledByOutsiderException: It is not allowed for
Baguette\Ore to call  in ~/friends/src/functions.php on line 35

Call Stack:
    0.0003     359288   1. {main}() ~/friends/tests/test.php:0
    0.0020     552824   2. Baguette\Omae->callFriend() ~/friends/tests/
test.php:33
    0.0020     552864   3. Baguette\Ore->__construct() ~/friends/tests/
test.php:20
    0.0020     552864   4. Baguette\assert_called_by_friend() ~/friends/
tests/test.php:12
```

（これを悪用すれば
どんな邪悪なことが
できるかは明白）

超べんりな用途（ロギング）

```
function db($query, array $param) {
    $trace = implode('; ', array_map(function ($t) {
        return "{$t['file']}({$t['line']})";
    }, debug_backtrace()));

    $stmt = PDO::query($query.' -- '.$trace);
    // SQLスロークエリログの末尾にトレースが
    // 残るので、原因の特定しやすくなる！
```

邪悪な用途

```
function hoge($a, $b, $c) {
    $trace = debug_backtrace(true, 2);
    $class = $trace[1]['class'] ?? false;
    $func  = $trace[1]['func'] ?? false;
    if ($class === "Foo" && $func === "bar") {
        // Foo::bar() はバグってるので値を上書き
        $b = "piyo";
    }
```

使ってはいけない度

★★★☆☆

ロギングは問題なし。

ワークアラウンドはギルティ。

（★が多いほど邪悪）

まとめ

今回は初級編

プログラミングのおもしろテクを知ったら使ってみたくなるのが人情

大いなる力には
大いなる責任が伴う

メタプログラミング
の採用で劇的に便利
になることもある

単に初心者殺しな
意味不明なコードに
なることがある

（経験豊富な先輩に「若いなあ」となまあたたかく見られるかも）

用法容量を守って
たのしい
プログラミングを

To Be Continued