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)

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id: zonuexe
- Emacs Lisper, PHPer
- 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

世の中にはいろんな

クールな言語がある

PHPには毛ほども興味のなさそうな皆さまにPHPの動的な言語機能を紹介します

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。
（メタプログラミングって基本的
にそーゆーものですけど…）

アジェンダ

0. 前提

1. 初級篇

2. 中級篇

3. 上級篇

前提

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

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

初級編

callable

関数として呼出し可
能な値を表す擬似型

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);
```

map

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

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

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

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'));
```

extract()
変数テーブルから
取り出して配列に

extract()

```
function (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

extract()

未使用変数の検出できない

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！！
```

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）

（これを悪用すればどんな邪悪なことができるかは明白なので省略）

中級編

マジック

メソッド

```
__get($name)
__set($name, $value)
```

__get()
アクセス不可プロパティ
を読み出そうとしたとき
に呼ばれる

__set()
アクセス不可プロパティ
を書き込もうとしたとき
に呼ばれる

__get(), __set()

```
class Hoge {
    private $values = [];
    public function __set($name, $value) {
        $values[$name] = $value;
    }
    public function __get($name) {
        return $values[$name];
    }
}
```

「アクセス不可」は未定義プロパティだけはなく

private, protected含む

読み出し専用プロパティ

```
/**
 * @property-read string $name
 */
class Hoge {
    private $name;
    public function __construct($name) {
        $this->name = $name;
    }
    public function __get($name) {
        return $values[$name];
    }
```

書き込み禁止プロパティ

```
    public function __set($name) {
        throw new OutOfRangeException();
    }
}
```

```
__call($name, $args)
```

__call()
アクセス不可メソッドを
起動しようとしたときに
呼ばれる

Proxyパターン

```
class HogeProxy {
    private $hoge;
    public function __construct(Hoge $hoge) {
        $this->hoge = $hoge;
    }
    public function __call($name, $args) {
        return call_user_func_array($name, $args);
    }
}
```

Proxyパターン（PHPDoc）

```
/**
 * @method Foo getFoo()
 * @method string format(string $tpl)
 * @method void close()
 */
class HogeProxy {
    /** @var Hoge */
    private $hoge;
    public function __construct(Hoge $hoge) {
        $this->hoge = $hoge;
    }
    public function __call($name, $args) {
```

上級編

spl_autoload_register()

クラスが定義されたファイルを自動ロードするための仕組み

(ふつうはComposerがやってくれるので自分で定義する必要ない)

ふつうのクラスローダー

```
spl_autoload_register(function ($class) {
    $file = strtr($class, '\\', '/') . '.php';
    $path = __DIR__ . '/' . $file;
    if (file_exists($path)) {
        require $path;
    }
});
```

クラスが読み込まれる前に介入できる

邪悪クラスローダー

```
if (file_exists($path)) {
    $evil_path = evil_path($path);
    if (file_exists($evil_path)) {
        $src = file_get_contents($path);
        // ファイルを魔改造
        $evil_src = evil_customize($src);
        file_put_contents($evil_path, $evil_src);
    }
    require $evil_path;
}
```

まとめ

PHPは制限の大い言語のようだが、遊べるポイントはいろいろある

大いなる力には

大いなる責任がry

PHPUnit, PsySH, そのほか
一部のフレームワークなどは
メタプログラミングのテクニッ
クを活用してる

それ用のライブラリ

もある。

(Go! AOP https://github.com/goaop/framework)

実用知識！

そなえよう