諸君とPHPと一級市民

First-class citizen in PHP with you

pixiv

2017-10-24 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
- ピクシブ株式会社技術基盤チーム（pixiv.net）
- Emacs Lisper, PHPer
  - 入社前は自宅警備をしながらRuby書いてた
  - Emacs PHP Modeのメンテナ引き継ぎました
- Qiitaに記事を書いたり変なコメントしてるよ

pixiv

はじめに

プログラムは
様々な要素の
組み合せで構成される

様々な要素？

PHPにおいては(ざっくり)、
構文(文法), 数字, 文字列, 真理値,
配列, リソース, 型, クラス, 関数,
オブジェクト, インターフェイス,
トレイト, ファイル, 外部API,
データベース, HTMLテンプレート
…etc.

われわれは
いろんなものを
組み合せて作る

pixiv

どうやって？

pixiv

いろんなものが現れる

```
<?php require 'bootstrap.php';
$db = mysqli_connect();
$stmt = mysqli_prepare($db, 'SELECT * FROM books');
mysqli_stmt_execute($stmt);
```

```
print '<h1>本の一覧</h1>';
print '<ul>';
foreach ($stmt as $s){
    print '<li>'; print $s->name; print  ‘</li>';
}
print '</ul>';
```

お気付きか

階級制度

いろんな
ものがある

ウィキペディア
フリー百科事典

メインページ
コミュニティ・ポータル
最近の出来事
新しいページ
最近の更新
おまかせ表示
練習用ページ
アップロード (ウィキメディア・コモンズ)

ヘルプ

ヘルプ
井戸端
お知らせ
バグの報告
寄付
ウィキペディアに関するお問い合わせ

印刷/書き出し

ブックの新規作成
PDF 形式でダウンロード

第一級オブジェクト

第一級オブジェクト（**ファーストクラスオブジェクト**、first-class object）は、あるプログラミング言語において、たとえば生成、代入、演算、（引数・戻り値としての）受け渡しといったその言語における基本的な操作を制限なしに使用できる対象のことである。ここで「オブジェクト」とは広く対象物・客体を意味し、必ずしもオブジェクト指向プログラミングにおけるオブジェクトを意味しない。第一級オブジェクトは「**第一級データ型**に属す」という。

この言葉は1960年代にChristopher Straceyによって「functions as first-class citizens」という文脈で初めて使われた。

言語によって異なるが、第一級オブジェクトは概ね次のような性質をもつ。

- 無名のリテラルとして表現可能である。
- 変数に格納可能である。
- データ構造に格納可能である。
- それ自体が独自に存在できる（名前とは独立している）。
- 他のものとの等値性の比較が可能である。
- プロシージャや関数のパラメータとして渡すことができる。
- プロシージャや関数の戻り値として返すことができる。
- 実行時に構築可能である。
- 表示可能である。
- 読み込むことができる。
- 分散したプロセス間で転送することができる。
- 実行中のプロセスの外に保存することができる。

“**第一級オブジェクト**（ファーストクラスオブジェクト、*first-class object*）は、あるプログラミング言語において、たとえば生成、代入、演算、（引数・戻り値としての）受け渡しといったその言語における基本的な操作を制限なしに使用できる対象のことである。ここで「オブジェクト」とは広く対象物・客体を意味し、必ずしもオブジェクト指向プログラミングにおけるオブジェクトを意味しない。”

–第一級オブジェクト (最終更新 *2017-04-23 19:00:11*)

https://ja.wikipedia.org/wiki/%E7%AC%AC%E4%B8%80%E7%B4%9A%E3%82%AA%E3%83%96%E3%82%B8%E3%82%A7%E3%82%AF%E3%83%88

PHPは

どうかな？

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

PHPは

どうかな？

明らかに一級市民ではないもの

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

構文

ifをforに変換する…

といったことは

できない

神

ココ→

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

string

int

string（文字列）

- $s = "string";
- echoとかで簡単に画面に出力できる
- $a['str'] みたいな感じでキーになれる
- あらゆるバイナリ列を格納できる
  - 例: 画像ファイルの中身

int（整数）

- $i = 12345;
- echoとかで簡単に画面に出力できる
- $a[0] みたいな感じでキーになれる
- PHP_INT_MINからPHP_INT_MAXの範囲のみ扱える（CPUのアーキテクチャ32bit, 64bitに依存）

上級市民

ココ→

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

float（浮動小数点数）

- $f =1.1; $l = PHP_INT_MAX + 1;
- echoとかで簡単に画面に出力できる
- 絶対に配列のキーにしてはいけない
  - エラーにもならずintに化ける
  - 基本的に等値比較(==, ===)
してはいけない

bool（真理値）

- $t = true; $f = (1 == 2);
- そのままechoなどで出力できない
  - trueは1に化ける
  - falseは何も出力されない
- 配列キーにすると

結果を一時変数に保存する

```php
<?php

if ($i == 1) echo '$iは1です';
```

```php
$is_success = do_something();
if ($is_success) {
    echo "成功";
} else {
    echo "しっぱい。。。";
}
```

結果を返す関数

```php
<?php

function is_valid_keyword($key)
{
    return in_array($key, ['public', 'private']);
}
```

シリアライズ
できる

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
ココ → 上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

array（配列）

- ```
  $a = array('a', 'b', 'c');
  $b = range(1, 10);
  $c = ['name' => $n , 'age' => $y];
  ```
- ほかの言語の配列や辞書(ハッシュ，連想配列)とか呼ばれるもの
- 内容にほかの型の値を含められる
  ```

object（オブジェクト）

- $dt = new DateTime('2112-09-03');
  $o = (object)['name' => $n , 'age' => $y];
- クラスを実体化したもの
  キャストするとstdClassになる
- 内容にほかの型の値を含められる

シリアライズ

```php
<?php
$s = serialize('string');
// "s:6:"string";"
$a = serialize(['apple', 'orange']);
//=> "a:2:{i:0;s:5:"apple";i:1;s:6:"orange";}"

$obj = new stdClass;
$obj->name = 'ミク'; $obj->age = 16;
//=> "O:8:"stdClass":2:{s:4:"name";s:6:"ミク";s:3:"age";i:16;}"
```

変数に代入できる

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
ココ → 一級市民	変数に代入できる・引数にできる
	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

Closure（無名関数）

- $x2 = function ($n){return $n*2};
- ローカル変数として関数を
  作ることができる
- グローバル関数は一度定義すると
  (同名で)作り直すことはできないが、
  これは何回でも作って捨てられる！

リソース(何者である何か)

- $c = curl_init(); $g = gmp_init();
- リソースって一括りにされてるけど
  互換性はない
  - http://php.net/resource
- resourceは型宣言に書けない

リソースは型宣言できない

```php
<?php

$f = function ($r) { var_dump($r); };
$g = function (resource $r) { var_dump($r); };

$x = fopen('php://temp', 'r');

$f($x);
$g($x);
```

関数や構文を経由して一級市民になれる

階層	できること
神	配列のキーになれる
貴族	簡単に画面に出力できる
上級市民	シリアライズできる
一級市民	変数に代入できる・引数にできる
ココ→	関数や構文を経由して一級市民になれる
	謎の関数によってアクセスできる
	相当がんばらないと操作できない

callable（関数っぽくできる何か）

- タイプヒントにcallableって書ける
- 単なる文字列・配列を紛れやすい
- 最近は引数で受け渡すときは
  callableよりも\Closureがオススメ
- PHP 7.1〜 \Closure::fromCallable()
  これでcallableをClosureに変換できる

callable（関数・メソッド）

```php
<?php

$p = 'printf';
$p('%s %s!', 'Hello', $p);

$c = \DateTimeImmutable::class;
$dt = new $c;

$m = [$dt, 'format'];
echo $m('Y-m-d H:i:s');
```

ファイル

```php
<?php

if (is_development()) {
    $fp = fopen('php://stderr');
} else
    $fp = fopen('/path/to/log.txt');
}

$logger = new MyLogger($fp);
$logger->write($data);
```

ここから下は

一級市民と呼び難い

謎の関数によって
アクセスできる

謎

- ローカル変数
  - compact(), extract()
- プロパティ
  - ReflectionProperty
- 定数
  - constant()

もう時間がない

Symfony\VarDumper

PsySH

whoops!

いい感じに
表示できる

市民権を拡張しよう