

ジェネレータで 無限を手玉に取る術

Techniques for controlling infinity using generator



2020-02-11 PHPerKaigi 2020 Day 2
Coconeri Hall, Nerima, Tokyo #phperkaigi

！ 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 pixiv運営本部
- Emacs Lisper, PHPer, Rubyist
- Emacs PHP Modeのメンテナ引き継ぎました
- 好きなリスプはEmacs Lispです
- Qiitaに記事を書いたり変なコメントしてるよ

pixiv



お絵描きがもっと楽しくなる場所を創る

pixiv 2018年度新卒
エントリー受付中



Friends of Emacs-PHP development

Join us!

<?php

- Repositories 25
- People 5
- Teams 1
- Projects 0
- Settings

Find a repository... Type: All Language: All

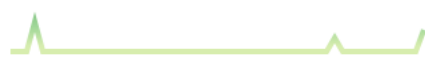
Customize pinned repositories New

php-runtime.el

PHP-Emacs bridge, call PHP function from Emacs

- php
- emacs
- melpa

Emacs Lisp 2 1 GPL-3.0 Updated 2 days ago



phpactor.el

Interface to Phpactor (an intelligent code-completion and refactoring tool for PHP)

Emacs Lisp 8 4 1 issue needs help Updated 3 days ago



php-mode

A PHP mode for GNU Emacs



Top languages

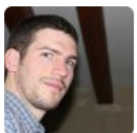
Emacs Lisp PHP

Most used topics Manage

emacs melpa php

major-mode

People 5 >



Invite someone



Search or jump to...



Pull requests

Issues

Marketplace

Explore



emacs-jp

Japan

<http://emacs-jp.github.com/>

Repositories 18

People 37

Teams 8

Projects 0

Settings

Find a repository...

Type: All

Language: All

Customize pinned repositories

New

reading-init.el

init.el読書会のページ

html

dotfiles

emacs

emacs-lisp

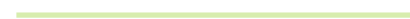
Ruby ★ 6 1 Updated 6 days ago



helm-c-yasnippet

Helm source for yasnippet

Emacs Lisp ★ 14 5 Updated on 21 Mar



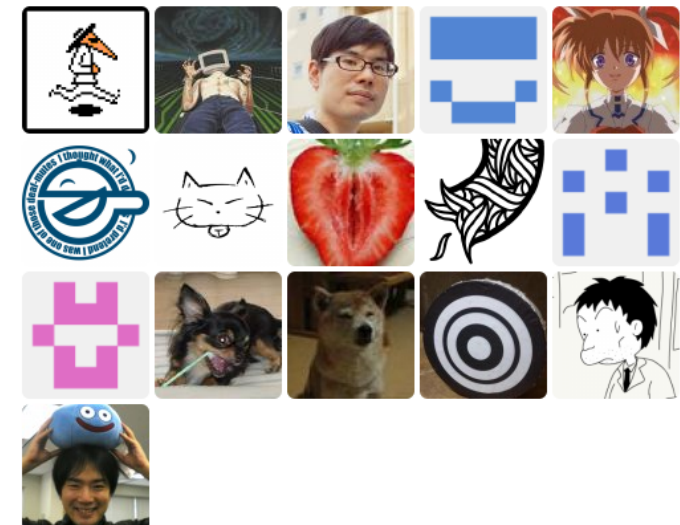
replace-colorthemes

Top languages

Emacs Lisp Ruby TeX

People

37



近況



@tadsan 2020年01月29日に更新 9643 views

このPHPがテンプレートエンジンのくせに慎重すぎる (前篇)

この記事では[Packagist](#)で公開可能な形式のPHPのライブラリ([Composerパッケージ](#))を公開するための道具立てを紹介します。あと、現代のPHPerはツールを組み合わせでさくっと開発しているんだという自慢です。

タイトルは「[この TypeScript が Hello, world! のくせに慎重すぎる](#)」と「[この勇者が俺TUEEEくせに慎重すぎる](#)」のぱく r... パロディです。

テンプレートエンジンのくせに型安全なんてなまいきな。

本日のお題

fortee

ジェネレータで無限を手玉にする術

by うさみけんた / @tadsan

 **PHPPerKaigi**
2020

—ジェネレータで無限を手玉にする術 PHPPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

無限を
手玉に取る

元ねたがある

“

in SlideShare | Search

関数型プログラミングの世界

Welcome to the world of the functional programming

村田 賢太
Kenta Murata

2011.06.11 #osc11do

Monday, June 13, 2011 1

 [Kenta Murata](#), A ruby committer and a software engineer at Recruit Holdings Co., Ltd.
✓ Following

”

いますぐ皆さんに
函数プログラミングを
学べという話ではない

(個人的に)
大事なものは
以下のページ

“

Lazy Evaluation

”

遅延 (怠惰) 評価

“

関数型プログラ

ミングの真髓

”

“

無限を手玉に取る力

”

言ってることは
よくわからんが
カッコイイ

何が
言いたいのか？

遅延評価
必要になるまで
計算を遅らせる

関数型知識ゼロでも
その神髄の一端を
いとも簡単に扱える
機能がPHPにはある

ここまで言えば
おわかりですね？

ジェネレータ Generator

ジエネレータ
とは何か？

「PHPでforeachで反復できるものは何でしょう？」配列、オブジェクト、Iterator、そして“ジェネレータ”です。ジェネレータは言語によってコルーチン、semicoroutineやFiberとも呼ばれるものと同等の言語機能であり、SICPのような計算機科学の教科書で説明されるストリーム・無限リストを簡潔に実現できます。foreachループで配列を反復するのは既に作成済みのデータを頭から辿っていくだけのものであるのに対して、ジェネレータは配列のキーや要素に相当するものを計算によって生み出すことができるため、省メモリで効率のよい計算が可能です。このトークではジェネレータの基礎概念と、実際に応用できるパターンについて紹介していきます。

—ジェネレータで無限を手玉に取る術 PHPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

この説明で全部
なのですが...

ざっくり説明
していきます

“

「PHPでforeachで反復できるものは何でしょう？」
配列、オブジェクト、Iterator、そして“ジェネレータ”
です。

”

—ジェネレータで無限を手玉に取る術 PHPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

| ジェネレータはforeachできる

- 配列を代用できる場面がある
 - 全てを置き換えるものではない
 - あくまでforeachで順次実行するもののみ
- その他の反復可能オブジェクトについては今回触れません

| iterable疑似型 (PHP 7.1)

- 疑似型とは実態のある一種類の型ではなく同じように扱える複数の型とクラスをひとまとめにした便宜上のもの
- iterable = foreachでできる値

！ おまけ: 擬似型

- 前述したように特定の型ではないが、型宣言(引数・戻り値)の型として記述できる
- callable擬似型
 - 関数として呼び出せる値

| おまけ: iterableと型定義

- DocCommentに型を書くとき
キーと値の型を書ける
- `iterable<int, string>`や
`Generator<int, string>`のように
- PhpStormは未対応なので
`iterable|string[]` で代用...

！ おまけ: iterableと型定義

- iterableの型をちゃんと書いておくとforeachの中身も静的解析できる
- PHPStanは書かないと叱ってくれる
- JetBrainsさん早くPhpStorm対応して(deep-assocプラグインで補完可能)
- 詳細は「2018年のPHPDoc事情とPSR-5」に書きました

| 置換できる可能性がある

```
<?php
```

```
// どこかから取得してきたIDリスト
```

```
$target_ids = [85490, 54352, 34242, 45090];
```

```
foreach ($target_ids as $id) {  
    shori($id);  
}
```

| 置換できる可能性がある

```
<?php

// ジェネレータのイメージ

$target_ids = function () { ... };

foreach ($target_ids as $id) {
    shori($id);
}
```

“

ジェネレータは言語によってコルーチン、
semicoroutineやFiberとも呼ばれるものと同等の言
語機能であり、

”

—ジェネレータで無限を手玉に取る術 PHPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

| 用語（呼ばれかた）

- 言語によって呼びかたや機能が若干違います
- JavaScriptやPythonのジェネレータはPHPと似ています
- RubyにはEnumeratorとFiberというものがあります

| コルーチン (coroutine)

- ジェネレータは古典的なコルーチンと同等のもののですが、現代でコルーチンと呼ばれているものは意味がかなり拡大しているので注意が必要です

“

SICPのような計算機科学の教科書で説明されるスト
リーム・無限リストを簡潔に実現できます。

”

—ジェネレータで無限を手玉に取る術 PHPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

SICP

計算機プログラムの
構造と解釈

amazon.co.jp prime

本 ▾

JP 宇佐美健太さん アカウント ▾

なか見!検索↓



計算機プログラムの構造と解釈 第2版 (日本語)
大型本 - 2014/5/17
ハロルド エイブルソン (著), ジュリー サスマン (著), & 5 その他
★★★★☆ ▾ 3個の評価

> その他 () の形式およびエディションを表示する

大型本
¥5,060 

¥2,500 より 20 中古品の出品
¥5,060 より 3 新品

—計算機プログラムの構造と解釈 第2版

<https://www.amazon.co.jp/dp/4798135984>

Webで無料で
も読めます

“

[[目次](#), [前節](#), [次節](#), [索引](#)]

計算機プログラムの
構造と解釈
第二版

ジェラルド・ジェイ・サスマン
ハロルド・エイブルソン
ジュリー・サスマン 共著

和田 英一 訳

”

2014-03-06 更新

[[目次](#), [前節](#), [次節](#), [索引](#)]

–計算機プログラムの構造と解釈 第2版

<https://sicp.iijlab.net/fulltext/>

アスペ日記

2015-10-30

非公式PDF版SICP・新訳

計算機プログラムの構造と解釈、通称SICPを一から翻訳し直しました。

ファイル: [SICP非公式日本語版 翻訳改訂版](#)

リポジトリ: <https://github.com/hiroshi-manabe/sicp-pdf>

また、今回の翻訳をするにあたって考えたことを別記事にまとめました。

–アスペ日記 非公式PDF版SICP・新訳

<https://takeda25.hatenablog.jp/entry/20151030/1446174031>

計算機科学の入門
書という扱いだが、
簡単な本ではない

この本の序盤
2章あたりに
書いてあること

ペア(二値の組を持つオブ
ジェクト)があれば様々
なデータ構造が作れる

ペアの片方を関数で
遅延させることで
ストリーム(遅延リス
ト)というものが作れる

ストリーム (遅延リスト)

- 配列・オブジェクトのようなものとクロージャがあれば実現できる
- PHPで実装してみた

<https://github.com/zonuxe/phperkaigi-generator/blob/master/stream.php>

ストリームでできる
ことはジェネレータ
で表現できる

“ foreachループで配列を反復するのは既に作成済みのデータを頭から辿っていくだけのものであるのに対して、ジェネレータは配列のキーや要素に相当するものを計算によって生み出すことができるため、省メモリで効率のよい計算が可能です。 ”

—ジェネレータで無限を手玉に取る術 PHPerKaigi 2020

<https://fortee.jp/phperkaigi-2020/proposal/5b1651a5-0a01-4b49-b338-e332f981bf5d>

ということで、
ジェネレータを
説明します

ジエネレータの
書きかた

| ジェネレータ=特別な関数

- 関数定義(メソッド・クロージャを含む)の中でyieldと書くとジェネレータになる
- yieldに渡した値はforeachの値として受け渡される

| PHPUnit dataProvider

- テストケースに必要なパラメータを生成するメソッド
- 配列を返すように作るのが一般的だが、実はジェネレータでも作れる

| 基本的なジェネレータ

```
<?php

// どこかから取得してきたIDリスト
f
$target_ids = [85490, 54352, 34242, 45090];

foreach ($target_ids as $id) {
    shori($id);
}
```

| 基本的なジェネレータ

```
<?php
```

```
// どこかから取得してきたIDリスト
```

```
f
```

```
$target_ids = [85490, 54352, 34242, 45090];
```

```
foreach ($target_ids as $id) {  
    shori($id);  
}
```