Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

Атрибуты »
« Синтаксис генераторов

- Руководство по PHP
- Справочник языка
- Генераторы

Change language: Russian

Сравнение генераторов с объектами класса Iterator

Главное преимущество генераторов - это их простота. Гораздо меньше шаблонного кода надо написать, по сравнению с реализацией объекта класса Iterator, и этот код гораздо более простой и понятный. К примеру, эта функция и класс делают одно и то же.

```
<?php
function getLinesFromFile($fileName) {
if (!$fileHandle = fopen($fileName, 'r')) {
return;
}

while (false !== $line = fgets($fileHandle)) {
yield $line;
}

fclose($fileHandle);
}

// Против...

class LineIterator implements Iterator {
protected $fileHandle;

protected $line;
protected $i;

public function __construct($fileName) {
if (!$this->fileHandle = fopen($fileName, 'r')) {
throw new RuntimeException('Невозможно открыть файл "' . $fileName . '"');
}
}

public function rewind() {
fseek($this->fileHandle, 0);
$this->line = fgets($this->fileHandle);
$this->i = 0;
}

public function valid() {
return false !== $this->line;
}

public function current() {
return $this->line;
}

public function key() {
return $this->i;
}

public function next() {
if (false !== $this->line) {
$this->line = fgets($this->fileHandle);
$this->i++;
}
}

public function __destruct() {
fclose($this->fileHandle);
}
}
?>
```

Однако за эту простоту, впрочем, приходится платить: генераторы могут быть только однонаправленными итераторами. Их нельзя перемотать назад после старта итерации. Это также означает, что один и тот же генератор нельзя использовать несколько раз: генератор необходимо пересоздавать каждый раз, снова вызвав функцию генератора.

Смотрите также

- Итераторы объектов

+ add a note

User Contributed Notes 4 notes

up
down
105
mNOSPAMsenghaa at nospam dot gmail dot com ¶
10 years ago

```
This hardly seems a fair comparison between the two examples, size-for-size. As noted, generators are forward-only, meaning that it should be compared to an iterator with a dummy rewind function defined. Also, to be fair, since the iterator throws an exception, shouldn't the generator example also throw the same exception? The code comparison would become more like this:

<?php
function getLinesFromFile($fileName) {
if (!$fileHandle = fopen($fileName, 'r')) {
throw new RuntimeException('Couldn\'t open file "' . $fileName . '"');
}

while (false !== $line = fgets($fileHandle)) {
yield $line;
}

fclose($fileHandle);
}

// versus...

class LineIterator implements Iterator {
protected $fileHandle;

protected $line;
protected $i;

public function __construct($fileName) {
if (!$this->fileHandle = fopen($fileName, 'r')) {
throw new RuntimeException('Couldn\'t open file "' . $fileName . '"');
}
}

public function rewind() { }

public function valid() {
return false !== $this->line;
}

public function current() {
return $this->line;
}
public function key() {
return $this->i;
}

public function next() {
if (false !== $this->line) {
$this->line = fgets($this->fileHandle);
$this->i++;
}
}

public function __destruct() {
fclose($this->fileHandle);
}
}
?>
```

The generator is still obviously much shorter, but this seems a more reasonable comparison.

up
down
20

sergeyzsg at yandex dot ru ¶

9 years ago

I think that this is bad generator example.
If user will not consume all lines then file will not be closed.

```
<?php
function getLinesFromFile($fileHandle) {
while (false !== $line = fgets($fileHandle)) {
yield $line;
}
}

if ($fileHandle = fopen($fileName, 'r')) {
/*
something with getLinesFromFile
*/
fclose($fileHandle);
}
?>
```

up
down
-8

sou at oand dot re ¶

10 years ago

I think to be more similar the samples in the function, throw a new exception is better. But looking into "Generator syntax" session, you can see there this: "An empty return statement is valid syntax within a generator and it will terminate the generator.". By this point of view, we can imagine that this is just to exemplify an usage of the empty return.

up
down
-7

jorgeley at gmail dot com ¶

4 years ago

I think the power of generators is underestimated here, look at my example:

```
<?php

/**
* simple example class just to have something to instantiate
*/
```

```
class obj {

private $i = 1;
private $a = [];

function __construct($i = 1) {
$this->i = $i;
$this->a = range(0, $i);
}

public function getI() {
return $this->i;
}
//more getters and setters...
}

/**
* this is a common way of returning objects in a bulk
* @param int $n
* @return \obj
*/
function returnObjects($n = 1000) {
$objs = [];
for ($i = 1; $i <= $n; $i++) {
$objs[] = new obj($i);
}
return $objs;
}

/**
* this is a better way using generator, rather than returning all objects,
* it returns one by one (it saves the state of the function in every call)
* @param int $n
*/
function generateObjects($n = 1000) {
for ($i = 1; $i <= $n; $i++) {
/**
* 'yield' returns the object and save the status of the function, so
* next call starts from next loop iteration and so on...
*/
yield (new obj($i));
}
}

//main script: get current memory, run one of the functions and calculate memory usage after
$m = memory_get_peak_usage();
/**
* comment 'returnObjects()' call bellow and uncomment 'generateObjects()' call
* if you want to see the generator memory usage
*/
//$objs = returnObjects();

/**
* comment 'generateObjects()' and uncomment 'returnObjects()' call if you
* want to see the common function return memory usage
*/
$objs = generateObjects();
foreach ($objs as $obj) {
echo get_class($obj) . ": {$obj->getI()}\n";
}
echo "total memory comsuption: " . (memory_get_peak_usage() - $m) . " bytes\n";
```

```
?>
```

```
what is the outcome? Using the 'returnObjects()' we return an array within 1000 objects, but using the 'generateObjects()'
we only instantiate one object since the yield returns (stop the loop) but also saves the state of the function, so next
call the function resumes rather than restarting. In my environment I got a difference of 37K to 25M
Thanks to my dear friend Ivan Frezza who helped me to understand this better!
```

+add a note