链滴

在 PHP 中使用 `yield` 来做内存优化

作者：xiaoxiezaijia
原文链接：https://ld246.com/article/1567150507592
来源网站：链滴

你有没有想过 "在 PHP 中使用 yield 会有什么益处"，我将为你节省一些谷歌搜索的时间； 我列出了些要向你介绍的要点来全面认知 yield：

1. 什么是 yield。
2. yield & return 的区别。
3. yield 有什么选项。
4. 结论。
5. 参考。

---

1. 什么是 "yield"

生成器函数看上去就像一个普通函数， 除了不是返回一个值之外， 生成器会根据需求产生更多的值。

来看以下的例子：

```
function getValues() {
    yield 'value';
}

// 输出字符串 "value"
echo getValues();
```

当然， 这不是他生效的方式， 前面的例子会给你一个致命的错误： 类生成器的对象不能被转换成字串， 让我们清楚的说明：

2. "yield" & "return" 的区别

前面的错误意味着 getValues() 方法不会如预期返回一个字符串，让我们检查一下他的类型：

```
function getValues() {
    return 'value';
}
var_dump(getValues()); // string(5) "value"

function getValues() {
    yield 'value';
}
var_dump(getValues()); // class Generator#1 (0) {}
```

生成器 类实现了 生成器 接口， 这意味着你必须遍历 getValue() 方法来取值：

```
foreach (getValues() as $value) {
   echo $value;
}

// 使用变量也是好的
$values = getValues();
```

```
foreach ($values as $value) {
   echo $value;
}
```

但这不是唯一的不同！

一个生成器运行你写使用循环来迭代一维数组的代码，而不需要在内存中创建是一个数组，这可能会致你超出内存限制。

在下面的例子里我们创建一个有 800,000 元素的数字同时从 getValues() 方法中返回他，同时在此期，我们将使用函数 memory_get_usage() 来获取分配给次脚本的内存， 我们将会每增加 200,000 个素来获取一下内存使用量，这意味着我们将会提出四个检查点：

```
<?php
function getValues() {
   $valuesArray = [];
   // 获取初始内存使用量
   echo round(memory_get_usage() / 1024 / 1024, 2) . ' MB' . PHP_EOL;
   for ($i = 1; $i < 800000; $i++) {
      $valuesArray[] = $i;
      // 为了让我们能进行分析，所以我们测量一下内存使用量
      if (($i % 200000) == 0) {
         // 来 MB 为单位获取内存使用量
         echo round(memory_get_usage() / 1024 / 1024, 2) . ' MB'. PHP_EOL;
      }
   }
   return $valuesArray;
}
$myValues = getValues(); // 一旦我们调用函数将会在这里创建数组
foreach ($myValues as $value) {}
```

前面例子发生的情况是这个脚本的内存消耗和输出：

```
0.34 MB
8.35 MB
16.35 MB
32.35 MB
```

这意味着我们的几行脚本消耗了超过 30 MB 的内存， 每次你你添加一个元素到 $valuesArray 数组， 都会增加他在内存中的大小。

让我们使用 yield 同样的例子:

```
<?php
function getValues() {
   // 获取内存使用数据
   echo round(memory_get_usage() / 1024 / 1024, 2) . ' MB' . PHP_EOL;
   for ($i = 1; $i < 800000; $i++) {
      yield $i;
      // 做性能分析，因此可测量内存使用率
      if (($i % 200000) == 0) {
```

```
        // 内存使用以 MB 为单位
        echo round(memory_get_usage() / 1024 / 1024, 2) . ' MB'. PHP_EOL;
    }
  }
}
$myValues = getValues(); // 在循环之前都不会有动作
foreach ($myValues as $value) {} // 开始生成数据
```

这个脚本的输出令人惊讶：

```
0.34 MB
0.34 MB
0.34 MB
0.34 MB
```

这不意味着你从 return 表达式迁移到 yield，但如果你在应用中创建会导致服务器上内存出问题的巨数组，则 yield 更加适合你的情况。

---

3. 什么是 "yield" 选项

这里有很多 yield 的选项， 我将强调他们中的几个：

a. 使用 yield， 你也可以使用 return。

```
function getValues() {
  yield 'value';
  return 'returnValue';
}
$values = getValues();
foreach ($values as $value) {}
echo $values->getReturn(); // 'returnValue'
```

b. 返回键值对：

```
function getValues() {
  yield 'key' => 'value';
}
$values = getValues();
foreach ($values as $key => $value) {
  echo $key . ' => ' . $value;
}
```

点击 这里 查看更多。

4. 结论

这个主题的主要原因是为了明确 yield 和 return 特别是在内存使用方面的区别，使用一些例子是因为

发现他对任何开发人员思考真的很重要。

5. 参考

1. http://php.net/manual/en/language.generators.syntax.php
2. http://php.net/manual/en/class.generator.php
3. http://php.net/manual/en/language.generators.php
4. http://php.net/manual/en/function.memory-get-usage.php

讨论请前往：https://laravel-china.org/topics/8704