PHP 7，让代码更优雅（译）

2018-11-01

PHP 7已发布很久，它可以让代码更加简洁，让我们一睹其风采。

标量类型声明

标量指string、int、float和bool。PHP 7之前，如果要验证一个函数的参数类型，需要手动检测并抛出异常：

```
<?php
function add($num1, $num2) {
    if (!is_int($num1)) {
        throw new Exception("$num1 is not an integer");
    }
    if (!is_int($num2)) {
        throw new Exception("$num2 is not an integer");
    }

    return ($num1 + $num2);
}

echo add(2, 4);     // 6
echo add(1.5, 4);   // Fatal error: Uncaught Exception
```

现在，可以直接声明参数类型：

```
<?php
function add(int $num1, int $num2) {
    return ($num1 + $num2);
}
echo add(2, 4);     // 6
echo add("2", 4);   // 6
echo add("sonething", 4);   // Fatal error: Uncaught TypeError
```

由于PHP默认运行在coercive模式，所以”2”被成功解析为2。可以使用declare函数启用严格模式：

```
<?php
declare(strict_types=1);

function add(int $num1, int $num2) {
    return ($num1 + $num2);
}
echo add(2, 4);     // 6
echo add("2", 4);   // Fatal error: Uncaught TypeError
```

返回类型声明

像参数一样，现在返回值也可以指定类型：

```
<?php
function add($num1, $num2):int {
    return ($num1 + $num2);
}

echo add(2, 4);     // 6
echo add(2.5, 4);   // 6
```

2.5 + 4返回了int类型的6，这是隐式类型转换。如果要避免隐式转换，可以使用严格模式来抛出异常：

```
<?php
```

```
declare(strict_types=1);

function add($num1, $num2):int{
    return ($num1 + $num2);
}

echo add(2, 4); //6
echo add(2.5, 4); //Fatal error: Uncaught TypeError
```

空合并运算符

在PHP5中，检测一个变量，如果未定义则为其赋初值，实现起来需要冗长的代码：

```
$username = isset($_GET['username']) ? $_GET['username'] : '';
```

在PHP 7中，可以使用新增的”??”运算符：

```
$username = $_GET['username'] ?? '';
```

这虽然仅仅是一个语法糖，但能让我们的代码简洁不少。

太空船运算符

也叫组合运算符，用于比较两表达式的大小。当$a小于、等于、大于$b时，分别返回-1、0、1。

```
echo 1 <=> 1;   // 0
echo 1 <=> 2;   // -1
echo 2 <=> 1;   // 1
```

批量导入声明

在相同命名空间下的类、函数、常量，现在可以使用一个use表达式一次导入：

```
<?php
// PHP 7之前
use net\smallyu\ClassA;
use net\smallyu\ClassB;
use net\smallyu\ClassC as C;

use function net\smallyu\funA;
use function net\smallyu\funB;
use function net\smallyu\funC;

use const net\smallyu\ConstA;
use const net\smallyu\ConstB;
use const net\smallyu\ConstC;

// PHP 7
use net\smallyu\{ClassA, ClassB, ClassC};
use function net\smallyu\{funA, funB, funC};
use const net\smallyu\{ConstA, ConstB, ConstC};
```

生成器相关特性

PHP中Generator函数和普通函数的形式相同。生成器使用在foreach的迭代中，比数组占用内存更少，效率更高。这是一个生成器的例子：

```
<?php
// 返回一个生成器
function getValues($max) {
    for ($i = 0; $i < $max; $i++) {
        yield $i * 2;
    }
```

```
}

// 使用生成器
foreach(getValues(99999) as $value) {
    echo "Values: $value \n";
}
```

代码中出现了yield表达式，它就像return一样，在函数中返回一个值，每次只执行一次，并且会从上一次停止的位置开始执行。

PHP 7之前不允许生成器函数使用return返回值，现在允许了，return不会影响yield的正常迭代，return的值也可以使用$gen->getReturn()来获取：

```
<?php
$gen = (function() {
    yield "First Yield";
    yield "Second Yield";

    return "return Value";
})();

foreach ($gen as $val) {
    echo $val, PHP_EOL;
}

echo $gen->getReturn();
```

PHP 7还支持生成器委派，可以在一个生成器函数中调用另一个生成器：

```
<?php
function gen() {
    yield "yield 1 from gen1";
    yield "yield 2 from gen1";
    yield from gen2();
}

function gen2() {
    yield "yield 3 from gen2";
    yield "yield 4 from gen2";
}

foreach (gen() as $val) {
    echo $val, PHP_EOL;
}
```

匿名类

PHP 7也有匿名类啦。

闭包

PHP 7对闭包的支持更加友好：

```
<?php
class A { private $x = 1; }

// PHP 7之前
$getAFun = function() {return $this->x;};
$getA = $getAFun->bindTo(new A, 'A'); // 中间层闭包
echo $getA();

// PHP 7之后
$getA = function() { return $this->x; };
echo $getA->call(new A);
```

可为空类型

Nullable types是PHP 7.1的新特性之一，在参数类型声明前加上一个问号，约定该参数只能是指定类型或者NULL。可以用在返回类型上:

```
<?php

function testReturn(): ?string {
    return 'testing';
}
var_dump(testReturn());     // string(7) "testing"

function testReturn2(): ?string {
    return null;
}
var_dump(testReturn2());    //NULL
```

也可以用在参数类型上:

```
<?php
function test(?string $name) {
    var_dump($name);
}

test('testing');    // string(7) "testing"
test(null);         // NULL
test();     // Fatal error: Uncaught ArgumentCountError
```

数组解构

list()函数的简化写法:

```
<?php

$records = [
    [1, 'smallyu'],
    [2, 'bigyu'],
];

// list() 风格
list($firstId, $firstName) = $records[0];

// [] 风格, PHP 7.1
[$firstId, $firstName] = $records[0];

var_dump($firstId);     // int(1)
var_dump($firstName);   // string(7) "smallyu"
```

另一个新特性是list()和[]都支持keys了:

```
<?php

$records = [
    ["id" => 1, "name" => 'smallyu'],
    ["id" => 2, "name" => 'bigyu'],
];

// list() 风格
list("id" => $firstId, "name" => $firstName) = $records[0];

// [] 风格, PHP 7.1
["id" => $firstId, "name" => $firstName] = $records[0];

var_dump($firstId);     // int(1)
var_dump($firstName);   // string(7) "smallyu"
```

参考

- 《Building REATful Web Services with PHP 7》 Chapter 2: PHP 7, To Code It Better
- PHP: 新特性 - Manual