日照职业技术学院

山东省“十三五”高校工程技术研发中心

基本代码规范

(PHP Standard Recommendations)

二零一八年一月

目 录

代码风格规范

本篇规范是 PSR-1 基本代码规范的继承与扩展。

本规范希望通过制定一系列规范化 PHP 代码的规则，以减少在浏览不同作者的代码时，因代码风格的不同而造成不便。

当多名程序员在多个项目中合作时，就需要一个共同的编码规范，

而本文中的风格规范源自于多个不同项目代码风格的共同特性，

因此，本规范的价值在于我们都遵循这个编码风格，而不是在于它本身。

关键词 “必须”("MUST")、“一定不可/一定不能”("MUST NOT")、“需要”("REQUIRED")、

“将会”("SHALL")、“不会”("SHALL NOT")、“应该”("SHOULD")、“不该”("SHOULD NOT")、

“推荐”("RECOMMENDED")、“可以”("MAY")和”可选“("OPTIONAL")的详细描述可参见 [RFC 2119]。

1. 概览

代码必须遵循 PSR-1 中的编码规范 。

代码必须使用 4 个空格符而不是 tab 键 进行缩进。

每行的字符数应该软性保持在 80 个之内， 理论上一定不可多于 120 个， 但一定不能有硬性限制。

每个 namespace 命名空间声明语句和 use 声明语句块后面，必须插入一个空白行。

类的开始花括号({)必须写在函数声明后自成一行，结束花括号(})也必须写在函数主体后自成一行。

方法的开始花括号({)必须写在函数声明后自成一行，结束花括号(})也必须写在函数主体后自成一行。

类的属性和方法必须添加访问修饰符（private、protected 以及 public）， abstract 以及 final 必须声明在访问修饰符之前，而 static 必须声明在访问修饰符之后。

控制结构的关键字后必须要有一个空格符，而调用方法或函数时则一定不能有。

控制结构的开始花括号({)必须写在声明的同一行，而结束花括号(})必须写在主体后自成一行。

控制结构的开始左括号后和结束右括号前，都一定不能有空格符。

1.1. 例子

以下例子程序简单地展示了以上大部分规范：

```php
<?php
namespace Vendor\Package;

use FooInterface;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class Foo extends Bar implements FooInterface
{
    public function sampleMethod($a, $b = null)
    {
        if ($a === $b) {
            bar();
        } elseif ($a > $b) {
            $foo->bar($arg1);
        } else {
            BazClass::bar($arg2, $arg3);
        }
    }

    final public static function bar()
    {
        // method body
```

```
    }
}
```

2. 通则

2.1 基本编码准则

代码必须符合 PSR-1 中的所有规范。

2.2 文件

所有 PHP 文件必须使用 Unix LF (linefeed)作为行的结束符。

所有 PHP 文件必须以一个空白行作为结束。

纯 PHP 代码文件必须省略最后的 ?> 结束标签。

2.3. 行

行的长度一定不能有硬性的约束。

软性的长度约束一定要限制在 120 个字符以内，若超过此长度，带代码规范检查的编辑器一定要发出警告，不过一定不可发出错误提示。

每行不应该多于 80 个字符，大于 80 字符的行应该折成多行。

非空行后一定不能有多余的空格符。

空行可以使得阅读代码更加方便以及有助于代码的分块。

每行一定不能存在多于一条语句。

2.4. 缩进

代码必须使用 4 个空格符的缩进，一定不能用 tab 键 。

备注: 使用空格而不是 tab 键缩进的好处在于，避免在比较代码差异、打补丁、重阅代码以及注释时产生混淆。使用空格还可以使调整细微的缩进来改变对齐变得简单。

2.5. 关键字 以及 True/False/Null

PHP 所有关键字必须全部小写。

常量 true 、false 和 null 也必须全部小写。

3. namespace 以及 use 声明

namespace 声明后必须插入一个空白行。

所有 use 必须在 namespace 后声明。

每条 use 声明语句必须只有一个 use 关键词。

use 声明语句块后必须要有一个空白行

例如：

```
<?php
namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

// ... additional PHP code ...
```

4. 类、属性和方法

此处的“类”泛指所有的 class 类、接口以及 traits 可复用代码块。

4.1. 扩展与继承

关键词 extends 和 implements 必须写在类名称的同一行。

类的开始花括号必须独占一行，结束花括号也必须在类主体后独占一行。

```
<?php
namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements \ArrayAccess,
\Countable
{
    // constants, properties, methods
}
```

implements 的继承列表也可以分成多行，这样的话，每个继承接口名称都必须分开独立成行，包括第一个。

```php
<?php
namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements
    \ArrayAccess,
    \Countable,
    \Serializable
{
    // constants, properties, methods
}
```

4.2. 属性

每个属性都必须添加访问修饰符。

一定不可使用关键字 var 声明一个属性。

每条语句一定不可定义超过一个属性。

不要使用下划线作为前缀，来区分属性是 protected 或 private。

以下是属性声明的一个范例：

```php
<?php
namespace Vendor\Package;

class ClassName
{
    public $foo = null;
}
```

4.3. 方法

所有方法都必须声明可见性（访问修饰符）。

方法名称不应该以单个下划线作为前缀来表示是 protected 或 private。

方法名称后一定不能有空格符，其开始花括号必须独占一行，结束花括号也必须在方法主体后单独成一行。参数左括号后和右括号前一定不能有空格。

一个标准的方法声明可参照以下范例，留意其括号、逗号、空格以及花括号的位置。

```php
<?php
namespace Vendor\Package;

class ClassName
{
    public function fooBarBaz($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

4.4. 方法的参数

参数列表中，每个参数后面必须要有一个空格，而前面一定不能有空格。

有默认值的参数，必须放到参数列表的末尾。

```php
<?php
namespace Vendor\Package;

class ClassName
{
    public function foo($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

参数列表可以被拆分成分别有一次缩进的多行，这样，列表中的第一项必须放在下一行，每一行必须只放一个参数。

拆分成多行的参数列表后，结束括号以及方法开始花括号必须写在同一行，中间用一个空格分隔，一起自成一行

```php
<?php
namespace Vendor\Package;

class ClassName
{
    public function aVeryLongMethodName(
        ClassTypeHint $arg1,
        &$arg2,
        array $arg3 = []
    ) {
        // method body
    }
}
```

4.5. abstract 、 final 、 以及 static

需要添加 abstract 或 final 声明时， 必须写在访问修饰符前，而 static 则必须写在其后。

```php
<?php
namespace Vendor\Package;

abstract class ClassName
{
    protected static $foo;

    abstract protected function zim();

    final public static function bar()
    {
        // method body
    }
}
```

4.6. 方法及函数调用

方法及函数调用时，方法名或函数名与参数左括号之间一定不能有空格，参数右括号前也一定不能有空格。每个参数前一定不能有空格，但其后必须有一个空格。

```
<?php
bar();
$foo->bar($arg1);
Foo::bar($arg2, $arg3);
```

参数可以分列成多行，此时包括第一个参数在内的每个参数都必须单独成行。

```
<?php
$foo->bar(
    $longArgument,
    $longerArgument,
    $muchLongerArgument
);
```

5. 控制结构

控制结构的基本规范如下：

- 控制结构关键词后必须有一个空格。
- 左括号（ 后一定不能有空格。
- 右括号 ） 前也一定不能有空格。
- 右括号 ） 与开始花括号 { 间一定有一个空格。
- 结构体主体一定要有一次缩进。
- 结束花括号 } 一定在结构体主体后单独成行。

每个结构体的主体都必须被包含在成对的花括号之中，这能让结构体更加结构话，以及减少加入新行时，出错的可能性。

5.1. if 、elseif 和 else

标准的 if 结构如下代码所示，留意 括号、空格以及花括号的位置，注意 else 和 elseif 都与前面的结束花括号在同一行。

```php
<?php
if (expr1) {
    // if body
} elseif (expr2) {
    // elseif body
} else {
    // else body;
}
```

应该使用关键词 elseif 代替所有 else if ，以使得所有的控制关键字都像是单独的一个词。

5.2. switch 和 case

标准的 switch 结构如下代码所示，留意括号、空格以及花括号的位置。

case 语句必须相对 switch 进行一次缩进，而 break 语句以及 case 内的其它语句都 必须 相对 case 进行一次缩进。

如果存在非空的 case 直穿语句，主体里必须有类似 // no break 的注释。

```php
<?php
switch (expr) {
    case 0:
        echo 'First case, with a break';
        break;
    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
```

```
            return;
        default:
            echo 'Default case';
            break;
}
```

5.3. while 和 do while

一个规范的 while 语句应该如下所示，注意其 括号、空格以及花括号的位置。

```
<?php
while (expr) {
    // structure body
}
```

标准的 do while 语句如下所示，同样的，注意其 括号、空格以及花括号的位置。

```
<?php
do {
    // structure body;
} while (expr);
```

5.4. for

标准的 for 语句如下所示，注意其 括号、空格以及花括号的位置。

```
<?php
for (i = 0; i < 10; i++) {
    // for body
}
```

5.5. foreach

标准的 foreach 语句如下所示，注意其 括号、空格以及花括号的位置。

```php
<?php
foreach ($iterable as $key => $value) {
    // foreach body
}
```

5.6. try, catch

标准的 try catch 语句如下所示，注意其 括号、空格以及花括号的位置。

```php
<?php
try {
    // try body
} catch (FirstExceptionType $e) {
    // catch body
} catch (OtherExceptionType $e) {
    // catch body
}
```

6. 闭包

闭包声明时，关键词 function 后以及关键词 use 的前后都必须要有一个空格。

开始花括号必须写在声明的同一行，结束花括号必须紧跟主体结束的下一行。

参数列表和变量列表的左括号后以及右括号前，必须不能有空格。

参数和变量列表中，逗号前必须不能有空格，而逗号后必须要有空格。

闭包中有默认值的参数必须放到列表的后面。

标准的闭包声明语句如下所示，注意其 括号、逗号、空格以及花括号的位置。

```
<?php
$closureWithArgs = function ($arg1, $arg2) {
    // body
};

$closureWithArgsAndVars = function ($arg1, $arg2) use ($var1, $var2)
{
    // body
};
```

参数列表以及变量列表可以分成多行，这样，包括第一个在内的每个参数或变量都必须单独成行，而列表的右括号与闭包的开始花括号必须放在同一行。

以下几个例子，包含了参数和变量列表被分成多行的多情况。

```
<?php
$longArgs_noVars = function (
    $longArgument,
    $longerArgument,
    $muchLongerArgument
) {
    // body
};

$noArgs_longVars = function () use (
    $longVar1,
    $longerVar2,
    $muchLongerVar3
) {
    // body
};

$longArgs_longVars = function (
    $longArgument,
    $longerArgument,
    $muchLongerArgument
) use (
```

```
        $longVar1,
        $longerVar2,
        $muchLongerVar3
) {
        // body
};

$longArgs_shortVars = function (
        $longArgument,
        $longerArgument,
        $muchLongerArgument
) use ($var1) {
        // body
};

$shortArgs_longVars = function ($arg) use (
        $longVar1,
        $longerVar2,
        $muchLongerVar3
) {
        // body
};
```

注意，闭包被直接用作函数或方法调用的参数时，以上规则仍然适用。

```
<?php
$foo->bar(
        $arg1,
        function ($arg2) use ($var1) {
                // body
        },
        $arg3
);
```

7. 总结

以上规范难免有疏忽，其中包括但不仅限于：

- 全局变量和常量的定义
- 函数的定义
- 操作符和赋值

- 行内对齐
- 注释和文档描述块
- 类名的前缀及后缀
- 最佳实践

附件 1 RFC 文档

RFC2119: Key words for use in RFCs to Indicate Requirement Levels

1.MUST 必须

该关键字，或是术语"REQUIRED"或"SHALL"，意味着它们定义的是一个规范的绝对必要条件。

2.MUST NOT 必须不

该词组，或是词组"SHALL NOT"，意味着它们定义的是一个规范的绝对禁止条件。

3.SHOULD 应该

该关键字，或是形容词" RECOMMENDED "，意味着在特殊的情况下可能存在正当的理由来忽略某特殊条款，但是必须能理解其完整的含义，且在选择不同的做法前慎重考虑。

4.SHOULD NOT 不应该

该词组，或是词组"NOT RECOMMENDED"，意味着在特殊的环境下可能存在正当的理由使得特殊的行为可接受或是甚至有实用价值，但是在实现任何以此标签描述的行为前，应该理解其完整的含义，并慎重权衡情况。

5.MAY 可能

该关键字，或是形容词"OPTIONAL"，意味着一个项目是真正可选的。若开发商选择支持此项目，可能是因为特殊的市场需求，也可能是因为觉得这样可以增强产品--另一个开发商可能会漏掉同一个项目。不支持某特定选项的实现【必须】做好与与另一个支持了该选项的实现--虽然或许功能作了简化--互操作的准备。同理，支持某特定选项的实现【必须】做好与与另一个不支持该选项的实现互操作（当然，该选项提供的特性除外）的准备

6.这些命令的使用指导

在本文档 中定义的命令必须小心、保守地使用。特别的，使用它们【必须】是出于互操作性的实际需要，或是为了限制可能导致潜在危害的行

为（例如限制重传）。举例来说，它们必须不被用来试图引入实现者的某种互操作性所不需要的特定方法。

7.安全方面的考虑

这些习语常被用来描述有安全含义的行为。没有实现一个 MUST 或 SHOULD，或做了规范说【必须不】或【应该不】做的事，产生的影响将是非常的敏感且微妙的。文章的作者应该花些时间来详细阐述不遵循建议或要求将带来的安全问题，因为大多数实现者并不是规范产生的经历和讨论的受益者。

8.致谢

这些术语的定义是一个来自许多 RFCs 的综合定义。另外，得到了许多人如 Robert Ullmann, ThomasNarten, Neal McBurnett, and Robert Elz 的建议。