**编码风格指南**

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。

**关于「能愿动词」的使用**

为了避免歧义，文档大量使用了「能愿动词」，对应的解释如下：

* 必须 (MUST)：绝对，严格遵循，请照做，无条件遵守；
* 一定不可 (MUST NOT)：禁令，严令禁止；
* 应该 (SHOULD) ：强烈建议这样做，但是不强求；
* 不该 (SHOULD NOT)：强烈不建议这样做，但是不强求；
* 可以 (MAY) 和 可选 (OPTIONAL) ：选择性高一点，在这个文档内，此词语使用较少；

**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 **一定不可** 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 **一定不可** 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 **一定不可** 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

namespace Vendor\Package;

use FooInterface;

use BarClass as Bar;

use OtherVendor\OtherPackage\BazClass;

class Foo extends Bar implements FooInterface

{

public function sampleFunction($a, $b = null)

{

if ($a === $b) {

bar();

} elseif ($a > $b) {

$foo->bar($arg1);

} else {

BazClass::bar($arg2, $arg3);

}

}

final public static function bar()

{

// 方法的内容

}

}

**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 **一定不可** 有硬性的约束。

软性的长度约束 **必须** 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 **必须** 要发出警告，不过 **一定不可** 发出错误提示。

每行 **不该** 多于80个字符，大于80字符的行 **应该** 折成多行。

非空行后 **一定不可** 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 **一定不可** 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，**一定不可** 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。  
并且，使用空格缩进，让对齐变得更方便。

**2.5. 关键字 以及 True/False/Null**

PHP所有 [关键字](http://php.net/manual/en/reserved.keywords.php) **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;

// ... 更多的 PHP 代码在这里 ...

**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

{

// 这里面是常量、属性、类方法

}

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

namespace Vendor\Package;

use FooClass;

use BarClass as Bar;

use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements

\ArrayAccess,

\Countable,

\Serializable

{

// 这里面是常量、属性、类方法

}

**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。

**不该** 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

namespace Vendor\Package;

class ClassName

{

public $foo = null;

}

**4.3. 方法**

所有方法都 **必须** 添加访问修饰符。

**不该** 使用下划线作为前缀，来区分方法是 protected 或 private。

方法名称后 **一定不可** 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 **一定不可** 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

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，每个逗号后面 **必须** 要有一个空格，而逗号前面 **一定不可** 有空格。

有默认值的参数，**必须** 放到参数列表的末尾。

<?php

namespace Vendor\Package;

class ClassName

{

public function foo($arg1, &$arg2, $arg3 = [])

{

// method body

}

}

参数列表 **可以** 分列成多行，这样，包括第一个参数在内的每个参数都 **必须** 单独成行。

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。

<?php

namespace Vendor\Package;

class ClassName

{

public function aVeryLongMethodName(

ClassTypeHint $arg1,

&$arg2,

array $arg3 = []

) {

// 方法的内容

}

}

**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

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 **一定不可** 有空格，参数右括号前也 **一定不可** 有空格。每个参数前 **一定不可** 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 **一定不可** 有空格。
* 右括号 ) 前也 **一定不可** 有空格。
* 右括号 ) 与开始花括号 { 间 **必须** 有一个空格。
* 结构体主体 **必须** 要有一次缩进。
* 结束花括号 } **必须** 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，请留意「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

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

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

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

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，**一定不可** 有空格。

参数和变量列表中，逗号前 **一定不可** 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」、「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

);

1. **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