**代码风格指南**

本手册是基础代码规范([PSR-1](https://github.com/hfcorriez/fig-standards/blob/zh_CN/%E6%8E%A5%E5%8F%97/PSR-1-basic-coding-standard.md))的继承和扩展。

为了尽可能的提升阅读其他人代码时的效率，下面例举了一系列的通用规则，特别是有关于PHP代码风格的。

各个成员项目间的共性组成了这组代码规范。当开发者们在多个项目中合作时，本指南将会成为所有这些项目中共用的一组代码规范。 因此，本指南的益处不在于这些规则本身，而在于在所有项目中共用这些规则。

[RFC 2119](http://www.ietf.org/rfc/rfc2119.txt)中的必须(MUST)，不可(MUST NOT)，建议(SHOULD)，不建议(SHOULD NOT)，可以/可能(MAY)等关键词将在本节用来做一些解释性的描述。

**1. 概述**

* 代码必须遵守 [PSR-1](https://github.com/hfcorriez/fig-standards/blob/zh_CN/%E6%8E%A5%E5%8F%97/PSR-1-basic-coding-standard.md)。
* 代码必须使用4个空格来进行缩进，而不是用制表符。
* 一行代码的长度不建议有硬限制；软限制必须为120个字符，建议每行代码80个字符或者更少。
* 在命名空间(namespace)的声明下面必须有一行空行，并且在导入(use)的声明下面也必须有一行空行。
* 类(class)的左花括号必须放到其声明下面自成一行，右花括号则必须放到类主体下面自成一行。
* 方法(method)的左花括号必须放到其声明下面自成一行，右花括号则必须放到方法主体的下一行。
* 所有的属性(property)和方法(method) 必须有可见性声明；抽象(abstract)和终结(final)声明必须在可见性声明之前；而静态(static)声明必须在可见性声明之后。
* 在控制结构关键字的后面必须有一个空格；而方法(method)和函数(function)的关键字的后面不可有空格。
* 控制结构的左花括号必须跟其放在同一行，右花括号必须放在该控制结构代码主体的下一行。
* 控制结构的左括号之后不可有空格，右括号之前也不可有空格。

**1.1. 示例**

这个示例中简单展示了上文中提到的一些规则：

<?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()

{

// 方法主体

}

}

**2. 通则**

**2.1 基础代码规范**

代码必须遵守 [PSR-1](https://github.com/hfcorriez/fig-standards/blob/zh_CN/%E6%8E%A5%E5%8F%97/PSR-1-basic-coding-standard.md) 中的所有规则。

**2.2 源文件**

所有的PHP源文件必须使用Unix LF(换行)作为行结束符。

所有PHP源文件必须以一个空行结束。

纯PHP代码源文件的关闭标签?> 必须省略。

**2.3. 行**

行长度不可有硬限制。

行长度的软限制必须是120个字符；对于软限制，代码风格检查器必须警告但不可报错。

一行代码的长度不建议超过80个字符；较长的行建议拆分成多个不超过80个字符的子行。

在非空行后面不可有空格。

空行可以用来增强可读性和区分相关代码块。

一行不可多于一个语句。

**2.4. 缩进**

代码必须使用4个空格，且不可使用制表符来作为缩进。

注意：代码中只使用空格，且不和制表符混合使用，将会对避免代码差异，补丁，历史和注解中的一些问题有帮助。空格的使用还可以使通过调整细微的缩进来改进行间对齐变得更加的简单。

**2.5. 关键字和 True/False/Null**

PHP关键字([keywords](http://php.net/manual/en/reserved.keywords.php))必须使用小写字母。

PHP常量true, false和null 必须使用小写字母。

**3. 命名空间(Namespace)和导入(Use)声明**

命名空间(namespace)的声明后面必须有一行空行。

所有的导入(use)声明必须放在命名空间(namespace)声明的下面。

一句声明中，必须只有一个导入(use)关键字。

在导入(use)声明代码块后面必须有一行空行。

示例：

<?php

namespace Vendor\Package;

use FooClass;

use BarClass as Bar;

use OtherVendor\OtherPackage\BazClass;

// ... 其它PHP代码 ...

**4. 类(class)，属性(property)和方法(method)**

术语“类”指所有的类(class)，接口(interface)和特性(trait)。

**4.1. 扩展(extend)和实现(implement)**

一个类的扩展(extend)和实现(implement)关键词必须和类名(class name)在同一行。

类(class)的左花括号必须放在下面自成一行；右花括号必须放在类(class)主体的后面自成一行。

<?php

namespace Vendor\Package;

use FooClass;

use BarClass as Bar;

use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements \ArrayAccess, \Countable

{

// 常量、属性、方法

}

实现(implement)列表可以被拆分为多个缩进了一次的子行。如果要拆成多个子行，列表的第一项必须要放在下一行，并且每行必须只有一个接口(interface)。

<?php

namespace Vendor\Package;

use FooClass;

use BarClass as Bar;

use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements

\ArrayAccess,

\Countable,

\Serializable

{

// 常量、属性、方法

}

**4.2. 属性(property)**

所有的属性(property)都必须声明其可见性。

变量(var)关键字不可用来声明一个属性(property)。

一条语句不可声明多个属性(property)。

属性名(property name) 不推荐用单个下划线作为前缀来表明其保护(protected)或私有(private)的可见性。

一个属性(property)声明看起来应该像下面这样。

<?php

namespace Vendor\Package;

class ClassName

{

public $foo = null;

}

**4.3. 方法(method)**

所有的方法(method)都必须声明其可见性。

方法名(method name) 不推荐用单个下划线作为前缀来表明其保护(protected)或私有(private)的可见性。

方法名(method name)在其声明后面不可有空格跟随。其左花括号必须放在下面自成一行，且右花括号必须放在方法主体的下面自成一行。左括号后面不可有空格，且右括号前面也不可有空格。

一个方法(method)声明看来应该像下面这样。 注意括号，逗号，空格和花括号的位置：

<?php

namespace Vendor\Package;

class ClassName

{

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

{

// 方法主体部分

}

}

**4.4. 方法(method)的参数**

在参数列表中，逗号之前不可有空格，而逗号之后则必须要有一个空格。

方法(method)中有默认值的参数必须放在参数列表的最后面。

<?php

namespace Vendor\Package;

class ClassName

{

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

{

// 方法主体部分

}

}

参数列表可以被拆分为多个缩进了一次的子行。如果要拆分成多个子行，参数列表的第一项必须放在下一行，并且每行必须只有一个参数。

当参数列表被拆分成多个子行，右括号和左花括号之间必须又一个空格并且自成一行。

<?php

namespace Vendor\Package;

class ClassName

{

public function aVeryLongMethodName(

ClassTypeHint $arg1,

&$arg2,

array $arg3 = []

) {

// 方法主体部分

}

}

**4.5. 抽象(abstract)，终结(final)和 静态(static)**

当用到�抽象(abstract)和终结(final)来做类声明时，它们必须放在可见性声明的前面。

而当用到静态(static)来做类声明时，则必须放在可见性声明的后面。

<?php

namespace Vendor\Package;

abstract class ClassName

{

protected static $foo;

abstract protected function zim();

final public static function bar()

{

// 方法主体部分

}

}

**4.6. 调用方法和函数**

调用一个方法或函数时，在方法名或者函数名和左括号之间不可有空格，左括号之后不可有空格，右括号之前也不可有空格。参数列表中，逗号之前不可有空格，逗号之后则必须有一个空格。

<?php

bar();

$foo->bar($arg1);

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

参数列表可以被拆分成多个缩进了一次的子行。如果拆分成子行，列表中的第一项必须放在下一行，并且每一行必须只能有一个参数。

<?php

$foo->bar(

$longArgument,

$longerArgument,

$muchLongerArgument

);

**5. 控制结构**

下面是对于控制结构代码风格的概括：

* 控制结构的关键词之后必须有一个空格。
* 控制结构的左括号之后不可有空格。
* 控制结构的右括号之前不可有空格。
* 控制结构的右括号和左花括号之间必须有一个空格。
* 控制结构的代码主体必须进行一次缩进。
* 控制结构的右花括号必须主体的下一行。

每个控制结构的代码主体必须被括在花括号里。这样可是使代码看上去更加标准化，并且加入新代码的时候还可以因此而减少引入错误的可能性。

**5.1. if，elseif，else**

下面是一个if条件控制结构的示例，注意其中括号，空格和花括号的位置。同时注意else和elseif要和前一个条件控制结构的右花括号在同一行。

<?php

if ($expr1) {

// if body

} elseif ($expr2) {

// elseif body

} else {

// else body;

}

推荐用elseif来替代else if，以保持所有的条件控制关键字看起来像是一个单词。

**5.2. switch，case**

下面是一个switch条件控制结构的示例，注意其中括号，空格和花括号的位置。case语句必须要缩进一级，而break关键字（或其他中止关键字）必须和case结构的代码主体在同一个缩进层级。如果一个有主体代码的case结构故意的继续向下执行则必须要有一个类似于// 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，do while**

下面是一个while循环控制结构的示例，注意其中括号，空格和花括号的位置。

<?php

while ($expr) {

// structure body

}

下面是一个do while循环控制结构的示例，注意其中括号，空格和花括号的位置。

<?php

do {

// structure body;

} while ($expr);

**5.4. for**

下面是一个for循环控制结构的示例，注意其中括号，空格和花括号的位置。

<?php

for ($i = 0; $i < 10; $i++) {

// for body

}

**5.5. foreach**

下面是一个foreach循环控制结构的示例，注意其中括号，空格和花括号的位置。

<?php

foreach ($iterable as $key => $value) {

// foreach body

}

**5.6. try, catch**

下面是一个try catch异常处理控制结构的示例，注意其中括号，空格和花括号的位置。

<?php

try {

// try body

} catch (FirstExceptionType $e) {

// catch body

} catch (OtherExceptionType $e) {

// catch body

}

**6. 闭包**

声明闭包时所用的function关键字之后必须要有一个空格，而use关键字的前后都要有一个空格。

闭包的左花括号必须跟其在同一行，而右花括号必须在闭包主体的下一行。

闭包的参数列表和变量列表的左括号后面不可有空格，右括号的前面也不可有空格。

闭包的参数列表和变量列表中逗号前面不可有空格，而逗号后面则必须有空格。

闭包的参数列表中带默认值的参数必须放在参数列表的结尾部分。

下面是一个闭包的示例。注意括号，空格和花括号的位置。

<?php

$closureWithArgs = function ($arg1, $arg2) {

// body

};

$closureWithArgsAndVars = function ($arg1, $arg2) use ($var1, $var2) {

// body

};

参数列表和变量列表可以被拆分成多个缩进了一级的子行。如果要拆分成多个子行，列表中的第一项必须放在下一行，并且每一行必须只放一个参数或变量。

当列表（不管是参数还是变量）最终被拆分成多个子行，右括号和左花括号之间必须要有一个空格并且自成一行。

下面是一个参数列表和变量列表被拆分成多个子行的示例。

<?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

};

把闭包作为一个参数在函数或者方法中调用时，依然要遵守上述规则。

<?php

$foo->bar(

$arg1,

function ($arg2) use ($var1) {

// body

},

$arg3

);

**7. 结论**

本指南有意的省略了许多元素的代码风格。主要包括：

* 全局变量和全局常量的声明
* 函数声明
* 操作符和赋值
* 行间对齐
* 注释和文档块
* 类名的前缀和后缀
* 最佳实践

以后的代码规范中可能会修正或扩展本指南中规定的代码风格。

**附录A 调查**

为了写这个风格指南，我们调查了各个项目以最终确定通用的代码风格。并把这次调查在这里公布出来。

**A.1. 调查数据**

url,http://www.horde.org/apps/horde/docs/CODING\_STANDARDS,http://pear.php.net/manual/en/standards.php,http://solarphp.com/manual/appendix-standards.style,http://framework.zend.com/manual/en/coding-standard.html,http://symfony.com/doc/2.0/contributing/code/standards.html,http://www.ppi.io/docs/coding-standards.html,https://github.com/ezsystems/ezp-next/wiki/codingstandards,http://book.cakephp.org/2.0/en/contributing/cakephp-coding-conventions.html,https://github.com/UnionOfRAD/lithium/wiki/Spec%3A-Coding,http://drupal.org/coding-standards,http://code.google.com/p/sabredav/,http://area51.phpbb.com/docs/31x/coding-guidelines.html,https://docs.google.com/a/zikula.org/document/edit?authkey=CPCU0Us&hgd=1&id=1fcqb93Sn-hR9c0mkN6m\_tyWnmEvoswKBtSc0tKkZmJA,http://www.chisimba.com,n/a,https://github.com/Respect/project-info/blob/master/coding-standards-sample.php,n/a,Object Calisthenics for PHP,http://doc.nette.org/en/coding-standard,http://flow3.typo3.org,https://github.com/propelorm/Propel2/wiki/Coding-Standards,http://developer.joomla.org/coding-standards.html

voting,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,no,no,no,?,yes,no,yes

indent\_type,4,4,4,4,4,tab,4,tab,tab,2,4,tab,4,4,4,4,4,4,tab,tab,4,tab

line\_length\_limit\_soft,75,75,75,75,no,85,120,120,80,80,80,no,100,80,80,?,?,120,80,120,no,150

line\_length\_limit\_hard,85,85,85,85,no,no,no,no,100,?,no,no,no,100,100,?,120,120,no,no,no,no

class\_names,studly,studly,studly,studly,studly,studly,studly,studly,studly,studly,studly,lower\_under,studly,lower,studly,studly,studly,studly,?,studly,studly,studly

class\_brace\_line,next,next,next,next,next,same,next,same,same,same,same,next,next,next,next,next,next,next,next,same,next,next

constant\_names,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper

true\_false\_null,lower,lower,lower,lower,lower,lower,lower,lower,lower,upper,lower,lower,lower,upper,lower,lower,lower,lower,lower,upper,lower,lower

method\_names,camel,camel,camel,camel,camel,camel,camel,camel,camel,camel,camel,lower\_under,camel,camel,camel,camel,camel,camel,camel,camel,camel,camel

method\_brace\_line,next,next,next,next,next,same,next,same,same,same,same,next,next,same,next,next,next,next,next,same,next,next

control\_brace\_line,same,same,same,same,same,same,next,same,same,same,same,next,same,same,next,same,same,same,same,same,same,next

control\_space\_after,yes,yes,yes,yes,yes,no,yes,yes,yes,yes,no,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes

always\_use\_control\_braces,yes,yes,yes,yes,yes,yes,no,yes,yes,yes,no,yes,yes,yes,yes,no,yes,yes,yes,yes,yes,yes

else\_elseif\_line,same,same,same,same,same,same,next,same,same,next,same,next,same,next,next,same,same,same,same,same,same,next

case\_break\_indent\_from\_switch,0/1,0/1,0/1,1/2,1/2,1/2,1/2,1/1,1/1,1/2,1/2,1/1,1/2,1/2,1/2,1/2,1/2,1/2,0/1,1/1,1/2,1/2

function\_space\_after,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no

closing\_php\_tag\_required,no,no,no,no,no,no,no,no,yes,no,no,no,no,yes,no,no,no,no,no,yes,no,no

line\_endings,LF,LF,LF,LF,LF,LF,LF,LF,?,LF,?,LF,LF,LF,LF,?,,LF,?,LF,LF,LF

static\_or\_visibility\_first,static,?,static,either,either,either,visibility,visibility,visibility,either,static,either,?,visibility,?,?,either,either,visibility,visibility,static,?

control\_space\_parens,no,no,no,no,no,no,yes,no,no,no,no,no,no,yes,?,no,no,no,no,no,no,no

blank\_line\_after\_php,no,no,no,no,yes,no,no,no,no,yes,yes,no,no,yes,?,yes,yes,no,yes,no,yes,no

class\_method\_control\_brace,next/next/same,next/next/same,next/next/same,next/next/same,next/next/same,same/same/same,next/next/next,same/same/same,same/same/same,same/same/same,same/same/same,next/next/next,next/next/same,next/same/same,next/next/next,next/next/same,next/next/same,next/next/same,next/next/same,same/same/same,next/next/same,next/next/next

**A.2. 调查说明**

indent\_type: 缩进类型。 tab = "使用制表符"，2 or 4 = "空格数量"

line\_length\_limit\_soft: 行长度的“软”限制，用字符。 ? = 不表示或者数字 no 意为不限制.

line\_length\_limit\_hard: 行长度的"硬"限制，用字符。 ? = 不表示或者数字, no 意为不限制.

class\_names: 类名如何命名 lower = 只是小写, lower\_under = 小写加下划线, studly = 骆驼型.

class\_brace\_line: 类的左花括号是放在同(same)一行还是在下(next)一行？

constant\_names: 类常量如何命名？upper = 大写加下划线分隔符。

true\_false\_null: 全小写或者全大写？

method\_names: 方法名如何命名？camel = 驼峰式, lower\_under = 小写加下划线分隔符。

method\_brace\_line: 方法的左花括号在同(same)一行还是在下(next)一行？

control\_brace\_line: 控制结构的左花括号在同(same)一行还是在下(next)一行？

control\_space\_after: 控制结构关键词后是否有空格？

always\_use\_control\_braces: 控制结构总是使用花括号？

else\_elseif\_line: 当使用else和elseif，是否放在同(same)一行还是在下(next)一行？

case\_break\_indent\_from\_switch: case和break分别从swith语句处缩进多少次？

function\_space\_after: 函数调用的函数名和左括号是否有空格？

closing\_php\_tag\_required: 如过是纯PHP文件，关闭标签?>是否需要？

line\_endings: 使用何种的行结束符？

static\_or\_visibility\_first: 在定义方法的时候static和可见性谁在前面？

control\_space\_parens: 在控制结构表达式中，左括号后面和右括号前面是否要有一个空格？yes = if ( $expr ), no =if ($expr).

blank\_line\_after\_php: PHP的开始标签后面是否需要一个空行？

class\_method\_control\_brace: 左花括号在类，方法和控制结构中的位置。

**A.3. 调查结果**

indent\_type:

tab: 7

2: 1

4: 14

line\_length\_limit\_soft:

?: 2

no: 3

75: 4

80: 6

85: 1

100: 1

120: 4

150: 1

line\_length\_limit\_hard:

?: 2

no: 11

85: 4

100: 3

120: 2

class\_names:

?: 1

lower: 1

lower\_under: 1

studly: 19

class\_brace\_line:

next: 16

same: 6

constant\_names:

upper: 22

true\_false\_null:

lower: 19

upper: 3

method\_names:

camel: 21

lower\_under: 1

method\_brace\_line:

next: 15

same: 7

control\_brace\_line:

next: 4

same: 18

control\_space\_after:

no: 2

yes: 20

always\_use\_control\_braces:

no: 3

yes: 19

else\_elseif\_line:

next: 6

same: 16

case\_break\_indent\_from\_switch:

0/1: 4

1/1: 4

1/2: 14

function\_space\_after:

no: 22

closing\_php\_tag\_required:

no: 19

yes: 3

line\_endings:

?: 5

LF: 17

static\_or\_visibility\_first:

?: 5

either: 7

static: 4

visibility: 6

control\_space\_parens:

?: 1

no: 19

yes: 2

blank\_line\_after\_php:

?: 1

no: 13

yes: 8

class\_method\_control\_brace:

next/next/next: 4

next/next/same: 11

next/same/same: 1

same/same/same: 6