PHP 语言代码风格

1. 概览

- 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 sampleFunction($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 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。

2.2 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。

2.3 缩进

代码**必须**使用 4 个空格符的缩进，**一定不能**用 tab 键 。

2.4 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

a. `namespace` 声明后必须插入一个空白行。

b. 所有 `use` 必须在 `namespace` 后声明。

c. 每条 `use` 声明语句必须只有一个 `use` 关键词。

d. `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;
```

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

a. 关键词 `extends` 和 `implements` **必须**写在类名称的同一行。

b. 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
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

a. 每个属性都**必须**添加访问修饰符。
b. **一定不可**使用关键字 var 声明一个属性。
c. 每条语句**一定不可**定义超过一个属性。
d. **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
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。

不要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
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**一定不能**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 = []
    ) {
        // 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
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
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
<?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
<?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
<?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

a. 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。

b. 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。

c. 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。

d. 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。

e. 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
};
```

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