OSR-0 2019(2345 PHP编码规范)

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。

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. 概述

- PHP代码文件 必须 以 <?php 或 <?= 标签开始。
- 所有PHP文件 必须 使用 Unix LF (linefeed) 作为行的结束符。
- 一个源文件 建议 只用来做声明（ 类(class) ， 函数(function) ， 常量(constant) 等）或者只用来做一些引起副作用的操作（例如：输出信息，修改 .ini 配置等）,但 不建议 同时做这两件事。
- 每行的字符数 应该 软性保持在 80 个之内，理论上 不该 多于 120 个。
- 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。
- 类名(class name) 建议 使用 骆驼式(StudlyCaps) 写法。
- 类中的常量所有字母都 必须 大写，单词间用下划线分隔。
- 方法名称 必须 符合 $camelCase 式的小写开头驼峰命名规范。
- 代码 必须 使用4个空格符而不是「Tab 键」进行缩进。
- 类(class) 的开始花括号 必须 写在函数声明后自成一行，结束花括号则 必须 放到类主体下面自成一行。
- 方法(method) 的开始花括号 必须 写在函数声明后自成一行，结束花括号则 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。
- 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，右花括号 必须 放在该控制结构代码主体的下一行。
- 控制结构的开始左括号后和结束右括号前，都 一定不可 有空格符。

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
<?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. 源文件

PHP代码 必须 只使用 长标签(<?php ?>) 或者 短输出式标签(<?= ?>)；而 一定不可 使用其他标签。

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. 副作用

一个源文件`建议`只用来做声明（`类(class)`，`函数(function)`，`常量(constant)`等）或者只用来做一些引起副作用的操作（例如：输出信息，修改`.ini`配置等）,但`不建议`同时做这两件事。

短语`副作用(side effects)`的意思是 *在包含文件时* 所执行的逻辑与所声明的`类(class)`，`函数(function)`，`常量(constant)`等没有直接的关系。

`副作用(side effects)`包含但不局限于：产生输出，显式地使用`require`或`include`，连接外部服务，修改ini配置，触发错误或异常，修改全局或者静态变量，读取或修改文件等等

下面是一个既包含声明又有副作用的示例文件；即应避免的例子：

```php
<?php

// 副作用：修改了ini配置
ini_set('error_reporting', E_ALL);

// 副作用：载入了文件
include "file.php";

// 副作用：产生了输出
echo "<html>\n";

// 声明
function foo()
{
    // 函数体
}
```

下面是一个仅包含声明的示例文件；即应提倡的例子：

```php
<?php

// 声明
function foo()
{
    // 函数体
}

// 条件式声明不算做是副作用
if (! function_exists('bar'))
{
    function bar()
    {
        // 函数体
    }
}
```

2.3. 行

行长度 `一定不可` 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 `一定不可` 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 `一定不可` 有空格。

空行 `可以` 使得阅读代码更加方便以及有助于代码的分块。

一行 `一定不可` 多于一个语句。

2.4. 缩进

代码 `必须` 使用4个空格，且 `一定不可` 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) `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)`、`导入(Use)`和`类名(class name)`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。

一个源文件中只能有一个 `类(class)`。

`类名(class name)` `建议` 使用 `StudlyCaps` 写法。

示例：

```php
<?php

namespace Vendor\Package;

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

class Foo
{
    // ... 其它PHP代码 ...
}
```

4. 类(class)，属性(property)和方法(method)

术语 类(class) 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
<?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
<?php

namespace Vendor\Package;
```

```php
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. 常量

类的常量中所有字母都 `必须` 大写，词间以`下划线(_)`分隔。

例子：

```php
<?php

namespace Vendor\Model;

class Foo
{
    const VERSION = '1.0';
    const DATE_APPROVED = '2012-06-01';
}
```

4.3. 数组

空`数组(Array)`声明的括号内`一定不可`有空格
在单行`数组(Array)`声明里最后一个值`一定不可`有逗号
`数组(Array)`运算符=>前后`必须`有空格，且运算符后`必须`有一个空格
多行`数组(Array)`里每个元素`必须`自成一行
多行`数组(Array)`声明时右括号`必须`自成一行
多行`数组(Array)`最后一个值`必须`有逗号
例子：

```php
<?php

$expr1 = array(1, 2, 3);
$expr2 = array(
    "a",
    "b",
```

```
);
$expr3 = array(
    "a" => 1,
    "b" => 2,
);
```

4.4. 属性(property)

所有的`属性(property)`都`必须`声明其可见性。

`变量(var)`关键字`一定不可`用来声明一个`属性(property)`。

一条语句`一定不可`声明多个`属性(property)`。

`属性名(property name)` `建议`使用`$camelCase`风格来声明。

`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
<?php

namespace Vendor\Package;

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

4.5. 方法(method)

所有的`方法(method)`都`必须`声明其可见性。

`方法名(method name)` `建议`使用`camelCase()`风格来声明。

`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`一定不可`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`一定不可`有空格，且右括号前面也`一定不可`有空格。

一个标准的`方法(method)`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 = [])
    {
        // 方法主体部分
    }
}
```

4.6. 方法(method)的参数

在参数列表中，逗号之前一定不可有空格，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
<?php

namespace Vendor\Package;

class ClassName
{
    public function foo($arg1, &$arg2, $arg3 = [])
    {
        // 方法主体部分
    }
}
```

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
<?php
namespace Vendor\Package;

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

4.7. 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
    }
}
```

4.8. 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一定不可有空格，左括号之后一定不可有空格，右括号之前也一定不可有空格。参数列表中，逗号之前一定不可有空格，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：

- 控制结构的关键词与结构之间 必须 有一个空格。
- 控制结构的左括号后 一定不可 有空格。
- 控制结构的右括号前也 一定不可 有空格。
- 控制结构的右括号与开始花括号间 必须 有一个空格。
- 控制结构的结构体主体 必须 要有一次缩进。
- 控制结构结束右花括号 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`, `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。

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;
}
```

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
<?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
<?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
<?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
<?php
```

```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
<?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
<?php

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

6. 注释

- 文件的头注释（程序签名），`需要`按照以下格式：

```php
<?php

/**
 * Copyright (c) 2012,2345
 * 摘    要：首页程序
 * 作    者：XXX
 * 修改日期：2012.06.05
 */
```

- 函数的头注释，必须按照以下格式：

```php
<?php

/**
 * 功     能：根据用户ID获取该用户详细信息。
 * 修改日期：2012-6-13（editplus中快捷键CTRL+D）
 *
 * @param int $aaa 这个参数解释
 * @param string $bbb 这个参数的解释
 * @return string[]
 */
function abc($aaa, $bbb)
{
    // php code
    return $result;
}
```

7. 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
<?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
<?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
);
```