PHP PSR-2 编码风格指南

PSR-2 Coding Style Guide

【整理于 2013.11.6 李文祥】

目录

PHP-FIG (PHP Framework Interoperability Group) ， 是一帮热心人为了方便PHPer开发、学习，而自由抱团在一起的组织。他们制定标准开发规范，并且落实在自己的项目中。目前参照PHP-FIG 提供的编码规范开发的项目已经有很多，并且多为知名项目。详见 http://www.php-fig.org

其中，PSR 的意思为Proposing a Standard Recommendation。其中RSR-0 为自动加载规范，定义了 namespace 与classname 的格式与文件存放路径；PSR-1 为PHP 代码编写基础建议，以及PSR-2 的代码风格建议；而PSR-3 为Log 接口建议。

极力建议各位认证阅读学习，虽然不强迫大家必须照此开发，但遵照此协议开发，养成习惯，若大家人人如此，则能造福所有PHPer。

这份文件(PSR-2)从 PSR -1 这份基本编码标准所延伸、扩充 。

本文件希望能藉由一套共享的规则让大家可以格式化 PHP 程序，以期降低大家在看各作者间程序代码时，因风格的不同所造成的冲击。

此处的风格规则由不同项目的成员所合作。各成员彼此合作于多个项目间，而这份指导方针让他们使用在各个项目间。因此，这份文件的优点就是没有规则，而唯一的规则就是分享。

在文件中所使用到的关键词 “**MUST**”, “**MUST NOT**”, “**REQUIRED**”, “**SHALL**”, “**SHALL NOT**”, “**SHOULD**”, “**SHOULD NOT**”, “**RECOMMENDED**”, “**MAY**”, 以及 “**OPTIONAL**” 皆引用自 RFC 2199 中说明。

译注：
*为了让语句顺畅，这边就不先针对每个字翻译；而为了维持原文件中之强调性，所以都会将这几个关键词加粗并在其后接上原字，例如**一定 (MUST)**。*

1. 总览

- 程序代码**一定 (MUST)** 得依循 PSR-1(翻译)。
- 程序代码的缩排**一定 (MUST)** 是用四个空白，而非 tab。
- 每行的字数长度**需 (SHOLD)** 得少于 80 字符；**一定 (MUST)** 要将相对限制(soft limit)设定在 120 字符；**必不要 (MUST NOT)** 写到绝对限制 (hard limit)。
- 声明 命名空间 (namespace) 之下**一定(MUST)** 要空一行，以及声明 use 之下一定(MUST) 也要空一行。
- 类别 (class) 所使用的成对花括号，**一定(MUST)** 要将开始的放在声明的下一行，以及程序代码本体结束的下一行。
- 方法 (method) 所使用的成对花括号，**一定(MUST)** 要将开始的放在声明的下一行，以及程序代码本体结束的下一行。

- 所有属性或是方法的可视属性(visibility)一定(MUST) 要声明， `abstract` 以及 `final` 一定(MUST) 是声明在可视属性之前； `static` 则一定(MUST) 是声明在可视属性之后。

译注：PHP 的 visibility 其实就是大家比较常听到的修饰词，private、protected 以及 public 这几个封装属性、函式用的文字。

- 控制结构 (control structure) 的关键词一定(MUST) 要在其后加入一个空白；方法 (Method) 与函式(Function)则一定不要(MUST NOT)。

- 控制结构的开始(左)花括号一定(MUST) 要在声明启始的同一行，而结束(右)花括号则一定(MUST) 要在其程序代码本体结束的下一行。

- 控制结构中，用到括号时，其开始(左)括号之后与结束(右)括号之前一定不要(MUST NOT) 有空白。

1.1 范例

下面这个范例包含一些总览中的规则

```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()
    {
        // method body
    }
```

```
}
```

2. 通则

2.1 基本编码标准

程序代码一定 (MUST) 要依循 PSR-1(翻译) 的说明。

2.2 文件

所有 PHP 文件**一定(MSUT)** 要使用 Unix LF(Line Feed) 做为行的结束（换行）。

所有 PHP 文件在最后结尾处**一定(MSUT)** 要有一空白行。

在只有 PHP 程序代码的文件里，最后**一定 (MUST)** 不要有结束标签 ?> 。

2.3 行

每行长度**一定不要 (MUST NOT)** 用到绝对限制 (hard limit)。

每行的相对限制 (soft limit) **一定 (MUST)** 要设定在 120 个字符；自动风格在侦测相对限制时，**一定 (MUST)** 要设定为警告(warn)，而**不要 (MUST NOT)** 设定为错误(error)。

每行**最好不要 (SHOULD NOT)** 超过 80 个字符；若是超过，**最好(SHOULD)** 是拆成多行并维持每行长度 80 字符内。

在非空行的尾端**一定不要(MUST NOT)** 有空白。

可以 (MAY) 增加空白行增加可读性，同时可以表达段程序代码是有关连的。

每行**一定不要 (MUST NOT)** 超过一个叙述 (statement)。

2.4 缩排

程序代码的缩排**一定 (MUST)** 是使用四个空白，而**不要 (MUST NOT)** 使用制表符（Tab）做为缩排。

备注：只使用空白，不要将空白与 tabs 混用，以避免代码差异、补丁、历史版本以及注释时所发生的问题。而使用空白让我们在进一步做子缩排时，都很容易地做到对齐的动作。

2.5 关键词以及 True/False/Null

PHP 的关键词一定 (MUST) 是小写。

PHP 的 `true`、 `false` 以及 `null` 这几个常数也一定 (MUST) 是小写。

3. Namespace 与 use 声明

当存在时，在声明 `namespace` 的下一行一定(MUST) 要是空行。

当存在时，所有 `use` 声明都一定(MUST) 要在 `namespace` 声明之后。

每个声明都一定(MUST) 要有关键词 `use` 。

在 `use` 区块之后一定(MUST) 要有一行空行隔开。

范例：

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

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

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

4. 类、属性以及函数

`Class` 一词包含所有类、接口以及其所有特性（traits）。

译注：trait 在 PHP 5.4 中泛指所有能被重复使用的函式。
http://php.net/manual/en/language.oop5.traits.php

4.1 继承与实现

这 `extends` 以及 `implements` 关键词与其类别名称必需被声明在同一行。

所使用的成对的大括号之左括号“{”一定 (MUST) 是自己独立一行，而右括号“}”也一定 (MUST) 是独立在程序本体的下一行。

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

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

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

而实作类别的清单**可以 (MAY)** 拆开成数行，而紧接在后的每一行皆要缩排。若是要这么做时，清单中的第一个类别**一定 (MUST)** 是在声明的下一行，而每行就是独立一个接口。

```
<?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

所以属性在声明时**一定 (MUST)** 要给予初始值。

属性的声明**一定不能 (MUST NOT)** 使用关键词 `var` 。

每一次的声明**必不能 (MUST NOT)** 超过一个属性。

属性若是 protected 或是 private 时，其名称**不需要 (SHOULD NOT)** 有个下划线做为前置。

一个属性的声明方式看起来如下。

```
<?php
namespace Vendor\Package;

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

4.3 方法

每个方法的可视属性(visibility) **一定(MUST)** 要声明。

方法名称**不需要(SHOULD NOT)** 在 protected 或是 private 这两个属性前加上下底线("_")为前置符号。

方法名称之后**一定不要(MUST NOT)** 有空白。开始(左)花括号请**务必(MUST)** 让他自己存在于一行，且结束(右)花括号也**一定(MUST)** 要置于程序代码本体之下一行。而开始(左)括号之后以及结束(右)括号之前**一定不要(MUST NOT)** 有空白。

一个方法的声明可参考下面范例。请注意括号、逗号、空白以及花括号的位置：

```
<?php
namespace Vendor\Package;

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

4.4 方法的参数

在参数清单中，每个逗号前一定不要 (MUST NOT) 有空白，而其后**一定 (MUST)** 要有空白。

方法的参数若是有默认值，则**一定(MUST)** 要放在参数清单的最后。

```
<?php
namespace Vendor\Package;

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

参数清单**可以(MAY)** 拆开成多行，而每行就是独立一个参数。若是要这么做，其第一个参数**一定(MUST)**要在声明的下一行，每行**一定(MUST)** 只有一个参数。

当参数清单要拆成多行，其结束(右)括号以及开始(左)花括号**一定(MUST)** 要在同一行，同时其中间要以空白隔开。

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

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

4.5 abstract、final 以及 static

当出现 `abstract` 以及 `final` 声明时，**一定(MUST)** 要在将之声明于可视属性(visbility) 之前。

当出现 `static` 声明时，**一定(MUST)** 要将之声明于可视属性(visbility) 之后。

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

abstract class ClassName
{
    protected static $foo;

    abstract protected function zim();

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

4.6 方法与函数调用

当建立一个方法(method)或是函式(function)时，在函式名称以及开始括号之间**一定不要(MUST NOT)**有空白，而开始括号后与结束括号前也**一定不要 (MUST NOT)** 有空白。在参数列所用的逗号前**一定不要 (MUST NOT)** 有空白，而每个逗号后**一要 (MUST)** 有空白。

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

参数列**可以 (MAY)** 拆成多行，而每行都缩排。若是要这么做，第一个参数**一定(MUST)** 要再下一行，而且每行就**一定 (MUST)** 只有一个参数。

```php
<?php
$foo->bar(
```

```
    $longArgument,
    $longerArgument,
    $muchLongerArgument
);
```

5. 控制结构

控制结构的通用规则如下：

- 在控制结构的关键词之后**一定 (MUST)** 要用一个空白隔开。
- 在左括号 "(" 之后**一定不要 (MUST NOT)** 有空白。
- 在右括号 ")" 之前**一定不要 (MUST NOT)** 有空白。
- 在右括号 ")" 以及开始大(花)括号 "{" 之间 一定 (MUST) 要有一个空白隔开。
- 结构本体**一定 (MUST)** 要有一次缩排。
- 结束的大括号 "}" **一定 (MUST)** 要在结构本体的下一行。

每个结构内容都**一定 (MUST)** 要被包覆在成对的大括号之间。这种标准化看起来会比较有结构，而且可以降低当有新内容要加入主体时错置的可能性。

5.1 if、elseif、else

一个 `if` 结构看起来如下。请注意括号、空白以及大括号的位置；还有 `else` 以及 `elseif` 跟结束的大括号 "}" 在主体前的同一行里。

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

所有 `else if` 这个关键词**需 (SHOULD)** 用 `elseif` 来取代之。

5.2 switch、case

一个 `switch` 结构看起来如下。请注意括号、空白以及大括号的位置。 `case` 叙述**一定 (MUST)** 要比 `switch` 再缩排一次，而关键词 `break` (或其他结束关键词) 的缩排层级一定 (MUST) 是同于 `case` 的程序本体。当今天有一个直透非空的 `case` 本体之状况一定 (MUST) 要批注 `// no break` 。

```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

一个 while 结构看起来如下。请注意括号、空白以及大括号的位置。

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

同样地，一个 do while 结构看起来如下。请注意括号、空白以及大括号的位置。

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

5.4 for

一个 for 叙述看起来如下。请注意括号、空白以及大括号的位置。

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

5.5 foreach

一个 foreach 叙述看起来如下。请注意括号、空白以及大括号的位置。

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

```
}
```

5.6 try、catch

一个 try catch 叙述看起来如下。请注意括号、空白以及大括号的位置。

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

6. 闭包

声明闭包时**一定(MUST)** 要有一个空白在 function 这个关键词之后，以及在关键词 user 的前后都要有一个空白。

开始(左)花括号一定(MUST)是与其声明起始在同一行，且结束(右)花括号**一定(MUST)** 在程序本体结束的下一行。

在开始(左)括号以及结束(右)括号与参数或是变量列表之间**一定不要(MUST NOT)** 有空白。

在参数以及变量列表中的逗号前**一定不要(MUST NOT)** 有空白，以及逗号之后**一定(MUST)** 要有一个空白。

闭包中的参数若是有默认值时，**一定(MUST)** 要放置在参数清单之最后。

闭包的声明可以参考下面范例。请注意括号、逗号、空白以及花括号之位置。

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

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

参数以及变量列表**可以(MAY)** 被拆成多行，其每行皆只有独立一个。当打算这么做时，清单中的第一个参数/变量**一定(MUST)** 要在在声明的下一行，以及每行**一定(MUST)** 只能有一个参数或变数。

当清单被拆成多行其最后(参数或变量皆是如此)，所使用的结束(右)括号以及开始(左)花括号**一定(MUST)**要在同一行并用空白将二者隔开。

下面是闭包的范例。

```
<?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
};
```

请注意若是要在闭包中的参数使用方法 (method) 或是函式 (function) 的格式如下：

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

```
    },
    $arg3
);
```

7. 总结

在这份文件中有许多刻意被省略的元素以及做法。这里仅列出部份：

- 声明全局变量与全局常数
- 声明功能
- 操作数与设定值
- 行内的对齐
- 批注以及文件说明区块
- 类别名称的前置以及结尾
- 最佳的实践

建议未来**可以(MAY)** 修改以及扩充这份文件以解决这些或是其他风格以及实作的元素。