php开发规范

目录

1. 前言

本规范由PSR (PHP Standards Recommendation)编程原则组成，融合并提炼了开发人员长时间积累下来的成熟经验，意在帮助形成良好一致的编程风格。

2. 适用范围

如无特殊说明，以下规则要求完全适用于百合所有PHP开发项目

3. 标准化的重要性和好处

1. 当一个项目尝试着遵守公共一致的标准时，可以使参与项目的开发人员更容易了解项目中的代码、弄清程序的状况。
2. 新的参与者可以很快的适应环境，防止部分参与者出于节省时间的需要，自创一套风格，导致其它人在阅读时浪费过多的时间和精力。
3. 在一致的环境下，也可以减少编码出错的机会。

4. PHP编码规范与原则

4.1. 概览

- 代码**必须**使用4个空格符而不是 tab键 进行缩进。
- 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。

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

4.2. 文件

所有PHP文件**必须**使用 `Unix LF（linefeed）` 作为行的结束符。

纯PHP代码文件**必须**省略最后的 `?>` 结束标签。

4.3. 行

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。

4.4. 缩进

代码**必须**使用4个空格符的缩进，**一定不能**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、注释时产生混淆。 并且，使用空格缩进，让对齐变得更方便。

4.5. 关键字 以及 true/false/null

PHP所有 [1]关键字 **必须**全部小写。

常量 `true` 、 `false` 和 `null` 也**必须**全部小写。

4.6. 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;

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

4.7. 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。

4.7.1. 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
{
    // 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
}
```

4.7.2. 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;
}
```

4.7.3. 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
    }
}
```

4.7.4. 方法的参数

参数列表中，每个参数后面**必须**要有一个空格，而前面**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
    }
}
```

4.7.5. 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
    }
}
```

4.7.6. 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
);
```

4.8. 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。

4.8.1. 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。

4.8.2. 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;
}
```

4.8.3. 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);
```

4.8.4. 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
}
```

4.8.5. `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
}
```

4.8.6. `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
}
```

4.9. 注释

每个程序均必须提供必要的注释，书写注释要求规范。

4.9.1 头部注释

头部注释主要用来说明代码提交人，提交日期，提交版本。请按照下列形式书写(你可以把它设置为代码模板)

```
<?php
/**
 * $Author$
 * $Date$
 * $Revision$
 */
```

`$Author$` `$Date$` `$Revision$` 是为了匹配svn的关键字，设置svn:keywords后，每次提交都会被替换为具体的信息。 例：

```
<?php
/**
 * $Author: chengliang@baihe.com $
 * $Date: 2014-08-19 15:26:05 +0800 (星期二, 19 八月 2014) $
 * $Revision: 52069 $
 */
```

4.9.2 类(接口)注释

一个类(接口)在声明的时候必须声明其作用:

- 如果是类库文件，则必须声明其包所属

此注释参考phpdoc规范

```
/**
 * mySQL操作类
 *
 * @package   DB
 * @author    chengliang@baihe.com
 * @version   Release: 3.0
 */
class mySQL extends DB
{
 ......
}
```

4.9.3 函数(方法)注释

一个函数(方法)在声明的时候必须声明其作用:

- 如果有参数，必须指明 `@param` 参数类型

- 如果是无返回函数，必须指明 @return void ，请尽量在函数参数表中使用已知类型
- 如果函数中抛出异常则必须指明 @throws <异常类型>
- 基础类库必须注明该方法或者函数具体用法样例

此注释参考phpdoc规范

```
/**
 *
 * 将一个二维数组按照指定字段的值分组
 *
 * @param array  $arr       数据源
 * @param string $key_field 作为分组依据的键名
 *
 * @return array 分组后的结果
 *
 * 用法:
 *
 * $rows = array(
 *     array('id' => 1, 'value' => '1-1', 'parent' => 1),
 *     array('id' => 2, 'value' => '2-1', 'parent' => 1),
 *     array('id' => 3, 'value' => '3-1', 'parent' => 1),
 *     array('id' => 4, 'value' => '4-1', 'parent' => 2),
 *     array('id' => 5, 'value' => '5-1', 'parent' => 2),
 *     array('id' => 6, 'value' => '6-1', 'parent' => 3),
 * );
 * $values = Util_Array::groupBy($rows, 'parent');
 */
public static function groupBy($arr, $key_field)
{
    .......
}
```

4.9.4 引用文件和定义常量注释

文件的引用和常量的定义一般都放置在文件的开头部分

```
/* 定义数据库适配器 */
define('__DB_ADAPTER__', 'Mysql');

/* 数据库异常类 */
require_once 'Db/Exception.php';
```

4.9.5 变量或语句注释

程序中变量或者语句的注释遵循以下原则:

- 写在变量或者语句的前面一行，而不写在同行或者后面
- 注释采用 // 的方式
- 把已经注释掉的代码删除，或者注明这些已经注释掉的代码仍然保留在源码中的特殊原因

```
//实例化适配器对象
$this->_adapter = new $adapter();
```

5. 框架与目录

标准项目结构:

```
+ public
  | - index.php             // Application entry
  | + static
     | + js                 // js script files
     | + img                // images files
     | + css                // css files
+ cli                       // eg: crontab shell etc.
+ vendor                    // third library
+ application
  | - Bootstrap.php        // Bootstrap
  | + conf
     | - application.ini   //Configure
  | + controllers
  | + views
  | + library              // project library
  | + models
  | + modules
  | + plugins
  | + services             // Custom Service
```

`DocumentRoot` 设置应用访问的Web根目录为public下

附件: FrameDir (/deepwiki-assets/attach/sr/FrameDir.zip)

1. 关键字 (http://php.net/manual/en/reserved.keywords.php) ↩