[**一、** **绪论** 2](#_Toc318272932)

[**1.** **适用范围** 2](#_Toc318272933)

[**2.** **目标** 2](#_Toc318272934)

[**二、** **PHP File文件格式** 2](#_Toc318272935)

[**1. 常规 （必须**） 2](#_Toc318272936)

[**2. 缩进 （参考）** 2](#_Toc318272937)

[**3. 行的最大长度 （参考）** 2](#_Toc318272938)

[**三、** **命名约定** 2](#_Toc318272939)

[1. 类 2](#_Toc318272940)

[2. 文件名 3](#_Toc318272941)

[3. 函数和方法 3](#_Toc318272942)

[4. 变量 3](#_Toc318272943)

[5. 常量 3](#_Toc318272944)

[**四、** **编码风格** 3](#_Toc318272945)

[1. PHP 代码划分（Demarcation） 3](#_Toc318272946)

[2. 字符串 3](#_Toc318272947)

[2.1. 字符串文字 3](#_Toc318272948)

[2.2. 包含单引号（'）的字符串文字 3](#_Toc318272949)

[2.3. 变量替换 3](#_Toc318272950)

[2.4. 字符串连接 3](#_Toc318272951)

[3. 数组 3](#_Toc318272952)

[3.1. 数字索引数组 3](#_Toc318272953)

[3.2. 关联数组 3](#_Toc318272954)

[4. 类 3](#_Toc318272955)

[4.1. 类的声明 3](#_Toc318272956)

[4.2. 类成员变量 3](#_Toc318272957)

[5. 函数和方法 3](#_Toc318272958)

[5.1. 函数和方法声明 3](#_Toc318272959)

[5.2. 函数和方法的用法 3](#_Toc318272960)

[6. 控制语句 3](#_Toc318272961)

[6.1. if/else/elseif 3](#_Toc318272962)

[6.2. switch 3](#_Toc318272963)

[7. 注释文档 3](#_Toc318272964)

[7.1. 格式 3](#_Toc318272965)

[7.2. 类 3](#_Toc318272966)

[7.4. 函数 3](#_Toc318272967)

[8. SQL 语句 3](#_Toc318272968)

[8.1. 关键字 3](#_Toc318272969)

[8.2. 换行 3](#_Toc318272970)

1. **绪论**
2. **适用范围**

后端php开发人员及涉及到的相关人员（架构师、设计师、前端开发工程师等）

1. **目标**

编码标准对任何开发项目都很重要，特别是很多开发者在同一项目上工作。编码标准帮助确保代码的高质量、少 bug 和容易维护。

1. **PHP File文件格式**

**1. 常规 （必须**）

对于只包含有 PHP 代码的文件，结束标志（"?>"）是不允许存在的，PHP自身不需要（"?>"）, 这样做, 可以防止它的末尾的被意外地注入相应。

**2. 缩进 （参考）**

缩进由四个空格组成，禁止使用制表符 TAB 。

**3. 行的最大长度 （参考）**

一行 80 字符以内是比较合适，就是说，开发者应当努力在可能的情况下保持每行代码少于 80 个字符，在有些情况下，长点也可以, 但最多为 120 个字符。

1. **命名约定**

## 1. 类

类命名总是对应于其所属文件的目录结构的，类 class放置正在lib,libs下面，所有 模块的类在其下按等级存放。

类名只允许有字母数字字符，在大部分情况下不鼓励使用数字。下划线只允许做路径分隔符；例如 User/User.php 文件里对应的类名称是 User\_User。

如果类名包含多个单词，每个单词的第一个字母必须大写，连续的大写是不允许的，例如 “USER\_USER” 是不允许的，而 "User\_User" 是可接受的。

## 2. 文件名

对于其它文件，只有字母数字字符、下划线和短横线（"-"）可用，空格是绝对不允许的。

包含任何 PHP 代码的任何文件应当以 ".php" 扩展名结尾，众所周知的视图脚本除外。下面这些例子给出类可接受的文件名：

Zend/Db.php

Zend/Controller/Front.php

Zend/View/Helper/FormRadio.php

文件名必须遵循上述的对应类名的规则。

## 3. 函数和方法

函数名只包含字母数字字符，下划线是不允许的。数字是允许的但大多数情况下不鼓励。

函数名总是以小写开头，当函数名包含多个单词，每个子的首字母必须大写，这就是所谓的 “驼峰” 格式。 (详细见http://baike.baidu.com/view/1985375.htm)

我们一般鼓励使用长的名字，函数名应当长到足以说明函数的意图和行为。

这些是可接受的函数名的例子：

filterInput()

getElementById()

widgetFactory()

对于面向对象编程，实例或静态变量的访问器总是以 "get" 或 "set" 为前缀。在设计模式实现方面，如单态模式（singleton）或工厂模式（factory）， 方法的名字应当包含模式的名字，这样名字更能描述整个行为。

private方法或者不公开的函数以下划线开头,如: private \_handle(),public 方法不包含下划线

全局函数 (如："floating functions") 允许但大多数情况下不鼓励，建议把这类函数封装到静态类里。

## 4. 变量

变量只包含数字字母字符，大多数情况下不鼓励使用数字

声明为 "private"的实例变量名必须以一个单个下划线开头，这是唯一的下划线在程序中的用法，声明为 "public" 的不应当以下划线开头。

对函数名（见上面 3.3 节）一样，变量名总以小写字母开头并遵循“驼峰式”命名约定。

我们一般鼓励使用长的名字，这样容易理解代码，开发者知道把数据存到哪里。除非在小循环里，不鼓励使用简洁的名字如 "$i" 和 "$n" 。如果一个循环超过 20 行代码，索引的变量名必须有个具有描述意义的名字。

## 5. 常量

常量包含数字字母字符和下划线，数字允许作为常量名。

常量名的所有字母必须大写。

常量中的单词必须以下划线分隔，例如可以这样 EMBED\_SUPPRESS\_EMBED\_EXCEPTION 但不许这样 EMBED\_SUPPRESSEMBEDEXCEPTION。

常量必须通过 "const" 定义为类的成员，强烈不鼓励使用 "define" 定义的全局常量。

1. **编码风格**

## 1. PHP 代码划分（Demarcation）

PHP 代码总是用完整的标准的 PHP 标签定界：

<?php

?>

短标签（ ）是不允许的，只包含 PHP 代码的文件，不要结束标签。

## 2. 字符串

### 2.1. 字符串文字

当字符串是文字(不包含变量)，应当用单引号（ apostrophe ）来括起来：

$a = 'Example String';

### 2.2. 包含单引号（'）的字符串文字

当文字字符串包含单引号（apostrophe ）就用双引号括起来，特别在 SQL 语句中有用：

$sql = "SELECT `id`, `name` from `people` WHERE `name`='Fred' OR `name`='Susan'";

在转义单引号时，上述语法是首选的，因为很容易阅读。

### 2.3. 变量替换

变量替换有下面这些形式：

$greeting = "Hello $name, welcome back!";

$greeting = "Hello {$name}, welcome back!";

为保持一致，这个形式不允许：

$greeting = "Hello ${name}, welcome back!";

### 2.4. 字符串连接

字符串必需用 "." 操作符连接，在它的前后加上空格以提高可读性：

$company = 'Zend' . ' ' . 'Technologies';

当用 "." 操作符连接字符串，鼓励把代码可以分成多个行，也是为提高可读性。

## 3. 数组

### 3.1. 数字索引数组

索引不能为负数

建议数组索引从 0 开始。

当用 array 函数声明有索引的数组，在每个逗号的后面间隔空格以提高可读性：

$sampleArray = array(1, 2, 3, 'Zend', 'Studio');

可以用 "array" 声明多行有索引的数组，在每个连续行的开头要用空格填补对齐：

$sampleArray = array(1, 2, 3, 'Zend', 'Studio',

$a, $b, $c,

56.44, $d, 500);

### 3.2. 关联数组

当用声明关联数组，array 我们鼓励把代码分成多行，在每个连续行的开头用空格填补来对齐键和值：

$sampleArray = array('firstKey' => 'firstValue',

'secondKey' => 'secondValue');

## 4. 类

### 4.1. 类的声明

用命名约定来命名类。

花括号可以从类名下一行或者该行开始

每个类必须有一个符合 PHPDocumentor 标准的文档块。

类中所有代码必需缩进。

每个 PHP 文件中只有一个类。

放另外的代码到类里允许但不鼓励。在这样的文件中，用两行空格来分隔类和其它代码。

下面是个可接受的类的例子：

/\*\*

\* Documentation Block Here

\*/

class SampleClass

{

// 类的所有内容

// 必需缩进

}

### 4.2. 类成员变量

必须用 变量名约定来命名类成员变量。

变量的声明必须在类的顶部，在方法的上方声明。

不允许使用 var，要用 private、 protected 或 public。

## 5. 函数和方法

### 5.1. 函数和方法声明

必须用函数名约定来命名函数。

在类中的函数必须用 private、 protected 或 public 声明它们的可见性。

象类一样，花括号从函数名的下一行开始或者该行开始

下面是可接受的在类中的函数声明的例子：

/\*\*

\* Documentation Block Here

\*/

class Foo

{

/\*\*

\* Documentation Block Here

\*/

public function bar()

{

// 函数的所有内容

// 必需缩进

}

}

注： 传址（Pass-by-reference）是在方法声明中允许的唯一的参数传递机制。

/\*\*

\* Documentation Block Here

\*/

class Foo

{

/\*\*

\* Documentation Block Here

\*/

public function bar(&$baz)

{}

}

传址在调用时是严格禁止的。

返回值不能在圆括号中，这妨碍可读性而且如果将来方法被修改成传址方式，代码会有问题。

/\*\*

\* Documentation Block Here

\*/

class Foo

{

/\*\*

\* WRONG

\*/

public function bar()

{

return($this->bar);

}

/\*\*

\* RIGHT

\*/

public function bar()

{

return $this->bar;

}

}

### 5.2. 函数和方法的用法

函数的参数应当用逗号和紧接着的空格分开，下面可接受的调用的例子中的函数带有三个参数：

threeArguments(1, 2, 3);

为了提高可读性,大于5个参数方法或者函数应该使用数组来进行参数传递,允许同时使用数组+非数组的方式

传址方式在调用的时候是严格禁止的，参见函数的声明一节如何正确使用函数的传址方式。

带有数组参数的函数，函数的调用可包括 "array" 提示并可以分成多行来提高可读性，同时，书写数组的标准仍然适用：

threeArguments(array(1, 2, 3), 2, 3);

threeArguments(array(1, 2, 3, 'Zend', 'Studio',

$a, $b, $c,

56.44, $d, 500), 2, 3);

## 6. 控制语句

### 6.1. if/else/elseif

在圆括号里的条件语句，操作符必须用空格分开，鼓励使用多重圆括号以提高在复杂的条件中划分逻辑组合。

前花括号必须和条件语句在同一行，后花括号单独在最后一行，其中的内容用四个空格缩进。

if ($a != 2) {

$a = 2;

}

对包括"elseif" 或 "else"的 "if" 语句，和 "if" 结构的格式类似， 下面的例子示例 "if" 语句， 包括 "elseif" 或 "else" 的格式约定：

if ($a != 2) {

$a = 2;

} else {

$a = 7;

}

if ($a != 2) {

$a = 2;

} elseif ($a == 3) {

$a = 4;

} else {

$a = 7;

}

"elseif" 是允许的但强烈不鼓励，我们支持 "else if" 组合。

### 6.2. switch

在 "switch" 结构里的控制语句在条件语句的圆括号前后必须都有一个单个的空格。

"switch" 里的代码必须有缩进，在"case"里的代码再缩进。

switch ($numPeople) {

case 1:

break;

case 2:

break;

default:

break;

}

switch 语句应当有 default。

注： 有时候，在 falls through 到下个 case 的 case 语句中不写 break or return 很有用。 为了区别于 bug，任何 case 语句中，所有不写 break or return 的地方应当有一个 "// break intentionally omitted" 这样的注释来表明 break 是故意忽略的。

## 7. 注释文档

### 7.1. 格式

所有文档块 ("docblocks") 必须和 phpDocumentor 格式兼容，phpDocumentor 格式的描述超出了本文档的范围，关于它的详情，参考：http://phpdoc.org/。

所有类文件必须在文件的顶部包含文件级 （"file-level"）的 docblock ，在每个类的顶部放置一个 "class-level" 的 docblock。下面是一些例子：

### 7.2. 类

每个类必须至少包含这些 phpDocumentor 标签：(参考)

/\*\*

\* 类的简述

\*

\* 类的详细描述 （如果有的话）... ...

\*

\* @author abc

\* @version Release: @package\_version@

\*/

### 7.4. 函数

每个函数，包括对象方法，必须有最少包含下列内容的文档块（docblock）：

函数的描述

所有参数

所有可能的返回值

因为访问级已经通过 "public"、 "private" 或 "protected" 声明， 不需要使用 "@access"。

如果函数/方法抛出一个异常，使用 @throws 于所有已知的异常类：

@throws exceptionclass [description]

## 8. SQL 语句

### 8.1. 关键字

SQL的关键字(如SELECT ,UPDATE ,WHERE ORDER BY ,DESC,AND ,OR,GROUP,HAVING,LIMIT,AS等)全部大写,如:

$sql="SELECT count(\*) AS total FROM cfwztg\_user ";

### 8.2. 换行

SELECT/FROM/WHERE/ORDER BY/GROUP BY等子句必须另其一行写;

SELECT 子句内容如果只有一项，与SELECT 同行写.

如:

$sql="SELECT \* FROM cfwztg\_user

ORDER BY id DESC LIMIT $startpos,$eachpage";