编码规范文档

ThinkSNS V3

智士软件（北京）有限公司

ZhiShiSoft (Beijing) Co,. Ltd.

修订记录

日期	修订版本	描述	作者
2012/8/26	1.0	新编写	韦心红

目录

1 文档编写目的与简介 4

2 编码规范 4

2.1 文件格式 4

2.2 命名约定 5

2.3 编码风格 8

1 文档编写目的与简介

本文描述系统编码规范，为开发及二次开发人员提供依据和参考。

2 编码规范

2.1 文件格式

1. 对于只含有 php 代码的文件，我们将在文件结尾处忽略掉 "?>" 。这是为了防止多余的空格或者其它字符影响到代码。

例如：

```
<?php
$foo = 'foo';
```

2. 缩进应该能够反映出代码的逻辑结果，尽量使用四个空格，禁止使用制表符TAB，因为这样能够保证有跨客户端编程器软件的灵活性。

例如：

```
if (1 == $x) {
    $indented_code = 1;
    if (1 == $new_line) {
        $more_indented_code = 1;
    }
}
```

3. 变量赋值必须保持相等间距和排列。

例如：

```
$variable = 'demo';
$var      = 'demo2';
```

4. 每行代码长度应控制在80个字符以内，最长不超过120个字符。因为 linux 读入文件一般以80列为单位，就是说如果一行代码超过80个字符，那么系统将为此付出额外操作指令。这个虽然看起来是小问题，但是对于追求完美的程序员来说也是值得注意并遵守的规范。

5. 每行结尾不允许有多余的空格。

2.2 命名约定

1. 类文件都是以“.class.php“为后缀，且类文件名只允许字母，使用驼峰法命名，并且首字母大写，例如：DbMysql.class.php 。

2. 配置和函数等其他类库文件之外的文件一般是分别以“.inc.php“和”.php“为后缀，且文件名命名使用小写字母和下划线的方式，多个单词之间以下划线分隔，例如 config.inc.php ， common.php，install_function.php 。

3. 确保文件的命名和调用大小写一致，是由于在类Unix系统上面，对大小写是敏感的。

4. 类名和文件名一致（包括上面说的大小写一致），且类名只允许字母，例如 UserAction类的文件命名是UserAction.class.php， InfoModel类的文件名是InfoModel.class.php 。

5. 控制器类以Action为后缀，例如 UserAction、InfoAction ，模型类以Model为后缀，例如 UserModel、InfoModel ，其他类也分别以相应分类为后缀，例如Service 、Widget。

6. 方法名只允许由字母组成，下划线是不允许的，首字母要小写，其后每个单词首字母要大写，即所谓的“驼峰法命名”规则 ,且越详细越好 ,应该能够描述清楚该方法的功能 ,例如switchModel、findPage。

7. 属性的命名只允许由小写字母和下划线组成，且建议用描述性的变量的命名，越详细越好。

8. 对于对象成员的访问，我们必须始终使用 “get” 和 “set” 方法。例如：

```
class Foo
{
    protected $test_obj;
    public function getTestObj()
   {
     return $this->test_obj;
    }
    public function setTestObj($test_obj)
   {
     $this->test_obj = $test_obj;
    }
}
```

9. 当类成员方法被声明为 private 时 ,必须以单下划线 "_"为开头 ;被声明为 protected 或public 时，不以下划线 "_" 为开头，而直接以小写字母开头。例如 ：

```
class Foo
{
   private function _example()
   {
     // ...
   }
    protected function example()
```

```
    {
     // ...
    }
    public function example()
    {
     // ...
    }
}
```

10. 如果我们需要把一些经常使用的方法定义为全局函数，那么应该把它们以静态 (static) 的形式定义在类中。例如：

```
class Think
{
    // ...
    static public function autoLoad($classname)
    {
     // ...
    }
}
```

11. 被声明为 private 的类成员属性必须以单下划线 "_" 作为开头；被声明为 protected 或 public 的类成员属性不以下划线 "_" 为开头，而直接以小写字母开头。

12. 函数的命名只允许由字母组成，下划线是不允许的，首字母要小写，其后每个单词首字母要大写，即所谓的 “驼峰法命名” 规则，且越详细越好，应该能够描述清楚该函数的功能，例如 getUserName、getUserInfo。

13. 当方法或函数参数不一定需要被赋值的时候，用 "null" 来代替 "false" 作为函数参数的默认值，除非该参数是 boolean 值。

14. 变量只允许由小写字母和下划线组成，且建议用描述性的变量的命名，越详细越好，以至于像 $i 或 $n 等等都是不鼓励使用的。

15. 类中的常量 constant 和全局范围内常量define，只能由大写字母和下划线组成，各个单词之间以下划线分割。

16. boolean 值和 null 值都采用小写。

2.3 编码风格

1. php 代码必须以完整的形式来定界（ <?php … ?> ），即不要使用php 短标签（ <? … ?> ），且保证在关闭标签后不要有任何空格。

2. 当一个字符串是纯文本组成的时候（即不含有变量），则必须总是以单引号（ ' ）作为定界符。例如：

```
$a = 'Example String';
```

3. 变量替换中的变量只允许用 $+变量名 的形式。例如：

```
$greeting = "Hello $name, welcome back!";   // 允许
$greeting = "Hello {$name}, welcome back!"; // 允许
$greeting = "Hello ${name}, welcome back!"; // 不允许
```

当用点号 "." 连接各字符串的时候，字符串与点号间必须用一个空格隔开，且允许把它分割成多行以增强可读性。在这种情况下，点号 "." 必须与等于号 "=" 对齐。例如：

```
$sql = "SELECT `id`, `name` " . " FROM `people` "
    . "WHERE `name` = 'Susan' "
     . "ORDER BY `name` ASC ";
```

当用 array 类型符号来构造数组的时候 ,必须在每个逗号之后加上一个空格来增强可读性。例如：

```
$sampleArray = array(1, 2, 3, 'Think', 'SNS');
```

4. 当使用 array 类型符声明关联数组的时候，我们鼓励把它分成多个行，只是我们必须同时保证每行的键与值的对齐，以保持美观。例如：

```
$sample_array = array(
          'firstKey'  => 'firstValue',
          'secondKey' => 'secondValue'
        );
```

5. 大括号的开始必须在类名的下一行顶格。例如：

```
class Think
{
   // ...
}
```

6. 类中的所有代码都必须用四个空格来进行缩进，不允许使用制表符Tab。

7. 每个 php 文件只允许声明一个类。在类文件里面写其它代码是允许的，但并不鼓励这样做。假如真要附加代码的话，必须用空行来分隔。

8. 任何类变量的声明都必须放在类顶部，先于任何函数的声明。

9. 不允许用 var 符号来声明变量，类成员变量必须以 private，protected 和 public 来声明。其次，把类成员声明为 public 而直接引用虽然是允许的，但通常更好的方法是使用 get 和 set 方法来访问类成员。

10. 方法必须总是用 private，protected 或者 public 来声明其作用域。

11. 静态 static 方法应该声明其作用域，且不应该再被声明为 private 私有，而应该为 protected 或者public ，如果只是不想被子类继承，则应该用 final 声明它们。

12. 函数或方法的初始大括号应该在函数声明的下一行顶格。例如：

```
// 允许
function getUserName()
{
    // ...
}
// 不允许
function getUserName() {
    // ...
}
```

13. 在函数或方法名与参数括号之间不允许出现多余的空格。例如：

```
// 允许
function getUserName()
{
    // ...
}
```

```
// 不允许
function getUserName()
{
    // ...
}
```

14. 引用只允许定义在函数参数中，实时传递引用是禁止的。例如：

```
// 引用定义在函数参数-允许的
function defineRefInMethod(&$a)
{
    $a = 'a';
}
defineRefInMethod($b);
echo $b; // 'a'
// 实时传递引用-禁止的
function callTimePassRef($a)
{
    $a = 'a';
}
callTimePassRef(&$c);
echo $c; // 'a'
```

15. 函数或方法返回值不可以用括号包住，不然会降低可读性，而且假如以后函数修改为返回引用的话，这将会抛出一个异常。

16. 在模块设计中，推荐使用类型提示。例如：

```
class Foo
{
    public function foo(SomeInterface $object)
   {
    }
    public function bar(array $options)
  {
    }
}
```

17. 函数和方法参数必须用逗号+空格来分隔。

18. 对于参数为数组的函数，参数中的数组应该分成多行以增强可读性。例如：

```
threeArguments(array(1, 2, 3), 2, 3);
threeArguments(array(1, 2, 3, 'Think',
                    'SNS', $a, $b, $c,
                    56.44, $d, 500), 2, 3);
```

19. 基于"if", "else"和"else if"的条件控制里，我们必须用空格间隔开语句和括号，大括号的开始 "{" 必须与条件控制语句位于同一行，结束 "}" 必须总是独占一行且顶格，控制流程内容必须用四个空格进行缩进，且不使用"elseif"。

```
if ($condition) {
    // ...
} else if ($_condition) {
```

```
    // ...
} else {
    // ...
}
```

20. 在条件控制语句的条件括号内，必须用空格将操作符与其它元素隔开。如果遇到很长的逻辑判断，则鼓励用内嵌括号来分割各个逻辑。例如：

```
if (($a != 2) and ($b == 1)) {
    $a = $b;
}
```

21. "switch" 条件控制语句中，必须用空格将待测参数与其它元素分隔开。例如：

```
switch ($num) {
    // ...
}
```

22. "switch" 语句的内容必须以四个空格缩进，"case" 条件控制的内容必须再加四个空格进行缩进。例如：

```
switch ($indentedSpaces) {
    case 2:
        echo "错误";
        break;
    case 4:
        echo "正确";
        break;
```

```
    default:
        break;
}
```

23. 在 "switch" 语句中应该总是包括 "default" 控制。

24. 有时候我们需要在 "case" 语境中省略掉 "break" 或 "return" ，这个时候我们必须为这些 "case" 语句加上 "// 此处无break" 注释。例如：

```
switch ($numPeople) {
case 1: // 此处无break
case 2:
    break;
default:
    break;
}
```