PHP编码规范说明书

(内部资料 请勿外传)

|  |  |  |  |
| --- | --- | --- | --- |
| **编 写：** | 刘凯 | **日 期：** | 2013-11-29 |
| **检 查：** |  | **日 期：** |  |
| **审 核：** |  | **日 期：** |  |
| **批 准：** |  | **日 期：** |  |

**版权所有 不得复制**

**文档变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 |  | 刘凯 | V1.0 | 2013-11-29 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 引言

## 编写目的

本需求说明书编写的目的是定义说明系统的组成，功能模块，包括不同点的功能描述，使用范围，相关局限等，为软件编程和系统维护提供借鉴。本说明书的预期读者为需求开发人员、需求评审人员，系统设计人员、软件开发人员、软件测试人员和项目评审人员。

程序员可以了解任何代码，弄清程序的状况

新人可以很快的适应环境   
 防止新接触php的人出于节省时间的需要，自创一套风格并养成终生的习惯   
 防止新接触php的人一次次的犯同样的错误   
 在一致的环境下，人们可以减少犯错的机会   
 程序员们有了一致的敌人

## 参考资料

列出有关资料的名称、作者、文件编号或版本等。参考资料包括：

a．需求说明书、架构设计说明书等；

b．本项目的其他已发表的文件；

c．引用文件、资料、软件开发标准等。

|  |  |  |  |
| --- | --- | --- | --- |
| 资料名称 | 作者 | 文件编号、版本 | 资料存放地点 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 文件格式

## 纯php文件忽略”?>”

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';*

## 代码缩进

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;    }}

## 变量赋值

变量赋值必须保持相等间距和排列。  
例如：  
$variable = 'demo';$var      = 'demo2';

## 每行代码长度

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。

## 结尾禁止有空格

每行结尾不允许有多余的空格。

# 命名规格

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，每个单词都用小写字母组成，且越详细越好，应该能够描述清楚该方法的功能，例如db\_connect、db\_check\_database\_support。（下划线是不允许的，首字母要大写，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，例如tablePrefix、tableName 。
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 $\_testObj;  
       public function getTestObj()*

*{  
        return $this->\_testObj;  
    }  
    public function setTestObj($testObj)*

*{  
        $this->testObj = $\_testObj;  
    }  
}*

1. 当类成员方法被声明为 private 时，必须分别以双下划线 "\_\_"为开头；被声明为 protected 时，必须分别以单下划线 "\_" 为开头；一般情况下的方法不含下划线。例如 ：  
   *class Foo  
   {  
       private function \_\_example()*

*{  
        // ...  
    }  
    protected function \_example()*

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

*{  
        // ...  
    }  
}*

1. 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)*

*{  
        // ...  
    }  
}*

1. 被声明为 private的类成员属性必须由双下划线 "\_\_" 作为开头；被声明为 protected 的类成员属性必须由下划线 "\_" 作为开头；而声明为 public 的成员属性则在任何时候都不允许含有下划线。
2. 函数的命名使用小写字母和下划线的方式，且越详细越好，应该能够描述清楚该函数的功能，例如 get\_client\_ip 。
3. 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。
4. 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。

函数参数用p\_开头，如$p\_hostname

局部变量用t\_开头，如$t\_db\_type

全局变量用g\_开头，如$g\_db\_connected

1. 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。如"HOME\_URL";
2. boolean 值和 null 值都采用小写。

# 代码重构

1. 函数或者方法体内的代码尽量控制在一个屏幕内。
2. 类中不使用的方法随机删除。
3. 修改别人的类中方法，要签名。
4. 在每个模块内写个readme文件(用于比较复杂业务的说明或代码说明)。
5. 尽量让每个类做自己的事，每个函数做一件事。

# 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';

1. 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：$sampleArray = array(1, 2, 3, 'Think', 'SNS');
2. 当使用 array 类型符声明关联数组的时候，我们鼓励把它分成多个行，只是我们必须同时保证每行的键与值的对齐，以保持美观。例如：  
   *$sampleArray = array(  
                      'firstKey'  => 'firstValue',  
                      'secondKey' => 'secondValue'  
                  );*
3. 大括号的开始必须在类名的下一行顶格。例如：  
   *class Think  
   {  
       // ...  
   }*
4. 类中的所有代码都必须用四个空格来进行缩进。
5. 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。
6. 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。
7. 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。
8. 方法必须总是用 private，protected 或者 public 来声明其作用域。
9. 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。
10. 函数或方法的初始大括号应该在函数声明的同一行。例如：   
    *function get\_client\_ip(){  
        // …  
    }*
11. 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：  
    *function get\_client\_ip(){  
        // …  
    }*
12. 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'*
13. 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。
14. 鼓励尽量使用类型提示，特别是在模块设计中。例如：  
    *class Foo  
    {  
        public function foo(SomeInterface $object)*

*{  
    }  
    public function bar(array $options)*

*{  
    }  
}*

1. 函数和方法参数必须用逗号+空格来分隔。
2. 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, $d, 500), 2, 3);*
3. 基于"if", "else"和"else if"的条件控制里，我们必须用空格间隔开语句和括号，大括号的开始 "{" 必须与条件控制语句位于同一行，结束 "}" 必须总是独占一行且顶格，控制流程内容必须用四个空格进行缩进，且不使用"elseif"。  
   *if ($condition) {  
       // ...  
   } else if ($\_condition) {  
       // ...  
   } else {  
       // ...  
   }*
4. 在条件控制语句的条件括号内，必须用空格将操作符与其它元素隔开。如果遇到很长的逻辑判断，则鼓励用内嵌括号来分割各个逻辑。例如：  
   *if (($a != 2) and ($b == 1)) {  
       $a = $b;  
   }*
5. "switch" 条件控制语句中，必须用空格将待测参数与其它元素分隔开。例如：  
   *switch ($num) {  
       // …  
   }*
6. "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 ($indentedSpaces) {  
       case 2:  
           echo "错误";  
           break;  
       case 4:  
           echo "正确";  
           break;  
       default:  
           break;  
   }*
7. 在 "switch" 语句中应该总是包括 "default" 控制。
8. 有时候我们需要在 "case" 语境中省略掉 "break" 或 "return" ，这个时候我们必须为这些 "case" 语句加上 "// 此处无break" 注释。例如：  
   *switch ($numPeople) {  
    case 1: // 此处无break*

*case 2:  
     break;  
 default:  
      break;  
}*

# 常用代码

## 判断"=="时，把常量放在前面

$a = 1;

if($a == 1){

echo '$a == 1';

}

改为：

$a = 1;

if(1 == $a){

echo '$a == 1';

}

## 查找表法

/\*错误码:4,5,7,8的时候返回状态1,错误码是1,3,6返回状态2\*/

$error = 4;

$state = 0;

if($error == 4 || $error == 5 || $error == 7 || $error == 8){

$state = 1;

}

if($error == 1 || $error == 3 || $error == 6){

$state = 2;

}

echo "$state \n";

改为：

/\*错误码:4,5,7,8的时候返回状态1,错误码是1,3,6返回状态2\*/

$error = 4;

$state = 0;

$arr = array(4 => 1, 5 => 1, 7 => 1, 8 => 1, 1 => 2, 3 => 2, 6 => 2);

isset($arr[$error]) && ($state = $arr[$error]);

echo "$state \n";