**PHP编码格式规范**

|  |
| --- |
| 版本：1.0  历史：  \* v1.0 2014.01.01 |

为什么有编码格式规范？

* 格式化代码为了让代码更容易阅读
* 格式化代码有益于改善代码质量
* 格式化代码是使代码被可持续维护的保障
* 格式化代码是一个优秀的程序员认真负责态度的体现
* 格式化代码是对阅读你代码读者礼貌的体现

因此，我们需要你必须这么做…..

缩进(Indentation)

用什么来缩进？缩进多少？

* 缩进让代码更容易阅读
* 使用制表符(Tab)缩进
* 请将制表符的缩进位置设置为4

|  |
| --- |
| function func()  {  if ($i > 10)  {  if ($j < 100)  {  while ($k < 1000)  {  ……  }  }  }  } |

空白字符(Wrapping Lines)

* 函数名后的” ()”无空格
* 函数的多个参数之间加空格，括号前后无空格

|  |
| --- |
| ***正确:*** |
| list($user, $pass) = explode(":", "Iuser:Nopass") |
| ***错误:*** |
| List ($user, $pass) = explode (":", "Iuser:Nopass") list($user,$pass) = explode(":","Iuser:Nopass") list( $user, $pass) = explode( ":", "Iuser:Nopass") list($user, $pass ) = explode(":", "Iuser:Nopass" ) |

* 赋值操作符、比较操作符、算术操作符、逻辑操作符、位域操作符，如“=”、“+=” “>=”、“<=”、“+”、“\*”、“%”、“&&”、“||”、“<<”,“^”等二元操作符的前后加空格
* 一元运算符、地址运算符“!”、“~”、“++”、“--”、“&”等前后无空格
* “->”， “::”操作符前后不加空格

|  |
| --- |
| ***正确:*** |
| $c = $a + $b;  $c++;  $book = new &book;  $book->getName(); |
| ***错误:*** |
| $c=$a+$b;  $c =$a +$b;  $c ++;  $book = new & book;  $book -> getName(); |

* 数组以=>分隔键值对，则写成单行，上下的等号对齐；
* 只要值的数组写成一行，则每项之间应该有空格

|  |
| --- |
| ***正确:*** |
| $c = array(  “user” => ‘user’,  “passwd” => ‘passwd’,  );  $c = array(“bank”, “card”); |

* 行末不要保留没有意义的空格符。Zend设置：Tool->Preferences->Editing->Strip trailing spaces on save

换行结束符(Lines)

* 为保持语句清晰，每一行只写一个语句
* 两个片段(section)之间，使用空行隔开，函数/方法之间使用一个空行隔开，类声明之间使用两个空行隔开

|  |
| --- |
| interface MyInterface  {  public function ifc1(){}  public function ifc2(){}  }  class Foo extends MyInterface  {  private function foo1()  {  ……  }  public function foo2()  {  ……  }  } |

* php代码起始格符，请严格使用”<?php “作为程序的第一行开始

括号(Parenthesis)

**大括号(花括号) ”{” 怎么对齐？**

* 将大括号放置在关键词下方的相同的位置

|  |
| --- |
| ***正确:*** |
| foreach ($mydata as $key => $value)  {  ……  } |
| ***错误:*** |
| foreach ($mydata as $key => $value){  ……  } |

注释(Comments)

**单行怎么注释？**

在语句结尾用双反斜杠”// “注释

**多行怎么注释？**

多行注视以”/\*”符号开头，以”\*/ “符号作为注释结束符

|  |
| --- |
| /\*  这是一段注释，这里需要描述很多内容:  从前有座山，山上有个老和尚  有一天老和尚对小和尚说…  \*/ |

**文件、类、函数的头注释怎么写？**

符合phpdoc标准 的写法，参见《phpdoc注释规范指南》

|  |
| --- |
| /\*\*  \* Copyright (c) 2009, 新浪网支付中心  \* All rights reserved.  \* 文件名称： index.php  \* 摘 要： 首页程序  \* 作 者： xxx  \* 版 本： 1.0  \* 修改日期:　2009.2.21  \*/ |

* 修改代码同时修改相应的注释，以保证注释与代码的一致性，没用的注释要及时删除。
* 注释应当准确、易懂，防止注释有二义性。
* 尽量避免在注释中使用缩写，特别是不常用缩写
* 当代码比较长，特别是有多重嵌套时，应当在一些段落的结束处加注释，便于阅读

**Todo与修改的注释**

* 在未完成或需要深入思考的地方添加To do注释，便于自己与他人补充、理解
* 程序被修改的地方，写上修改人和时间

命名(Give name)

单字之间不以空格断开（例：camel case）或用连接号（-，例：camel-case）、底线（\_，例：camel\_case）连结，有两种格式：

小驼峰式命名法（lower camel case）：

第一个单字以小写字母开始；第二个单字的首字母大写，例如：firstName、lastName。

大驼峰式命名法（upper camel case）：

每一个单字的首字母都采用大写字母，例如：FirstName、LastName、CamelCase，也被称为 Pascal 命名法

**为了避免 成员命名偏好的不同，形成能共同遵守的程序规范，约定如下清晰、易懂、易执行的命名规范：**

1. **类采用大驼峰命名法，如：DBFariy，ZendDB**
2. **函数名/类的方法，以大写字母作为词的分隔，其他的字母均使用小写，单词的首个字母使用小写，不使用下划线，如：DB\_Fairy::makeSelectSql()**
3. **变量名全部小写，下划线连接，如：$g\_cgi**
4. **全局变量：所有字母都使用大写，以下划线分隔每个单词，如：$WEB\_SITE,define(‘WEB\_ROOT’,’name’);**
5. **数组变量：以变量的命名方式一致，以\_array作为后缀，如：$serise\_array();**
6. **对象变量：以变量的命名方式一致，以\_object作为后缀，如：$serise\_object = new SeriseParent();**
7. **静态变量：以变量的命名方式一致，以\_static作为后缀，如：$serise\_static= ‘’;**
8. **DBFairy类对应DBFairy.php；HttpUtil类对应HttpUtil.php文件**
9. **采用英文单词或其组合，便于记忆和阅读，切忌使用汉语拼音来命名**

**如何为文件命名？**

* 为保证在线服务的安全，禁止使用以非”.php”为后缀的文件名，例如.inc .conf等，请以 .inc.php or .conf.php替代；更不允许有程序的打包文件，如.tar.gz
* 文件名全部用小写字母
* 模板名和PHP程序一一对应：login.html
* 类文件的命名：clasname.class php
* 配置文件的命名：db.conf.php

**如何为类命名？**

* 类名应该和文件名相匹配，例如对于类HelloWord，相应的文件名应为helloworld.class.php

**如何为函数/方法命名？**

* 通常方法一般为一个动作或行为动词，函数/方法的命名采用下划线

|  |
| --- |
| function run()  function run\_fast()  function get\_background() |

* 尽量用有意义，描述性的词语来命名

用check\_errors()代替error\_check()，用dump\_data2file()代替data\_file()。

* 有时前缀名是有用的：

is - 含义为问一个关于某样事物的问题。无论何时，当人们看到Is就会知道这是一个问题。

get - 含义为取得一个数值。

set - 含义为设定一个数值

例如：is\_hit\_retry\_limit

* 内部成员函数命名应该是以 “\_”开始，如：function \_getValue()

|  |
| --- |
| function \_is\_user\_ticket() |

**如何为变量命名？**

* 用有意义的，描述性的词语来命名变量
* 别用不常见的缩写。用name, address, salary等代替 nam, addr, sal
* 每个项目的初期，应该约定常用变量名，参数名等
* 全局变量以”g\_” 开头
* 别使用单个字母的变量象i, n, x 等. 使用 index, temp等 ，用于循环迭代的变量例外：

|  |
| --- |
| for ($i = 0; $i < count; $i++)  {  ...  } |

**如何为常量命名？**

常量全部使用大写字母和下滑线组成,常量的名称中不允许出现小写字母

文件布局(File structure)

|  |
| --- |
| |  |---- user（自定义的应用程序目录）  | |  | |---- login.php  | |  | |---- database.interface.php  |  |---- \_protect（系统文件目录）  | |  | |---- corelib  | |---- applib  | |---- system  | |---- conf  | |---- corelib  | |---- tmpl  |---- \_static（静态文件目录）  | |---- css  | |---- img  | |---- js  | |---- swf  | |

我们还有些建议….

**使用什么开发工具？**

我们建议使用Zend studio 7.1以上版本

**字符串的连接方式？**

|  |
| --- |
| A、$myConcat = "Hello {$name}, welcome back!";  B、$myConcat = "Hello" . $name . "welcome back!";  C、$myConcat = "Hello $name welcome back! "; |

我们更推荐使用第一种方式，它看起来更简洁。注意变量名用 { } 括起来

**函数参数的顺序？**

将有默认值或者可以不确定需要传递的参数放置在参数的后面，必须的参数靠前

**参数最多个数有限制吗？**

避免函数有太多的参数，参数个数尽量控制在5个以内。

**循环规律？**

如果循环体内存在逻辑判断，并且循环次数很大，宜将逻辑判断移到循环体的外面

**控制语句的逻辑判断的格式？**

|  |
| --- |
| A、if (NULL === $expression)  B、if ($expression === NULL) |

我们更推荐使用第一种方式，能有效减少笔误情况的发生

**数组的赋值**

|  |
| --- |
| A、$arr = array(  “id1” => 1,  “id2” => 2,  );  B、  $arr[‘id1’] = 1;  $arr[‘id2’] = 2; |

在给数组赋多个值的时候，推荐使用第一种方式