[PHP书写规范 PHP Coding Standard](http://www.cnblogs.com/sink_cup/archive/2011/07/07/php_coding_standard.html)

PHP书写规范  
作者：sink <sink.cup@gmail.com>  
最后修改：2011-7-13  
  
参考资料：  
PHP Manual  
http://www.php.net/manual/zh/language.oop5.basic.php  
PEAR Coding Standards  
http://pear.php.net/manual/en/standards.php  
C++ Coding Standard  
http://www.possibility.com/Cpp/CppCodingStandard.html  
Google C++ Style Guide  
http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml  
Code Conventions for the Java  
http://www.oracle.com/technetwork/java/codeconvtoc-136057.html  
  
制定规范时要注意：  
1、一般不要出现2个都行的情况。  
比如tab和4个空格都行，结果导致代码混乱。  
  
通用原则：  
1、语义化  
看到名字，就知道意思。  
  
2、通用前缀  
is表示是否、get表示读、set表示写。is后面优先跟形容词，而不是名词，比如是否多语言文字，应使用is\_multilingual，而不是is\_multilanguage。  
  
3、单数与复数  
参考js的函数命名规则：getElementById、getElementsByTagName、getElementsByName。  
例如：  
取我的多个好友的名字，应使用getFriendsName，而不是getFriendNames或者getFriendName  
取一个用户，是getUser  
取多个用户，是getUsers  
  
4、冗余后缀  
尽量不使用data、list、info后缀，除非特殊情况。  
比如，js的命名就很注意，使用getElementsByTagName而不是getElementsInfoByTagName。  
应该使用getFriends或者getFriendsUserId，而不是getFriendsList；应该使用getUser，而不使用getUserInfo或者getUserData。  
不过有时候很难避免，比如有2个函数，分别是取用户基本信息，和取用户详细信息。  
取用户基本信息：昵称、头像URI，函数名getUserBasic还是getUserBasicInfo？函数名以形容词结尾感觉不合适。待讨论。讨论结果：getUserBasicInfo合适。  
取用户详细信息：昵称、头像URI、签名、生日，函数名getUser没问题。  
  
5、含义模糊的类名、文件名、目录名  
每当使用common、util、functions、class、object、basic作为文件名时要慎重，由于这些词太通用，发展下去里面东西可 能越来越多，变成垃圾箱。要给这些起一个准确的名字，比如要做字符串处理的类，可以叫StringLib.php，放在lib目录里。  
  
6、lib、plugin与addon的区别  
有些类、函数算做lib、plugin还是addon。待讨论。讨论结果：目前增强函数算是Lib，以后再考虑plugin和addon。  
  
7、常用词汇  
优先使用URI，而不是URL。因为更严谨，新的命名开始使用URI。比如js的encodeURI，PHP的$\_SERVER['REQUEST\_URI']。  
deadline与TTL：deadline表示最后时刻，TTL表示存活时间。比如现在时间是1310449710，TTL是60秒，则deadline是1310449710 + 60 = 1310449770。  
  
类名：  
大写字母开头，驼峰命名。一般使用名词，比如配置解析类ConfigParser，而不是ParseConfig。  
与Java、C++一致。  
例如：class UserModel  
  
类的文件名：  
与类名相同。这与php autoload有关，为了autoload，类名总要很长。待讨论。讨论结果：遵守驼峰，也能实现自动类载入。  
与Java一致。  
例如：class UserModel的文件名为UserModel.php  
  
非类文件名：  
全小写，下划线分隔，不得使用空格。比如get\_user.php。  
  
目录名：  
全小写，下划线分隔，不得使用空格。比如model、www。  
  
函数名：  
小写字母开头，驼峰命名，例如：function addBlog()。  
与Java、C++一致。  
函数表示功能，即动作，所以动词优先，例如使用editBlog，而不用blogEdit。  
PHP内置函数由于历史原因，有多种风格，do\_something,something\_do,dosomething,比较新的函数用了doSomething，才与目前主流语言保持一致。  
比如：paser\_str、json\_encode、substr、fetchAll。  
历史原因可能无法改变，但我们能保证新的代码是严谨的，不要让自己成为历史原因。  
  
类中的函数：  
两个函数中间空一行。如果有时间的话，各个函数按英文字母排序，免得太混乱。  
例如：  
class BlogModel  
{  
    public function addBlog()  
    {  
  
    }  
      
    public function updateBlog()  
    {  
  
    }  
}  
  
文件注释：  
注释紧跟<?php下一行。注明作者。@version暂不需要写，因为svn提供了版本管理。  
格式按照PHPdoc的要求：http://manual.phpdoc.org/HTMLframesConverter/default/phpDocumentor/tutorial\_tags.author.pkg.html  
<?php  
/\*\*  
 \* blog的各种业务：添加、更新  
 \* @author sink  
 \*  
 \*/  
class BlogModel  
{  
  
}  
?>  
  
API注释：  
一定要写输入参数，和输出格式。写清楚正确时输出什么，错误时输出什么。  
否则别人无法使用。  
  
函数注释：  
一定要写输出格式。写清楚正确时输出什么，错误时输出什么。  
如果输入参数比较复杂，包含数组，看参数无法一目了然，则要写输入参数的注释。  
文档注释与函数之间不能有空行。  
如果函数内部步骤比较复杂，需要写“行内注释”。  
例如：  
/\*\*  
 \* 更新blog  
 \* @param int $id blog\_id  
 \* @param array $data array(  
    "content" => "", //内容  
    "tags" => "", //标签  
    "update\_time" => "", //更新时间  
 )  
  \* @return bool  
 \*/  
public function updateBlog($id,$data)  
{  
    step1 //第一步：asdf  
    step2 //第二步：qwer  
}  
  
URI：  
根据rfc1034国际标准的规定，域名中禁止出现下划线“\_”，域名不区分大小写。  
比如http://dl\_dir.qq.com/是错误域名。  
http://example.com与http://EXAMPLE.COM相同。  
所以优先在URI中使用全小写，GET的name小写，但是GET的值除外。  
比如  
http://www.google.com/?hl=zh-CN  
http://www.google.com/?hl=zh-cn  
URI中非参数的专有名词的缩写是否使用小写，有争议无定论。  
比如  
http://fedoraproject.org/zh\_CN/  
http://zh.wikipedia.org/zh-cn/  
http://code.google.com/intl/zh-CN/  
http://www.microsoft.com/en-us/  
语言文字代码是专有名词，ISO规定必须是减号，且建议地区使用大写。  
fedora的用法很奇怪，使用了自己制造的zh\_CN，而不是zh-CN。而且不建议在URI中使用下划线。  
wiki用了小写，google用了大写，微软用了小写。  
  
优先在URI中使用减号“-”，而不是下划线，GET的name除外。  
比如  
http://example.com/1-2-2  
http://example.com/?user\_id=123  
如果希望用户手动输入URI，则不要区分大小写，且优先使用小写，因为用户输入更方便。  
实际情况是：用户一般是手动输入域名，而不手动输入URI，因为URI很长。在这种情况下，URI小写是否有意义，如果使用 http://example.com/?userId=123，变量名就可以使用驼峰$userId = $\_GET['userId']，就能够和Java、C++保持一致，这样数据库也要驼峰命名。待讨论。讨论结果：使用?user\_id=123。  
  
变量：  
全小写，下划线分隔，例如：$user\_id。  
与Java、C++不一致。讨论结果：使用$user\_id。  
类的成员变量、函数的形参、类实例化成一个对象，都遵守变量的命名规则。  
原因：URI、数据库有小写惯例，从$\_GET、$\_POST中获得参数入库，所以用小写。  
PHP内置变量$\_GET、$\_POST使用下划线开头，全大写。自定义的变量无论多么重要，都不要使用下划线开头，以免将来与内置变量冲突。  
比如：不要使用$\_PUT、$\_DELETE。  
  
常量：  
全大写，下划线分隔。例如：const MEMCACHE\_TTL = 600;  
  
PHP短标签：  
使用<?php ?>，不使用短标签<? ?>。因为与xml冲突，且不利于部署。  
  
类大括号换行：  
可以采用大括号单独占一行，也可以大括号与别的放在一行，有争议无定论，待讨论。讨论结果：使用“同行”。  
class UserModel {  
  
}  
支持换行者：  
http://www.php.net/manual/zh/language.oop5.basic.php  
http://pear.php.net/manual/en/standards.classdef.php  
  
函数大括号换行：  
有争议无定论，待讨论。讨论结果：使用“同行”。  
function getUser() {  
  
}  
支持换行者：  
http://www.php.net/manual/zh/language.oop5.basic.php  
http://pear.php.net/manual/en/standards.funcdef.php  
  
if大括号换行：  
有争议无定论，待讨论。讨论结果：使用“同行”。  
例如：  
if(!empty($name))  
{  
  
}  
或者  
if(!empty($name)) { //确定  
  
}  
支持换行者：  
http://www.possibility.com/Cpp/CppCodingStandard.html#brace  
  
支持同行者：  
http://www.php.net/manual/zh/language.oop5.basic.php  
http://pear.php.net/manual/en/standards.control.php  
  
switch大括号换行：  
讨论结果：使用“同行”。  
switch (...) {  
    case 1:  
        ...  
        break;  
  
    default:  
}  
支持换行者：  
http://www.possibility.com/Cpp/CppCodingStandard.html#switch  
  
数组小括号换行：  
有争议无定论。讨论结果：使用“同行”。  
$user = array(  
    "id" => "123",  
    "name" => "user1",  
    "email" => "a@example.com",  
)  
支持同行者：  
http://pear.php.net/manual/en/standards.arrays.php  
  
数组内部换行：  
2维及以上数组的数组内部换行。  
如  
$user = array(  
    'id' => '123',  
    'name' => 'user1',  
    'email' => 'a@example.com',  
);  
1维数组内部不换行。讨论结果：1维数组内部不换行。  
如  
$users\_id = array('23','12','24');//确定  
  
数组最后的逗号：  
数组每一行最后要有逗号，这样方便以后添加。不过前端JSON最后不能有逗号，否则有的浏览器不支持，待讨论。讨论结果：都行，因为后端不用考虑IE前端。  
比如  
$user = array(  
    'id' => '123',  
    'name' => 'user1', //都行，优点：大数组，经常添加一行，方便。如果没有逗号，确实太难以添加了。  
);  
$user = array(  
    'id' => '123',  
    'name' => 'user1' //都行，优点：严谨，逗号表示分隔，最后一个不需要分隔。  
);  
  
单引号与双引号：  
优先使用单引号，当需要转义时使用双引号，变量不放在双引号中。这与JSON不同，JSON全是双引号，待讨论。讨论结果：优先使用单引号。  
比如：  
echo 'name is:' . $name . '.' . "\n";  
$user = array(  
    'id' => '123',  
);  
  
条件判断的大括号：  
必须有大括号，即使只有一行。  
正确：  
if(!empty($name)){  
    doSomething();  
}  
错误：  
if(!empty($name))  
    doSomething();  
  
回车换行：  
使用换行LF（\n，0a，Unix风格）。不使用CR+LF（Windows风格）。  
参考：http://zh.wikipedia.org/zh-cn/%E6%8F%9B%E8%A1%8C  
eclipse——》workspace——》New text file line delimiter——》Other：Unix  
  
编码：  
使用UTF-8 no BOM。不得使用Windows记事本进行保存，因为记事本是UTF-8 BOM CR+LF。  
eclipse——》workspace——》Text file encoding——》Other：UTF-8  
  
缩进：  
使用4个空格进行缩进，也可以采用tab进行缩进。讨论结果：4个空格。  
支持4个空格者：//确定  
http://www.oracle.com/technetwork/java/codeconventions-136091.html#262  
  
支持2个空格者：  
http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml#Spaces\_vs.\_Tabs  
  
支持3、4或8个空格者：  
http://www.possibility.com/Cpp/CppCodingStandard.html#indent  
  
要保证缩进正确，如果使用4个空格，一定不要出现5个空格或者11个空格。  
eclipse——》General——》Editor——》Text Editors——》show whitespace characters  
vim ~/.vimrc  
set expandtab  
set softtabstop=4  
set shiftwidth=4  
set autoindent  
  
HTTP协议缓存：  
文章使用Last Modified表示最后修改时间，不禁止缓存。  
header('Last Modified:Sat, 30 Oct 2010 13:21:21 GMT');  
需要用户登录的页面，禁止缓存。  
header('Cache-Control:max-age=0');  
header('Cache-Control:private');  
  
HTTP协议编码与mime：  
HTTP输出一定要声明编码与mime。charset与分号之间要有一个空格。小写utf-8还是大写UTF-8，尚未找到文档，待调研。  
比如  
header('Content-Type:application/json; charset=UTF-8');  
header('Content-Type:application/xml; charset=UTF-8');  
header('Content-Type:application/xhtml+xml; charset=UTF-8');  
header('Content-Type:text/plain; charset=UTF-8');  
header('Content-Type:text/html; charset=UTF-8');  
  
专有名词大小写：  
在类、函数、文件名、目录名等各种地方，不特殊对待专有名词，不采用全大写。讨论结果：使用小写。  
原因：专有名词难以界定，比如HTML、CSS、CRUD。而且全大写导致与驼峰冲突，比如页面助手类，全大写是HTMLHelper，不如HtmlHelper。  
支持不特殊处理：  
HTML是专有名词，但mime中就使用Content-Type:text/html，而不是text/HTML。  
例子：  
采用UserDb.php，而不是UserDB.php。