**PHP编码规范**

**请把规范变成习惯**

1. 文件结构

文件结构和分类包括文件层次、命名含义等方面的约束，清晰的结构有助于源码的维护和理解。有意义的命名也是清晰结构的一个要素。比如config，一般就放系统配置文件。

注意：有些细节要求是结合当前使用的conan框架来要求的（下同）。

1. 命名规范
2. 文件夹命名

文件夹命名一般采用小写英文字母，一般不超过20个字符，尽量言简意赅。常用的有images, inc, config, log, database, cache, upload等

1. 文件命名

与文件夹类似，一般都采用小写英文字母。

一般来说，controller里的文件名与其对应的model都保持一致，如果对应的view是一个文件，那么文件名也保持一致，如果对应的是一组页面，那么与文件夹名称保持一致。

1. 类名

类文件是比较特殊的一类文件，类名是一个名词，可以反映出类的功能，一般采用小写英文字符（首字母大写），要尽量使你的类名简洁而又描述清楚，但避免使用不常见的缩写词。

注意：类名一般与其文件名保持一致，类名包含文件目录（自动加载）的除外，如：

class Controller\_Plan{} //controller目录下的plan.php文件

class Model\_Plan{} //model目录下的plan.php文件

1. 函数命名

函数、方法名是个动词，采用大小写英文字母结合方式命名，首字母小写，其余单词首字母大写。通常每个方法都是执行一个动作的，所以对它的命名应该说清楚是干什么的，比如function setName(), checkForms(), exportData(), getContent()等。

注意：

1. controller里的action方法（会在url上出现的）一律以小写字母命名，比如aciton\_index(), action\_ajaxlist()等
2. model里的方法定义如下命名规则：

类本身的方法用get来取，比如getStatus();

数据库查询采用find（查询）、add（添加）、update（更新）、delete（删除）、count（计数）等开头，以下是常用的命名方式：

find()：根据主键查询一条记录

findAll()：根据主键查询所有记录

findField()：根据主键查询某字段

findByField()：根据其他字段查询一条记录

findAllByField()：根据其他字段查询所有记录

updateField()：根据主键更新某字段

count()：根据主键计数

countByField()：根据某字段计数

……

1. 常量

常量命名应该全部大写，单词间用下划线隔开，以免混淆，比如MAX\_WIDTH。

常用的常量如状态、类型命名规则如下：

STATUS\_NORMAL(正常)、 STATUS\_PAUSE(暂停)、 STATUS\_DELETE(删除)

TYPE\_TEXT(文字)、 TYPE\_PIC(图片)等

1. 变量

变量命名方式与函数命名类似，采用大小写字母结合的方式。比如$adType

变量名应简短且富于描述，尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i, j, m,n，它们一般用于整型。

其他：

（1）数组里面的key的value一律用k和v，如 foreach($arr as $k=>$v){}

（2）循环处理数组的变量命名为 r，如foreach($planlist as $r){}

1. 类变量

类里面的变量申明规则：

Public类型的遵循一般变量命名方式；

Protected、private类型的变量一般用下划线开头，比如protected $\_var;

1. 源文件编码规范
2. 引用

当需要引入其他文件时，引入语句一般位于文件头部，并说明引入文件的作用，如：

//数据库操作类

require\_once(‘db.php’);

1. 申明

一行一个申明，不要将不同类型变量的申明放在同一行内

只在代码块的开始处声明变量，尽量在声明局部变量的同时初始化

申明一个类的实例时，应该在类后面加括弧，如：

$db = new Database; //错误，这是一种兼容php老版的做法，现在没必要

$db = new Database(); //正确

1. 空行

一个方法的两个片段之间、一个方法内的两个逻辑之间、两个方法之间都应空行，用以提高代码可读性。必要的地方可以空两行，但避免更多。

注意：切忌每行代码之间都有空行。

1. 空格
2. 不使用空格的情况

--代码缩进时不直接用空格键敲出空格，而要使用Tab键实现缩进

注意：请在编码时养成良好习惯，而不是后续去修改。

1. 使用空格的情况

--函数中多个参数之间用空格隔开，如：getInfo($id, $name);

--if条件句中的比较运算符两端的条件表达式之间要空格，如：if ($a == $b);

--使用三元操作符时，注意赋值、比较运算符两端及三元操作符两端都需要留空格，如：

$foo = ($bar == $foo) ? $foo : $bar;

$foo = $bar ? $foo : $bar;

赋值如果是表达式的话要用括弧括起来，如：

$foo = $bar ? ($foo+2) : $bar;

1. 换行

尽量避免一行的长度超过80个字符，遇到这种情况时需要换行，换行的一般规则：

--在逗号后面断开

--在操作符前面断开

--新行应该与其同级的上一行对齐

--逻辑复杂时，尽量简化它，不要在同一行内出现多个逻辑运算

总之，换行后要保持逻辑、代码清晰

注意：切忌使用滚动条来看代码

1. 注释

写注释的原则：言简意赅、突出重点

不是所有函数、变量都需要写，也不是所有注释都要非常详细，该写的写清楚就行

1. 文件头注释

所有源文件在开头都应有注释，列出文件描述、作者、日期，如果是类的话加类名。如果对文件作了重要修改，也可以在此说明，并列出修改目的、修改人、修改日期等信息。比如：

/\*\*

\* 这里写类名或描述文件功能

\* @author Hexiaofang（请采用真实姓名或拼音）

\* @date 2011-02-16

\*/

1. 方法、函数注释

一般方法和函数注释包括功能描述、参数、返回值，最好能指明类型，比如：

/\*\*

\* 根据分类id获取新闻列表

\* **@param** int $classId

\* **@return** array $newsList

\*/

1. 代码注释

代码中重要的地方、复杂逻辑处理的地方都应加上注释，提高代码可读性

1. 变量注释

对定义变量的功能、范围、缺省条件等加注释

1. 补充
2. 如果系统定义了一些特定意义的常量，使用时一定要用常量名，而不是直接使用它的值。比如在模型中定义了一些常量（如：S1=1，S2=0），那么在控制器和视图中使用时一定要用S1和S2，切忌直接在代码里写1、0
3. 一般来说，一个函数的代码行数不得超过30行
4. 页面的js代码采用下面的格式编写，避免公共函数混用，一般页面js直接写在页面底部，无需另外创建单独文件

var page1 = {

'function1':function(x){

alert(‘this is page1’s function1’);

},

'function2':function(){

//todo something;

},

....

}

page1.function1();//调用