# PHP 开发规范

1. 安全规范
   1.1. 包含文件
      1.1.1. 命名规则
         1.1.1.1. 类文件
            a) 类名.php #例Database.php
         1.1.1.2. 控制文件
            a) 实意.Controller.php #例IndexController. php
         1.1.1.3. include 文件
            a) 多个 include 文件包含多个页面，请把所有的 include 文件封装在一个文件里面，具体到页面只包含一个 include 文件。
      1.1.2. 存放规则
         1.1.2.1. 类文件
            a) 类文件以及包含文件应放在不能直接用浏览器访问的目录或较深层次的目录。
   1.2. 安全规则
      1.2.1. 数据的合法性
         a) 检查用户输入数据的合法性，类型、长度、内容等。
         例：id，检查用户输入的值是否是数字型；
         content，内容长度是否超出数据库表字段长度限制以及特殊字符(';",\)
      1.2.2. php.ini
         a) register_globals = Off #全局变量，若为 On，那么自动识别参数为变量名
         b) expose_php = Off #防止 php 在 http 文件头中泄露信息
         c) error_reporting = E_ALL & ~E_NOTICE#关闭通知类错误
         d) display_errors = Off #关闭调试模式，错误信息不显示在浏览器上
         e) allow_url_fopen = Off #可以在一定程度上防止被采集
         f) magic_quotes_gpc = On#传递参数值特殊字符直接加上反斜杠(\)转义
         g) allow_url_include = Off#关闭远程直接引用(include)
         h) 不必要的扩展不要开启
2. 编码规范
   2.1. 命名规范
      2.1.1. 变量命名
         2.1.1.1. 普通变量
            a) 首字母使用小写
            b) 对于一个变量使用多个单词的，从第二个单词起每个单词的首字母大写
            例：$username, $userType
         2.1.1.2. 全局常量
            a) 所有字母都使用大写
            b) 对于一个常量使用多个单词的，使用下划线(_)隔开
            例：ROOT, ROOT_INCLUDE

2.1.1.3. SESSION 变量

a) 使用大写(S_)开头
b) 所有字母都使用大写
c) 对于一个变量使用多个单词的，使用下划线(_)隔开
例：$S_USER, $S_USER_TYPE

2.1.1.4. COOKIE 变量

a) 使用大写(C_)开头
b) 所有字母都使用大写
c) 对于一个变量使用多个单词的，使用下划线(_)隔开
d) 例：$C_USER, $C_USER_TYPE

2.1.2. 类命名

a) 以大写字母开头(首字母大写)
b) 多个单词组成的类名，单词之间不用间隔，各个单词首字母大写
例：class Db{}　　class DbMysql{}

2.1.3. 方法或函数

a) 以小写字母开头(首字母小写)
b) 多个单词组成的函数名，单词之间不用间隔，除首单词各个单词首字母大写
例：function save(){}　　function saveText(){}

2.1.4. 缩写词

a) 当变量名或者其他命名中遇到缩写词时，参照具体的命名规则，而不采用缩写词原来的全部大写的方式。
例：function getHtml(){}，而不是 function getHTML(){}

2.1.5. 数据库表名

a) 全部使用小写
b) 对于一个表名使用多个单词的，使用下划线(_)隔开
c) 如果是视图以(v_)开头

2.1.6. 数据库字段

a) 全部使用小写
b) 对于一个字段名使用多个单词的，使用下划线(_)隔开

2.1.7. KEY

a) 全部使用小写
b) 对于一个字段名使用多个单词的，使用下划线(_)隔开

2.2. 书写规范

2.2.1. 代码缩进

a) 使用 Tab(4 个空格长度)键作为缩进

```
for ( $i=0; $i<10; $i++ )
{
    echo $i;
}
```

2.2.2. 大括号{ }书写规则

a) 花括号写在控制语句的下一行
b) 控制语句有：if、for、while、switch 等

```
for ( $i=0; $i<10; $i++ )
```

```
{
    echo $i;
}
```

2.2.3. 小括号( )和函数、关键词

a) 不要把小括号和关键词紧贴在一起，要用一个空格间隔
   例：if ( $a<$b )
b) 小括号和函数名间没有空格
   例：$test = date('Y-m-d H:i:s');
c) 除非必要，不要在 return 返回语句中使用小括号
   例：return $ret;

2.2.4. =符号书写

a) 在=符号的两侧，均需留出一个空格
   例：$a = 100;

2.2.5. 控制语句书写

a) 在 if 条件判断中，如果用到常量判断条件，将常量放在等号或不等号的左边，且比较运算符两端没有空格。
   例：if ( 100==$a )
b) 在 if 条件判断中，如果有多个条件语句，逻辑运算符两端各留出一个空格。
   例：if ( 100==$a && 200==$b )
c) switch 结构中必须要有 default 块
d) 在 for 和 wiile 的循环使用中，要警惕 continue、break 的使用，避免产生类似 goto 的问题。

2.2.6. 类的构造函数

a) 不能在构造函数中有太多实际操作，顶多用来初始化一些值和变量
b) 不能在构造函数中因为使用操作而返回 false 或者错误，因为在声明和实例化一个对象的时候，是不能返回错误的。

2.2.7. 语句断行

a) 尽量保证程序语句一行就是一句，而不要让一行语句太长产生折行
b) 尽量不要使一行的代码太长，一般控制在 80 个字符以内
c) 如果一行代码太长，请使用类似 .= 的方式断行书写
   例：
```
$a = 'abcdefghijklmn'.
        'opqrstuvwxyz';
if ( 100==$a && 200==$b &&
        300==$c )
```
d) 不要在实参处直接写过长的字符串或表达式，而先用变量定义，然后在方法实参处使用变量。

2.2.8. 不要使用不可思议的数字，应该定义常量

2.3. 程序注释

a) 文档注释和方法注释几个必要的注释项要写明
   @purpose 功能描述
   @param 参数类型 参数名
   @return 返回值类型

@author 开发者
@editor 最后修改人
@created 开发时间
@updated 最后修改时间

b) 注释内容必须写在被注释对象的前面，不写在一行或者后面

2.3.1. 变量或者语句注释

a) 写在变量或者语句的前面一行，而不写在同行或者后面
b) 注释采用/* */的方式
c) 把已经注释掉的代码删除，或者注明这些已经注释掉的代码仍然保留在源码中的特殊原因

2.4. 其他规范

2.4.1. PHP 代码块标记

a) 所有的 php 程序代码块标记均使用

2.4.2. 程序文件名、目录名

a) 程序文件名和目录名命名均采用有意义的英文方式命名
b) 不使用无意义的字母，尽量少使用拼音或者不使用拼音
c) 必须使用小写字母，多个词间使用_间隔

2.4.3. PHP 项目通常的文件目录结构

a) / 项目根目录
b) /_source 源文件存放目录(线上 CSS 和 JS 都是编译压缩后的文件)
存放原始 CSS、JS、PSD 文件
c) /controllers 控制器文件目录
d) /views 视图文件目录
e) /css CSS 样式表文件目录
f) /images 图片文件目录
g) /js 脚本目录

2.4.4. PHP 项目开发中的程序逻辑结构

a) 强烈建议使用 OOP 思想编程
b) 将独立的功能模块尽量写成函数调用
c) 对应一整块业务逻辑，我们建议封装成类，既可以提高代码可读性，也可以提高代码重用性

3. 特殊规范

3.1. 变量定义

a) 特定环境下必须先声明后使用

3.2. 引用的使用

a) 在形参处变量前加(&)，而不能在实参变量名前加(&)
b) 即使 php.ini 的配置 allow_call_time_pass_reference=On，也不推荐使用。

3.3. 变量的输入输出

a) 对 web 通过 GET 或者 POST 方法传递来的参数均要求进行严格的过滤和合法性验证，不推荐使用直接的$_GET、$_POST 或者$_REQUEST 获取，而通过模块获取和过滤处理。