2018

【应用软件开发技术规范】

[江西省职业技能鉴定服务管理平台-应用软件开发技术规范]

[北京博奥网络教育科技股份有限公司版权所有]

目录

[一： 导言 4](#_Toc525888178)

[1.1 目的 4](#_Toc525888179)

[1.2 开发背景 4](#_Toc525888180)

[二： 开发技术规范 5](#_Toc525888181)

[2.1 整体要求 5](#_Toc525888182)

[2.2 安全规范 5](#_Toc525888183)

[2.2.1 包含文件 5](#_Toc525888184)

[2.2.1.1 命名规则 5](#_Toc525888185)

[2.2.1.2 存放规则 6](#_Toc525888186)

[2.2.1.3 安全规则 6](#_Toc525888187)

[2.2.1.4 PHP规则 7](#_Toc525888188)

[2.2.1.5 其它规则 8](#_Toc525888189)

[2.2.1.5.1 参数转义 8](#_Toc525888190)

[2.2.1.5.2 HTML文本 8](#_Toc525888191)

[2.3 编码规范 8](#_Toc525888192)

[2.3.1 命名规范 8](#_Toc525888193)

[2.3.1.1 变量命名 8](#_Toc525888194)

[2.3.1.1.1 普通变量 9](#_Toc525888195)

[2.3.1.1.2 静态变量 9](#_Toc525888196)

[2.3.1.1.3 类变量 9](#_Toc525888197)

[2.3.1.1.4 全局变量 9](#_Toc525888198)

[2.3.1.1.5 全局常量 10](#_Toc525888199)

[2.3.1.1.6 session变量 10](#_Toc525888200)

[2.3.1.2 类 10](#_Toc525888201)

[2.3.1.3 方法或函数 10](#_Toc525888202)

[2.3.1.4 缩写词 10](#_Toc525888203)

[2.3.1.5 数据库表名 11](#_Toc525888204)

[2.3.1.6 数据库列名 11](#_Toc525888205)

[2.3.1.7 数据库其它规则 11](#_Toc525888206)

[2.3.2 书写规则 12](#_Toc525888207)

[2.3.2.1 代码缩进 12](#_Toc525888208)

[2.3.2.2 大括号{ } 12](#_Toc525888209)

[2.3.2.3 小括号（ ）和函数、关键词等 13](#_Toc525888210)

[2.3.2.4 ＝符号书写 13](#_Toc525888211)

[2.3.2.5 if else swith for while等书写 14](#_Toc525888212)

[2.3.2.6 类的构造函数 14](#_Toc525888213)

[2.3.2.7 语句断行 15](#_Toc525888214)

[2.3.2.8 数字逻辑 15](#_Toc525888215)

[2.3.2.9 true/false 17](#_Toc525888216)

[2.3.2.10 避免嵌入式赋值 17](#_Toc525888217)

[2.3.2.11 错误返回检测规则 18](#_Toc525888218)

[2.3.3 程序注释 18](#_Toc525888219)

[2.3.3.1 程序头注释块 18](#_Toc525888220)

[2.3.3.2 类的注释 19](#_Toc525888221)

[2.3.3.3 函数和方法的注释 19](#_Toc525888222)

[2.3.3.4 变量或者语句注释 20](#_Toc525888223)

[2.3.3.5 其他规范（建议） 21](#_Toc525888224)

[2.3.3.5.1 代码标记 21](#_Toc525888225)

[2.3.3.5.2 程序文件名、目录名 21](#_Toc525888226)

[2.3.3.5.3 文件目录结构 21](#_Toc525888227)

[2.3.3.5.4 代码的分离 22](#_Toc525888228)

[2.3.3.5.5 程序逻辑结构 22](#_Toc525888229)

[2.4 编码特殊规范 23](#_Toc525888230)

[2.4.1 变量定义 23](#_Toc525888231)

[2.4.2 引用的使用 23](#_Toc525888232)

[2.4.3 变量的输入输出 24](#_Toc525888233)

# 导言

## 目的

为了更好的提高技术部的工作效率，保证开发的有效性和合理性，并可最大程度的提高程序代码的可读性和可重复利用性，制定此规范。开发团队根据自己的实际情况，可以对本规范进行补充或裁减。

## 开发背景

开发语言：Php-v5.6.39

开发环境：Linux/Windows

开发框架：ThinkPHP 5.0.20

数据库：msyql v5.6-5.7

服务器：Apache-2.4.33

其他：

* apr-1.6.3
* apr-util-1.6.1
* libiconv-1.15
* libzip-1.2.0
* PDO\_MYSQL-1.0.2
* zip-1.15.3
* mcrypt-2.6.8
* mhash-0.9.9.9
* libmcrypt-2.5.8

# 开发技术规范

## 整体要求

技术部php开发规范将参照遵循PSR-2、PSR-4规范，基本采用指定的规范，在其基础上增加、修改或删除部分适合具体开发环境的规范。本规范只针对PHP开发过程中编码的规范，对于PHP开发项目中文件、目录、数据库等方面的规范，将不重点涉及。

本规范包含了PHP开发时程序编码中命名规范、代码缩进规则、控制结构、函数调用、函数定义、注释、包含代码、PHP标记、文件头的注释块、CVS标记、URL样例、常量命名等方面的规则。

## 安全规范

### 包含文件

#### 命名规则

提取出来具有通用函数的包含文件，文件后缀以.inc来命名，表明这是一个包含文件。

如果有多个.inc文件需要包含多页面，请把所有.inc文件封装在一个文件里面，具体到页面只需要包换一个.inc文件就可以了

如：xxx\_session.inc、xxx\_comm.inc、xxx\_setting.inc、mysql\_db.inc

把以上文件以以下方式，封装在xxx.basic.inc文件里面

require\_once（”xxx\_session.inc”）;

require\_once（”xxx\_comm.inc”）;

require\_once（”xxx\_setting.inc”）;

require\_once（”mysql\_db.inc”）;

注：是否需要封装到一个文件，视情况而定，如果每个inc的功能是分散到不同的页面使用的话，就不建议封装。

#### 存放规则

一般包含文件不需要直接暴露给用户，所以应该放在 Web Server访问不到的目录，避免因为配置问题而泄露设置信息。

#### 安全规则

**输入和输出**

检查是否做了HTML代码的过滤；可能出现的问题：如果有人输入恶意的HTML代码，会导致窃取cookie， 产生恶意登录表单，和破坏网站。

**数据库escape**

检查变量做数据库操作之前是否做了escape；可能出现的问题：如果一个要写入查询语句的字符串变量包含了某些特殊的字符，比如引号（’ ，”）或者分号（;） 可能造成执行了预期之外的操作。

建议采用的方法：使用mysql\_escape\_string（） 或实现类似功能的函数。

**数值合法性**

可能出现的问题：异常的数值会造成问题。如果对输入的数值不做检查会造成不合法的或者错误的数据存入UDB、存入其它的数据库或者导致意料之外的程序操作发生。

如果程序以用户输入的参数值做为文件名，进行文件操作，恶意输入系统文件名会造成系统损毁。

**cookie使用**

可能出现的问题：不正确的cookie使用可能造成用户数据泄漏。

**访问控制**

对内部使用的产品或者供合作方使用的产品，要考虑增加访问控制。

**logs**

确保用户的保密信息没有记在log中（例如：用户的密码）

确保对关键的用户操作保存了完整的用户访问记录

**https**

对敏感数据的传输要采用https

#### PHP规则

设置 register\_globals = off （Y！PHP 已经禁止了register\_globals，如果你使用Y！PHP可以不考虑这项设置）。

设置 error\_reporting = E\_ALL （Y！PHP 的缺省设置），并且要修正所有的error和warning。

将实际的操作放在被引用的文件中。把引用文件放到不可以被直接浏览的目录下。

#### 其它规则

##### 参数转义

页面接到参数需要SQL操作，这时候需要做转义，尤其需要注意”;”。

如：$a = ” Let’s go ” ;

$sql = “Insert into tmp（col） values（’$a’）” ;

这种情况出现错误的不确定性。

##### HTML文本

很多时候需要存放一大段HTML文本供页面使用，像用户定制页头页脚等。需要剔除脚本标记，避免执行恶意php代码。转换”《“”》“号，保证代码完整。

## 编码规范

### 命名规范

制定统一的命名规范对于项目开发来说非常重要，不但可以养成程序员一个良好的开发习惯，还能增加程序的可读性、可移植性和可重用性，还能很好的提高项目开发的效率。

#### 变量命名

变量命名分为普通变量、静态变量、局部变量、全局变量、Session变量等方面的命名规则。

变量用一个美元符号$后面跟变量名表示。变量名区分大小写。一个有效变量名由字母或者下划线开头，后面跟任意数量的字母、数字、下划线。不应该在变量中使用中文等ASCII字符。变量命名规则分驼峰法（又名小驼峰法: 除第一个单词之外，其他单词首字母大写）和Pascal法（又名大驼峰法：所有单词首字母大写）

##### 普通变量

普通变量命名遵循驼峰法规则，常于方法内部使用，变量开头为表示类型的前缀。

例子：字符串：$strName、数组：$arrName、对象：$objName、布尔值：$isTrue

##### 静态变量

静态变量命名和普通变量一样遵循驼峰法命名规则，并且使用小写的s\_开头；

例子：$s\_baseDir、$s\_redRosePrise等。

##### 实例变量

实例变量命名遵循Pascal法规则：所有单词首字母大写，并且用首字母M和S来区模型层和服务层。

例子：$SUser:用户服务类UserService的实例化对象、$MUser：用户模型类UserModel的实例化对象

##### 全局变量

全局变量命名单词全用大写，多个单词使用下划线，并且以$G\_开头

例子：$G\_LOG\_LEVEL、$G\_LOG\_PATH;

##### 全局常量

全局常量命名单词全用大写，多个单词使用下划线，并且以$C\_开头

例子：$C\_LOG\_LEVEL、$C\_LOG\_PATH;

##### session变量

session变量命名单词全用大写，多个单词使用下划线，并且以$S\_开头

例子：$S\_LOG\_LEVEL、$S\_LOG\_PATH;

#### 类

php中类命名遵循Pascal法规则：以大写字母开头，多个单词组成的变量名，单词之间不用间隔，各个单词首字母大写。

例子：class MyClass 或class DbOracle等。

#### 方法或函数

方法或函数命名遵循驼峰法规则：首字母小写开头，多个单词组成的变量名，单词之间不用间隔。

例子：function userLogin （）或function userLoginCheck（）等。

#### 缩写词

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。

例子：function myPear（不是myPEAR） functio getHtmlSource（不是getHTMLSource）。

#### 数据库表名

数据库表名命名遵循以下规范：

1. 表名均使用小写字母；
2. 对于多个单词组成的表名，使用\_间隔；
3. 关系表多个单子组成的，主表名要全拼，次名和其他名可简写；
4. 表名单词数一般不超过4个；
5. 统一模块相关表名可加上统一前缀标识；

#### 数据库列名

数据库列名命名遵循以下规范：

1. 全部使用小写；
2. 多个单词间使用\_间隔；

#### 数据库其它规则

1. 如无备注，则表中的第一个id字段一定是主键且为自动增长；
2. 如无备注，则数值类型的字段请使用UNSIGNED属性；
3. 如无备注，排序字段sort在程序中默认使用降序排列；
4. 如无备注，所有字段都设置NOT NULL，并设置默认值；
5. 如无备注，所有的布尔值字段，都必须设置一个默认值，并设为0；
6. 所有的数字类型字段，都必须设置一个默认值，并设为0；
7. 针对varchar类型字段的程序处理，请验证用户输入，不要超出其预设的长度；
8. 建表时将数据字典中的字段中文名和属性备注写入数据表的备注中(“PK、自动增长”不用写)；
9. 如无备注，建表时一律采用innodb引擎；日志及报表类表建议用myisam，与交易，审核，金额相关的表建议用innodb引擎。总体来讲数据库默认innodb；
10. 默认使用utf8字符集（由于数据库定义使用了默认，数据表可以不再定义，但为保险起见，建议都写上）；
11. 所有不常见的表都必须有备注，写明白这个表中存放的数据内容；

### 书写规则

书写规则是指在编写php程序时，代码书写的规则，包括缩进、结构控制等方面规范。

#### 代码缩进

在书写代码的时候，必须注意代码的缩进规则，我们规定代码缩进规则如下：

使用4个空格作为缩进，而不使用tab缩进（对于ultraedit，可以进行预先设置）。

#### 大括号{ }

在程序中进行结构控制代码编写，如if、for、while、switch等结构，大括号传统的有两种书写习惯，分别如下：

a．{直接跟在控制语句之后，不换行，如

for （$i=0;$i《$count;$i++） {

echo “test”;

}

b．{在控制语句下一行，如

for（$i=0;$i《$count;$i++）

{

echo “test”;

}

其中，a是编码规范建议的方式，但是从实际书写中来讲，这并不影响程序的规范和影响用phpdoc实现文档，所以可以根据个人习惯来采用上面的两种方式，但是要求在同一个程序中，只使用其中一种，以免造成阅读的不方便。

#### 小括号（ ）和函数、关键词等

小括号、关键词和函数遵循以下规则：

1. 不要把小括号和关键词紧贴在一起，要用一个空格间隔；如if （ $a《$b ）；
2. 小括号和函数名间没有空格；如$test = date（“ymdhis”）；
3. 除非必要，不要在Return返回语句中使用小括号。 如Return $a；

#### ＝符号书写

在程序中=符号的书写遵循以下规则：

1. 在=符号的两侧，均需留出一个空格；如$a = $b 、if（$a = = $b）等；
2. 在一个申明块，或者实现同样功能的一个块中，要求=号尽量上下对其，左边可以为了保持对齐使用多个空格，而右边要求空一个空格；

如下例：

$testa = $aaa;

$testaa = $bbb;

$testaaa = $ccc;

#### if else swith for while等书写

对于控制结构的书写遵循以下规则：

1. 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，例如：if （ 6 == $errorNum ），因为如果你在等式中漏了一个等号，语法检查器会为你报错，可以很快找到错误位置，这样的写法要多注意；
2. switch结构中必须要有default块；
3. 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；

#### 类的构造函数

如果要在类里面编写构造函数，必须遵循以下规则：

1. 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；
2. 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；

#### 语句断行

在代码书写中，遵循以下原则：

1. 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；
2. 尽量不要使一行的代码太长，一般控制在80个字符以内；
3. 如果一行代码太长，请使用类似 。= 的方式断行书写；
4. 对于执行数据库的sql语句操作，尽量不要在函数内写sql语句，而先用变量定义sql语句，然后在执行操作的函数中调用定义的变量；

例子：

$sql = “SELECT username，password，address，age，postcode F[ROM](http://www.elecfans.com/tags/ROM/) test\_t ”;

$res = mysql\_query（$sql）;

#### 数字逻辑

一个在源代码中使用了的赤裸裸的数字是不可思议的数字，因为包括作者，在三个月内，没人它的含义。例如：

if （22 == $foo）

{

start\_thermo\_nuclear\_war（）;

}

else if （19 == $foo）

{

refund\_lotso\_money（）;

}

else

{

cry\_cause\_im\_lost（）;

}

你应该用define（）来给你想表示某样东西的数值一个真正的名字，而不是采用赤裸裸的数字，例如：

define（“PRESIDENT\_WENT\_CRAZY”， “22”）;

define（“WE\_GOOFED”， “19”）;

if （ PRESIDENT\_WENT\_CRAZY == $foo）

{

start\_thermo\_nuclear\_war（）;

}

else if （WE\_GOOFED == $foo）

{

refund\_lotso\_money（）;

}

else

{

happy\_days\_i\_know\_why\_im\_here（）;

}

#### true/false

遵循以下规则：

1. 不能使用0/1代替true/false，在PHP中，这是不相等的；
2. 不要使用非零的表达式、变量或者方法直接进行true/false判断，而必须使用严格的完整true/false判断；

如：不使用if （$a） 或者if （checka（）） 而使用if （FALSE ！= $a）或者 if （FALSE ！= check（））

#### 避免嵌入式赋值

在程序中避免下面例子中的嵌入式赋值：

不使用这样的方式：

while （$a ！= （$c = getchar（）））

{

process the character

}

#### 错误返回检测规则

检查所有的系统调用的错误信息，除非你要忽略错误。

为每条系统错误消息定义好系统错误文本，并记录错误LOG。

### 程序注释

每个程序均必须提供必要的注释，书写注释要求规范，参照PSR提供的注释要求，为今后利用phpdoc生成php文档做准备。程序注释的原则如下：

1. 注释中除了文件头的注释块外，其他地方都不使用//注释，而使用/\* \*/的注释；
2. 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；

#### 程序头注释块

每个程序头部必须有统一的注释块，规则如下：

1. 必须包含本程序的描述；
2. 必须包含作者；
3. 必须包含书写日期；
4. 必须包含版本信息；
5. 必须包含项目名称；
6. 必须包含文件的名称；
7. 重要的使用说明，如类的调用方法、注意事项等；

#### 类的注释

类的注释采用里面的参考例子方式：

/\*\*

\* @ Purpose：

\* 访问数据库的类，以ODBC作为通用访问接口

\* @Package Name： Database

\* @Author： Forrest Gump gump@crtvu.edu.cn

\* @Modifications：

\* No20020523-100：

\* odbc\_fetch\_into（）参数位置第二和第三个位置调换

\* John Johnson John@crtvu.edu.cn

\* @See： （参照）

\*/

class Database

{

……

}

#### 函数和方法的注释

函数和方法的注释写在函数和方法的前面，采用类似下面例子的规则：

/\*\*

\* @Purpose：

\* 执行一次查询

\* @Method Name： Query（）

\*

\* @Pa[ram](http://www.elecfans.com/tags/ram/)： string $queryStr SQL查询字符串

\* @Param： string $username 用户名

\*

\* @Author： Michael Lee

\*

\* @Return： mixed 查询返回值（结果集对象）

\*/

function（$queryStr，$username）

{……}

#### 变量或者语句注释

程序中变量或者语句的注释遵循以下原则：

1. 写在变量或者语句的前面一行，而不写在同行或者后面；
2. 注释采用/\* \*/的方式；
3. 每个函数前面要包含一个注释块。内容包括函数功能简述，输入/输出参数，预期的返回值，出错代码定义。
4. 注释完整规范。
5. 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。

#### 其他规范（建议）

##### 代码标记

所有的php程序代码块标记均使用。

##### 程序文件名、目录名

程序文件名和目录名命名均采用有意义的英文方式命名，不使用拼音或无意义的字母，同时均必须使用小写字母，多个词间使用\_间隔。

##### 文件目录结构

建议在开发规范的独立的PHP项目时，使用规范的文件目录结构，这有助于提高项目的逻辑结构合理性，对应扩展和合作，以及团队开发均有好处。

一个完整独立的PHP项目通常的文件和目录结构如下：

/ 项目根目录

/manage 后台管理文件存放目录

/css css文件存放目录

/doc 存放项目文档

/images 所有图片文件存放路径（在里面根据目录结构设立子目录）

/scripts 客户端js脚本存放目录

/tpl 网站所有html的模版文件存放目录

/error.php 错误处理文件（可以定义到apache的错误处理中）

以上目录结构是通常的目录结构，根据具体应用的具体情况，可以考虑不用完全遵循，但是尽量做到规范化。

##### 代码的分离

对性能要求不是很高的项目和应用，我们建议不采用PHP和HTML代码直接混排的方式书写代码，而采用PHP和HTML代码分离的方式，即采用模版的方式处理，这样一方面对程序逻辑结构更加清晰有利，也有助于开发过程中人员的分工安排，同时还对日后项目的页面升级该版提供更多便利。

对于一些特殊情况，比如对性能要求很高的应用，可以不采用模版方式。

##### 程序逻辑结构

对于PHP项目开发，尽量采用OOP的思想开发，尤其在PHP5以后，对于面向对象的开发功能大大提高。

在PHP项目中，我们建议将独立的功能模块尽量写成函数调用，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。比如，我们通常将对数据库的接口封装成数据库类，有利于平台的移植。

重复的代码要做成公共的库。（除了我们在plug-in产品上遇到的情况，该产品系列有多个相类似的产品，为了尽可能地减少安装包尺寸，不适合将这些产品共用的所有函数做成公共的库）。

## 编码特殊规范

### 变量定义

php代码编写要求所有的变量均需要先申明后使用，否则会有错误信息，对于数组，在使用一个不确定的key时，比如先进行isset（）的判断，然后再使用；比如下面的代码：

$array ＝ array（）;

$var ＝ isset（$array［3］） ？ $array［3］ ： “”；

### 引用的使用

引用在程序中使用比较多，为了公用同一个内存，而不需要另外进行复制，XXX环境下的引用使用时，需要注意下面的情况；

在对函数的输入参数中使用引用时，不能在调用的时候在输入参数前加&来引用，而直接使用该变量即可，同时必须在函数定义的时候说明输入参数来自引用，比如下面的代码：

$a ＝ 1；

function ab（&$var）

{

$var ++；

return $var；

}

$b ＝ ab（$a） // 注意，此处不能使用 $b ＝ ab（&$a）的方式；

echo $b.”/n”；

echo $a.”/n”；

此时 $a和$b都是2；

开发环境下对引用的特殊要求源自php.ini文件里面的allow\_call\_time\_pass\_reference 项设置， 对外公开的版本是 On ，这样就可以支持&直接加到调用函数时变量前面进行引用，但是这一方法遭到抗议，并可能在将来版本的PHP/Zend里不再支持。受到鼓励的指定哪些参数按引用传递的方法是在函数声明里。你被鼓励尝试关闭这一选项（使用 off，XXX的所有运行环境下都是off）并确认你的脚本仍能正常工作，以保证在将来版本的语言里它们仍能工作。

### 变量的输入输出

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，而通过相应的模块提供的方法获取和过滤处理。