PHP开发编码规范

1 编写目的

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，指定此规范。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。

2 整体要求

技术部php开发规范将参照PEAR的规范，基本采用PEAR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。

3 安全规范

3.1 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

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。

检查输入数值的合法性

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数据库或者导致意料之外的程序操作发生。

举例：

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。

核实对cookie的使用以及对用户数据的处理

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

3.2 其它处理规则

3.2.1 对输入参数值进行转义处理

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。

3.2.2 操作大HTML文本

很多时候需要存放一大段HTML文本供页面使用，象用户定制页头页脚等。

需要剔除脚本标记，避免执行恶意代码攻击。

转换”<" ">“号，保证代码完整。

4 编码规范

4.1 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。

4.1.1 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。

4.1.1.1 普通变量

普通变量命名遵循以下规则：

a． 采用小驼峰命名法，除第一个单词外，其他单词首字母大写;

例如：$baseDir、$redRosePrice等。

4.1.1.2 静态变量

静态变量命名遵循以下规则：

a． 静态变量使用小写的s\_开头;

b． s\_开头之后采用小驼峰命名法，除第一个单词外，其他单词首字母大写;

例子：$s\_baseDir、$s\_redRosePrise等。

4.1.1.3 局部变量

局部变量命名遵循以下规则：

a． 所有字母使用小写;

b． 变量使用'\_'开头;

c． \_开头之后采用小驼峰命名法，除第一个单词外，其他单词首字母大写。

例子：$\_baseDir、$\_redRosePrice等。

4.1.1.4 全局变量

a．全局变量使用小写的g\_开头;

b． g\_开头之后采用小驼峰命名法，除第一个单词外，其他单词首字母大写;

例如

$g\_baseDir、$g\_redRosePrise等。

4.1.1.5 全局常量

全局常量命名遵循以下规则：

a． 所有字母使用大写

b．全局常量多个单词间使用'\_'作为间隔。

例子：$BASE\_DIR、$RED\_ROSE\_PRICE等。

4.1.1.6 session变量

session变量命名遵循以下规则：

a． 所有字母使用大写;

b． session变量名使用'S\_'开头;

c． 多个单词间使用'\_'间隔。

例子：$S\_BASE\_DIR、$S\_RED\_ROSE\_PRICE等。

4.1.2 类

php中类命名遵循以下规则：

a． 以大写字母开头;

b． 大驼峰命名，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。

4.1.3 方法或函数

方法或函数命名遵循以下规则：

a． 首字母小写;

b． 多个单词间不使用间隔，除第一个单词外，其他单词首字母大写。

例子：function myFunction ()或function myDbOracle ()等。

4.1.4 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） function getHtmlSource（不是getHTMLSource）。

4.1.5 数据库表名

数据库表名命名遵循以下规范：

a． 表名均使用小写字母;

b．对于视图，使用pv\_开头;

c． 对于方法，使用func\_开头;

d． 对于多个单词组成的表名，使用\_间隔;

例子：fund\_information和pv\_dc\_product等

4.1.6 数据库字段

数据库字段命名遵循以下规范：

a． 全部使用小写;

b． 多个单词间使用\_间隔。

例子：user\_name、rose\_price等。

4.2 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：

4.2.1 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：

a． 使用4个空格作为缩进，而不使用tab缩进（部分编辑器可以进行预先设置成等效转换）

例子：

for ( $i=0;$i<$count;$i++ ) {

echo "test";

}

4.2.2 大括号{ }书写规则

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PEAR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。

4.2.3 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：

a． 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 );

b． 小括号和函数名间没有空格;如$test = date("ymdhis");

c． 除非必要，不要在return返回语句中使用小括号。 如return($a);

4.2.4 ＝符号书写

在程序中=符号的书写遵循以下规则：

a． 在=符号的两侧，均需留出一个空格;如$a = $b 、if($a == $b)等;

b． 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;

4.2.5 if else swith for while等书写

对于控制结构的书写遵循以下规则：

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，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;

b． switch结构中必须要有default块;

c． 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;

4.2.6 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：

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;

4.2.7 语句断行, 每行控制在80个字符以内

在代码书写中，遵循以下原则：

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;

d． 对于执行数据库的sql语句操作，尽量不要在函数内写sql字符语句，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 fund\_id,fund\_name,fund\_type FROM fund\_information";

$sql .= " WHERE fund\_id=’HF00000008’";

$res = mysql\_query($sql);

4.2.8 不要不可思议的数字

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 (1 == $fundType) {

……;

}else if (2 == $ fundType) {

……;

} else{

……;

}

可以用define()来给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：

//信托类型

define("FUND\_TYPE\_TRUST", "1");

//有限合伙类型

define("FUND\_TYPE\_LIMITED\_PARTNERSHIP", "2");

//清算状态

define("FUND\_STATUS\_CLEAR ", "4");

if (FUND\_TYPE\_TRUST == $fundType) {

……;

} else if (FUND\_TYPE\_LIMITED\_PARTNERSHIP == $ fundType) {

……;

}

4.2.9 TRUE/FALSE和0/1判断

遵循以下规则：

a． 不能使用0/1代替TRUE/FALSE，在PHP中，这是不相等的;

b． 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())

4.2.10 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

}

4.2.11 错误返回检测规则

检查所有的系统调用的错误信息，除非是要忽略错误。

为每条系统错误消息定义好系统错误文本，并记录错误LOG。

4.3 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PEAR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：

a． 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;

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;

4.3.1 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: ltotal@simuwang.com

\* @Modifications:

\* No20160120-100:

\* odbc\_fetch\_into()参数位置第二和第三个位置调换

\* ltotal ltotal@simuwang.com

\* @See: (参照)

\*/

class Database

{

……

}

4.3.2 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：

/\*\*

\* @Method Name: Query()

\* @Param: string $queryStr SQL查询字符串

\* @Param: string $username 用户名

\* @Author: ltotal

\* @Return: mixed 查询返回值（结果集对象）

\*/

function（$queryStr,$username）{……}

4.3.3 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：

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． 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。

d． 注释完整规范。

e． 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。

f．

例子：

/\*\*

\* @Purpose:

\* 数据库连接用户名

\* @Attribute/Variable Name: db\_user\_name

\* @Type: string

\*/

var db\_user\_name;

4.4 其他规范（建议）

4.4.1 php代码标记

所有的php程序代码块标记均使用

4.4.2 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。

4.4.3 PHP项目通常的文件目录结构

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。

4.4.4 PHP和HTML代码的分离问题

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。

4.4.5 PHP项目开发中的程序逻辑结构

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。

5 特定环境下PHP编码特殊规范

5.1 变量定义

XXX环境下的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] : ““;

5.2 变量的输入输出

在ThinkPHP开发环境下，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ThinkPHP的专门获取参数的方法I()，再配合配置文件中定义预过滤函数的”DEFAULT\_FILTER”配置项进行获取和过滤处理。

224服务器主业务项目目录结构

