PHP开发编码规范

祁劲松

目录

一、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。

二、整体要求

- 技术部php开发规范将参照PEAR（PHP Extension and Application Repository，PHP扩展与应用库）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。
- 来自PHP中文网 https://www.php.cn/course/982.html

三、安全规范

3.1 PHP包含文件 - PHP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

- 把以上文件以一下方式，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。

3.1 PHP包含文件 - PHP程序存放规则

- 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。

3.2 PHP安全规则

- 输入和输出：检查是否做了HTML代码的过滤，检查输入数值的合法性
- 核实对cookie的使用以及对用户数据的处理
- 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

3.3 一些针对PHP的规则

- 设置 register_globals = off (PHP 已经禁止了register_globals，如果你使用PHP可以不考虑这项设置)
- 设置 error_reporting = E_ALL (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

3.3 一些针对PHP的规则 - PHP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。

3.3 一些针对PHP的规则 - PHP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php代码。
- 转换”<"">“号，保证代码完整。

4.1 PHP命名规范

PHP命名规范 - PHP 变量命名

- 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。

1）普通变量

- 普通变量命名遵循以下规则：
- a． 所有字母都使用小写；
- b． 对于一个变量使用多个单词的，使用'_'作为每个词的间隔。
- 例如：$base_dir、$red_rose_price等

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. 静态变量所有字母都使用小写；
- c. 多个单词组成的变量名使用'_'作为每个词的间隔。
- 例子：$s_base_dir、$s_red_rose_prise等。

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. 多个单词组成的局部变量名使用'_'作为每个词间的间隔。
- 例子：$_base_dir、$_red_rose_price等。

4) 全局变量

- 全局变量应该带前缀'g',知道一个变量的作用域是非常重要的。
- 例如
- global $gLOG_LEVEL;
- global $gLOG_PATH;

5) 全局常量

- 全局变量命名遵循以下规则：
- a. 所有字母使用大写
- b. 全局变量多个单词间使用'_'作为间隔。
- 例子：$BASE_DIR、$RED_ROSE_PRICE等。

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。

PHP命名规范 - PHP 类命名

- php中类命名遵循以下规则：
- a． 以大写字母开头；
- b． 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。

PHP命名规范 - PHP 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。

PHP命名规范 - PHP 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）
- 又例如：function getHtmlSource（不是getHTMLSource）。

PHP命名规范 - PHP 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. 对于普通数据表，使用_t结尾；
- c. 对于视图，使用_v结尾；
- d. 对于多个单词组成的表名，使用_间隔；
- 例子：user_info_t和book_store_v等
- 但在Drupal或者MediaWiki的项目中并没有遵循以上规范，可以用Drupal或者MediaWiki的规范

PHP命名规范 - PHP 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 PHP书写规则

PHP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：

PHP书写规则 - PHP 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（对于ultraedit，可以进行预先设置）
- 例子：

```
for ( $i = 0; $i < $count; $i++ ) {
    echo "test";
}
```

PHP书写规则 - PHP 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。

PHP书写规则 - PHP 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;

PHP书写规则 - PHP ＝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;

PHP书写规则 - PHP 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while的循环使用中，要警惕continue、break的使用，避免产生类似goto的问题；

PHP书写规则 - PHP 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；

PHP书写规则 - PHP 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 FROM test_t ";
$sql .= " WHERE username='aaa'";
$res = mysql_query($sql);
```

PHP书写规则 - PHP 不要含义不清的数字

- 一个在源代码中使用了的直接的数字是不可被理解的数字，因为包括作者，在三个月内，没人记得它的含义。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，而不是采用直接数字，例如：

```
define("PRESIDENT_WENT_CRAZY", "22");
define("WE_GOOFED", "19");
define("THEY_DIDNT_PAY", "16");

if ( PRESIDENT_WENT_CRAZY == $foo) {
    start_thermo_nuclear_war();
}
else if (WE_GOOFED == $foo) {
    refund_lotso_money();
}
else if (THEY_DIDNT_PAY == $foo)
```

PHP书写规则 - PHP 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 (check()) 而使用if (FALSE != $a)或者 if (FALSE != check())

PHP书写规则 - PHP 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
}
```

PHP书写规则 - PHP 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。

4.3 PHP程序注释

PHP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：
- a. 必须包含本程序的描述；
- b. 必须包含作者；
- c. 必须包含书写日期；
- d. 必须包含版本信息；
- e. 必须包含项目名称；
- f. 必须包含文件的名称；
- g. 重要的使用说明，如类的调用方法、注意事项等；

- 参考例子如下：

```
<?php
//
// +----------------------------------------------------------+
// | PHP version 4.0                                          |
// +----------------------------------------------------------+
// | Copyright (c) 1997-2001 The PHP Group                    |
// +----------------------------------------------------------+
// | This source file is subject to  of the PHP license,      |
// | that is bundled with this packafile LICENSE, and is      |
// | available at through the world-web at                    |
// | http://www.php.net/license/2_02.txt.                     |
// | If you did not receive a copy of the  and are unable to  |
// | obtain it through the world-wide-web,end a note to       |
// | license@php.net so we can mail you a immediately.        |
// +----------------------------------------------------------+
// | Authors: Stig Bakken <ssb@fast.no>                       |
// |          Tomas V.V.Cox <cox@idecnet.com>                 |
// |                                                          |
// +----------------------------------------------------------+
//
// $Id: Common.php,v 1.8.2.3 2001/11/13 01:26:48 ssb Exp $
```

PHP类的注释

- PHP类之前必须有注释块，规则如下：
- a. 必须包含本类的描述；
- b. 必须包含作者；
- c. 必须包含书写日期；
- d. 必须包含版本信息；
- e. 必须包含类名称；
- g. 重要的使用说明，如类的调用方法、注意事项等；
- 参考例子：

```
/**
* @ 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 {
……
}
```

PHP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，每个函数前面要包含一个注释块。采用类似下面例子的规则，包含：
- 函数或者方法的目的
- 函数或者方法的名称
- 函数或者方法的作者
- 函数或者方法的输入参数（如果是多个参数每个都要描写）
- 函数预期的返回值、出错代码定义等

```
/**
* @Purpose:
* 执行一次查询
* @Method Name: Query()
*
* @Param: 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)
{……}
```

PHP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. 注释完整规范。
- d. 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。
- 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 PHP其他规范（建议）

PHP代码标记

- 所有的php程序代码块标记均使用
- <?php
- ?>

PHP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。

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。
- 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。

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。

PHP项目开发中的程序逻辑结构

- 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）

五、特定环境

PHP变量定义

- 特殊环境下的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，需要先进行isset()的判断，然后再使用；比如下面的代码：

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

PHP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，特殊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；

- 特殊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。

PHP变量的输入输出

- 在特殊环境下，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，而通过特定CMS或者框架的模块提供的方法获取和过滤处理。

LARAVEL规范

文档目的

- 来源：https://blog.csdn.net/qq_22823581/article/details/87689797
- https://learnku.com/docs/laravel-specification/5.5/whats-the-use-of-standards/510
- 本规范希望通过制定一系列规范化PHP代码的规则，统一风格，便于代码阅读，提高效率。
- 文档适用范围
- 适用于所有PHP开发人员

格式

- PHP代码文件必须以不带BOM的UTF-8编码； 纯PHP文件必须省略最后 ?>结束标签。
- 使用tab键来缩进，每个tab键长度必须设置4个空格。
- 每行最多120个字符；非空行不能有多余的空格；每行不能有多条语句；
- 适当空行有利于代码阅读，但不能滥用。

变量

- PHP所有关键字必须小写，bool值true，false，null 也必须小写；
- 关键字有：abstract、array、as、break、case、catch、class、const、try、use、public等等。
- 函数：
- 每个函数或方法不能多于60行。
- 数组：
- 当有两个以上键值对时，请换行，如：
- $where = [
- 'id' => 789,
- 'user_name' => '52php'
-];

命名

- 1、 类名 类名使用大驼峰式的写法，如下：

```
class ClassName
{
    // constants, properties, methods
}
```

类的方法名

2、 类的方法名 类方法名使用小驼峰的写法，方法名紧跟括号，中间不能有空格，参数之间要有空格且大括号必须自成一行如下：

```
class ClassName
{
    public function fooBarBaz($storeName, $storeId, array $info = [])
    {
        // method body
    }
}
```

类文件名

- 类文件名 类文件名必须和对应的类名一致，也是大驼峰式写法
- 其他文件名 如视图、配置、脚本文件等全部用小写字母+下划线命名

函数名

- 普通函数使用小写字母+下划线命名，且函数名紧跟括号，中间不能有空格，参数之间要有空格，大括号换行如：

```
function curl_request($a, $b)
{
    // method body
}
```

变量、常量名

- 变量名使用小驼峰的写法，常量名全部用大写+下划线，如下
- //变量
- $userName = 'xia';
- //常量
- const VERSION = '1.0';
- const SITE_URL = 'http://www.baidu.com';

数据库命名

- 数据库全部用小写命名，数据表及字段采用小写+下划线方式命名，且表中主键id尽可能加上表名，如：title_id。

注释

- 类、方法、行数，要以/* */格式注释，第二行开始写注释内容，表明作者、时间、参数格式、返回格式等，常用标签有：@autor、@since、@param 、@return等，如：

- 其他注释 尽量用 //,注释开始前必须一个空格

控制结构

- 控制结构关键词（if、elseif、else、foreach、switch、case、try、catch、for等）后必须有一个空格；
- 左括号 (后一定不能有空格，右括号) 前也一定不能有空格；
- 右括号) 与开始花括号 { 间一定有一个空格；
- 结束花括号 } 一定在结构体主体后单独成行。
- 在 if 条件判断中，如果用到常量，将常量放在等号或不等号的左边，如下：

```
if (6 == $a) {
  //body
}
```

-

```
if ($expr1) {
  // if body
} elseif ($expr2) {
  // elseif body
} else {
  // else body;
}
```

SWITCH

- 当大于两个 if 时，请使用 switch 代替 if，如：

```
<?php
switch ($expr) {
    case 0:
        echo 'First case, with a break';
        break;
    case 1:
        echo 'Second case, which falls through';
        break;
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        break;
    default:
        echo 'Default case';
}
```

一些经验教训

- 读取网址中参数，可以用$_GET的，不要用$_SERVER[‘REQUEST_URI’]或者$_SERVER[‘QUERY_STRING’]
- drupal中定义了arg(0)、arg(1)等，比自己读取$_SERVER[‘REQUEST_URI’]好
- drupal中操作数据库用Drupal提供的API比用原生sql语句好
- 结论：尽量多使用Drupal提供的API