# 服务端业务编写规范

## 一、公用部分

数据库中存放用户输入数据

模板输出的时候需要转义

1、如何创建新的仓库

2、更新仓库

3、配置文件（系统配置，业务配置）

不允许有业务配置文件,部署配置全部放在env文件中.业务相关的配置应该是放在对应model的属性中,通过对应的get方法访问

4、版本相关

内部接口或者api接口只维护最新的5个版本

5、单元测试

## 二、service层

数据验证的处理

在更新数据前至少得从数据库获取一次数据,保证被更新的数据存在,没有被别的进程线更新,还有其他的一些业务逻辑条件都可以了,才能更新数据

数量自增的操作要注意大并发的情况,应该使用数据库或对应存储的原子性操作,而不能将数据取出后加1再存入数据库

对于主要业务流程,提倡多记录info类的日志,方便debug,同时也可以增强代码的健壮性.通过开发环境和线上环境env文件内容的不同,可以将不同level的日志发送到日志服务

对任何数据的更改和删除的操作,需要先从获取该数据(不允许物理删除数据,归档数据除外),根据其数据状态做对应的操作.所以获取单条数据的时候千万不要deletetime=0.

## 三、web层

视图层不允许由echo函数,必须使用{{}}输出数据

ajax控制器需放在ajax文件夹下,注意注释,有doc系统扫描统一生成文档

## 四、rpc服务

## 五、php语法部分

（1）在PHP中echo、exit(die)、return、continue、break、include、include\_once、require、require\_once等都属于语法结构，大部分语法结构都有两种形式：  
echo 'This is a string';  
echo('This is a string');  
在PHP规定的允许使用的格式下，尽可能使用前一种语法结构的格式，而不要使用函数参数/表达式的形式，仅在参数包含表达式时才需要用括号将其括起来。当返回一个变量时通常不用括号，也建议不要用，这样既可以降低 PHP 的负担，又可以避免一些错误（见下）。   
注: exit(die)只能使用括号结构  
（2）由于echo()是语法结构、没有返回值，所以速度比print()快。在输出多个字符串时，请使用echo的多参数方式（逗号,间隔），会比字符串连接方式 (点.间隔)有更好的性能。

### 5.1 站点自身的规则约束

1. 对在第一屏无法显示的图片建议进行延迟加载处理。  
2. 如没有特别要求，所有的JavaScript代码必须在页面最底部载入。  
3. 所有发布出去的Javascript代码必须进行初级混淆压缩处理（框架处理）。  
4. 前后通信数据格式统一使用JSON格式，不使用XML。  
5. 所有JavaScript代码均使用utf-8编码。  
6. 修改数据的cgi均应使用post方式获取数据  
7. 框架BLL层只能通过dao获取配置，其他数据均需要从接口中传入  
8. 不允许物理删除数据，而是更新数据状态值来完成数据无效化

### 5.2 关于安全漏洞及html转义

1. 所有用户输入的内容都是不可信的，都需要转义（框架处理）。  
2. 输出数据时需要转义。包含两种情况：php输出html页面，可以直接使用$aHtml数组

### 5.3其他

* 不建议使用eval

不推荐使用eval来执行脚本。除非js中用来解释json数据。

关联数组当中，请用单引号将key的字符串嵌套（例如：$row['id']比$row[id]快7倍），如果key是变量则不要使用引号嵌套：

1）对于提交/返回值，尽可能的使用===来检查类型是否匹配，这一点对于区分false,null, '',0四种空值尤其有用（定义了的变量但未定义值，其值为null）。函数返回值请用true和false来代替1和0 。  
（2）对于比较字符串，必须使用strcmp()或者===，这样才进行类型检查。例如：var\_dump('01' == 1)，返回结果为true，但是var\_dump('01' === 1)的结果就是false了  
（3）对于in\_array()函数，请加入第三个参数，设置为true，例如： in\_array('01', array('1'))，将不判断类型，返回结果为true。但是如果使用in\_array('01', array('1'), true)，将判断类型，返回结果为false。

递增/递减运算符

（1）不允许在表达式中使用递增运算符 (前加+$i，后加$i+)和递减运算符 (前减-$i，后减$i-)，这些运算符只能独自一行使用。  
（2）递增／递减运算符不影响布尔值。递减 NULL 值也没有效果，但是递增 NULL 的结果是 1 。字符串变量只能递增（使用PERL的方式而非C的方式），不能递减。例如：   
/\* wrong \*/  
$array[++$i] = $j;  
$array[$i++] = $k;   
/\* right \*/  
$i++;  
$array[$i] = $j;   
$array[$i] = $k;  
$i++;

* 递增使用前加+$i比后加$i+快，递减使用前减-$i比后减$i-快；
* 在方法里递增、递减局部变量是最快的，在函数里调用局部变量也是；
* 递增、递减一个全局变量比局部变量要慢两倍；
* 递增、递减一个对象的属性 (例如 $this->prop++) 比局部变量慢3倍；
* 递增、递减一个未定义的局部变量比一个预定义过的慢9-10倍；

### 4．三元条件运算符

三元条件运算符更适用于赋值，使得程序更为简洁。三元条件运算符不适合用作函数调用或者其它任何复杂的代码，这将会让代码变得晦涩难懂。 例如：   
/\* Bad place to use them \*/  
(($i < $size) && ($j > $size)) ? do\_stuff($foo) : do\_stuff($bar);   
/\* OK place to use them \*/  
$min = ($i < $j) ? $i : $j;

### 5．未初始化的变量

请不要使用未初始化的变量！对于用户在HTML中提交的输入值，需要使用isset()来检测是否已被初始化。例如：   
/\* Error way \*/  
if ($\_POST['forum'])   
{  
...  
}   
/\* Right way \*/  
if (isset($\_POST['forum']))   
{  
...  
}

### 6．变量/数组是否为空的检测

php中非特殊情况尽量不要使用empty()来判断变量是否为空，因为empty()函数中包含了多种可能性，会降低代码可读性。  
The following things are considered to be empty:

* "" (an empty string)
* 0 (0 as an integer)
* 0.0 (0 as a float)
* "0" (0 as a string)
* NULL
* FALSE
* array() (an empty array)
* var $var; (a variable declared, but without a value in a class)

### 7．正则表达式

（1）PHP支持两种风格的正则表达式，分别是[Perl-Compatible (preg)](http://pms.ipo.com/pages/createpage.action?spaceKey=kaifawendang&title=Perl-Compatible+%28preg%29&linkCreation=true&fromPageId=4063265) (<http://www.php.net/manual/en/ref.pcre.php>) 和 [POSIX Extended (ereg)](http://www.php.net/manual/en/ref.regex.php) (<http://www.php.net/manual/en/ref.regex.php>)。preg风格相对于ereg风格来说更为安全，速度更快。因此，请不要使用ereg风格的正则表达式。   
（2）在使用正则表达式的时候，首选的匹配词语定界符是斜线/，如果您的匹配词语中含有斜线，那么您可以使用#作为匹配词语定界符。 例如：  
/\* Error way \*/  
if (ereg('something', $myvar))   
{  
...  
}   
/\* Right way \*/  
if(preg\_match('/something/', $myvar)){  
...  
}   
/\* Alternative \*/  
if(preg\_match('#/etc/php.ini#', $myvar)){  
...  
}  
（3）正则表达式最好使用单引号, "\w"是无效转义，应该写成'\w'或 "  
w"。  
（4）尽可能使用strncasecmp()、strpbrk()、stripos() 等str系列字符串函数而不是正则表达式。另外同样是str系列函数，性能也有差异：str\_replace()比preg\_replace()快，但strtr()比str\_replace()还要快4倍 。  
（5）如果只是需要做字符串的格式类型检查，可以直接使用ctype函数库，无需使用正则或者字符串匹配。

### 8．使用file\_get\_contents/file\_put\_contents

（1）PHP4.3.0以上版本提供了file\_get\_contents()函数，用这个函数读取文件内容为一个字符串，其速度是使用fopen()/file()的三十倍。我们要求所有读取文件内容的程序，除非必须确保不允许被服务器cache，否则一律使用file\_get\_contents()函数。  
（2）通过fopen()打开、flock()锁定、fwrite()写入、fclose()关闭，这四个函数组合实现对文件的写操作，如果在Daemon/Server中的请不要使用这种方式，一旦文件被其它进程删除，则数据永远无法写入（因为每次写入前不检查文件是否存在）。PHP5.0以上版本提供了file\_put\_contents()函数，该函数支持FILE\_APPEND和LOCK\_EX等Flag，可以安全的写入文件。

### 9．Global/Request/Session/Server 变量

以上变量均能通过框架中特定方法获取（特殊情况需review）。

### 10．输出header()

（1）对于需要输出的header()，请调用框架中发的方法，必须将每条header调用一次，不要将多个header放在同一行内，出于安全因素考虑PHP早已不再支持这种组合header输出。  
（2）如果print/echo输出HTML会被Apache加上缓存头，如果不希望被缓存，则需要加上上面的输出无缓存头代码来避免。

### 11．上传文件类型

请不要使用PHP的$\_FILES中的type来判定文件的类型，此类型由浏览器提供，容易被黑客伪造。请使用getimagesize()函数来进行判定：  
$info = getimagesize($file);  
$width = $info[0];  
$height = $info[1];  
$type = $info[2];  
$imgstr = $info[3];  
$mime = $info['mime'];  
其中$type，1 = GIF，2 = JPG，3 = PNG，4 = SWF，5 = PSD，6 = BMP，7 = TIFF(intel byte order)，8 = TIFF(motorola byte order)，9 = JPC，10 = JP2，11 = JPX，12 = JB2，13 = SWC，14 = IFF，15 = WBMP，16 = XBM

### 12．字符串的大小写转换

请不要使用PHP的strtolower()函数来进行字符串的大小写转换，这在安装非中文系统的服务器下会导致将汉字转换为乱码。请使用strtr($string, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') 来进行大小写转换。

### 13．检查字符串长度

（1）在PHP底层zval结构体里保存了字符串长度，strlen()可以直接获取而无需计算，但仍旧有一次函数调用，速度上使用 isset() 语法结构会更快。例如：   
if(strlen($foo) < 5){ … }  
vs.  
if(!isset($foo{5})){ … }   
（2）使用isset($foo{5})比strlen($foo)<5更快

### 14．包含文件

（1）对于include()/requrie()所包含的文件请尽量使用绝对路径或者./开头的相对路径。  
（2）对于include\_once()/require\_once()所包含的文件必须使用绝对路径（每次调用都会打开目标文件，如果用绝对路径在PHP 5.2以上不存在这个问题）。  
（3）目录通常叫做Dir，文件全路径通常叫做Path，文件名本身通常叫做FileName

### 15．遍历数组

（1）如果需要遍历数组，请使用foreach()循环，避免使用for()或者while()。  
（2）尽量避免在循环内部修改数组key的值（会导致写时复制）

### 16．避免修改for循环内的条件值

避免重新计算for循环内的条件值（比如count()/strlen()/sizeof()等），因为PHP的解析器不能够删除循环不变量。请在循环前就设定循环的条件值，而不是在循环内。   
/\* Error way \*/  
for($i=0; $i < count($array); $i++){  
...  
}   
/\* Right way \*/  
$total = count($array);  
for($i=0; $i < $total; $i++){  
...  
}

### 17．用常量代替函数

如果存在和函数等价的常量，则不要使用函数。  
php\_uname('s') == PHP\_OS;  
php\_version() == PHP\_VERSION;  
php\_sapi\_name() == PHP\_SAPI;

### 18．目录权限

创建、修改目录，需要设置正确的权限，必须先使用umask(0000)，然后再mkdir()或chmod()，例如：  
if(!is\_dir($dir)){  
umask(0000);  
mkdir($dir, 0777);  
}   
if(!is\_writeable($dir)){  
umask(0000);  
chmod($dir, 0777);  
}

### 19．临时文件

请使用tmpfile()、tempnam()来创建临时文件。

 20. 静态方法和静态调用

（1）如果类的某个方法是静态的，那就用static显式定义成静态方法，这样可以快4倍。  
（2）如果类的成员并非静态static，则在类的外部静态调用类的成员不要使用静态方法，静态方法只能调用类的静态成员。  
class Foo{  
function bar(){  
echo 'foobar';  
}  
}   
$foo = new Foo();   
/\* Right way \*/  
$foo->far();   
/\* Error way \*/  
Foo::bar();

### 21．时间问题

（1）WEB服务器下运行的业务代码如果只是取当前时间，请不要使用time()，框架中的方法（PHP 5.1以上版本下）  
（2）WEB服务器中是没有$\_SERVER['REQUEST\_TIME']变量的。  
（3）请在程序中指定时区 date\_default\_timezone\_set('Asia/Shanghai')。  
（4）如果需要取微秒，请使用microtime(true)，返回浮点数。

### 22．绝对路径

所有以目录或文件路径做为参数的函数，basename、file\_exists、opendir等等函数，均要求使用绝对路径

### 23．错误信息

开发时，系统会将所有异常及错误都显示出来。使用框架扩展出来的异常，错误码定义常量，以ERRC\_开头