**一、基本约定**

**1、源文件**

（1）、纯PHP代码源文件只使用 <?php 标签，省略关闭标签 ?> ；

（2）、源文件中PHP代码的编码格式必须是无BOM的UTF-8格式；

（3）、使用 Unix LF(换行符)作为行结束符；

（4）、一个源文件只做一种类型的声明，即，这个文件专门用来声明Class, 那个文件专门用来设置配置信息，别混在一起写；

**2、缩进**

使用Tab键来缩进，每个Tab键长度设置为4个空格；

**3、行**

一行推荐的是最多写120个字符，多于这个字符就应该换行了，一般的编辑器是可以设置的。

**4、关键字 和 True/False/Null**

PHP的关键字，必须小写，boolean值：true，false，null 也必须小写。

下面是PHP的“关键字”，必须小写：

'\_\_halt\_compiler', 'abstract', 'and', 'array', 'as', 'break', 'callable', 'case', 'catch', 'class', 'clone', 'const', 'continue', 'declare', 'default', 'die', 'do', 'echo', 'else', 'elseif', 'empty', 'enddeclare', 'endfor', 'endforeach', 'endif', 'endswitch', 'endwhile', 'eval', 'exit', 'extends', 'final', 'for', 'foreach', 'function', 'global', 'goto', 'if', 'implements', 'include', 'include\_once', 'instanceof', 'insteadof', 'interface', 'isset', 'list', 'namespace', 'new', 'or', 'print', 'private', 'protected', 'public', 'require', 'require\_once', 'return', 'static', 'switch', 'throw', 'trait', 'try', 'unset', 'use', 'var', 'while', 'xor'

**5、命名**

（1）、类名 使用大驼峰式（StudlyCaps）写法；

（2）、（类的）方法名 使用小驼峰（cameCase）写法；

（3）、函数名使用 小写字母 + 下划线 写法，如 function http\_send\_post()；

（4）、变量名 使用小驼峰写法，如 $userName；

**6、代码注释标签**

如 函数注释、变量注释等，常用标签有 [@package](https://phpdoc.org/docs/latest/references/phpdoc/tags/package.html)、[@var](https://phpdoc.org/docs/latest/references/phpdoc/tags/var.html" \t "_blank)、[@param](https://phpdoc.org/docs/latest/references/phpdoc/tags/param.html" \t "_blank)、[@return](https://phpdoc.org/docs/latest/references/phpdoc/tags/return.html" \t "_blank)、[@author](https://phpdoc.org/docs/latest/references/phpdoc/tags/author.html" \t "_blank)、[@todo](https://phpdoc.org/docs/latest/references/phpdoc/tags/todo.html" \t "_blank)、[@throws](https://phpdoc.org/docs/latest/references/phpdoc/tags/throws.html" \t "_blank)

必须遵守 phpDocument 标签规则，不要另外去创造新的标签，更多标签查看 [phpDocument官网](https://phpdoc.org/docs/latest/references/phpdoc/tags/param.html" \t "_blank)

**7、业务模块**

（1）、涉及到多个数据表 更新/添加 操作时，最外层要用事务，保证数据库操作的原子性；

（2）、Model层，只做简单的数据表的查询；

（3）、业务逻辑统一封装到 Logic层；

（4）、控制器只做URL路由，不要当作 业务方法 调用；

（5） 、 控制器层不能出现SQL操作语句，如 ThinkPHP框架的 where()、order() 等模型方法，

即，控制器中，不要出现类似这样的SQL语句：D('XXX')->where()->order()->limit()->find();

where()、order()、limit() 等SQL方法只能出现在 Model层、业务层！

**二、代码样式风格**

**1、命名空间(Namespace) 和 导入(Use)声明**

先简单文字描述下：

1. 命名空间(namespace)的声明后面必须有一行空行；
2. 所有的导入(use)声明必须放在命名空间(namespace)声明的下面；
3. 一句声明中，必须只有一个导入(use)关键字；
4. 在导入(use)声明代码块后面必须有一行空行；

用代码来说明下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php  namespace Lib\Databases; // 下面必须空格一行    class Mysql {    } |

namespace下空一行，才能使用use，再空一行，才能声明class

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  namespace Lib\Databases; // 下面必须空格一行    use FooInterface; // use 必须在namespace 后面声明  use BarClass as Bar;  use OtherVendor\OtherPackage\BazClass; // 下面必须空格一行    class Mysql {    } |

**2、类(class)，属性(property)和方法(method)**

（1）、继承(extends) 和实现(implement) 必须和 class name 写在一行。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php  namespace Lib\Databaes;    class Mysql extends ParentClass implements \PDO, \DB { // 写一行    } |

（2）、属性(property)必须声明其可见性，到底是 public 还是 protected 还是 private，不能省略，也不能使用var, var是php老版本中的什么方式，等用于public。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  namespace Lib\Databaes;    class Mysql extends ParentClass implements \PDO, \DB { // 写一行      public $foo = null;      private $name = 'yangyi';      protected $age = '17';  } |

（3）、方法(method)，必须 声明其可见性，到底是 public 还是 protected 还是 private，不能省略。如果有多个参数，第一个参数后紧接“,” ，再加一个空格：function\_name ($par, $par2, $pa3), 如果参数有默认值，“=”左右各有一个空格分开。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  namespace Lib\Databaes;    class Mysql extends ParentClass implements \PDO, \DB { // 写一行      public getInfo($name, $age, $gender = 1) { // 参数之间有一个空格。默认参数的“=”左右各有一个空格，) 与 { 之间有一个空格        }  } |

（4）、当用到抽象(abstract)和终结(final)来做类声明时，它们必须放在可见性声明 （public 还是protected还是private）的前面。而当用到静态(static)来做类声明时，则必须放在可见性声明的后面。

直接上代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?php  namespace Vendor\Package;    abstract class ClassName {      protected static $foo; // static放后面      abstract protected function zim(); // abstract放前面        final public static function bar() { // final放前面，static放最后。          // 方法主体部分      }  } |

**3、控制结构**

控制接口，就是 if else while switch等。这一类的写法规范也是经常容易出现问题的，也要规范一下。

（1）、if，elseif，else写法，直接上规范代码吧：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  if ($expr1) { // if 与 ( 之间有一个空格，) 与 { 之间有一个空格    } elseif ($expr2) { // elesif 连着写，与 ( 之间有一个空格，) 与 { 之间有一个空格    } else { // else 左右各一个空格    } |

（2）、switch，case 注意空格和换行，还是直接上规范代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php  switch ($expr) { // switch 与 ( 之间有一个空格，) 与 { 之间有一个空格      case 0:          echo 'First case, with a break'; // 对齐          break; // 换行写break，也对齐。      case 1:          echo 'Second case, which falls through';          // no break      case 2:      case 3:      case 4:          echo 'Third case, return instead of break';          return;      default:          echo 'Default case';          break;  } |

（3）、while，do while 的写法也是类似，上代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  while ($expr) { // while 与 ( 之间有一个空格， ) 与 { 之间有一个空格    }    do { // do 与 { 之间有一个空格    } while ($expr); // while 左右各有一个空格 |

（4）、for的写法

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  for ($i = 0; $i < 10; $i++) { // for 与 ( 之间有一个空格，二元操作符 "="、"<" 左右各有一个空格，) 与 { 之间有一个空格    } |

（5）、foreach的写法

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  foreach ($iterable as $key => $value) { // foreach 与 ( 之间有一个空格，"=>" 左右各有一个空格，) 与 { 之间有一个空格    } |

（6）、try catch的写法

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  try { // try 右边有一个空格    } catch (FirstExceptionType $e) { // catch 与 ( 之间有一个空格，) 与 { 之间有一个空格    } catch (OtherExceptionType $e) { // catch 与 ( 之间有一个空格，) 与 { 之间有一个空格    } |

**4、注释**

（1）、行注释

// 后面需要加一个空格；

如果 // 前面有非空字符，则 // 前面需要加一个空格；

（2）、函数注释

参数名、属性名、标签的文本 上下要对齐；

在第一个标签前加一个空行；

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?php  /\*\*   \* This is a sample function to illustrate additional PHP   \* formatter options.   \*   \* @param        $one   The first parameter   \* @param int    $two   The second parameter   \* @param string $three The third parameter with a longer   \*                      comment to illustrate wrapping.   \* @return void   \* @author  52php.cnblogs.com   \* @license GPL   \*/  function foo($one, $two = 0, $three = "String") {    } |

**5、空格**

（1）、赋值操作符（=，+= 等）、逻辑操作符（&&，||）、等号操作符（==，!=）、关系运算符（<，>，<=，>=）、按位操作符（&，|，^）、连接符（.） 左右各有一个空格；

（2）、if，else，elseif，while，do，switch，for，foreach，try，catch，finally 等 与 紧挨的左括号“(”之间有一个空格；

（3）、函数、方法的各个参数之间，逗号（","）后面有一个空格；

**6、空行**

（1）、所有左花括号 { 都不换行，并且 ｛ 紧挨着的下方，一定不是空行；

（2）、同级代码（缩进相同）的 注释（行注释/块注释）前面，必须有一个空行；

（3）、各个方法/函数 之间有一个空行；

（4）、namespace语句、use语句、clase语句 之间有一个空行；

（5）、return语句

如果 return 语句之前只有一行PHP代码，return 语句之前不需要空行；

如果 return 语句之前有至少二行PHP代码，return 语句之前加一个空行；

（5）、if，while，switch，for，foreach、try 等代码块之间 以及 与其他代码之间有一个空行；

【**参考示例 汇总**】

参考1：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  namespace Lib\Databaes;    class Mysql extends ParentClass implements \PDO, \DB { // 写一行      public getInfo($name, $age, $gender = 1) { // 参数之间有一个空格。默认参数的“=”左右各有一个空格，) 与 { 之间有一个空格        }  } |

参考2:

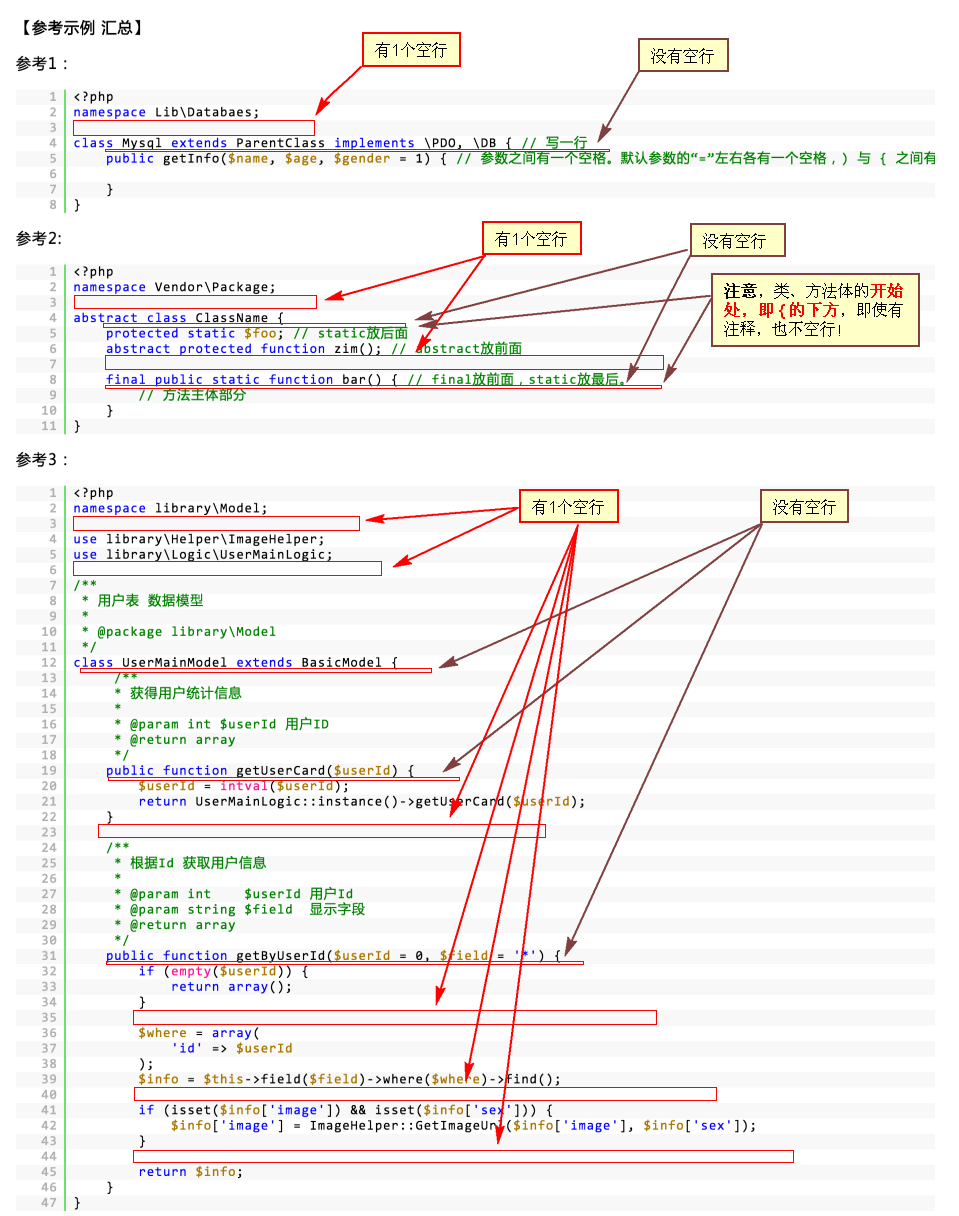
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?php  namespace Vendor\Package;    abstract class ClassName {      protected static $foo; // static放后面      abstract protected function zim(); // abstract放前面        final public static function bar() { // final放前面，static放最后。          // 方法主体部分      }  } |

参考3：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | <?php  namespace library\Model;    use library\Helper\ImageHelper;  use library\Logic\UserMainLogic;    /\*\*   \* 用户表 数据模型   \*   \* @package library\Model   \*/  class UserMainModel extends BasicModel {       /\*\*       \* 获得用户统计信息       \*       \* @param int $userId 用户ID       \* @return array       \*/      public function getUserCard($userId) {          $userId = intval($userId);          return UserMainLogic::instance()->getUserCard($userId);      }        /\*\*       \* 根据Id 获取用户信息       \*       \* @param int    $userId 用户Id       \* @param string $field  显示字段       \* @return array       \*/      public function getByUserId($userId = 0, $field = '\*') {          if (empty($userId)) {              return array();          }            $where = array('id' => $userId);          $info = $this->field($field)->where($where)->find();            if (isset($info['image']) && isset($info['sex'])) {              $info['image'] = ImageHelper::GetImageUrl($info['image'], $info['sex']);          }            return $info;      }  } |

参考4：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | $serv = new swoole\_server("127.0.0.1", 9502);    // sets server configuration, we set task\_worker\_num config greater than 0 to enable task workers support  $serv->set(array('task\_worker\_num' => 4));    // attach handler for receive event, which have explained above.  $serv->on('receive', function($serv, $fd, $from\_id, $data) {      // we dispath a task to task workers by invoke the task() method of $serv      // this method returns a task id as the identity of ths task      $task\_id = $serv->task($data);      echo "Dispath AsyncTask: id=$task\_id\n";  });    // attach handler for task event, the handler will be executed in task workers.  $serv->on('task', function ($serv, $task\_id, $from\_id, $data) {      // handle the task, do what you want with $data      echo "New AsyncTask[id=$task\_id]".PHP\_EOL;        // after the task task is handled, we return the results to caller worker.      $serv->finish("$data -> OK");  });    // attach handler for finish event, the handler will be executed in server workers, the same worker dispatched this task before.  $serv->on('finish', function ($serv, $task\_id, $data) {      echo "AsyncTask[$task\_id] Finish: $data".PHP\_EOL;  });    $serv->start(); |



**总结：所有左花括号 { 都不换行，并且 ｛ 紧挨着的下方，一定不是空行！**

上面的 代码样式规范，参考了 Java、JavaScript、Objective-C、Go 等开发语言的规范！

Java语言对中国程序员影响太深，大部分人还是习惯把 左花括号 **{**不换行！

书写原则：做到 代码紧凑 而又不失 小模块化 ！

**PSR-4 规范**

PSR-4规范是刚出没多久的一条新的规范，它也是规范 自动加载(autoload)的，是对PSR-0的修改，属于补充规范，

我简单说下，主要是以下几点：

1. 废除了PSR-0中\_就是目录分割符的写法，\_下划线在完全限定类名中是没有特殊含义了。
2. 类文件名要以 .php 结尾。
3. 类名必须要和对应的文件名要一模一样，大小写也要一模一样。