编码规范 & 代码样式风格规范

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、@var、@param、@return、@author、@todo、@throws

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

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：

```php
<?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

```php
<?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。

```php
<?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。

```php
<?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。

```php
<?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：

```php
<?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：

```php
<?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：

```php
<?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：

```php
<?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

```php
<?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

```php
<?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

```php
<?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；

```php
<?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 $three The third parameter with a longer
 *                      comment to illustrate wrapping.
 * @return void
 * @author  phpgo.cnblogs.com
 * @license GPL
 */
function foo($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：

```php
<?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:

```php
<?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：

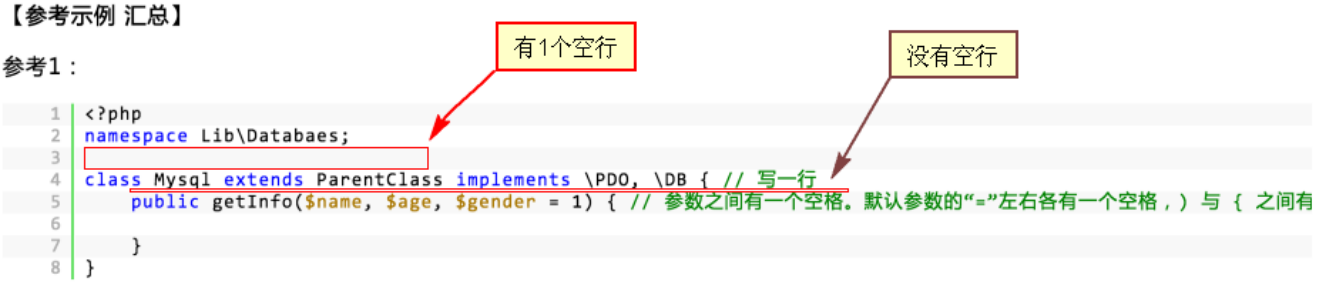
```php
<?php
namespace library\Model;

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：

```php
$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();
```

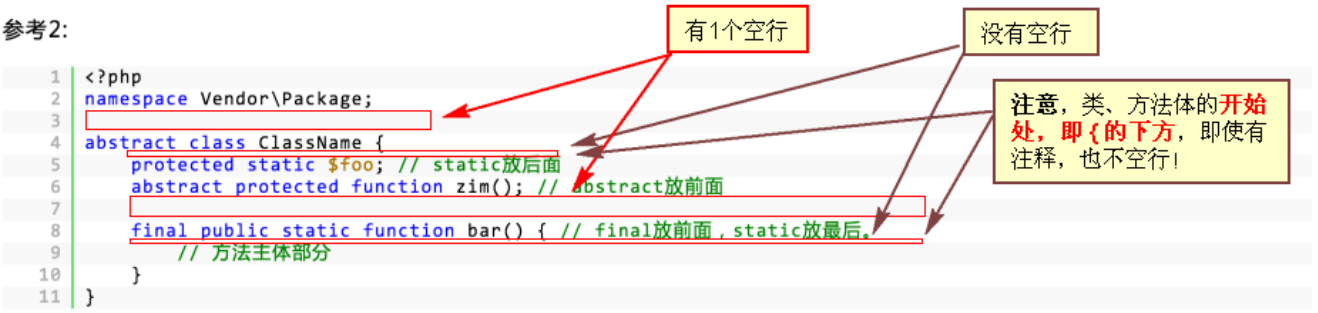
【参考示例 汇总】

参考1：



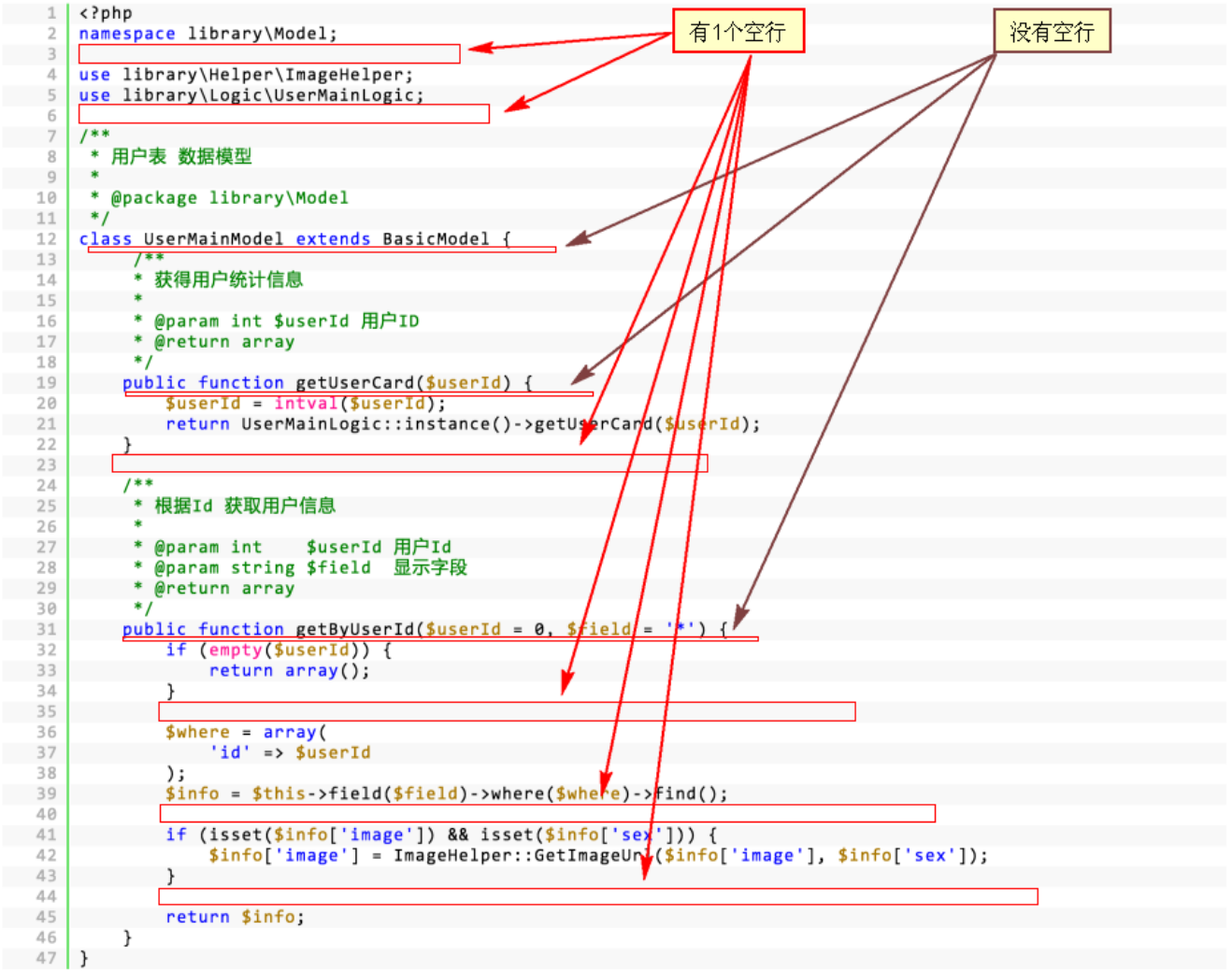

```php
<?php
namespace Lib\Databaes;

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

    }
}
```

参考2:




```php
<?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：




```php
<?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;
    }
}
```

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！