|  |
| --- |
| 杭州青橄榄网络技术有限公司 |
| 青橄榄开发规范 |
| Version 1.0 |

|  |
| --- |
| 2015-11-30 |

1. **开发规范范围**

开发规范包括项目的环境规范、命名规范、目录结构规范、数据库规范、SVN规范和代码注释等。

常用词汇将以附录形式添加。附录内容包括常用词汇中英文对照、常用注释公用模板。

1. **项目使用环境规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **所属** | **类别** | **使用技术** | **版本** |
| 系统环境 | 后台语言 | PHP | 5.4+ |
| 数据库 | MySQL | 5.6.12+ |
| 服务器 | Apache2.4+或nginx1.6+ | 视情况而定 |
|  |  |  |  |
| 前台配置 | 样式表 | CSS | Css 3.0 |
| 文本语言 | HTML | 5.0 |
| JS框架 | jQuery | 1.18+ |
| 前端框架 | bootstrap | 2.3.2+ |
| 表单验证 | jquery.validate | 1.12+ |
| 图表创建 | hicharts | 4.0.1+ |
| 富文本编辑器 | UEditor | 1.4.3+ |
|  |  |  |
| 后台配置 | 后台框架 | ThinkPHP | 3.2.0及以上 |

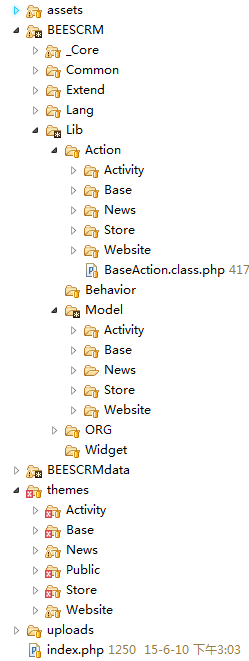
1. **项目结构规范**

**3.1 项目环境划分**

项目环境按照开发中不同阶段可以分为：开发环境development、测试环境test、正式环境product。

**3.2 目录结构划分**

公司PHP项目一律采用ThinkPHP框架开发，以简化框架的学习与文档阅读。项目目录结构如下图所示：

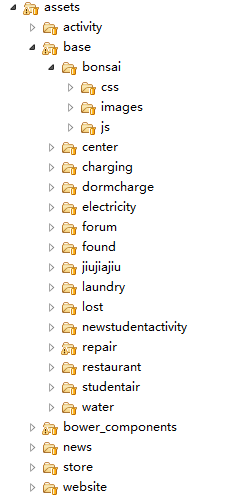


项目的引导文件与路由文件全部放在根目录中。TP核心文件在\_Core文件夹中。

项目新建后，按照模块划分，每一个模块一个文件夹。上图中Activity与Base就是两个独立的模块,每个模块下有各自的Controller\Model\View.

**3.3** assets**文件夹说明**

系统中assets的用于项目的公用资源文件的存放。assets文件夹下项目目录标准结构如图所示：



每个模块对应各自的资源文件所在的文件夹，**文件夹名称要求和模块名称一致，强制要求一律用小写字母，不允许使用下划线等其他字符。**细化到模块，如 bonsai模块资源，包含css,images,js三个文件夹。其中css文件夹中存放模块需要用到的CSS文件。fonts存放项目中的需要用的字体可以为空文件夹。images用于存放素材图片。js则为项目的Javascript文件。项目中需要的插件则全部以文件的形式存放于对应项目的公用资源文件夹中，必要时可新建plugin插件文件夹。

.htaccess和index.php 是项目的主控文件不能予以修改，BEESCRM/conf中的配置文件不允许随意修改，禁止提交至项目SVN 。

1. **命名规范**

该命名法则主要针对于后台编写，但同样适用于前台的JS文件编写。前台CSS文件的编写遵循CSS自身的编写规范。

**4.1 驼峰法则**

所有的命名统一遵循驼峰法则，即单词与单词之间不以空格、连接号或者底线连结，直接放置。驼峰法则又可分为大驼峰法则与小驼峰法则。

大驼峰法则：每一个单词的首字母都采用大写字母

小驼峰法则：第一个单词小写字母开始，第二个单词的首字母大写

例如：BonsaiAction(大驼峰)、BonsaiAction(大驼峰)、getList(小驼峰)。

驼峰法则中不允许使用单词缩写。

**4.2 法则的使用范围**

代码编写中，类名统一采用大驼峰法则编写，变量名和方法名一律使用小驼峰法则。

代码中常量，则不再使用驼峰法则，所有的常量所有字母全部大写。单词与单词之间使用下划线分隔。

类的私有变量采用小驼峰法则，但是变量需以下划线开头。实例如表3-1所示。

|  |  |
| --- | --- |
| 类型 | 示例 |
| 类名 | BonsaiAction、ShopModel |
| 方法名 | Index()、edit()、addStaff() |
| 变量名 | uid、id、money、lastNumber |
| 常量名 | MAX\_VALUE、MIN\_VALUE |
| 私有变量 | \_totalNumber |
| 私有方法 | \_cutNumber() |

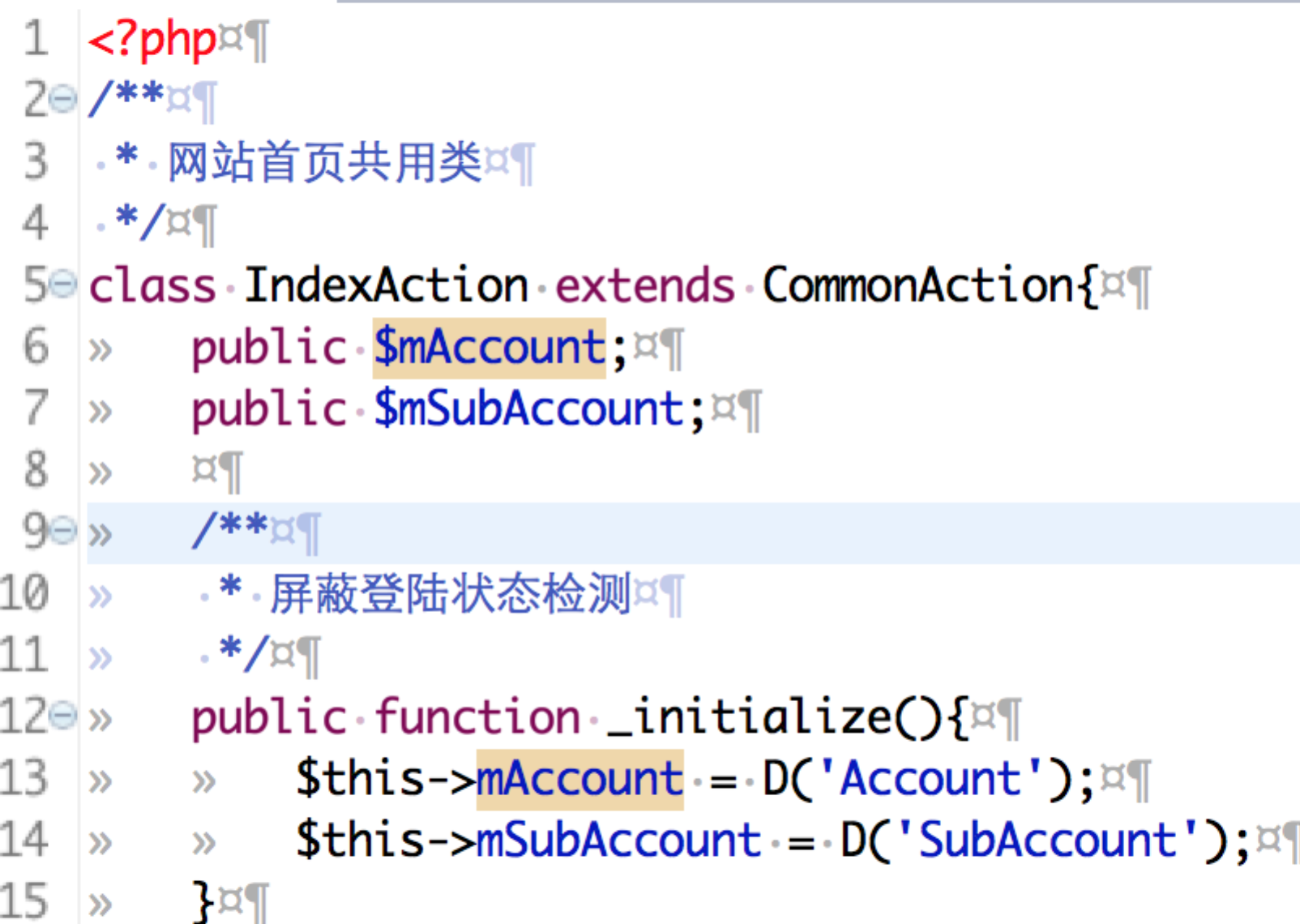
**4.3 特殊命名**

在代码编写过程中，我们时常需要实例化模型。实例化出来的模型变量需要以小写m开头，表示出其类型。例如：

$mBonsai = M(‘bonsai);

$mBonsai = D(‘bonsai);

类文件内多次引用同一个model，建议定义为全局变量使用。例如：



类文件引入实例化采用小写c开头。例如：

vendor(‘Wechat.WechatApi’);

$cWechatApi = new WechatApi($accountId);

**4.4 固定命名**

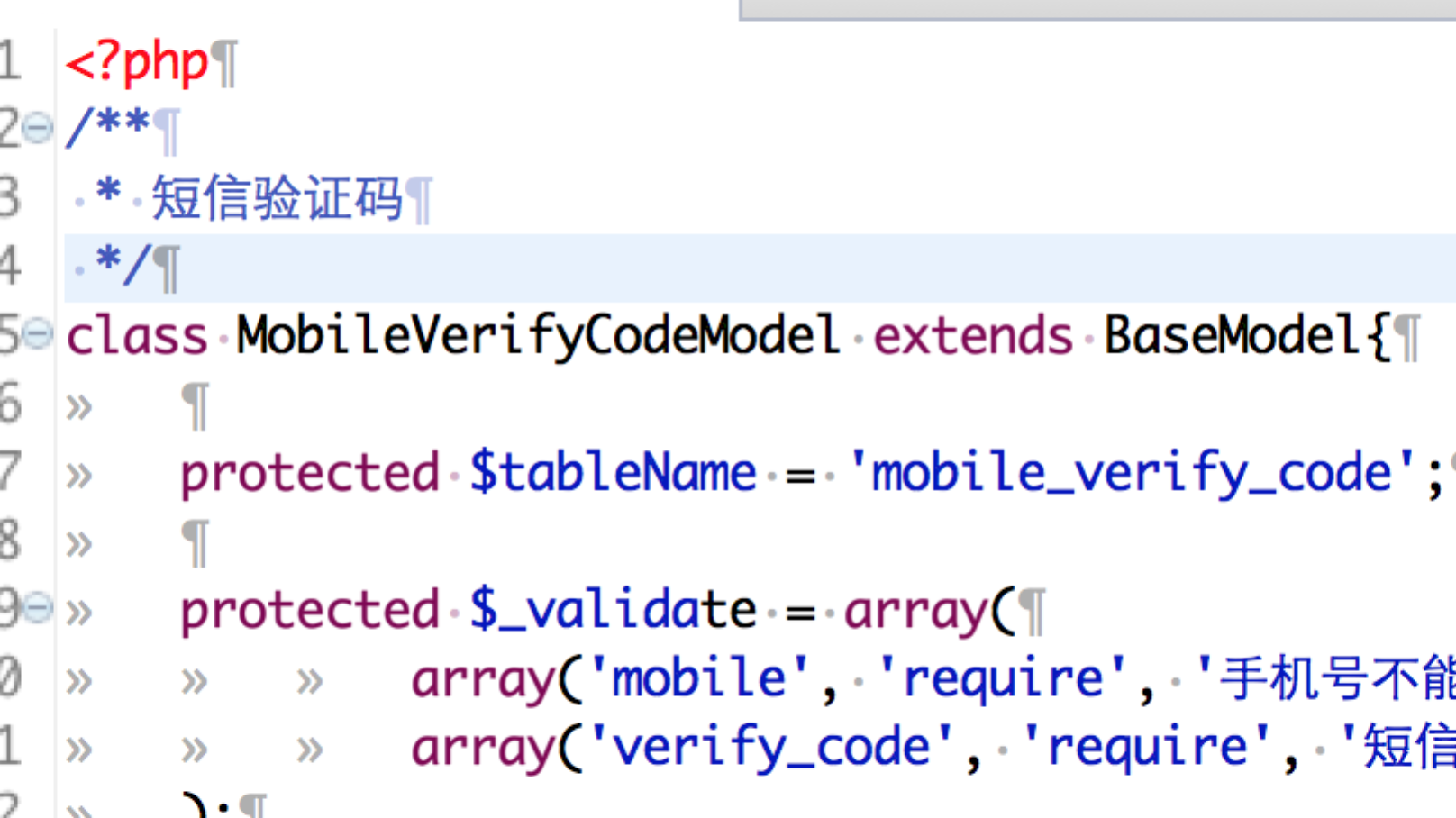
数据查询返回参数名建议采用与数据相关的名称。例如：

$accountInfo = $mAccount->find();

$subAccountList = $mSubAccount->getList();

**4.5 文件注释**

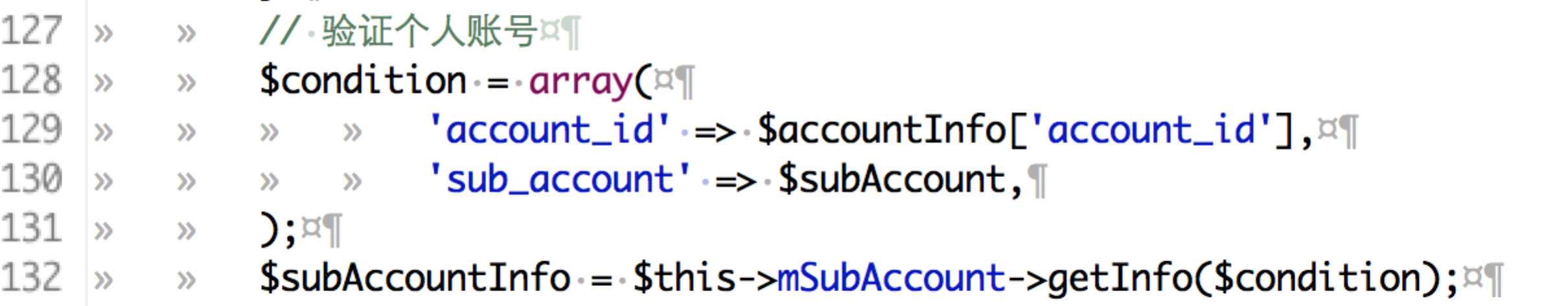
1. 类文件注释：类文件注释只描述文件用途或名称。



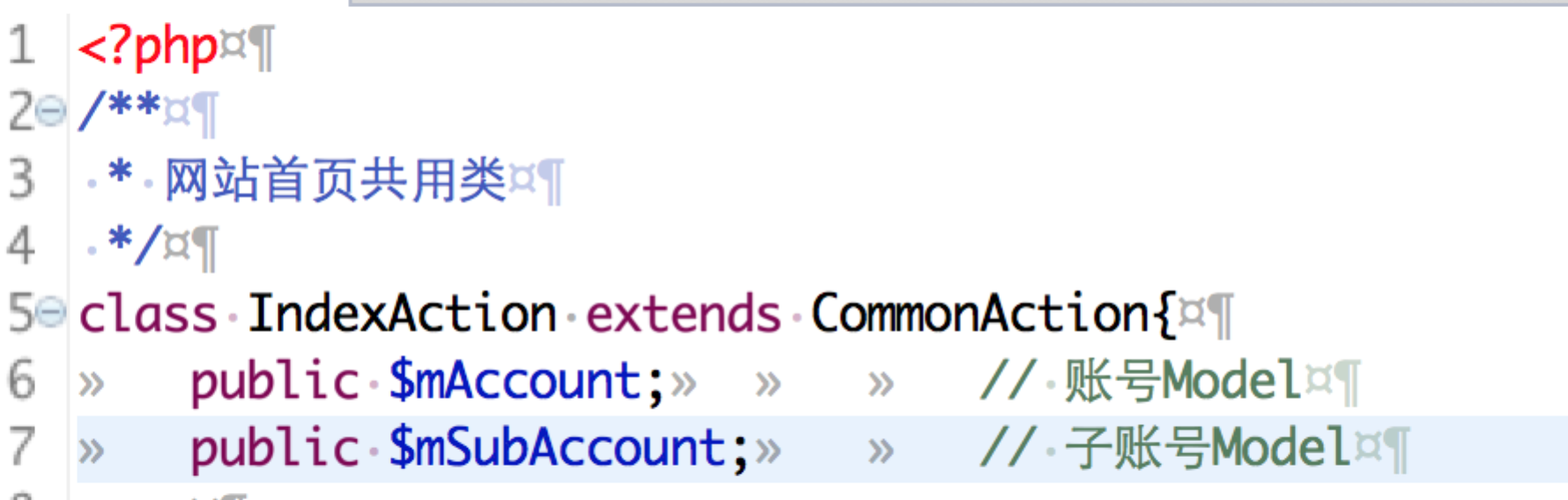
2. 方法注释：方法注释内容包括方法名称或用户，传递参数以及返回参数。



3. 代码注释：采用双斜杠注释法，双斜杠和注释间以一个空格隔开。

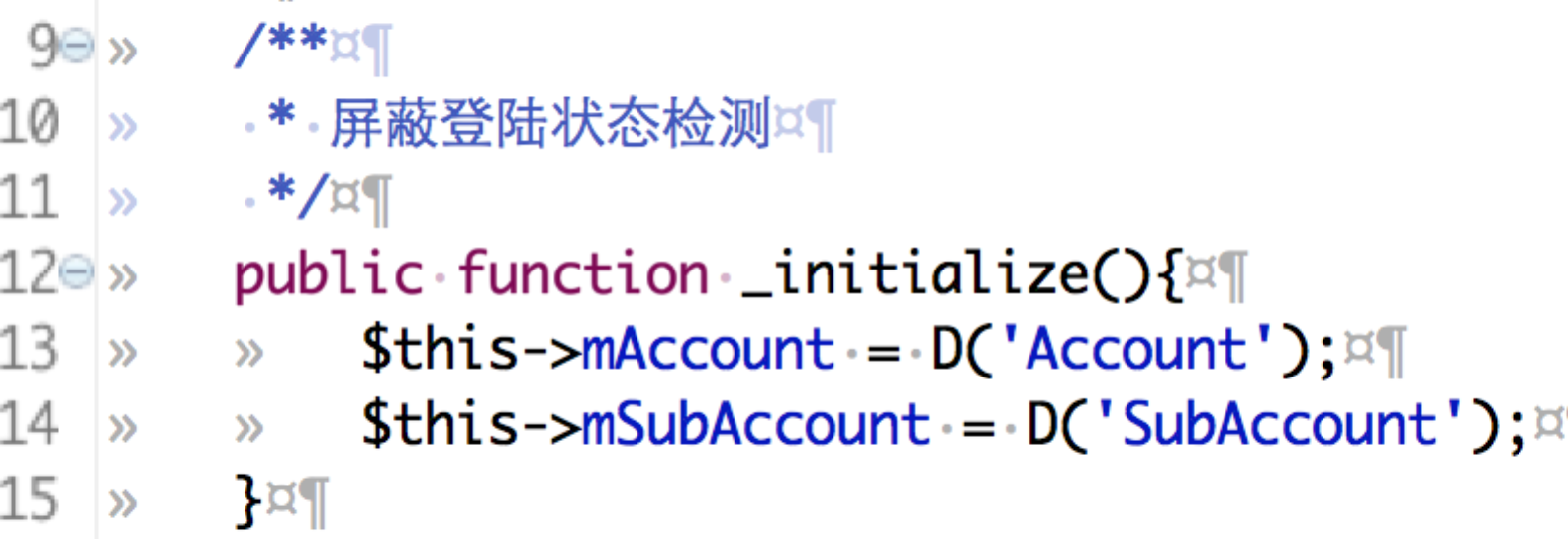


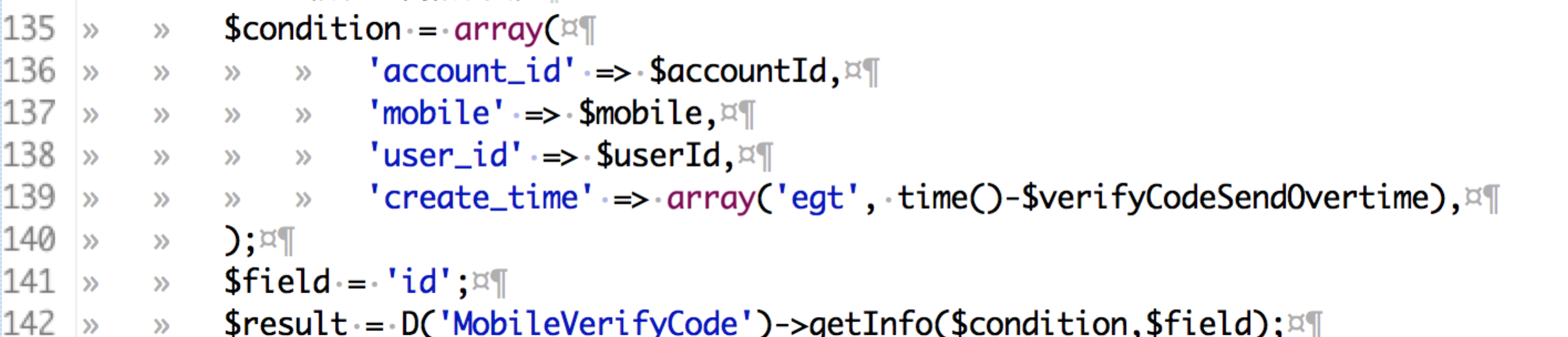
4. 全局变量注释：采用同行注释法，双斜杠和注释间以一个空格隔开。



**4.6 代码空格**

1. 运算符和数组键值和value值之间的=>采用空格隔开。





2. 数组参数之间按逗号以空格分割。



**4.7 数据查询**

1. 单条根据主键查询

$accountInfo = $mAccount->getByPk($pk);

2. 单条根据条件查询

$accountInfo = $mAccount->getInfo($condition);

3. 多条根据条件查询

$accountList = $mAccount->getList($condition);

4. 根据条件统计总数

$accountCount = $mAccount->getCount($condition);

5. 单条根据字段查询

$accountInfo = $mAccount->getByAccount($account);

6. 单个字段根据字段查询（获取多个字段当前不支持）

$accountId = $mAccount->getFieldByAccount($account，‘account\_id’);

7．页面渲染和AJAX提交方法分开，AJAX提交统一采用do开头

页面渲染方法add(),提交对应为doAdd()

**4.8 数据更新**

1. 数据更新统一采用update

$return = $mAccount->update($update);

2. 完成查询后，可使用对象更新

1. $mUser = D(’User’);
2. $mUser->find(1); // 查找主键为1的数据
3. $mUser->name = 'TOPThink'; // 修改数据对象
4. $mUser->save(); // 保存当前数据对象
5. **数据库规范**

**5.1 数据库编码**

数据库的字符集一律使用：utf8

排序规则一律使用：utf8\_general\_ci

**5.2 数据库的命名规范**

**数据库名称：**数据库名称除正式环境以外,需要注明所属域名、所属环境，互相之间使用下划线分开。数据库名称一律使用小写，且不允许使用缩写。基本结构为：数据库名\_所属域名\_环境。

**数据库表名：**数据库表名需要统一前缀，公司约定，主数据库表前缀一律使用beescrm\_。如涉及多个数据库连接，则可以在特定环境中临时商讨决定。

除开前缀以外，数据库表名一律与对应的对象使用相同名称。如果涉及过个对象的关联则使用下划线分开多个变量名，变量名一律使用小写，且不允许缩写不允许使用复数。

格式：16860\_对象名[\_对象名[..]]

例如：16860\_member\_address

**数据库字段名：**数据库表中的字段名，一律使用小写允许长字段名（除下划线外，超过10个字节）缩写，不允许使用复数格式。如果涉及关联表内容，则允许使用表名作为前缀，并以下划线分隔

格式：属性名[\_属性名]

例如：username\gender\staff\_id

**索引名称：**索引保持字段同样的名称，但首字母大写，多个字段相关索引间按照首字母顺序使用下划线分开。

格式：属性名\_属性名（首字母大写）

例如：Name\_Position\Id

**5.3 索引建立**

数据库表除去主键以外，唯一字段需要建立unique索引。经常使用查询的字段需要建立btree索引。

一律从1开始。

**5.4 关于数据库注释**

数据库中所有的表、字段、索引都需要有响应的注释，标明其各自的用途。

1. **SVN规范**

SVN一律使用分支管理。主干仅用作代码的合并与线上预览使用。初始版本时在主干中建立第一版，确保主干的内容没有错误，所有人检出初始版本。开发人员建立各自的分支进行开发工作。各自代码提交的时候，只能够提交到自己的分支中。

所有开发人员，下班前必须提交各自当天代码到分支中。

1. **SVN推送审核**

SVN推送至正式服前，需经过严格的测试流程。项目或活动上线之后新增的需求，待开发、测试完成之后，需由项目负责人或项目主管审核无误后才可推送至正式服。