

SportsZone运动社交

详细设计文档



2016-11-1

南京大学软件学院

张文玘

学号：141250192

| 修改人员 | 修改目的 | 更新日期 | 版本号 |
| --- | --- | --- | --- |
| 张文玘 | 完成文档初稿 | 2015-11-01 | V1.0 |

目录

[1引言 3](#_Toc465983237)

[1.1 编制目的 3](#_Toc465983238)

[1.2 词汇表 3](#_Toc465983239)

[1.3参考资料 3](#_Toc465983240)

[2.总体设计 3](#_Toc465983241)

[2.1开发环境 3](#_Toc465983242)

[2.2模块结构 4](#_Toc465983243)

[2.3系统设计模式 4](#_Toc465983244)

[2.4软件架构 5](#_Toc465983245)

[3导航设计 5](#_Toc465983246)

[3.1整体导航设计 5](#_Toc465983247)

[3.2 活动导航设计 5](#_Toc465983248)

[3.3 社区导航设计 5](#_Toc465983249)

[3.4 运动导航设计 6](#_Toc465983250)

[3.5 个人中心导航设计 6](#_Toc465983251)

[4.详细设计 7](#_Toc465983252)

[4.1登录/注册 7](#_Toc465983253)

[4.2 活动管理 7](#_Toc465983254)

[4.3 动态管理 8](#_Toc465983255)

[4.4 小组管理 10](#_Toc465983256)

[4.5 个人管理 12](#_Toc465983257)

[4.6 好友管理 12](#_Toc465983258)

[4.7 运动管理 13](#_Toc465983259)

[5.交互设计 14](#_Toc465983260)

[6.数据设计 16](#_Toc465983261)

# 1引言

## 1.1 编制目的

本文档是SportsZone运动社交系统的详细设计文档，用于指导后续的软件开发，实现与测试及用户的沟通。

本文档说明的内容可能在项目实施过程中发生变更，但是必须由项目执行者仔细分析最终决定，建立持续有效的版本控制。

## 1.2 词汇表

无

## 1.3参考资料

1）IEEE标准

2）SportsZone运动社交系统需求文档

# 2.总体设计

## 2.1开发环境

本系统采用php作为主要开发语言，服务端主要使用php+sqlite+Apache，客户端使用html+css+js。用Apache作为服务器，采用sqlite作为后台数据管理系统。

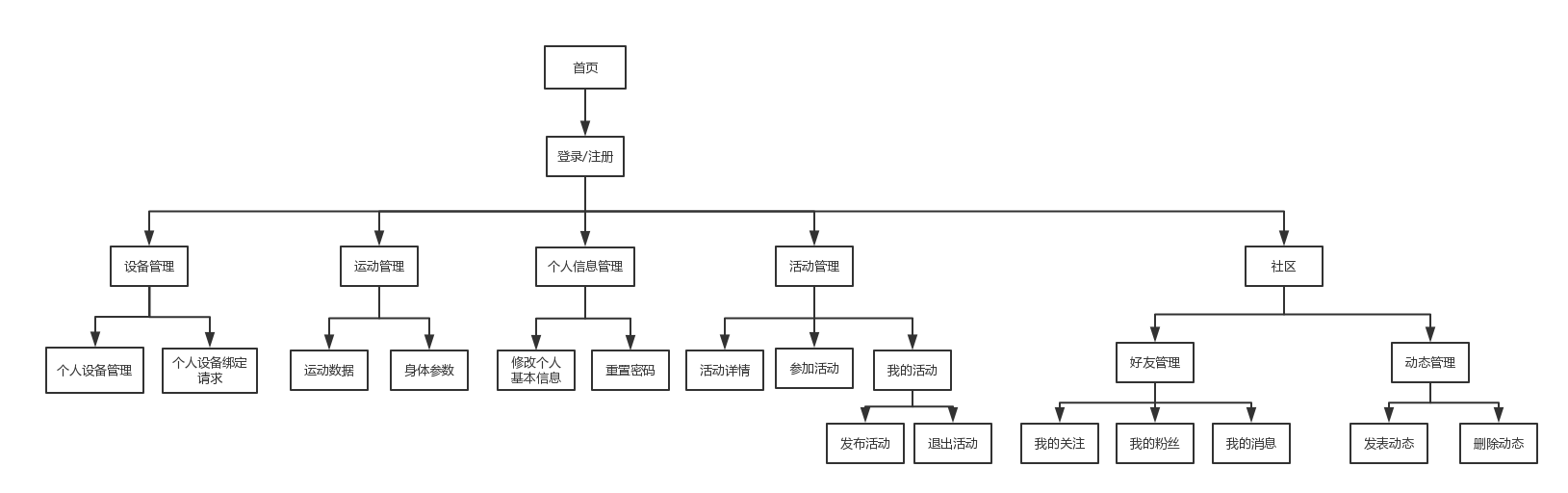
开发环境：windows 10

开发工具：phpstorm

测试浏览器：chorme

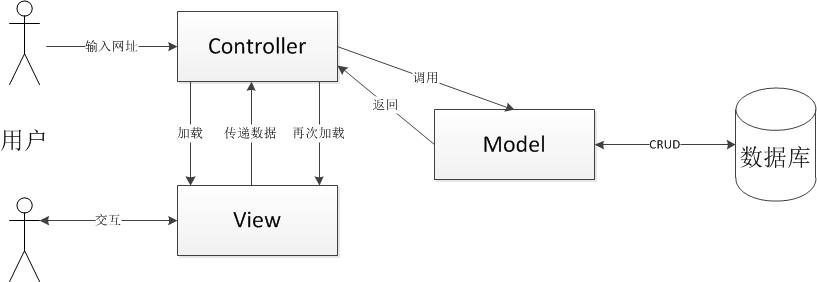
服务器：Apache2.4

## 2.2模块结构

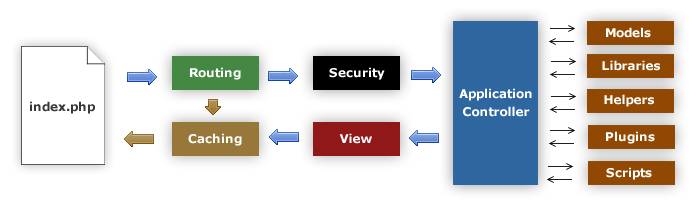


## 2.3系统设计模式

本系统采用CodeIgnitor框架进行开发，基于MVC设计模式



CI基本框架：



1. index.php 文件作为前端控制器，初始化运行CodeIgniter 所需的基本资源；
2. Router 检查 HTTP 请求，以确定如何处理该请求；
3. 如果存在缓存文件，将直接输出到浏览器，不用走下面正常的系统流程；
4. 在加载应用程序控制器之前，对 HTTP 请求以及任何用户提交的数据进行安全检查；
5. 控制器加载模型、核心类库、辅助函数以及其他所有处理请求所需的资源；
6. 最后一步，渲染视图并发送至浏览器，如果开启了缓存，视图被会先缓存起来用于后续的请求。

## 2.4软件架构

Restful架构

# 3导航设计

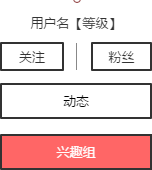
## 3.1整体导航设计



## 3.2 活动导航设计



## 3.3 社区导航设计



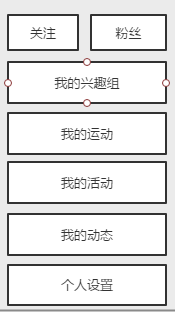
#### 3.3.1 兴趣组导航



## 3.4 运动导航设计



## 3.5 个人中心导航设计



#### 3.5.1 个人设置导航设计



# 4.详细设计

## 4.1登录/注册

#### 4.1.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 登录请求 | url | /Login |
| 方法类型 | Post |
| 控制器 | AuthController |
| 方法 | Login(Request$request) |
| 参数 | 用户名，密码 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回登录信息，失败返回失败提示信息 |
| 注册请求 | url | /SignUp |
| 方法类型 | Post |
| 控制器 | AuthController |
| 方法 | Register(Request$request) |
| 参数 | 邮箱，用户名，密码 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回注册结果 |

#### 4.1.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::create | 创建用户并加入数据库 |
| User::find | 查找用户 |
| User::save | 保存用户信息并更新数据库 |

## 4.2 活动管理

#### 4.2.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 活动界面 | url | /Activity |
| 方法类型 | Get |
| 控制器 | ActivityController |
| 方法 | getActivityList() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动界面 |
| 活动详情 | url | /Activity |
| 方法类型 | Get |
| 控制器 | ActivityController |
| 方法 | getActivity(Request$request) |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动详情 |
| 发布活动 | url | /Activity |
| 方法类型 | Post |
| 控制器 | ActivityController |
| 方法 | createActivity(Request$request) |
| 参数 | 活动信息（包括名称、简介、时间、类型） |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回发布结果 |
| 参与活动 | url | /Activity |
| 方法类型 | Post |
| 控制器 | ActivityController |
| 方法 | ParticipateIn(Request$request) |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回参与结果 |

#### 4.2.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Activity::getActivityList | 获取活动列表 |
| Activity::getActivity | 获取对应活动的活动详情 |
| Activity::createActivity | 新建活动并将该活动信息更新到数据库中 |
| Activity::participateIn | 参与活动，更新活动参与者列表与用户个人活动列表 |

## 4.3 动态管理

#### 4.3.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 动态界面 | url | /trend |
| 方法类型 | Get |
| 控制器 | CommunityController |
| 方法 | getTrendsList() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回该用户已关注用户和互相关注的用户的动态 |
| 点赞动态 | url | /Trend |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | favor (Request$request) |
| 参数 | 动态id |
| 前置条件 | 用户已登录且给某条动态点赞 |
| 后置条件 | 系统更新动态信息 |
| 发表动态 | url | /Trend |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | release(Request$request) |
| 参数 | 动态内容 |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新动态信息 |
| 删除动态 | url | /Trend |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | delete(Request$request) |
| 参数 | 动态id |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统删除对应动态信息 |

#### 4.3.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Trend::getTrendsList | 获取动态列表 |
| Trend::updateFavorList | 更新对应动态的点赞列表 |
| Trend::createTrend | 添加发布的动态信息到数据库 |
| Trend::deleteTrend | 删除数据库中对应动态信息 |

## 4.4 小组管理

#### 4.4.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 小组界面 | url | /group |
| 方法类型 | Get |
| 控制器 | CommunityController |
| 方法 | getAllGroupList() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回所有的小组列表 |
| 我的小组 | url | /group |
| 方法类型 | Get |
| 控制器 | CommunityController |
| 方法 | getMyGroupList () |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统返回用户加入的小组列表 |
| 创建小组 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | createGroup(Request&request) |
| 参数 | 小组名称、小组描述 |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新小组信息 |
| 删除小组 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | deleteGroup(Request$request) |
| 参数 | 小组id |
| 前置条件 | 用户已登录且该小组的创建者为此用户 |
| 后置条件 | 系统删除对应小组信息 |
| 发布话题 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | createTopic(Request$request) |
| 参数 | 话题名称、话题内容、小组id |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新对应小组下的话题内容 |
| 删除话题 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | deleteTopic(Request$request) |
| 参数 | 话题id、小组id |
| 前置条件 | 用户已登录且该话题的创建者为该用户 |
| 后置条件 | 系统删除对应小组下的话题 |
| 加入小组 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | ParticipateInGroup(Request$request) |
| 参数 | 小组id |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新小组成员列表与用户小组列表 |
| 退出小组 | url | /Group |
| 方法类型 | Post |
| 控制器 | CommunityController |
| 方法 | ExitGroup(Request$request) |
| 参数 | 小组id |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新小组成员列表与用户小组列表 |

#### 4.4.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Group::getGroupList | 获取对应情况下的小组列表 |
| Group::createGroup | 添加对应的小组到数据库 |
| Group::deleteGroup | 删除对应小组 |
| Group::participateIn | 更新对应小组与用户的信息 |
| Group::exitGroup | 更新对应小组与用户信息 |
| Group::createTopic | 更新对应小组的话题列表 |
| Group::deleteTopic | 更新对应小组的话题列表 |

## 4.5 个人管理

#### 4.5.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 修改基本信息 | url | /SelfInfo |
| 方法类型 | Post |
| 控制器 | UserController |
| 方法 | infoUpdate(Request$request) |
| 参数 | 修改的个人信息 |
| 前置条件 | 用户已登录 |
| 后置条件 | 持久化更新个人信息 |
| 重置密码请求 | url | /selfInfo |
| 方法类型 | Post |
| 控制器 | UserController |
| 方法 | PasswordUpdate(Request$request) |
| 参数 | 修改密码 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回修改结果 |

#### 4.5.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::updateUserInfo | 更新用户基本信息 |
| User::updatePassword | 更新用户密码 |

## 4.6 好友管理

#### 4.6.1 提供的接口设计

|  |  |  |
| --- | --- | --- |
| 关注 | url | /Friends |
| 方法类型 | Post |
| 控制器 | FriendController |
| 方法 | Follow(Request$request) |
| 参数 | 用户id |
| 前置条件 | 用户已登录 |
| 后置条件 | 更新对应用户的关注与粉丝列表 |
| 取消关注 | url | /Friends |
| 方法类型 | Post |
| 控制器 | FriendController |
| 方法 | Unfollow(Request$request) |
| 参数 | 用户id |
| 前置条件 | 用户已登录 |
| 后置条件 | 更新对应用户的关注与粉丝列表 |
| 查看好友列表 | url | /Friends |
| 方法类型 | Get |
| 控制器 | FriendController |
| 方法 | getFriends() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统返回与用户互相关注的用户列表 |

#### 4.6.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::addFollowers | 更新对应用户的粉丝列表 |
| User::addFollowing | 更新对应用户的关注列表 |
| User::removeFollowers | 更新对应用户的粉丝列表 |
| User::removeFollowing | 更新对应用户的关注列表 |
| User::getFollowers | 获取对应用户的粉丝列表 |
| User::getFollowing | 获取对应用户的关注列表 |

## 4.7 运动管理

#### 4.7.1 提供的接口设计

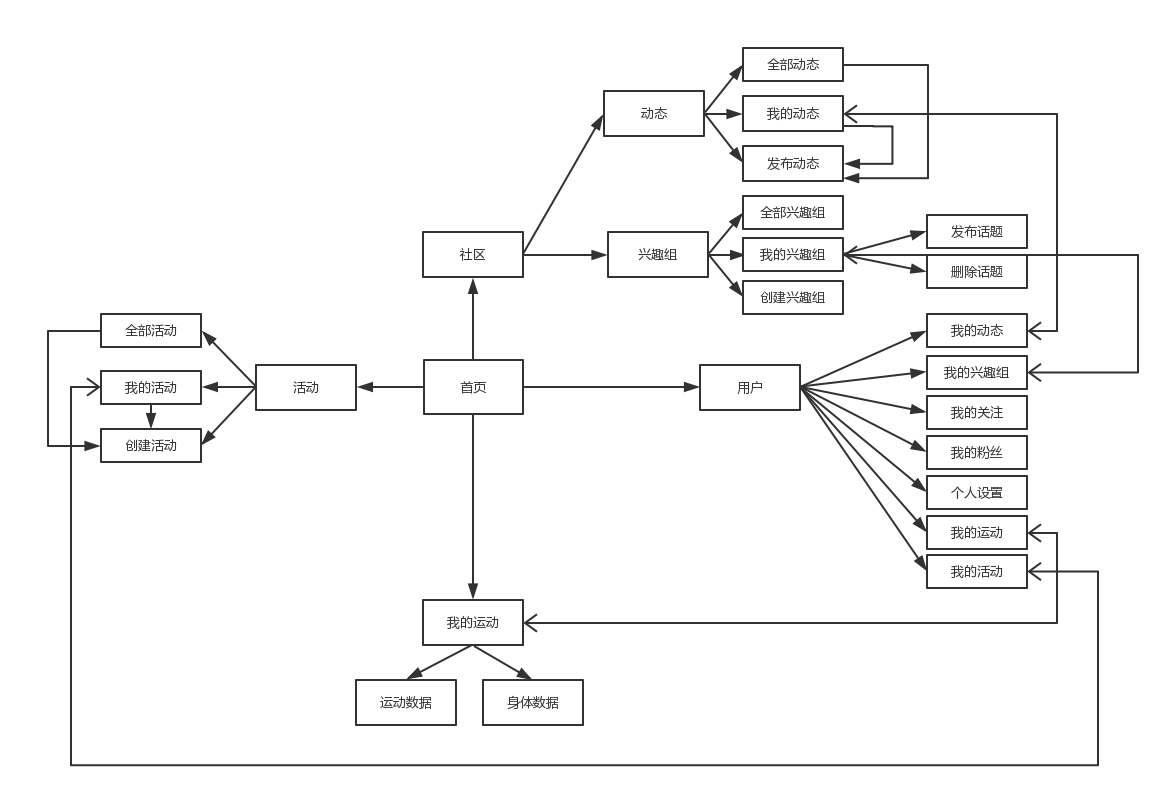
|  |  |  |
| --- | --- | --- |
| 运动数据 | url | /Statistics |
| 方法类型 | Get |
| 控制器 | StatisticController |
| 方法 | getStatistic(Request$request) |
| 参数 | 查询类型（日/周/总） |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回用户的运动信息 |
| 排名 | url | /Statistics |
| 方法类型 | Get |
| 控制器 | StatisticController |
| 方法 | getRanked(Request$request) |
| 参数 | 排名时间段（日/周/总） |
| 前置条件 | 用户已登录且有上传数据 |
| 后置条件 | 返回对应时间段的前十名用户id与运动距离 |
| 身体参数设置 | url | /Statistics |
| 方法类型 | Post |
| 控制器 | StatisticController |
| 方法 | bodyUpdate(Request$request) |
| 参数 | 身体信息 |
| 前置条件 | 用户已登录 |
| 后置条件 | 更改对应用户的身体参数 |

#### 4.7.2 需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Statistic::distance | 返回用户运动距离数据 |
| Statistic::kalorie | 返回用户消耗卡路里数据 |
| Statistic::days | 返回用户运动天数数据 |
| Statistic::rank | 返回所有用户列表中排名前十的用户id |
| Statistic::bodyUpdate | 更新用户身体数据设置 |

# 5.交互设计

用户界面跳转图



注：其中双箭头表示两个页面是一样的，单箭头表示跳转

# 6.数据设计

