Sportmate系统设计文档

余旻晨

141250177

| 修改人员 | 修改目的 | 更新日期 | 版本号 |
| --- | --- | --- | --- |
| 余旻晨 | 完成文档初稿 | 2016/11/1 | V1.0 |
| 余旻晨 | 完成数据定义 | 2016/11/2 | V1.1 |
| 余旻晨 | 完成数据关系建模 | 2016/11/4 | V1.2 |

1.引言 4

1.1编制目的 4

1.2词汇表 4

1.3参考资料 4

2.总体设计 4

2.1开发环境 4

2.2软件架构 4

2.3设计模式 4

3.导航设计 4

3.1整体导航设计 5

3.2个人活动导航 5

3.3个人信息导航 6

3.4好友导航 6

3.5活动列表筛选导航 6

4.详细设计 7

4.1登录/注册 7

4.1.1提供的接口设计 7

4.1.2需要的接口 7

4.2设备管理 8

4.2.1提供的接口设计 8

4.2.2需要的接口 8

4.3活动管理 8

4.3.1提供的接口设计 8

4.3.2需要的接口 9

4.4个人信息管理 9

4.4.1提供的接口设计 9

4.4.2需要的接口 9

4.5用户管理 10

4.5.1提供的接口设计 10

4.5.2需要的接口 10

4.6运动管理 10

4.6.1提供的接口设计 10

4.6.2需要的接口 11

4.7社交平台管理 11

4.7.1提供的接口设计 11

4.7.2需要的接口 12

5.数据定义 12

5.1数据种类 12

5.2数据描述 12

5.2.1用户信息数据 12

5.2.2活动信息数据 13

5.2.3好友信息数据 13

5.2.4朋友圈数据 13

5.2.5运动数据 14

5.2.6建议数据 14

5.2数据关系模型 14

1.引言

1.1编制目的

本文档是Sportmate运动社交系统详细设计文档，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）Sportmate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：OS X 10.11.6

开发工具：phpstorm、MAMP

测试浏览器：Safari，Chrome

服务器：Apache2

2.2软件架构

采用Restful架构

2.3设计模式

系统基于MVC设计模式，各个部分的功能如下。

入口脚本：终端用户能直接访问的PHP脚本，负责启动一个请求处理周期。

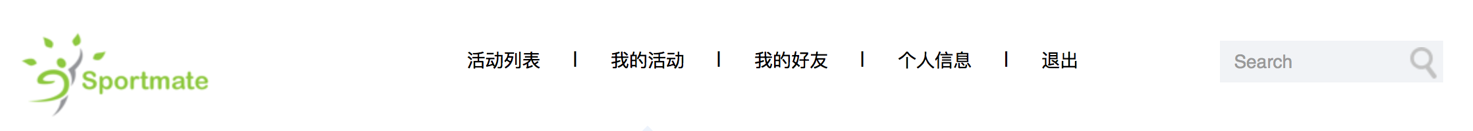
应用：能全局范围内访问的对象，管理协调组件来完成请求.

应用组件：在应用中注册的对象，提供不同的功能来完成请求。

模块：包含完整MVC结构的独立包，一个应用可以由多个模块组建。

3.导航设计

3.1整体导航设计



网站的主导航包括：活动列表、我的活动、我的好友、以及个人信息。



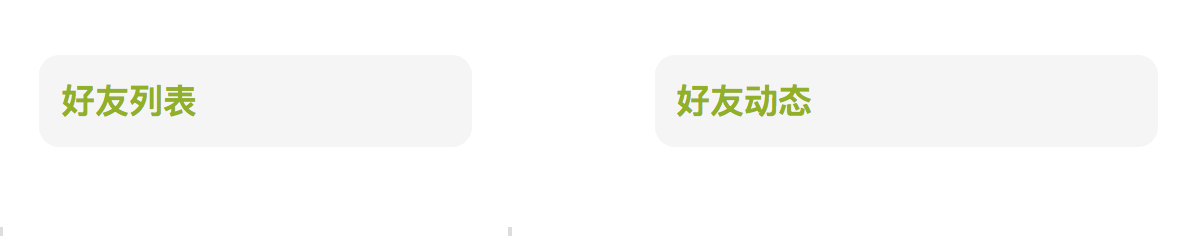
3.2个人活动导航



3.3个人信息导航



3.4好友导航



3.5活动列表筛选导航



4.详细设计

4.1登录/注册

4.1.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 登录界面 | url | /Login |
| 方法类型 | Get |
| 控制器 | LoginController |
| 方法 | getLogin() |
| 参数 | 无 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回登录界面 |
| 登录请求 | url | /Login |
| 方法类型 | Post |
| 控制器 | LoginController |
| 方法 | postLogin(Request$request) |
| 参数 | 用户名，密码（密文） |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回登录信息，失败返回失败提示信息 |
| 注册界面 | url | /SignUp |
| 方法类型 | Get |
| 控制器 | LoginController |
| 方法 | getRegister() |
| 参数 | 无 |
| 前置条件 | 用户未登录 |
| 后置条件 | 返回注册界面 |
| 注册请求 | url | /SignUp |
| 方法类型 | Post |
| 控制器 | LoginController |
| 方法 | postRegister(Request$request) |
| 参数 | 邮箱，用户名，密码（密文） |
| 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设备管理

4.2.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 个人设备管理 | url | /Device |
| 方法类型 | Get |
| 控制器 | UserController |
| 方法 | getDevice() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回设备管理界面 |
| 个人绑定设备请求 | url | /device |
| 方法类型 | Post |
| 控制器 | UserController |
| 方法 | postDeviceBind(Request$request) |
| 参数 | 设备编号 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回绑定结果 |

4.2.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::find | 查找用户 |
| User::save | 保存用户信息并更新数据库 |

4.3活动管理

4.3.1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() |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动详情 |
| 参与活动 | url | /Activity |
| 方法类型 | Post |
| 控制器 | GameController |
| 方法 | participateIn(Request$request) |
| 参数 | 活动id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回活动界面 |
| 查看我的活动 | url | /Activity |
| 方法类型 | Get |
| 控制器 | ActivityController |
| 方法 | getMyActivity() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回我的活动 |

4.3.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| Activity::getActivityList | 获得活动列表 |
| Activity::getActivity | 获得活动详情 |
| Activity::participateInActivity | 参与活动 |
| User::getMyActivity | 查看我的活动 |

4.4个人信息管理

4.4.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 个人信息界面 | url | /SelfInfo |
| 方法类型 | Get |
| 控制器 | UserController |
| 方法 | getUserInfo() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回个人信息界面 |
| 修改个人信息请求 | url | /SelfInfo |
| 方法类型 | Post |
| 控制器 | UserController |
| 方法 | postReset(Request$request) |
| 参数 | 昵称，个性域名，性别，生日，所在地，兴趣，运动宣言 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回修改结果 |

4.4.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::find | 查找用户 |
| User::save | 保存用户信息并更新数据库 |

4.5用户管理

4.5.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 用户权限管理界面 | url | /UserAdmin |
| 方法类型 | Get |
| 控制器 | UserAdminController |
| 方法 | UserAdminIndex() |
| 参数 | 无 |
| 前置条件 | 管理员已登录 |
| 后置条件 | 返回用户权限管理界面 |
| 权限修改请求 | url | /UserAdmin |
| 方法类型 | Post |
| 控制器 | UserAdminController |
| 方法 | modifyUserAdmin(Request$request) |
| 参数 | 用户信息，变更权限 |
| 前置条件 | 管理员已登录 |
| 后置条件 | 保存用户权限 |

4.5.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::find | 查找用户 |
| Admin::save | 保存权限修改信息 |

4.6运动管理

4.6.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 运动管理界面 | url | /SportsData |
| 方法类型 | Get |
| 控制器 | SportsController |
| 方法 | sportsData() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回运动管理界面 |
| 设置数据类型 | url | / SportsData |
| 方法类型 | Get |
| 控制器 | SportsController |
| 方法 | setDataType() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 运动数据管理界面更新 |
| 查询运动建议 | url | / SportsAdvice |
| 方法类型 | Get |
| 控制器 | SportsController |
| 方法 | getSportsAdvice () |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 显示运动建议界面 |

4.6.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::getSportsData | 获得运动信息 |
| User::getSportsAdvice | 获得运动建议 |

4.7社交平台管理

4.7.1提供的接口设计

|  |  |  |
| --- | --- | --- |
| 朋友圈界面 | url | /SelfInfo/friends |
| 方法类型 | Get |
| 控制器 | FriendsController |
| 方法 | getFriends() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回好友列表 |
| 查找好友 | url | /SelfInfo/friends |
| 方法类型 | Post |
| 控制器 | FriendsController |
| 方法 | findFriend() |
| 参数 | 好友id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回查找结果 |
| 添加好友 | url | /SelfInfo/friends |
| 方法类型 | Post |
| 控制器 | FriendsController |
| 方法 | makeFriend() |
| 参数 | 用户id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回添加结果 |
| 删除好友 | url | /SelfInfo/friends |
| 方法类型 | Delete |
| 控制器 | FriendsController |
| 方法 | deleteFriend(Request$request) |
| 参数 | 好友id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回查找好友界面 |
| 发布朋友圈消息 | url | /SelfInfo/friends |
| 方法类型 | Post |
| 控制器 | FriendsController |
| 方法 | postNews(Request$request) |
| 参数 | 消息内容 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回发布情况 |
| 查看朋友圈消息 | url | /SelfInfo/friends |
| 方法类型 | Get |
| 控制器 | FriendsController |
| 方法 | getNews() |
| 参数 | 无 |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回朋友圈消息 |
| 删除朋友圈消息 | url | /SelfInfo/friends |
| 方法类型 | Delete |
| 控制器 | FriendsController |
| 方法 | deleteNews(Request$request) |
| 参数 | 消息id |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回删除情况 |

4.7.2需要的接口

|  |  |
| --- | --- |
| 服务名 | 服务 |
| User::getFriends | 获得好友列表 |
| User::findFriend | 查找好友 |
| User::makeFriend | 添加好友 |
| User::deleteFriend | 删除好友 |
| User::getNews | 查找朋友圈消息 |
| User::deleteNews | 删除朋友圈消息 |

5.数据定义

5.1数据种类

本系统涉及的数据包括：用户信息数据、活动信息数据、好友信息数据、朋友圈数据、运动数据、建议数据。在数据描述中对每一类数据进行详细定义。

5.2数据描述

5.2.1用户信息数据

用户信息数据描述：

用户ID：整数类型

用户名：字符串类型

登录密码：字符串类型

生日：日期类型

性别：枚举类型（男/女）

身高：数字类型

体重：数字类型

所在地：确定的字符串类型

兴趣：字符串类型

头像：图片类型

5.2.2活动信息数据

活动信息数据描述：

活动ID：整数类型

发起者ID：整数类型

内容：字符串类型

创建时间：日期类型

开始时间：日期类型

结束时间：日期类型

人数限制：整数类型

活动参与数据描述：

活动ID：整数类型

参与者ID：整数类型

5.2.3好友信息数据

好友信息数据描述：

用户ID：整数类型

好友ID：整数类型

5.2.4朋友圈数据

朋友圈数据描述：

用户ID：整数类型

朋友圈ID：整数类型

内容：字符串类型

附图：图片类型

创建时间：日期类型

朋友圈评论数据描述：

朋友圈ID：整数类型

评论者ID：整数类型

内容：字符串类型

创建时间：日期类型

5.2.5运动数据

运动数据描述：

ID：整数类型

用户ID：整数类型

运动开始时间：日期类型

运动结束时间：日期类型

运动种类：确定的字符串类型

运动量：数字类型

5.2.6建议数据

建议数据描述：

用户ID：整数类型

建议ID：整数类型

建议内容：字符串类型

创建时间：日期类型

5.2数据关系模型

对各类数据进行建表，关系模型如下：

