

**毕 业 论 文（设 计）**

题目：房地产运营支撑系统

(英文)：Real estate operation support system

院别：电子与信息学院

专业：电子信息工程

姓名：梁锦银

学号：2014044243013

指导教师：王春安

日期：2018年5月

# 房地产运营支撑系统

**摘要**

如今公司如雨后春笋搬地涌现，每一家公司有具有着自己的独特，独特的运营模式，还有各种机构等等。公司内部的人员，特别是高层，他们需要一个系统来管理，和查看公司的运营情况。本文将结合现在的房地产经营情况开发一套房地产运营支撑系统。系统运用的Java语言编写，采用SSM（SpringMVC+spring+mybatis）三层架构框架进行开发,前端使用JQuery+CSS+JSP+bootstrap技术对视图渲染，后台使用mysql对数据进行处理和存储，运用shiro进行功能权限的控制。系统实现了客户管理，案场管理，会议纪要，系统基本模块；可以清晰快捷的查询案场销售情况和销售人的具体销售。系统简单易懂，运行稳定，适用于中小型房地产公司。□□

**关键词：**springMVC；Spring；mybatis；shiro；MySQL等技术

Real estate operation support system

Abstract

Now companies are springing up, and each company has its own unique, unique operating model, a variety of institutions and so on. The people inside the company, especially at the top, need a system to manage it. This paper will develop a set of real estate operation support system combined with the current real estate management situation. The system will be written in Java language and developed with a three-tier framework of SSM(SpringMVC spring mybatis. The front end uses JQuery CSS JSP bootstrap technology to render the view, the background uses mysql to process and store the data, uses shiro to carry on the function authority control. The system has realized the customer management, the case field management, the meeting minutes, the system basic module; Can be clear and quick inquiry case field sales and specific sales of the salesperson. The system is easy to understand, stable operation, suitable for small and medium-sized real estate companies. --

Key words: spring MVC, Spring MVC, Spring MVC, Spring MVC, Spring MVC, Spring MVC, Spring MVC, Spring

# 1绪论

## 1.1研究背景与意义

## 1.2研究现状

# 2系统框架技术

本系统主要是由SSM（springMVC+spring+mybatis）+shiro框架构建整合框架，并整合前端框架bootstrap实现视图渲染。

## 2.1 springMVC框架



图2.1SpringMVC架构图

SpringMVC其中的MVC （Model-View-Control）模型-视图-控制；其性质的C 层要完成的主要工作：封装web 请求为一个数据对象、调用业务逻辑层来处理数据对象、返回处理数据结果及相应的视图给用户。Spring C 层框架的核心是 DispatcherServlet，它的作用是将请求分发给不同的后端处理器，也即使用了一种被称为Front Controller 的模式（后面对此模式有简要说明）。 Spring 的C 层框架使用了后端控制器来、映射处理器和视图解析器来共同完成C 层框架的主要工作。并且spring 的C 层框架还真正地把业务层处理的数据结果和相应的视图拼成一个对象，即ModelAndView 对象。

## 2.2 spring框架

## Spring框架一个开源的轻量级框架，它不仅可以解决许多J2EE开发中一些常见的问题，同时也可替代EJB技术，Spring的产生很大部分是企业开发应用的复杂性促使的。Spring提供了组件核心底层机制的一些运作情况，例如组件的创立、获得、结束、资源管理、状态管理等生命周期的相关支持功能，是一个组件技术框架。Spring作为一种面向切面（AOP）的容器以及轻量级的控制反转（IOC）框架，可以很方便快捷地实现简单的组合或者一些复杂的应用以及组件的配置等，对于开发人员来说，这可以帮助他们编写出方便管理并且易于测试的代码。

## 2.3 mybatis框架

MyBatis 本是[apache](http://baike.baidu.com/view/28283.htm)的一个开源项目[iBatis](http://baike.baidu.com/view/628102.htm), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis，实质上Mybatis对ibatis进行一些改进。 目前mybatis在github上托管。git（分布式版本控制，当前比较流程）MyBatis是一个优秀的持久层框架，它对jdbc的操作数据库的过程进行封装，使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、创建connection、创建statement、手动设置参数、结果集检索等jdbc繁杂的过程代码。Mybatis通过xml或注解的方式将要执行的各种statement（statement、preparedStatemnt、CallableStatement）配置起来，并通过java对象和statement中的sql进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射成java对象并返回。

## 2.4 shiro框架

apache shiro 是一个安全认证框架，和 spring security 相比，在于他使用了比较简洁易懂的认证和授权方式。其提供的 native-session（即把用户认证后的授权信息保存在其自身提供Session 中）机制，这样就可以和 HttpSession、EJB Session Bean 的基于容器的 Session脱耦，到和客户端应用、Flex 应用、远程方法调用等都可以使用它来配置权限认证。 在exit-web-framework 里的 vcs-admin 例子用到该框架，具体使用说明可以参考官方帮助文档。在这里主要讲解如何与 spring 结合、动态创建 filterchaindefinitions、以及认证、授权、和缓存处理。

## 2.5 SSM整合

# 3系统需求分析与设计

## 3.1 系统需求

系统实现应该具备以下功能：

一、客户管理：客户管理包含客户信息、合作伙伴。

1、 客户信息，客户信息模块应关联楼盘的目标户型，楼盘的改变会改 变客户的目标户型，还要关联置业顾问；客户信息基本的增删改查 功能；客户信息的导入、导出（Excel）功能。

2、合作伙伴，应具备查看合作伙伴、添加合作伙伴、修改合作伙伴、删 除合作伙伴功能。

二、商机管理

三、案场管理：案场管理包含案场看板、案场信息、房型信息、销售看板。

1、案场信息，应具备基本增删改查功能，并关联系统内部人员。

2、案场看板，具有查看本月楼盘销售量，剩余数量，销售额。

3、房型信息，具备基本增删改查功能，并关联上案场维度。

4、销售看板，应有查看置业顾问的月销售额，签单数，完成指标率。

四、我的面板：包含个人信息、个人来往信箱、会议纪要。

1、个人信息，可以查看自己的电话、邮箱、职位、所属部门等基本信息。

2、信箱，具有同公司人员发送邮箱功能，可以发送工作汇报、请假事宜。

3、会议纪要，可以记录会议主题，会议内容，解决的问题。应具有在会 议纪要中发送邮箱到公司其他人员。

五、系统管理：应包含用户管理、机构管理、角色管理、字典管理

1、用户管理，应具有增删改查功能，记录用户的基本信息。

2、机构管理，应拥有增删改查功能，用树状图展示机构的层级关系。

3、角色管理，应具有增删改查功能，具有分配权限和分配用户功能。

六、数据权限和功能权限

1、数据权限，系统用户应具有职级，职级不同看到的数据不同。职级越 高查看的数据就越多。

2、功能权限，职级的等级越高界面的功能就越多。

## 3.2 系统整体结构设计

系统整体结构分为三部分。系统logo和名称作为界面的头部，左侧是菜单栏，右边是功能性页面展示区。如图3.1所示：

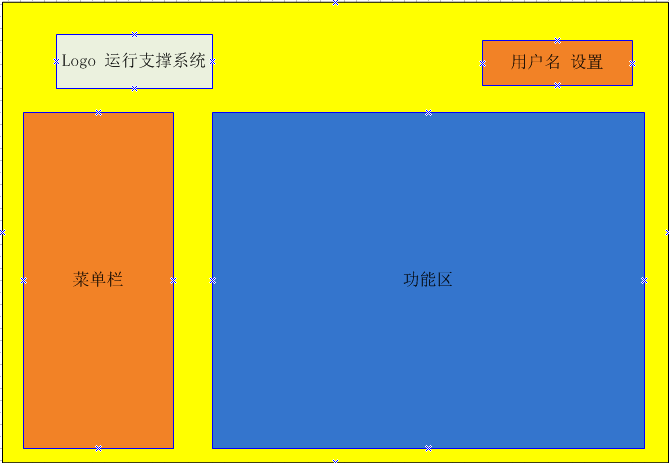


图3.1 系统整体结构设计

其中设置中包含了修改密码和退出系统；

## 3.3各功能需求分析与设计

### 3.3.1用户权限控制

用户可分为三种，普通置业顾问、案场负责人（项目负责人）、总主管（admin）。

普通的置业顾问只能操作所属自己的客户数据，案场负责人可以操作所属案场的所有数据，admin可以操作所有数据。一个用户可以具有多角色，角色与用户之间的关系是多对多。权限的控制包括数据权限和功能权限，根据角色的不同查看的数据数目不同，职级越高查看的数据就越多，职级的等级越高界面的功能就越多。权限可由admin管理员配置。

### 3.3.2客户管理模块

客户管理分为客户信息和合作伙伴两部分，其中客户信息具有增删改查功能、导入导出功能。如图3.2所示

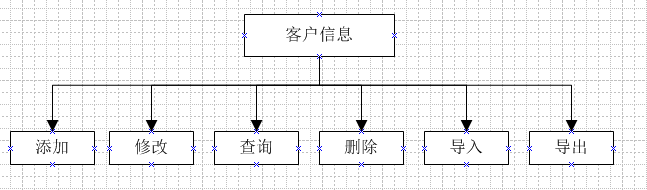


图3.2

1、添加新客户

添加客户个人信息，客户姓名、联系电话、客户状态、购房用途、认知途径、关联案场、意向户型、案场负责人。

2、修改客户

修改客户信息。客户的有三种状态：潜在客户、正在跟进客户、已交易客户。潜在客户：经过大数据的匹配的有意图卖房的客户；正在跟进客户：来案场咨询或是来电咨询客户；已交易客户：已进行金钱交易，有合同保障客户。一旦客户的状态改变为已交易客户，该客户的置业顾问的销售额会添加对应客户选中的案场房型的金额值，案场的房型数量也会减少1，本季度的案场销售额亦添加相应的金额。客户信息中包含的楼盘属性与房型和案场联系人有关联，若是楼盘的变更，其目标户型和案场负责人也会随之变化。

3、查询客户信息

具有客户名称检索、客户状态查询。

4、删除客户

删除客户信息。

5、导入导出

客户信息可通过Excel文件导入导出。

合作伙伴具有增删改查功能。如图3.3所示

只有admin管理员才能对其进行增删改功能，其他用户只能查看。

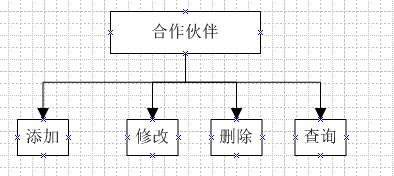


图3.3

### 3.3.3案场管理

案场管理包含案场看板、案场信息、房型信息、销售看板四个节点。

结构图如图3.4所示

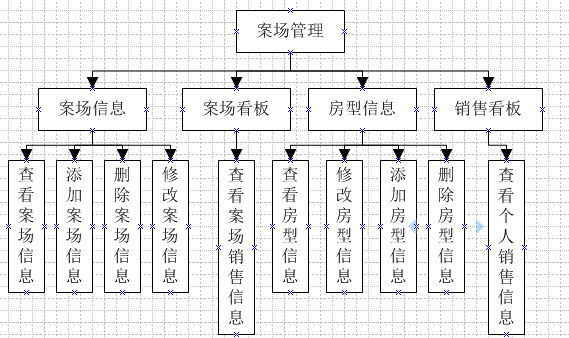


图3.4

#### 3.3.3.1案场信息

1、添加案场信息

添加案场信息，其中维度有案场名称、案场地址、所属集团、案场负责人、负责人电话、规定月销售额、开盘时间、结束时间、关联团队成员、备注信息。

其中案场负责人、团队成员关联用户，案场信息一般由案场负责人添加。

2、查询案场信息

查询案场信息有三种检索方式，案场负责人、开盘时间、所属集团的搜索。其中案场负责人和所属集团运用的是模糊查询。

3、删除案场信息

根据案场id删除案场信息。

4、修改案场信息

案场信息的修改由案场负责人修改。

#### 3.3.3.2案场看板

案场看板包含了案场名称、已售楼房数量、剩余楼房数量、本月销售额、本月出售楼房数量。看板的统计来自于客户的成交量。当客户信息中的客户状态为已交易状态，那么客户对应的目标户型关联的楼盘（案场）的已售楼房数量会加1，从而剩余楼房数量会减1，本月销售额会增加目标户型对应的价格数目，本月出售数量也是会加1。

#### 3.3.3.3房型信息

1、添加房型信息

添加房型信息，其中维度有房子数目、房总价格、房型、面积、剩余数量、所属楼盘、备注。剩余数量=房数目-已售数目。所属楼盘关联案场，房型信息一般由案场负责人添加。

2、查询房型信息

查询房型有两种索方式，房子总价格、套房面积，都是运用模糊查询。

3、删除房型信息

根据案场id删除房型信息。

4、修改房型信息

房型信息的修改由案场负责人修改，套房的数目的改变会改变案场套房总数目。

#### 3.3.3.4销售看板

销售看板包含了销售负责人（置业顾问）、楼房的销售指标、个人的签单数、本月的销售额、完成率。销售指标是由案场负责人定义的，每个案场的销售指标可能不同；个人签单数就是销售套房的数量；完成率就是本月有没有达到指标数，完成率的公式：完成率=指标数\*100%。销售额前三名的职业顾问会作为公司的榜样在首页展示。

## 3.3.4我的面板

我的面板包括三个节点，个人信息、邮箱、会议纪要。结构图如图3.5所示

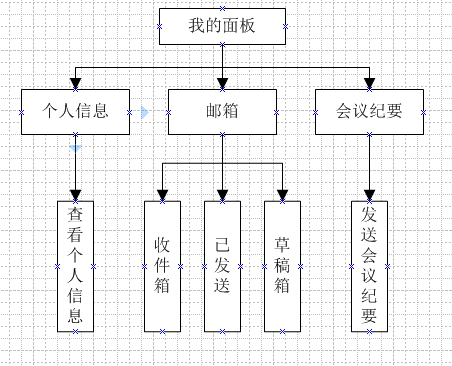


图3.5

#### 3.3.4.1个人信息

个人信息，可以查看个人的手机号、邮箱、所属部门、公司、所属角色等。

#### 3.3.4.2邮箱

1、收件箱

接收别人发送过来的邮件。邮件有增删查功能，可以对邮件的正文或是标题进行模糊查询；删除可以批量删除或是删除单条数据；发送邮件，可以发送到公司内部人员。

2、已发送

查看已发送的邮件。一是在邮箱发送的邮件，二是在会议纪要发送邮件。

3、草稿箱

一是信箱里编辑的信件草稿，二是会议纪要中保存的草稿。

#### 3.3.4.3会议纪要

记录会议的主题、出席会议的人员、会议内容、会议地点、会议时间。可以以邮件的方式发送给内部人员。

### 3.3.5系统设置

系统设置包含了用户管理、机构管理、角色管理。结构图如图3.6所示

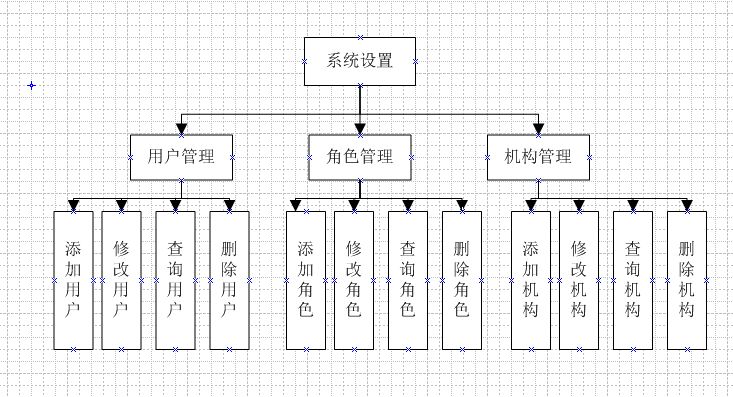


图3.6

#### 3.3.5.1用户管理

1. 添加用户

用户的维度有用户名称、登陆名、邮箱、手机号、工号、用户类型、密码、关联机构、角色。添加用户一般由管理员添加。

1. 修改用户

可以修改用户名称、角色、登陆名、邮箱、手机号、工号、用户类型、密码；一般由管理员修改。

1. 查询用户

可以根据登陆名、姓名进行模糊查询，也可以根据用户所属的机构进行检索。

1. 删除用户

支持批量删除和根据id删除。

#### 3.3.5.2角色管理

## 3.4据库表设计

### 3.4.1数据库的需求分析

根据系统的需求，数据库表也是分为四个模块。分别是客户管理、案场管理、我的面板、系统管理。其中客户管理模块包括客户信息表（fdc\_c\_customer）和合作伙伴表（fdc\_c\_partner）；案场管理有案场信息表(fdc\_b\_building)，房型信息表和销售记录表；我的面板包含信箱表、会议纪要表；系统管理包括用户信息表、机构信息表、角色信息表。

表与表之间的关系如下图所示。

客户表与房型信息表、案场信息表是一对一关系，一个客户只能买一套房；案场表与房型信息表是多对多关系，一个案场包含多种房型，一种房型也能对应多个案场；用户表与机构表是多对一的关系，一个员工能属于一个部门，一个部门有多个员工；用户表与角色表是多对多关系，一个用户有多个角色，一个角色也具有多个用户。

### 3.4.2数据库的概念结构设计

以下是数据表的逻辑结构设计，如下表3.1-3.9所示。

**表3.1 客户信息信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| name | varchar | 64 | 客户名称 |
| way | varchar | 64 | 认知途径 |
| contacts | varchar | 64 | 置业顾问 |
| phone | varchar | 64 | 电话号码 |
| target | varchar | 64 | 目标户型 |
| houses | varchar | 64 | 楼盘 |
| status | varchar | 64 | 客户状态 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |

**表3.1 合作伙伴信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| name | varchar | 64 | 合作伙伴名称 |
| signing\_time | datetime | 64 | 签约时间 |
| project | varchar | 64 | 合作项目 |
| contacts | varchar | 64 | 联系人 |
| phone | varchar | 64 | 联系人电话 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |

**表3.1 邮件信息表1**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| title | varchar | 64 | 主题 |
| overview | varchar | 128 | 内容概要 |
| content | varchar | 500 | 信件内容 |

**表3.1 邮件信息表2**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| mailid | int | 11 | 邮件外键 |
| senderId | varchar | 64 | 发件人 |
| receiverId | varchar | 64 | 收件人 |
| sendtime | datatime | 64 | 发送时间 |
| readstatus | varchar | 1 | 状态 0 未读 1 已读 |

**表3.1 会议信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| theme | varchar | 64 | 会议主题 |
| caller | varchar | 64 | 召集人 |
| address | varchar | 64 | 会议地址 |
| recorder | varchar | 64 | 记录人 |
| purpose | varchar | 64 | 会议目的 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |
| attender | varchar | 64 | 出席者 |
| absentee | varchar | 64 | 缺席者 |
| content | varchar | 500 | 会议内容 |
| beginTime | datetime | 64 | 会议开始时间 |
| mainEmail | varchar | 64 | 主送人 |
| addEmail | Varchar | 64 | 抄送人 |
| status | Varchar | 64 | 邮件状态 |
| meet\_time | varchar | 64 | 会议消耗时间 |

**表3.1 用户信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| office\_id | varchar | 64 | 所属部门 |
| login\_name | varchar | 64 | 登陆名称 |
| password | varchar | 64 | 密码 |
| no | varchar | 64 | 工号 |
| name | varchar | 64 | 名称 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |
| email | varchar | 64 | 邮箱号 |
| phone | varchar | 64 | 电话号码 |
| user\_type | varchar | 8 | 用户类型 |
| login\_ip | varchar | 128 | 最后登陆ip |
| login\_date | datetime | 64 | 登陆时间 |

**表3.1 角色信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| name | varchar | 64 | 角色名称 |
| enname | varchar | 64 | 英文名称 |
| role\_type | varchar | 64 | 角色类型 |
| data\_scope | varchar | 64 | 数据范围 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |

**表3.1 机构信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| name | varchar | 64 | 机构名称 |
| parent\_id | varchar | 64 | 机构父级编号 |
| parent\_ids | varchar | 64 | 所有父级编号 |
| sort | varchar | 64 | 排序 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |
| type | char | 1 | 机构类型 |
| grade | char | 1 | 机构等级 |

**表3.1 菜单信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段类型 | 字段宽度 | 备注 |
| id | Int | 11 | 主键 |
| name | varchar | 64 | 菜单名称 |
| parent\_id | varchar | 64 | 菜单父级编号 |
| parent\_ids | varchar | 64 | 所有父级编号 |
| sort | varchar | 64 | 排序 |
| create\_by | varchar | 64 | 创建人 |
| create\_date | datetime | 64 | 创建时间 |
| update\_by | varchar | 64 | 更新者 |
| update\_date | datetime | 64 | 更新时间 |
| del\_flag | char | 1 | 删除标识 |
| href | varchar | 128 | 连接 |
| permission | varchar | 128 | 权限 |

# 4系统功能实现

## 4.1权限控制

该系统具有数据权限和功能权限，功能权限在用户登录的时候，继承由shiro框架提供的AuthorizingRealm类进行授权和验证。不同的角色的功能权限和数据权限不同，admin（管理员）的权限是最大的，可以访问各个模块的所有权限控制的功能。如图4.1所示；而普通的角色只能访问客户模块、案场模块、我的面板，系统设置模块是不可见。如图4.2所示。



图4.1

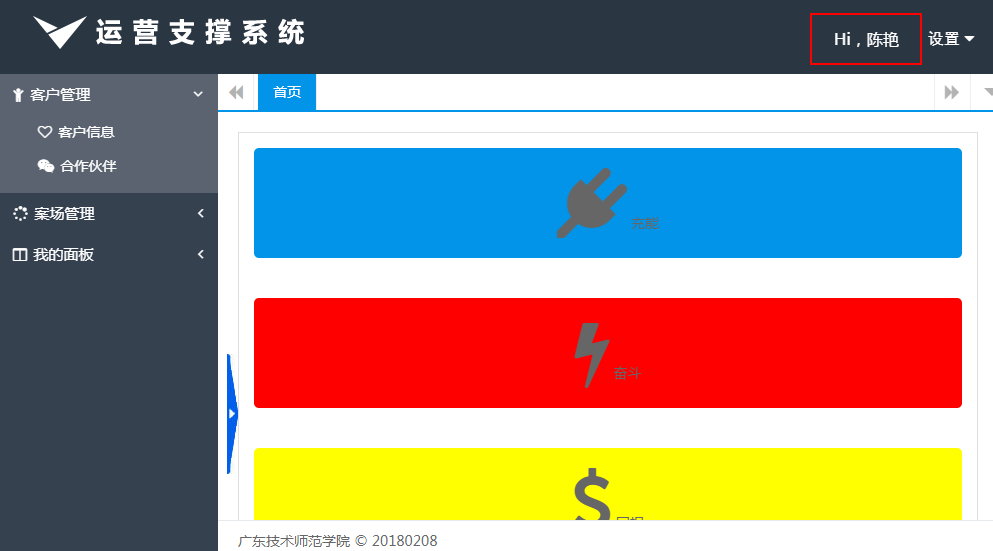


图4.2

图

主要代码实现：

前端代码：

<shiro:hasPermission name="customer:customer:add">

<table:addRow url="${ctx}/customer/customer/form" title="客户" width="540px;" height="600px;"></table:addRow><!-- 增加按钮 -->

</shiro:hasPermission>

<shiro:hasPermission name="customer:customer:del">

<table:delRow url="${ctx}/customer/customer/deleteAll" id="contentTable"></table:delRow><!-- 删除按钮 -->

</shiro:hasPermission>

<shiro:hasPermission name="customer:customer:import">

<table:importExcel url="${ctx}/customer/customer/import"></table:importExcel><!-- 导入按钮 -->

</shiro:hasPermission>

<shiro:hasPermission name="customer:customer:export">

<table:exportExcel url="${ctx}/customer/customer/export"></table:exportExcel><!-- 导出按钮 -->

</shiro:hasPermission>

后台代码：

User user = getSystemService().getUserByLoginName(principal.getLoginName());

if (user != null) {

SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();

List<Menu> list = UserUtils.getMenuList();

for (Menu menu : list){

if (StringUtils.isNotBlank(menu.getPermission())){

// 添加基于Permission的权限信息

for (String permission : StringUtils.split(menu.getPermission(),",")){

info.addStringPermission(permission);

}

}

}

// 添加用户权限

info.addStringPermission("user");

// 添加用户角色信息

for (Role role : user.getRoleList()){

info.addRole(role.getEnname());

}

## 4.2客户管理模块

### 4.2.1客户信息模块

1. 客户信息

由置业顾问或是案场管理人员添加自己接待的客户，界面显示客户名称、置业顾问、置业顾问号码、目标户型、客户状态、更新时间；操作栏可以修改和删除客户，修改案场时，目标户型会动态变化。如图4.3所示。排序是根据更新时间排序的，页面也可以根据各个字段排序。一面可以浏览10条数据，数据支持导入导出功能。如图4.4所示

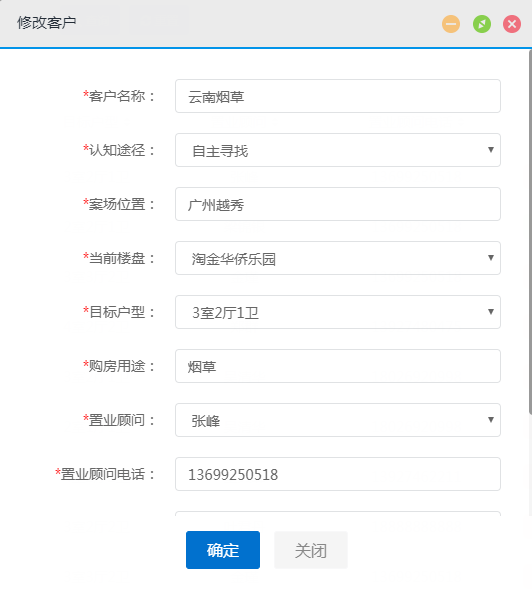


图4.3

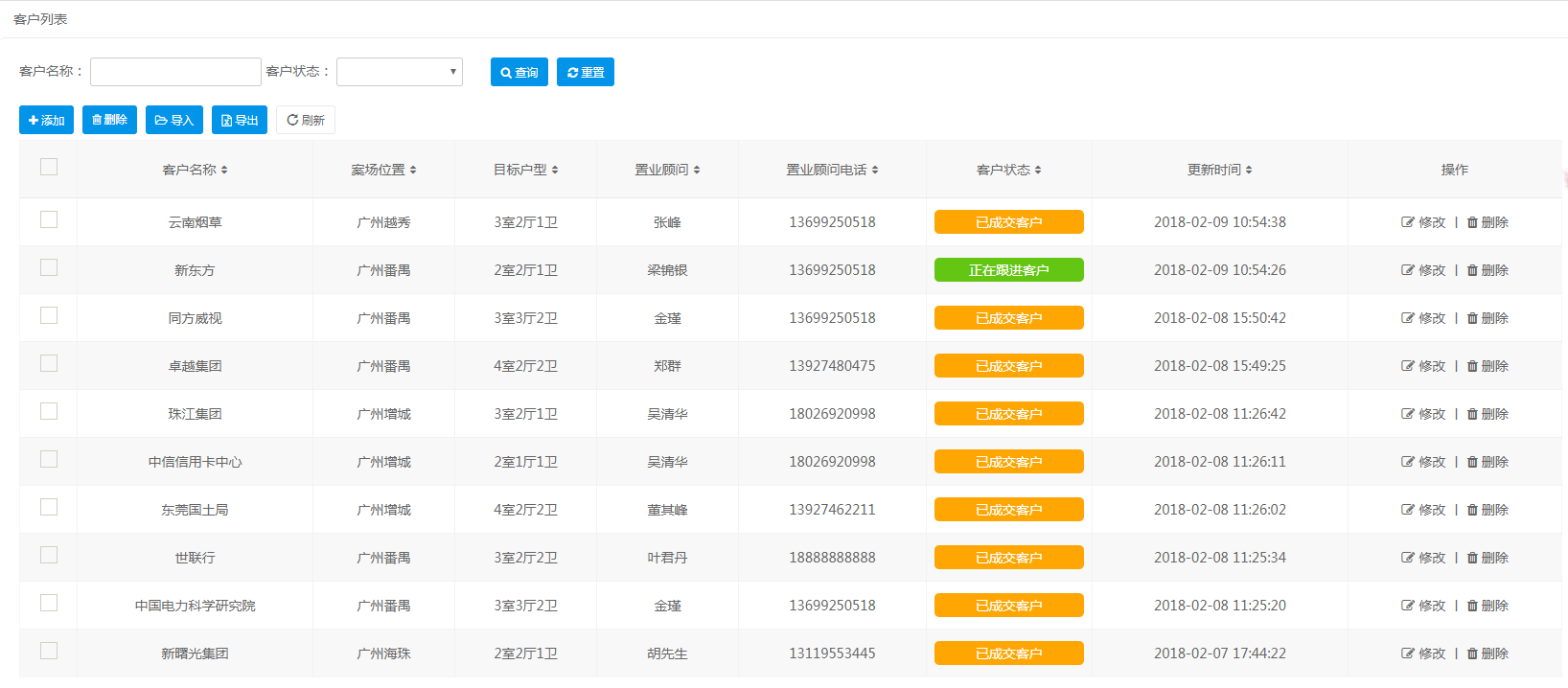


图4.4

主要实现代码：

/\*\*

\* 客户列表页面

\*/

@RequiresPermissions("customer:customer:list")

@RequestMapping(value = {"list", ""})

**public** String list(Customer customer, HttpServletRequest request, HttpServletResponse response, Model model) {

Page<Customer> page = customerService.findPage(**new** Page<Customer>(request, response), customer);

model.addAttribute("page", page);

**return** "modules/customer/customerList";

}

/\*\*

\* 查看，增加，编辑客户表单页面

\*/

@RequiresPermissions(value={"customer:customer:view","customer:customer:add","customer:customer:edit"},logical=Logical.***OR***)

@RequestMapping(value = "form")

**public** String form(Customer customer, Model model) {

model.addAttribute("customer", customer);

List<Cases> casesList = casesService.getAllCasesList();

List<Houses> housesList = housesService.getListByCasesName(customer.getHouses());

model.addAttribute("casesList", casesList);

model.addAttribute("housesList", housesList);

**return** "modules/customer/customerForm";

}

/\*\*

\* 保存客户

\*/

@RequiresPermissions(value={"customer:customer:add","customer:customer:edit"},logical=Logical.***OR***)

@RequestMapping(value = "save")

**public** String save(Customer customer, Model model,HttpServletRequest request,HttpServletResponse response) **throws** Exception{

**if** (!beanValidator(model, customer)){

**return** form(customer, model);

}

**if**(!customer.getIsNewRecord()){//编辑表单保存

Customer t = customerService.get(customer.getId());//从数据库取出记录的值

MyBeanUtils.*copyBeanNotNull2Bean*(customer, t);//将编辑表单中的非NULL值覆盖数据库记录中的值

customerService.save(t);//保存

}**else**{//新增表单保存

customerService.save(customer);//保存

}

addMessage(model, "保存客户成功");

// return list(customer, request, response, model);

**return** "redirect:"+Global.*getAdminPath*()+"/customer/customer/?repage";

}

1. 合作伙伴

合作伙伴，是公司项目的合作人。信息有置业顾问填写，界面列表显示合作伙伴名称、合作公司简介、签约时间、合作项目、联系人、联系人电话，有合作伙伴名称、公司简介、签约时间进行检索。如图4.5所示。



图4.5

主要代码实现：

/\*\*

\* 合作伙伴列表页面

\*/

@RequiresPermissions("partner:partner:list")//权限控制

@RequestMapping(value = {"list", ""})//访问路径

public String list(Partner partner, HttpServletRequest request, HttpServletResponse response, Model model) {

Page<Partner> page = partnerService.findPage(new Page<Partner>(request, response), partner);

model.addAttribute("page", page);//进行视图渲染

return "modules/partner/partnerList";//放回到页面

}

<!-- 分页代码 -->

<table:page page="${page}"></table:page>

## 4.4我的面板

### 4.4.1邮箱

邮箱是提供系统内部人员使用的，可以群发邮件。可以用来提交周工作任务报告。随时随地依靠一封E-mail就能立即让相关人员及时收到文件，大大提高工作效率。将员工E-mail作为电子工作证进行管理，一旦关闭邮箱相应人员就不再有资格参与公司的工作和代表公司与客户接洽。确保了企业正常、安全的运作。整体功能界面如图4.6所示。



图4.6

主要代码实现：

@RequestMapping(value = {"list", ""})//访问路径

public String list(MailBox mailBox, HttpServletRequest request, HttpServletResponse response, Model model) {

mailBox.setReceiver(UserUtils.getUser());

Page<MailBox> page = mailBoxService.findPage(new MailPage<MailBox>(request, response), mailBox);

model.addAttribute("page", page);

//查询未读的条数

MailBox serachBox = new MailBox();

serachBox.setReadstatus("0");

serachBox.setReceiver(UserUtils.getUser());

model.addAttribute("noReadCount", mailBoxService.getCount(serachBox));

//查询总条数

MailBox serachBox2 = new MailBox();

serachBox2.setReceiver(UserUtils.getUser());

model.addAttribute("mailBoxCount", mailBoxService.getCount(serachBox2));

//查询已发送条数

MailCompose serachBox3 = new MailCompose();

serachBox3.setSender(UserUtils.getUser());

serachBox3.setStatus("1");//已发送

model.addAttribute("mailComposeCount", mailComposeService.getCount(serachBox3));

//查询草稿箱条数

MailCompose serachBox4 = new MailCompose();

serachBox4.setSender(UserUtils.getUser());

serachBox4.setStatus("0");//草稿

model.addAttribute("mailDraftCount", mailComposeService.getCount(serachBox4));

return "modules/iim/mailBoxList";

}

### 4.4.2会议纪要

记录会议的主题、出席会议的人员、会议内容、会议地点、会议时间。可以以邮件的方式发送给内部人员。整体功能界面如图4.7所示。



图4.7

主要代码实现：

/\*\*

\* 发送邮件

\* @param mail 邮件

\* @param receiverNames 接收者

\* @param status 0 草稿 1 发送

\*/

private void sendMassage(Mail mail,String receiverNames,String status){

List<User> receiverList=Lists.newArrayList();

for (User user : DictUtils.fundUsers())

{

if(receiverNames!=null&&receiverNames.contains(user.getName())){

receiverList.add(user);

}

}

MailCompose mailCompose=new MailCompose();

mailCompose.setStatus(status);

mailCompose.setReceiverList(receiverList);

mailService.saveOnlyMain(mail);

mailCompose.setMail(mail);

Date date = new Date(System.currentTimeMillis());

mailCompose.setSender(UserUtils.getUser());

mailCompose.setSendtime(date);

for(User receiver : receiverList){

mailCompose.setReceiver(receiver);

mailCompose.setId(null);//标记为新纪录，每次往发件箱插入一条记录

mailCompose.setIsNewRecord(true);

mailComposeService.save(mailCompose);//0 显示在草稿箱，1 显示在已发送需同时保存到收信人的收件箱。

if(mailCompose.getStatus().equals("1"))//已发送，同时保存到收信人的收件箱

{

MailBox mailBox = new MailBox();

mailBox.setReadstatus("0");

mailBox.setReceiver(receiver);

mailBox.setSender(UserUtils.getUser());

mailBox.setMail(mailCompose.getMail());

mailBox.setSendtime(date);

mailBoxService.save(mailBox);

}

}

}