成 绩:\_\_\_\_\_\_\_\_\_\_\_\_

教师签名:\_\_\_\_\_\_\_\_\_\_\_\_



**需求分析说明书**

**项目名称:** 物业信息管理

教 师: 胡峡进

姓 名: 唐可寅

学 号: 201710253110

班 级: 17gb计算3班

日 期: 2020年07月06日

目录

[一、概述 1](#_Toc1501021252)

[1、摘要 1](#_Toc1173781055)

[2、系统的应用背景 1](#_Toc953410043)

[3、系统的开发意义 1](#_Toc1587102434)

[二、需求分析 3](#_Toc536228851)

[1、可行性研究 3](#_Toc1556915945)

[2、 系统需求分析 3](#_Toc2145532567)

[三、系统总体设计 4](#_Toc1567936792)

[1、系统设计 4](#_Toc541830807)

[项目环境和依赖 4](#_Toc1219709969)

[分层分析 6](#_Toc1934038368)

[2、系统详细设计 11](#_Toc1070369984)

[前言 11](#_Toc237810169)

[概要设计 11](#_Toc408443316)

[详细设计 11](#_Toc1349076200)

[四、系统的编码、测试和总结 16](#_Toc791348374)

[1、系统代码 16](#_Toc825895947)

[2、系统的测试 21](#_Toc1646370668)

[五、总结 23](#_Toc225025481)

# 一、概述

## 1、摘要

随着国民经济的进步和互联网的快速发展，传统的依靠人工进行物业管理的模式已然无法满足当下物业管理需求。将计算机技术应用物业管理中，能够有助于帮助改善并习成一套标准化的住宅管理模式。对于管理层而言，这将大幅降低物业企业管理成本，提高物业企业收益率。与此同时，依靠人工进行物业管理的模式已不能日渐发展的新社区应提供的服务需要，再此基础上本次课程设计利用当下流行的Java程序设计语言进行系统设计，并整合SSM框架，定义各种业务逻辑类和方法来编写并实现最终业务逻辑。

本次课程设计选用Java语言整合SSM框架，数据库选用MySQL进行数据管理，选用Tomcat作为应用服务器。

本次课程设计实现管理用户登录、注册以及进行角色管理的功能。

本次设计首先对系统建设对各个层面进行了基本定义，并在其基础上进行了简单分析，阐述中给出了部分UML图以说明系统结构。本次设计以基于文本信息传送和接收功能的聊天室系统为基准实行此次系统的开发。

## 2、系统的应用背景

开发的软件系统的名称：物业信息管理。

我国已经逐步从4G时代迈入5G时代，而信息产业的飞速发展是其最大的推动力，软件作为现代数据信息的重要支撑形式，发挥着越来越重要的作用。在当今社会，随着我国改革开放的深入，市场经济飞速发展，很多人都过上了小康生活，从农村地区走向了大城市。大城市人口越来越多，就自然而然带来了住房问题。对于住房问题就不可避免的会涉及到进行物业管理，这是一个非常重要的问题，人们的生活离不开衣食住行，所以这个问题必须解决，而对于衣食住行问题中住的问题不能忽略的就是物业管理。

随着社会的进程和科学技术的发展，越来越多的商品房拔地而起，楼层越来越高，单位面积容载的人越来越多。我国早期采取的是分房制的房产制度。后来慢慢发展成了商品房经济，商品房模式为我国经济增添了新活力，解决了人们的居住问题，提高了我国人民的生活水平。随着市场饱和的同时也逐渐带来了新的问题，当单位面积居住的人越来越多时，对于传统物业管理困难的问题就越来越严峻。由于物业管理具有客户多、事务广的特点，给管理工作大大增加了难度，针对这些问题很多学者都寻求过解决方案，现在普遍被认可的方案就是基于数字化管理系统来整合资源，提升物业管理水平和效率。本次设计主要利用Java程序，结合数据库形成一个小型的物业管理系统，主要功能有提供给管理员用户的登录注册功能，系统角色管理等功能。

## 3、系统的开发意义

物业管理系统一旦使用基本下包含以下几个方面的意义：

1. 对于物业来说首先要将整个小区的资产信息全部罗列整理出来，原本纸质版文件不仅保存麻烦，并且查询的时候还非常困难。如果引入这个系通过后将所有信息全部打包存入数据库中，就非常节约储存空间。并且查询时还非常方面任何人想要查询直接在前端输入一些信息就可以了；
2. 对于物业管理来说肯定少不了物业费，传统的收费方式是一家一家上门收取，这种方法极大的浪费时间和人力成本，并且有些业主由于工作等关系可能无法按时缴纳物业费。那么在线上缴费就不受时间、空间的限制可以随时随地缴费。并且对于一些和小区有关的信息也可以通过网络发布，这样就避免了传统张贴信息方式而被业主忽略；
3. 小区业主在使用住房过程中肯定会出现而各种各样的问题，在出现公共问题时都是找物业进行维修、更换。在没有引入物业管理系统之前，这些都是靠业主线下前往物业进行报修，而引入物业管理系统后这些都可以线上进行操作；

除了上述列举的 3 个主要积极意义外，物业管理系统对物业公司、租户、业主三方都有着不同程度的意义，以此不管从业主，物业还是社会与时俱进方面来说对于物业管理系统的研究都非常重要。

# 二、需求分析

## 1、可行性研究

（1）管理可行性

随着时代的发展，人员素质己经逐步提高，不论是对于计算机系统的基本操作还是对于系统的维护都有了一定的基础。同时还可以配置专业的计算机维护人员来维护电脑，不必担心计算机故障问题。

1. 技术可行性

随着国内软件开发的日益发展壮大，网站开发的软件层出不穷，诸如Dreamweaver, Freehand, Fireworks, Flash 等工具都是网站开发的理想工具。本系统是在IntelliJ IDEA中进行代码设计与Java编程的，它属于一个 Web 数据库应用类的系统，其基本操作是建立 Web 界面和对数据库中信息进行录入、删除、修改等操作。从技术结构上来说是可行的。

1. 社会因素可行性分析

目前已存在许多成功的物业信息管理系统案例，社会需要更加全面化和智能化的物业管理信息系统。本系统的开发也将严格的遵照国家规定的相关法律进行。执行国家规定的费率标准，及宣传法规。另外，系统的操作和工作方式也符合管理人员的日常业务处理习惯。而且操作方便灵活，便于学习，日常的维护更新。因此，具有可行性。

## 系统需求分析

系统的工作总体规划由该系统管理人员在系统中完成对各种所需的基本数据的维护，包括相关列车信息的增加、修改以及对各项信息的变动都将在这进行操作。

后台管理主要由系统建设人员和系统管理员进行操作。他们将可以对系统用户进行管理、对用户角色进行管理。

前台主要是为了让物业管理人员能够快速方便地进行收缴费、维护物业系统用户权限等功能。

本系统包含如下基本功能：

1. 浏览器端页面
   1. 提供管理员用户登录注册的功能；
   2. 提供对用户角色进行增删改查的功能；
   3. 提供基于搜索获取用户角色信息的功能；
   4. 提供友好的用户界面；
2. 服务端
   1. 提供向所有使用者提供服务的能力；
   2. 提供用户登录注册的能力；
   3. 提供对用户角色进行增删改查的能力。

# 三、系统总体设计

## 1、系统设计

### **项目环境和依赖**

* 开发环境
  + 开发语言：C#
  + 开发工具：Visual Studio 2019 & JetBrains Rider 2020.1.4
  + 数据库：Microsoft SQL SERVER 2019
* 环境配置
  + 访问路径：http://127.0.0.1:8080/bang/
  + 服务端口号：8080
* 第三方库依赖
  + org.springframework: spring-beans
  + org.springframework: spring-instrument
  + org.springframework: spring-jdbc
  + org.springframework: spring-jms
  + org.springframework: spring-test
  + org.springframework: spring-core
  + org.springframework: spring-aspects
  + org.springframework: spring-oxm
  + org.springframework: spring-tx
  + org.springframework: spring-jdbc
  + org.springframework: spring-aop
  + org.springframework: spring-context
  + org.springframework: spring-context-support
  + org.springframework: spring-expression
  + org.springframework: spring-orm
  + org.springframework: spring-web
  + org.springframework: spring-webmvc
  + org.mybatis: mybatis
  + org.mybatis: mybatis-spring 1.2.2
  + mysql: mysql-connector-java 8.0.20
  + commons-dbcp: commons-dbcp 1.4
  + jstl: jstl 1.2
  + log4j: log4j 1.2.17
  + org.slf4j: slf4j-api
  + org.slf4j: slf4j-log4j12
  + com.alibaba: druid
  + com.alibaba: fastjson 1.2.72
  + org.projectlombok: lombok 1.18.12
  + junit: junit 4.13
  + javax.servlet: javax.servlet-api 3.1.0
  + javax.servlet.jsp: jsp-api 2.2
  + javax.servlet: jstl 1.2

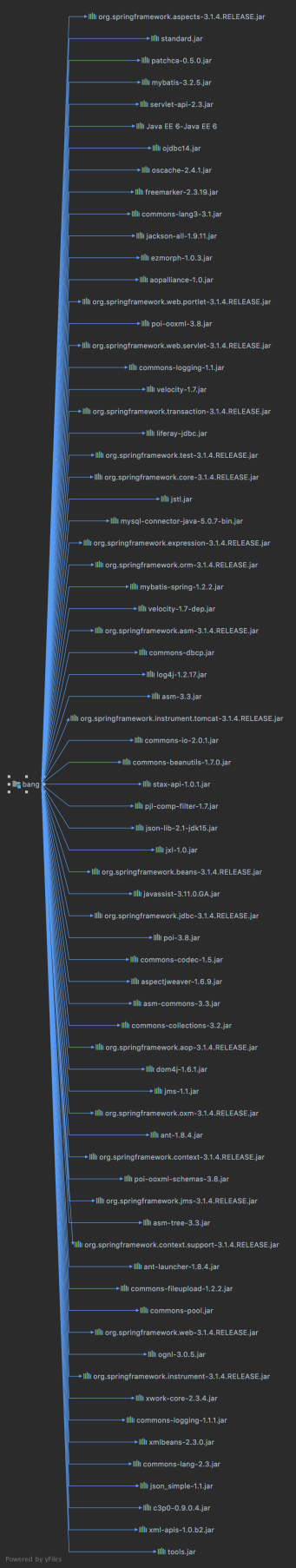


图 1第三方库依赖

### 分层分析

1. 驱动和中间介层
   1. 数据库驱动-JDBC

JDBC（Java Database Connectivity，Java 数据库连接）是一种用于执行 SQL 语句的 Java API 可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

JDBC 是个“低级”接口，也就是说，它用于直接调用 SQL 命令。在这方面它的功能极佳，并比其它的数据库连接 API 易于使用，但它同时也被设计为一种基础接口，在它之上可以建立高级接口和工具。高级接口是“对用户友好的”接口，它使用的是一种更易理解和更为方便的 API，这种 API 在幕后被转换为诸如 JDBC这样的低级接口。

在关系数据库的“对象/关系”映射中，表中的每行对应于类的一个实例，而每列的值对应于该实例的一个属性。于是，程序员可直接对 Java 对象进行操作；存取数据所需的 SQL 调用将在“掩盖下”自动生成。此外还可提供更复杂的映射，例如将多个表中的行结合进一个 Java 类中。

随着人们对 JDBC 的兴趣日益増涨，越来越多的开发人员一直在使用基于 JDBC 的工具，以使程序的编写更加容易。程序员也一直在编写力图使最终用户对数据库的访问变得更为简单的应用程序。例如，应用程序可提供一个选择数据库任务的菜单。任务被选定后，应用程序将给出提示及空白供填写执行选定任务所需的信息。所需信息输入应用程序将自动调用所需的 SQL 命令。在这样一种程序的协助下，即使用户根本不懂 SQL 的语法，也可以执行数据库任务。

JavaSoft 提供三种 JDBC 产品组件，它们是 Java 开发工具包（JDK）的组成部份：JDBC 驱动程管理器、JDBC 驱动程序测试工具包和 JDBC-ODBC 桥。

JDBC 驱动程序管理器是 JDBC 体系结构的支柱。它实际上很小，也很简单；其主要作用是把 Java 应用程序连接到正确的 JDBC 驱动程序上，然后即退出。

JDBC 驱动程序測试工具包为使 JDBC 驱动程序运行您的程序提供一定的可信度。只有通过 JDBC 驱动程序测试的驱动程序才被认为是符合 JDBC 标准 TM 的。

JDBC-ODBC 桥使 ODBC 驱动程序可被用作 JDBC 驱动程序。它的实现为 JDBC 的快速发展供了一条途径，其长远目标提供一种访问某些不常见的 DBMS（如果对这些不常见的 DBMS 未实现 JDBC）的方法。

使用的时候，需要首先下载相应的驱动，然后配置 Java 工程里面的环境，再书写代码。因为 JDBC 是一个开放的标准，所以在使用的时候。需要先用反射加载驱动。可以使用 ClassLoader，也可以直接用 Class.forName 进行反射，反射的时候会自动运行驱动程序类的静态代码块，在静态代码块里面会进行数据库连接的初始化。同时因为这个关系，如果静态代码块初始出现异常，数据库功能将不能使用，除非重启程序。

1. Spring 框架层

Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。Spring的核心是控制反转（IoC）和面向切面（AOP）。简单来说，Spring是一个分层的JavaSE/EE full-stack(一站式) 轻量级开源框架。

1. MyBatis 框架层

MyBatis是一个半ORM（对象关系映射）框架，它内部封装了JDBC，开发时只需要关注SQL语句本身，不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。程序员直接编写原生态SQL，可以严格控制SQL执行性能，灵活度高。

MyBatis 可以使用 XML 或注解来配置和映射原生信息，将 POJO映射成数据库中的记录，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

通过XML文件或注解的方式将要执行的各种 statement 配置起来，并通过java对象和 statement中SQL的动态参数进行映射生成最终执行的SQL语句，最后由MyBatis 框架执行SQL并将结果映射为Java对象并返回。（从执行SQL到返回result的过程）。

1. Model和View层
   1. Model和View层的关系

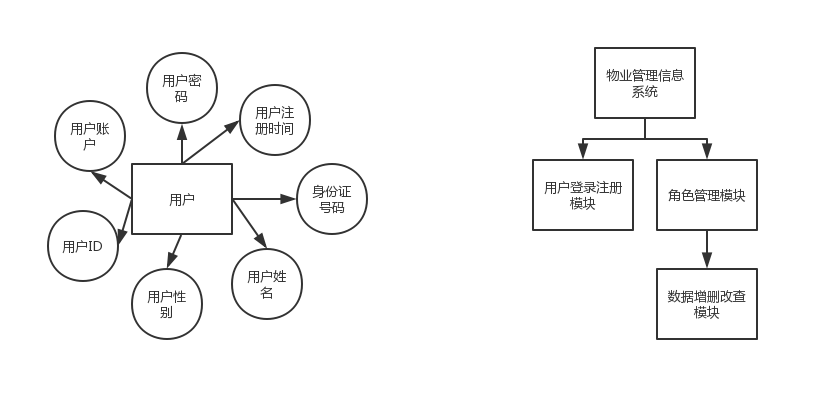


图 2 Model和View层的关系E-R图

1. Constant层

该层根据阿里巴巴Java代码设计在应用模块包下建立constant包，建立ConstConfig程序类存放应用级共享常量，这将可以消除编码过程中可能出现的绝大多数魔法值，使应用返回信息统一，实现一处修改多处同步，使程序复杂度降低。

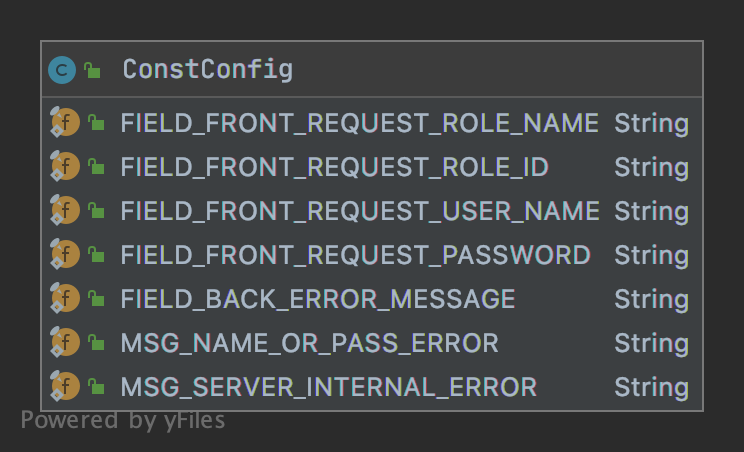


图 3 Constant层ConstConfig类图

1. DAO层

角色信息表：用来存储用户角色信息，如角色ID、角色名称等。

用户信息表：用来存储用户信息，如用户ID、用户账户、用户密码、用户注册时间、身份证号码、用户姓名、用户性别等。

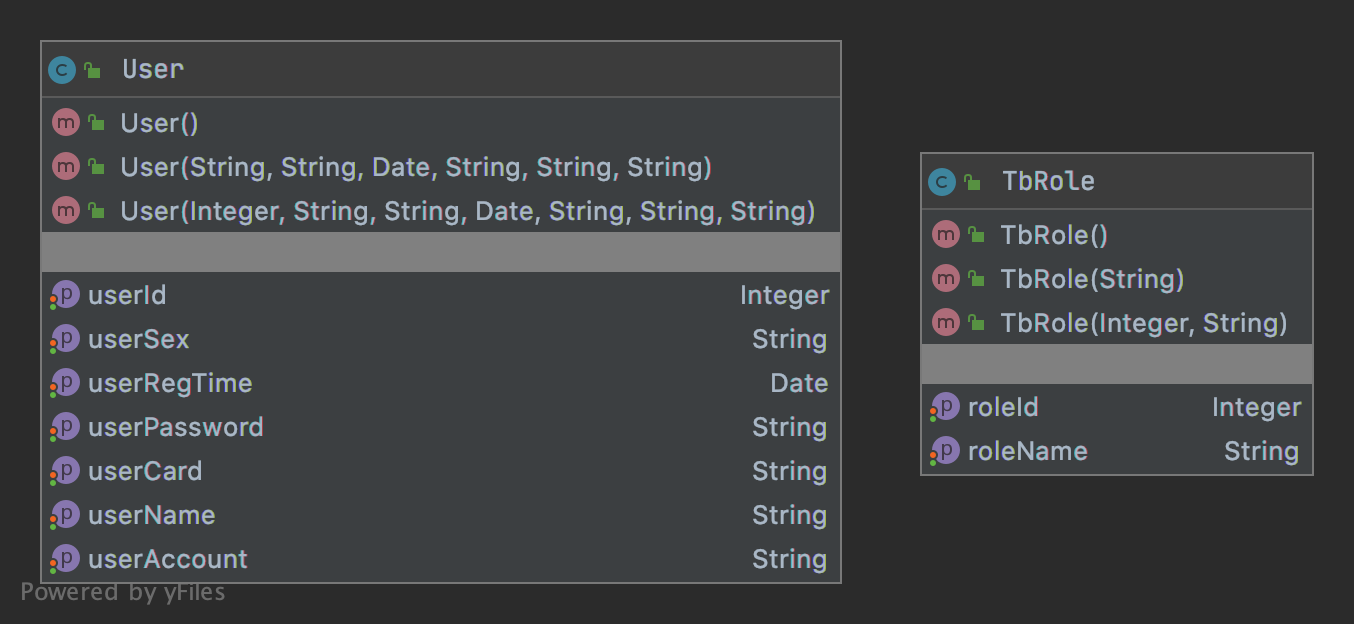


图 4 Entity实体类

1. E-R图

以上表都是独立的实体，这些实体的 E-R 图设计为：

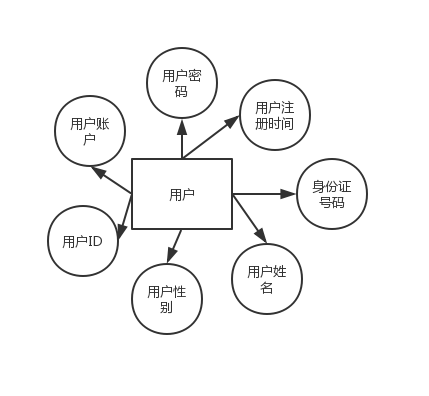


图 5实体E-R图

1. Service层

该层包含的主要是对 DAO 层的操作。这里提供了针对DAO层TrainInfo表的操作方法。根据以上实体之间的关系，设计出本层的UML类图。从图中我们可以清楚得看出类间依赖关系。

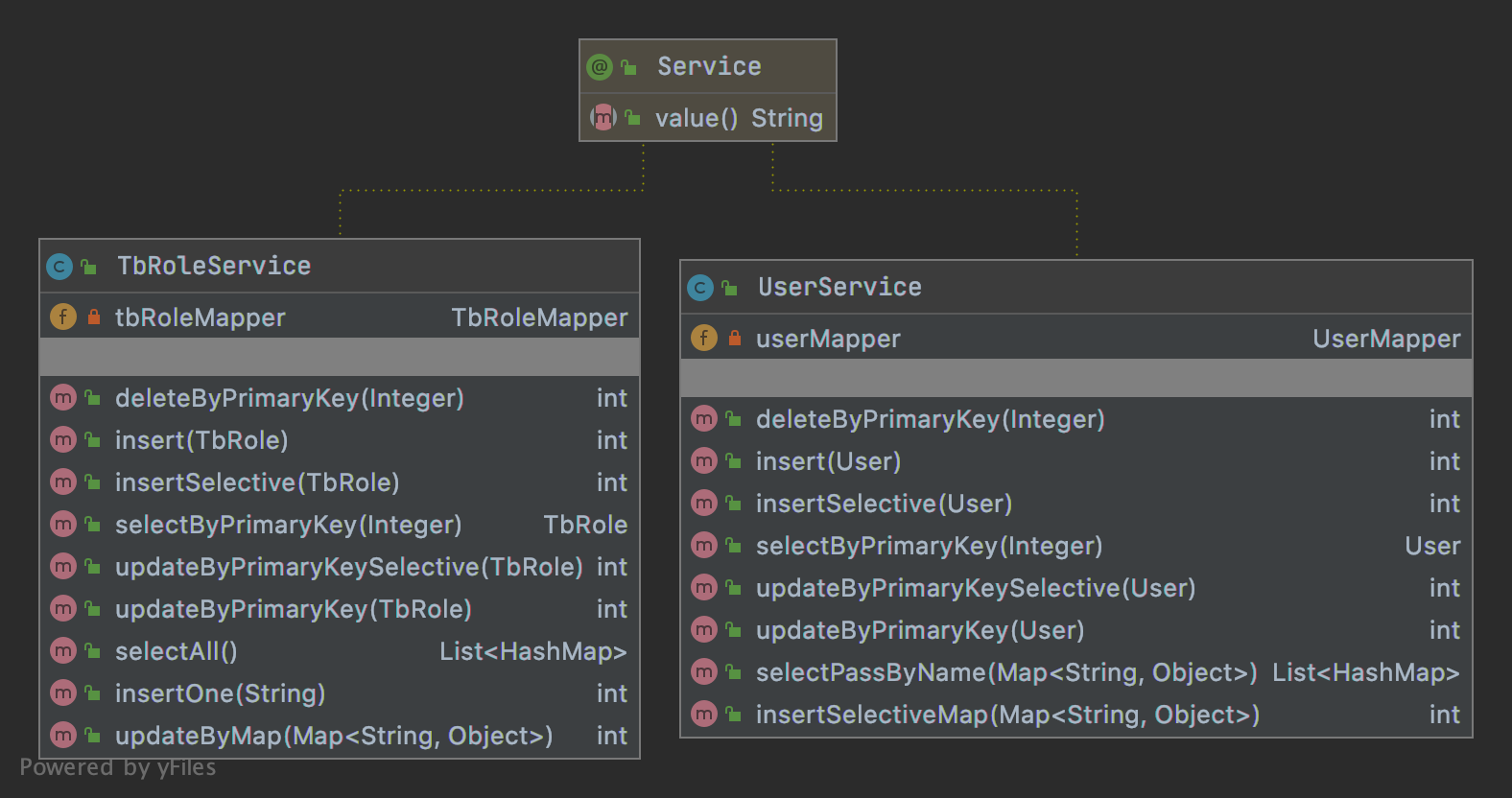


图 6 Service层E-R图

1. Controller 层
   1. IndexController、LoginController、RoleController

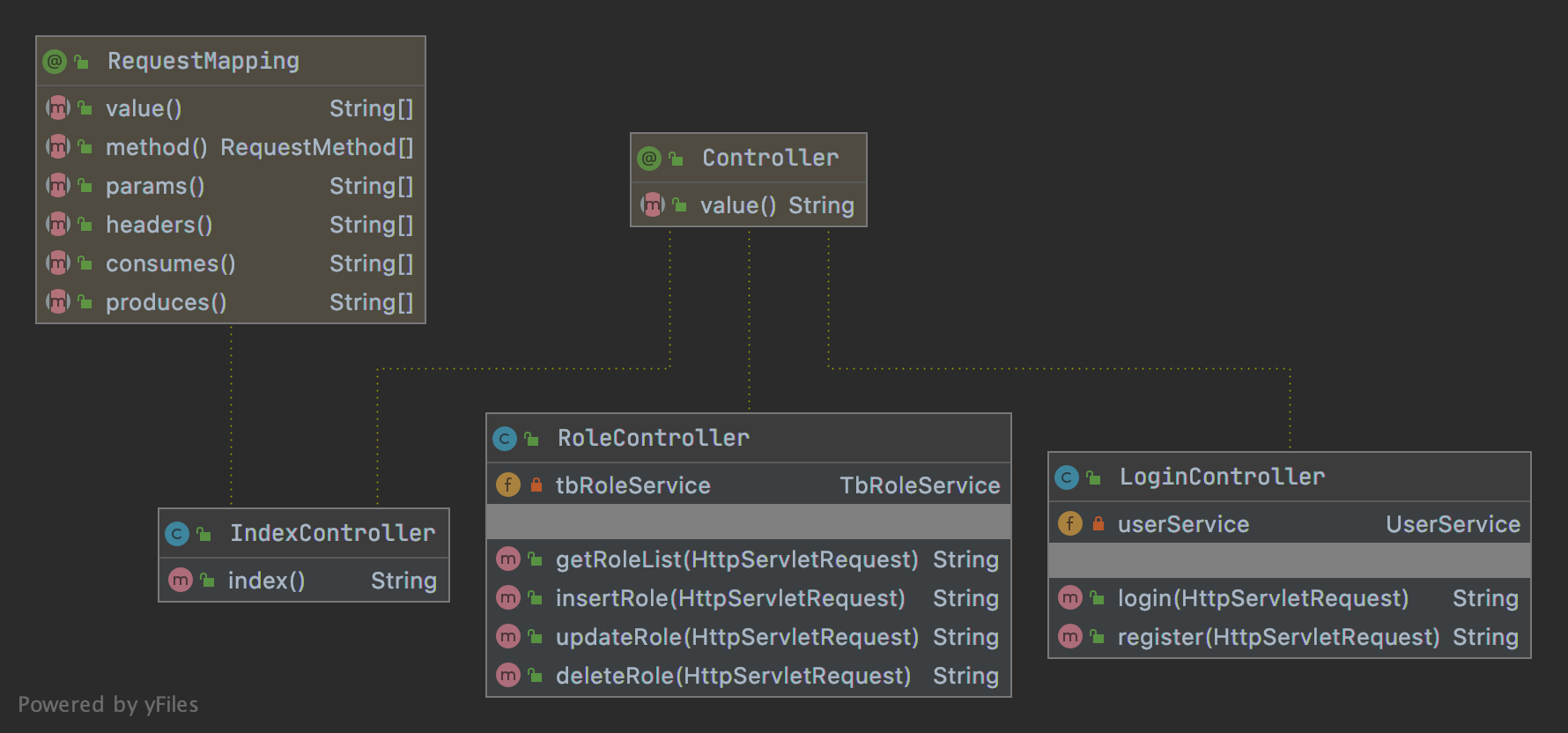


图 7 Controller UML类图

1. UML

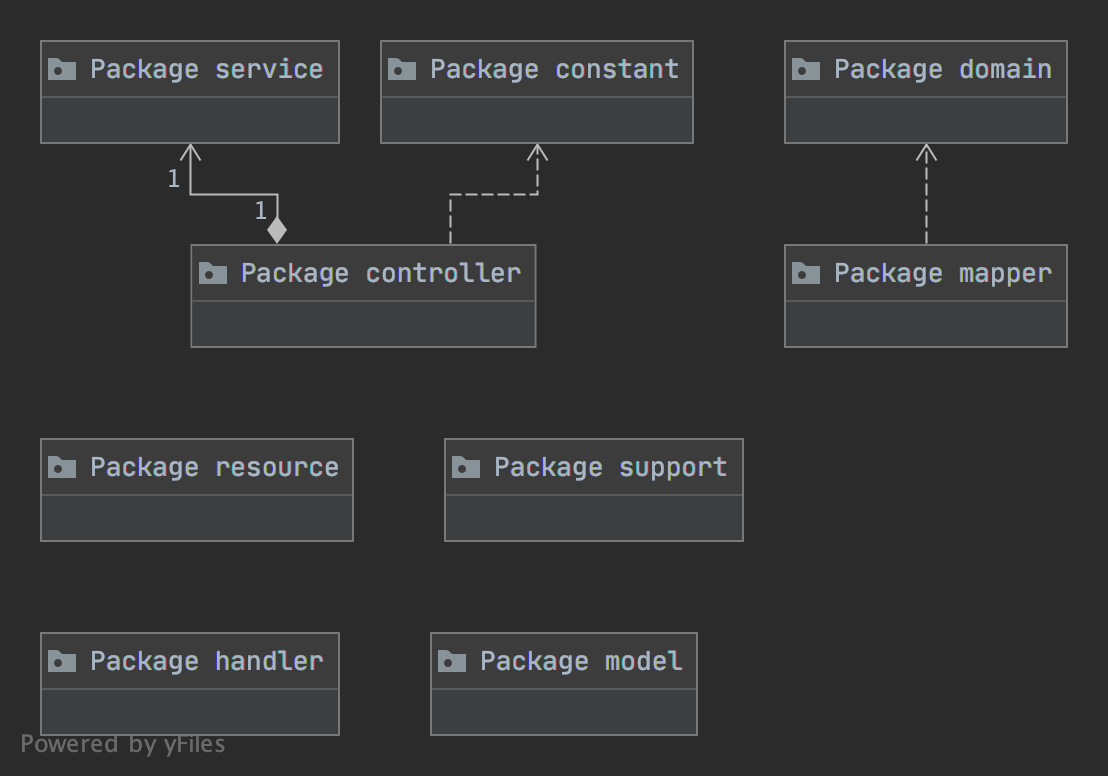


图 8程序包依赖图

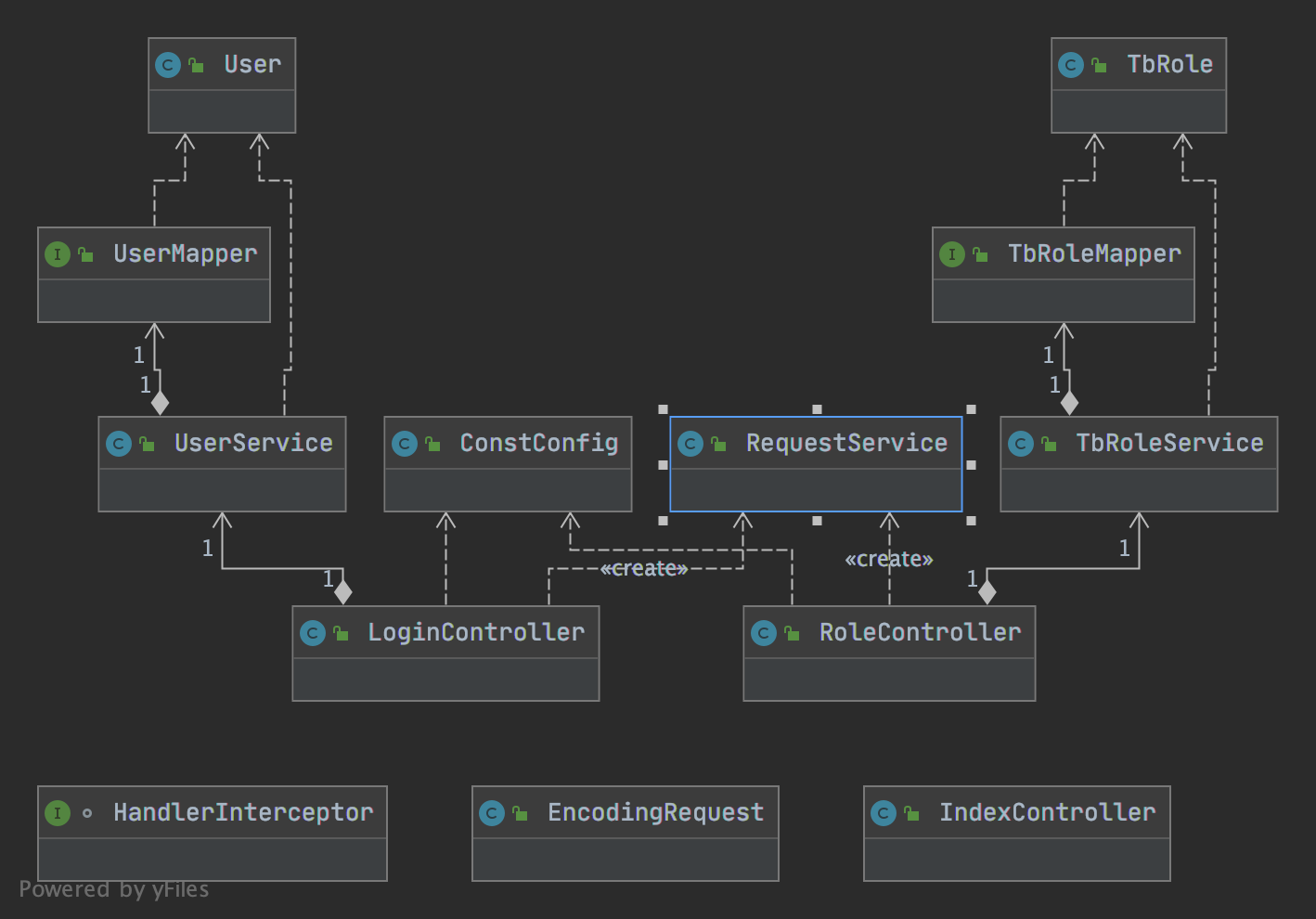


图 9 UML 类图

## 2、系统详细设计

### 前言

随着国民经济的进步和互联网的快速发展，传统的依靠人工进行物业管理的模式已然无法满足当下物业管理需求。将计算机技术应用物业管理中，能够有助于帮助改善并习成一套标准化的住宅管理模式。对于管理层而言，这将大幅降低物业企业管理成本，提高物业企业收益率。与此同时，依靠人工进行物业管理的模式已不能日渐发展的新社区应提供的服务需要，再此基础上本次课程设计利用当下流行的Java程序设计语言进行系统设计，并整合SSM框架，定义各种业务逻辑类和方法来编写并实现最终业务逻辑。

本系统主要利用 Spring MVC + Spring + MyBatis 技术进行开发，有很强的逻辑性、可扩展性，便于维护。物业管理信息系统设计主要使用了CSS进行聊天界面的美化。其次本系统设计还使用了 SSM、JSP 、JDBC 、JavaBean 、CSS、JavaScript、jQuery、AJAX等技术。

### 概要设计

1. 物业管理信息系统简要功能介绍

本次课程设计提供的简易物业管理信息系统是为社区物业服务公司提供的物业管理系统，能基本实现管理员用户登录和用户角色管理的功能。

1. 用户登录注册

功能描述：面向管理员用户提供的登录注册功能；

1. 用户角色管理

功能描述：面向管理员用户提供的用户角色管理功能；

### 详细设计

1. 服务端设计
   1. 应用级常量
      1. ConstConfig.java
         1. 前台回传/请求数据字段（input标签name属性名）名

|  |  |
| --- | --- |
| 常量名 | 常量值 |
| FIELD\_FRONT\_REQUEST\_ROLE\_NAME | "roleName" |
| FIELD\_FRONT\_REQUEST\_ROLE\_ID | "roleId" |
| FIELD\_FRONT\_REQUEST\_USER\_NAME | "userName" |
| FIELD\_FRONT\_REQUEST\_PASSWORD | "password" |
| FIELD\_BACK\_ERROR\_MESSAGE | "errMsg" |

* + - 1. 后台向前台发送的回显消息

|  |  |
| --- | --- |
| 常量名 | 常量值 |
| MSG\_NAME\_OR\_PASS\_ERROR | "用户名或密码错误" |
| MSG\_SERVER\_INTERNAL\_ERROR | "服务器内部错误" |

* 1. Package Service
     1. TbRoleService.java
        1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.service

* + - 1. 功能描述

此程序类依赖于TbRoleMapper.java接口，它完成了对MyBatis Mapper的操纵功能，对数据库中Tb\_role表进行操作，能帮助开发人员快速的扩展接口能力，完成对数据库的增删改查。

* + 1. UserService.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.service

* + - 1. 功能描述

此程序类依赖于UserMapper.java接口，它完成了对MyBatis Mapper的操纵功能，对数据库中tb\_user表进行操作，能帮助开发人员快速的扩展接口能力，完成对数据库的增删改查。

* 1. Package domain
     1. User.java
        1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.domain

* + - 1. 功能描述

此程序类是管理员用户的实体类，用于处理对管理员用户信息实体的操作。

此程序类包含七个字段：userId（用户编号）、userAccount（用户账户）、userPassword（用户密码）、userRegTime（用户注册时间）、userCard（用户身份证号码）、userName（用户姓名）、userSex（用户性别）。

此程序类包含重写的toString()方法和三个构造方法，以及对每个字段的Getter & Setter方法。

* + 1. TbRole.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.domain

* + - 1. 功能描述

此程序类是用户角色信息的实体类，用于处理对用户角色信息实体的操作。

此程序类包含两个字段：roleId（角色编号）、roleName（角色名称）。

此程序类包含重写的toString()方法和三个构造方法，以及对每个字段的Getter & Setter方法。

* 1. Package controller
     1. IndexController.java
        1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.controller

* + - 1. 功能描述

此程序类1项方法：index()。

* + - * 1. index()

此方法向用户提供项目首页跳转的能力，即使用虚拟路径映射访问项目首页。

* + 1. LoginController.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.controller

* + - 1. 功能描述

此程序类2项方法：login()register()。

* + - * 1. login()

此方法向用户提供管理员用户进行登录的能力，使用此方法将向数据库中提取用户输入的账号信息并进行用户身份核验。

* + - * 1. register()

此方法向用户提供管理员用户进行注册的能力，使用此方法将向数据库添加用户提交的新用户信息。

* + 1. RoleController.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.controller

* + - 1. 功能描述

此程序类4项方法：getRoleList()、getTrain()和deleteTrainItem()。

* + - * 1. getRoleList()

此方法向用户提供获得角色信息列表的能力，使用此方法系统将从数据库中提取所有的角色信息，并在Session中存储，执行完成后浏览器页面将重定向至角色信息列表页。

* + - * 1. insertRole()

此方法向用户提供插入角色信息的能力，使用此方法将向数据库添加用户提交的新角色信息。

* + - * 1. updateRole()

此方法向用户提供修改已有角色信息的能力，使用此方法将向数据库更新用户提交的角色信息。

* + - * 1. deleteRole()

此方法向用户提供删除角色信息的能力，使用此方法将向数据库删除用户于前台指定的部分数据。

* 1. Package interceptor
     1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.interceptor

* + 1. 功能描述

此程序类是本系统权限访问拦截器，在每次用户请求之时过滤未登录的用户并进行拦截操作，被拦截的用户浏览器将会自动跳转至登录页面。

某些用户在已知聊天室系统主界面URL的情况下，可能会未经登录直接访问，进而将可能造成系统异常、其他用户读取到脏数据，使其他用户使用体验造成不良影响，即本系统不允许使用匿名访问功能。

* 1. Package mapper
     1. TbRoleMapper.java
        1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.mapper

* + - 1. 功能描述

此程序接口依赖于TbRoleMapper.xml配置文件，由@Repository注解进行标注，依赖于MyBatis持久层框架提供的服务对数据库进行操作，此接口操作数据库对象为Tb\_role。

* + 1. UserMapper.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.mapper

* + - 1. 功能描述

此程序接口依赖于UserMapper.xml配置文件，由@Repository注解进行标注，依赖于MyBatis持久层框架提供的服务对数据库进行操作，此接口操作数据库对象为tb\_user。

* 1. Package model
     1. TbRoleMapper.java
        1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.model

* + - 1. 功能描述

此配置文件是MyBatis持久层框架进行ORM操作的SQL操纵文件，其中包含会数据库的增删改查操作，此配置文件操纵数据库为Tb\_role。

* + 1. UserMapper.java
       1. 程序包路径：

com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.model

* + - 1. 功能描述

此配置文件是MyBatis持久层框架进行ORM操作的SQL操纵文件，其中包含会数据库的增删改查操作，此配置文件操纵数据库为tb\_user。

* 1. Package com.kevinkda.core
     1. SubPackage com.kevinkda.core.util.web.support
        1. EncodingRequest.java
           1. 程序包路径：

com.kevinkda.core.util.web.support

* + - * 1. 功能描述

此程序类继承自HttpServletRequestWrapper类，通过覆写父类方法以将Filter中的原有ServletRequest对象进行替换，得到新的HttpServletRequest对象。通过此方法可以对Request对象进行更多功能的扩充。

此程序类在此处的方法是将用户从前台提交的ISO-8859-1字符编码转换为UTF-8字符编码。

* + 1. SubPackage com.kevinkda.web.servlet.request
       1. RequestService.java
          1. 程序包路径：

com.kevinkda.web.servlet.request

* + - * 1. 功能描述

此程序类提供获得Request请求数据的能力，能够一次性获取前台用户提交的请求头、请求体数据。此程序类能够帮助开发人员进行更加高效的开发。

1. 前台页面设计
   1. JavaScript

本系统中使用jQuery技术完成JS的快速开发。其版本使用jQuery 1.12.4，本系统开发过程中采用离线依赖，文件存储于web/js/目录下。

* 1. JSP
     1. 根目录
        1. index.jsp

此页面是项目首页，但此页面会自动跳转至login.jsp，进行用户登录。

* + - 1. login.jsp

此页面是用户登录页，提供给用户进行登录控制。

此页面提供的基于CSS样式的美化，给到用户良好的使用体验。页面背景采用CSS渐变色样式设计，具体代码如下：

background: linear-gradient(to top right, #5df5d5, #9ff8b8, #c8f3b1, #f4ffbb);

* + - 1. register.jsp

此页面是用户注册页，提供给用户进行注册操作。

此页面提供的基于CSS样式的美化，给到用户良好的使用体验。页面背景采用CSS渐变色样式设计，具体代码如下：

background: linear-gradient(to top right, #5df5d5, #9ff8b8, #c8f3b1, #f4ffbb);

* + 1. admin目录
       1. roleList.jsp

此页面将会向本系统获取角色信息列表，并向用户显示。

此页面提供的基于CSS样式的美化，给到用户良好的使用体验。页面背景采用CSS渐变色样式设计，具体代码如下：

background: linear-gradient(to right, #91c9e8, #47a0e5);

* + - 1. insertRole.jsp

此页面将用于向数据库添加角色信息，此页面使用Post方式提交数据。

此页面提供的基于CSS样式的美化，给到用户良好的使用体验。页面背景采用CSS带透明值的纯色背景样式设计，具体代码如下：

background: rgba(126, 225, 255, 0.68);

* + - 1. msg\_info.jsp

此页面将用于向数据库修改角色信息，此页面使用Post方式提交数据。

此页面提供的基于CSS样式的美化，给到用户良好的使用体验。页面背景采用CSS渐变色样式设计，具体代码如下：

background: linear-gradient(to right, #91c9e8, #47a0e5);

# 四、系统的编码、测试和总结

## 1、系统代码

获得Request请求信息功能代码：

RequestService.java

public class RequestService {

// 处理Text形式的数据

/\*\*

\* 获得所有请求头和请求体数据，请求数据类型需为Text

\*

\* @param request 传入请求的 Request {@link HttpServletRequest}

\* @return {@code java.util.List<java.util.Map<java.lang.String,java.lang.Object>>} 回传请求头和请求体数据

\* @author Kevin KDA on 2020/5/11 21:46

\* @description RequestService / getParametersForText

\* @version 1.0.0

\* @apiNote <p>本方法返回{@link HttpServletRequest}包含的请求头和请求体数据</p>

\* <p>其中数组下标0是请求头数据</p>

\* <p>其中数组下标1是请求体数据</p>

\* @since 3.1.2

\*/

public List<Map<String, Object>> getParametersForText(HttpServletRequest request) {

// 返回请求头和请求体数组

return new ArrayList<>(

Arrays.asList(

// 加入请求头Map

getRequestHead(request),

// 加入请求体Map

getRequestBody(request)

)

);

}

/\*\*

\* 获得请求头数据

\*

\* @param request 传入请求的Request {@link HttpServletRequest}

\* @return {@code java.util.Map<java.lang.String, java.lang.Object>} 回传请求头数据

\* @author Kevin KDA on 2020/5/1 14:37

\* @description RequestService / getRequestHead

\* @version 1.0.0

\* @apiNote <p></p>

\* @since 3.0.5

\*/

public Map<String, Object> getRequestHead(HttpServletRequest request) {

// 获取所有的头Head信息

Enumeration<String> requestHeaderNames = request.getHeaderNames();

// 存储所有请求头数据

Map<String, Object> requestHead = new HashMap<>();

// 临时Key变量的存储空间，减少变量开辟和 GC 工作

String tempKey;

// requestHeaderNames 有值则继续

while (requestHeaderNames.hasMoreElements()) {

// 获得头信息类型，临时保存供取得值使用

tempKey = requestHeaderNames.nextElement();

// 获得请求类型值，并存入Map中

requestHead.put(tempKey, request.getHeader(tempKey));

}

// 返回所有请求头信息

return requestHead;

}

/\*\*

\* 获得请求体数据

\*

\* @param request 传入请求的Request {@link HttpServletRequest}

\* @return {@code java.util.Map<java.lang.String, java.lang.Object>} 回传请求体数据

\* @author Kevin KDA on 2020/5/11 21:54

\* @description RequestService / getRequestBody

\* @version 1.0.0

\* @apiNote <p></p>

\* @since 3.1.2

\*/

public Map<String, Object> getRequestBody(HttpServletRequest request) {

// 获取所有的头Head信息

Enumeration<String> requestHeaderNames = request.getParameterNames();

// 存储所有请求头数据

Map<String, Object> requestBody = new HashMap<>();

Map<?, ?> map = request.getParameterMap();

// 临时Key和Value变量的存储空间，减少变量开辟和 GC 工作

String paramName;

String[] paramValues;

// requestHeaderNames 有值则继续

while (requestHeaderNames.hasMoreElements()) {

// 获得头信息类型，临时保存供取得值使用

paramName = requestHeaderNames.nextElement();

// 获得头信息值，此处可能是数组，e.g.前端Checkbox传入的组对象

paramValues = request.getParameterValues(paramName);

// 检查头信息值是否存在，如数组长度仅一位，保存数组首位

if (paramValues.length == 1) {

// 获得临时数据，用于检验值的有效性，即数据是否为空，为空即舍弃

String paramValue = paramValues[0];

// 检验值的有效性，即数据是否为空，为空即舍弃

// if (paramValue.length() != 0) {

if (!(null == paramValue || "".equals(paramValue))) {

// 数据不为空存入Map

requestBody.put(paramName, paramValue);

}

} else if (paramValues.length > 1) {

// 如数组长度大于一位，保存数组

requestBody.put(paramName, paramValues);

}

}

// 返回所有请求头信息

return requestBody;

}

/\*\*

\* 打印所有参数

\*

\* @param map 传入已获得的请求参数，需为 {@code Map<String, Object>} 对象

\* @return void 本方法直接输出，无返回值

\* @author Kevin KDA on 2020/5/11 22:15

\* @description RequestService / printParameters

\* @version 1.0.0

\* @apiNote <p></p>

\* @since 3.1.2

\*/

public void printParameters(Map<String, Object> map) {

Set<Map.Entry<String, Object>> set = map.entrySet();

for (Map.Entry entry :

set) {

System.out.println(entry.getKey() + ":" + entry.getValue());

}

// 等价于上forEach

// set.stream().map(entry -> entry.getKey() + ":" + entry.getValue()).forEach(System.out::println);

}

/\*\*

\* 显示获得的所有参数

\*

\* @param map 传入已获得的请求参数，需为 {@code Map<String, Object>} 对象

\* @return java.lang.String[] 返回所有参数的String[]信息

\* @author Kevin KDA on 2020/5/11 22:15

\* @description RequestService / showParameters

\* @version 1.0.0

\* @apiNote <p></p>

\* @since 3.1.2

\*/

public String[] showParameters(Map<String, Object> map) {

List<String> list = new ArrayList<>();

Set<Map.Entry<String, Object>> set = map.entrySet();

for (Map.Entry entry :

set) {

list.add(entry.getKey() + ":" + entry.getValue());

}

// 等价于上forEach

// (String[]) set.stream().map(entry -> entry.getKey() + ":" + entry.getValue()).toArray();

return (String[]) list.toArray();

}

/\*\*

\* 获得Post请求体数据

\*

\* @param request 传入请求的Request {@link HttpServletRequest}

\* @return java.lang.String 返回请求的方法体

\* @author Kevin KDA on 2020/5/1 14:30

\* @description RequestService / getPostBody

\* @version 1.0.0

\* @apiNote <p></p>

\* @since 3.0.5

\*/

public String getPostBody(HttpServletRequest request) {

// 记录请求体数据

String bodyInfo = "";

try {

// 获得输入流对象

InputStream i;

// 从request中取得输入流，即获得请求体

i = request.getInputStream();

// 获得请求体，并转换为UTF-8字符编码字符串

bodyInfo = IOUtils.toString(i, String.valueOf(StandardCharsets.UTF\_8));

} catch (IOException e) {

e.printStackTrace();

}

// 返回请求体数据

return bodyInfo;

}

// 处理JSON形式的数据

/\*\*

\* 获得所有请求头和请求体数据，请求数据类型需为JSON

\*

\* @param request 传入请求的 Request {@link HttpServletRequest}

\* @return {@code java.util.List<java.util.Map<java.lang.String,java.lang.Object>>} 回传请求头和请求体数据

\* @author Kevin KDA on 2020/5/11 21:46

\* @description RequestService / getParametersForText

\* @version 1.0.0

\* @apiNote <p>本方法返回{@link HttpServletRequest}包含的请求头和请求体数据</p>

\* <p>其中数组下标0是请求头数据</p>

\* <p>其中数组下标1是请求体数据</p>

\* @since 3.1.2

\*/

public List<Map<String, Object>> getParametersForJson(HttpServletRequest request) {

//// 返回请求头和请求体数组

// return new ArrayList<>(

// Arrays.asList(

//// 加入请求头Map

// getRequestHead(request),

//// 加入请求体Map

// getRequestBody(request)

// )

// );

return null;

}

// Constructor、Getter、Setter

public RequestService() {

}

}

包扫描器代码：

<context:component-scan base-package="com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.service" />

<context:component-scan base-package="com.kevinkda.univ.curriculum.design.enterprise.framework.estate.manager.controller"/>

## 2、系统的测试

软件测试的目的和原则

在开发软件系统的过程中，我们需要面对错综复杂的问题，不可避免的在每个生产阶段都会产生一些错误，我们力求进行严格的技术审查在每个阶段任务结束前尽可能多的发现和纠正软件中的错误。而软件测试的目的就是在软件投入生产环境前尽可能的修正软件中的错误，目前软件测试仍然是软件质量控制的最后防线。

软件测试分以下三种类型：

1. 单元测试：单元测试的用例从单元详细设计中导出。在单元测试中可以采用功能性测试和结构性测试两种；
2. 集成测试：在这个测试步骤中所发现的往往是概要设计的错误；
3. 系统测试：在这个测试步骤中所发现的往往是需求规格说明的错误。一般来说，系统测试是功能性测试，不是结构性测试。

本次设计进行了如下调试和测试，并提出相应的解决方案：

1. 访问失败

检查浏览器访问的路径是否为 http://127.0.0.1:8097/estate/ ；

1. 前端脚本失效

检查是不是有语法错误导致后面的脚本全被屏蔽了；

1. 数据库连接出错

检查 JDBC 版本；检査数据库服务监听是否成功开启；检查数据库 URL、用户名密码等是否正确；

1. 配置文件加载失败

为 XML 的配置文件有些是 XSD 验证的，所以可能需要联网下载验证规则，此时应该保证网络畅通。其他就是可能配置文件书写的时候出现了语法错误；

1. Jdbctype 和 javatype 以及数据库 Type 问题

需要查询这三种 type 的对应关系，印象最深的就是不同数据库和 jdbctype 对应关系不一样。

# 五、总结

这次框架企业级应用课程设计是一门非常不错的课程，在此之前我写服务器基本都是自己造轮子自己写，但是这次课设使得我有机会使用现在还在使用Spring MVC + Spring + MyBatis架构进行开发，同时，这次实验对我来说是一个巨大的挑战。因为之前只是对SSM框架有所耳闻，原先用 Servlet的时候也是自己从GET和 POST处理函数为入口自己写处理，所以一开始对于SSM的写法还不是很熟练。但是，经过持续约一个月坚持不懈的努力，我终于完成了本次的实验任务。

当然在开发的过程中也遇到了很多问题。老师曾说，对于我们现在接触到的困难，之前肯定有人接触过，所以要好好利用互联网。课程设计，从安装 MySQL，到配置数据库，到使用JDBC进行数据的增删改查，每步都曾遇到很棘手的问题，多亏了老师的帮助，也感谢在社区和论坛上帮我解决问题的朋友们。

采用B/S结构与JDBC框架。动态 WEB 中，程序依然使用客户端和服务端，客户端依然使用浏览器（Chrome、Safari、Firefox 等），通过网络（Network）连接到服务器上，使用 HTTP 协议发起请求（Request），现在的所有请求都先经过一个 WEB Server Plugin（服务器插件）来处理，此插件用于区分是请求的是静态资源（\*.htm 或者是 \*.html）还是动态资源。

当然，这次实验的另外一个体会就是，时间安排非常重要。老师在很早的时候就把任务书发了下来，但是当时我没在意，所以在之后做得非常紧。由此了解到，做好一个项目，每一阶段的时间都必须规划好。因为我们学生现阶段采用开发模式基本都是瀑布开发模式，就必须保证每一步的正确性和及时性。总之，本次设计锻炼了实践能力，増长了我发现问题解决问题的能力，大大丰富了自己的专业知识和技能。由于各种原因，系统的功能还没有得到进一步的完善，以后会继续努力，在今后的学习过程中，不断地提高自己。