

**《软件开发综合实践》课程设计报告**

**题 目 ：** 企业员工管理系统

**专 业 班 级 ：** 软件1702

**学 生 姓 名 ：** 刘文博

**学 号 ：** 201716040224

**指 导 教 师 ：** 王雪涛

**课程设计时间：** 2020.6.15—2019.7.3

**软件工程 专业软件开发综合实践课程设计任务书**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学生姓名** | **刘文博** | **专业班级** | | 软件1702 | **学号** | | **201716040224** |
| **题 目** | 部门员工管理系统 | | | | | | |
| **课题性质** | 工程设计 | | **课题来源** | | | 自拟课题 | |
| **指导教师** | **王雪涛** | | **同组姓名** | | | 无 | |
| **主要内容** | 设计完成一个部门员工管理系统。使用Java语言，与存储在MYSQL数据库中的数据进行交互，恰当地使用集合对数据相应的操作，通过控制台或图形界面拟实现以下功能：   1. 员工信息的录入、修改、删除、查询 2. 员工的签到与签退 3. 员工请假的申请与审批 4. 部门与在职/离职员工的管理 | | | | | | |
| **任务要求** | （1）能够综合运用软件工程的理论知识对实际问题进行分析、设计和实现，在设计实现中能够体现良好的身心素质和团队精神；  （2）撰写课程设计报告，包含需求分析、概要设计、运行环境、开发工具、详细设计、调试分析、测试结果等内容，并对关键部分给出UML图说明，文字、段落和图表格式规范；  （3）在答辩过程中要求具有良好的表达能力，专业的描述方法，并能进行良好的沟通和交流。 | | | | | | |
| **参考文献** | [1]《软件工程-第4版》. Shari Lawrence Pfleeger等. 人民邮电出版社.2019.  [2]《数据库系统概论（第5版）》.王珊 萨师煊. 高等教育出版社.2014  [3] 《基于SSH架构的Web应用开发案例教程》.范新灿.电子工业出版社.2019  [3]《Spring+Spring MVC+MyBatis从零开始学》.吴为胜、杨章伟.清华大学出版社.2019. | | | | | | |
| **审查意见** | **教研室主任签字： 2020年 6月 15日** | | | | | | |

目录

[1 系统分析 1](#_Toc44319668)

[2 系统设计 2](#_Toc44319669)

[3 开发工具和运行环境 3](#_Toc44319670)

[5 详细设计与实现 4](#_Toc44319671)

[5.1数据库设计 4](#_Toc44319672)

[5.2类的设计 7](#_Toc44319673)

[5.2.1 实体类设计 7](#_Toc44319674)

[5.2.2 Dao层类的设计 10](#_Toc44319675)

[5.2.3 Service层类设计 12](#_Toc44319676)

[5.2.4 Controller层类设计 15](#_Toc44319677)

[5.3具体功能实现 18](#_Toc44319678)

[6 系统测试 24](#_Toc44319679)

[7 总结 30](#_Toc44319680)

[8 参考文献 31](#_Toc44319681)

## 1 系统分析

该项目为企业员工管理系统，用来管理员工的各种信息，例如：员工个人信息，员工签到信息，员工请假信息，员工职位变动信息，员工在职与离职信息等，该系统还涉及到权限管理，拥有最大权限的管理员，拥有管理部门员工权限的部门领导，还有管理自己信息的员工。

系统应该在用户输入账号与密码的时候自动进行判断，进行不同主页的跳转，但是不管哪位员工登录，都应该有签到页面还有需要可以查看自己的信息，签到信息，请假信息，加班信息，以及修改自己的信息，这五个最基本的功能就是普通员工锁具有的功能，部门领导，人事部员工/领导拥有比普通员工更高的权限，要在不同的主页中进行不同的展示。以避免员工权限的越界。

部门员工的权限：查看/修改个人信息，查看考勤信息，查看加班信息，申请请假/查看请假记录

部门领导的权限：查看/修改个人信息，查看考勤信息，查看加班信息，审批部门员工的请假申请/查看请假记录

人事部员工的权限：查看/修改个人信息，查看考勤信息，查看加班信息，申请请假/查看请假记录，员工管理，考勤管理

人事部领导的权限：查看/修改个人信息，查看考勤信息，查看加班信息，申请请假/查看请假记录，员工管理，考勤管理，部门管理

## 2 系统设计

综合上述系统分析，总结了员工的不同权限，需要在数据库中设计相应的字段来表示不同的权限，还需要在后台代码中，根据登陆的账号来判断登录者的权限，从而跳转到不同的页面。

将权限信息汇总了如下表格：

部门权限信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 权限  职位 | 个人信息管理 | 员工信息管理 | 考勤管理 | 请假管理 | 部门管理 |
| 部门员工 | 管理 | 无 | 查看 | 申请 | 无 |
| 部门领导 | 管理 | 查看 | 查看 | 审批 | 无 |
| 人事部员工 | 管理 | 管理 | 管理 | 无 | 无 |
| 人事部领导 | 管理 | 管理 | 管理 | 管理 | 管理 |

系统采用Spring + Spring MVC + Mybatis plus 框架完成，Spring框架用来完成类的管理与事务的支持，Spring MVC用来完成控制类与页面跳转的管理，Mybatis plus用来完成数据的获取与持久化操作。前端框架采用Bootstrap框架完成页面的显示。

## 3 开发工具和运行环境

前台开发框架：Bootstrap

后台开发框架：Spring + Spring MVC + Mybatis plus

开发使用IDE：IntelliJ IDEA 2018.3.6 x64

数据库：MYSQL 5.7

数据库开发工具：Navicat Premium 12

测试框架：Junit

服务器及版本：Tomcat 9.0.271

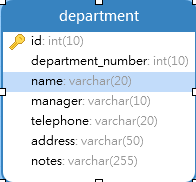
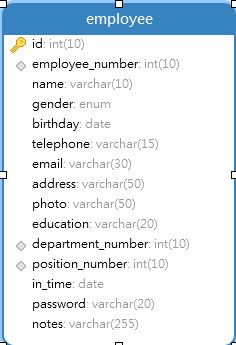
操作系统及版本：Windows 10 专业版 2004

## 5 详细设计与实现

### 5.1数据库设计

企业员工信息管理系统，最主要的就是员工这个属性了，同时员工这个表对应的属性又是非常之多的。

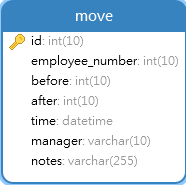
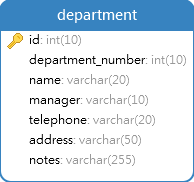
1. 员工表拥有员工编号，姓名，性别，生日，电话号码，电子邮件地址，家庭住址，照片，学历，登录密码，备注，同时还需要拥有一个部门编号（外键）与职位编号（外键）



**图 5.1-1 员工表**   **图 5.1-2 部门表**

2. 部门表中拥有部门编号，部门的名称，部门的负责人，部门电话号码，部门的所在地址，备注等信息。

3. 职位表中拥有职位编号，职位名称，职位等级以及备注信息。

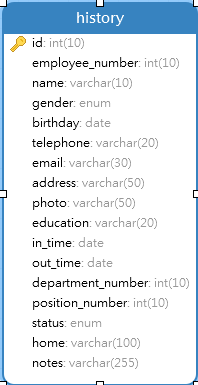
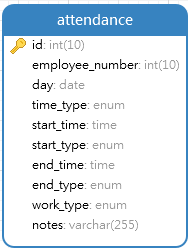


**图 5.1-3 部门表 图 5.1-4 加班表 图 5.1-5 员工调动表**

4. 加班表中拥有员工编号，部门编号，加班日期，开始时间，结束时间，备注等信息。

5. 员工调度表中有员工编号，调度前所属部门，调度后所属部门，调度时间，管理者姓名，备注等信息

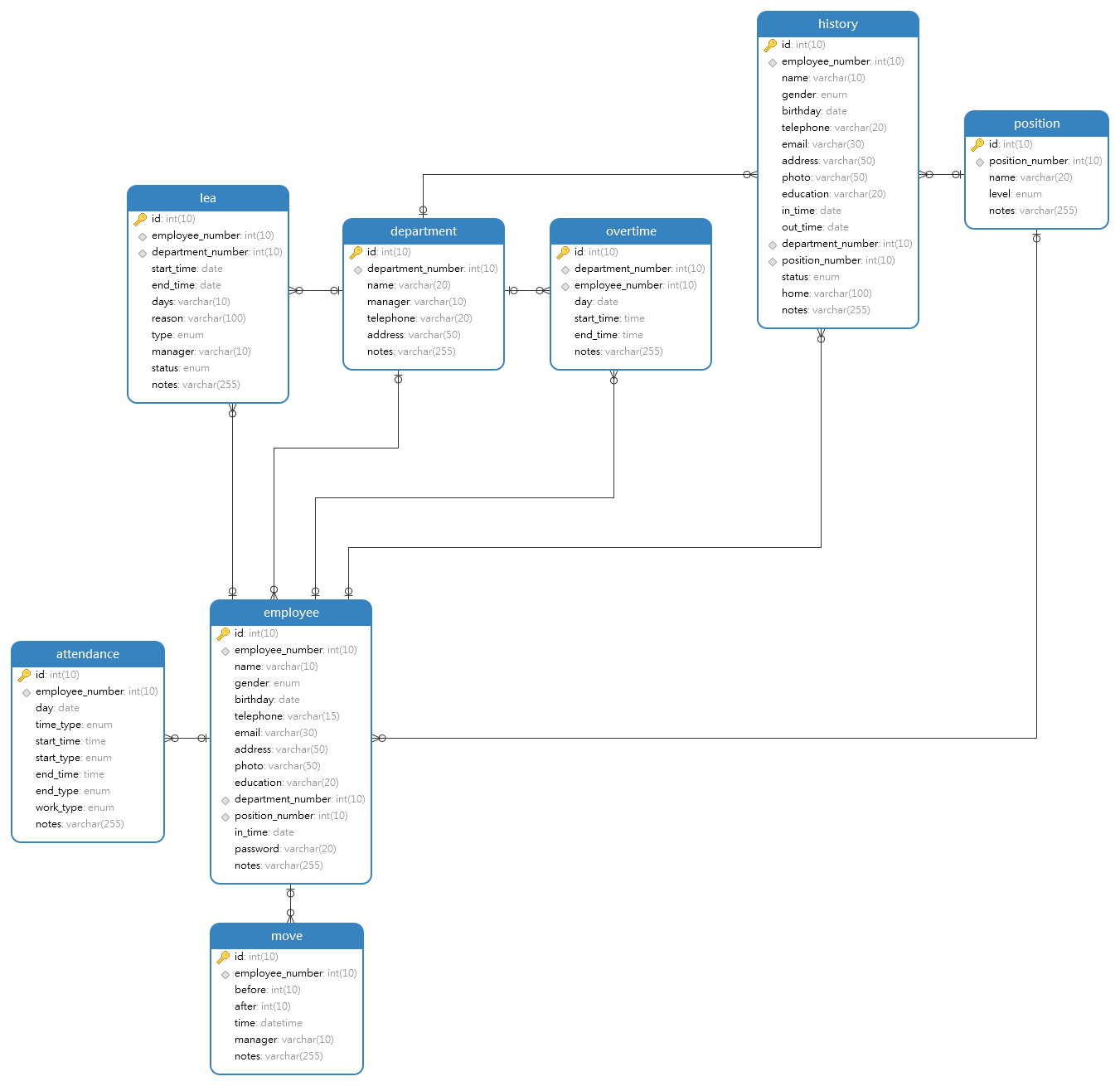
6. 员工档案表拥有员工信息，姓名，性别，电话，电子邮件地址，地址，照片，学历，入职时间，离职时间，部门编号，职位编号，在职状态，家庭住址，备注等信息

**图 5.1-6 员工调动表 图 5.1-7 考勤表**

7. 考勤表拥有员工信息，日期，时间类型（上/下午），开始时间，开始类型（迟到/正常），结束时间，结束类型（正常/早退），工作类型（正常/加班），备注等信息

下面是所有表之间的关系：

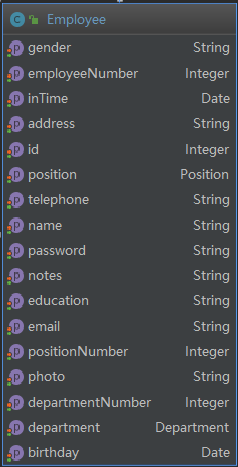
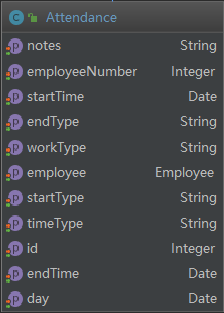
****

**图5.1-1 数据库ER图**

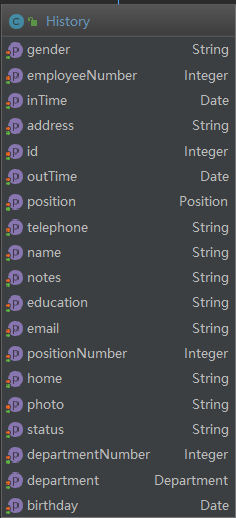
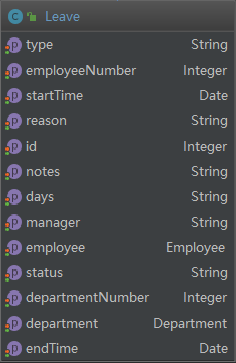
### 5.2类的设计

项目采用MVC的思想，前台的每一个系列的动作都对应一个控制类进行交互，每一个控制类又有一个相应的Service类进行具体功能的实现，在相应的Service类的实现中，每一个实体类都有一个与之对应的Mapper接口类与之对应，进行数据的读取与持久化操作。

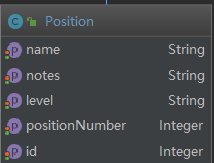
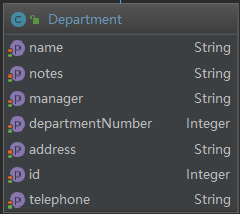
#### 5.2.1 实体类设计

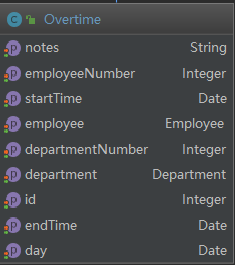
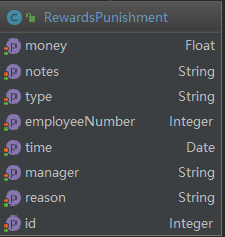
**图5.2.1-1 雇员类类图 图5.2.1-2 签到类类图**

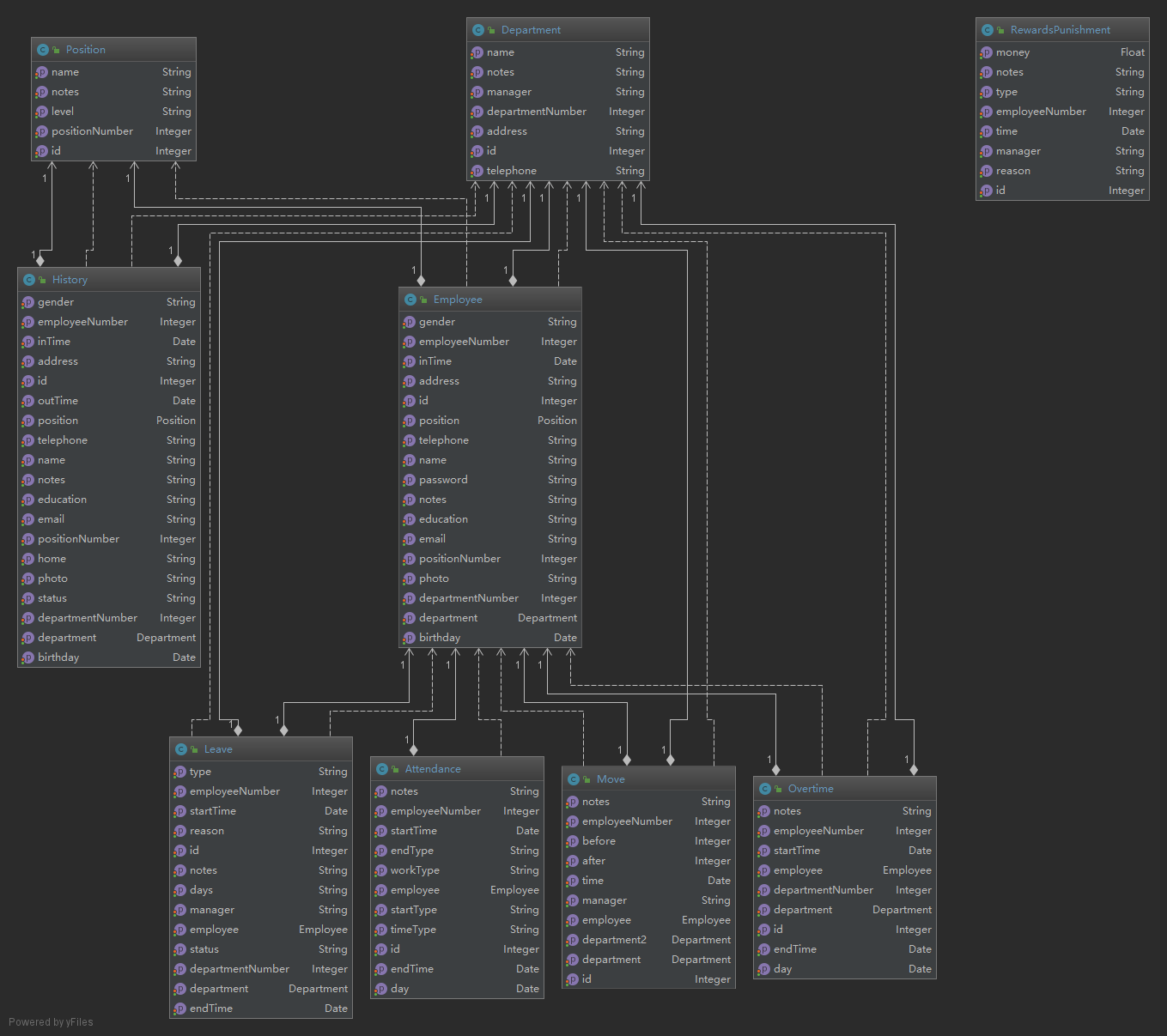
**图5.2.1-3 员工档案类类图 图5.2.1-4 员工请假类类图**

**图5.2.1-5 职位类类图**  **图5.2.1-6 部门类类图**

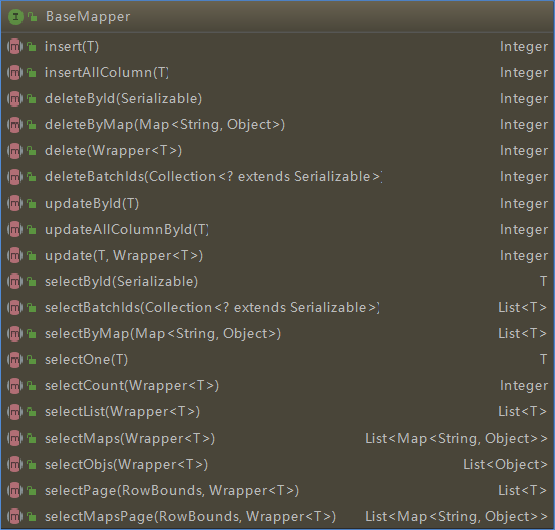
 

**5.2.1-7 加班类类图**  **5.2.1-8 惩奖纪录类类图**

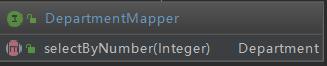
****

**图5.2.1-9 实体类关系图**

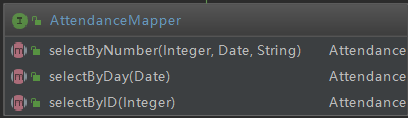
#### 5.2.2 Dao层类的设计



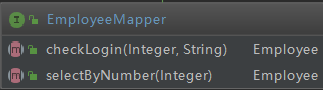
**图5.2.2-1 基类Mapper类BaseMapper类类图**



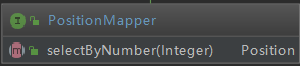
**图5.2.2-2 部门类Mapper类类图**



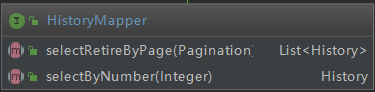
**图5.2.2-3 出勤类Mapper类类图**



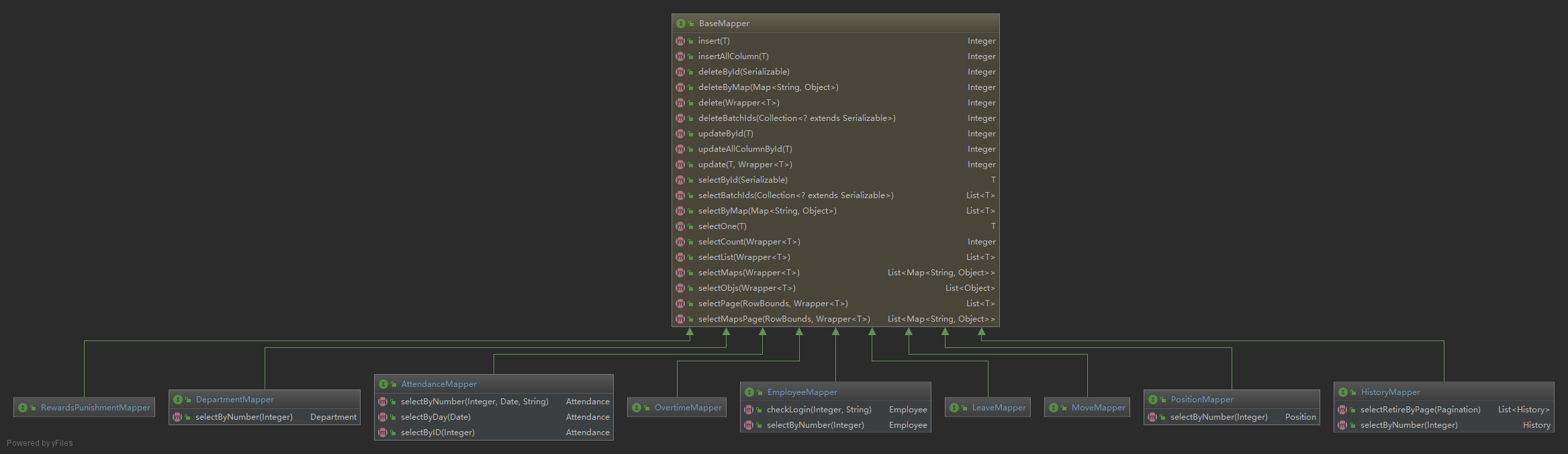
**图5.2.2-4 员工类Mapper类类图**



**图5.2.2-5 职位类Mapper类类图**

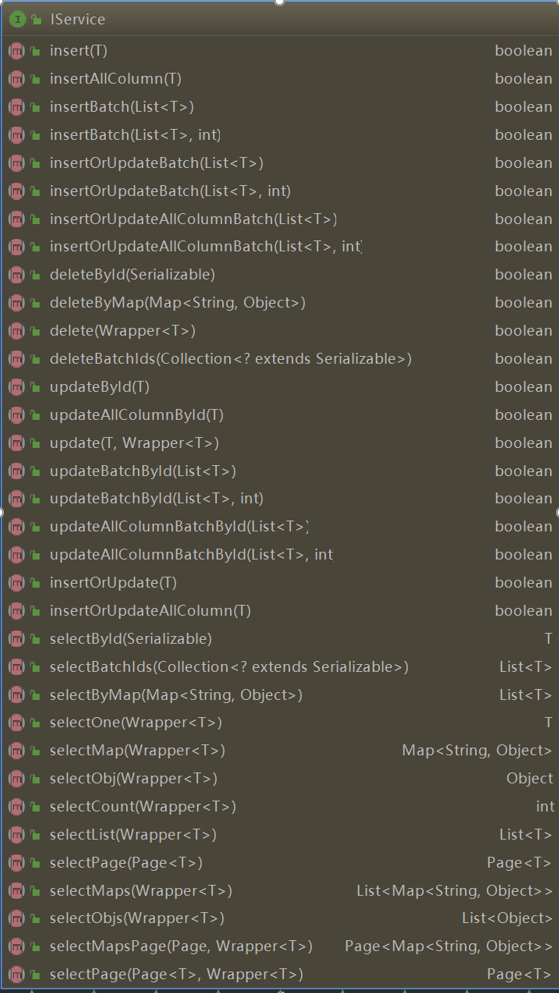


**图5.2.2-6 员工档案类Mapper类类图**

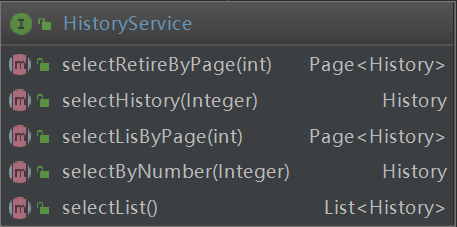
****

**图5.2.2-7 Mapper类关系图**

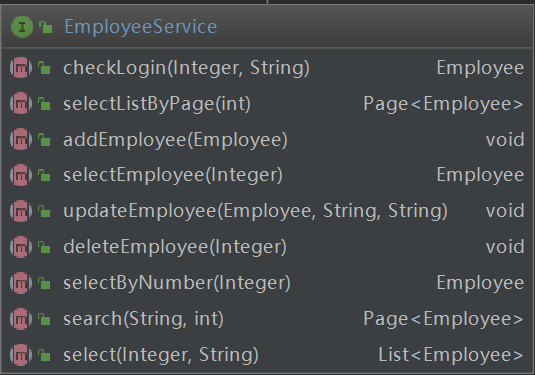
#### 5.2.3 Service层类设计



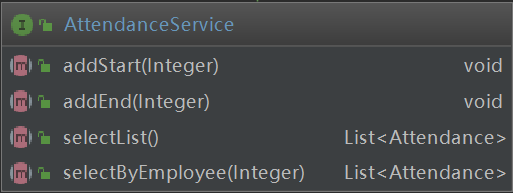
**图5.2.3-1 Service层基类**



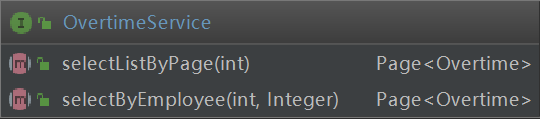
**图5.2.3-2 员工档案Service类类图**



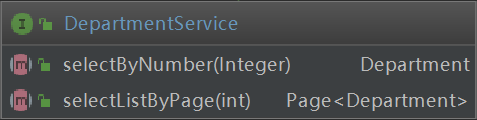
**图5.2.3-3 员工Service类类图**



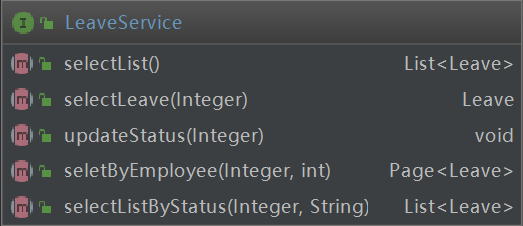
**图5.2.3-4 签到Service类类图**



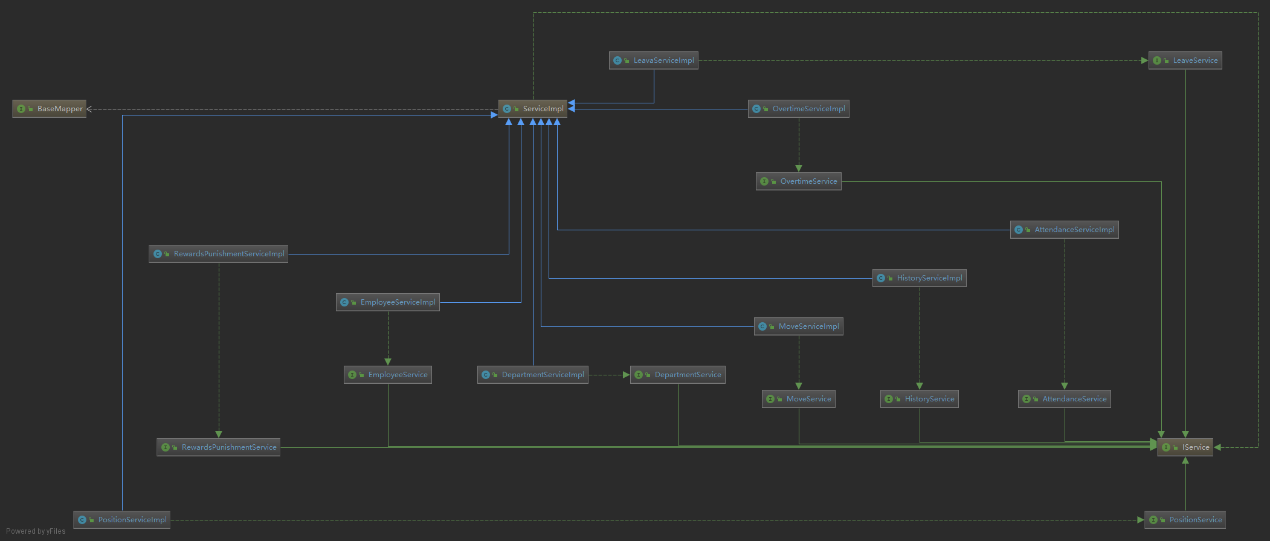
**图5.2.3-5 加班Service类类图**



**图5.2.3-6 部门Service类类图**

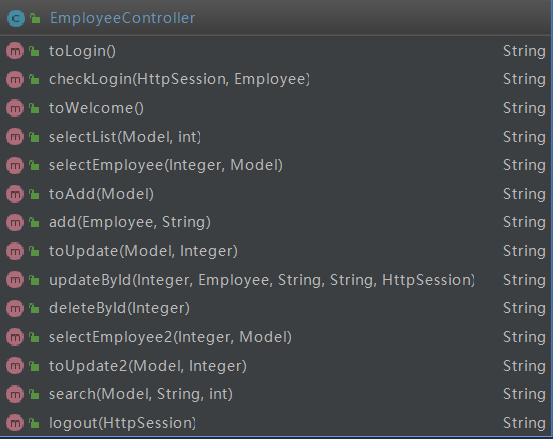


**图5.2.3-7 请假Service类类图**

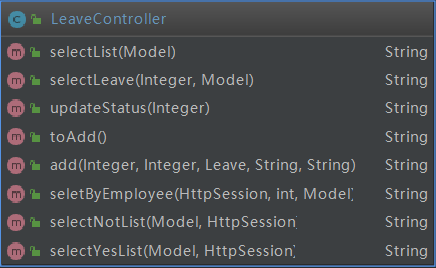
****

**图5.2.3-7 Service层关系图**

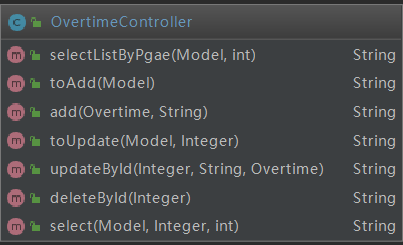
#### 5.2.4 Controller层类设计



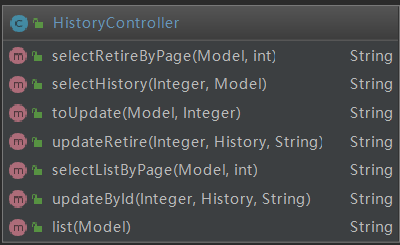
**图 5.2.4-1 员工控制类**



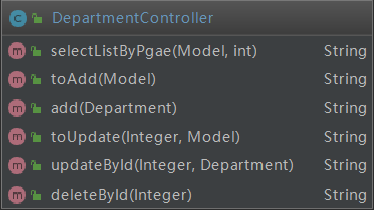
**图 5.2.4-2 请假控制类**



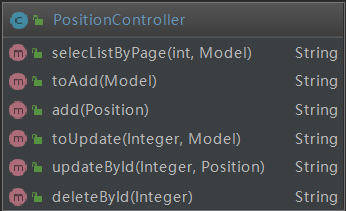
**图 5.2.4-3 加班控制类**



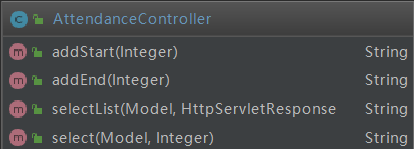
**图 5.2.4-4 员工调度控制类**



**图 5.2.4-5 部门控制类**



**图 5.2.4-6 职位控制类**



**图 5.2.4-7 考勤控制类**

### 5.3具体功能实现

签到功能的实现：



**图5.3-1 签到界面图**

签到功能分为签到与签退，签到功能主要实现思路：员工点击签到，将请求发送到后台，系统立即获取当前时间，并生成一个考勤类，自动装填前半部分的表项，根据已经设定好的上下班时间段判断当前的签到属于正常上班还是迟到，并纪录当前时间段，将数据保存到数据库。签退功能的实现思路：用户点击签退，将请求发送到后台，系统立即获取当前时间系统根据当前所属的时间段（上午/下午/加班）与当前的日期作为条件，从数据库中读取之前的签到记录，再根据当前时间与规定的时间相比，判断是早退还是正常下班，将表单内容补充完整，将信息更新到数据库，完成签退的操作。

相关代码如下：

|  |
| --- |
| /\*\*签到与签退的控制层相关代码 \*\*/  @RequestMapping("/addStart.do")  public String addStart(Integer employeeNumber){  attendanceService.addStart(employeeNumber);  return "welcome";  }    @RequestMapping("/addEnd.do")  public String addEnd(Integer employeeNumber){  attendanceService.addEnd(employeeNumber);  return "welcome";  } |

|  |
| --- |
| /\*\* Service层的签到与签退的相关代码 \*\*/  @Override  public void addStart(Integer employeeNumber){  //获取当前时间  Date nowTime = MTimeUtil.nowTime();  //获取当前日期  Date nowDate = MTimeUtil.nowDate();  Attendance attendance = new Attendance();  attendance.setEmployeeNumber(employeeNumber);  attendance.setDay(nowDate);  attendance.setStartTime(nowTime);  if (nowTime.after(amTime) && nowTime.before(amEndTime)) {  Attendance att = baseMapper.selectByNumber(employeeNumber, nowDate, "上午");  if (att == null) {  attendance.setTimeType("上午");  if (nowTime.before(amStartTime)) {  attendance.setStartType("正常");  }else{  attendance.setStartType("迟到");  }  baseMapper.insert(attendance);  }  }else if(nowTime.after(pmTime) && nowTime.before(pmEndTime)){  Attendance att = baseMapper.selectByNumber(employeeNumber, nowDate, "下午");  if (att == null) {  attendance.setTimeType("下午");  if (nowTime.before(pmStartTime)) {  attendance.setStartType("正常");  }else{  attendance.setStartType("迟到");  }  baseMapper.insert(attendance);  }  }else if(nowTime.after(ovTime) && nowTime.before(ovEndTime)){  Attendance att = baseMapper.selectByNumber(employeeNumber, nowDate, "加班");  if (att == null) {  attendance.setTimeType("加班");  if (nowTime.before(ovStartTime)) {  attendance.setStartType("正常");  }else{  attendance.setStartType("迟到");  }  baseMapper.insert(attendance);  }  }  }  @Override  public void addEnd(Integer employeeNumber) {  Date nowTime = MTimeUtil.nowTime();  Date nowDate = MTimeUtil.nowDate();  if (nowTime.after(amStartTime) && nowTime.before(pmStartTime)) {  Attendance attendance = baseMapper.selectByNumber(employeeNumber, nowDate, "上午");  if (attendance.getEndTime() == null) {  attendance.setEndTime(nowTime);  if (nowTime.after(amEndTime)) {  attendance.setEndType("正常");  }else{  attendance.setEndType("早退");  }  baseMapper.updateById(attendance);  }  }else if(nowTime.after(pmStartTime) && nowTime.before(ovStartTime)){  Attendance attendance = baseMapper.selectByNumber(employeeNumber, nowDate, "下午");  if (attendance.getEndTime() == null) {  attendance.setEndTime(nowTime);  if (nowTime.after(pmEndTime)) {  attendance.setEndType("正常");  }else{  attendance.setEndType("早退");  }  baseMapper.updateById(attendance);  }  }else if(nowTime.after(ovStartTime)){  Attendance attendance = baseMapper.selectByNumber(employeeNumber, nowDate, "加班");  if (attendance.getEndTime() == null) {  attendance.setEndTime(nowTime);  if (nowTime.after(ovEndTime)) {  attendance.setEndType("正常");  }else{  attendance.setEndType("早退");  }  baseMapper.updateById(attendance);  }  }  } |

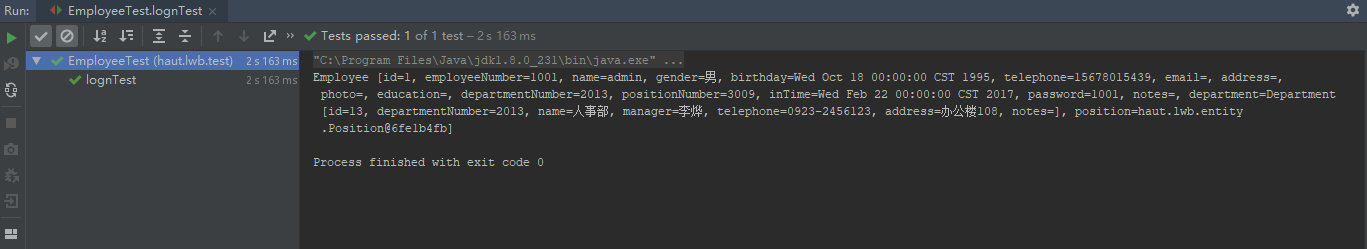
|  |
| --- |
| /\*\* 签到签退相关的Mapper层的代码 \*\*/  /\*\*  \* 根据employeeNumber和day查询记录  \* @param employeeNumber  \* @return  \*/  Attendance selectByNumber(@Param("employeeNumber")Integer employeeNumber,  @Param("day")Date day, @Param("timeType")String timeType);  Attendance selectByDay(@Param("day")Date day);  Attendance selectByID(@Param("employeeNumber")Integer employeeNumber);  <select id="selectByNumber" resultType="Attendance">  select  <include refid="Base\_Column\_List" />  from attendance  where employee\_number = #{employeeNumber}  and day = #{day} and time\_type = #{timeType}  </select> |

## 6 系统测试

1. **单元测试**

1. 员工的实体类，Mapper层，Service层，还有Controller层的代码已经完成，数据库表的设计以及数据的初始化工作也已经完成，下面模拟用户登录，测试各个层之间的信息与数据的交流，以及是否可以正确的从数据库中提取数据，在单元测试中模拟用户编号为1001的用户进行登录，查看代码的运行情况。

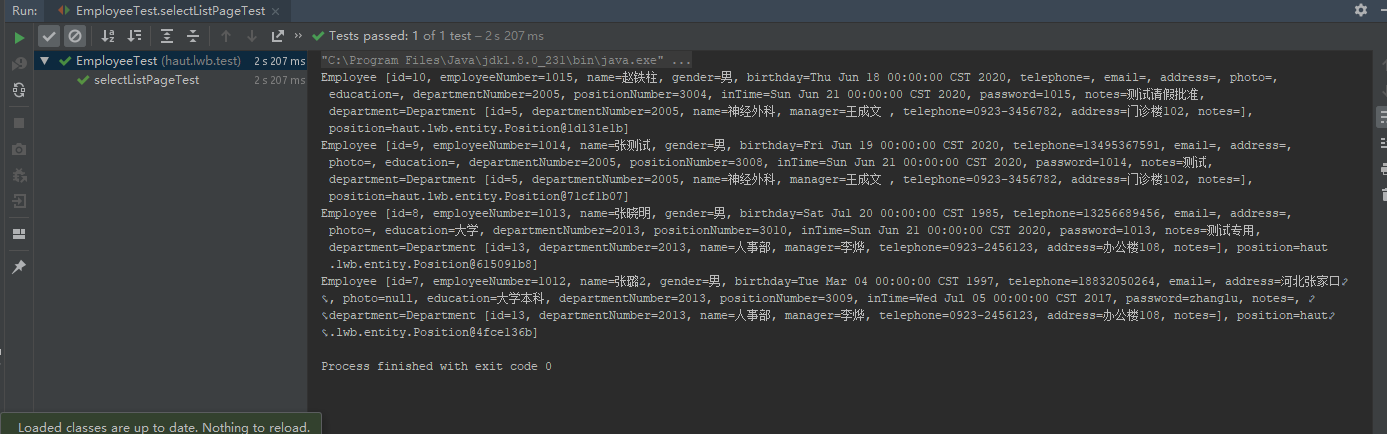
|  |
| --- |
| @SuppressWarnings("resource")  @Test  public void lognTest(){  String xmlPath="spring/spring.xml";  ApplicationContext context = new ClassPathXmlApplicationContext(xmlPath);  EmployeeService eService = (EmployeeService) context.getBean("employeeService");  Employee employee = eService.checkLogin(1001, "1001");  System.out.println(employee.toString());  } |



**图 6.1-1 测试用户登录图**

2. 测试分页查询 使用Mybatis plus的分页插件实现分页查询，减少对系统资源以及处理机的过度占用。

|  |
| --- |
| @SuppressWarnings("resource")  @Test  public void selectListPageTest(){  String xmlPath="spring/spring.xml";  ApplicationContext context = new ClassPathXmlApplicationContext(xmlPath);  EmployeeService eService = (EmployeeService) context.getBean("employeeService");  Page<Employee> page = eService.selectListByPage(1);  List<Employee> eList = page.getRecords();  for(Employee e : eList){  System.out.println(e.toString());  }  } |



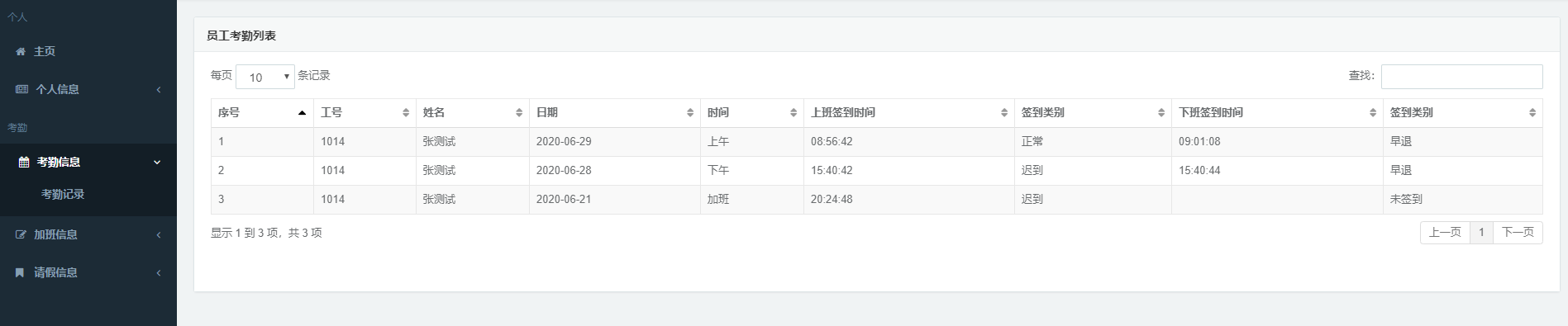
**图 6.1-1 测试分页查询图**

**（2）签到/签退用例测试**

|  |
| --- |
| **/\*上下班时间\*/**  **public static String AMStartTime = "9:00:00";**  **public static String AMEndTime = "11:59:59";**  **public static String PMStartTime = "15:00:00";**  **public static String PMEndTime = "18:00:00";**  **public static String OVStartTime = "20:00:00";**  **public static String OVEndTime = "22:00:00";**    **/\*上午、下午、加班分割时间\*/**  **public static String AMTime = "8:00:00";**  **public static String PMTime = "14:00:00";**  **public static String OVTime = "19:00:00";** |

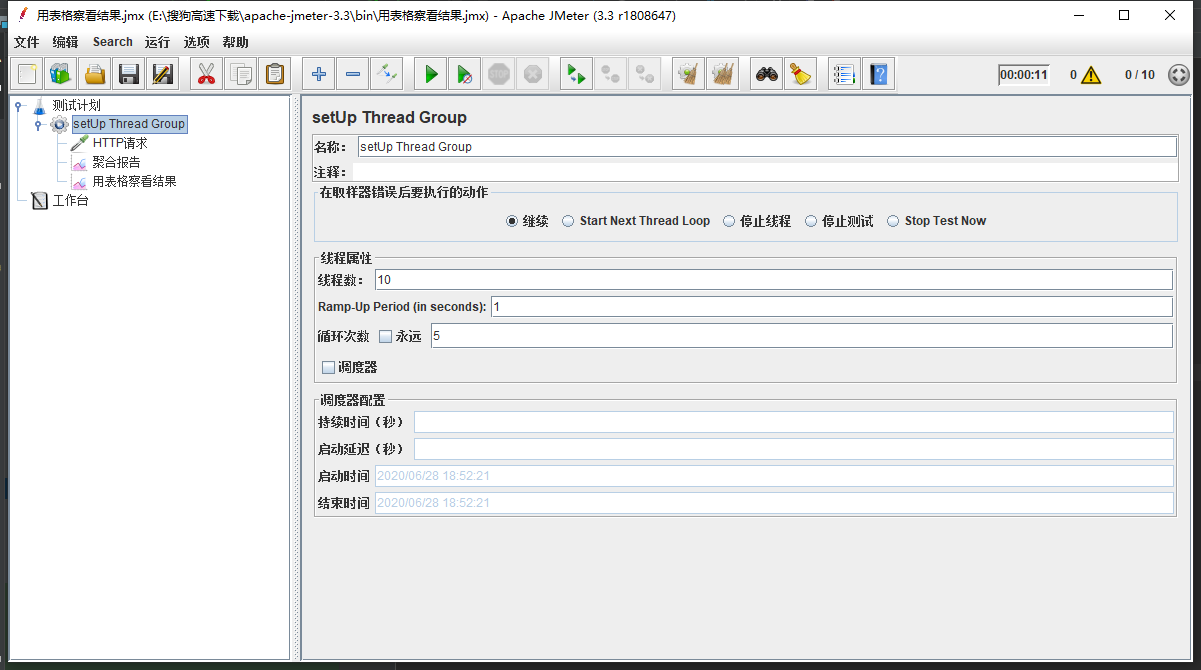
**系统签到/签退功能的测试**

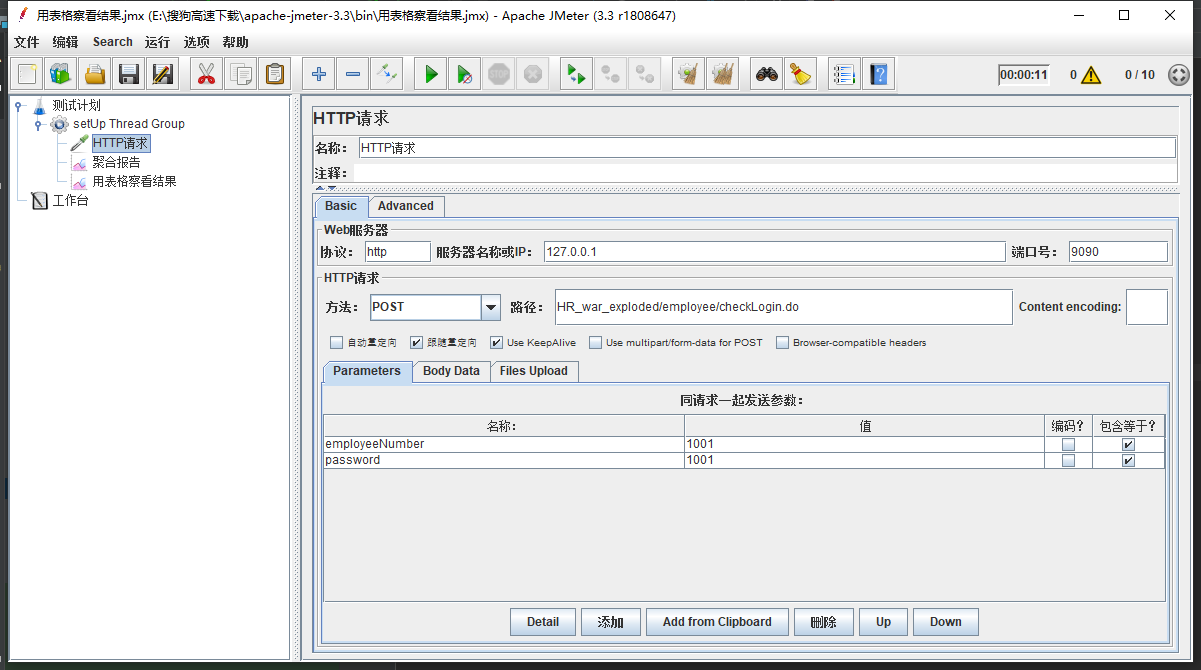
|  |  |  |  |
| --- | --- | --- | --- |
| 签到时间 | 预期输出 | 实际输出 | 异常情况 |
| 上午 10:01 | 迟到 | 迟到 | 无 |
| 下午 15:40 | 迟到 | 迟到 | 无 |
| 上午 8:56 | 正常 | 正常 | 无 |
| 签退时间 |  |  |  |
| 上午 11:36 | 早退 | 早退 | 无 |
| 下午 15:41 | 早退 | 早退 | 无 |
| 上午 9:01 | 早退 | 早退 | 无 |

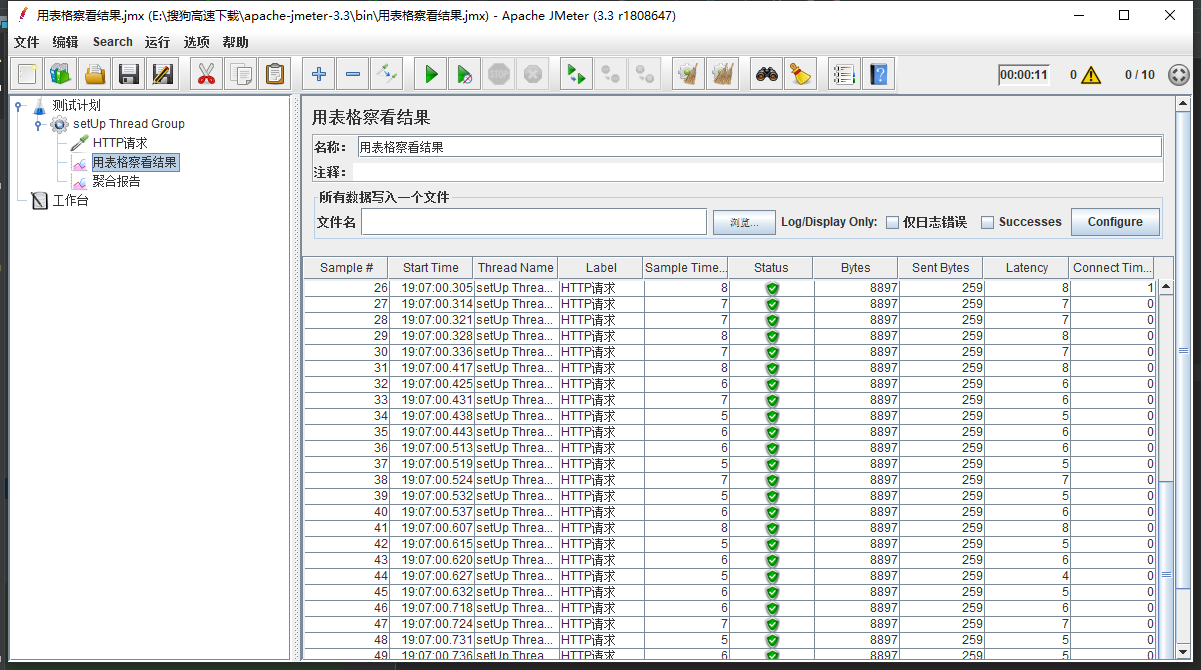


**图 6.2-1 签到用例测试图**

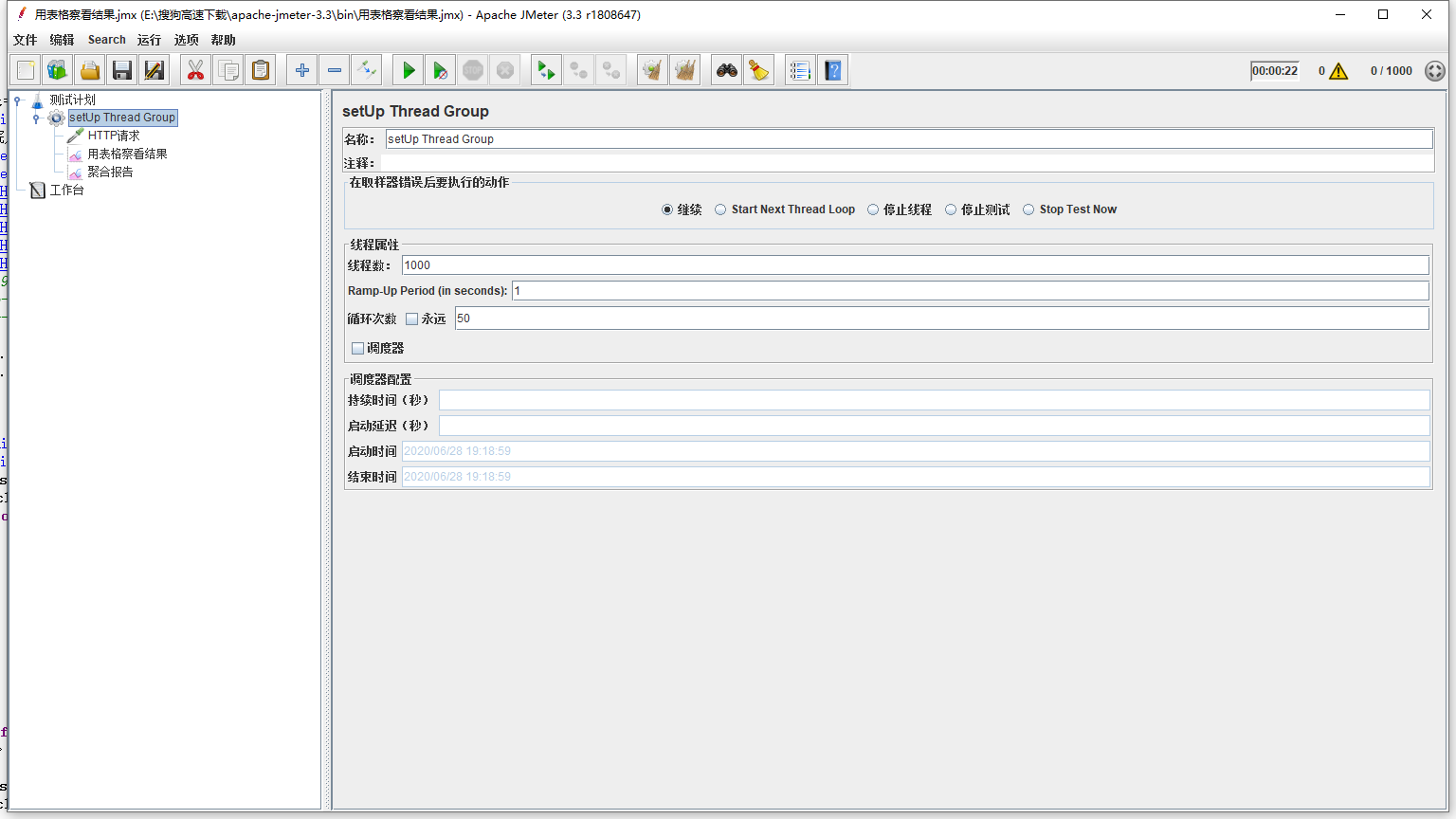
**（3）压力测试**

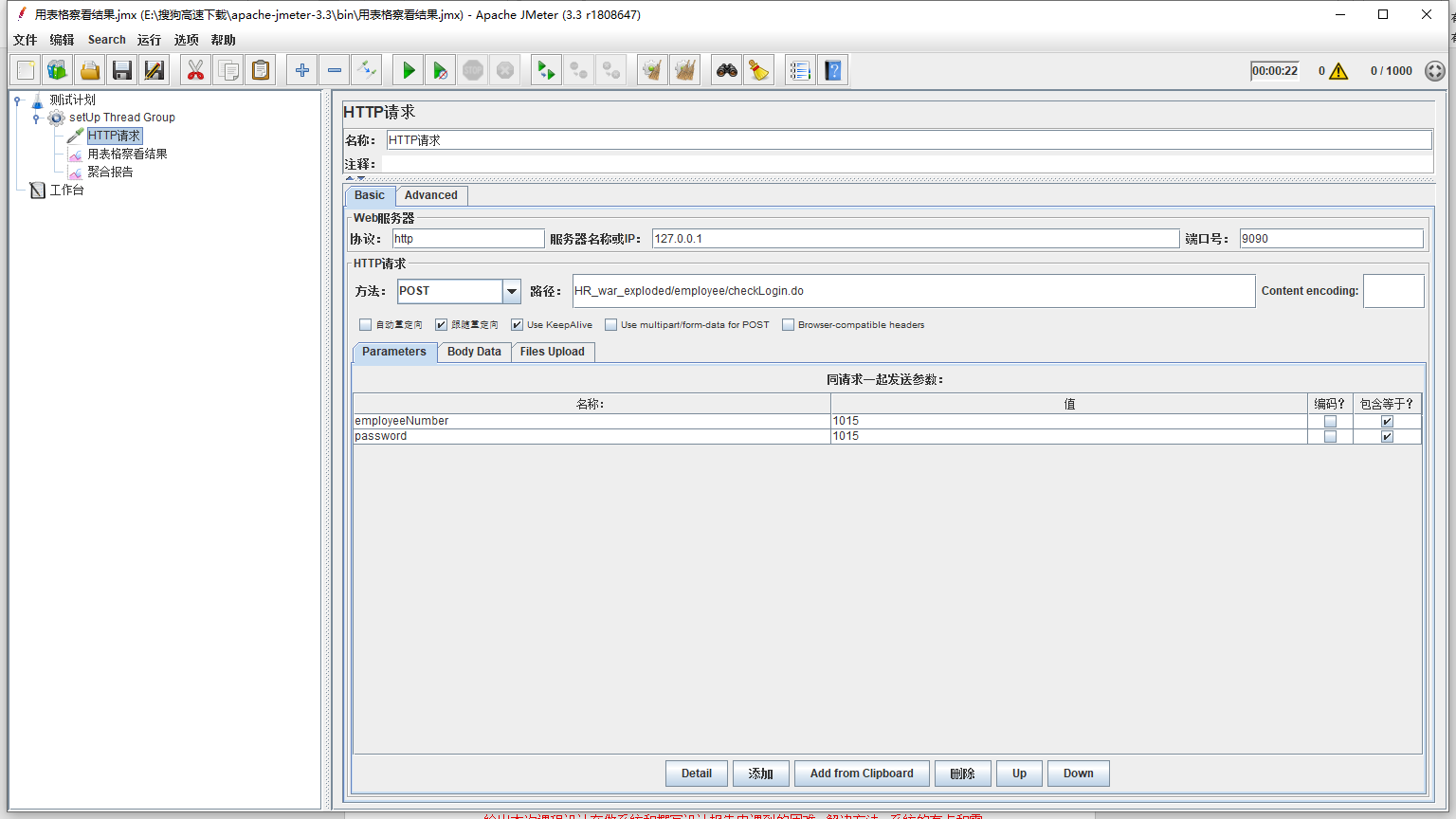


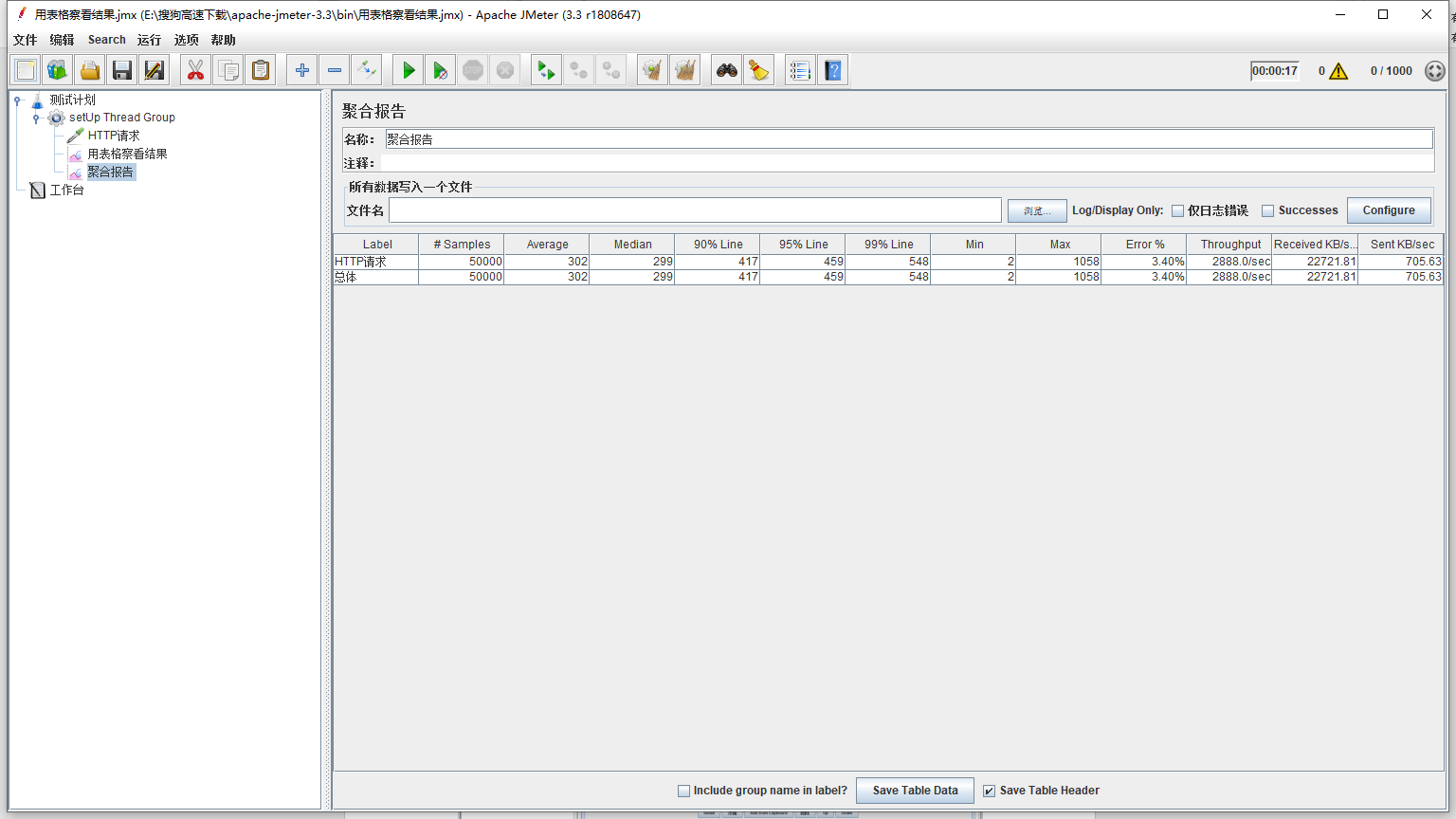




**设置50个并发，全部访问成功，没有宕机。**







**设置了1000\*50=500000次并发访问，出现了3.4%的访问出现了问题，单个服务器理论上无法支持这么多的并发访问，猜测是Mybatis的缓存机制，由于每次登陆的都是相同的账号，使得程序不需要每次都访问服务器，只需要访问缓存就可以进行登录，才使得这么大的并发量，服务器也没有宕机。**

## 7 总结

在学习使用mybatis plus的过程中，在配置文件设置的时候，出现了一些小问题，导致浪费了很长的时间。当配置完成，对照网上的代码进行仿照查询语句的时候，mybatis plus相对于mybatis的优势就显现了出来，mybatis plus的sql语句的编写量大大下降，可以根据框架提供的BaseMapper类进行大多数的增删查改操作，根据函数名字就可以进行简单地查询等操作，跟Spring data JPA有些相似。在进行员工信息查询的时候，框架也可以进行对象导航查询，极大地简便了查询的难度，减少了sql语句的编写量。在请假审批的过程设计中，最开始没有考虑到部门领导审批该部门的员工的请假申请，导致权限越界，在测试中发现并改正了。在用户登录的时候，权限的管理是通过简单的判断语句实现的，这么做安全性很低，在日后学习有关安全的权限框架的时候，可以对项目进行适当地改进，使安全性得到提升。

## 8 参考文献

[1]《软件工程-第4版》. Shari Lawrence Pfleeger等. 人民邮电出版社.2019.

[2]《数据库系统概论（第5版）》.王珊 萨师煊. 高等教育出版社.2014

[3] 《基于SSH架构的Web应用开发案例教程》.范新灿.电子工业出版社.2019

[4]《Spring+Spring MVC+MyBatis从零开始学》.吴为胜、杨章伟.清华大学出版社.2019.

[5] 张宇,王映辉,张翔南．基于Spring的MVC框架设计与实现[J]．计算机工程．2010(04)．

[6] 徐雯,高建华．基于Spring MVC及MyBatis的Web应用框架研究[J]．微型电脑应用. 2012(07)