武汉纺织大学

Web应用开发课程设计

**基于SSM框架的图书馆管理系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11801**

**姓 名： 张玉清**

**学 号： 1804280411**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月20日**

目 录

[1 需求分析 1](#_Toc60601694)

[1.1 主页未登录 1](#_Toc60601695)

[1.2 用户注册 1](#_Toc60601696)

[1.3 用户登录 2](#_Toc60601697)

[1.4功能介绍 2](#_Toc60601698)

[1.4.1 书籍列表 2](#_Toc60601699)

[1.4.2 借书记录 4](#_Toc60601700)

[1.4.3 个人中心 4](#_Toc60601701)

[1.5 整体跳转 5](#_Toc60601702)

[2 系统设计 6](#_Toc60601703)

[2.1用户用例图 6](#_Toc60601704)

[2.2 ER图 6](#_Toc60601705)

[2.3 UML类图（Class Diagram） 7](#_Toc60601706)

[2.4 整体流程图 8](#_Toc60601707)

[3 系统实现 8](#_Toc60601708)

[3.1 项目结构 8](#_Toc60601709)

[3.2 配置文件 9](#_Toc60601710)

[3.2.1 mybatis/sqlMapperConfig.xml 9](#_Toc60601711)

[3.2.2 spring/applicationContext.xml 10](#_Toc60601712)

[3.2.3 springmvc/applicationContext-mvc.xml 13](#_Toc60601713)

[3.3 dao类 14](#_Toc60601714)

[3.4 DAO接口类 16](#_Toc60601715)

[3.5 接口映射文件 17](#_Toc60601716)

[3.6 service层 17](#_Toc60601717)

[3.8 controller层 17](#_Toc60601718)

[4 系统总结 27](#_Toc60601719)

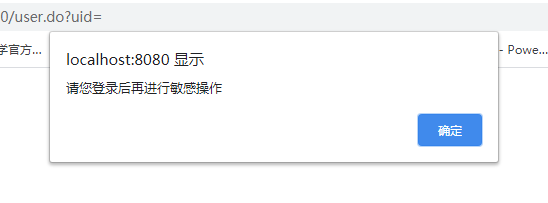
# 1 需求分析

设计一个图书馆管理系统，实现“用户登陆”、“用户注册”、“用户更改账户信息”、“用户借书”、“用户还书”、“图书馆书籍查询”，“借阅记录查询”等功能，具体要求如下：

## 1.1 主页未登录

用户未登录时，主页面提示“登录”，“注册”，不可进入“个人中心页面”，但可以正常的进入“书籍列表”搜索图书馆现有书籍，并查阅“借书记录”。

未登录时强行进入登录页面将会出现警告信息，并跳回主页提示登录。



## 1.2 用户注册

运行程序，根据<http://localhost:8080/index.do> 进入主页面，点击主页右上侧注册按钮进入注册页面。



在注册页面根据提示信息输入学号，用户名，密码。学号一经注册不可修改，且学号唯一。注册成功后自动跳转去主页，可自行选择是否登录，或其他操作。



## 1.3 用户登录

运行程序，根据<http://localhost:8080/index.do> 进入主页面，点击主页右上侧登录按钮进入登录页面

在登录页面根据学号和密码进行登录，学号和密码不能为空，为空将出现警告信息。登录成功后跳转到主页。登录失败将提示登录失败，并刷新登录页面。



## 1.4功能介绍

登录后主页显示学号对应的用户名。并可进行“借书”，“还书”，“用户中心”等操作。



### 1.4.1 书籍列表

用户点击书籍列表进入图书馆书籍管理页面，根据是否登录显示不同提示语。

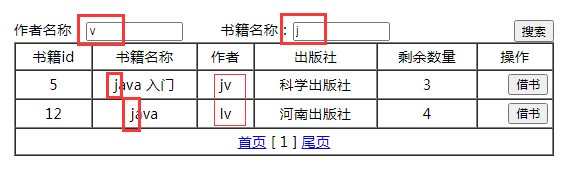
#### 1.4.1.1 分页

书籍列表进行分页显示，每页五条数据，可进行上下页操作与模糊查询。



#### 1.4.1.2 模糊查询

根据用户输入的作者名称及书籍名称，模糊查询数据库里包含用户所输入的模糊查询条件信息，依旧分页显示在表格中。当用户进行上一下下一页操作时，模糊查询条件会进行数据回显，依旧会显示在输入框中。

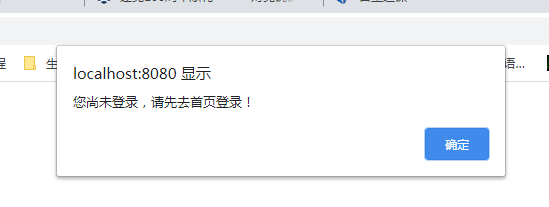


#### 1.4.1.3 未登录提示

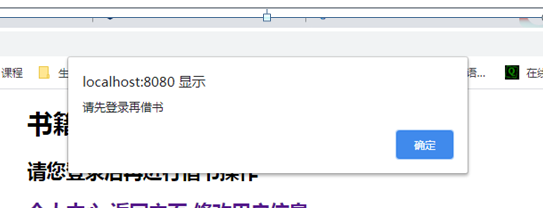
未登录状态下提示登录，否则不可借书，并且不可进入个人中心与修改用户信息。



强行点击进入个人中心或修改用户信息则进行错误提示，并跳转回主页。



未登录下禁止借书操作



#### 1.4.1.4 登录后借书

用户登录状态下给出欢迎语，并可进行借书操作，还可进入个人中心或者修改用户信息。

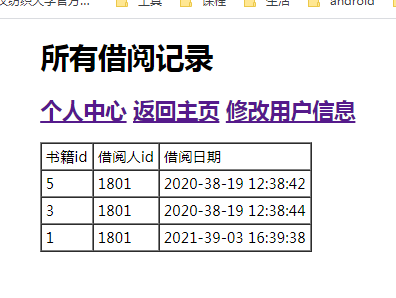


点击借书按钮即可借书，借书后仍在书籍管理页面，但点击借书的书籍总数目减一。同时在后台增加一条用户借书信息。



### 1.4.2 借书记录

借书记录页面不区分用户是否登录，进入便可以查看记录。但用户登录时可进入“个人中心”，“修改用户信息”页面。若未登录则不可进入，强行进入将出现1.4.1.3中对应错误提示。



### 1.4.3 个人中心

登录后进入页面如下图，显示用户名对应的欢迎语，用户当前已借书籍。可进行还书操作。



#### 1.4.3.1 还书

用户点击还书按钮进行还书操作，还书后仍在此页面，但所借书籍清单进行刷新。

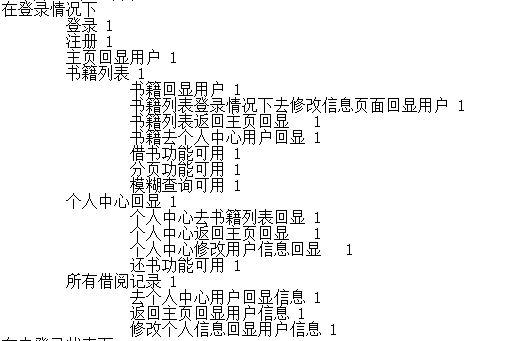
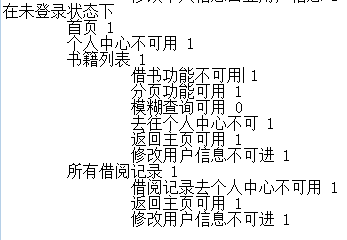


#### 1.4.3.2 修改用户信息

当点击“修改用户信息”时跳转到修改页面，首先进行用户当前信息的回显，用户学号不可变更，其余信息可以更改，两次密码要输入一致，不然会出现警告信息。修改成功后会自动退出登录，并跳转到主页。



## 1.5 整体跳转

# 2 系统设计

## 2.1用户用例图

系统分为登录与未登录两种状态，用例如下图2-1所示：

用户注册

用户登录

个人中心

还书

改信息

借阅记录

改信息

书籍列表

借书

分页

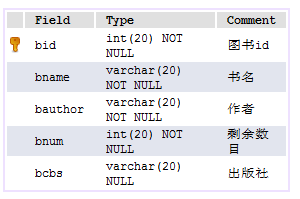
模糊查询

改信息

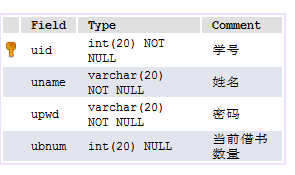
图2-1 用例图

## 2.2 ER图

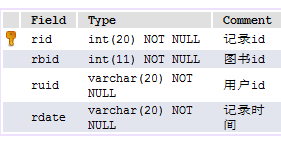
数据库设计3张表，用户表，书籍表，借书记录表，对应的ER图如图3-1所示：



书籍表books



用户表users



记录表recordss

1-n

1-n

图3-1 数据库ER图

## 2.3 UML类图（Class Diagram）

对于用户功能模块，共设计如下6个类。

* 数据库连接类DatabaseConnection：连接数据库并获取连接对象，关闭连接对象。
* VO实体类User：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录
* DAO接口IUserDAO：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。
* DAO实现类UserDAOImpl：DAO接口的真实实现类，主要完成具体数据库操作，但不负责数据库的打开和关闭。
* DAO工厂类DAOFactory：通过工厂类取得一个DAO的实例化对象
* 业务逻辑实现类UserService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）
* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*：

各类的结构及类之间的关系如图2-5所示：

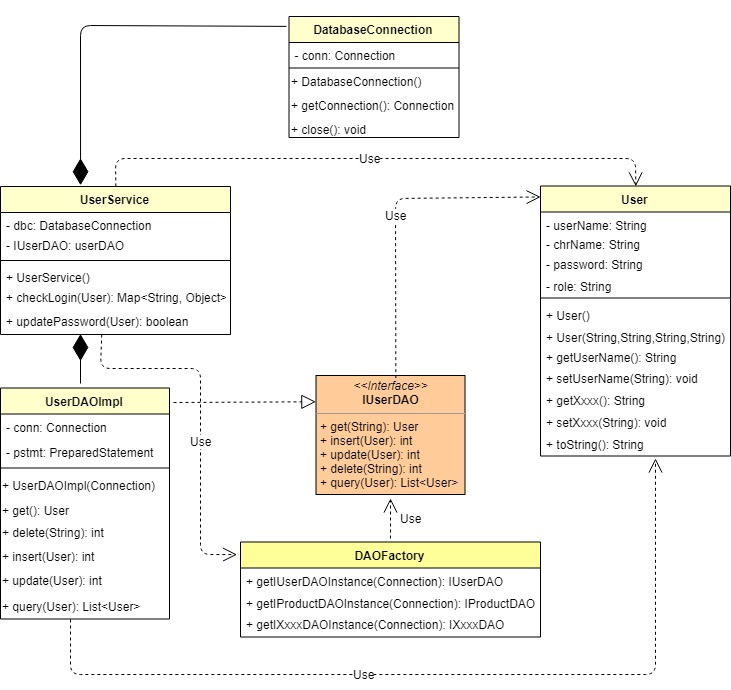
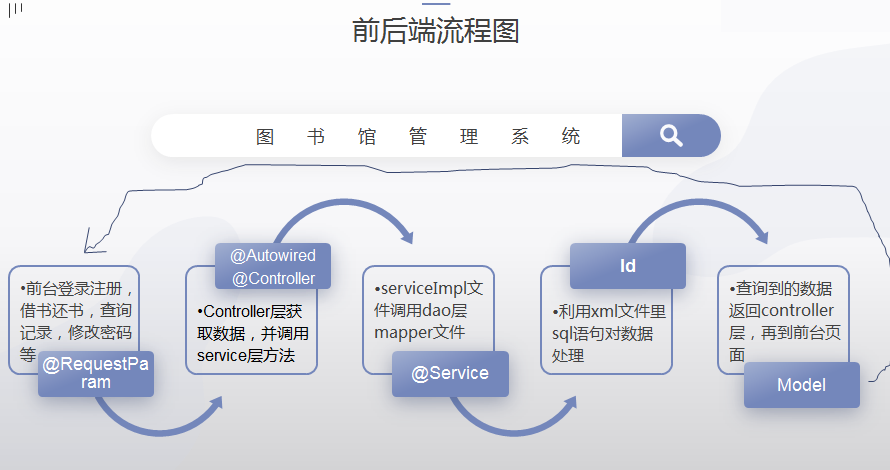


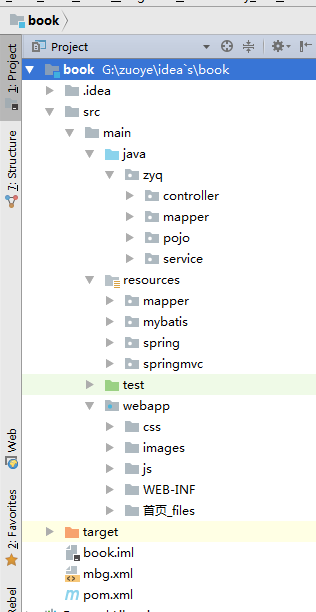
图2-5 用户功能类图

## 2.4 整体流程图

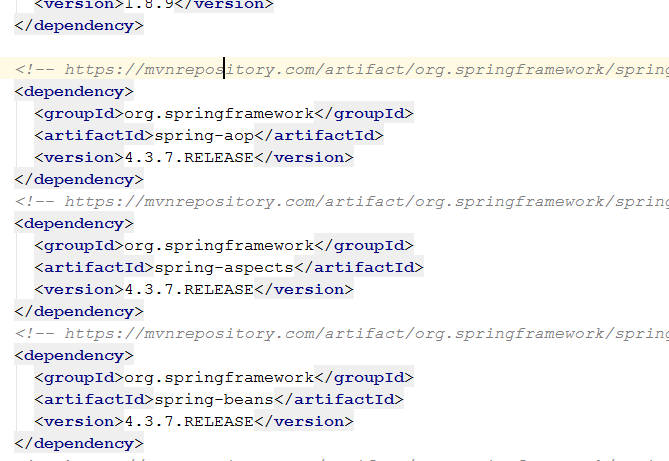


# 3 系统实现

## 3.1 项目结构

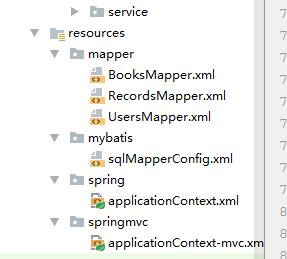


pom.xml配置插件和引入maven依赖，取部分如下图：



## 3.2 配置文件

新建resources文件夹，设置为源文件夹



### 3.2.1 mybatis/sqlMapperConfig.xml

配置信息如下：

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  *<!--配置信息-->  <!--设置sql打印语句-->* <**settings**>  <**setting name="logImpl" value="STDOUT\_LOGGING"** />  </**settings**>  *<!--设置类型别名库，type表示类的类地址，alias表示别名-->* <**typeAliases**>  *<!--以包名的的形式进行扫描别名，别名叫实体类的类名-->* <**package name="zyq.pojo"**/>  </**typeAliases**>   *<!--分页插件-->* <**plugins**>  <**plugin interceptor="com.github.pagehelper.PageInterceptor"**>  <**property name="reasonable" value="true"**></**property**>  </**plugin**>  </**plugins**>  </**configuration**> |

### 3.2.2 spring/applicationContext.xml

描述该文件作用，并对里面的配置进行解释

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"**>    *<!-- 1.开启spring注解驱动 -->* <**context:component-scan base-package="zyq.service"**></**context:component-scan**>  *<!--2.读取properties配置文件-->  <!-- <context:property-placeholder location="classpath:properties/db.properties"/>-->   <!-- 3.连接数据库 连接池是c3p0-->* <**bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"** >  <**property name="driverClass" value="com.mysql.jdbc.Driver"**/>  *<!-- 配置Jdbc的Url -->* <**property name="jdbcUrl" value="jdbc:mysql://localhost:3306/book?characterEncoding=utf-8"**/>  *<!-- 配置用户名 -->* <**property name="user" value="root"**/>  *<!-- 密码 -->* <**property name="password" value="root"**/>  </**bean**>   *<!-- 4.配置事务管理器 -->* <**bean id="transactionManager"  class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>  *<!-- 事务管理数据库连接池 -->* <**property name="dataSource" ref="dataSource"**></**property**>  </**bean**>   *<!--5.配置注解版事务开启注解-->* <**tx:annotation-driven transaction-manager="transactionManager"**/>   *<!--6.spring管理mybatis配置文件-->* <**bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>  *<!--dataSource属性指定要用到的连接池 ref映射的是id=dataSource是bean的标签 读取连接数据库信息 -->* <**property name="dataSource" ref="dataSource"**></**property**>  *<!--configuration属性指定mybatis核心配置文件 管理mybatis的配置文件-->* <**property name="configLocation" value="classpath:mybatis/sqlMapperConfig.xml"**/>  *<!--所有配置的mapper文件，mybatis的实体类配置文件所有的sql映射文件-->* <**property name="mapperLocations" value="classpath:mapper/\*.xml"**></**property**>  </**bean**>  *<!--7.spring管理mybatis映射接口和sql映射文件之间关联关系-->* <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  <**property name="basePackage" value="zyq.mapper"**/>  <**property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"**/>  </**bean**>   *<!--文件上传的配置-->* <**bean id="multipartResolver"  class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>  <**property name="maxUploadSize" value="500000"**/>  <**property name="defaultEncoding" value="UTF-8"**/>  </**bean**> </**beans**> |

### 3.2.3 springmvc/applicationContext-mvc.xml

描述该文件作用，并对里面的配置进行解释

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd"**>  *<!-- <context:property-placeholder location="classpath:properties/db.properties"/>-->   <!-- 开启注解驱动 -->* <**context:component-scan base-package="zyq.controller"**></**context:component-scan**>   *<!-- 开启mvc注解驱动 -->* <**mvc:annotation-driven**/>  <**mvc:default-servlet-handler**/>  *<!-- 视图解析器  如果程序中的css,js,html文件包放在webapp目录下，那么可以不用设置试图解析器  反之如果放在WEB-INF下，则需要试图解析器配置-->  <!-- <mvc:resources mapping="/css/\*\*" location="/css/" />  <mvc:resources mapping="/js/\*\*" location="/js/" />  <mvc:resources mapping="/jsp/\*\*" location="/jsp/" />  <mvc:resources mapping="/html/\*\*" location="/html/" />  <mvc:resources location="/css/" mapping="/css/\*\*"></mvc:resources>  <mvc:resources location="/js" mapping="/js/\*\*"></mvc:resources>-->   <!-- 处理jsp页面所在的前缀和后缀 -->* <**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  *<!-- 前缀,确定当前访问的页面路径地址 -->* <**property name="prefix"**>  <**value**>/WEB-INF/jsp/</**value**>*<!-- 放在WEB-INF文件夹之下的page文件夹 -->* </**property**>  *<!-- 后缀,确定要访问的文件类型 -->* <**property name="suffix"**>  <**value**>.jsp</**value**>  </**property**>  </**bean**> </**beans**> |

## 3.3 dao类

Users.java Books.java Records.java

描述该文件作用，并对里面的关键代码进行解释

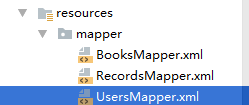
|  |
| --- |
| **package** zyq.pojo;  **public class** Users {  **private** Integer **uid**;   **private** String **uname**;   **private** String **upwd**;   **private** Integer **ubnum**;  //getset方法省略  } |
| **package** zyq.pojo;  **public class** Records {  **private** Integer **rid**;   **private** Integer **rbid**;   **private** Integer **ruid**;   **private** String **rdate**;  //getset方法省略  **}** |
| **package** zyq.pojo;  **public class** Books {  **private** Integer **bid**;   **private** String **bname**;   **private** String **bauthor**;   **private** Integer **bnum**;   **private** String **bcbs**;  **}** |

## 3.4 DAO接口类

描述该文件作用，并对里面的关键代码进行解释

|  |
| --- |
| **public interface** BooksMapper {   **int** insert(Books record);   List<Books> selectBooks();   Books selectByPrimaryKey(Integer bid);   **int** updateBookNum(@Param(**"bid"**)Integer bid, @Param(**"bnum"**)Integer bnum);   **int** huanshu(Integer bid);   List<Books> selectBookByTjlike(@Param(**"bauthor"**) String bauthor, @Param(**"bname"**) String bname);   List<Books> selectBookByTjlikePage(@Param(**"bname"**)String bname, @Param(**"bauthor"**)String bauthor, @Param(**"pageNum"**)Integer pageNum, @Param(**"pageSize"**)Integer pageSize); } |
| **public interface** RecordsMapper {   **int** deleteByPrimaryKey(Integer rid);   **int** insert(Records record);   List<Records> selectByExample(RecordsExample example);   List<Records> selectByPrimaryKey(Integer rid);   List<Records> selectRecords(); } |
| @Mapper **public interface** UsersMapper {  **int** insert(Users users);   List<Users> selectByExample(UsersExample example);   Users selectByPrimaryKey(Integer uid);   Users selectUser(@Param(**"uid"**) Integer uid,@Param(**"upwd"**) String upwd);   **int** updateByPrimaryKeySelective(Users record);  **void** jiaUbnum(Integer uid);   **void** jianUbnum(Integer uid); } |

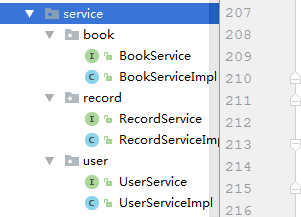
## 3.5 接口映射文件



自动生成部分过多，请自行查看源代码

## 3.6 service层

将dao层与映射文件连接起来



## 3.8 controller层

获取前台数据，并将数据进行处理

|  |
| --- |
| **package** zyq.controller;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Controller; **import** org.springframework.ui.Model; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestMethod; **import** org.springframework.web.bind.annotation.RequestParam; **import** zyq.pojo.Books; **import** zyq.pojo.Users; **import** zyq.service.book.BookService; **import** zyq.service.user.UserService;  **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.List;  */\*\*  \* Created by Administrator on 2020/12/18.  \*/* @Controller **public class** BookController {   @Autowired  **private** BookService **bs**;   @Autowired  **private** UserService **us**;   *//查询所有的书籍信息* @RequestMapping(value=**"list"**,method= RequestMethod.***GET***)  **public** String list(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){   List<Books> books = **bs**.selectBooks();  Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);  model.addAttribute(**"books"**,books);  **return "list"**;    }    *//查询所有的书籍信息* @RequestMapping(**"/bookList"**)  **public** String bookList(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  @RequestParam(value = **"bauthor"**,defaultValue = **""**)String bauthor,  @RequestParam(value = **"bname"**,defaultValue = **""**)String bname,  @RequestParam(value = **"pageNum"**,defaultValue = **"0"**)Integer pageNum,  Model model  ){  **int** pageSize=5;  **int** size = **bs**.selectBookByTjlike(bauthor, bname).size();  **int** total = size%pageSize==0?size/pageSize:size/pageSize+1;  List<Books> books = **bs**.selectBookByTjlikePage(bname, bauthor, pageNum, pageSize);   Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);  model.addAttribute(**"books"**,books);   model.addAttribute(**"pageNum"**,pageNum);  model.addAttribute(**"total"**,total);   model.addAttribute(**"bauthor"**,bauthor);  model.addAttribute(**"bname"**,bname);  **return "list"**;    }   *//用户点击借书时对应书籍数目减一* @RequestMapping(value=**"jieshu"**,method= RequestMethod.***GET***)  **public** String jieshu(  @RequestParam(value = **"bid"**,defaultValue = **""**)Integer bid,  @RequestParam(value = **"bnum"**,defaultValue = **""**)Integer bnum,  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){   bnum--;  **bs**.updateBookNum(bid, bnum);  model.addAttribute(**"uid"**,uid);  model.addAttribute(**"bid"**,bid);  **return "redirect:addRecord.do"**;    }   *//用户点击借书时对应书籍数目加一* @RequestMapping(value=**"huanshu"**,method= RequestMethod.***GET***)  **public** String huashu(  @RequestParam(value = **"rbid"**,defaultValue = **""**)Integer rbid,  @RequestParam(value = **"bid"**,defaultValue = **""**)Integer bid,  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){  **bs**.huanshu(bid);  model.addAttribute(**"uid"**,uid);  model.addAttribute(**"rbid"**,rbid);  **return "redirect:delRecord.do"**;  }    } |
| **package** zyq.controller;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Controller; **import** org.springframework.ui.Model; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestMethod; **import** org.springframework.web.bind.annotation.RequestParam; **import** zyq.pojo.Books; **import** zyq.pojo.Records; **import** zyq.pojo.Users; **import** zyq.service.book.BookService; **import** zyq.service.record.RecordService; **import** zyq.service.user.UserService;  **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.List;  */\*\*  \* Created by Administrator on 2020/12/18.  \*/* @Controller **public class** RecordController {   @Autowired  **private** RecordService **rs**;   @Autowired  **private** UserService **us**;    *//查询所有的借书记录* @RequestMapping(value=**"Recordlist"**,method= RequestMethod.***GET***)  **public** String Recordlist(  Model model,@RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid   ){  List<Records> records = **rs**.selectRecords();  **if** (uid==**null**){  *//用户不登录也可以查看所有借阅记录* }**else** {  *//但登录的话就要记住登录信息* Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);   }  model.addAttribute(**"records"**,records);  **return "record"**;  }   *//点击借书后增加一条借书记录* @RequestMapping(value=**"addRecord"**,method= RequestMethod.***GET***)  **public** String jieshu(  @RequestParam(value = **"bid"**,defaultValue = **""**)Integer rbid,  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer ruid,  Model model  ){   SimpleDateFormat format = **new** SimpleDateFormat(**"yyyy-mm-dd HH:mm:ss"**);  String date =format.format(**new** Date());   Records records = **new** Records(**null**,rbid,ruid,date);  **rs**.insert(records);  model.addAttribute(**"uid"**,ruid);  **return "redirect:jiaUbnum.do"**;  }   *//点击还书后删除对应的借书记录* @RequestMapping(value=**"delRecord"**,method= RequestMethod.***GET***)  **public** String huanshu(  @RequestParam(value = **"rbid"**,defaultValue = **""**)Integer rbid,  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){   **rs**.deleteByPrimaryKey(rbid);  model.addAttribute(**"uid"**,uid);  **return "redirect:jianUbnum.do"**;  }    } |
| **package** zyq.controller;  **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Controller; **import** org.springframework.ui.Model; **import** org.springframework.ui.ModelMap; **import** org.springframework.web.bind.annotation.\*; **import** org.springframework.web.servlet.mvc.support.RedirectAttributes; **import** zyq.pojo.Books; **import** zyq.pojo.Records; **import** zyq.pojo.Users; **import** zyq.service.book.BookService; **import** zyq.service.record.RecordService; **import** zyq.service.user.UserService;  **import** javax.servlet.http.HttpServletRequest; **import** javax.servlet.http.HttpServletResponse; **import** javax.servlet.http.HttpSession; **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.List;  */\*\*  \* Created by Administrator on 2020/12/17.  \*/* @Controller **public class** UserController {   @Autowired  **private** UserService **us**;   @Autowired  **private** BookService **bs**;   @Autowired  **private** RecordService **rs**;   *//记录uid并返回主页的方法* @RequestMapping(**"/index"**)  **public** String index(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){  **if**(uid==**null**){  **return "index"**;  }**else** {  Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);  **return "index"**;  }   }     *//去登录页面的方法* @RequestMapping(value=**"tologin"**,method= RequestMethod.***GET***)  **public** String tologin(){  **return "login"**;  }   *//验证登录的方法* @RequestMapping(value=**"login"**,method=RequestMethod.***POST***)  **public** String login(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  @RequestParam(value = **"upwd"**,defaultValue = **"非"**)String upwd,  Model model,HttpSession session  ){  Users user = **us**.selectUser(uid, upwd);  */\* Users users = us.selectByPrimaryKey(uid);\*/* **if** (user!=**null**){  model.addAttribute(**"user"**,user);  **return "index"**;  }**else** {  session.setAttribute(**"flag"**,0);  **return "index"**;  }   }    *//去注册页面方法* @RequestMapping(value=**"toadd"**,method= RequestMethod.***GET***)  **public** String toadd(){  **return "register"**;  }   *//注册账号结束后返回主页* @RequestMapping(value=**"add"**,method=RequestMethod.***POST***)  **public** String add(Users users){  **us**.insert(users);  **return "index"**;  }    *//用户修改信息数据回显方法* @RequestMapping(value=**"toupdateUser"**,method= RequestMethod.***GET***)  **public** String toupdateUser(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){  Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);  **return "updateUser"**;  }    *//修改好数据后返回主页* @RequestMapping(value=**"updateUser"**,method= RequestMethod.***POST***)  **public** String updateUser(Users users){  **us**.updateByPrimaryKeySelective(users);  **return "index"**;  }   *//点击借书后用户名下书籍数目加1* @RequestMapping(value=**"jiaUbnum"**,method= RequestMethod.***GET***)  **public** String jieshu(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){  **us**.jiaUbnum(uid);  model.addAttribute(**"uid"**,uid);  **return "redirect:bookList.do"**;  }   *//点击借书后用户名下书籍数目减1* @RequestMapping(value=**"jianUbnum"**,method= RequestMethod.***GET***)  **public** String huanshu(  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid,  Model model  ){  **us**.jianUbnum(uid);  model.addAttribute(**"uid"**,uid);  **return "redirect:user.do"**;  }   *//去用户中心方法* @RequestMapping(**"/user"**)  **public** String user(  Model model,Users users,HttpSession session,  @RequestParam(value = **"uid"**,defaultValue = **""**)Integer uid  ){   **if** (uid==**null**){  session.setAttribute(**"flag"**,1);  **return "index"**;  }**else** {  Users user = **us**.selectByPrimaryKey(uid);  model.addAttribute(**"user"**,user);  }   List<Records> records = **rs**.selectByPrimaryKey(uid);  List<Books> books = **bs**.selectBooks();  model.addAttribute(**"books"**,books);  model.addAttribute(**"records"**,records);  **return "user"**;  }   } |

# 4 系统总结

本次实验中我所遇最严重的问题就是数据分页处理，当数据分页的时候会因为软件缓存导致sql语句出现，后清缓存就可解决。

本门课程中我收货良多，对web前后端结合也有了比较系统和全面的认识，大大提升了我的编程能力与思考能力。

很感谢老师对我的栽培。