陕西科技大学
SHAANXI UNIVERSITY OF SCIENCE & TECHNOLOGY

专业综合设计 II 课程设计 说明书

题目： 基于 SSM 的在线书籍管理系统

学生姓名： 左雅雯

学　　号： 201806060112

院 （系）： 电子信息与人工智能学院

专　　业： 计算机科学与技术

指导教师： 陈景霞、王梅嘉、吴华

2021 年 6 月 9 日

目 录

1 课题简介

1.1 背景及研究现状

随着现代人们知识层次的提高，图书馆成为了日常生活中不可缺少的一部分。而图书馆的藏书数量和读者的人数过多，使得图书馆的图书管理业务量十分庞大，依靠于传统的记账方式早已不可行，因此，便有了图书馆管理系统的产生，在当今这个信息化的社会，变得尤为重要。图书管理系统主要的面向对象为学校以及社会公开性的图书馆，为管理员提供读者信息的管理，对于借阅日志的管理，以及图书的增删改查功能的实现，与此同时，在用户端，系统还可以让用户对于图书信息进行查询以及相应的借书还书操作，对于现在这个数据量庞杂的社会而言，不失为一个较好的系统开发。

1.2 设计内容与设计思路

（1）设计内容

设计一个包含图书管理（增删改查）、用户管理（增删改查）、日志管理的系统，数据内容包括图书信息的必要属性。

（2）设计思路

系统采用SSM（Spring+SpringMVC+Mybatis）框架结构搭建系统后台，前端采用Jquery、Bootstrap的框架技术搭建完成前端页面的设计。确定各个界面的需求和交互逻辑，完成后进行最后的测试

1.3 设计目的及意义

建立图书管理系统，使图书管理工作规范化、系统化、程序化、避免图书管理的随意性，提高信息处理的速度和准确性，能够及时，准确，有效的查询和修改图书情况。这样有利于学校以及图书管理员对于图书的信息化管理，使得图书馆管理工作更加规范。用户也可以更加方便快捷的进行图书的查询以及借阅操作。

2 系统分析与设计

2.1 可行性分析

2.1.1 经济可行性

1）支出分析：系统项目的支出可以分为一次性支出和非一次性支出两类。一次性支出包括开发费、培训费、初数据录入费、设备购置费。非一次性支出包括软、硬件租金、人员工资及维护等费用。而本系统开发简单，同时操作简便，因此前期开发费用、培训费

用等相对较低。而本系统可以采用数据库的备份文件、日志文件等进行维护恢复，维护成本也相对较低。

2）收益分析：本系统是用于在线图书管理，因此不能带来直接经济收益。但其开发简单、操作简便的特点可以大大降低人力成本，同时也能吸引大众前来图书馆，带来间接的经济收益。

2.1.2 技术可行性

该系统开发所需要的的系统开发工具及技术：

客户端（界面）开发所需技术：Jquery、Bootstrap

服务端开发技术：JSP+JavaBean+Servlet+JDBC

数据库服务器：MySQL

服务器：Jetty

开发框架：SpringMVC+Spring+Mybatis

开发所需要的技术和编程语言及框架是当下主流的，应用的开发工具和数据库更是相对普遍的，因此本系统在技术方面完全是可行的。

2.1.3 操作可行性

系统界面简洁通俗易懂，数据录入迅速、规范、可靠，数据统计准确，适应力强，容易扩充。具有易用性、灵活性、开放性、可视性，这些基本的都可以在该系统中进行实现。

从以上经济、技术、操作三方面的分析可以看出，本系统的开发时机成熟，从多种角度考虑，能够使整个系统内部工作简化、提高工作效益。

2.2 需求分析

2.2.1 系统设计目标

对于管理员来说，掌握图书信息、读者信息、借阅日志是非常必要的，传统的纸质管理方式只要保存得当数据就不会丢失，但是图书信息、读者信息有变更，或者想要查询相关的图书或是读者的信息，将会造成很大的工作量。该系统旨在减少管理员对书籍信息管理的工作量，能够通过直接登录管理员账号对图书信息、读者信息进行增加、删除、修改、查询，并管理借阅日志。

对于读者来说，线下借阅书籍的手续过于繁琐，查询图书更是困难，因此在线书籍管理系统必须要满足图书的查询、借阅和归还，除此之外，还应当实现个人信息的修改，以及个人借阅的日志查询。

综上，本系统的设计目标是为了满足两者的需求。

2.2.2 系统功能需求分析

本在线书籍管理系统最终要实现的功能如下：

1、管理员登录：

1） 进入系统前验证用户名以及用户密码，输入正确且符合要求方可进入系统。

2） 管理员对于图书信息进行相应的增删改查。

3） 管理员对于借阅日志进行相应的删除。

4） 管理员对于读者信息进行相应的增加和修改。

5） 管理员对于账户密码进行修改。

2、读者登录：

1） 进入系统前验证用户名以及用户密码，输入正确且符合要求方可进入系统。

2） 读者可以对图书信息进行查询，进行相应的借书还书操作。

3） 读者可以对个人信息进行相应查询和修改。

4） 读者可以对自己的借还信息进行相应查询。

5) 读者对于账户密码进行修改。

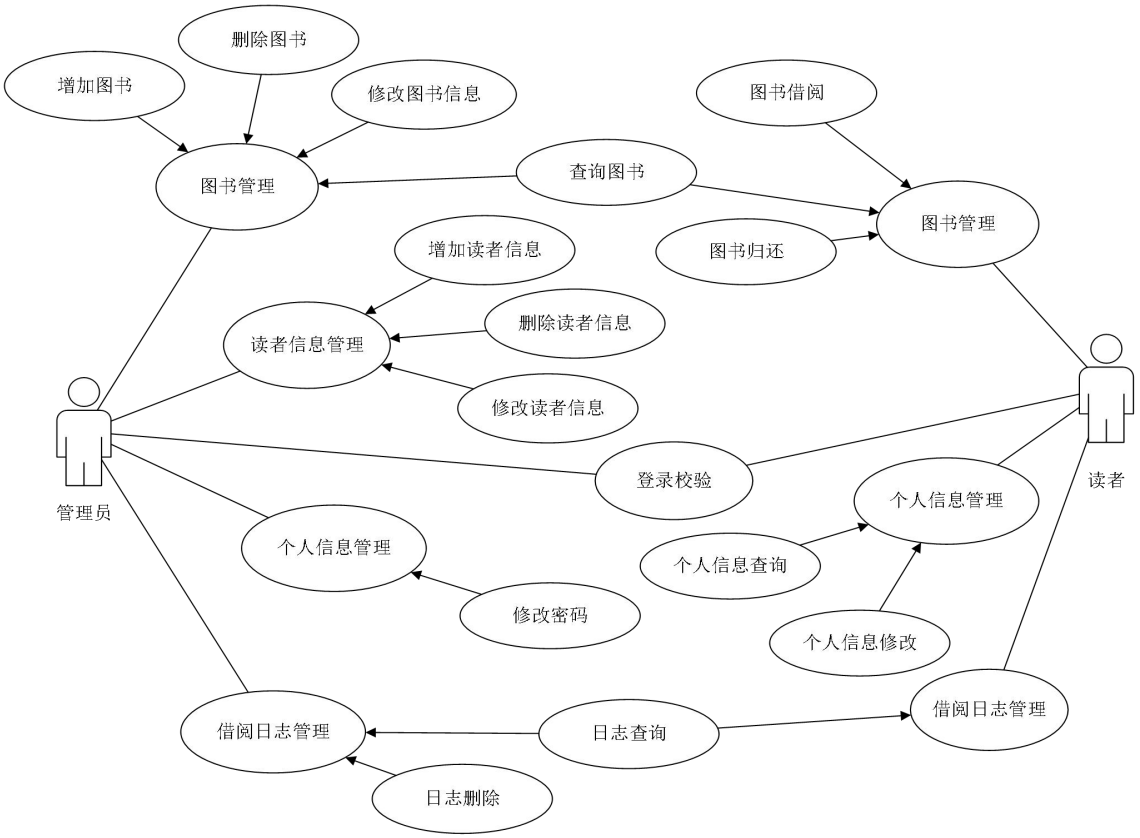

图 2-1 系统功能用例图

2.2.3 系统性能需求分析

1）可维护：系统的基本维护必须简单，不要求需专业技术人员进行维护，通过一般的技术维护人员操作系统的维护功能，即可达到基本的维护目的，例如：数据备份、恢复；数据导入/导出等维护性操作。

2）可靠性：由于系统需要有较高的可靠性，在系统出现错误时，要求应用系统能报告相应的详细错误信息或原因给操作员，或者老师或学生给管理员留言，提示错误和问题，以便理解与分析。最好能够给出一个解决方案。

3）安全性：在硬件方面可考虑采用性能优异的防火墙，根据规则过滤或代理应用数据包，防止非法网络活动。系统方面对访问系统的用户分权限管理，系统管理员拥有对系统所有的权限，用户只能进行某些特定的功能的操作。防止未授权用户的非法登陆，并对用户对系统的操作做好记录，有利于在发现系统故障时快速查找原因。

4）功能性：具有必要和充分功能的程度，这些功能将满足用户需要，

5）可移植性：系统或部件能从一种硬件或者软件环境转化至另外一种环境的特性，具有较高的适应性，不需采用额外的活动或手段 就能适应不同指定环境的能力。

6）易用性：与用户使用软件所花费的努力及其对使用的及评价相关的特性，具有可理解、可学习、可操作性。

2.3 系统总体设计

2.3.1 系统总体结构设计

1）B/S 架构

B/S 架构采取浏览器请求，服务器响应的工作模式。其工作原理是用户可以通过浏览器去访问 Internet 上由 Web 服务器产生的文本、数据、图片、动画、视频点播和声音等信息；而每一个 Web 服务器又可以通过各种方式与数据库服务器连接，大量的数据实际存放在数据库服务器中；从 Web 服务器上下载程序到本地来执行，在下载过程中若遇到与数据库有关的指令，由 Web 服务器交给数据库服务器来解释执行，并返回给 Web 服务器，Web 服务器又返回给用户。在这种结构中，将许许多多的网连接到一块，形成一个巨大的网，即全球网。而各个企业可以在此结构的基础上建立自己的 Internet。

工作流程：客户端发送请求： 用户在客户端【浏览器页面】提交表单操作，向服务器发送请求，等待服务器响应；服务器端处理请求： 服务器端接收并处理请求，应用服务器端通常使用服务器端技术，如 JSP 等，对请求进行数据处理，并产生响应；服务器端发送响应： 服务器端把用户请求的数据（网页文件、图片、声音等等）返回给浏览器。浏览器解释执行 HTML 文件，呈现用户界面。

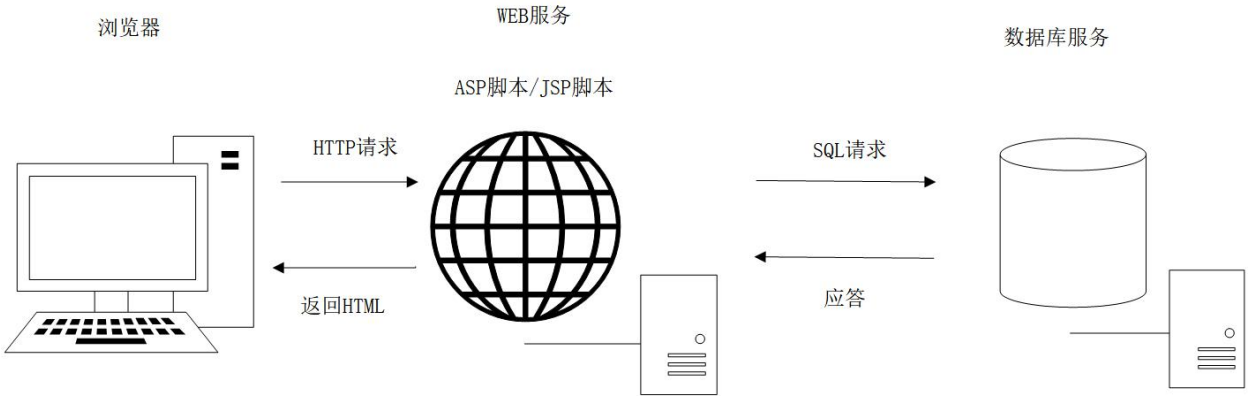

图 2-2 B/S 架构图

2）Model2 MVC 设计模式

在 MVC 设计模式中：C 代表 Controller，负责用户界面和业务逻辑层的通信控制，一方面解释来自用户界面的输入，识别用户动作（如点击按钮等），调用相应 Model 中的方法，另一方面处理来自 Model 的事件和返回的执行结果，调用适当的 View 显示给用户，Controller 主要由 Servlet 完成。M 代表 Model，负责整个解决方案的业务逻辑实现，底层的数据库也由 Model 访问和操作；V 代表 View，负责系统向用户的展示，主要由 HTML 及 JSP 等完成。

本系统中，webapp/WEB-INF/jsp 目录下存放了所有的 JSP 页面，即 View 层，系统向用户的展示；src/main/java/com.library/controller 目录下存放了所有的控制器即 Controller 层，由 Sevlet 来识别用户动作调用相应的方法，或是处理返回的结果；src/main/java/com.library/service 目录下存放的文件用于整个业务逻辑实现，包括底层数据库，即 Model 层。

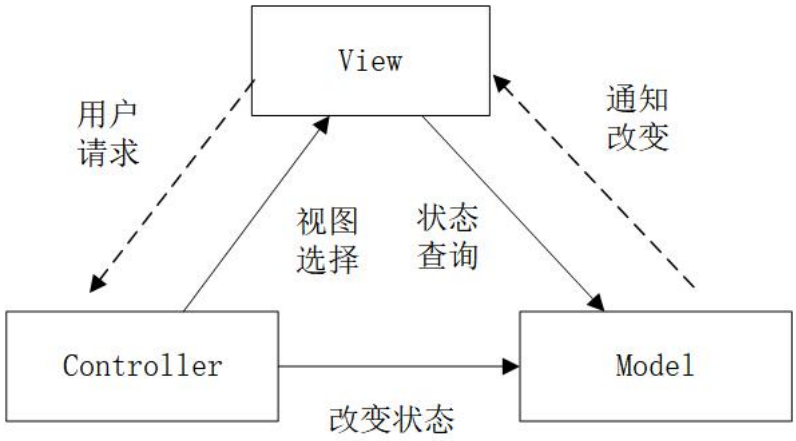

图 2-3 MVC 设计模式图

3）系统后台 SSM（Spring+SpringMVC+Mybatis）框架

SSM 框架是 Spring MVC ，Spring 和 Mybatis 框架的整合，是标准的 MVC 模式，将整个系统划分为 View 层，Controller 层，Service 层，DAO 层四层。Spring 实现业务对象管理，Spring MVC 负责请求的转发和视图管理，Mybatis 作为数据对象的持久化引擎。

（a）框架的总体结构：

①Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式，组成 Spring 框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。

② Spring MVC 负责请求的转发和视图管理，Spring 的 MVC 框架主要由 DispatcherServlet、处理器映射、处理器(控制器)、视图解析器、视图组成。

③Mybatis 作为数据对象的持久化引擎。

（b）框架的基本原理：

①Spring 原理：

让对象与对象（模块与模块）之间的关系没有通过代码来关联，都是通过配置类说明管理的（Spring 根据这些配置 内部通过反射去动态的组装对象） ，Spring 是一个容器，凡是在容器里的对象才会有 Spring 所提供的这些服务和功能。

②SpringMVC 运行原理:

客户端请求提交到 DispatcherServlet；由 DispatcherServlet 控制器查询一个或多个 HandlerMapping，找到处理请求的 Controller；DispatcherServlet 将请求提交到 Controller；Controller 调用业务逻辑处理后，返回 ModelAndView；DispatcherServlet 查询一个或多个 ViewResoler 视图解析器，找到 ModelAndView 指定的视图；视图负责将结果显示到客户端。

③Mybatis 的工作原理：

读取 MyBatis 配置文件：mybatis-config.xml 为 MyBatis 的全局配置文件，配置了 MyBatis 的运行环境等信息，例如数据库连接信息。

加载映射文件。映射文件即 SQL 映射文件，该文件中配置了操作数据库的 SQL 语句，需要在 MyBatis 配置文件 mybatis-config.xml 中加载。mybatis-config.xml 文件可以加载多个映射文件，每个文件对应数据库中的一张表。

构造会话工厂：通过 MyBatis 的环境等配置信息构建会话工厂 SqlSessionFactory。

创建会话对象：由会话工厂创建 SqlSession 对象，该对象中包含了执行 SQL 语句的所有方法。

Executor 执行器：MyBatis 底层定义了一个 Executor 接口来操作数据库，它将根据 SqlSession 传递的参数动态地生成需要执行的 SQL 语句，同时负责查询缓存的维护。

MappedStatement 对象：在 Executor 接口的执行方法中有一个 MappedStatement 类型的参数，该参数是对映射信息的封装，用于存储要映射的 SQL 语句的 id、参数等信息。

输入参数映射：输入参数类型可以是 Map、List 等集合类型，也可以是基本数据类型和 POJO 类型。输入参数映射过程类似于 JDBC 对 preparedStatement 对象设置参数的过程。

输出结果映射：输出结果类型可以是 Map、 List 等集合类型，也可以是基本数据类型和 POJO 类型。输出结果映射过程类似于 JDBC 对结果集的解析过程。

4）系统前端 bootstrap 框架

（a）总体结构：Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。集合 CSS、HTML 和 JavaScript，使用了最新的浏览器技术，为实现快速开发提供了一套前端工具包，包括布局、栅格、表格、按钮、表单、导航和提示等。

（b）基本原理：

流体网格布局： 可伸缩的网格（大小宽高 都是可以伸缩， 可用 flex 或者百分比来控制），布局上面元素大小不固定可伸缩

弹性图片，图片宽高不固定（可设置 min-width = 100%）

媒体查询： 让网页在不同设备的终端上面展示效果相同（让用户体验相同，用着更爽）， 在不同的设备（大小不同，分辨率不同）上面均展示合适的页面

主要断点， 设备宽度的临界点， 大小区别， 宽度不同， 根据不同宽度展示不同的样式

媒体查询： 向不同设备提供不同样式的一种方式，他让每种类型的用户提供最佳体验。

2.3.2 系统功能模块设计

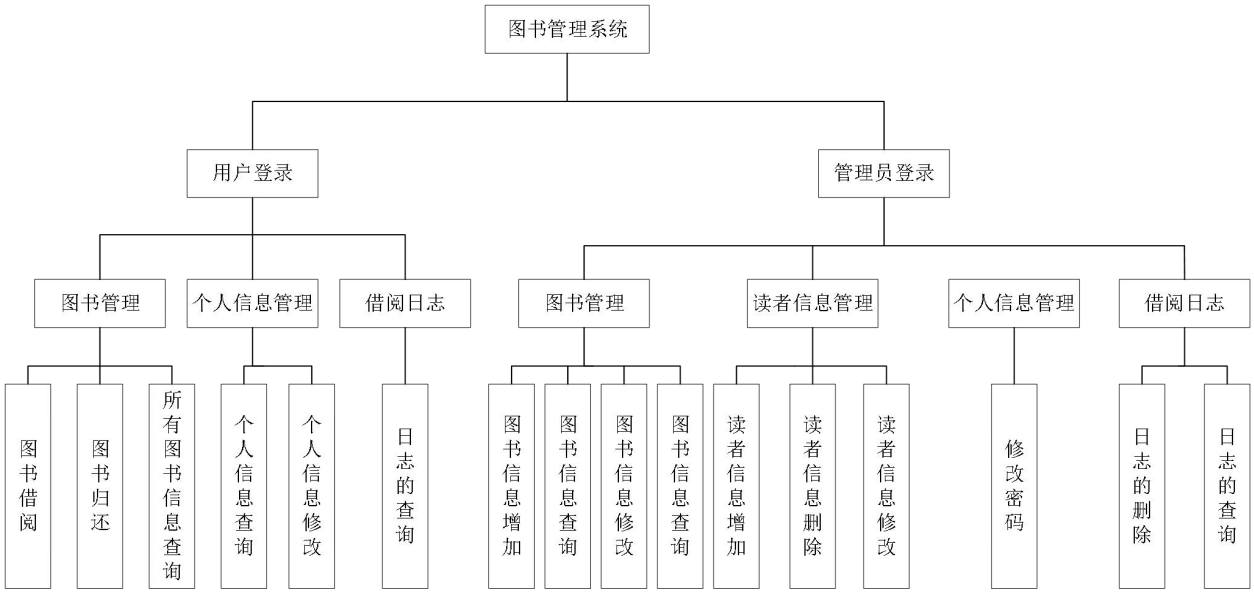

图 2-4 系统功能模块图

该系统具有两个部分，管理员和用户登录入口。

（1）用户登录，具有图书管理和个人信息管理两部分。

图书信息管理：借阅、归还、查询图书；

个人信息管理：查询、修改个人信息。

（2）管理员登录，具有图书管理、读者信息管理、个人信息管理、借阅信息管理四部分。

图书信息管理：增加、删除、修改、查询图书；

读者信息管理：增加、删除、修改读者信息；

个人信息管理：查询、修改个人信息；

借阅信息管理：删除、查询日志信息。

2.3.3 主要业务流程

1）用户登录系统模块

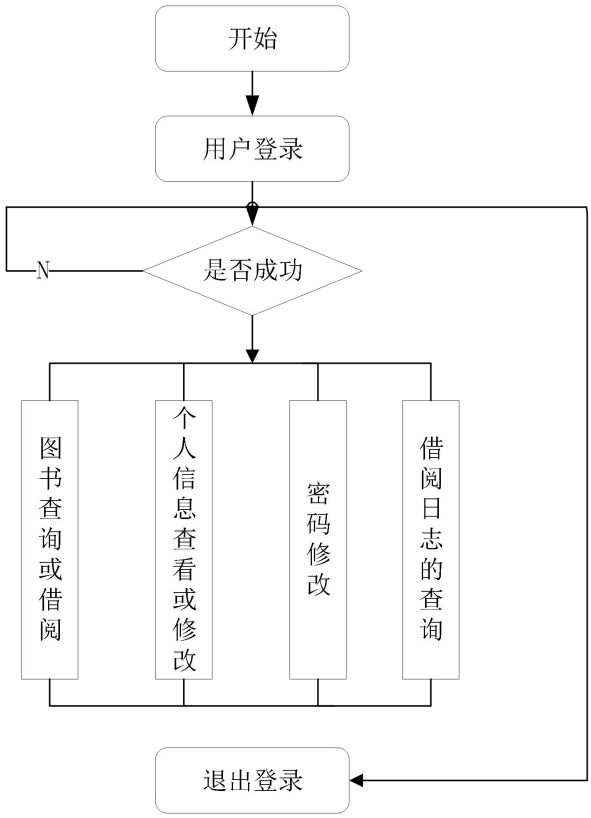

图 2-5 用户登录流程图

2）管理员登录系统模块

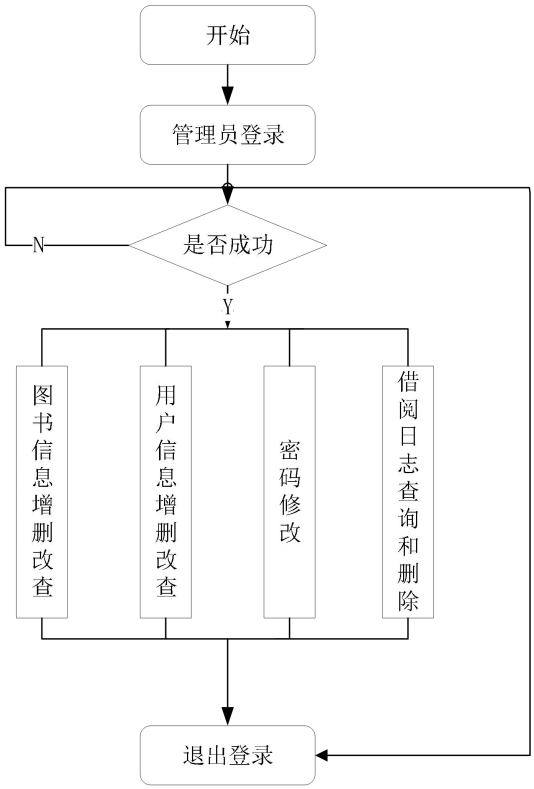

图 2-6 管理员登录流程图

2.4 系统数据库设计

2.4.1 数据库概念模型设计

1）读者实体属性

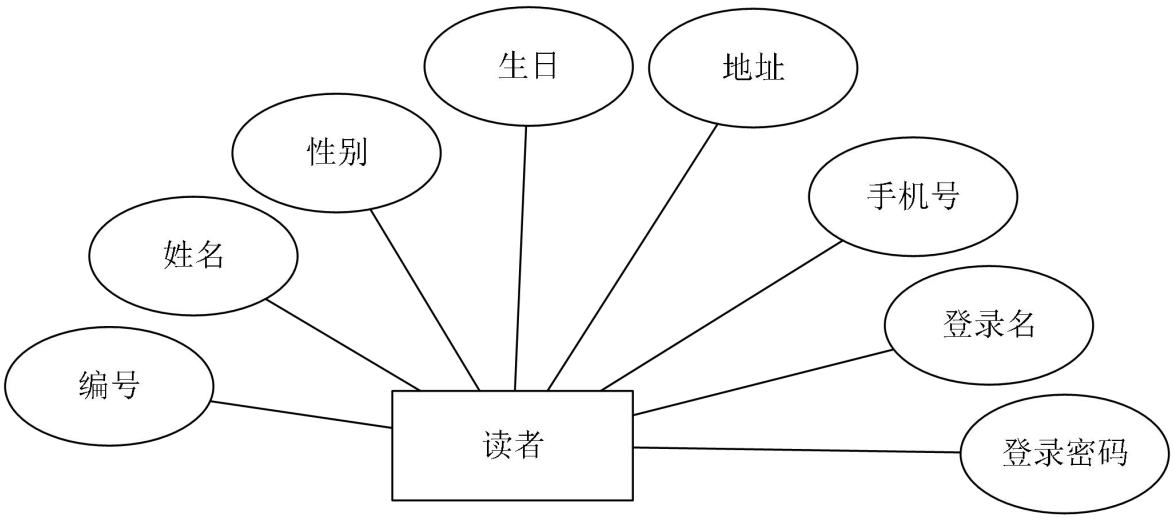

图 2-7 读者实体属性

2）图书实体属性

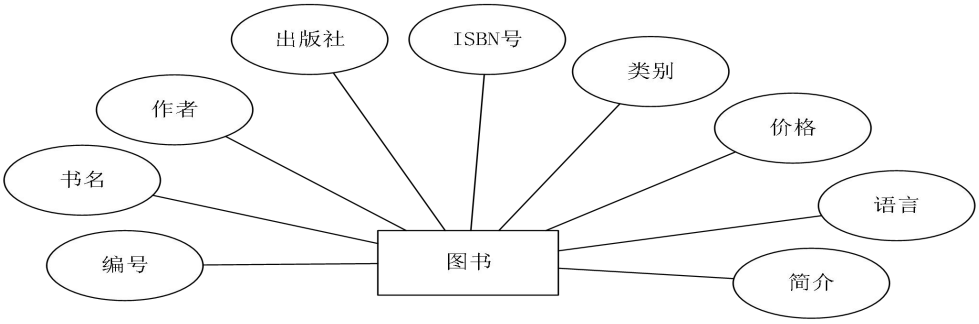

图 2-8 图书实体属性

3）管理员实体属性

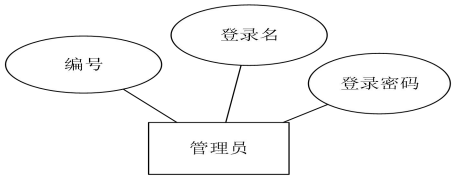

图 2-9 管理员实体属性

4）数据库设计总的 E-R 图

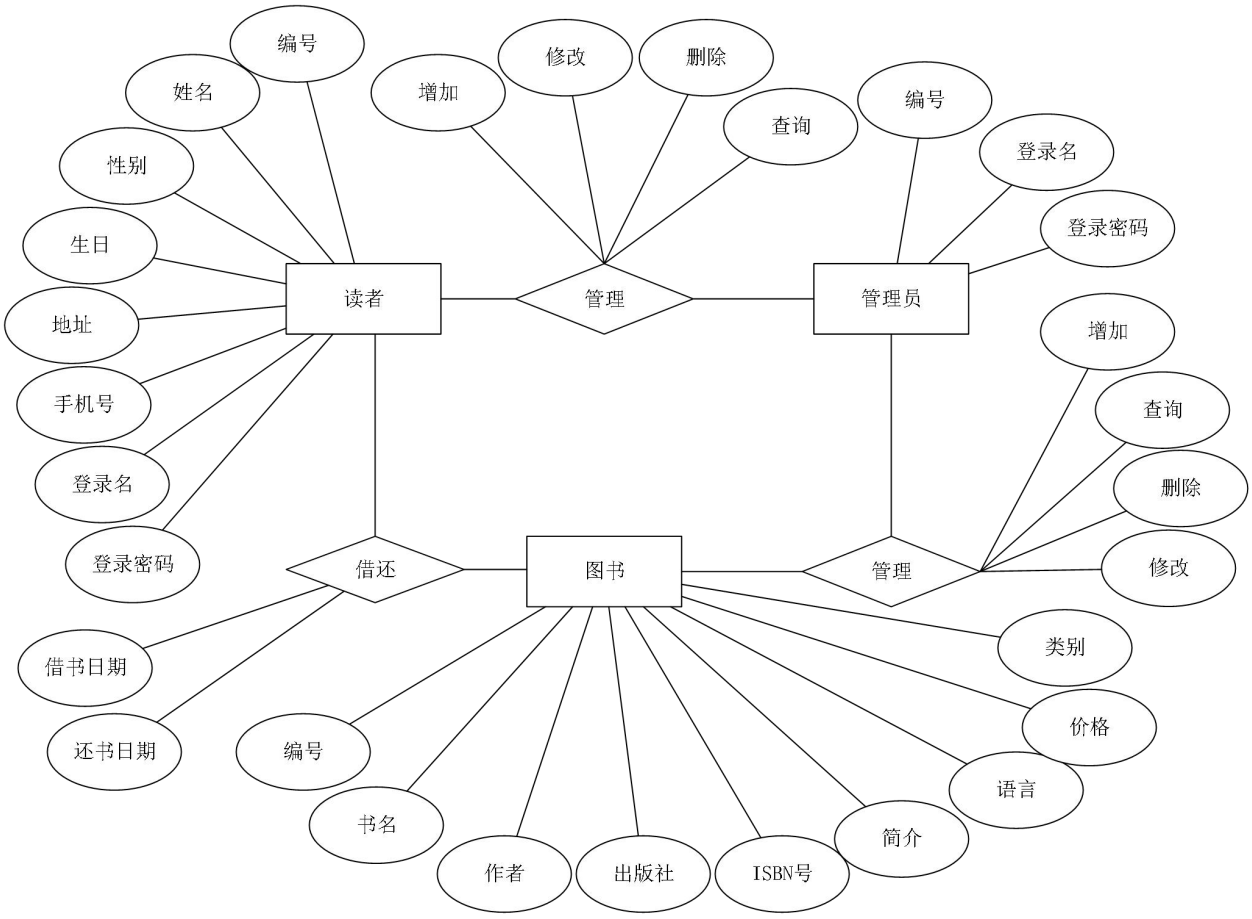

图 2-10 总体 E-R 图

2.4.2 数据库逻辑结构设计

系统采用 Mysql8.0 进行数据库及设计，由于 MySQL 数据库体积小、速度快、总体拥有成本低、开放源代码，因本次成为网站开发的首选。

给图书管理系统一共创建了 6 个数据表，数据表的逻辑结构设计，如下表格所示：

1、图书书目表 book_info

表 2-1 图书书目表

名	类型	长度	小数点	NULL	用途	键
book_id	bigint	20	0	否	图书号	是
name	varchar	20	0	否	书名	
author	varchar	15	0	否	作者	
publish	varchar	20	0	否	出版社	
ISBN	varchar	15	0	否	标准书号	
introduction	Text	0	0	是	简介	
language	varchar	4	0	否	语言	
price	decimal	10	2	否	价格	
pub_date	date	0	0	否	出版时间	
class_id	int	11	0	是	分类号	
number	int	11	0	是	剩余数量	

2、数据库管理员表 admin

表 2-2 数据库管理员表

名	类型	长度	小数点	NULL	用途	键
admin_id	bigint	20	0	否	账号	是
password	varchar	15	0	否	密码	
username	varchar	15	0	是	用户名	

3、图书分类表 class_info

表 2-3 图书分类表

名	类型	长度	小数点	NULL	用途	键
class_id	int	11	0	否	类别名	是
class_name	varchar	15	0	否	类别号	

4、借阅信息表 lend_list

表 2-4 借阅信息表

名	类型	长度	小数点	NULL	用途	键
ser_num	bigint	20	0	否	流水号	是
book_id	bigint	20	0	否	读者证号	
reader_id	bigint	20	0	否	图书号	
lend_date	date	0	0	是	借书日期	
back_date	date	0	0	是	还书日期	

5、借阅卡信息表 reader_card

表 2-5 借阅卡信息表

名	类型	长度	小数点	NULL	用途	键
reader_id	bigint	20	0	否	读者证号	是
password	varchar	15	0	否	密码	
username	varchar	15	0	是	用户名	

6、读者信息表 reader_info

表 2-6 读者信息表

名	类型	长度	小数点	NULL	用途	键
reader_id	bigint	20	0	否	读者证号	是
name	varchar	10	0	否	姓名	
sex	varchar	2	0	否	性别	
birth	date	0	0	否	生日	
address	varchar	50	0	否	地址	
phone	varchar	15	0	否	电话	

2.4.3 数据库关系设计

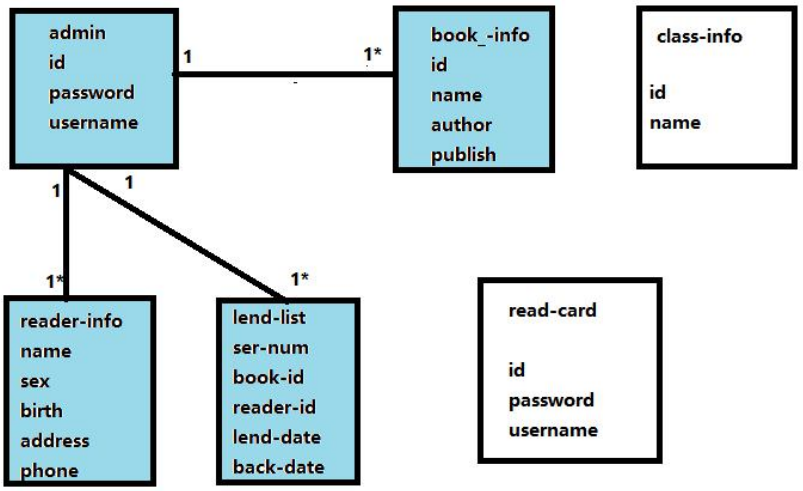

图 2-11 数据库关系图

3 系统详细设计

3.1 系统开发及运行环境

表 3-1 系统开发及运行环境

需求名称	详细要求
开发语言	Java
软件环境	数据库 Mysql5.7 以上版本，JDK12，Intellij IDEA
硬件环境	Windows10 系统、64 位操作系统、Intel i5 处理器
其他需求	所需硬盘空间在 5MB 以上

3.2 系统采用的关键技术

3.2.1 页面显示逻辑与业务逻辑相分离

在面向对象的设计时，不可避免的要涉及到对象的持久化问题。由于目前关系型数据库的流行及 SQL 语言的强大功能，一般都以关系型数据库存储要持久化的对象。

但一般认为数据库是实现细节，应该和业务逻辑对象分离，以保持业务对象的可复用性及可扩展性。为了业务逻辑与数据库分离，页面显示逻辑部分主要想采用 jsp、CSS 进行代码编写，业务逻辑主要在 DAO 以及 Service、controller 层进行实现。

service 层：存放业务逻辑处理，也是一些关于数据库处理的操作，但不是直接和数据库打交道，他有接口还有接口的实现方法。

dao 层：对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，而 service 层是针对 controller。

controller 层：控制器，导入 service 层，controller 通过接收前端传过来的参数进行业务操作，在返回一个指定的路径或者数据表。

3.2.2 mybatis 数据持久化

mybatis 数据访问层用于读取数据库中是否存在当前登录的用户信息，并返回与用户信息对应的行数。

mybatis 底层还是采用原生 jdbc 来对数据库进行操作的，只是通过 SqlSessionFactory，SqlSession Executor,StatementHandler，ParameterHandler,ResultHandler 和 TypeHandler 等几个处理器封装了这些过程。

3.3 系统框架的实现（或基本配置）

3.3.1 Bootstrap 前端框架搭建：

- 下载用于生产环境的 Bootstrap
- 解压复制粘贴到 src/main/static(新建)
- 同时在 src/main/static/js 下引入 js 文件
- 然后在页面中引入即可

具体代码如下：

```
<!-- 引入 Jquery -->
<script type="text/javascript" src="static/js/jquery-1.12.4.min.js"></script>
<!-- 引入样式 -->
 <link href="static/bootstrap-3.3.7-dist/css/bootstrap.min.css" rel="stylesheet">
 <!-- 引入 js -->
 <script src="static/bootstrap-3.3.7-dist/js/bootstrap.min.js"></script>
```

3.3.2 Spring 框架搭建：

Spring 的配置主要有以下三点：数据源、与 MyBatis 整合、事务控制

1）数据源

类路径下新建文件 db.properties

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/library?useUnicode=true&characterEncoding=UTF-8

jdbc.username=root

jdbc.password=root

2）MyBatis 整合

类路径下新建文件 book_context.xml 文件

<!--配置和 Mybatis 的整合-->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<!--数据库连接池--> <property name="dataSource" ref="dataSource"/>

<!--加载 mybatis 的全局配置文件--><!--映射文件-->

<property name="mapperLocations" value="classpath:MyBatis/*"/>

</bean>

3.3.3 主要的类与接口

表 3-2 主要的类和接口

功能模块	描述	实现方法	具体代码
表现层	用户界面层	JSP、JQuary、CSS	admin_book admin_book_detail admin_header admin_info admin_reader
控制层	控制业务流程	Spring 配置文件中进行相关配置	BookController LendController LoginController ReaderController WebConfig
业务逻辑层	处理系统业务逻辑	调用相关 Dao 层接口完成相关操作	BookService LendService LoginService ReaderService ReaderInfoService

持久层	对系统数据库操作 确保信息更新和存储	采用 Mybatis 框架，使用简单 XML 注解，将接口进行映射	AdminDao BookDao LendDao ReaderDao ReaderInfoDao
模型类	主要用于封装实体类信息	对用户的属性以及构造方法进行设置	Admin Book Lend ReaderCard ReaderInfo

3.3.4 系统主要配置文件

（1）控制层负责处理 loginCheck.html 请求，请求参数会根据参数名称默认契约自动绑定到相应方法的入参中，通过业务层方法获取用户的相应身份权限，并将用户信息通过散列表返回。

```
@RequestMapping(value = "/api/loginCheck", method = RequestMethod.POST)
@ResponseBody
public Object loginCheck(HttpServletRequest request) {
    long id = Long.parseLong(request.getParameter("id"));
    String passwd = request.getParameter("passwd");
    boolean isReader = loginService.hasMatchReader(id, passwd);
    boolean isAdmin = loginService.hasMatchAdmin(id, passwd);
    HashMap<String, String> res = new HashMap<>();
    if (isAdmin) {
        Admin admin = new Admin();
        admin.setAdminId(id);
        admin.setPassword(passwd);
        String username = loginService.getAdminUsername(id);
        admin.setUsername(username);
        request.getSession().setAttribute("admin", admin);
        res.put("stateCode", "1");
```

```
            res.put("msg", "管理员登陆成功！");
        } else if (isReader) {
            ReaderCard readerCard = loginService.findReaderCardByReaderId(id);
            request.getSession().setAttribute("readercard", readerCard);
            res.put("stateCode", "2");
            res.put("msg", "读者登陆成功！");
        } else {
            res.put("stateCode", "0");
            res.put("msg", "账号或密码错误！");
        }
        return res;
}
```

（2）负责解耦的服务层，作为控制层和 mapper 层之间缓冲的层，用于判断中是否存在当前登录的用户信息，如果存在，返回 true，否则，返回 false，具体实现代码如下：

```
@Override
public boolean hasMatchReader(long readerId, String password){
    return    readerCardmapper.getIdMatchCount(readerId, password)>0;
}
@Override
public boolean hasMatchAdmin(long adminId, String password){
    return adminmapper.getMatchCount(adminId, password) == 1;
}
```

（3）mybatis 数据访问层用于读取数据库中是否存在当前登录的用户信息，并返回与用户信息对应的行数，xml 代码及映射接口的具体实现代码如下：

```
<select id="getMatchCount" resultType="int">
    select count(*) from admin
    where admin_id = #{admin_id}
    and password = #{password}
</select>
<select id="getIdMatchCount" resultType="int">
    select count(*) from reader_card where
    reader_id = #{reader_id} and password = #{password}
</select>
int getIdMatchCount(final long reader_id, final String password);
int getMatchCount(final long admin_id, final String password);
```

（4）用户界面层后台代码直接获取用户输入的信息，首先验证是否输入为空，若为空，提示错误，若正确，则通过对象调用业务层代码验证用户是否存在，如果存在，则跳转到系统主界面，否则，提示错误。当跳转到系统主界面时，主界面后台代码根据当前登录的用户信息查找该用户拥有的所有角色，通过角色编号获取角色所拥有的模块菜单及角色信息，形成页面头部菜单导航信息，供用户使用。

3.4 具体功能模块的实现

在这个项目中，我主要负责的是编辑图书模块，也就是实现在管理员方面对图书的信息进行增删改查的功能。下面是具体实现：

3.4.1 数据模型

Book 类将图书对应的相关属性进行封装：

```
package com.library.bean;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Date;

public class Book implements Serializable {

    private long book_id;
    private String name;
    private String author;
    private String publish;
    private String ISBN;
    private String introduction;
    private String language;
    private BigDecimal price;
    private Date pub_date;
    private int class_id;
    private int number;

    public long getBookId() {
        return book_id;
    }
```

```
public void setBookId(long book_id) {
        this.book_id = book_id;
}

public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}

public String getAuthor() {
        return author;
}
public void setAuthor(String author) {
        this.author = author;
}

public String getPublish() {
        return publish;
}
public void setPublish(String publish) {
        this.publish = publish;
}

public String getIsbn() {
        return ISBN;
}
public void setIsbn(String isbn) {
        this.ISBN = isbn;
}

public String getIntroduction() {
        return introduction;
}
public void setIntroduction(String introduction) {
```

```
        this.introduction = introduction;
    }

    public String getLanguage() {
        return language;
    }
    public void setLanguage(String language) {
        this.language = language;
    }

    public BigDecimal getPrice() {
        return price;
    }
    public void setPrice(BigDecimal price) {
        this.price = price;
    }

    public Date getPubdate() {
        return pub_date;
    }
    public void setPubdate(Date pub_date) {
        this.pub_date = pub_date;
    }

    public int getClassId() {
        return class_id;
    }
    public void setClassId(int class_id) {
        this.class_id = class_id;
    }

    public int getNumber() {
        return number;
    }
    public void setNumber(int number) {
        this.number = number;
```

```
    }
}
```

3.4.2 数据库操作模型

BookDao 中定义方法：根据图书名和作者查询图书、查询全部图书、增加图书、编辑图书信息、删除图书的相关方法。

```
package com.library.dao;

import com.library.bean.Book;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.stereotype.Repository;

import javax.annotation.Resource;
import java.util.ArrayList;
import java.util.List;

@Repository
public class BookDao<SqlSession> {

    private final static String NAMESPACE = "com.library.dao.BookDao.";
    @Resource
    private SqlSessionTemplate sqlSessionTemplate;

    public int matchBook(final String searchWord) {
        String search = "%" + searchWord + "%";
        return sqlSessionTemplate.selectOne(NAMESPACE + "matchBook", search);
    }

    public ArrayList<Book> queryBook(final String searchWord) {
        String search = "%" + searchWord + "%";
        List<Book> result = sqlSessionTemplate.selectList(NAMESPACE + "queryBook", search);
        return (ArrayList<Book>) result;
    }
```

```
    public ArrayList<Book> getAllBooks() {
        List<Book> result = sqlSessionTemplate.selectList(NAMESPACE + "getAllBooks");
        return (ArrayList<Book>) result;
    }

    public int addBook(final Book book) {
        return sqlSessionTemplate.insert(NAMESPACE + "addBook", book);
    }

    public Book getBook(final long bookId) {
        return sqlSessionTemplate.selectOne(NAMESPACE + "getBook", bookId);
    }

    public int editBook(final Book book) {
        return sqlSessionTemplate.update(NAMESPACE + "editBook", book);
    }

    public int deleteBook(final long bookId) {
        return sqlSessionTemplate.delete(NAMESPACE + "deleteBook", bookId);
    }
}
```

3.4.3 控制器属性以及方法

BookController：

属性：bookService、lendService

方法：获取时间、查询图书、增加图书、编辑图书、图书详情显示等；

```
package com.library.controller;
import com.library.bean.Book;
import com.library.bean.Lend;
import com.library.bean.ReaderCard;
import com.library.service.BookService;
import com.library.service.LendService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.http.HttpServletRequest;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
@Controller
public class BookController {
    @Autowired
    private BookService bookService;
    @Autowired
    private LendService lendService;
    private Date getDate(String pubstr) {
        try {
            SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");
            return df.parse(pubstr);
        } catch (ParseException e) {
            e.printStackTrace();
            return new Date();
        }
    }
    @RequestMapping("/querybook.html")
    public ModelAndView queryBookDo(String searchWord) {
        if (bookService.matchBook(searchWord)) {
            ArrayList<Book> books = bookService.queryBook(searchWord);
            ModelAndView modelAndView = new ModelAndView("admin_books");
            modelAndView.addObject("books", books);
            return modelAndView;
        } else {
            return new ModelAndView("admin_books", "error", "没有匹配的图书");
        }
    }
    @RequestMapping("/reader_querybook_do.html")
    public ModelAndView readerQueryBookDo(String searchWord) {
```

```
        if (bookService.matchBook(searchWord)) {
            ArrayList<Book> books = bookService.queryBook(searchWord);
            ModelAndView modelAndView = new ModelAndView("reader_books");
            modelAndView.addObject("books", books);
            return modelAndView;
        } else {
            return new ModelAndView("reader_books", "error", "没有匹配的图书");
        }
    }
    @RequestMapping("/admin_books.html")
    public ModelAndView adminBooks() {
        ArrayList<Book> books = bookService.getAllBooks();
        ModelAndView modelAndView = new ModelAndView("admin_books");
        modelAndView.addObject("books", books);
        return modelAndView;
    }
    @RequestMapping("/book_add.html")
    public ModelAndView addBook() {
        return new ModelAndView("admin_book_add");
    }
    @RequestMapping("/book_add_do.html")
    public String addBookDo(@RequestParam(value = "pubstr") String pubstr, Book book,
RedirectAttributes redirectAttributes) {
        book.setPubdate(getDate(pubstr));
        if (bookService.addBook(book)) {
            redirectAttributes.addFlashAttribute("succ", "图书添加成功！");
        } else {
            redirectAttributes.addFlashAttribute("succ", "图书添加失败！");
        }
        return "redirect:/admin_books.html";
    }
    @RequestMapping("/updatebook.html")
    public ModelAndView bookEdit(HttpServletRequest request) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        Book book = bookService.getBook(bookId);
        ModelAndView modelAndView = new ModelAndView("admin_book_edit");
```

```
        modelAndView.addObject("detail", book);
        return modelAndView;
    }
    @RequestMapping("/book_edit_do.html")
    public String bookEditDo(@RequestParam(value = "pubstr") String pubstr, Book book,
RedirectAttributes redirectAttributes) {
        book.setPubdate(getDate(pubstr));
        if (bookService.editBook(book)) {
            redirectAttributes.addFlashAttribute("succ", "图书修改成功！");
        } else {
            redirectAttributes.addFlashAttribute("error", "图书修改失败！");
        }
        return "redirect:/admin_books.html";
    }
    @RequestMapping("/admin_book_detail.html")
    public ModelAndView adminBookDetail(HttpServletRequest request) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        Book book = bookService.getBook(bookId);
        ModelAndView modelAndView = new ModelAndView("admin_book_detail");
        modelAndView.addObject("detail", book);
        return modelAndView;
    }
    @RequestMapping("/reader_book_detail.html")
    public ModelAndView readerBookDetail(HttpServletRequest request) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        Book book = bookService.getBook(bookId);
        ModelAndView modelAndView = new ModelAndView("reader_book_detail");
        modelAndView.addObject("detail", book);
        return modelAndView;
    }
    @RequestMapping("/admin_header.html")
    public ModelAndView admin_header() {
        return new ModelAndView("admin_header");
    }
    @RequestMapping("/reader_header.html")
    public ModelAndView reader_header() {
```

```
        return new ModelAndView("reader_header");
    }
    @RequestMapping("/reader_books.html")
    public ModelAndView readerBooks(HttpServletRequest request) {
        ArrayList<Book> books = bookService.getAllBooks();
        ReaderCard readerCard = (ReaderCard) request.getSession().getAttribute("readercard");
        ArrayList<Lend> myAllLendList = lendService.myLendList(readerCard.getReaderId());
        ArrayList<Long> myLendList = new ArrayList<>();
        for (Lend lend : myAllLendList) {
            // 是否已归还
            if (lend.getBackDate() == null) {
                myLendList.add(lend.getBookId());
            }
        }
        ModelAndView modelAndView = new ModelAndView("reader_books");
        modelAndView.addObject("books", books);
        modelAndView.addObject("myLendList", myLendList);
        return modelAndView;
    }
}
```

LendController：（这部分我只负责了 bookService）

属性:bookService、lendService

方法：删除图书、借书信息表、图书归还、删除。

```
package com.library.controller;
import com.library.bean.ReaderCard;
import com.library.service.BookService;
import com.library.service.LendService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.http.HttpServletRequest;
@Controller
public class LendController {
    @Autowired
```

```
    private LendService lendService;

    @Autowired
    private BookService bookService;
    @RequestMapping("/deletebook.html")
    public String deleteBook(HttpServletRequest request, RedirectAttributes redirectAttributes) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        if (bookService.deleteBook(bookId)) {
            redirectAttributes.addFlashAttribute("succ", "图书删除成功！");
        } else {
            redirectAttributes.addFlashAttribute("error", "图书删除失败！");
        }
        return "redirect:/admin_books.html";
    }
    @RequestMapping("/lendlist.html")
    public ModelAndView lendList(HttpServletRequest request) {
        ModelAndView modelAndView = new ModelAndView("admin_lend_list");
        modelAndView.addObject("list", lendService.lendList());
        return modelAndView;
    }
    @RequestMapping("/mylend.html")
    public ModelAndView myLend(HttpServletRequest request) {
        ReaderCard readerCard = (ReaderCard) request.getSession().getAttribute("readercard");
        ModelAndView modelAndView = new ModelAndView("reader_lend_list");
        modelAndView.addObject("list", lendService.myLendList(readerCard.getReaderId()));
        return modelAndView;
    }
    @RequestMapping("/deletelend.html")
    public String deleteLend(HttpServletRequest request, RedirectAttributes redirectAttributes) {
        long serNum = Long.parseLong(request.getParameter("serNum"));
        if (lendService.deleteLend(serNum) > 0) {
            redirectAttributes.addFlashAttribute("succ", "记录删除成功！");
        } else {
            redirectAttributes.addFlashAttribute("error", "记录删除失败！");
        }
        return "redirect:/lendlist.html";
```

```
    }
    @RequestMapping("/lendbook.html")
    public String bookLend(HttpServletRequest request, RedirectAttributes redirectAttributes) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        long readerId = ((ReaderCard) request.getSession().getAttribute("readercard")).getReaderId();
        if (lendService.lendBook(bookId, readerId)) {
            redirectAttributes.addFlashAttribute("succ", "图书借阅成功！");
        } else {
            redirectAttributes.addFlashAttribute("succ", "图书借阅成功！");
        }
        return "redirect:/reader_books.html";
    }
    @RequestMapping("/returnbook.html")
    public String bookReturn(HttpServletRequest request, RedirectAttributes redirectAttributes) {
        long bookId = Long.parseLong(request.getParameter("bookId"));
        long readerId = ((ReaderCard) request.getSession().getAttribute("readercard")).getReaderId();
        if (lendService.returnBook(bookId, readerId)) {
            redirectAttributes.addFlashAttribute("succ", "图书归还成功！");
        } else {
            redirectAttributes.addFlashAttribute("error", "图书归还失败！");
        }
        return "redirect:/reader_books.html";
    }
}
```

4 系统测试

4.1 系统测试方法

1、单元测试

单元测试是最微小规模的测试，测试的是某个功能或代码块。

2、动态测试

通过运行软件来检验软件的动态行为和运行结果的正确性。

具体操作就是输入相应的测试数据，检查输出结果和预期结果是否相符的过程。

3、集成测试

将系统的各个部件的联合测试，验证他们能否在一起共同工作并没有冲突。

4、系统测试（System Testing）：

系统测试是将整个软件系统看做一个整体进行测试，包括对功能、性能，以及软件所运行的软硬件环境进行测试

4.2 系统测试

1、增加图书信息

增加图书“催眠师手记”信息

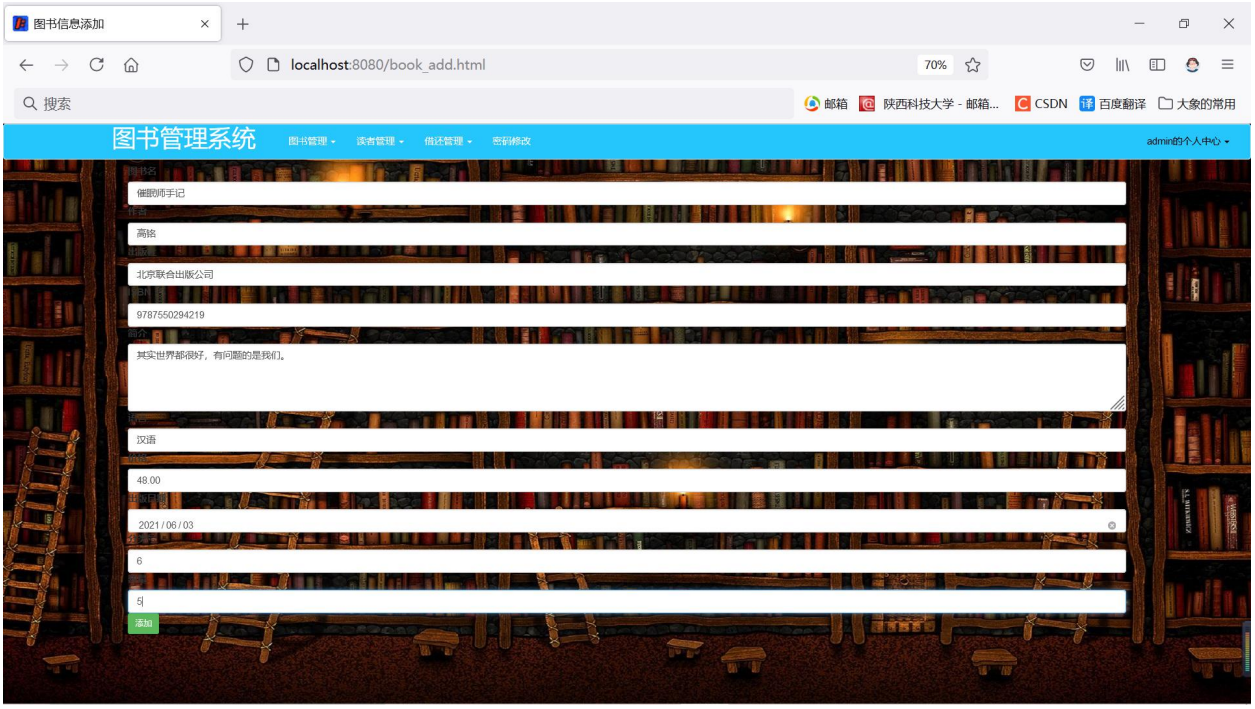

图 1 增加图书信息

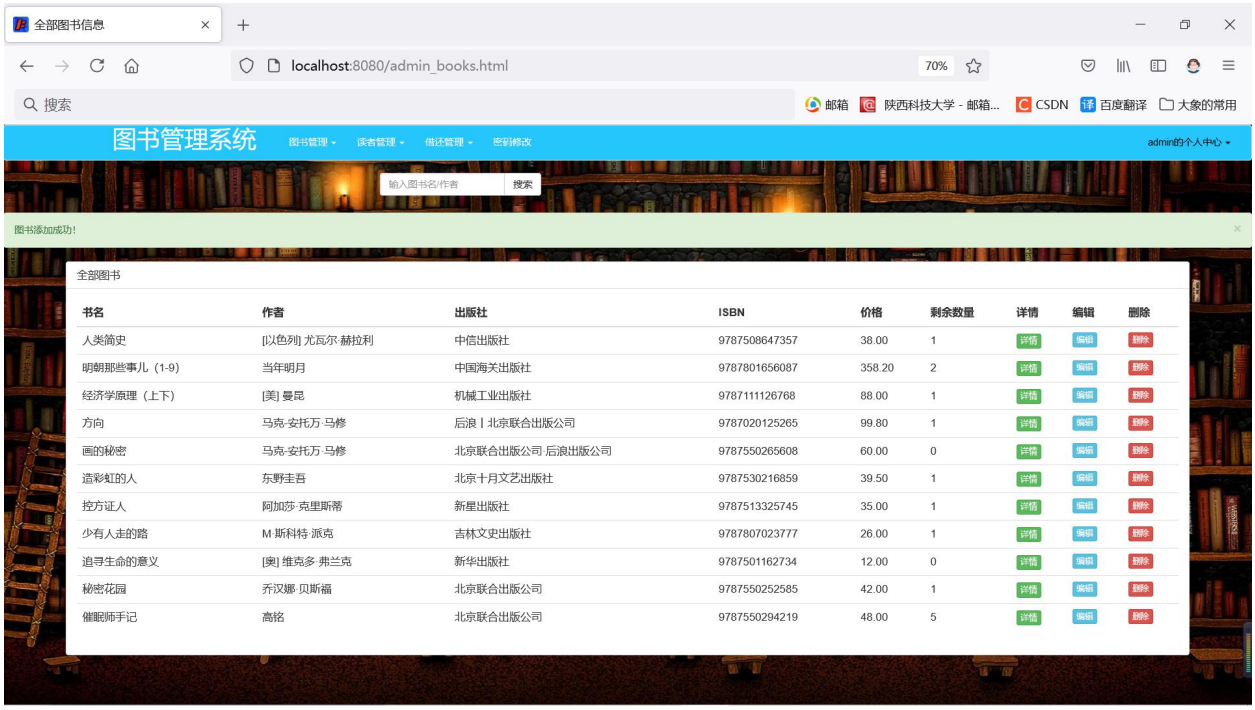

图 2 增加成功

2、删除信息

对图书“催眠师手记”信息进行删除

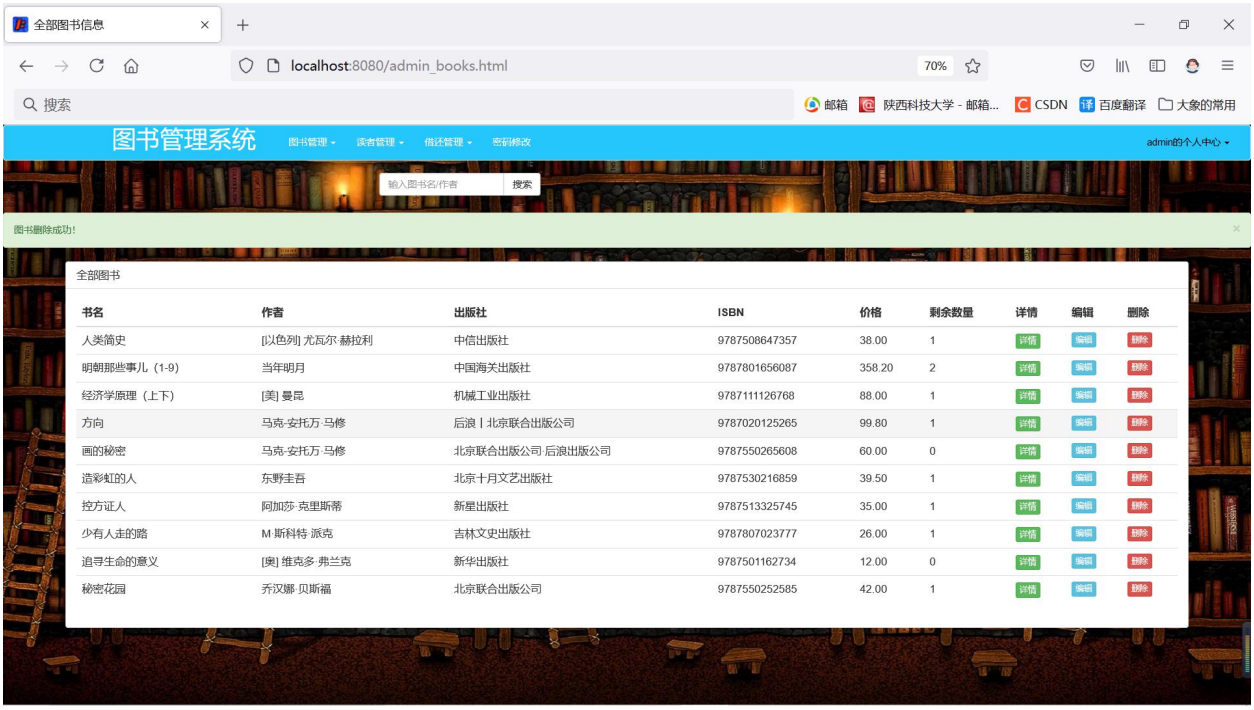

图 3 删除图书信息

3、查询图书信息

（1）查询全部图书

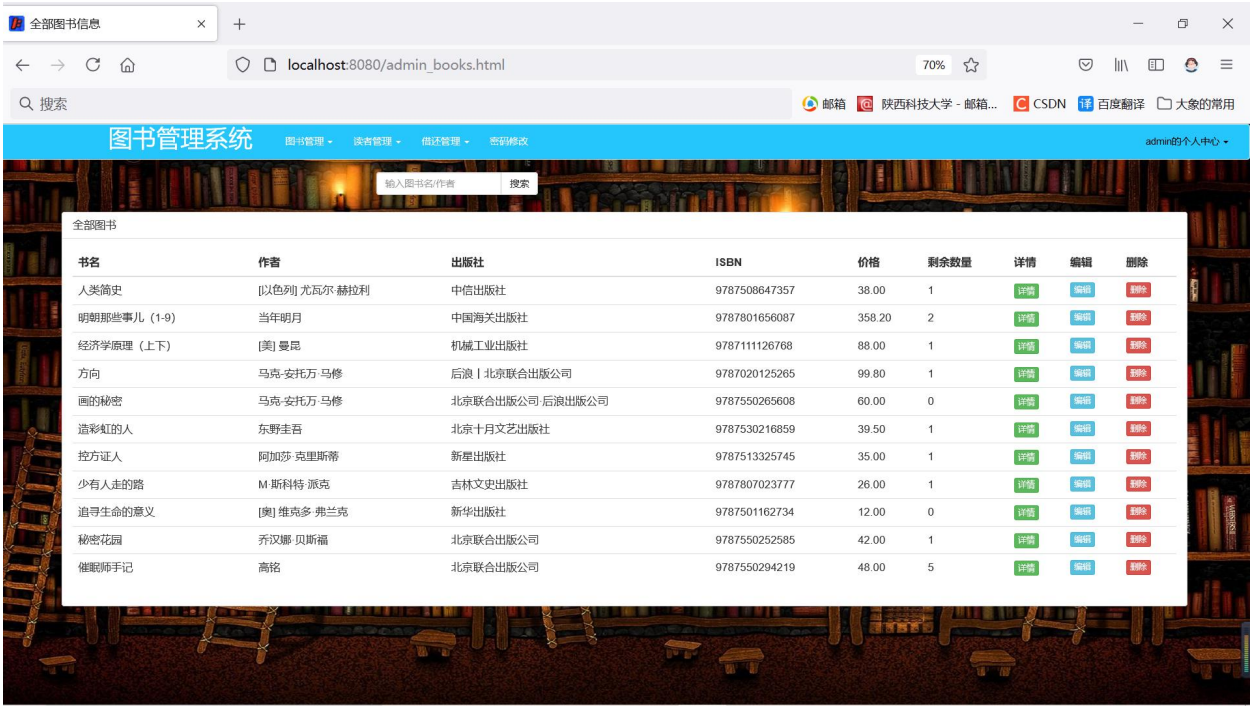

图 4 查询全部图书

（2）根据作者“马克·安托万·马修”查询图书

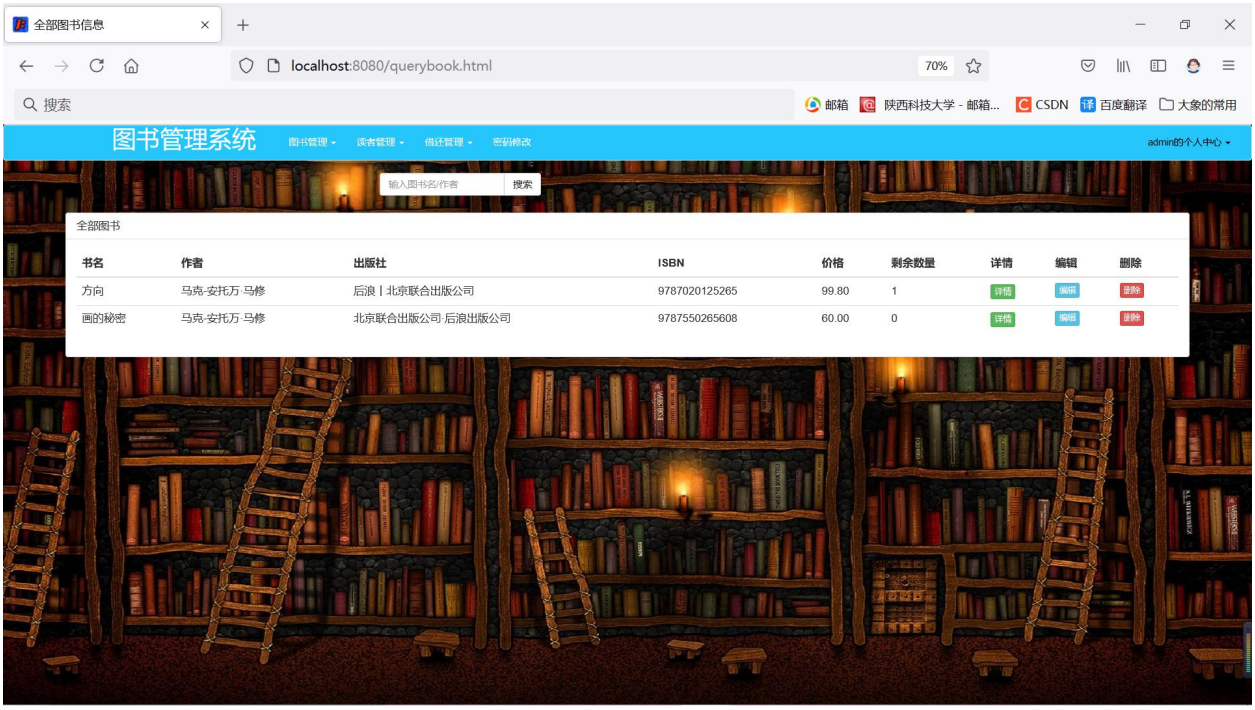

图 5 根据作者查询图书界面

（3）根据书名“秘密花园”查询图书

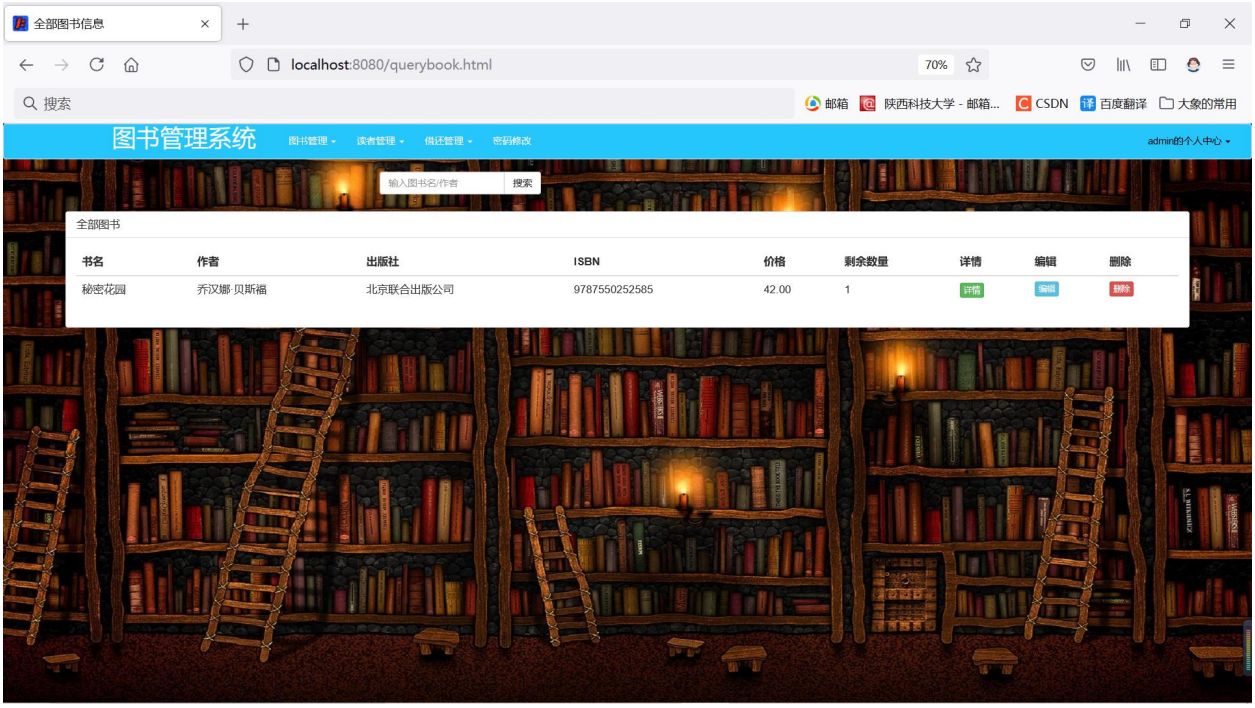

图 6 根据书名查询图书界面

4、修改图书信息

将图书“明朝那些事儿”的价格从 358.20 修改为 35.20.

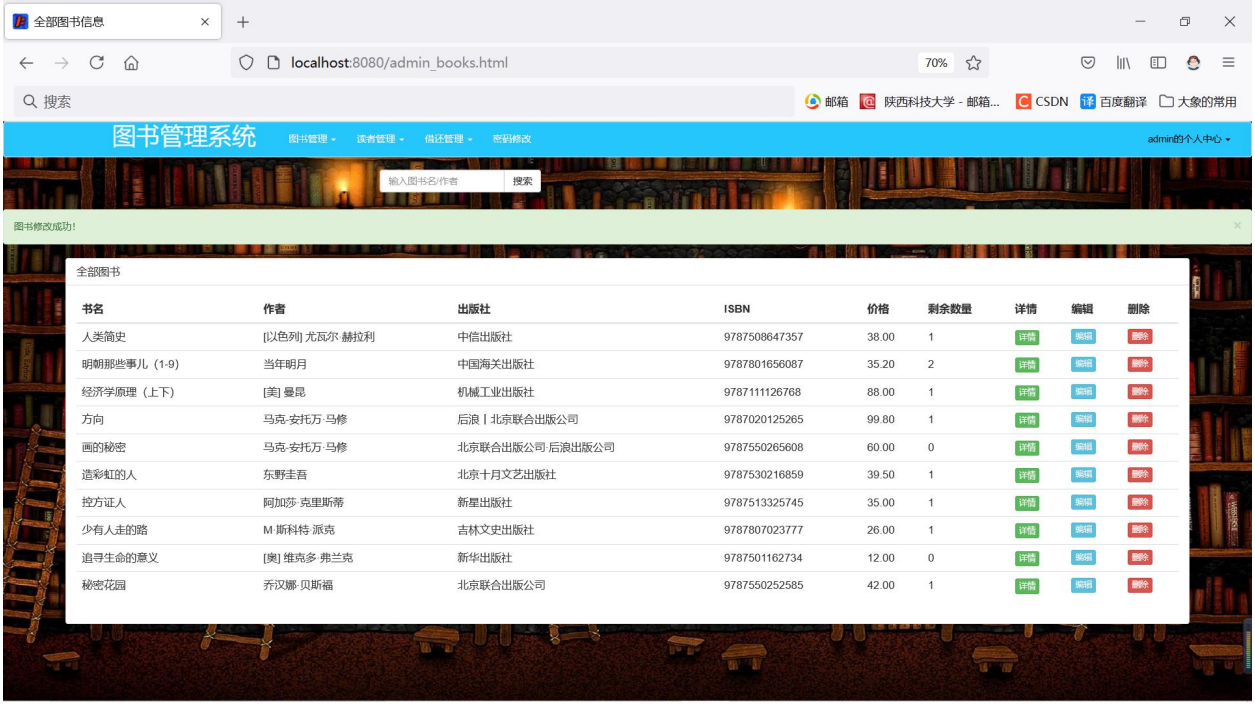

图 7 修改图书信息

4.3 系统测试结果

管理员能够实现对图书信息的增删改查。

5 总结

5.1 系统工作总结

我们小组实现的在线书籍管理系统，主要模块实现包括：借阅日志模块，管理员对借阅日志进行管理；读者信息模块，管理员对读者信息进行增删改查；个人信息模块，用户对自己的信息进行修改；编辑图书模块，管理员对图书进行增删改查；借阅图书模块，读者可以借阅和归还图书；登录及布局模块，包括对登录界面、页面顶部和主体的功能和布局的设计

通过最终的各个模块的集成和测试，最终实现了一个较为美观的书籍管理系统，也实现了书籍管理的基本完备功能，该系统操作简单、界面简洁，实现了管理者和用户两种用户的登录模式。创新之处在于：前端页面使用了 Bootstrap 简化了代码量；框架使用了 mybatis 持久性框架免除了大量 jdbc 代码以及设置参数获取结果集的操作；实现了图书信息的模糊查询；可以通过日历选择框输入日期。达到了使图书管理工作规范化、系统化、程序化；信息处理及时、准确、有效的；用户方便快捷的进行图书的查询以及借阅操作的目标。

这是我第一次完整开发出一个功能。虽然是和组员合作完成，成就感还是满满。整个开发的过程也是一个不断的学习过程，包括环境的配置，各种框架的搭建等等。比如这次开始设计前端页面时，才知道 Bootstrap 框架，在它的官网中就有一些可以直接使用的框架，在它的基础上进一步美化，这样就提高了开发效率，把更多的时间用在后端的逻辑处理上...查阅了各种资料、博客论文，同时过程中也问了很多的同学。最大的收获就是对

web 程序的开发有了更深层次的理解，对框架技术的使用也有了不一样的体会，对自己的编码能力以及逻辑思维能力也有了一些提升。

5.2 存在的不足及改进

1.在进行书籍信息的删除时，目前是删除之后更新界面，并在界面处直接提示删除成功，后续应该修改为出现弹窗来提示用户是否确认删除，确认之后再更新。

2.修改图书信息的时候，每次都需要重新选择出版日期，而不是像其它属性按需修改，后续应该把它也设置成上一次保存的信息，需要修改再选中修改。

3.查询图书的方式只有通过书名和作者，比较单一，后续应再添加其他查询方式。

参 考 文 献

[1] 陈红,基于 SSM 框架的智能 web 管理系统的研发设计[D],北京化工大学。
[2] 谢萍,基于 Web 的图书管理系统的设计与实现[D],成都电子科技大学。
[3] 常建国, Java Web 典型模块与项目实战大全[M],北京:清华大学出版社。
[4] 沈泽刚，Java Web 编程技术[M],北京:清华大学出版社。
[5] 赵志成，基于 J2EE 协同办公管理系统的设计与实现[J]，哈尔滨师范大学。
[6] 陈夫真.基于 SSM 的某高校图书馆管理信息系统的设计与实现[D]，苏州大学。