目 录

第一章 简介

1.1 项目内容

在学习了有关 Java 的基础知识，以及有关 JavaWeb 的相关知识过后，为检验本人的学习效果，故打算做一个基于 JavaWeb 以及 Tomcat10 的网站项目。经过反复思考过后，本人决定在所学知识基础上，实现一个“网上书城”在线网站。

注：本人在哔哩哔哩录制的一份效果视频:【大二学生 JavaWeb 课程设计代码效果展示-哔哩哔哩】 https://b23.tv/XUdYmkq

1.2 项目目标

项目实现目标：包括但不限于：

（1）用户登录、注册（包括账号密码注册，以及基本信息完善）、注销账户功能；

（2）商城首页：对书籍信息的显示与“增删改查”功能，图片显示与图片上传、存储功能。（书籍增加功能包括：弹窗添加以及页面添加）。

（3）用户界面：显示用户注册时的基本信息，以及添加退出、完善个人信息的按钮。

（4）购物车：用户能够在首页添加书籍至购物车，购物车里面能够查看相关书籍信息，以及对书籍数目的增减。

（5）同时为提升用户体验，所有操作在进行完成后，应当自动刷新界面。当用户再次登陆时，无需刷新，自动加载已有购物车数据。在用户进行一些涉及个人账户安全信息的操作时，将进行初级的验证（账号、密码）以及再次确认是否进行该操作。

第二章 功能需求分析

2.1 登录、注册、注销账户功能

用户在进入书城之前就要要求登录，如果没有登录将不会允许访问书城内部内容。同时，如果用户在登录时未输入正确的验证码、账号、密码都将判定登陆失败，并报相应的提示。

如果用户未注册相关账户，则要求用户进行账户注册。当用户成功注册账户后，会要求完善个人信息。当然，用户也可以选择此时不进行个人信息完善。如果用户未完善个人信息，当其每次登录时，都会要求用户进行个人信息完善，如果用户仍未完善个人信息，当然也是可以正常登录的。进入书城主界面后，用户可以随时在用户中心完善个人信息。

如果用户不再想要当前账户，可以选择注销账户，注销账户时系统会进行提示，确认用户是否真的想要注销账号，并且注销时会要求用户输入密码进行验证，以此初步判断是否为本人操作。

2.2 书籍信息的显示与“增删改查”功能

在书城主界面，用户将看到以下内容：书籍编号（id）、书籍名称、书籍价格、书籍作者、书籍图片、书籍描述、操作（包括：加入购物车、查看具体内容、修改、删除），同时，书城上方还提供书籍查询按钮，以及页面添加与弹窗添加两者添加书籍的方式供用户选择。

当每一页的书籍数目达到 8 个时，如果有超出的书籍信息则将在下一页显示，同时，界面左下角提供有全选、反选、批量删除按钮，方便用户对书籍进行批量操作。

2.3 用户中心设置功能

在用户中心，会根据用户登录的信息，加载对应用户的个人信息，同时允许用户更改自己的头像，用户可以上传自己的本地图片进行头像更换。如果用户想要完善或者修改个人信息，也可以在该界面下进行。当用户对用户中心的个人数据进行修改后，会自动刷新界面，加载最新数据。

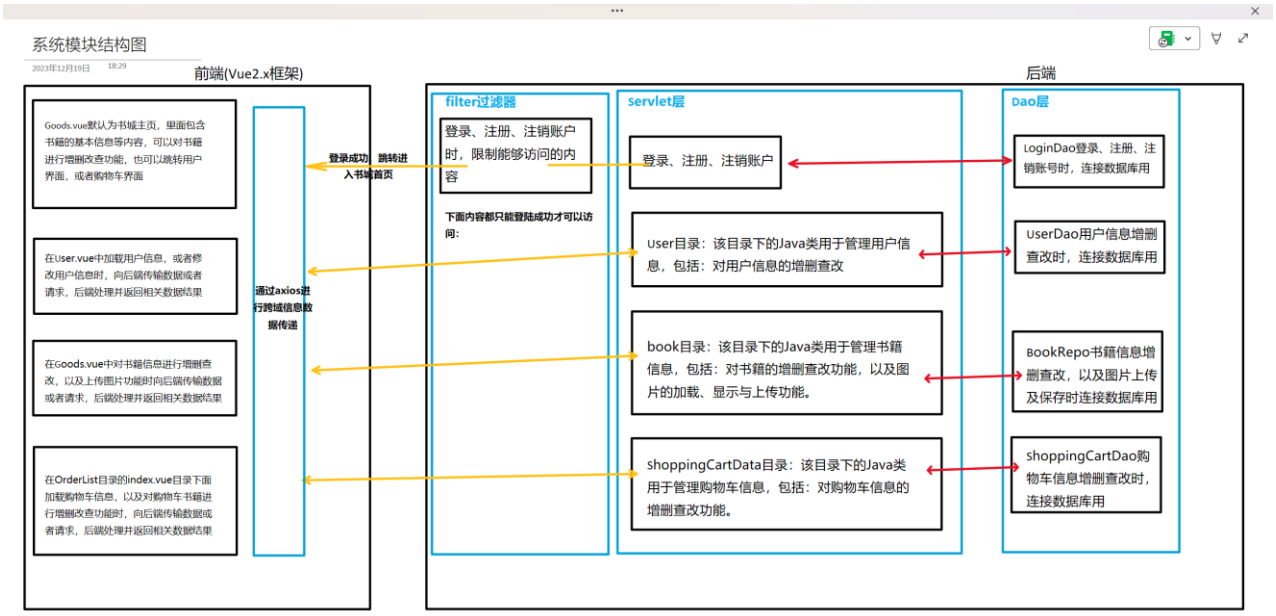
2.4 购物车功能

当用户在主界面选择想要的书籍，并点击“加购”按钮过后，数据会保存一份到数据库，然后购物车实时获取数据库里面的信息，实现实时展示最新信息。用户即可在购物车界面查看自己所选购的书籍。

当然。用户也可以在购物车界面更改书籍购买数量，更改后将自动刷新界面，并展示最新购物车数据信息。用户也可以点击查看按钮查看书籍具体的一些信息。如果购物车的书籍过多，用户也可以通过查询按钮进行查询操作，输入书籍名称，将为用户找到相应的书籍信息。最后，用户如果不需要某些书籍时，可以通过删除按钮删除不需要的书籍，在删除之前，用户会被要求进行再次确认，以免造成误删情况发生。

第三章 系统结构设计

3.1 系统模块结构图



3.2　登录、注册、注销账户模块

在后端新建 user 目录，用于实现有关用户登录、注册、注销账户的接口。同时，在 Dao 层新建 Java 实体类 LoginDao，用于实现 user 目录下，servlet 层与 MySQL 数据库的连接。

在 MySQL 数据库中，新建表 users。其中包含字段： account（varchar、自动递增、主键、用户注册时的账号名）、password（varchar、登录密码）。

在前端的 Login.vue 文件里面，借助 Html 与 Css 语言完成登录界面的主要显示框架，同时借助 vue2 让各个小模块组件化，便于实现网页的流畅切换。通过 axios 库，实现前端相关 button 按钮与后端相关接口的连接。进而实现前端向后端传输数据或请求，后端做相应处理并返回结果数据。

（1）如果登录失败，用户将不会进入商城界面，要求重新登录，或者注册账号。

（2）注册账户包括：第一步：注册账号、密码；第二步：完善个人信息。用户如果只是注册了账户而没有完善个人信息，在登录时会再要求完善。如果仍然没有完善，也会登录成功，用户以后也可以通过用户界面进行个人信息的完善。

（3）如果登陆成功，并且个人信息完善（如果没有完善，如同第（2）步所示，仍然正确登录，以后可以在个人中心再随时进行完善）将跳转进入商城首页。

（4）用户不再需要账号时，可以通过注销账户按钮进行账户注销，账户注销后将删除数据库中与该用户有关的数据信息记录。

3.3　书籍管理模块

后端新建 book 目录，用于实现有关书籍的加载、增删改查功能的接口。同时，在 Dao 层下新建一个 Java 实体类，用于实现 book 目录下，servlet 层与 MySQL 数据库的连接。

在 MySQL 数据库中，新建表 book。其中包含字段：id（int、自动递增、主键）、name（varchar、书籍名字）、author（varchar、书籍作者）、imageurl（varchar、图片地址）、price（decimal、书籍价格）、content（varchar、书籍简介）、num（int、书籍数量）。

前端首先，为实现跨域的数据请求，我们在 vue.config.js 中进行相关配置。包括：允许跨域、跨域目标路径、重写路径等。同时，为了能够访问 Goods.vue 对应的界面，我们进行路由配置，在 router 目录的 index.js 文件中导入 Goods.vue 的文件路径，并为其重命名访问名称。

在前端的 Goods.vue 文件里面，借助 Html 与 Css 语言完成商城界面的主要显示框架，同时借助 vue2 让各个小模块组件化，便于实现网页的流畅切换。通过 axios 库，实现前端相关 button 按钮与后端相关接口的连接。进而实现前端向后端传输数据或请求，后端做相应处理并返回结果数据。

3.4　用户信息模块

在后端的 user 目录中加入一些 Java 类，用于实现有关用户信息的加载、增删改查功能的接口。同时，在 Dao 层新建 Java 实体类 UserDao，用于实现 user 目录下，servlet 层与 MySQL 数据库的连接。

在 MySQL 数据库中，新建表 usersdata。其中包含字段：userid（int、自动递增、主键）、account（varchar、用户注册时的账号名）、password（varchar、登录密码）、nickname（varchar、用户昵称）、email（varchar、用户邮箱）、phone（varchar、用户电话）、address（varchar、用户地址）、firstname（varchar、用户姓氏）、lastname（varchar、用户名字）、city（varchar、居住城市）、country（varchar、居住国家）、introduce（varchar、用户自我描述（简介））。

前端首先，为了能够访问 User.vue 对应的界面，我们进行路由配置，在 router 目录的 index.js 文件中导入 Goods.vue 的文件路径，并为其重命名访问名称。

在前端的 User.vue 文件里面，借助 Html 与 Css 语言完成用户界面的主要显示框架，同时借助 vue2 让各个小模块组件化，便于实现网页的流畅切换。通过 axios 库，实现前端相关 button 按钮与后端相关接口的连接。进而实现前端向后端传输数据或请求，后端做相应处理并返回结果数据，进而完成对用户数据的增删改查功能。

3.5　购物车管理模块

在后端新建 ShoppingCart 目录，用于实现有关购物车数据的加载、增删改查功能的接口。同时，在 Dao 层新建 Java 实体类 ShoppingCartDao，用于实现 ShoppingCart 目录下，servlet 层与 MySQL 数据库的连接。

在 MySQL 数据库中，新建表 shoppingcart。其中包含字段：id（int、自动递增、主键）、name（varchar、书籍名字）、author（varchar、书籍作者）、imageurl（varchar、图片地址）、price（decimal、书籍价格）、content（varchar、书籍简介）、num（int、书籍数量）。

前端首先，为了能够访问 OrderList 目录下面的 index.vue 对应的界面，我们进行路由配

置，在 router 目录的 index.js 文件中导入 OrderList 目录下面的 index.vue 的文件路径，并为其重命名访问名称。

在前端的该 index.vue 文件里面，借助 Html 与 Css 语言完成购物车界面的主要显示框架，同时借助 vue2 让各个小模块组件化，便于实现网页的流畅切换。通过 axios 库，实现前端相关 button 按钮与后端相关接口的连接。进而实现前端向后端传输数据或请求，后端做相应处理并返回结果数据，进而完成对购物车数据的增删改查功能。

第四章 系统实现

4.1 系统实现介绍

项目实现手段：

项目前端采用 Vue2.x 框架，项目后端采用 Java17+Tomcat10，数据库选用 MySQL8.1，前后端的跨域请求借助 axios 网络请求库，通过 json 数据格式进行前后端数据交互。

通过以上手段，项目预期目标基本实现，效果基本达到预定效果，下面将介绍本系统的关键功能，关键技术，技术难点实现结果。

4.1.1 关键功能

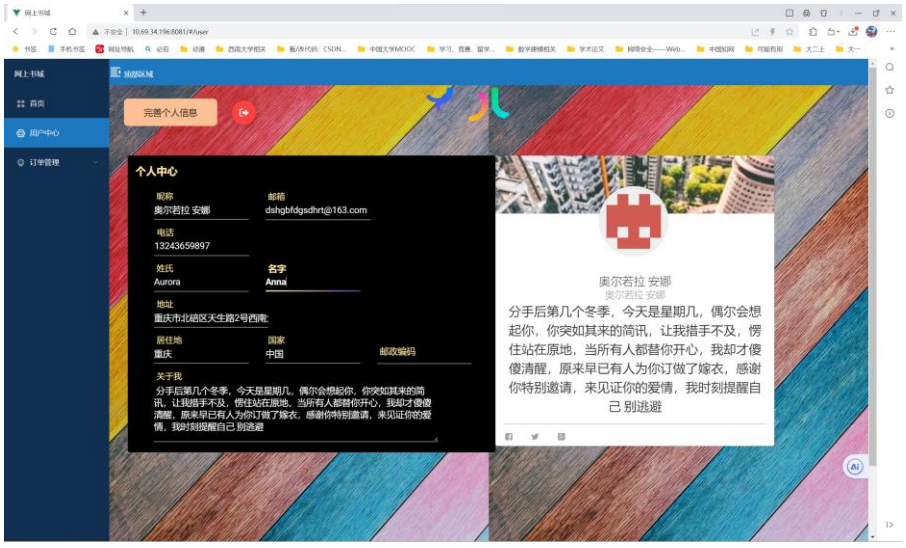
1.用户登录结合验证码，注册、注销功能

用户在前端进行登录时，点击登录按钮，后端 LoginServlet.java 将对用户登录请求进行检验。首先如果验证码不正确，将不会再检验账号密码是否正确。如果验证码正确，程序将会把账号、密码数据传入 Dao 层，与数据库里面的数据进行比对，如果输入错误，或者不存在账户，程序都将返回相关信息，同时用户重新输入，或者注册账户。

在注册账户时，我们会要求用户仔细确认密码，并完善个人信息。如果个人信息未完善，以后每次登陆时，都将提示一下用户去完善个人信息，当然用户没有完善个人信息，也将正常登录。用户可以随时在个人中心完善个人信息。

在注销账户时，会反复提醒用户是否确实要注销账户，最后在进行账户注销工作。

登陆后，用户中心界面如下：

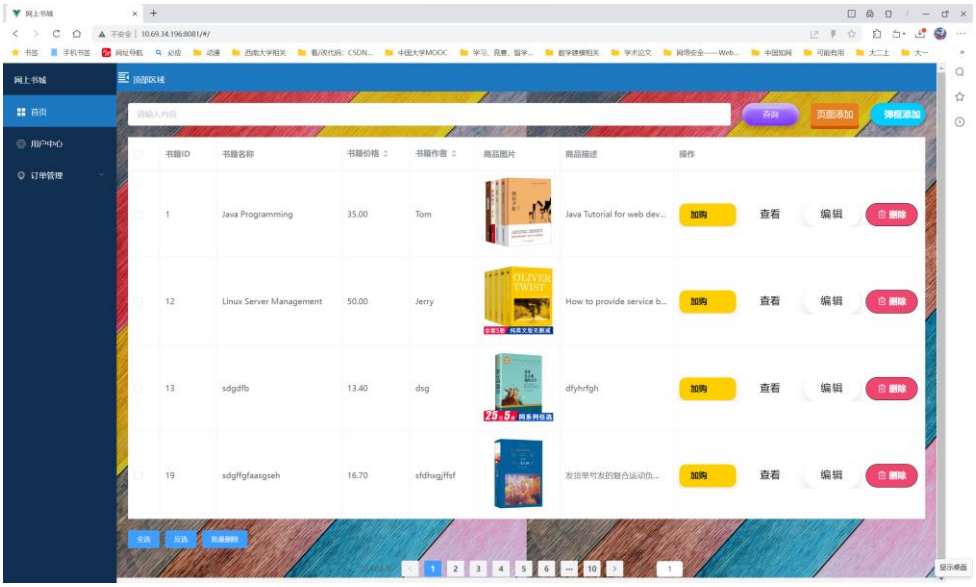


2. 主页面对书籍的“增删查改”功能，以及图片上传，保存功能

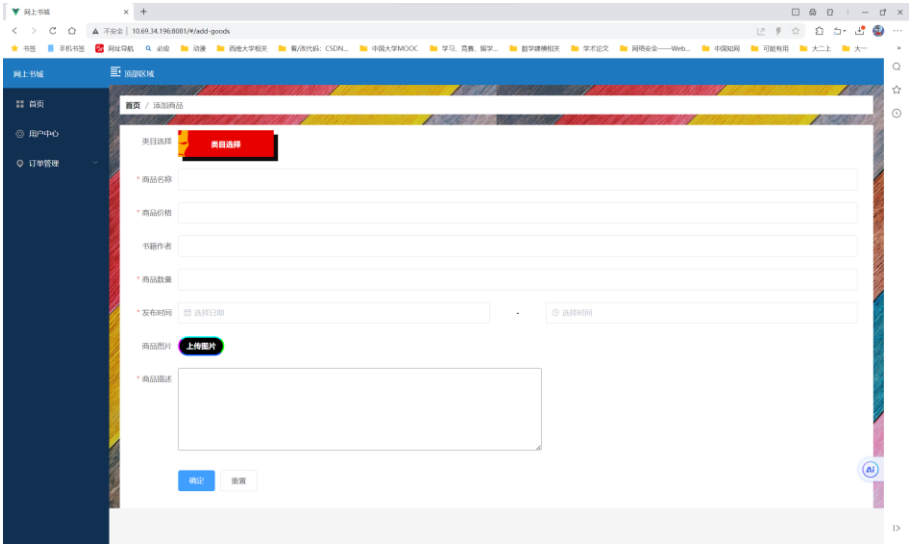
主页面显示如下图，里面包含了对书籍进行“增删查改”功能的相关按钮，提供了多元化的选择以满足用户需求。同时，由于 vue 的路由性质，会使得界面跳转变得十分流畅，完

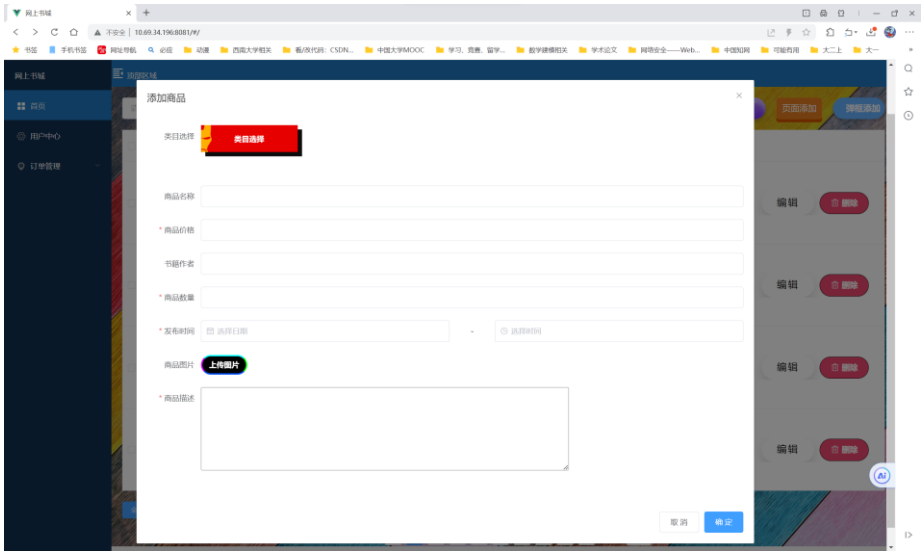
全不必担心卡顿问题。

这个优点还体现在，修改、删除、添加、查找商品后，点击按钮，界面会实时刷新，用户不会看见明显的切换界面时的刷新效果（所以说，界面切换、更新十分流畅），也不需要用户自己去点击刷新按钮刷新界面（所以说，这种方法的可行性极高，用户友好型界面）

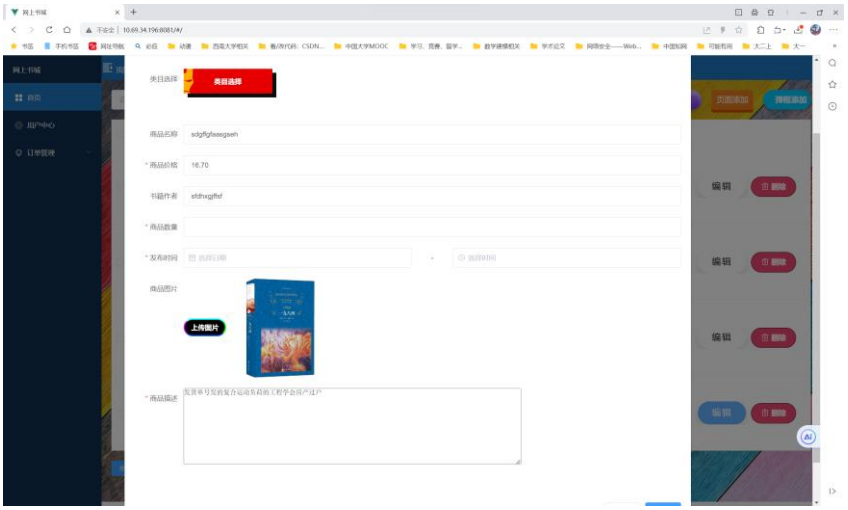


（1）添加书籍时，可以选择页面添加，弹窗添加（一个是打开新界面。一个是打开一个弹窗）：



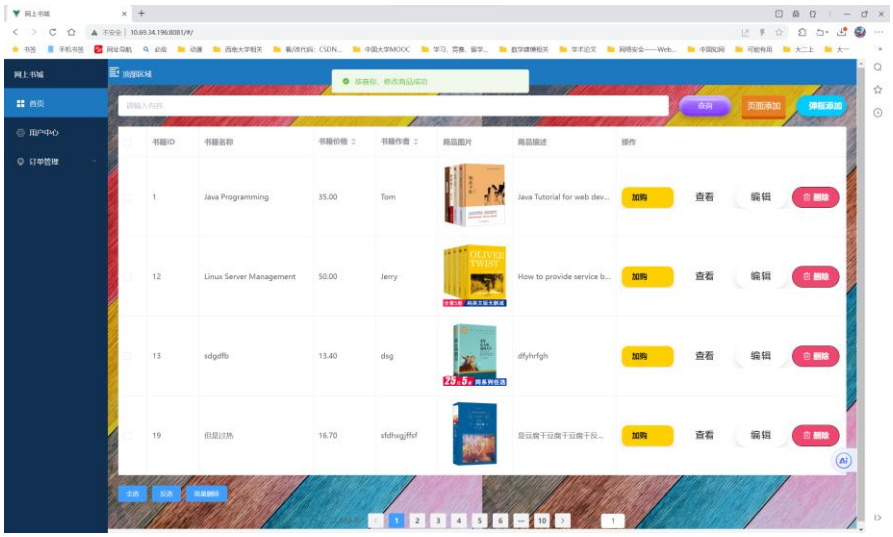


（2）修改商品时，会弹出弹窗，让用户根据需求修改：



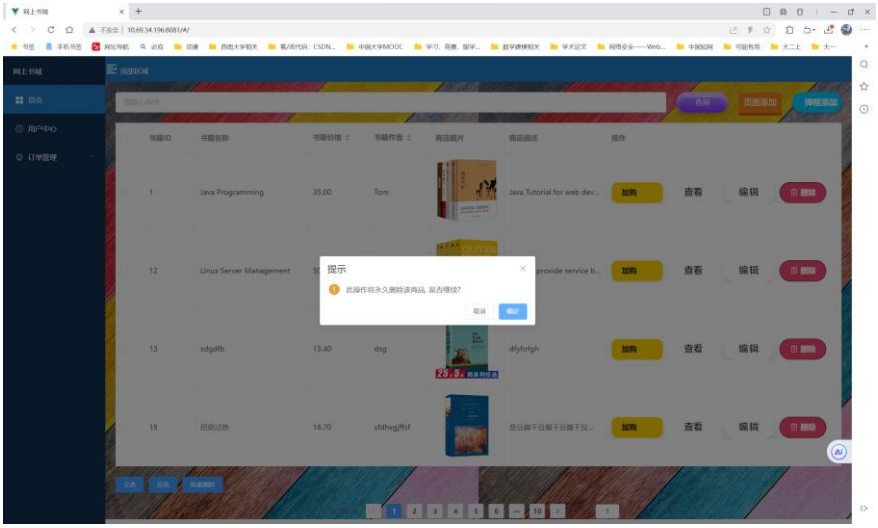
例如，改变书名，及简介：

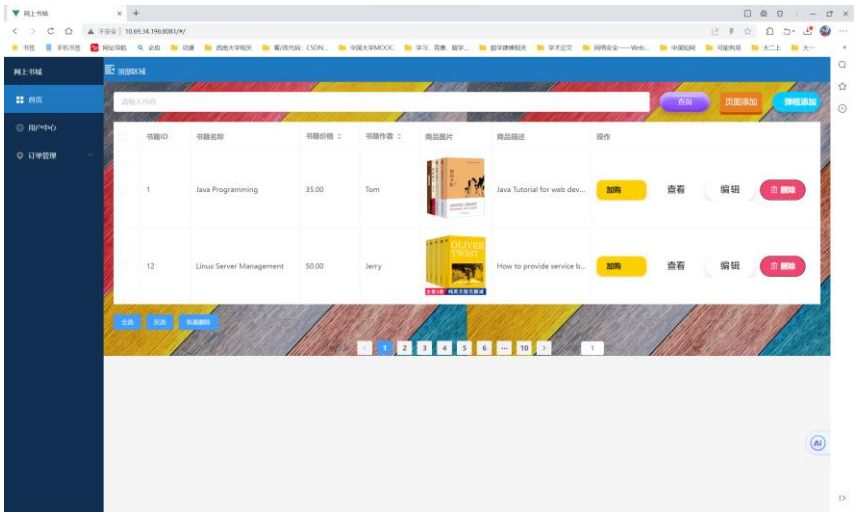
修改信息后（无论是否成功），上面会有弹窗提示；如果修改成功，会发现下面相关书籍的信息发生变化。



（3）删除书籍会有如下效果：

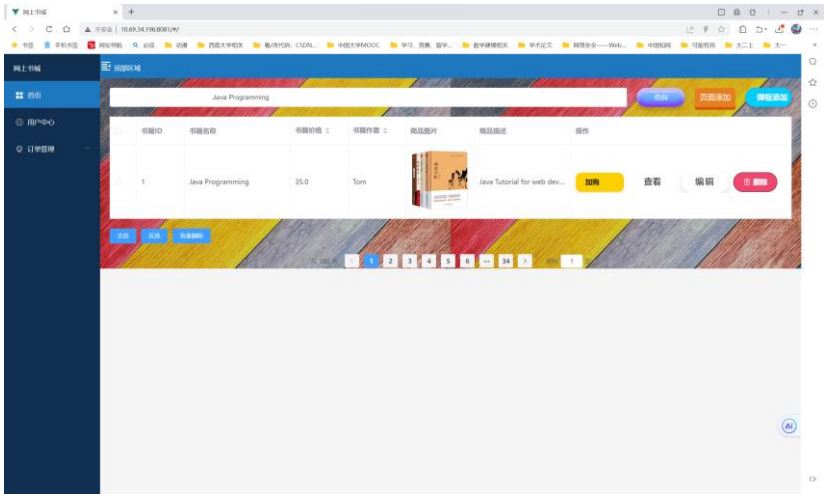
会先进行询问，而不是直接删除，这有助于防止误删书籍的情况发生。





（4）查找书籍时，会有以下效果：

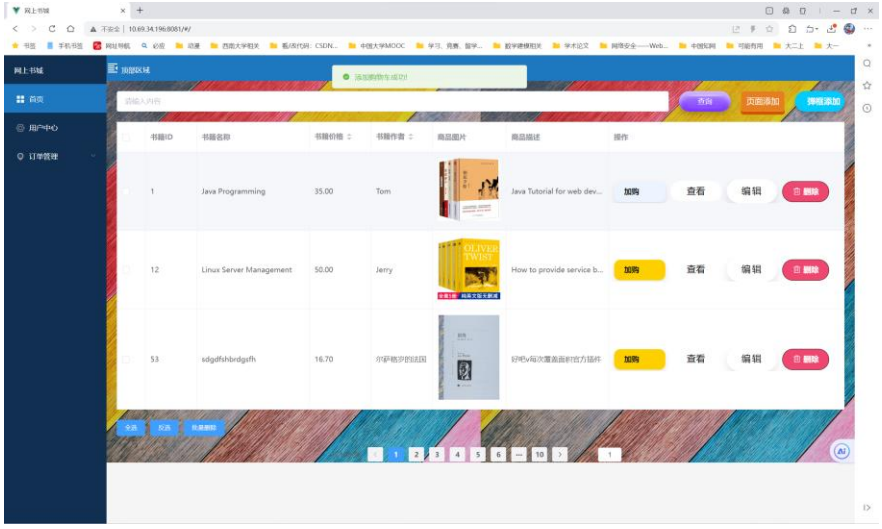
查询书籍时，它会自动处理多余的空格，可以说十分方便。



3. 购物车功能

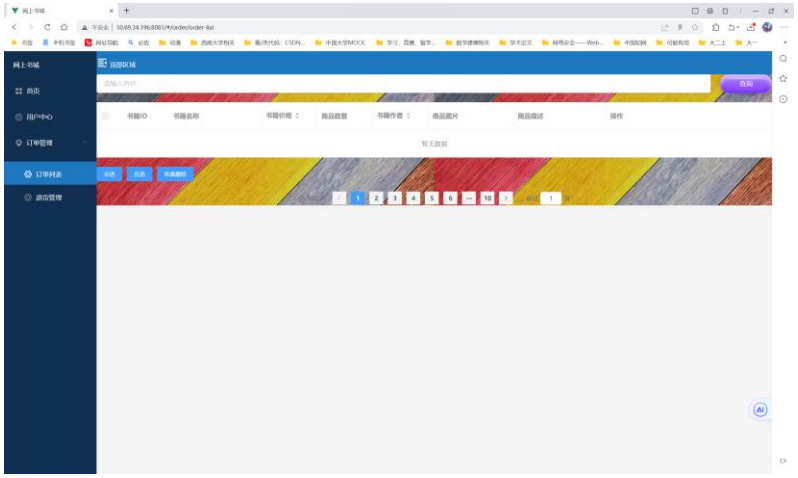
当用户从首页选择书籍后，可以点击加购，让想要的书籍进入购物车，如下：

（1）首先，点击“加购”按钮后，如果添加购物车成功，则会有弹窗提示：

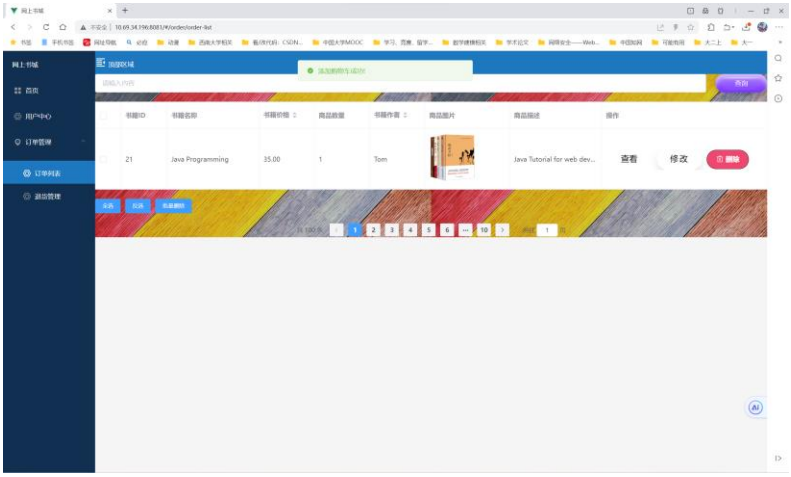


（2）购物车的界面变化如下：

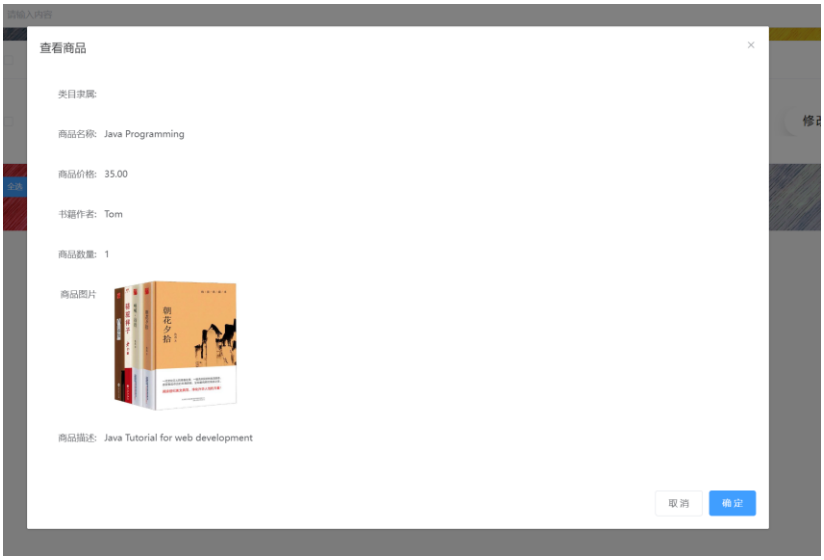
没有添加书籍时：



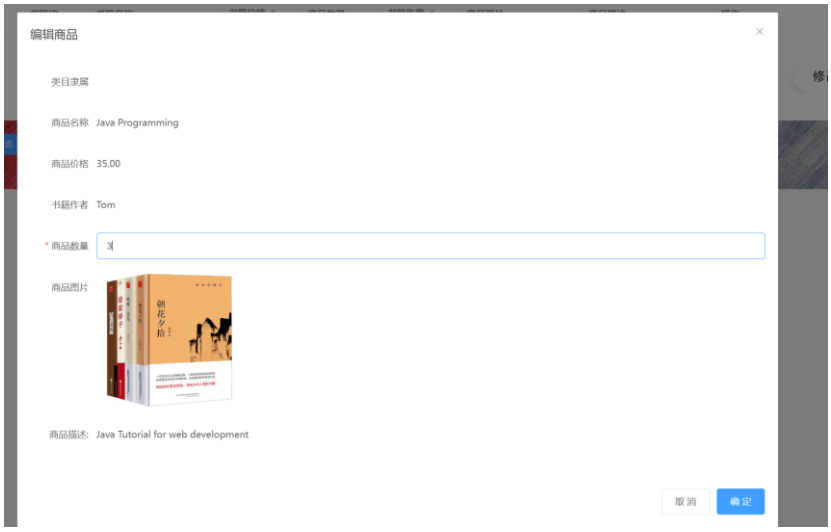
添加书籍后：

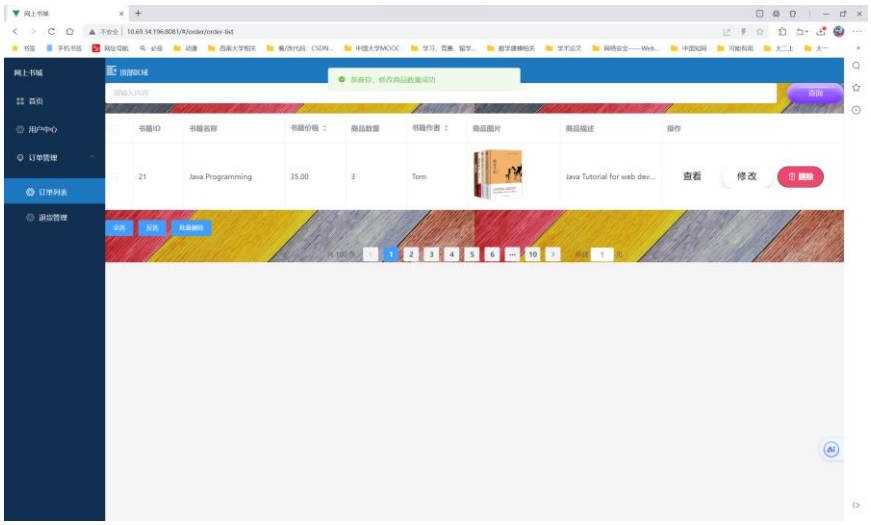


（3）而且，用户可以在购物车界面查看自己所买书籍的详细信息：

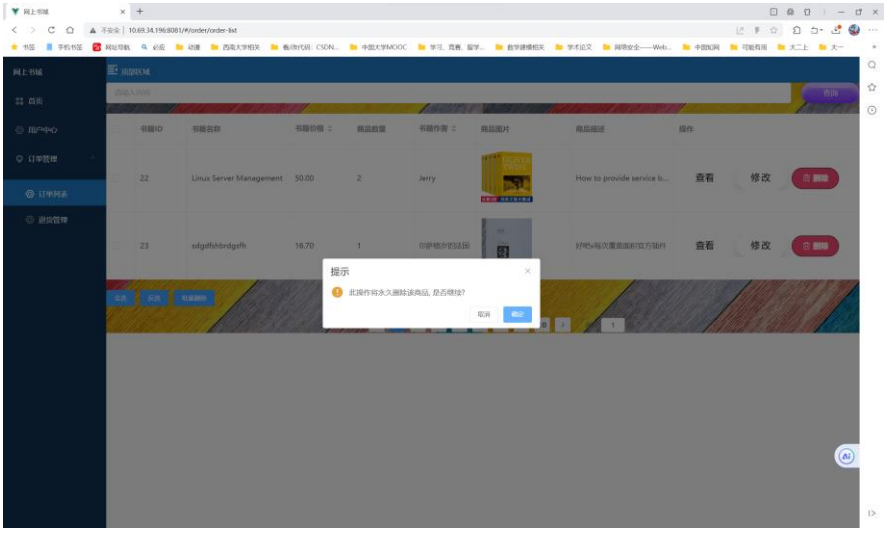


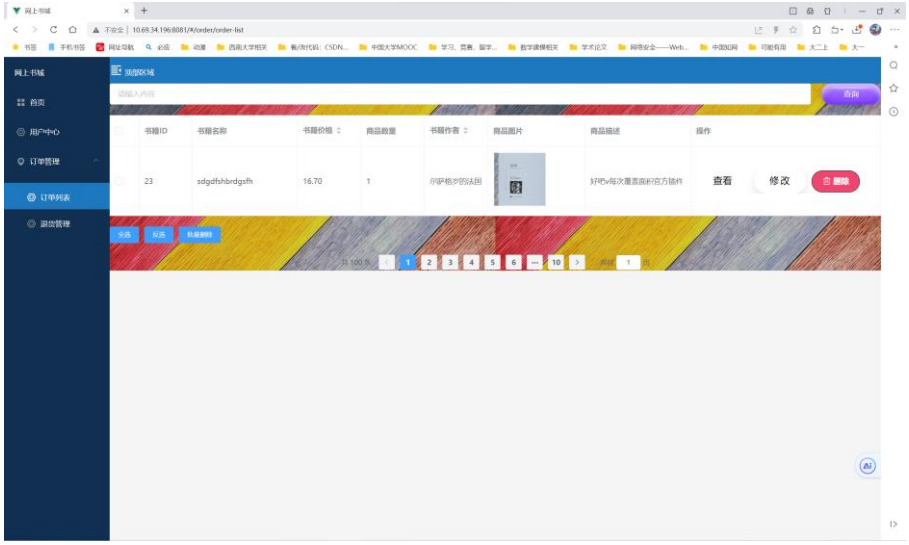
（4）用户也可以修改想要购买的书籍数量：（只展示修改界面与修改过后界面。修改前界面前面已显示）。商品修改成功后，也会有弹窗提示。





（5）当然，你也可以删除不需要的书籍。同样，在删除前会进行询问。





4.1.2 关键技术

1. vue2.x 构建用户界面的渐进式的 JavaScript 框架

采用 vue 框架实现的前端界面有一个优点，修改、删除、添加、查找商品后，点击按钮，界面会实时刷新，用户不会看见明显的切换界面时的刷新效果（所以说，界面切换、更新十分流畅），也不需要用户自己去点击刷新按钮刷新界面（所以说，这种方法的可行性极高，用户友好型界面）

Vue 可以自底向上逐层的应用，简单应用只需要一个轻量小巧的核心库，复杂应用可以引入各式各样的 Vue 插件。而且采用组件化模式，提高了代码复用率、且让代码更好维护。

2.axios 网络请求库

Axios 是一个基于 promis 的网络请求库,作用于 node.js 和浏览器中,它是 isomorphic 的(即同一套代码可以运行在浏览器和 node.js 中)。在服务端它使用原生 node.js http 模块, 而在客户端 (浏览端) 则使用 XMLHttpRequest.

3.wangEditor 富文本编辑器

富文本编辑器，Multi-function Text Editor,简称 MTE,是一种可内嵌于浏览器，所见即所得的文本编辑器。富文本编辑器不同于文本编辑器，程序员可到网上下载免费的富文本编辑器内嵌于自己的网站或程序里，方便用户编辑文章或信息。

它提供类似于 Microsoft Word 的编辑功能，容易被不会编写 HTML 的用户并需要设置各种文本格式的用户所喜爱。它的应用也越来越广泛。最先只有 IE 浏览器支持，其它浏览器相继跟进，在功能的丰富性来说，还是 IE 强些。虽然没有一个统一的标准，但对于最基本的功能，各浏览器提供的 API 基本一致，从而使编写一个跨浏览器的富文本编辑器成为可能。

4.Tomcat Web 应用服务器

Tomcat 服务器是一个免费的开放源代码的 Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好 Apache 服务器，可利用它响应 HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上 Tomcat 是 Apache 服务器的扩展，但运行时它是独立运行的，所以当公司运行 tomcat 时，它实际上作为一个与 Apache 独立的进程单独运行的

4.1.3 技术难点实现结果

1.跨域数据传输问题

由于本系统后端与前端的端口不一致，故在传输数据时难免会存在跨域问题。此问题困扰本人许久，在经过多方搜集资料过后，本人在前端用 js 实现相关请求函数，通过 axois 进行跨域数据传输或者请求，指定调用后端接口；在后端，本人新建 Java 实体类，通过在

获取数据库数据之后，将其转换成 JSON 数据格式，并将 json 数据返回到前端，以此实现跨域数据传输问题。

例如：Goods.vue 加载书籍信息时：

前端代码：

```
/**
 * 商品列表获取
 */
http(page) {
  axios.get('/api/getBooks')
  .then(res => {
    // console.log(res)
    this.books = res.data;
    // console.log(res.data);
    // console.log(this.books);

    //获取'}'出现的次数
    var index = this.books.indexOf('}')
    var num = 0;
    var tempNum = 0;
    while(index !== -1){
      this.book = this.books.slice(tempNum+1,index+1);
      num++;
      //商品行号
      this.row[num-1] = num;
      // console.log(this.row);
      index = this.books.indexOf('}',index + 1);
    }
```

后端代码：

```
@Override
public void service(HttpServletRequest request, HttpServletResponse response) throws IOException {
    //根据获取所有的书籍信息
    List<Book> books = new ArrayList<>();
//      System.out.println("in getBooks");
//      books = BookRepo.getAllBook();
    books = BookRepo.getAllBook2();

    String[] bookJson = new String[books.size()];
    int i = 0;
    for (Book book : books) {
        bookJson[i] = String.format(
                "{\n" +
                        "\t\"id\": \"%s\",\n" +
                        "\t\"name\": \"%s\",\n" +
                        "\t\"imageurl\": \"%s\",\n" +
                        "\t\"author\": \"%s\",\n" +
                        "\t\"price\": \"%s\",\n" +
                        "\t\"content\": \"%s\"\n" +
                        "}",
                book.getId(), book.getName(), book.getImageUrl(),
                book.getAuthor(), book.getPrice(), book.getContent());
        i++;
    }

    //返回结果
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    // 获取PrintWriter对象
    PrintWriter out = response.getWriter();
    out.print(Arrays.toString(bookJson));
    // 释放PrintWriter对象
    out.flush();
    out.close();

}
```

其他情况以此类推，根据需求实现即可。

2.本地图片上传与显示问题

本人本来打算能否直接加载硬盘里面的本地图片，但是一番操作下来发现，由于浏览器存在安全保护机制，是不允许直接加载本地图片的，至此这条路径夭折。

后来本人又想把图片放入前端 assets 目录的 image 文件夹下面，但是当图片地址从后端上传至前端后，我们发现，数据是成功从传输了的（控制台有打印），但是将图片路径更换为前端相应路径后，浏览器确实显示图片了，只不过显示的是图片的存储地址，而非图片本身，无论本人怎么修改代码都无法显示图片。而只是显示图片地址，这或许有更深层的原因，但也宣告这条路径不好走。

后来本人了解到，一般网页图片都存放在服务器端，服务器端的图片可以正常加载。我深受启发，我们的服务器端不就是 Java-Tomcat 后端吗，于是本人转换思路将图片的存储路径一律修改为如下格式：http://localhost:8080/demo1/image/xx.jpg 当我再次打开前端刷新界面时，看到了正确显示的图片。看来图片存放在服务器端确实能够正常加载，由此解决图片上传与显示问题。

3.购物车对书籍数目与价格更新问题

有关购物车的上述问题，本人有许多种想法去解决它，但在经过多方考虑之后，决定先做一个相对简单的实现方式，至于后续优化版本，可以寒假在来完善。

由于通过+/-按钮进行书籍数目与价格更新所需修改的东西比较多，故本人打算采取一种折中的方案，通过直接修改数目值来进行更新操作。

这种操作只需要用户在修改书籍数目后，将新的书籍数目传入后端，后端用新的书籍数目进行数据库更新，在更新之前，用该数目*单价，即可得书籍总价。由此解决购物车对书籍数目与价格更新问题。

4.2 系统实现的不足

系统的不足主要有以下几点：

（1）用户注销账户时，最好能够进行实名认证，入获取绑定手机/邮箱验证码这种形式。

（2）购物车对书籍数目与价格更新是一个比较折中的方案，其实还可以升级优化，借助+/-按钮进行实现，而不是修改数目。

（3）还没有加入退货（售后）管理这一部分，虽然已经规划了相关界面。

（4）数据库有关用户的信息都是明文存储，保密性不高。

第五章 总结与展望

学习与实践总是一个连续而又相互影响的过程，在学习了 Java 与 JavaWeb 相关知识后，完成一个项目，绝对是最有效的检验所学知识的一种途径。在网上书城项目中，我们运用相关知识，例如：等，由此构建起属于我们自己的后端接口，对初学者来说，这是一项大工程，是对自己能力的检验。

为了实现前后端分离，本人决定采用 Vue2.x 框架重写前端，由于是首次使用 vue 框架来实现前端代码，在这期间难免会遇到许多问题，遇到的最大的问题就是跨域数据传递问题。这花费了我将近一周的时间去解决，但是在解决了这个问题过后，所收获的东西是十分巨大的。跨过这道坎，本人学会了解决前后端跨域问题的基本方法，学习了一部分 Vue 与 js 的语法知识，也学会了基本的前后端分离技术（Vue2+Java+Tomcat）

同时，在实现购物车功能时，本人其实脑子里面浮现出许多想法。但是碍于时间与精力，本人最终只采用了一种折中方案，这算是一个小小的遗憾。本人打算在寒假继续完善该项目，届时将进行购物车的全面优化升级。

可以说，这次的项目是一个知识的集合体，也是在遇到困难时，现学现做的学习过程，通过这次项目，个人感觉到了一种成就感与自豪感，这是个人的第一个网上项目，有始有终代表着一个好的开始。