

一、 用户端设计 2

- 1、网络书城主界面 (index.jsp) 2
- 2、注册界面 (register.jsp) 3
- 3、登录界面 (login.jsp) 4
- 4、分类显示界面 (fenleixianshi.jsp) 5
- 5、模糊查询显示界面 (chaxunxianshi.jsp) 6
- 6、书籍详细信息显示界面 (xinxi.jsp) 6
- 7、购物车界面 (gouwuche.jsp) 7
- 8、订单中心界面 (dingdanzhongxin.jsp) 8
- 9、找回密码界面 (lost.jsp) 9
- 10、修改信息 (update.jsp) 9

二、 设计要求及功能需求分析 10

- 1、主要界面架构 10
- 2、主要功能 10
- 3、其他要求 12

三、 系统概要和层次图 12

- 1.整合配置文件 14
- 2.DAO层 18
 - 2.1.BaseDao 接口类18
 - 2.2BaseDaOImpl实现类 18
- 3.逻辑层 (Service-服务层) 19
 - 3.1用户管理子系统服务层接口UserService.java 20
 - 3.2图书管理子系统服务层接口BookService.java 20
 - 3.3订单管理子系统服务层接口OrderService.java 20
 - 3.4购物车管理子系统服务层接口ShoppingcartService.java 21
 - 3.5用户管理服务类UserServiceImpl.java 21
 - 3.6图书管理服务类BookServiceImpl.java 22
 - 3.7订单管理服务类OrderServiceImpl.java 23
 - 3.8购物车管理服务类ShoppingcartServiceImple.java 23
- 4.控制层 25
- 5.视图层 29

四、 系统目录及页面跳转流程图 30

- 1、系统目录 30
- 2、页面跳转流程图 30

五、 管理员端设计 31

- 1.功能需求和分析 31
- 2.前端的界面 32
- 3.部分源代码 36

一、用户端设计



1、网络书城主界面 (index.jsp)

①：显示登录信息，若登录则会显示登录账号，否则提供“注册”、“登录”、“我的订单”、“修改资料”、“购物车”。若没有登录则会显示弹出如下提示框：

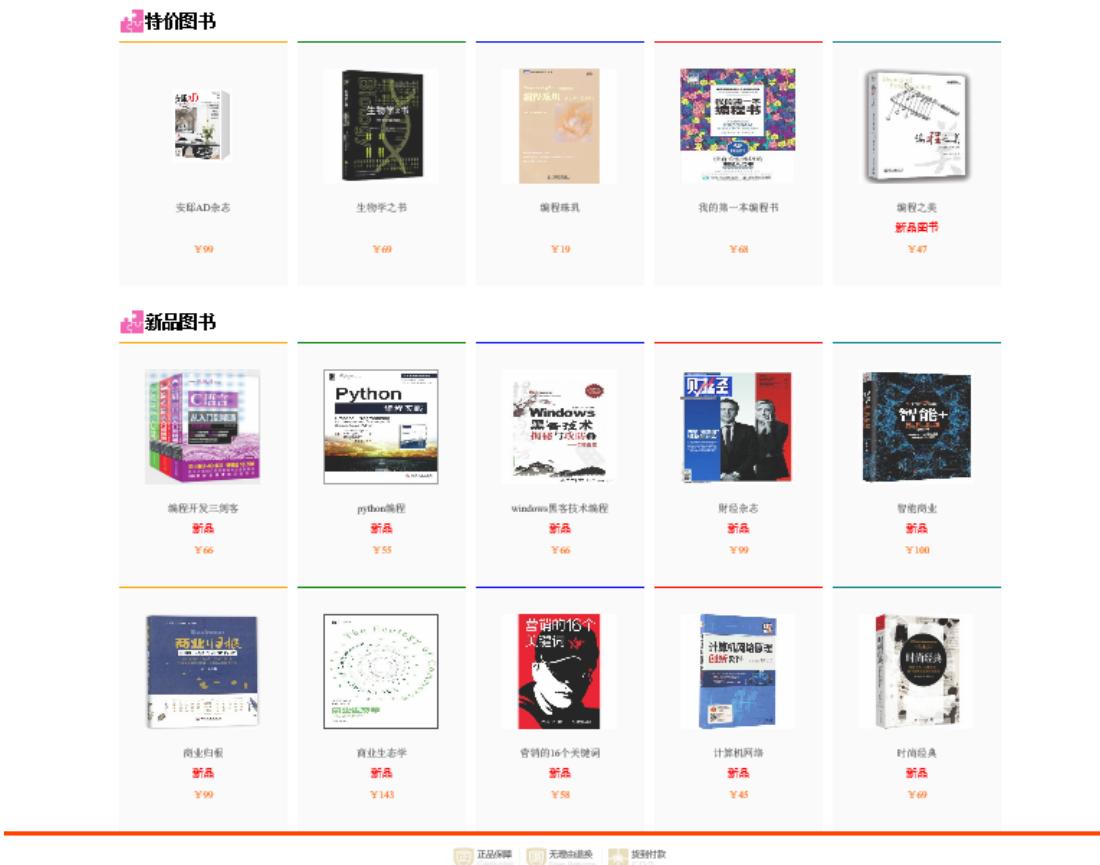


②：模糊查询的搜索框，可供用户在不输入完整书籍信息的情况下查找有关的书籍，例如：查询“VC”，则会显示与“VC”有关的书籍。

③：鼠标移到该选框，会出现如下图所示的下拉二级菜单，可供分类查询书籍。



- ④：此处可自动或手动滚动播放图片，提供书城的部分信息或者有关的动态。
- ⑤：主页向下拉，则会显示部分特价图书，也可显示部分热销书籍、新品书籍等，可供用户有更多的选择（如下图）。



2、注册界面 (register.jsp)

会员注册

返回首页

①	用户名： <input type="text" value="test"/> 密码： <input type="password"/> 确认密码： <input type="password"/> 地址： <input type="text"/> 邮箱： <input type="text"/>	<small>(*必填)只允许字母数字汉字1到8位</small> <small>(*必填)两次密码要输入一致哦</small> <small>只允许字母数字汉字1到15位</small> <small>请输入邮箱正确格式</small>
---	--	--

②	手机号： <input type="text"/> 验证码： <input type="text"/> <small>免费获取验证码</small>	<small>只允许数字，0到11位</small>
---	---	----------------------------

立即注册

①
：
输入
用
户
的

基本信息

②：输入正确的手机号，并点击“免费获取验证码”，会向手机发送短信验证码，输入正确的验证码才可完成注册。如下图：



- ③：此处显示需要输入的信息的要求。
- ④：如果输入的信息符合要求，则显示“✓”，否则显示“格式错误”，无法完成注册。
- ⑤：若用户名已经被注册，则显示“该用户已存在，请重新注册”。

(注：该系统测试提供的测试账号：test 密码：123)

3、登录界面 (login.jsp)



- ①：忘记密码以及立即注册的链接，方便用户的操作。
- ②：输入正确的账号和密码后，输入正确的验证码才可实现登录。若验证码图片看不清，点击该验证码的图片可以更换图片。

4、分类显示界面 (fenleixianshi.jsp)

The screenshot shows a web page for a network bookstore. At the top, there is a navigation bar with links for '你好, test' [Logout], '我的订单' [My Orders], '修改资料' [Edit Profile], a shopping cart icon, and a search bar. Below the navigation bar is the bookstore logo '网络书城 Bookstore'. A dropdown menu labeled '商品分类' [Product Category] is open, showing '商业/财经/营销' [Business/Finance/Marketing] as the selected category. The main content area displays five book covers under this category. Each book has a title, a brief description, and a price. A blue rectangular box highlights the category name '商业/财经/营销' and the sorting controls below it. Three numbered callouts point to specific elements: ① points to the category name; ② points to the sorting controls; ③ points to the list of books.

书名	类别	售价
营销的16个关键词	营销	68
财经杂志	财经	111
智能+	智能	111
商业归根	商业	155
商业生态学	生态	156

- ①：显示所查询的分类对应的名称，查询其他的分类也可显示其他分类的名称。
- ②：若商品数量较多，则可选择“按价格升序排序”，“按价格降序排序”，按价格区间进行排序，选择后则可以按照对应的要求排列书籍。例如：选择“按价格升序”，效果如下图：

This screenshot shows the same bookstore interface as above, but the books are now sorted by price in ascending order. The books are listed from lowest price to highest. The sorting controls at the top of the book list are highlighted with a red box. Three numbered callouts point to specific elements: ① points to the category name '商业/财经/营销'; ② points to the sorting controls; ③ points to the list of books.

书名	类别	售价
营销的16个关键词	营销	68
财经杂志	财经	111
智能+	智能	111
商业归根	商业	155
商业生态学	生态	156

5、模糊查询显示界面 (chaxunxianshi.jsp)



- ①：显示输入查询的内容。
- ②：与分类显示相同，可对查询到的结果进行排序。
- ③：若没有查询到结果，则显示如下界面（截取部分）

非常抱歉，没有查询到任何结果。

[返回主页](#)

6、书籍详细信息显示界面 (xinxi.jsp)



- ①: 显示该商品的图片
- ②: 显示该商品的名称
- ③: 显示该商品的类别
- ④: 显示想要购买的数量。可进行调整，实现数量的加减操作，并可按照对应的数量提交到购物车或订单。
- ⑤: 显示该商品的介绍
- ⑥: 显示该商品的价格，若商品是特价商品，则显示“价格”和“特价”，否则显示“价格”。
- ⑦: 可提供“立即购买”（直接添加到订单）和“加入购物车”（添加到购物车）

(注：以上信息显示全部非静态，均从数据库中获取。购买的数量改变后可提交，购物车中会直接显示对应的数量，并可在购物车中更改数量。)



- ⑧: 示例调整商品数量，即可改变购买数量。
- ⑨: 示例若商品为特价商品，则显示价格的同时显示特价。

7、购物车界面 (gouwuche.jsp)

The screenshot shows a shopping cart interface with the following details:

商品	单价	数量	操作
java编程思想	50.0	1	X
终极斗罗2	78.0	3	X
肖秀荣考研政治	69.0	5	X
java web应用开发技术	199.0	7	X

At the bottom, there is a summary bar with the text: 合计(不含运费): ¥0.0元 and a 提交订单 button.

网络书城设计报告

- ①：可勾选商品进行购买，点击全选即可选择全选或取消全选。
- ②：显示商品的数量，并可进行数量的更改。（数量的初始值为提交到购物车时选择的数量）。若添加到购物车的商品有与购物车重复的商品，会将在购物车中的该商品的数量叠加，而不会产生新的购物车信息。
- ③：可进行商品的删除操作
- ④：点击“继续购物”按钮可返回主页继续进行购物
- ⑤：显示价格合计（根据勾选的商品及其价格和数量实时计算总价，方便用户查看），并可执行“提交订单”，提交到个人的订单中。

(注：相关功能演示如下图)

The screenshot shows a shopping cart interface. At the top, there's a header with a logo, the title '我的购物车', a message about purchase success, and user information. Below is a table for managing items:

<input type="checkbox"/> 全选	商品	单价	数量	操作
<input checked="" type="checkbox"/>	java编程思想	50.0	<input type="button" value="-"/> 1 <input type="button" value="+"/>	<input type="button" value="x"/>
<input type="checkbox"/>	终极斗罗2	78.0	<input type="button" value="-"/> 3 <input type="button" value="+"/>	<input type="button" value="x"/>
<input checked="" type="checkbox"/>	肖秀荣考研政治	69.0	<input type="button" value="-"/> 5 <input type="button" value="+"/>	<input type="button" value="x"/>
<input type="checkbox"/>	java web应用开发技术	199.0	<input type="button" value="-"/> 7 <input type="button" value="+"/>	<input type="button" value="x"/>

At the bottom, there are buttons for 'Continue Shopping' and 'Place Order'. A red box highlights the total price '合计(不含运费) : ¥540元' and the 'Place Order' button. A red circle labeled ⑦ points to the price column, and another labeled ⑧ points to the total price.

- ⑦：显示商品的价格。若商品为特价商品，则直接显示商品的特价，否则显示原价。（如《考研政治》原价为98，折扣价为69，则按照69计算）
- ⑧：实时计算价格并显示 ($50*1 + 69*5 = 540$) 。

8、订单中心界面 (dingdanzhongxin.jsp)

The screenshot shows an order center interface. At the top, there's a header with user info, a search bar, and a navigation bar. Below is a table for pending orders:

交易订单					
我的宝贝订单信息如下：					
商品订单号	状态	总价	日期	收货地址	
335a16c0b2ed45ec8839c2a6da5c2583	未发货	555	2019-01-02 16:36:37	山东省泰安市山东农业大学	
3ec2ff7af6d384a02a99f2b9cd4412a10	未发货	490	2019-01-02 22:59:03	山东省泰安市山东农业大学	
d13faa3872bc4402b5b8d48551548190	未发货	594	2019-01-02 16:36:40	山东省泰安市山东农业大学	
e341c99a482a4598832ebaf6ba9fc25e2	未发货	50	2019-01-02 22:58:53	山东省泰安市山东农业大学	

- ①：显示生成的商品订单号
- ②：显示订单的发货状态，其状态由管理员端进行更改，此处不做赘述。

9、找回密码界面（lost.jsp）



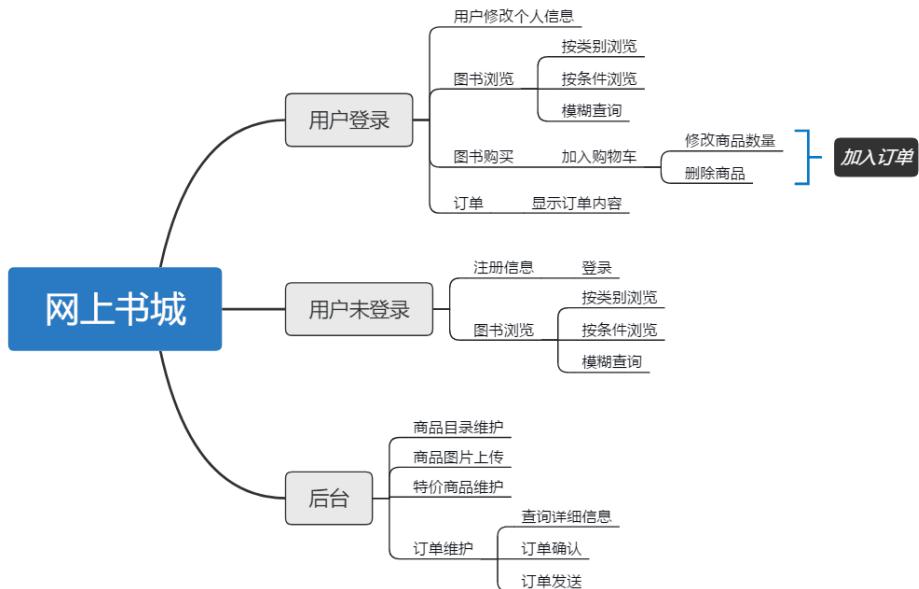
输入用户名，以及注册时的手机号，获取短信验证码，短信验证码正确后直接跳转到修改资料页面进行密码以及其他资料的修改。

10、修改信息（update.jsp）

The screenshot shows a '修改资料' (Modify Information) form. The '用户名' (Username) field, containing 'test', is highlighted with a red circle labeled ①. The entire form area, including fields for '密 码' (Password), '确认密码' (Confirm Password), '手 机 号' (Mobile Number), '邮 箱' (Email), and '地 址' (Address), is highlighted with a red circle labeled ②. An orange '立即修改' (Immediately Modify) button is at the bottom.

- ①：用户名字段为仅可读，即用户无法修改用户名。
- ②：用户可修改自己的全部或部分信息，但同样需要符合规定要求。具体如同注册界面。

二、设计要求及功能需求分析



本系统的主要架构与功能实现如下所述

1、主要界面架构

- (1) 主界面：完成信息的浏览，功能的链接，注册登录的入口等。
- (2) 注册登录界面：完成信息的注册，用户的登录以使用更多的功能。
- (3) 对于图书的显示，排列方式的界面实现，可供用户更加直观的浏览图书。
- (4) 图书信息的显示，显示图书的简介，名称，价格，分类等基本信息，并可供用户加入到购物车或者订单。
- (5) 购物车界面用列表的方式显示用户购物车中的信息，包括名称，图片，数量，价格等信息，并可通过js实现实时价格的计算，可供用户更直观的使用。
- (6) 订单界面列表显示用户的订单信息，包括订单号，数量，价格，收货地址，发货状态等基本信息。用户的订单信息可由管理员端进行修改，例如可修改商品的发货状态。

2、主要功能

2.1、用户注册

- (1) 用户填写信息进行注册，主要包括用户名，密码，收货地址，邮箱等基本信息，其中各种信息的填写需要符合系统设定的规范，不符合规范的系统会给出提示并且无法顺利完成注册。

(2) 在用户注册功能中添加了“手机短信验证码”的功能，即用户填写正确的手机号，会向该手机发送短信验证码（6位），用户用手机接收到短信验证码后，需填写正确的短信验证码才可顺利完成注册。

2.2、用户登录

(1) 用户填写自己的用户名和密码进行登录，登录后方可完成购买图书，添加到购物车等功能。

(2) 用户在登录时需填写图片验证码中的内容，若填写正确则可以完成登录，否则会显示“验证码不正确”。

2.3、分类查询

(1) 在该系统首页有“商品分类”的选项，鼠标放在该选项上会出现下拉的二级菜单，显示图书的分类，点击即可跳转到该类别的图书显示的界面。

2.4、模糊查询

(1) 在每一个界面的右上角都有搜索框，用户可通过在搜索框中输入所要查询的书籍的全部或部分信息即可。

(2) 用户点击“搜索”按钮，即可跳转到图书的显示界面，显示与用户搜索信息有关的图书。

2.5、图书显示

(1) 图书按照用户选择的方式显示出来，可供用户进一步查看。

(2) 图书显示界面提供了排序方式，用户可以选择“按价格升序排序”，“按价格降序排序”，“默认显示”，以及可以输入想要查询的价格区间进行查询，用户使用如上几项功能后，系统会按照需求进一步对图书进行排列显示，供用户更方便的查看。

2.6、图书详细信息

(1) 用户在查找到自己想要的图书后，可点击图书的图片或名称进入该图书的详细信息页面。

(2) 该图书的详细信息，包括该图书的图片，名称，分类，价格，简介，以及用户想要购买的数量，并且为用户提供直接购买（添加到订单）或加入购物车的功能。

(3) 用户想要调整购买的图书的数量，可通过“+”“-”按钮进行操作，该数量会直接加入到购物车或者订单中。

2.7、购物车显示

(1) 系统根据用户提交到购物车中的信息，列表显示。

(2) 用户可在购物车中更改所购买的商品的数量。

(3) 用户可选择部分或全选购物车中的商品，提供全选框实现所有商品的选择，也可勾选部分商品进行价格的计算和购买。

(4) 用户可在购物车中查看所选择商品的价格，即购物车会根据用户勾选的商品的价格、数量计算总价并进行显示。

- (5) 购物车提供继续购物，可供用户返回主页继续购买商品。
- (6) 购物车提供提交订单的功能，即用户可将勾选的商品提交到订单，同时系统自动删除购物车中的对应信息。
- (7) 用户可对购物车中的信息进行删除操作。

2.8、订单显示

- (1) 用户可在订单中心查看自己的订单，包括发货状态等。
- (2) 订单信息可由管理员端进行更改。

2.9、找回密码

- (1) 用户忘记密码，可输入用户名并填写注册时填写的手机号，通过手机短信验证码进行验证。
- (2) 若验证通过，则跳过登陆页面，进入修改资料页面进行密码的修改。

3.0、修改信息

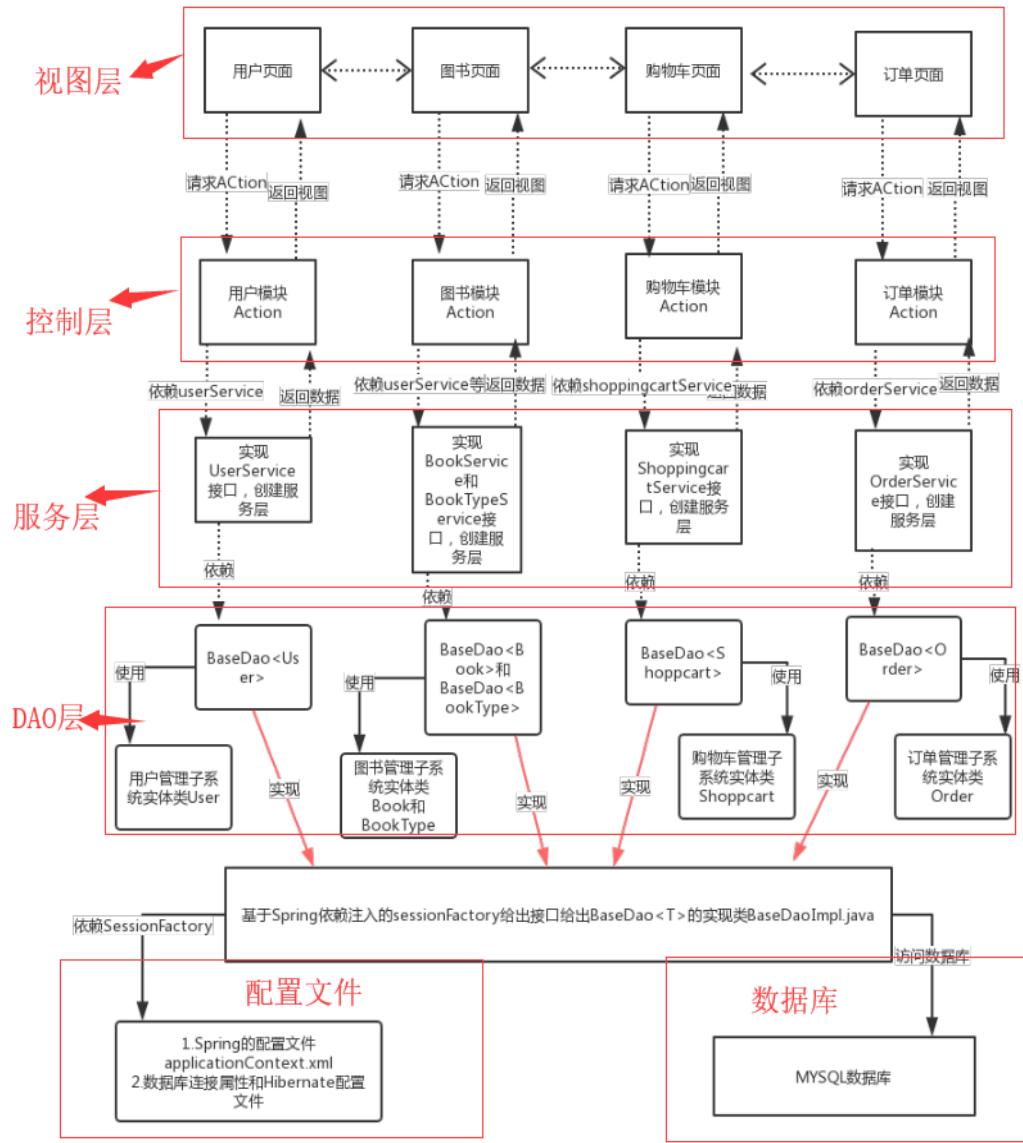
- (1) 用户可通过“修改信息”页面进行对个人信息的修改。
- (2) 用户名一栏设置为“只读”，即不可进行修改。

3、其他要求

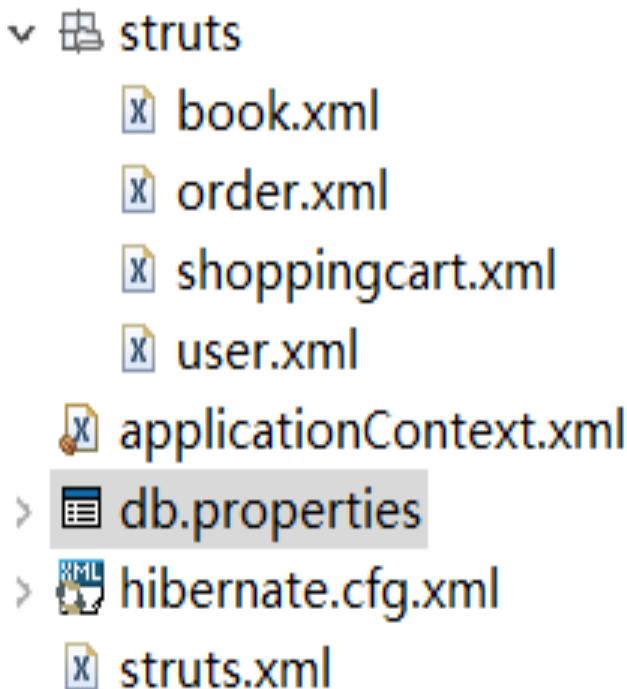
- (1) 本系统的实现采用Struts2+Hibernate框架。
- (2) 数据库使用Mysql，数据表建立后可由全体成员进行访问和进行增删改查操作，由管理员进行统一管理。
- (3) 使用部分js以实现有关的功能。

三、系统概要和层次图

以SSH框架（c3p0连接池）构建一个网上书城系统，包括用户子系统（登录、注册、修改密码、找回密码 实现短信验证码和图片验证码的验证等），图书子系统（显示图书，分类查询图书，模糊查找图书，显示图片信息价格，按价格升序降序或区间查询或显示图书等），购物车子系统（选择图书和数量加入购物车，显示购物车中的图书信息、删除购物车、显示已选物品总价，支持全选和全不选等），订单子系统（将购物车页面选择的图书提交到订单，实现订单的显示等）



1.整合配置文件



(1) applicationContext.xml Spring最基本的配置信息（数据源，SessionFactory和事务处理等）

关键代码：

```

<!-- 自动加载构建bean -->
<context:component-scan base-package="com.java1234" />
    <!-- 配置对属性文件访问的支持 -->
    <context:property-placeholder
location="classpath:db.properties"/>
        <!-- 定义数据源 -->
        <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
            <property name="driverClass" value="${driverClass}"></
property>
            <property name="jdbcUrl" value="${url}"></property>
            <property name="user" value="${user}"></property>
            <property name="password" value="${password}"></
property>
            <property name="initialPoolSize" value="$
{initPoolSize}"></property>
            <property name="maxPoolSize" value="${maxPoolSize}"></
property>
        </bean>

        <!-- session工厂 -->
<bean id="sessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryB
ean">
            <property name="dataSource" ref="dataSource"></

```

```

property>
    <property name="configLocation"
value="classpath:hibernate.cfg.xml"/>
    <!-- 自动扫描hibernate类文件 -->
    <property name="packagesToScan">
        <list>
            <value>com.java1234.entity</value>
        </list>
    </property>
</bean>

<!-- 配置事务管理器 -->
<bean id="transactionManager"
class="org.springframework.orm.hibernate4.HibernateTransaction
Manager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>

```

(2) db.properties 数据库属性文件，存放连接数据库所需要的参数

关键代码：

```

user=root
password=myg0820
driverClass=com.mysql.jdbc.Driver
url=jdbc:mysql://47.93.102.163:3306/javaee?
useUnicode=true&characterEncoding=UTF-8
initPoolSize=5
maxPoolSize=10

```

(3) hibernate.cfg.xml 配置Hibernate基本属性

关键代码：

```

<session-factory>
    <!--方言-->
    <property
name="dialect">org.hibernate.dialect.SQLServerDialect</
property>
    <!-- 显示sql语句 -->
    <property name="show_sql">true</property>
    <!-- 自动更新 -->
    <property name="hbm2ddl.auto">update</property>

    <property
name="hibernate.connection.autocommit">true</property>
</session-factory>

```

(4) struts.xml 配置Struts2的基本属性和公共Action等信息

关键代码:

```
<struts>
    <constant name="struts.action.extension" value="action" />
        <include file="struts/*.xml"></include>
</struts>
```

(5) 文件夹Struts,四个业务逻辑对应模块的Action的配置文件。

a) User.xml关键代码:

```
<action name="Login" method="Login"
class="com.java1234.action.UserAction">
    <result name="success" type="redirect">index.jsp</result><!--
    type="redirect" -->
    <result name="error">login.jsp</result>
</action>

    <action name="Register" method="Register"
class="com.java1234.action.UserAction">
        <result name="success">index.jsp</result><!--
        type="redirect" -->
        <result name="error">register.jsp</result>
    </action>

    <action name="Logout" method="Logout"
class="com.java1234.action.UserAction">
        <result name="success">/index.jsp</result>
    </action>
```

b) Book.xml关键代码:

```
<action name="findbytype" method="FindByType"
class="com.java1234.action.BookAction">
<result name="success">/fenleixianshi.jsp</result>
<result name="error">/fenleixianshi.jsp</result>
<result name="input">/fenleixianshi.jsp</result>
</action>
<action name="findtype" method="FindType"
class="com.java1234.action.BookTypeAction">

<result name="success">/success.jsp</result>
<result name="error">/register.jsp</result>
<result name="input">/register.jsp</result>
</action>
```

c) Shoppingcart.xml关键代码:

```
<action name="Shoppingcart" method="Shoppingcart"
class="com.java1234.action.ShoppingcartAction">
    <result name="success">gouwuche.jsp</result>
    <result name="error">login.jsp</result>
</action>
```

```
<action name="InserttoShoppingcart"
method="InserttoShoppingcart"
class="com.java1234.action.ShoppingCartAction">
    <result name="success">xinxi.jsp</result>
    <result name="error">xinxi.jsp</result>
</action>
```

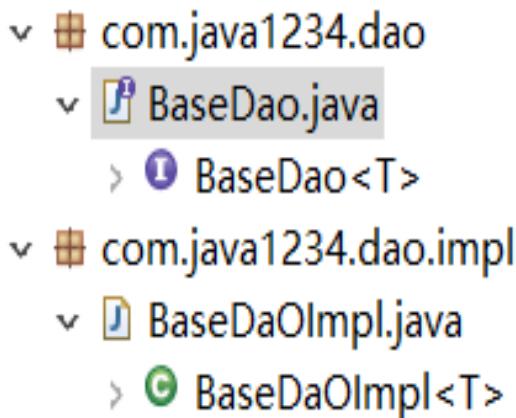
d) Order.xml关键代码:

```
<action name="Inserttoorder" method="InserttoOrder"
class="com.java1234.action.OrderAction">
    <result name="success">gouwuche.jsp</result>
    <result name="error">gouwuche.jsp</result>
    <result name="input">gouwuche.jsp</result>
</action>

<action name="findallorder" method="FindAllOrder"
class="com.java1234.action.OrderAction">
<result name="success">dingdanzhongxin.jsp</result>
<result name="error">dingdanzhongxin.jsp</result>
<result name="input">dingdanzhongxin.jsp</result>
</action>
```

2.DAO层

(采用Hibernate框架实现对数据库的访问)



2.1.BaseDao 接口类

关键代码：

```

public interface BaseDao<T> {
    public Serializable save(T o);
    public void delete(T o);
    public void update(T o);
    public void saveOrUpdate(T o);
    public List<T> find(String hql);
  
```

2.2BaseDaOImpl实现类

(使用Spring依赖注入的sessionFactory对象)

关键代码：

```

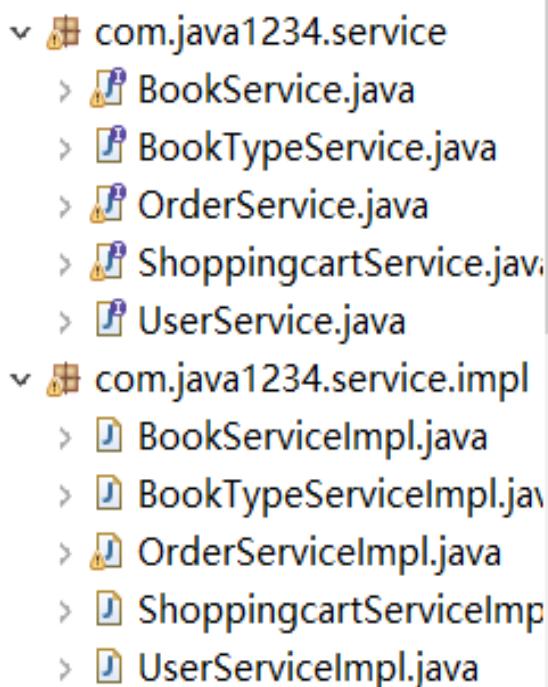
@Repository("baseDao")
@SuppressWarnings("all")
public class BaseDaOImpl<T> implements BaseDao<T>
{
    private SessionFactory sessionFactory;
    public SessionFactory getSessionFactory()
    {return sessionFactory;}
    @Autowired
    public void setSessionFactory(SessionFactory
  
```

```
sessionFactory)
    {this.sessionFactory = sessionFactory;}
    private Session getCurrentSession() {return
sessionFactory.getCurrentSession();}
    public Serializable save(T o) {return
this.getCurrentSession().save(o);}


```

3.逻辑层（Service-服务层）

位于数据访问层与控制层之间，承上启下，面向接口。



图中组件

3.1 用户管理子系统服务层接口UserService.java

```
public interface UserService {
    public void saveUser(User user);
    public void updateUser(User user);
    public User findUserById(int id);
    public User findUserByName(User user);
    public User findUserByPhone(User user);
    public void deleteUser(User user);
    public List<User> findAllList();
    public User findUserBy2(User user);
}
```

3.2 图书管理子系统服务层接口BookService.java

```
public interface BookService {
    public List<Book> findBookByType(Book book,int cs,int price[]);
    public Book findBookById(String bookId);
    public List<Book> findbylike(String information,int cs,int price[]);
}
```

3.3 订单管理子系统服务层接口OrderService.java

```
public interface OrderService {
    public void insertintoorder(OrderEntity order);
    public List findallorder(User user);
}
```

3.4 购物车管理子系统服务层接口 ShoppingCartService.java

```

public interface ShoppingCartService {
    public boolean insert(Shoppingcart shoppingcart); //插入
    public List<Shoppingcart> Findlist(String userId); //获取
    购物车表
    public Shoppingcart findshoppingcart(String
    user_id, String book_id); //获取shoppingcart对象
    public List<Book> findbyid(List<Shoppingcart> list); //通
    过购物车表获取图书表
    public boolean detelebook(Shoppingcart shoppingcart); //
    删除表
}

```

3.5 用户管理服务类 UserServiceImpl.java

依赖baseDao<User>,并实现BaseDaOImpl的接口

```

@Service("userService")
public class UserServiceImpl implements UserService{
    @Resource
    private BaseDao<User> baseDao;

    @Override
    public void saveUser(User user) {
        baseDao.save(user);
    }
    @Override
    public void updateUser(User user) {
        // TODO Auto-generated method stub
        baseDao.update(user);
    }

    @Override
    public User findUserById(int id) {
        return baseDao.get(User.class, id);
    }

    @Override
    public void deleteUser(User user) {
        baseDao.delete(user);
    }

    @Override
    public List<User> findAllList() {
        return baseDao.find("from User");
    }
}

```

```

@Override
public User findUserBy2(User user) {
    return baseDao.get("from User u where u.user_name=? and u.password=?",
new Object[]{user.getUser_name(),user.getPassword()});
}
public User findUserByPhone(User user)
{
    return baseDao.get("from User u where u.user_name=? and
u.telephone=?",
new Object[]{user.getUser_name(),user.getTelephone()});
}
public User findUserByName(User user)
{
    return baseDao.get("from User u where u.user_name=?",
new Object[]
{user.getUser_name()});
}
}

```

3.6图书管理服务类BookServiceImpl.java

依赖baseDao<Book>,并实现BaseDaOImpl的接口

```

@Service("bookService")

public class BookServiceImpl implements BookService {
    @Resource
    private BaseDao<Book> baseDao;

    @Override

    public List<Book> findBookByType(Book book,int cs,int price[]){
        List<Book> list = null;
        String hql =null;
        if(cs==0)
            hql = "from Book b where b.type = "+book.getType()+"";
        else if(cs==1) //升序
            hql = "from Book b where b.type = "+book.getType()+" order by b.price
asc";
        else if(cs==2) //降序
            hql = "from Book b where b.type = "+book.getType()+" order by b.price
desc";
        else //区间
            hql = "from Book b where b.type = "+book.getType()+" and b.price
between "+price[0]+" and "+price[1]+""";
        String param[] = {};
        list = baseDao.find(hql,param);

        System.out.println("结束");
        return list;
    }
    public List<Book> findbylike(String information,int cs,int price[]){
        List <Book> list = null;
//        String hql = "from Book b ";
        String hql = "from Book b ";
        String Field[] = {"b.bookName","b.type","b.description"};
        list = baseDao.findByFields(hql,Field,information,cs,price);
    }
}

```

```

        return list;
    }
    @Override
    public Book findBookById(String bookId) {
        String hql = "from Book b where b.bookId=?";
        String param[] = {bookId};
        Book book=baseDao.get(hql, param);
        // TODO Auto-generated method stub
        return book;
    }
}

```

3.7 订单管理服务类OrderServiceImpl.java

依赖baseDao<Order>,并实现BaseDaOImpl的接口

```

@Service("OrderService")
public class OrderServiceImpl implements OrderService{
    @Resource
    private BaseDao<Orders1Entity> baseDao;

    @Override
    public void insertintoorder(Orders1Entity order) {
        baseDao.save(order);
    }
    @Override
    public List findallorder(User user) {
        String hql = "from Orders1Entity o where o.userId = ?";
        String param[] = {user.getUser_id()};
        List list = baseDao.find(hql,param);
        return list;
    }
}

```

3.8 购物车管理服务类ShoppingcartServiceImpl.java

依赖baseDao<Shoppingcart>,并实现BaseDaOImpl的接口

```

@Service("ShoppingcartService")
public class ShoppingcartServiceImpl implements ShoppingcartService{
    @Resource
    private BaseDao<Shoppingcart> baseDao;
    @Resource
    private BaseDao<Book> baseDao1;
    public List<Shoppingcart> Findlist(String userId){ //通过用户ID获取购物车表
        List<Shoppingcart> list = null;
        String hql = "from Shoppingcart s where s.user_id = ?";
        String param[] = {userId};

```

```

list = baseDao.find(hql,param);
    return list;
}
//通过userid和bookid获得一个shoppingcart对象
public ShoppingCart findShoppingCart(String user_id,String book_id)
{
    String hql = "from ShoppingCart s where s.user_id = ? and book_id=?";
    String param[] = {user_id,book_id};
    return baseDao.get(hql, param);
}

}
//删除一个shoppingcat记录
public boolean deleteBook(ShoppingCart shoppingCart)
{
    try{
        baseDao.delete(shoppingCart);
        return true;
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    return false;
}

//插入到购物车
public boolean insert(ShoppingCart shoppingCart){
    try{
        ShoppingCart sp1=null;
        String hql="from ShoppingCart s where s.user_id =? and s.book_id =?";
        if (baseDao.get(hql, new Object[]
{shoppingCart.getUser_id(),shoppingCart.getBook_id()})!=null)
        {
            // System.out.println("2");
            sp1 = baseDao.get(hql, new Object[]
{shoppingCart.getUser_id(),shoppingCart.getBook_id()});
            sp1.setCount(shoppingCart.getCount()+sp1.getCount());
            baseDao.save(sp1);
            // System.out.println("1");
        }
        else
            baseDao.save(shoppingCart);
        return true;
    }catch(Exception e){
        e.printStackTrace();
        return false;
    }
}

@Override
public List<Book> findByID(List<ShoppingCart> list){ //通过购物车列表获取图书
表
    ArrayList<Book> list1=new ArrayList<Book>();
    String hql = "from Book b where b.bookId =?";
    try { for (int i = 0; i < list.size(); i++){
        String param[] = {list.get(i).getBook_id()};

```

```

        list1.add(i,baseDao1.get(hql,param));
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
return list1;
}
}

```

4.控制层

由struts2的Action实现该层的功能

组件

- ✓  com.java1234.action
 - >  BookAction.java
 - >  BookTypeAction.java
 - >  CreateImageAction.java
 - >  OrderAction.java
 - >  ShoppingCartAction.java
 - >  UserAction.java

用户模块Action类（登录包括发送短信验证码，注册，找回密码，修改资料等）

UserAction.java部分代码

```

public String Login() throws Exception{
    System.out.println("用户尝试登录");
    HttpSession session=request.getSession();
    String checkCode2 = (String)
session.getAttribute("checkCode");
    System.out.println("本次登录验证码: "+checkCode2+" 用户填入
的验证码"+checkCode);
    if (!checkCode.equals(checkCode2)) {
        System.out.println("登录失败: 验证码错误");
        msg="验证码错误, 请重新输入";
        return ERROR;
    }
}

```

```

    else
    {
        User currentUser=userService.findUserBy2(user);
        if(currentUser!=null){
            System.out.println("用户登录成
功, "+currentUser.getUser_name());
            user.setUser_id(currentUser.getUser_id());
            session.setAttribute("currentUser", currentUser);

            session.setAttribute("userId", currentUser.getUser_id());
            return SUCCESS;
        }else{
            if((User)userService.findUserByName(user)!=null)
            {System.out.println("密码错误 ");}
            msg="密码错误";
        }
        else
        {System.out.println("用户名不存在 ");}
        msg="用户名不存在";}
        return ERROR;
    }
}
}
}

```

图书模块Action类（查询显示等）

BookAction.java部分代码

```

//分类显示
public String FindByType(){
    int cs=0; //默认不做任何筛选排序
    int [] price = new int[2];

    if(request.getParameter("asc")!=null) cs=1; //升序
    else if(request.getParameter("desc")!=null) cs=2; //降序
    else if(request.getParameter("minPrice")!=null){
        cs=3;
    }
    price[0]=Integer.parseInt((String)request.getParameter("minPri
ce"));

    price[1]=Integer.parseInt(request.getParameter("maxPrice"));
}

String forward = "";
List<Book> list = null;
list = bookService.findBookByType(book,cs,price);
if (list.size() != 0){

```

```

        msg = "";
        ValueStack stack =
ActionContext.getContext().getValueStack(); //?
        stack.set("bookList",list); //
        forword = "success";
    }

    else{
        msg= "所选类型暂无书籍在售";
        forword = "error";
    }

    return forword;
}

```

图书类型Action类（显示图书类型，分类显示等）
BookTypeAction.java

生成验证码图片Action类
CreateImageAction.java

订单类型Action类（增加订单，显示订单等）
OrderAction.java

```

public String InserttoOrder(){
    String forword = "";
    HttpSession session = request.getSession();
    ArrayList<Shoppingcart> list = (ArrayList)
session.getAttribute("list");
    ArrayList<Book> list1 = (ArrayList)
session.getAttribute("list1");
    Orders1Entity order1 = new Orders1Entity();
    user = (User) session.getAttribute("currentUser");
    int flag1 = 0;
    System.out.println(choice);
    for (int i = 1; i < choice.length()-1; i++){//传过来
的值是'012'，所以从1开始，到length-1结束
        //int j = choice.charAt(i)-48;
        String str = String.valueOf(choice.charAt(i));
        int j = Integer.parseInt(str);
        System.out.println(j);

    order1.setOrderId(UUID.randomUUID().toString().replace("-", ""));
    order1.setBookId(list1.get(j).getBookId());
}

```

```

// System.out.println(user.getTelephone());
// order1.setTelephone(user.getTelephone());
// order1.setAdress(user.getAdress());
// System.out.println(user.getAdress());s
// order1.setCount(list.get(j).getCount());
// order1.setUserId(user.getUser_id());
// order1.setStatus(0);
String current = new SimpleDateFormat("yyyy-MM-
dd HH:mm:ss").format( new Date());
order1.setTimeee(current);

order1.setPrice(list.get(j).getCount()*list1.get(j).getPrice()
);

orderService.insertintoorder(order1);

deletebook(list1.get(j).getBookId());

}

msg = "订单已经生成！";
return "success";
}

```

购物车类型Action类（增加到购物车，显示购物车删除等）
ShoppingcartAction.java

```

//显示购物车
@SuppressWarnings("unchecked")
public String Shoppingcart()
{
    HttpSession session=request.getSession();
    if(session.getAttribute("userId")==null)
    {
        msg="请先登录";return ERROR;
    }
    else
    {
        String userId=(String)session.getAttribute("userId");
        System.out.println("ID:"+userId);
        Map session1 = ActionContext.getContext().getSession();

        List<Shoppingcart> list = null; //购物车列表
        list = shoppingcartService.Findlist(userId);
        System.out.println("购物车商品数量: "+list.size());

        List<Book> list1 = new ArrayList<Book>(); //通过购

```

车list表 得到图书表

```

list1 = shoppingcartService.findbyid(list);
for (int i = 0; i < list.size(); i++) {
    session1.put("book"+i,list1.get(i));
    session1.put("shoppingcart"+i,list.get(i)); //将
图书表和购物车表都放入Map session中
}
session.setAttribute("saleprice",0);
session.setAttribute("sumprice",0);
if (list.size() != 0 && list1.size() != 0){ //将表放入
栈中
    ValueStack stack =
ActionContext.getContext().getValueStack();
    stack.set("list",list);
    stack.set("list1",list1);
    session.setAttribute("list",list); //购车list
    session.setAttribute("list1",list1); //图书表
list1 都存放到session中，可供删除等方法使用
    msg = "查找成功!";
    return SUCCESS;
}
else
    return ERROR;
}

```

5.视图层

用Jsp,Struts标签, Js和HTML等技术设计的页面

四、系统目录及页面跳转流程图

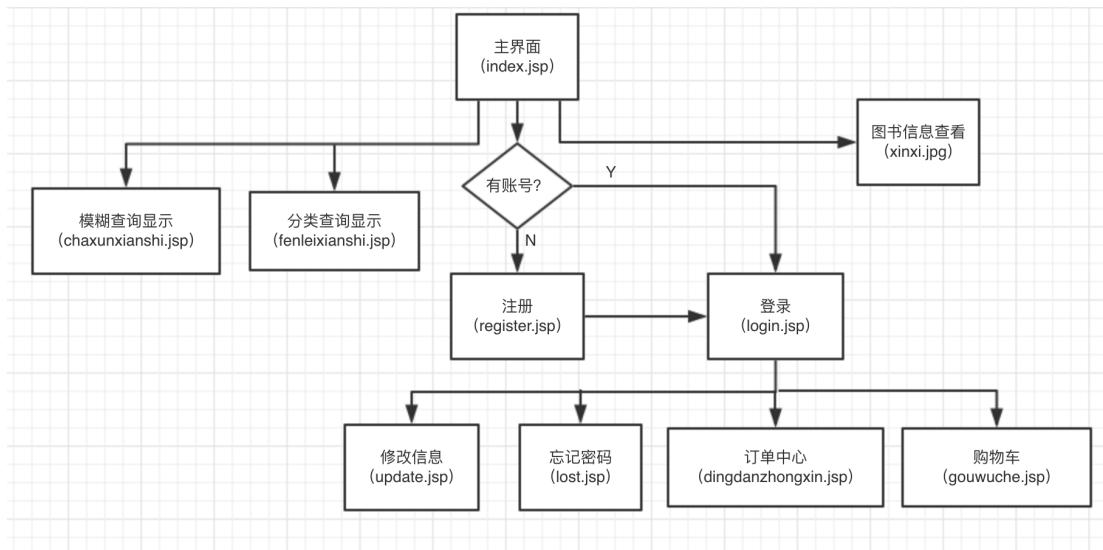
1、系统目录

```

> admin
> css
> image
> js
> META-INF
> WEB-INF
  1.jsp
  about.jsp
  chaxunxianshi.jsp
  dingdanzhongxin.jsp
  fenlei.jsp
  fenleixianshi.jsp
  findword.jsp
  gouwuche.jsp
  hot.jsp
  index.jsp
  login.jsp
  lost.jsp
  new.jsp
  order.jsp
  register.jsp
  self.jsp
  success.jsp
> test.js
> update.jsp
> xinxi.jsp

```

2、页面跳转流程图



五、管理员端设计

1. 功能需求和分析

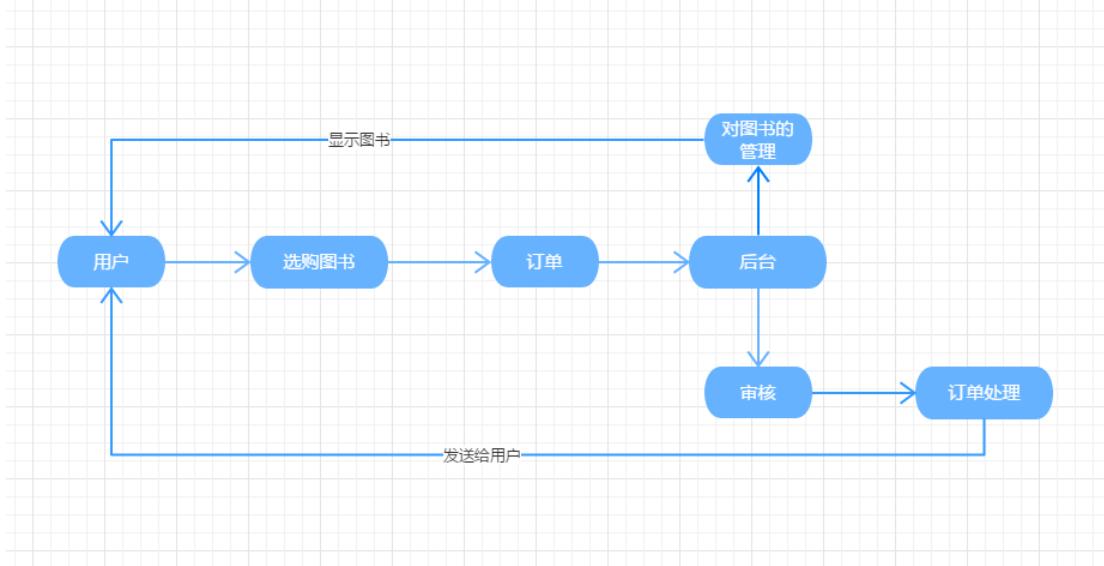
图书商城后台主要是对图书的查询、插入和修改，以及对订单的查询和修改等功能的实现，其主要是通过Ajax与后台交互实现。

管理端实现了订单管理，订单发送，图书管理，上传图书，维护图书价格，修改特价图书等功能，管理员通过管理员账号登录管理端，前后端使用ajax进行异步请求，为简化代码使用了jquery，使用json进行数据交互，管理员点击页面上的按钮后触发js事件向后台发送ajax请求，后台处理数据，通过hibernate操作数据库，查询数据库后将结果封装为json格式发送给前端，js渲染到页面上。后台service层，action层都可使用spring注入到spring容器中，让spring管理对象，在使用这些对象时只需要从容器中取出即可。

使用Ajax有以下几个优点：

1. 通过异步模式，提高用户体验
2. 优化浏览器与服务器之前传输，按需索取
3. Ajax引擎是在客户端运行的，承载了一部分本来由服务器承担的工作

后台管理图书以及处理发来的订单请求的流程图：



首先编写实体类，主要定义对象的属性和功能。例如，图书需要有编号，名称，图片，种类等属性。除此之外还要写好实现功能的接口，映射文件和xml以便对功能的实现。

部分实体类代码如下：

完成后再对后台的前端进行考虑，首先对后台分为两个主要部分：一个是图书管理，一个是订单管理。

图书管理更详细的划分为：

1. 添加图书
2. 对图书的全部显示
3. 显示特价图书
4. 显示非特价图书

5.修改图书

订单的管理可以划分为：

- 1.订单显示
- 2.显示已确认订单
- 3.显示已发送订单
- 4.修改订单

确认好要编写的功能后，首先写图书的模块

对不同的文件进行分类，为不同的文件建立不同的文件夹，像图片，html，css，js等，对他们分类存放

2.前端的界面

为了让界面显示更美观，在前端使用了css进行美化，所以在写的时候引入所需要的class等。

显示所有图书界面：

显示所有图书是显示数据库内的所有图书信息，主要是在js中通过for循环把所有的数据库信息调出来。

显示界面如下图所示：

	图书编号	图书名称	图书图片	图书种类	剩余数量	图书价格	是否是特价	图书特价价格
1	生物化学之书	bkimg/1.png	7	60	65	1	69	
10	编程珠玑	bkimg/10.jpg	4	50	60	0	19	
11	我的第一本编程书	bkimg/11.jpg	4	60	98	0	68	
12	编程之美	bkimg/12.jpg	4	44	78	1	47	
13	编程开发三剑客	bkimg/13.png	4	34	76	1	66	
14	python编程	bkimg/14.jpg	4	44	77	1	55	
15	windows黑客技术编程	bkimg/15.jpg	4	14	77	1	66	
16	财经杂志	bkimg/16.jpg	1	100	111	0	99	
17	智能商业	bkimg/17.jpg	1	44	111	0	100	
18	商业白银	bkimg/18.jpg	1	60	155	1	99	
19	商业生态学	bkimg/19.jpg	1	64	156	0	143	
2	java编程思想	bkimg/2.jpg	2	11	50	0	40	
20	营销的16个关键词	bkimg/20.jpg	1	50	68	0	58	

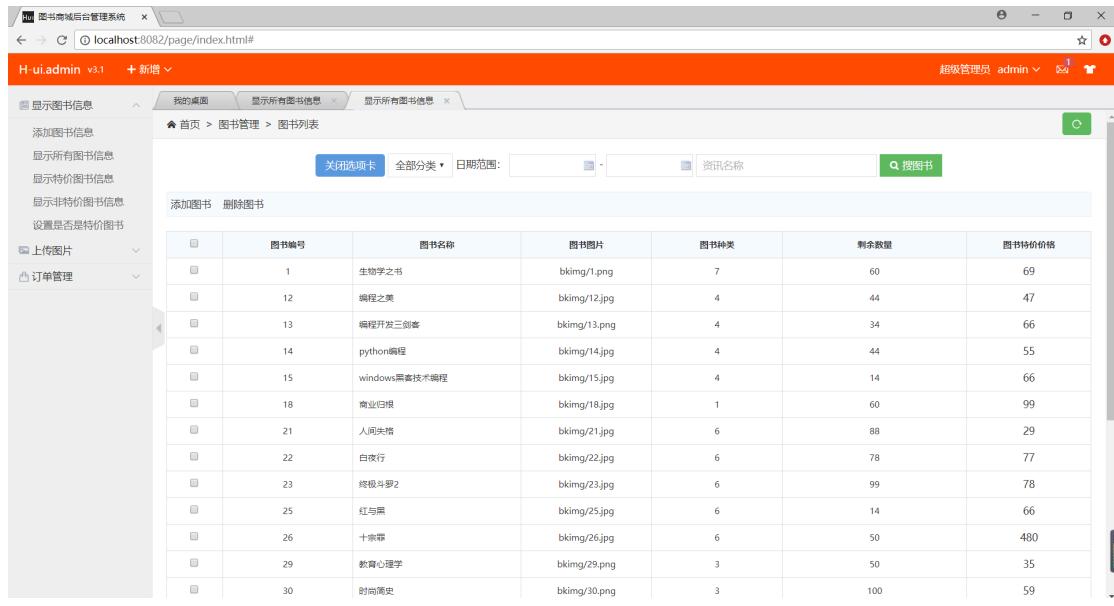
限时特价图书信息和非特价图书信息：

条件显示图书信息是通过数据库中isspecial（是否是特价）来完成的，0是非特价，1是特价。然后把符合内容的全部书籍下显示出来。

界面如下图所示：

显示特价图书信息：

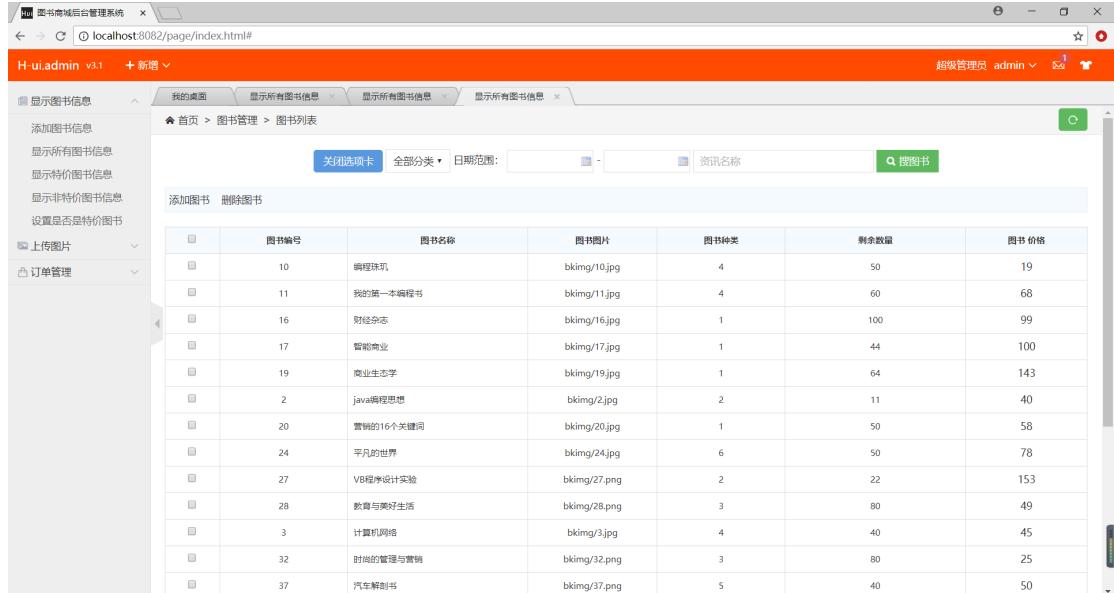
网络书城设计报告



This screenshot shows the 'Book Management' section of the system. The left sidebar includes options like 'Display Book Information', 'Add Book', 'Upload Pictures', and 'Order Management'. The main area displays a table of books with columns: 图书编号 (Book ID), 图书名称 (Book Name), 图书图片 (Book Image), 图书种类 (Book Type), 剩余数量 (Remaining Quantity), and 图书特价价格 (Special Price). The table contains 30 entries.

图书编号	图书名称	图书图片	图书种类	剩余数量	图书特价价格
1	生物学之书	bkimg/1.png	7	60	69
12	编程之美	bkimg/12.jpg	4	44	47
13	编程开发三剑客	bkimg/13.png	4	34	66
14	python编程	bkimg/14.jpg	4	44	55
15	windows黑客技术编程	bkimg/15.jpg	4	14	66
18	商业回报	bkimg/18.jpg	1	60	99
21	人机交互	bkimg/21.jpg	6	88	29
22	白夜行	bkimg/22.jpg	6	78	77
23	终极斗罗2	bkimg/23.jpg	6	99	78
25	红与黑	bkimg/25.jpg	6	14	66
26	十宗罪	bkimg/26.jpg	6	50	480
29	教育心理学	bkimg/29.png	3	50	35
30	时尚简史	bkimg/30.png	3	100	59

显示非特价图书信息：



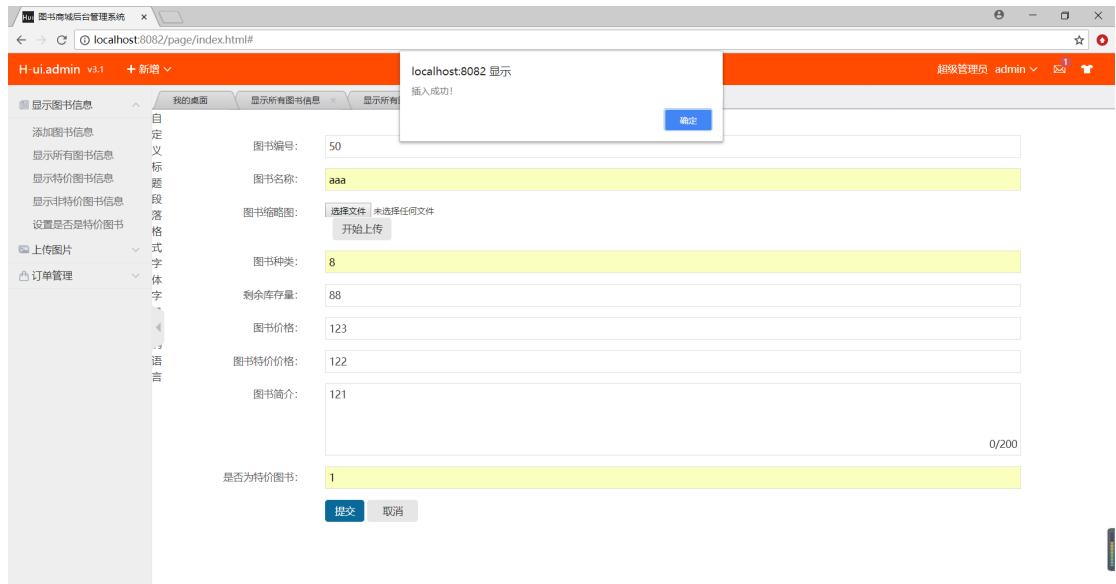
This screenshot shows the 'Book Management' section with a specific filter applied. The table displays books with a special price of 0. The left sidebar and search bar are identical to the previous screenshot.

图书编号	图书名称	图书图片	图书种类	剩余数量	图书价格
10	编程珠玑	bkimg/10.jpg	4	50	19
11	我的第一本编程书	bkimg/11.jpg	4	60	68
16	财经杂志	bkimg/16.jpg	1	100	99
17	智能商业	bkimg/17.jpg	1	44	100
19	商业生态学	bkimg/19.jpg	1	64	143
2	java编程思想	bkimg/2.jpg	2	11	40
20	营销的16个关键词	bkimg/20.jpg	1	50	58
24	平凡的世界	bkimg/24.jpg	6	50	78
27	VB程序设计实验	bkimg/27.png	2	22	153
28	教育与美好生活	bkimg/28.png	3	80	49
3	计算机网络	bkimg/3.jpg	4	40	45
32	时尚的管理与营销	bkimg/32.png	3	80	25
37	汽车解剖书	bkimg/37.png	5	40	50

添加图书信息：

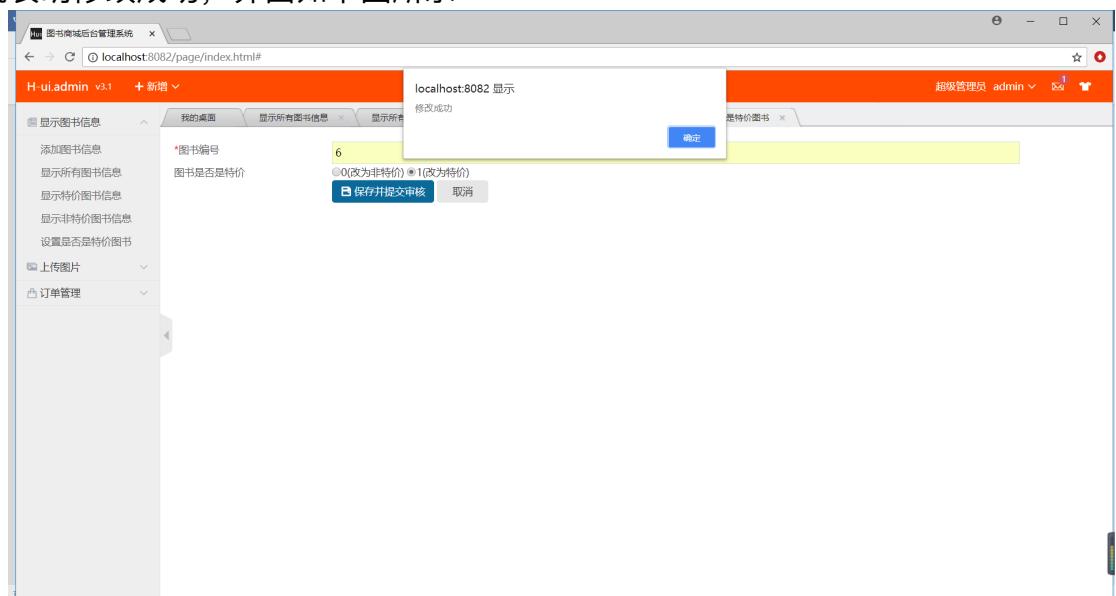
添加图书信息把相关的内容添加到数据库中，其中id是主键，添加成功后会显示添加成功弹框，界面如下所示：

网络书城设计报告



修改图书信息：

首先通过id选择修要修改的图书，然后修改相应的信息，如果弹出修改成功就表明修改成功，界面如下图所示：



完成图书相应的操作之后，然后在写订单部分：

订单部分首先可以显示所有订单信息，跟显示全部图书一样，界面如下图显示：

网络书城设计报告

The screenshot shows a web-based management system for a bookstore. The top navigation bar includes links for '我的桌面' (My Desktop), '查询信息' (Query Information), '首页' (Home), '产品管理' (Product Management), and '品牌管理' (Brand Management). The main content area is titled '修改订单状态' (Modify Order Status) and displays a table of 15 orders. Each row contains columns for '订单编号' (Order ID), '图书编号' (Book ID), '库存' (Inventory), '地址' (Address), '电话' (Phone), and '订单状态' (Order Status). The status column shows values like 0, 1, or 2.

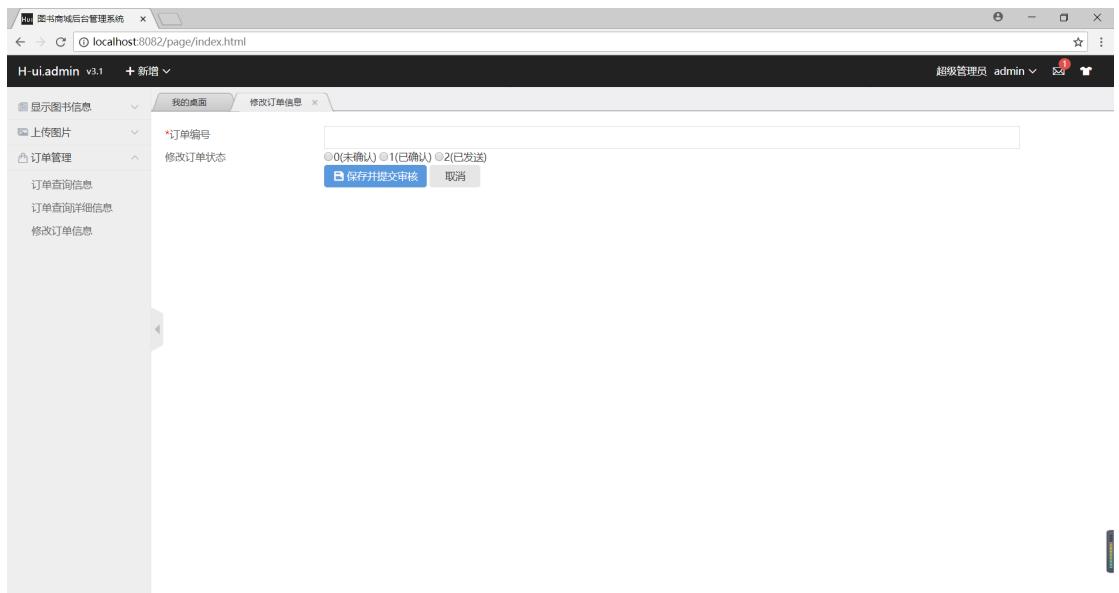
订单编号	图书编号	库存	地址	电话	订单状态
041a3cbce6714edb3b4785fc230fed2	28	5	6789	6789	0
057816ec90ab46eb3404ea59662f7e3	16	6	6789	6789	0
08897d104dae46c3855ba6ca2aeac385	17	5	6789	6789	0
0ca879b7a7d1472f9d4679b1def241b5	45	2	6789	6789	0
1413ef233c674889f73245fc58915905	45	2	6789	6789	0
16aea062614a4d1f8944c3095795f0f	27	2	山东省泰安市山东农业大学	18765332127	0
16c2aca649b34e958875807d0489cca5	16	1	6789	6789	0
189a1c2ef7bd0446b688228a882154d203	17	3	6789	6789	0
19d1b3740c4145028087bd3c418fa481	19	5	6789	6789	0
1a8c2d0e0a0a408dafb191776dcfecc	17	1	6789	6789	0
1b2298cec354febe9913690c4aa33fb	18	2	6789	6789	0
1bbebe7d2e79a4f28b6f797bee3496b5b	27	2	山东省泰安市山东农业大学	18765332127	0
1c9efef03b79441da1dd5c7c35356007	17	7	6789	6789	0

此外还可以通过查询订单号单个显示详细订单信息，显示结果如下图所示：

This screenshot shows a detailed view of a specific order. The top navigation bar and sidebar are identical to the previous screenshot. The main content area is titled '订单查询详细信息' (Detailed Order Query Information) and displays a table with one row of data. The table has columns for '订单编号' (Order ID), '图书编号' (Book ID), '库存' (Inventory), '地址' (Address), '电话' (Phone), and '订单状态' (Order Status). The status is shown as 0.

订单编号	图书编号	库存	地址	电话	订单状态
041a3cbce6714edb3b4785fc230fed2		80	6789	6789	0

还有通过修改订单状态来让用户知道自己的图书是否被发出，1代表已确认，2代表已发送，后台可以修改这个数值，显示页面如下：



3.部分源代码

```

index.html
<div class="menu_dropdown bk_2">
    <dl id="menu-article">
        <dt><i class="Hui-iconfont"></i> 显示图书信息<i class="Hui-iconfont menu_dropdown-arrow"></i></dt>
        <dd>
            <ul>
                <li><a data-href="article-add.html" data-title="添加图书信息" href="javascript:void(0)">添加图书信息</a></li>
                <li><a data-href="article-list.html" data-title="显示所有图书信息" href="javascript:void(0)">显示所有图书信息</a></li>
                <li><a data-href="special.html" data-title="显示所有图书信息" href="javascript:void(0)">显示特价图书信息</a></li>
                <li><a data-href="normal.html" data-title="显示所有图书信息" href="javascript:void(0)">显示非特价图书信息</a></li>
                <li><a data-href="setbook.html" data-title="设置是否是特价图书" href="javascript:void(0)">设置是否是特价图书</a></li>
            </ul>
        </dd>
    </dl>
    <dl id="menu-picture">
        <dt><i class="Hui-iconfont"></i> 上传图片<i class="Hui-iconfont menu_dropdown-arrow"></i></dt>
        <dd>
            <ul>
                <li><a data-href="picture-list.html" data-title="上传图片" href="javascript:void(0)">上传图片</a></li>
            </ul>
        </dd>
    </dl>
    <dl id="menu-product">
        <dt><i class="Hui-iconfont"></i> 订单管理<i class="Hui-iconfont menu_dropdown-arrow"></i></dt>
        <dd>
            <ul>
                <li><a data-href="product-brand.html" data-title="查询信息" href="javascript:void(0)">订单查询信息</a></li>
            </ul>
        </dd>
    </dl>

```

```

        <li><a data-href="orderDetailed.html" data-title="订单查询详细信息"
href="javascript:void(0)">订单查询详细信息</a></li>
        <!--<li><a data-href="product-category.html" data-title="已确认的订单"
href="javascript:void(0)">已确认的订单</a></li>-->
        <!--<li><a data-href="product-list.html" data-title="已发送的订单"
href="javascript:void(0)">已发送的订单</a></li>-->
        <li><a data-href="product-modify.html" data-title="修改订单信息"
href="javascript:void(0)">修改订单信息</a></li>
    </ul>
</dd>
</dl>
</div>
添加页面
<article class="page-container">
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书编号：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text" value="" placeholder="" id="bookId" name="bookId">
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书名称：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text" value="" placeholder="" id="bookName" name="bookName">
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书缩略图：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <div class="uploader-thum-container">
                <div id="fileList" class="uploader-list"></div>
                <input type="file" class="bookfile" multiple="multipl" id="bookfile">
            </div>
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书种类：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text bookType" value="" placeholder="" id="type" name="type">
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">剩余库存量：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text" value="" placeholder="" id="count" name="count">
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书价格：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text" placeholder="" id="price" name="price">
        </div>
    </div>
    <div class="row cl">
        <label class="form-label col-xs-4 col-sm-2">图书特价价格：</label>
        <div class="formControls col-xs-8 col-sm-9">
            <input type="text" class="input-text" value="" placeholder="" id="specialPrice" name="specialPrice">
        </div>
    </div>
</article>

```

```

name="specialPrice">
    </div>
</div>
<div class="row cl">
    <label class="form-label col-xs-4 col-sm-2">图书简介: </label>
    <div class="formControls col-xs-8 col-sm-9">
        <textarea name="description" id="description" cols="" rows="" class="textarea"
            placeholder="说点什么...最少输入10个字符" datatype="*10-100" dragonfly="true"
            nullmsg="备注不能为空! "></textarea>
        <p class="textarea-numberbar"><em class="textarea-length">0</em>/200</p>
    </div>
</div>

<div class="row cl">
    <label class="form-label col-xs-4 col-sm-2">是否为特价图书: </label>
    <div class="formControls col-xs-8 col-sm-9">
        <input type="text" class="input-text" placeholder="" id="isspecial" name="isspecial">
    </div>
</div>

<div class="row cl">
    <div class="col-xs-8 col-sm-9 col-xs-offset-4 col-sm-offset-2">
        <button class="btn btn-primary radius" id="addbook">提交</button>
        <a href="welcome.html">
            <button class="btn btn-default radius" type="button">&ampnbsp&ampnbsp取消&ampnbsp&ampnbsp&ampnbsp</button>
        </a>
    </div>
</div>
显示全部图书js
$(document).ready(function () {
    // var id($('.d');
    var price=300;
    var bookId=1;
    // $(".button1").click(function () {
    $.ajax({
        url: '/books',// 跳转到 action
        data: {//向后台发送的数据
            // drugname: drugname,
            // price:price
        },
        type: 'get',
        cache: false,
        dataType: 'json',
        success: function (data) {
            console.log(data);
            for (var i = 0; i < data.length; i++) {
                $('.tbody').append('<tr class="text-c">\n' +
                    '    <td><input type="checkbox" value="" name=""></td>\n' +
                    '    <td class="bookid">' + data[i].bookId + '</td>\n' +
                    '    <td class="text-l">' + data[i].bookName + '</td>\n' +
                    '    <td>' + data[i].picture + '</td>\n' +
                    '    <td>' + data[i].type + '</td>\n' +
                    '    <td>' + data[i].count + '</td>\n' +
                    '    <td>' + data[i].price + '</td>\n' +
                    '    <td class="td-status">' + data[i].isspecial + '</td>\n' +
                    '    <td class="f-14 td-manage">' + data[i].specialPrice + '</td>\n' +
                    '</tr>');
            }
        },
    });
}
)

```

```

error: function () {
    // view("异常！");
    alert("未找到该类图书");
}
});
});
添加图书js
function submit() {
    var userId=$("#userId").val();
    var userName=$("#userName").val();
    var password=$("#password").val();
    $.ajax({
        type:'put',
        url:'books',
        data:{
            userId:userId,
            userName:userName,
            password:password
        },
        success: function(data){ //回调函数， data为形参， 是从login-cl.php页面返回的值
            {
                $('#window').dialog("close");
                //判断是否成功
                if(data.result=="true")
                {
                    $('#btn_close').click();
                    alert('恭喜你， 修改成功！');
                }else{
                    alert('抱歉， 修改失败！');
                }
            }
        }
    })
}

```