



Heilongjiang University of Science and Technology

计算机与信息工程学院

软件工程|工程专业

专业综合训练 II 设计报告

专业班级：____ 软件工程 16-4 班

姓 名：____ 艾 程

学 号：____ 2016024427 (08)

指导教师：____ 顾泽元

2019 年 7 月 4 日

成绩评定

设计题目：_____图书网购系统_____

负责模块：_____购物车管理模块_____

考核项目		分值	A	C	得分
设计态度 (共 10 分)		10	按要求出勤, 设计态度认真、积极。	设计态度比较认真。	
设计情况 (共 30 分)	设计工作量及难度	10	设计工作量达到要求, 设计具有一定难度。	设计工作量与难度一般, 基本达到了要求。	
	设计方案	5	设计方案正确、合理。	设计方案较正确、基本合理。	
	设计完成情况	15	能够按照软件工程思想完成选题的分析与设计, 界面设计合理, 算法设计正确, 运行结果正确。	基本完成了选题的设计内容及主要功能, 相关算法设计基本正确, 运行结果基本正确。	
设计报告 (共 30 分)	报告组织结构及内容	15	报告内容充实、结构合理、层次清晰, 能合理利用图表辅助说明, 专业术语应用准确。	报告内容较充实、结构较合理、层次较清晰, 图表应用基本得当, 专业术语使用基本准确。	
	报告排版格式	15	格式规范, 完全符合版式要求。	格式基本规范, 基本符合版式要求。	
答辩情况 (共 30 分)		30	陈述思路清晰、语言流利, 系统运行流畅, 问题回答准确。	陈述过程较清晰、流利, 系统主体功能运行正常, 问题回答基本准确。	
综合得分					
成绩评定人员签名					

1 总体设计

1.1 设计简介

随着科技的发达以及网络的普及，网购成为人们生活中必不可少的一部分。人们在去线下书店购书的同时，也逐渐因为网购的便利性、优惠性选择上网购书，因此本次课程设计我们组设计了一个图书网购系统，既跟得上时代要求，也顺应了大众需要。

本系统采用 Hibernate 和 Spring 框架技术完成系统编码。前端使用采用 JavaScript+Jsp 语言，运用了 jquery+css 技术，数据库使用 MySQL 保存数据。系统主要的目标是开发一个页面对用户友好，用户方便操作的图书网购系统。

主要功能有用户账号注册登录、购物车管理、后台管理功能实现、图书搜索、页面信息展示等。通过信息化的建设，使图书信息能更好地展示给大家。并可以附带达成根据数据研究用户图书喜好，做到精准推荐提高竞争力的目的。设计的具体实现功能包括：图书详细信息页面显示、用户注册登录、搜索图书、加入图书到购物车、下单、查询历史订单等功能。管理员可以通过后台登陆添加图书、查看订单、修改类目等功能。

总而言之，对于用户，核心是：下单买书，浏览图书信息，能更方便搜索到图书信息。对于管理员而言，核心是：管理书店、图书信息，而查看订单、用户管理等是附带功能。

1.2 功能模块及分析

根据设计简介，可知我们需要的主要模块是：对于用户而言，需要账号的注册登录操作模块，以及实现下单功能的模块，快速搜索图书的功能模块以及浏览界面的模块。而对于管理员而言，需要后台模块管理图书、用户、订单等信息。

因此本系统业务逻辑模块有五大类——用户账号模块、购物车管理模块、后台管理模块、图书搜索模块、页面信息展示模块。根据功能，其中用户账号模块又分为用户账号注册/登陆管理子模块。购物车管理模块又分为订单查询、增删图书、账目结算子模块。后台管理模块又包括了订单管理、顾客管理、图书管理、类目管理、用户管理子模块。页面展示模块则是展示各种各样的信息。而我在其中负责购物车管理模块。

1.2.1 总体功能模块介绍

由上述所言，系统总体功能模块图如图 1 所示。

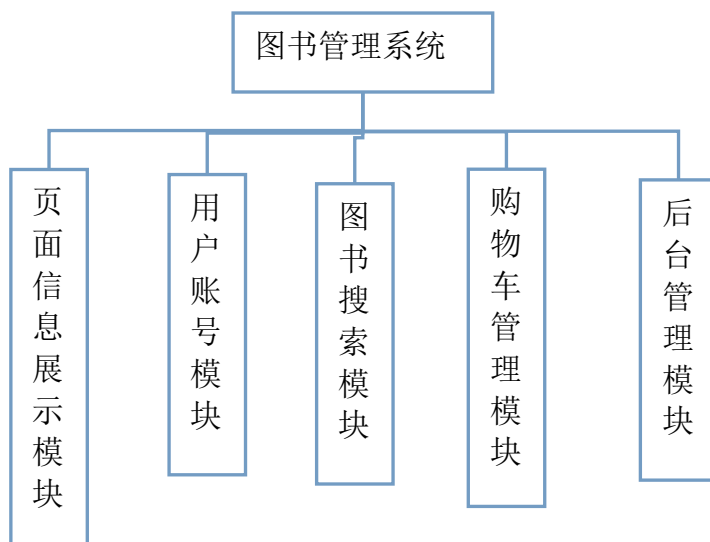


图 1 图书网购系统总体功能模块图

1.2.2 用户账号模块

用户账号模块主要与用户账号相关信息有关，例如用户的注册、登陆功能。功能模块图如图 2 所示。

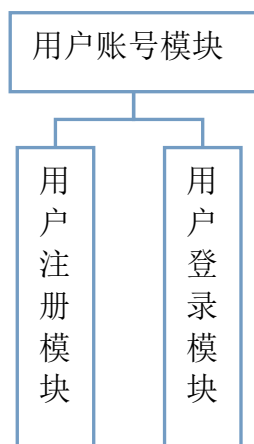


图 2 用户账号模块图

1.2.3 页面信息展示模块

页面信息展示模块主要是实现将图书信息展示给用户的功能。根据后台数据库的数据，其中包括书店首页、书店简介、精品推荐、最新出版、优惠促销的信息展示。功能模块图如图 3 所示。

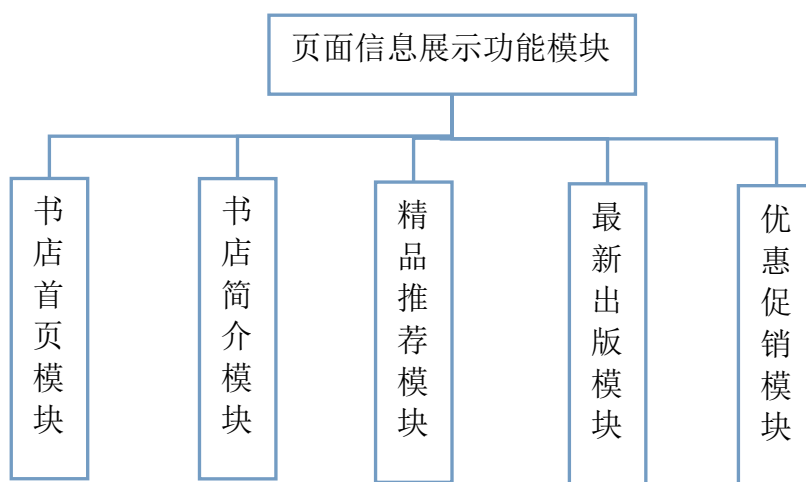


图 3 页面信息展示功能模块图

1.2.4 图书搜索模块

图书搜索模块虽然和页面信息模块类似，都相当于“查”操作，但是前者是主动的，用户可选的，后者是由系统被动提供的，用户必须看的。因此单开一个功能模块，以便区分。

1.2.5 购物车管理模块

购物车管理模块是我负责的一个模块，是针对用户而言的。根据需求，其中涉及到的功能有：用户能够查看、修改购物车中的商品数目、种类以及能够查看统计的商品总价格，完成下单操作，以及查看历史订单功能。如图 4 所示。

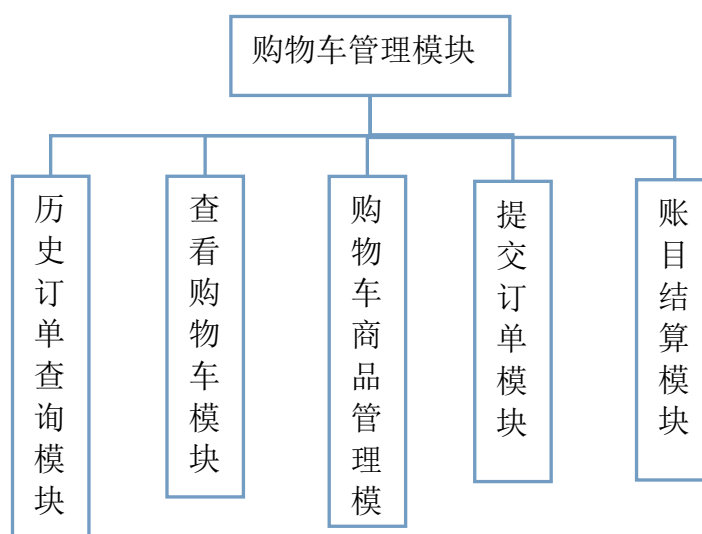


图 4 用户的购物车管理模块图

1.2.6 后台管理模块

后台管理模块主要是用来给管理员操作的，用户账号是接触不到的，其主要功能包括了订单管理（可以查看未处理/已处理订单）、顾客管理（可以查看顾客列表、添加顾客）、图书管理（这里面的信息提供给用户查阅）、类目管理（图书分类）、用户管理功能子模块。功能模块图如图 5 所示。

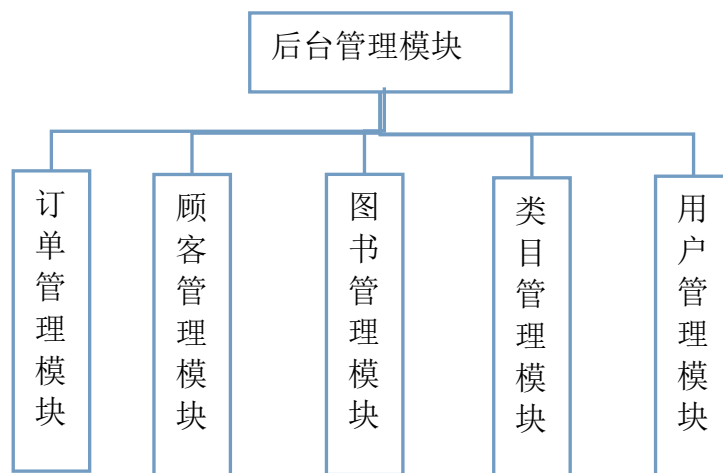


图 5 后台管理功能模块图

1.3 数据库设计

1.3.1 数据库环境说明

由于我负责的是购物车管理模块，因此我在此仅说明购物车管理模块的数据库设计部分。本设计采用 MySQL 作为数据库。

MySQL 是一种开放源代码的关系型数据库管理系统，使用最常用的数据库管理语言--结构化查询语言（SQL）进行数据库管理。本学期我们学习了 MySQL 数据库的相关知识和操作。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。所以本次设计也采用 MySQL 作为网站数据库。

1.3.2 逻辑设计

根据系统功能需求和分析，可以得到我负责的购物车模块中需要的核心的数据表是：订单表。由于订单表需要包含下单账号、收件人信息、图书商品信息，而有时候一个订单不止有一个图书商品，因此在此单开一个数据表——订单商品列表，防止订单表中的数据冗余、可阅读性，用它存放订单中的商品信息，例如书籍编号、数量、总价等。

1. 订单表实体

订单表实体 `indent`，根据功能需要属性包括订单号、下单时间、下单人账号、收件人姓名、收件人手机号、收件人地址、商品总数、商品总价格、订单状态。实体关系图如图 6 所示。

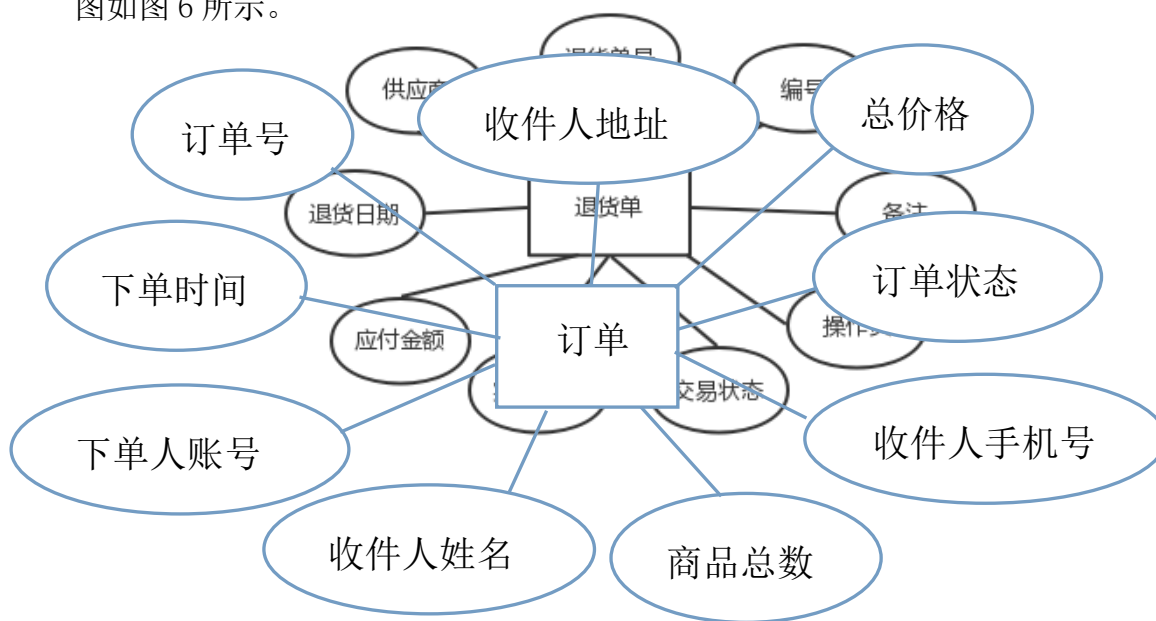


图 6 订单实体属性图

2. 订单商品列表实体

订单商品列表实体 `items`，属性包括购物车号、图书单价、订单编号、图书编号、图书数量。实体关系图如图 7 所示。

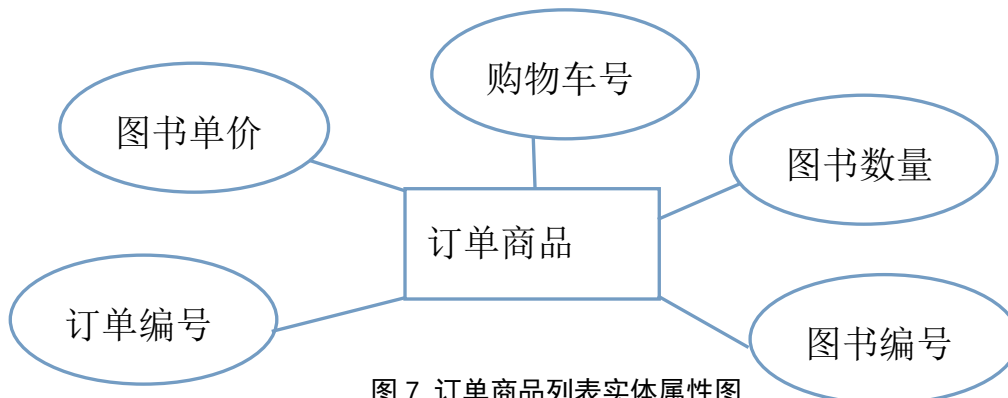


图 7 订单商品列表实体属性图

1.3.3 物理设计

根据实体关系图，可以得到对应的物理实现。

1. 订单表

订单表如表 1 所示。其中关键字是商品总价格，商品总数，其中必要的订单信息还包括：收件人姓名、电话号码以及地址，为了保证订单信息正确，顺势记录下单时间以及账号。订单表的物理设计如下表表 1 所示。

表 1 订单表 indent 表

列名	数据类型	可否为空	说明
id	int	NOT NULL	订单号
total	float	NOT NULL	商品总价格
amount	int	NOT NULL	商品总数
status	int	NOT NULL	订单成功状态值
name	vachar	NOT NULL	收件人姓名
phone	int	NOT NULL	收件人手机号码
address	varchar	NOT NULL	收件人地址
systime	datetime	NOT NULL	下单时间
user_id	int	NOT NULL	下单人账号

2. 订单商品列表 items 表

订单商品列表是订单表的关联表，其关键是订单中图书所属的种类、图书数量、图书单价，相当于对订单中商品信息部分的详细补充。其中 id 编号没有什么具体含义，它按照操作顺序依次自动生成，以便数据表的查询。订单商品列表如表 2 所示。

表 2 订单商品列表 items 表

列名	数据类型	可否为空	说明
id	int	NOT NULL	按顺序自增
price	float	NOT NULL	图书单价
amount	int	NOT NULL	图书数量
book_id	datetime	NOT NULL	图书编号
indent_id	int	NOT NULL	所属订单编号

2 详细设计

2.1 程序设计说明

2.1.1 基础实体 Entity 类

Inden.javat 类：订单类

该类位于 entity 包下，与数据表 1 相对应。列举属性及属性含义如下表 3 所示。该类并无特殊方法，都是下面属性的 getter 与 setter 方法，在此不赘述。

表 3 订单表的实体类实现

成员变量描述	变量类型	名称
订单号	int	id
商品总价格	float	total
商品总数	int	amount
订单成功状态值	int	status
收件人姓名	String	name
收件人手机号码	int	phone
收件人地址	String	address
下单时间	Date	sysstime
下单人账号	User	user

Items 类：订单商品列表类

该类位于 entity 包下，与数据表 2 相对应。列举属性及属性含义如下表 4 所示。该类并无特殊方法，都是下面属性的 getter 与 setter 方法，同样在此不赘述。

表 4 订单商品列表的实体类实现

成员变量描述	变量类型	名称
按顺序自增, 无特别含义	int	id
图书单价	float	price
图书数量	int	amount
图书种类	Book	book
所属订单	Indent	indent

2.1.2 DAO 类

DAO 层是用于和数据库打交道的，购物车管理模块中的用于查询历史订单、提交订单的与数据库有所交互的代码分别放在了 dao 包下的 IndentDao.java 类以及 BaseDao.java 类中，在此摘录关键的具体代码以及注释说明功能。

@Repository // 注册 dao 层 bean 等同于@Component

```
public class IndentDao extends BaseDao{
    /**
     * 按用户账号，查找历史订单记录
     * @param userid 用户账号编号
     * @return 历史订单记录
     */
    public List<Indent> getListByUserid(int userid) {
        return getSession().createQuery("from Indent where user_id="+userid+" order by id desc",
Indent.class).list();
    }
}
```

```
public class BaseDao {
    /**
     * 提交订单数据到数据库中
     * @param object 存放的具体数据结构类型
     * @return 如果成功，返回订单号，否则返回 0
     */
    public Integer save(Object object){
        return (Integer) getSession().save(object);
    }
}
```

2.1.3 BO 类

购物车管理功能模块的业务逻辑类是 IndentService.java 类以及 IndexAction.java 类。

前者是 service 包下的，用于搜索存放 DAO 层中需要的数据类型，然后再将该数据给 DAO 层，经过 DAO 层的代码，将数据提交给数据库。即打包了需要与数据库打交道的功能所需要的数据，例如“提交订单”、“查询某用户历史订单”所需要的数据。当数据经由 DAO 层代码上传成功后，数据也会清零。

而后者是 `action` 包下的，根据用户的功能操作处理数据。用于实现图 4 购物车管理模块涉及到的其他功能，比如查看购物车，购物车中商品数量、种类的增添、删除、减少功能，以及发现“当待提交的订单数据为空时”、“用户未登录”等异常状况。

在此摘录具体的关键代码以及注释说明功能。其中导入需要用到的数据结构类、spring 框架所用需要的包、所需要的实体类由于并非重点内容，因此在此省略。

```
package com.service;

@Service          // 注解为 service 层 spring 管理 bean
@Transactional    // 注解此类所有方法加入 spring 事务，具体设置默认

public class IndentService {

    @Resource
    private IndentDao indentDao;

    /**
     * 创建订单
     * @param bookid
     * @return
     */
    public Indent createIndent(Book book) {
        List<Items> itemList = new ArrayList<Items>();
        itemList.add(createItems(book));
        Indent indent = new Indent();
        indent.setItemList(itemList);
        indent.setTotal(book.getPrice());
        indent.setAmount(1);
        return indent;
    }

    /**
     * 创建订单商品列表
     * @param book
     * @return
     */
    private Items createItems(Book book) {
```

```

        Items item = new Items();
        item.setBook(book);
        item.setPrice(book.getPrice());
        item.setAmount(1);
        item.setTotal(item.getPrice() * item.getAmount());
        return item;
    }

    /**
     * 获取商品总数
     */
    public int getTotal(int status) {
        return (int)indentDao.getTotal(status);
    }

    /**
     * 向订单商品列表中添加项目
     * @param indentList
     * @param bookid
     * @return
     */
    public Indent addItem(Indent indent, Book book) {
        List<Items> itemList = indent.getItemList();
        itemList = itemList==null ? new ArrayList<Items>() : itemList;
        // 如果订单商品列表中已有此书, 数量+1
        boolean noThisBook = true;
        for (Items item : itemList) {
            if (item.getBook().getId() == book.getId()) {
                item.setPrice(book.getPrice());
                item.setAmount(item.getAmount() + 1);
                item.setTotal(item.getPrice() * item.getAmount());
                noThisBook = false;
            }
        }
        // 如果订单商品列表中没有此书, 创建新条目
    }

```

```

        if (noThisBook) {
            itemList.add(createItems(book));
        }
        indent.setTotal(indent.getTotal() + book.getPrice());
        indent.setAmount(indent.getAmount() + 1);
        return indent;
    }

```

```

/**
 * 保存订单，并且提交订单、订单商品列表
 * @param indent
 */
public void saveIndent(Indent indent) {
    indent.setStatus(1);
    indent.setSysTime(new Date());
    int indentId = indentDao.save(indent);
    for (Items item : indent.getItemList()) {
        item.setIndent(indentDao.get(Indent.class, indentId));
        indentDao.save(item);
    }
}

```

```

/**
 * 若提交订单成功，则清空购物车
 */
public Indent deleteIndentItem(Indent indent, Book book) {
    List<Items> itemList = indent.getItemList();
    itemList = itemList == null ? new ArrayList<Items>() : itemList;
    // 如果购物车已有此项目，数量清零
    boolean noneThis = true;
    int itemAmount = 0;
    List<Items> resultList = new ArrayList<Items>();
    for (Items item : itemList) {

```

```

        if (item.getBook().getId() == book.getId()) {
            itemAmount = item.getAmount();
            noneThis = false;
            continue;
        }
        resultList.add(item);
    }
    // 如果已经没有项目, 返回 null
    if (resultList.isEmpty()) {
        return null;
    }
    indent.setItemList(resultList);
    // 如果当前购物车没有项目, 直接返回
    if (noneThis) {
        return indent;
    }
    indent.setTotal(indent.getTotal() - book.getPrice() * itemAmount);
    indent.setAmount(indent.getAmount() - itemAmount);
    return indent;
}

/**
 * 获取某人历史全部订单
 * @param userid
 * @return
 */
public List<Indent> getListByUserId(int userid) {
    return indentDao.getListByUserId(userid);
}
}

package com.action;

public class IndexAction extends BaseAction{
    /**

```

```

    * 查看购物车
    * @return
    */
    @Action("cart")
    public String cart() {
        categoryList = categoryService.getList();
        saleList = bookService.getSpecialList(Book.type_sale, 1, 2);
        Object username = getSession().getAttribute("username");
        if (username!=null && !username.toString().isEmpty()) {
            List<Indent> indentList =
indentService.getListByUserid(userService.get(username.toString()).getId());
            if (indentList!=null && !indentList.isEmpty()) {
                indent = indentList.get(0); // 最后一次订单信息
            }
        }
        return "cart";
    }

    /**
    * 购物车中增加商品
    * @return
    */
    @Action("buy")
    public void buy(){
        Indent indent = (Indent) getSession().getAttribute(indentKey);
        if (indent==null) {
            getSession().setAttribute(indentKey,
indentService.createIndent(bookService.get(bookid)));
        }else {
            getSession().setAttribute(indentKey, indentService.addIndentItem(indent,
bookService.get(bookid)));
        }
        sendResponseMsg("ok");
    }

```

```

/**
 * 购物车中减少商品
 */
@Action("lessen")
public void lessen(){
    Indent indent = (Indent) getSession().getAttribute(indentKey);
    if (indent != null) {
        getSession().setAttribute(indentKey,      indentService.lessenIndentItem(indent,
bookService.get(bookid)));
    }
    sendResponseMsg("ok");
}

/**
 * 购物车中删除商品
 */
@Action("delete")
public void delete(){
    Indent indent = (Indent) getSession().getAttribute(indentKey);
    if (indent != null) {
        getSession().setAttribute(indentKey,      indentService.deleteIndentItem(indent,
bookService.get(bookid)));
    }
    sendResponseMsg("ok");
}

/**
 * 提交订单
 * @return
 */
@Action("save")
public String save(){
    Object username = getSession().getAttribute("username");
    if (username==null || username.toString().isEmpty()) {
        getRequest().setAttribute("msg", "请登录后提交订单!");
    }
}

```



```

        return "login";
    }
    Indent indentSession = (Indent) getSession().getAttribute(indentKey);
    Users user = userService.get(username.toString());
    indentSession.setUser(user);
    indentSession.setName(indent.getName());
    indentSession.setPhone(indent.getPhone());
    indentSession.setAddress(indent.getAddress());
    indentService.saveIndent(indentSession); // 保存订单
    getSession().removeAttribute(indentKey); // 清除购物车
    getRequest().setAttribute("msg", "提交订单成功!");
    return cart();
}

/**
 * 查看用户历史订单
 * @return
 */
@RequestMapping("order")
public String order(){
    categoryList = categoryService.getList();
    saleList = bookService.getSpecialList(Book.type_sale, 1, 2);
    Object username = getSession().getAttribute("username");
    if (username==null || username.toString().isEmpty()) {
        getRequest().setAttribute("msg", "登录后提交订单!");
        return "login";
    }
    indentList
    indentService.getListByUserid(userService.get(username.toString()).getId());
    if (indentList!=null && !indentList.isEmpty()) {
        for(Indent indent : indentList){
            indent.setItemList(indentService.getItemList(indent.getId(), 1, 100)); // 暂
            不分页
        }
    }
}

```

```
        return "order";
    }

}
```

2.1.4 Util 类

使用 java 的 util 包下的 ArrayList 类、Date 类、List 类等数据结构实现购物车管理模块所需要数据的存放，这些数据结构用于与 DAO 层打交道，并且将数据存放到数据库中。

2.2 程序运行效果

启动服务器后，打开浏览器，地址栏输入 https://localhost:8080/book_ssh_mysql/index，进入系统主页面，如图 8 所示。



图 8 系统主页面

任意选择一本書籍，查看詳情，如图 9 所示。



图 9 图书详情页面

点击加入购物车后，主页右侧会有提示，显示购物车中的书籍详情，如图 10 所示。



图 10 右侧提示页面

可以再任选一本书，点击“查看购物车”，可以看到购物车页面的总价与右侧提示的价钱是一致的，有如下图 11 所示。



图 11 购物车页面

可以对购物车中的书籍商品进行增删操作，下图图 12 为增加操作，《相对论》数量加两本。



图 12 购物车增加书本操作页面

同样的，可以进行“减少”操作，如下图 13 所示。

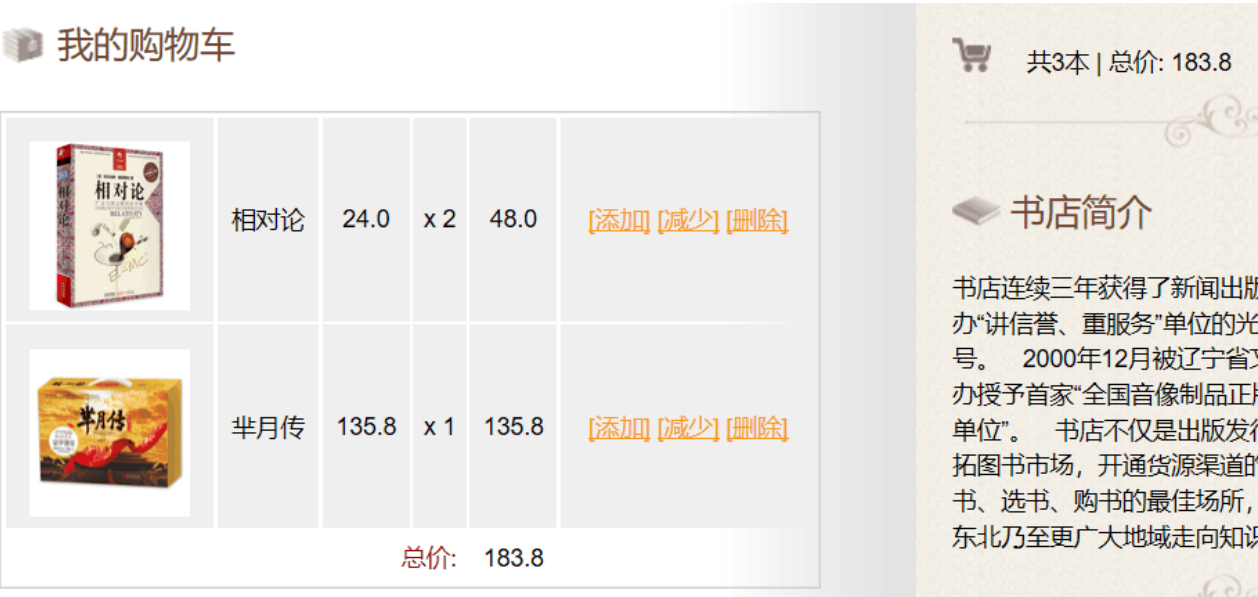


图 13 购物车减少书本操作页面

也可以进行“删除”操作，如下图 14 所示。



图 14 购物车删除书本操作页面

确认了购物车商品信息后，可以点击右下角的“提交订单”，如果该用户还未登录，则提交

订单失败，自动跳转到用户登录页面，如下图图 15 所示。

The image shows a user login interface. At the top left, there is a small icon of a book and the text "用户登录". Below this, there is a dashed-line box containing the text "用户登录" in a brown box, followed by "请登录提交订单!" in red. There are two input fields: "用户:" and "密码:". Below the "密码:" field, there is a link "没有账户? 点击注册" and a brown button labeled "登录".

图 15 因用户未登录提交订单失败页面

点击提交订单时，如果字段是空白的，则会提示是“必填字段”，并且无法成功提交订单。如下图图 16 所示。

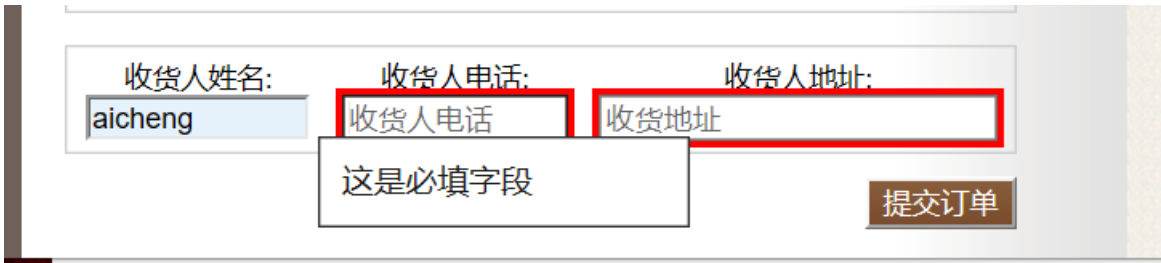
The image shows a form for submitting an order. It has three input fields: "收货人姓名:" with the value "aicheng", "收货人电话:" which is empty, and "收货人地址:" which is empty. A red box highlights the empty "收货人电话:" and "收货人地址:" fields. A tooltip with the text "这是必填字段" points to the empty "收货人电话:" field. A brown button labeled "提交订单" is at the bottom right.

图 16 因字段空白提交订单失败页面

填写完具体信息后，才可以点击提交订单，如下图图 17 所示。

The image shows the same form as in Figure 16, but now all fields are filled. "收货人姓名:" has the value "aicheng", "收货人电话:" has the value "232", and "收货人地址:" has the value "黑科技". The "提交订单" button is still present at the bottom right.

图 17 如实填写收货信息页面

点击“提交订单”成功后，页面显示如下图图 18。



图 18 提交订单成功提示页面

为了便于说明，我选择了图 13 所示的商品下单。查看数据库中的订单 indent 表，可以看到图，该订单的确成功提交到了数据库中。参考最后一行信息即可，前两行是历史测试记录。如下图图 19 所示。

indent @book (Shujuku) - 表

id	total	amount	status	name	phone	address	sysstime	user_id
1	72	3	1	aa	3254546	45645646	2019-07-02	2
2	543	4	1	asdf	sdfg	asdf	2019-07-03	3
3	184	3	1	aicheng	23	黑科技	2019-07-03	3

图 19 数据库中 indent 表

其中订单具体的商品列表 items 表如下。参考三四行即可看出（一二行是测试数据），两个商品分成了两行数据存在一张表中，联系它们的是表示订单号的 indent_id 字段。如下图图 20 所示。

items @book (Shujuku) - 表

id	price	amount	book_id	indent_id
1	24	3	6	1
2	136	4	9	2
3	24	2	6	3
4	136	1	9	3

图 20 数据库中 items 表

点击菜单栏中的“我的订单”，可以查看到刚刚的订单，包括历史订单。如下图图 21 所示。



图 21 用户的历史订单页面

下完订单后可以看到购物车清空了，并且右侧提示框也没有了，如下图图 22 所示。



图 22 下单后的购物车列表页面

3 结论

I 遇到的问题及解决办法

本次项目中，我所涉及的购物车管理功能模块的代码编写涉及到的技术并不困难，大体框架容易写，只是更多的时间花费在了细枝末节的小 bug 上，解决了这些 bug，就相当于学到了一些新的东西：

① 页面不跳转

由于 Spring 中可以使用注解代替配置文件中部分代码，因此在学习过程中大部分使用注解，以代替配置文件中的代码。然而由于注解容易被忽略，编程时有时候容易遗忘注解，因此导致了页面不调整/数据传输失败的问题。

② 数据库连接失败

优先查看配置文件，其次再去寻找具体的代码问题才更有效率。通常是连接 JDBC 的配置文件出现了问题。通过检查数据库名字、账号、密码可以改正过来。

③ 如果新增记录的某个属性（例如编号）不需要我们写，需要自动生成，则应将该属性设置成自增（例如 items 表中的 id 值），否则报错。

④ 传递给数据库的值的数据类型要和数据库的一一匹配，否则报错。但是得到数据库的值的时候，数据类型可以不一样。

还有一些操作失误、细节错误，比如出现了计算总数错误、忘记导包导致语句失效等的小问题。这种很容易找出问题所在并且下次不容易再犯的错误，这里就不一一记录。

II 本人学到的东西有

本次课设是组队的项目，通过本次课设，我深刻了解了互相合作配合的流程以及重要性。并且也在本次课设中再次学习到了很多。

加深了课堂上曾经学到过的知识：

- 1、重温了如何利用 Spring 框架连接数据库，以及数据传输。
- 2、重温了数据库的数据传输与交换的逻辑代码的书写流程。
- 3、重温了如何写 hibernate 配置文件。
- 4、通过阅读队友的代码，复习了去年学习的 js 语言，jQuery 框架的用法，。
- 5、加深了对空数据异常的处理方法，懂得考虑功能的健壮性。

也同时学到了新的知识：

- 1、了解了 Spring 中的映射原理。
- 2、清楚了 Spring 中能够代替部分配置代码的注解写法。
- 3、学会如何利用 MVC 模型分层，并且分工合作。
- 4、了解了 Spring Boot 框架技术。
- 5、了解了 Spring Cloud 为服务架构技术。

III 可改进的问题

购物车中未下单的商品也可以生成一个列表可以暂时存到数据库中，而不是暂时存放到 session 中。上传到数据库的数据可以拿来研究：为什么这个人加入了购物车却不购买？是因为价格太贵么？以及可以通过购物车中的书籍，分析出顾客想要买什么书，做到精准推销。使得这个项目系统更强大。

IV 本程序用到的设计思想

- (1) 面向对象的原则，本程序中有实体类。
- (2) 界面和数据逻辑分离的原则，初始化过程包括界面的初始化和数据初始化，两者相互独立。
- (3) 代码封装原则，多次调用的语句集写成接口供调用，没有冗余的代码，比如本程序中用到的 BaseDao 类、以及 Util 包下的类。