## 一、背景

当今网络高速发展，网络购物已经不再新奇，它满足人们对各种物品的需求。“书籍是人类进步的阶梯”，对于网上书店来说，无疑是网络购物中心最好的学习例子，作为电商项目它可能比较简单，但麻雀虽小五脏俱全。我们小组决定设计一个拥有用户购物模块和管理员模块的网上书店。

## 二、需求分析

### （一）系统需求综述

网上书店系统应该具有这样的功能：对于用户来说，可以浏览所出售书籍，根据价格或书名等信息查询，可以将商品添加至购物车，可以进行购买，对订单进行签收操作；对于管理者来说，可以对图书进行增删改的处理，对用户订单进行发货等操作。

**1.1具体功能模块及描述**

1. 安全性管理：管理员通过账号和密码登录网上书店系统，管理书籍，管理订单信息，拥有最大的权限；对于普通用户来说，有一定的操作权限，但仅仅限于用户模块。通过服务器端的权限设置和数据库登录权限设置来实现整个系统的安全性。

（1）管理员：对图书进行增删改操作，对订单进行确认发货、查看详情操作。也有普通用户的权限，将商品添加到购物车，清空购物车去结账。

（2）用户：对购物车进行增删改操作，对订单进行结算查看操作，进行登录注册操作。

1. 读者登录管理：该功能模块用于管理用户登录注册注销信息，包括的子功能模块如下图所示

用户管理

用

户

注

销

用

户

注

册

用

户

登

录

图2—1用户管理功能模块

通过对用户登录状态的检查，赋予用户不同权限，登录状态下与未登录状态下模块区别为购物车结账模块和订单查询模块的权限不同。

1. 图书管理：该功能模块用于管理图书相关的信息包含的功能模块如下图所示

图书管理

图

书基本信息

图书增加

图书修改

图

书删除

图2—2“图书管理”功能模块

“图书基本信息”包括名称、价格、作者、销售、库存；“图书增删改”可以对图书进行相关的操作。

4、购物车管理：此功能模块用于管理图书流通环节的相关的操作，包括如下图所示的功能模块。

购物车管理

商品增加

商品删除

商品结账

图2—3“购物车管理”功能模块

“购物车管理”对于用户添加进购物车的图书进行删除、清空、结账等操作。

5、订单模块：订单模块分为管理者订单和用户订单，两者唯一的不同在于管理者订单可以查询所有用户的购买情况，对订单进行发货操作；而用户只可以查询自己的订单信息以及对订单进行收货操作。

订单管理

查询订单信息

用户确认收货

管理者确认发货

管理者对所有订单进行查询和确认发货操作；用户对自己的订单进行查询和确认收货操作。

**1.2 系统总功能图**

**网上书店系统**

**购物车管理**

**图书管理**

**订单管理**

图书增加

图书删除

图书查询

删除

增加

查询

状态修改

图书修改

**用户管理**

登录

注册

注销

图2—4系统总功能图

### （二）数据流图与数据字典

**1．数据流图：**

信息验证成功

用户

网上书店系统

登录或注册

购物车信息

读者

用户

添加商品

订单信息

图书信息

物品结账

图书管理

管理员

管理员

订单信息信息

订单管理

账号信息

用户名、密码

图2-5 顶层数据流图

2

图书管理

总订单列表

订单列表

购物车列表

用户信息表

用户名、密码

结账

1

用户管理

4

系统管理

购物信息

用户信息

用户登录

图书信息

3

订单管理

状态修改

图书类型

查询

图书信息

图2-6 0层数据流图

结账

购书

3.2

结账

3.3

订单管理

总书目表

购物车列表

3.1

加入购物车

未登录

图2-8 2层数据流图

超过图书数量的订单

没有超过库存

3.1.1

购书数量审核

有效

购物车添加

3.1.1

结账订单处理

总书目表

订单号列表

3.1.1

登录状态审核

用户登录

图2-7 1层数据流图

**2．数据字典**

**2.11数据项：**

数据元素名称：书店管理员

别名：管理员

简述：网上书店系统的管理人员

组成：id bigint(20)

密码 varchar (15)

用户名varchar (15)

邮箱 email(30)

有关处理逻辑：系统管理权限的代表

数据元素名称：书店用户

别名：用户

简述：网上书店系统的用户

组成：id bigint(20)

密码 varchar (15)

用户名varchar (15)

邮箱 email(30)

有关处理逻辑：系统管理权限的代表

有关处理逻辑：用户权限的代表

**2.2数据存储**

数据存储名称：书目信息表

别名：书本信息

简述：对书本信息的记录

组成： 编号 id int(11)

书名 name varchar(100)

价格 decimal(11,2)

作者 author varchar(100)

销售 sales int(11)

库存 stock int(11)

图片 img\_path varchar(200)

数据存储名称：订单信息表

别名：订单表

简述：用户的订单信息

组成： 订单号 order\_id varchar(50)

创建时间 create\_time datetime

价格 decimal(11,2)

状态 int(11)

用户编号 user\_id int(11)

数据存储名称：购物车信息表

别名：购物表信息

简述：购物车信息

组成： 物品编号 id int(11)

书名 name varchar(100)

数量 int(11)

单价 decima(11,2)

总价 decimal(11,2)

订单号 order\_id varchar(50)

**2.3数据流**

数据流名称：查询要求

简述：用户输入给系统的查询条件

组成：作者 varchar 20 / 书名 varchar 20 /价格范围 int-int

数据流来源：用户

数据流去处：处理逻辑“查询处理”

数据流名称：查询图书信息

简述：系统输出给管理员和用户的书目信息

组成：编号 id int(11)

书名 name varchar(100)

价格 decimal(11,2)

作者 author varchar(100)

销售 sales int(11)

库存 stock int(11)

图片 img\_path varchar(200)

数据流来源：系统

数据流去处：管理员和读者

数据流名称：购物车信息

简述：用户的购书要求信息

组成：用户id bigint(20)

密码 varchar (15)

用户名varchar (15)

数据流来源：用户

数据流去处：处理逻辑“购书管理”

流通量：100本/天

高峰值流通量：150本/天

数据流名称：订单信息

简述：用户的订单信息

组成：用户id bigint(20)

密码 varchar (15)

用户名varchar (15)

数据流来源：用户

数据流去处：处理逻辑“订单管理”

流通量：100次/天

高峰值流通量：150次/天

数据流名称：增加图书要求

简述：图书的入库信息

组成：编号 id int(11)

书名 name varchar(100)

价格 decimal(11,2)

作者 author varchar(100)

销售 sales int(11)

库存 stock int(11)

图片 img\_path varchar(200)

数据流来源：管理者输入

数据流去处：处理逻辑“入库管理”

流通量：100次/天

高峰值流通量：150次/天

**2.4数据处理**

数据处理名称：查询处理

简述：接收和处理从用户输入的查询要求

输入：查询要求

来源：用户

输出：1、查询成功 图书信息

2、查询失败

功能描述：当用户输入查询条件时，系统将会按要求输出相应的信息

数据处理名称：购物车管理

简述：处理由用户操作的购书要求

输入：购书信息

来源：用户

输出：1、购书成功 去处：外部用户

2、购书失败 去处：系统提示失败

功能描述：用户输入购书要求，系统根据要求处理。

数据处理名称：订单管理

简述：处理用户或管理员订单管理

输入：订单信息

来源：用户/管理员

输出：用户确认收货成功/管理员确认发货 去处：外部用户/外部用户

功能描述：由管理员或用户提出修改订单状态操作，系统处理。

数据处理名称：增加图书管理

简述：管理员请求系统对图书进行入库录入工作

输入：入库要求、书目信息

来源：管理员

输出：1、数目以建立，更改图书数量 去处：管理员

2、建立数目，录入信息成功 去处：管理员

功能描述：由管理员输入入库要求和数目信息，系统根据信息按不同要去处理

### 系统功能需求概述

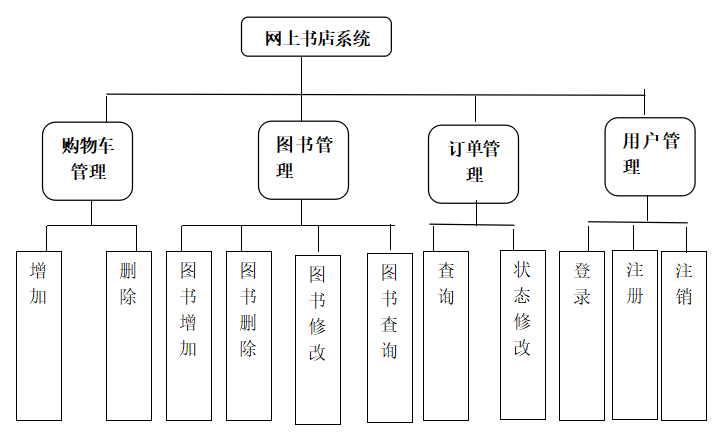


图2-9 系统功能模块图

## 三、系统数据库设计

### （一）概念结构设计

**1、系统E-R模型**

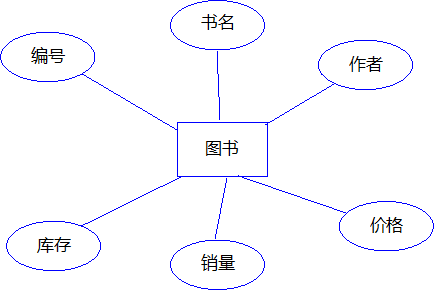


图3-1 图书信息E-R图

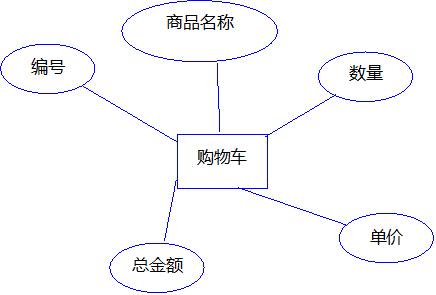
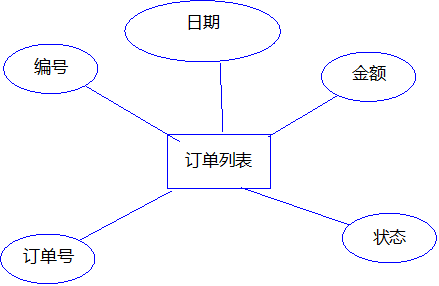


图3-2 购物车信息E-R图



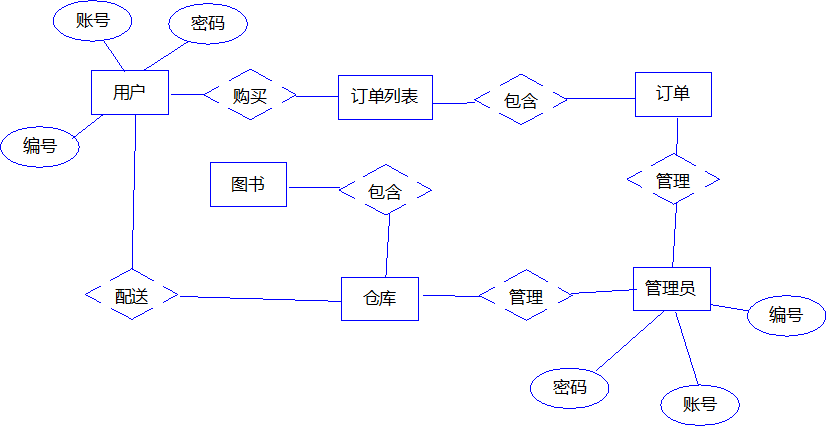
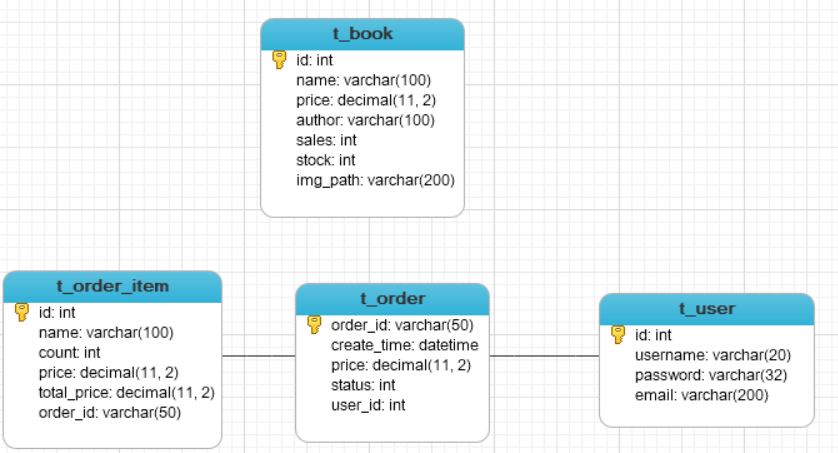


图3-3 订单列表E-R图

图3-4 网上书店系统总的E-R图

### 逻辑结构设计

根据ER图有如下数据：

用户（id，用户名，密码,邮箱）

管理员（id，用户名，密码，邮箱）

图书信息（编号，书名，作者，价格，销量，库存）

购物车信息（编号，数量，书名，单价，总金额）

订单信息（编号，日期，金额，状态，订单号）

图3-4 设计表图（来源于MySQL可视化工具navicat）

## 系统功能分析与设计

### （一）前台模块

查看、检索图书

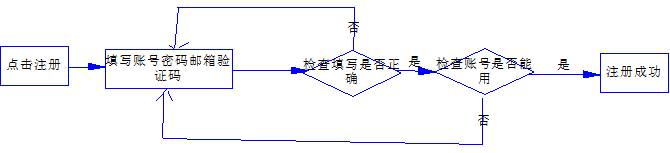
购物车管理

用户订单管理

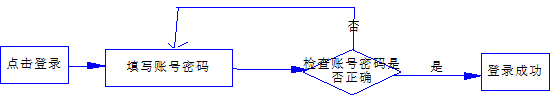
用户

注册登录管理

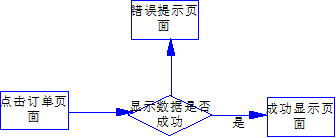
1. **用户注册**



1. **用户登录**



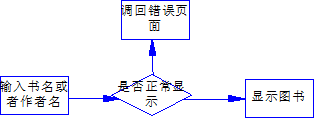
1. **订单查看**



1. **购物车结算**



1. **检索图书**



**（**二）后台模块

图书管理

管理员

管理员订单管理

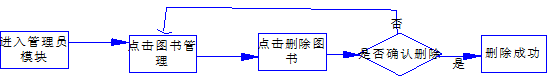
1. 图书增加



1. 图书修改



1. 图书删除



1. 订单查看



## [详细设计](#_Toc32321)

### [（一）开发平台及工具](#_Toc5391)

**开发工具：**

后端java、使用IDEA编写

前端HTML、CSS、JavaScript、JSP

服务器：Tomcat

**DBMS**：MySQL（图形化界面MySQLworkbench/SQLyog/navicat）

**建模工具**：Sybase PowerDesigner

**说明：**

我们小组所要做的是B/S模式的网上书店系统，那么首选的语言就是java，数据库我们选择mysql和java进行JDBC编程，实现前端和数据库的连接。在建模方面我们使用PowerDesigner来实现系统模型分析，使用navicat来分析数据库表之间的联系。

总的开发流程就是前端给出页面设计之后，数据库设计跟上，然后就是后端的java开发。之后就是整个小组一起来进行调试各个错误，以此实现本次课设。

### 用户界面设计

 小组成员利用前端HTML、CSS、JS制作出了前端页面：

1. **首页**
2. **登录页面**

****

1. **注册页面**



1. **用户购物车页面**
2. **我的订单页面**
3. **图书管理页面**



1. **订单管理页面**



**说明：**

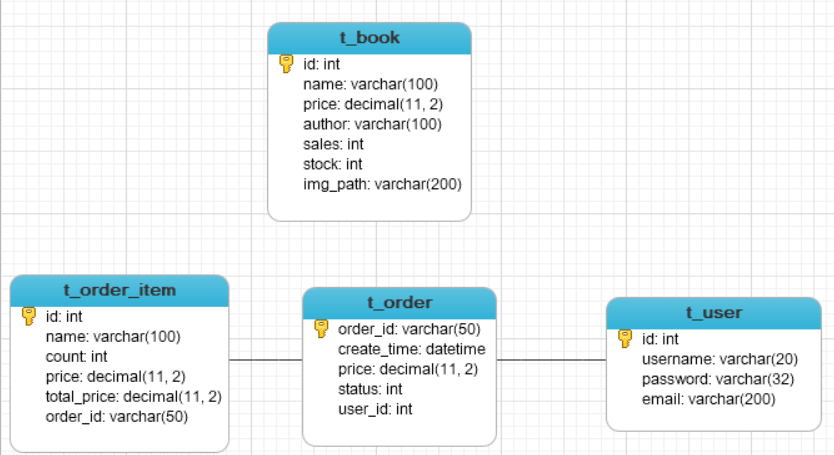
其中我们在注册页面用了EL表达式对用户输入的账号、密码和邮箱进行前端第一步的验证操作。其它页面均为HTML静态页面，之后就开始数据库设计。

### 数据库存储设计

根据前端页面以及之前大实验的分析，我们决定创建book数据库，建库语句如下：

并决定创建4张表如下：

drop database if exists book; #如果原来的数据库存在，就删除  
create database book; #创建数据库  
use book; # 切换到数据库



图为navicat for mysql提供

建表语句如下：

用户表：

create table t\_user(  
 `id` int primary key auto\_increment,  
 `username` varchar(50) not null unique,  
 `password` varchar(32) not null,  
 `email` varchar(50)   
);

图书表：

create table t\_book(  
 `id` int primary key auto\_increment,  
 `name` varchar(100),  
 `price` decimal(11,2),  
 `author` varchar(100),  
 `sales` int,  
 `stock` int,  
 `img\_path` varchar(200)  
);

订单表：  
create table t\_order(  
 `order\_id` varchar(50) primary key, ##订单号  
 `create\_time` datetime not null, ##订单时间  
 `total\_money` decimal(11,2) not null, ##总金额   
 `status` int not null default 0, ##物流状态：0-未发货、1-等待用户签收、2-用户已签收  
 `user\_id` int not null, ##用户编号  
 foreign key (`user\_id`) references t\_user(`id`)  
);

订单项：  
create table t\_order\_item(  
 `id` int primary key auto\_increment,  
 `name` varchar(30) not null, ##商品名  
 `price` decimal(11,2), ##商品单价  
 `total\_money` decimal(11,2), ##商品总金额  
 `count` int not null, ##商品数量  
 `order\_id` varchar(50) not null, ##订单号  
 foreign key (`order\_id`) references t\_order(`order\_id`)  
);

说明：

创建数据库表之后，插入部分图书的信息，然后开始java后端的设计。

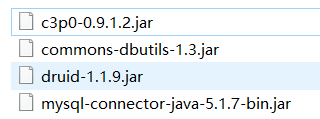
### java后端设计

首先我们确定总的设计模式为MVC（没有选用当下流行的SSM或SpringBoot设计，选用java web设计中简单的MVC模式），选好总体设计模式之后，在IDEA项目中创建好java ee项目，将项目框架的文件夹创建好：web层、service层、dao层、javabean。

java后端开发分为前台模块和后台模块两大部分，其中又可以分为用户模块、图书模块、购物车模块、订单模块和权限管理以及事务管理设计。这里我们就按照后者进行设计，在权限管理部分再将前后台设置清楚。

**用户模块：**

根据数据库表的设计创建我们的javaBean，创建User类，数据与数据库t\_user表相对应。之后要设计我们的数据库层面的连接，这里导入数据库连接的jar包，已经测试包：

 编写基本的数据库连接操作，这里使用JDBCUtils，创建基本的配置文件，在JDBCUtils中编写代码：

static {  
 try {  
 Properties properties = new Properties();  
 //读取jdbc.properties属性配置文件  
 InputStream inputStream = JdbcUtils.class.getClassLoader().getResourceAsStream("jdbc.properties" );  
 //从流中加载数据  
 properties.load(inputStream);  
 //创建数据库连接池  
 dataSource = (DruidDataSource) DruidDataSourceFactory.createDataSource(properties);  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
}

可以进行打印连接进行测试，看看是否连接到了数据库。之后我们编写baseDao类，在里面增加一些增删改查的方法，以便之后连接继承使用。

配置好环境之后，我们编写UserDao接口，在其中添加根据用户名查询用户、根据用户名和密码查询用户、保存用户等方法，然后在UserDaoImpl中实现这个接口。主要为编写sql语句并返回，部分语句如下：

public class UserDaoImpl extends BaseDao implements UserDao {  
 @Override  
 public User querybyUsername(String username) {  
 String sql = "select `id`,`username`,`password`,`email` from t\_user where username = ?";  
 return queryForOne(User.class,sql,username);  
 }  
}

编写完Dao层之后，我们观看前端关于用户页面的服务就有登录、注册等服务，所以我们编写service层，service层用来调用Dao，例如登录的时候调用UserDao中的根据用户名和密码查询用户的方法，通过得到的User类对象是否为空来判断用户登录是否成功，而相应我们前端请求的自然就是Web层的servlet代码。所以我们编写完service层之后要去web层编写相应的UserServlet，并记得在web.xml配置，以便在前端页面引用。这个时候我们可以将之前的前端HTML5页面全部替换为JSP动态页面，其实很简单，就是在前面加上

<%@ page contentType="text/html;charset=UTF-8" language="java" %>一行代码即可，方便后续我们进行动态引入从servlet来的数据，动态的显示。

关于用户登录的模块也就是：



其中在Dao层要进行一次test，在service层也要进行测试，如果出现错误，那么分层开发的优势就可以显示出来，我们可以很快的找出是哪一层出现了错误。在servlet层中我们用到了HTTP的相关知识，使用了doPost或者doGet方法，因此我们将servlet层中复用的代码放到baseServlet中，并利用反射技术去调用我们响应的方法（即通过类找到方法）。

那么用户注册的模块也和登录一致，其中DAO、service层是一至的，我们只需要在servlet层中添加注册的方法即可。

**图书模块**

根据数据库设计javabean，并在book中添加get()、set()方法、toSting()方法以及有参无参构造函数。

然后设计我们的bookDao层，因为我们书籍一般都有增删改查的操作，所以在bookDao中我们有通过id查询图书、删除图书、改变图书的操作，之后测试sql语句是否正确，例如查询操作：

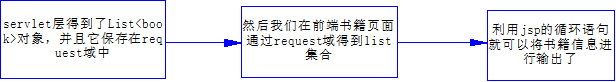
public Book queryBookById(Integer id) {  
 String sql = "select \* from t\_book where id = ?";  
 return queryForOne(Book.class,sql,id);  
}

就是一个简单的sql语句，然后利用我们的数据库编程类库，之间得到我们查询之后的图书对象。

之后的service层也是通过dao层来实现方法，而我们的servlet层也是如此，这个时候我们要去管理员的图书管理页面去设置我们数据显示。这里就用到了JSP技术，也就是为什么之前要将HTML换位JSP页面的原因，可以动态的显示书籍信息，例如：

<div id="main">  
 <table style="height: 350px">  
 <tr>  
 <td>名称</td>  
 <td>价格</td>  
 <td>作者</td>  
 <td>销量</td>  
 <td>库存</td>  
 <td colspan="2">操作</td>  
 </tr>  
 <c:forEach items="${requestScope.page.items}" var="book">  
 <tr>  
 <td>${book.name}</td>  
 <td>${book.price}</td>  
 <td>${book.author}</td>  
 <td>${book.sales}</td>  
 <td>${book.stock}</td>  
 <td><a href="manager/bookServlet?&action=getBook&id=${book.id}&pageNo=${requestScope.page.pageNo}">修改</a></td>  
 <td><a class="deleteClass" href="manager/bookServlet?action=delete&id=${book.id}&pageNo=${requestScope.page.pageNo}">删除</a></td>  
 </tr>  
 </c:forEach>

而整个的逻辑也是很好理解的：



以上我们就可以试下图书的增加、删除和修改操作，在删除操作中我们需要在web页面上添加提示，避免用户不小心删除了数据。

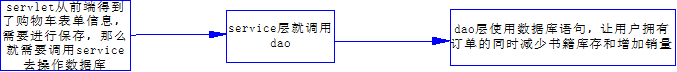
**购物车模块**

购物车模块可以用session来设计，也可以用mysql来设计，或者用redis缓存技术。这里我们小组商量之后决定使用session技术，它的生命周期也就是浏览器的生命周期，即关闭浏览器购物车生命结束，不会保存用户购物车的信息。

选用了session之后，我们直接创建相应的servlet，在里面实现增加、删除的方法。从前端获得数据之后直接保存到session域中，所以我们的增删操作也就是session信息的操作，但是在结账操的时候，就使用数据库操作，将购物车里面的item和order保存在数据库中。这一块和订单模块相连接。

**订单模块**

为了保存用户购买书籍的信息和订单信息，我们依旧和之前一样，这里我们可以从servlet开始，反向操作。我们在servlet中的操作逆行逻辑如下：

****

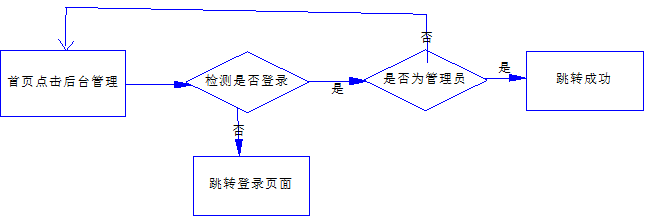
**事务处理**

从上面我们小组分析得到创建订单的同时（也就是用户付款完毕之后），我们需要同时对图书的销售量和存储量进行操作，这一步必须一起执行，或者一起失败，所以这里我们要用事务处理机制。这里我们没有选择在数据库中使用触发器或者设置事务操作。我们选择用ThreadLocal，也就是线程。这里我们首先设想的是利用一个线程关联一个数据的特性，手动的设置事务，成功则提交事务，失败则回滚事务，并在相应操作，也就会上面所说的创建订单+库存销售的修改上加上try-catch操作来实现事务。后面发现是可行的，但是如果后面有更多需要事务的地方就不好处理，所以我们利用Filter来进行过滤，在doFiltr()方法中进行ThreadLocal操作来实现整个的事务处理。

**权限控制**

编写好基础页面之后，小组进行讨论，需要实现前后台的功能，也就是说用户的权限是不能涉及后台管理的，所以我们使用了Filter来控制。我们利用session域来获得登录的对象（如果没有登录那么跳转登录页面），对登录对象进行管理员用户的检测，以此实现权限管理（并在web.xml中进行配置）。

逻辑也就是



那么我们的网上书店就完成了！

## 六、测试及错误总结

### （一）错误总结：

1. 没有500、404错误提示页面，用户体验不好。

解决：增加简单的错误提示页面以及返回链接，将数据库连接中的异常抛给Tomcat服务器，在web.xml配置错误页面来达到次功能。

1. 在将商品添加至购物车时页面每次都跳转一下，页面跳转速度与用户点击速度不匹配。

解决：所以这里我们不在用页面跳转，而是是哟AJAX技术，将我们要提示的字符串用json解析然后发个前端页面即可。

1. 用户可以注册表单二次提交

解决：我们使用response.sendRedirect(req.getHeader("Referer"))重定向来实现，但这个并不是最好的方法，所以后面我们引入了google验证码的第三方jar包，然后在web.xml进行配置，利用验证码来组织表单重复提交。

1. 订单详情页面数据没有显示

解决：从dao层开始寻找问题，发现是因为javabean和sql数据表中的字段不一致，于是加了as别名将数据库表的字段变为了javabean，看来之后一定要一定啊，不然会出现错误。

### （二）总体测试

1. **登录功能**



1. **结账功能**





1. **图书增删改查功能**



### （三）优化

1. 将前端页面相同部分的代码汇总，可以方便之后的修改。
2. 通过查看资料将resquest域中的数据利用webUtils工具类快速注入到对象中。
3. 增加了500、404等错误提示页面。

## 七、[课程设计总结与](#_Toc7461)展望

冲冲冲！

## 八、参考文献

1. 《数据库系统概论第5版》 王珊等 高等教育出版社

2.《UML用户指南（第2版·修订版）》 [美]Grady Booch等 人民邮电出版社

3. CSDN mythreecatup的博客<https://blog.csdn.net/qq_43919400> Servlet+JSP一文完结

4. JDKAPI1.8文档

5. jQueryAPI1.7文档