JIANGXI NORMAL UNIVERSITY

江西师范大学

本科生毕业设计(论文)

中文题目 网上购物管理系统之在线购物子系统的设计与实现

外文题目 Design and Implementation of Online Shopping Management System

<<实词大写开头，虚词小写开头。>>

学　　号 _____

姓　　名 _____

学　　院 软件学院

专　　业 软件工程

指导教师 _____

完成时间 _____

江西师范大学教务处制

声　　明

本人郑重声明：

所呈交的毕业设计（论文）是本人在指导教师指导下进行的研究工作及取得的研究成果。其中除加以标注和致谢的地方，以及法律规定允许的之外，不包含其他人已经发表或撰写完成并以某种方式公开过的研究成果，也不包含为获得其他教育机构的学位或证书而作的材料。其他同志对本研究所做的任何贡献均已在文中作了明确的说明并表示谢意。

本毕业设计（论文）成果是本人在江西师范大学读书期间在指导教师指导下取得的，成果归江西师范大学所有。

特此声明。

声明人（毕业设计（论文）作者）学号：

声明人（毕业设计（论文）作者）签名：

签名日期：　　年　　月　　日

摘 要

<<总共 200~400 字左右，分三段撰写。论文首先就看摘要，所以摘要必须精心组织，简单明了>>

第一段写系统的开发背景（即为什么开发该系统，使用该系统有什么优点）。

第二段写采用什么技术开发，系统完成了哪些业务功能。如：系统基于**架构，应用***等技术开发，实现了**等功能。

第三段写本论文的主要内容。按论文的目录结构，通过一些连接词（首先、其次、然后、接着、最后等）结合动词（介绍、阐述等）串联起来。如本文首先**介绍了系统的开发背景，然后**；从**方面介绍了；详细介绍了**等

关键词：3—5 个（业务词汇+ 方法词汇+ 技术词汇）

Abstract

<<英文摘要待中文摘要定稿后翻译。一定要通顺，最起码不要有明显的语法错误。>>

Key Words: <<中文关键字的翻译>>

目录

第 1 章 绪论

<<这个部分引导读者阅读和理解你的论文，读者从这个部分可以大致理解你要阐述的工作。绪论篇幅不宜过多，但也不能太少，2~3 面为宜。>>

<<论文格式要求：正文内容的字体为宋体、字号为小四、行间距为固定值 20 磅。>>

1.1 系统开发的背景

<<此处可分两个层面层面阐述：>>

<<首先，为什么作者需要开发该系统？ 突出社会发展背景、本系统的出发点等内容>>

<<然后，展开叙述一些传统技术手段及现有同类系统存在的问题>>

1.2 系统开发的目标、意义

<<此处可分两段来写。>>

<<第一段写目标：描述项目愿景，即系统要解决哪些问题（从业务上描述），达到哪些要求（从技术上描述，如安全、性能、移植性、架构等）>>

<<第二段写意义：使用本系统具有哪些好处（相对于不使用本系统）>>

1.3 本文的主要工作

<此处插入作者本人在系统中所负责或者完成的具体工作。如果论文所述项目为集体合作，则应该简略指出作者本人所负责的工作。>

在此项目中，本人完成了**、**、**等模块的设计与实现。

1.4 论文结构

<<此处插入该论文的整体结构，简述论文的整体形式结构，并简述每一章的主旨。>>

第 2 章 需求分析

<<本章是需求分析和系统设计的主要内容简化之后的结果。如果是多人完成一个系统，则主要介绍自己所完成的功能对应的部分。>>

2.1 系统功能需求

<<首先是概述，然后是具体介绍。概述部分可列出主要功能和主要用户类型。具体介绍的功能则比较详细，但只包括后面测试部分涉及的功能。>>

2.1.1 角色分析

<<从系统的角度分析系统的参与者，并给出每一个参与者的描述。>>

从网上购物系统的实际需求分析，系统涉及到以下角色如表 2-1 所示：

表 2-1 系统角色分析

角色	职责
客户(买家)	负责选购商品、支付货款、订单管理和退货处理。
卖家	负责商品管理、发货管理、退货管理和统计查询。
系统管理员	管理和维护整个系统的用户组织结构，负责对用户、角色、用户级别的增、删、改、查等管理。

2.1.2 总体用例分析

<<从系统的使用者的角度使用 UML 的用例图描述系统的用例，并给出每一个用例的用例描述。>>

<<首先，给出系统的总体用例图，下面以网上购物系统为例>>

网上购物系统的参与者主要包括买家、卖家和管理员三个，经过分析，系统包括商品选购、货款支付、订单管理、发货管理、退货管理、登录系统、商品管理、统计查询、用户管理、角色管理和日志管理等 11 个用例，如下图 2-1 所示：

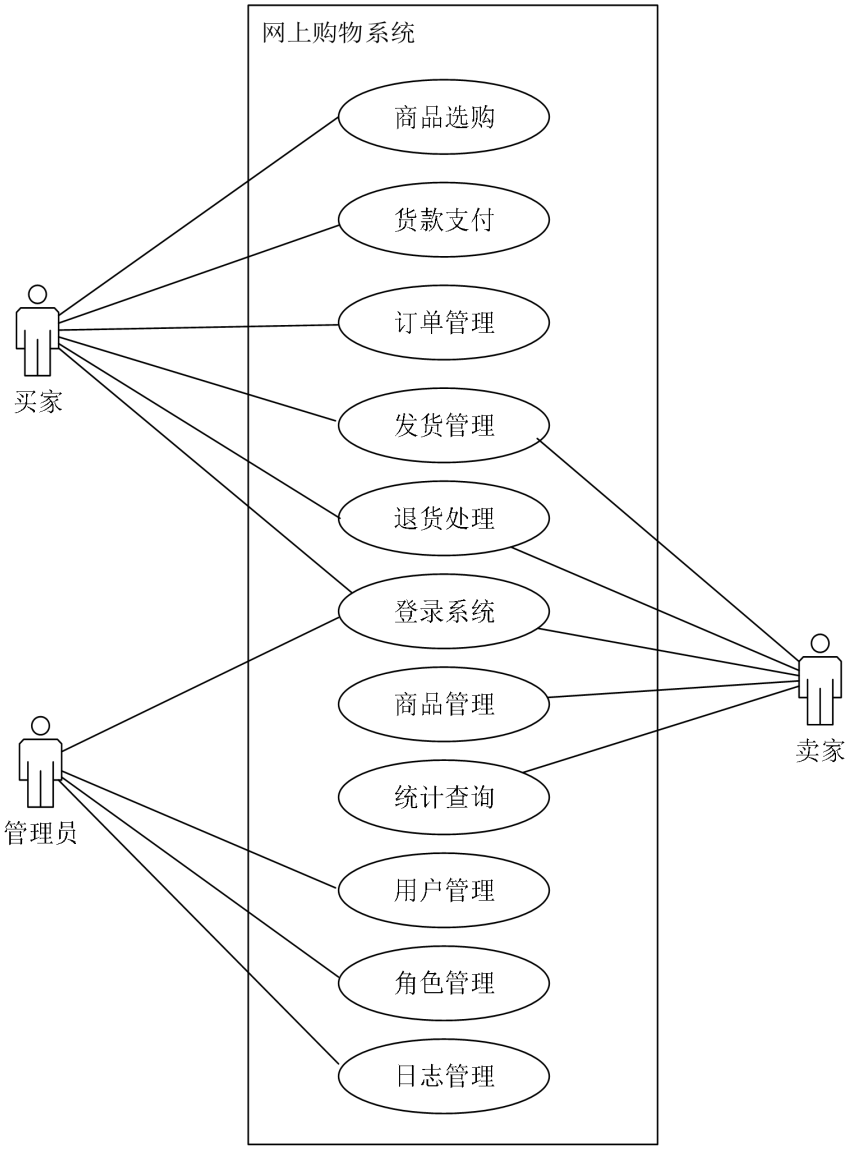


图 2-1 系统总体用例图

<<然后，对各个子用例进行分析，如果是集体项目，则需要进行文字说明，并且只对所负责的部分进行分析。例如：>>

在本网上购物系统的开发中，本人负责商品选购、货款支付、订单管理、发货管理和退货处理等 5 个功能（以下称为**在线购物子系统**）的分析、设计与实现。下面对相关功能进行分析。

2.1.3 商品选购分析

<<1、给出用例图及用例规约（即用例描述）。

注意事项：用例规约的事件流旨在描述参与者与系统的交互过程，不涉及界面细节。>>

商品选购用例中包括检索商品、查看商品详情、将商品放入购物车、将商品

移出购物车和设置购买数量等 5 个子用例，其用例图如图 2-2 所示。

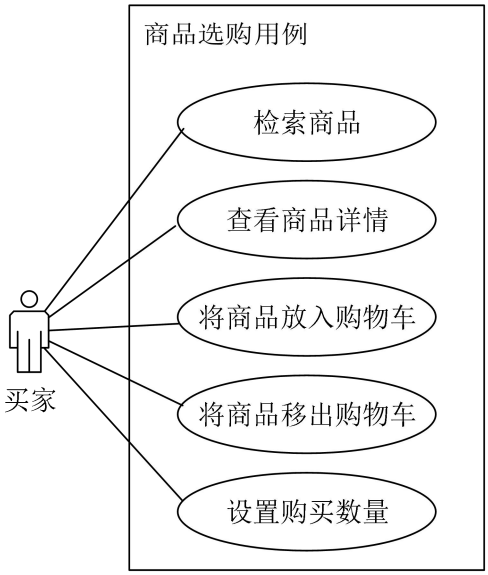


图 2-2 商品选购用例图

下面以检索商品子用例、查看商品详情子用例进行分析，它们的用例描述分别如表 2-2、2-3 所示。

表 2-2 检索商品用例描述

描述项	**说明**
用例名称	检索商品
用例描述	描述了买家使用本系统进行检索商品的过程
参与者表	客户（买家）
前置条件	用户已登录系统
后置条件	系统显示所检索的商品列表
基本操作流	1.用户提供关键字； 2.查看关键字相关商品列表。 ....
可选操作流	无

表 2-3 查看商品详情用例描述

描述项	**说明**
用例名称	查看商品详情
用例描述	描述了买家使用本系统查看商品详情的过程

参与者表	客户（买家）
前置条件	系统已提供商品列表
后置条件	系统显示所选中的商品详情信息
基本操作流	1.用户选中待查看的商品记录； 2.查看相关商品的详情信息。
可选操作流	无

<<以下其它子用例分析按照以上内容展开>>

2.1.4 货款支付分析

2.1.5 订单管理分析

2.1.6 发货管理分析

2.1.7 退货处理分析

2.2 数据需求

<此处应根据 2.1 节的用例图，相对应的在本小节给出类图的概念设计，包含界面类、控制类和实体类，不展开类的属性和方法，采用类图的简化,例如>

经过分析后得知系统中所需的类和各类之间的关系，利用概念类图进行描述。图中后台管理界面类提供对系统角色、系统用户、用户级别、系统日志、商品类别的维护工作。购物界面类提供对商品信息、购物车、订单、支付记录、发货记录、退款申请的各类操作，如图 2-3 所示。

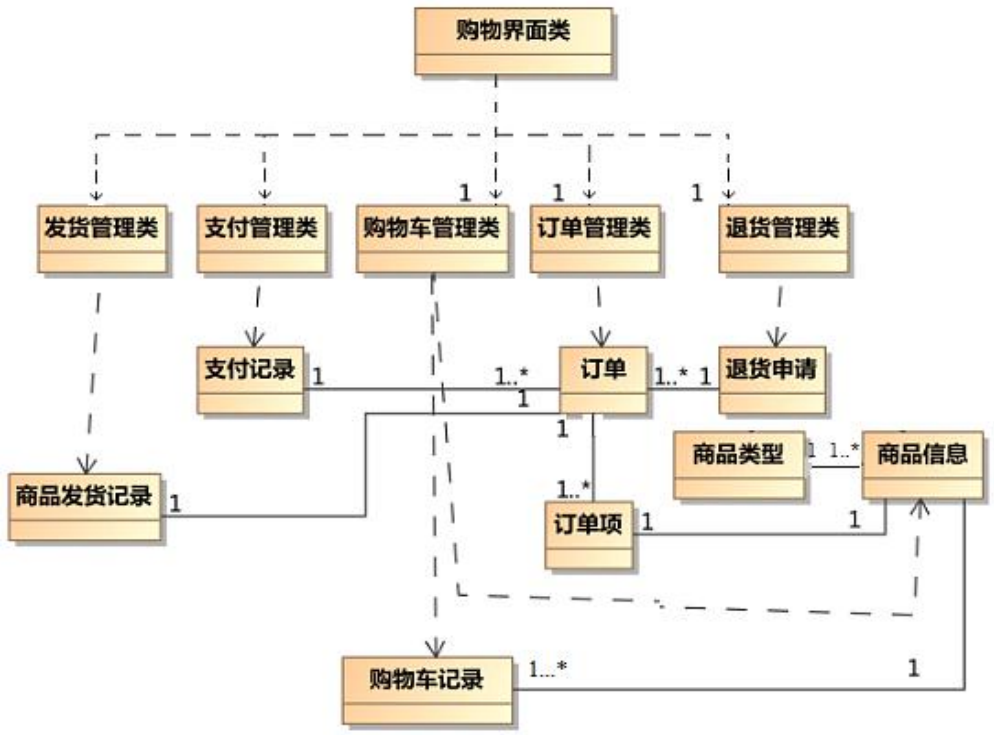


图 2-3 系统概念类图

<<根据概念类图中实体类的内容，给出 ER 图或类图。ER 图，就一个图，实体的属性不需要画出来（用关系模式描述）。然后通过文字描述实体之间的关系。利用文字描述实体及其关系，再给出 ER 图。>>

根据以上概念类图的内容，可知网上购物系统包括商品类型、商品信息、购物车、买家（即：用户）、订单、订单项、退货申请和发货记录等实体。其中，商品类型和商品信息的对应关系为一对多，购物车记录和商品信息为一对一的关系。买家和购物车为一对多关系；一次购物车记录对应一个订单，二者为一对一关系；一个订单具有多个订单项，二者关系为一对多关系。订单和退货申请实体为多对多关系。一个订单可能有多次发货记录，二者为一对多关系。在线购物子系统的 ER 图如图 2-4 所示。

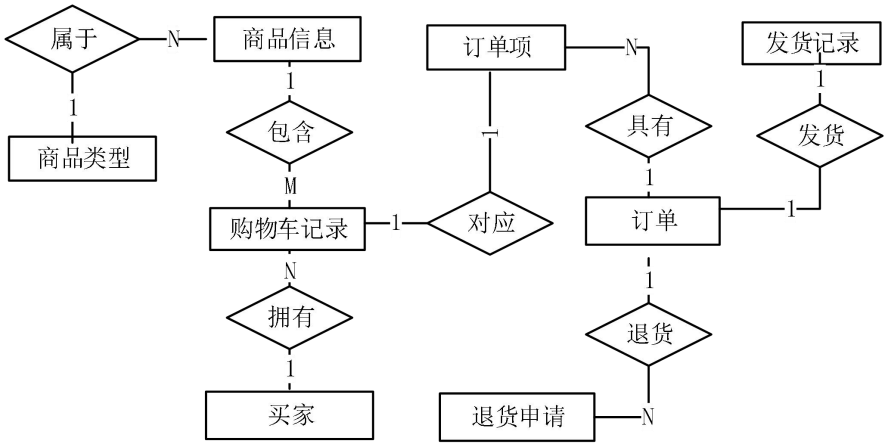


图 2-4 在线购物子系统 E-R 图

2.3 系统非功能性需求

<<性能、兼容性、安全性、可靠性等方面的需求，根据论文后继章节的内容选择。>>

第 3 章 系统设计

3.1 功能模块设计

<<主要阐述有关系统设计之功能架构设计，给出层次功能框图（或功能包图）定义软件系统各主要部件及成分之间的关系，以下为给出包图的范本>>
<<可以通过 UML 的包图来表示，系统功能一般对应需求分析中的用例，例如：>>

根据需求阶段的功能分析可知，在线购物子系统可以划分为商品选购、货款支付、订单管理、发货管理和退货处理等 5 个模块，其功能包图如图 3-1 所示。

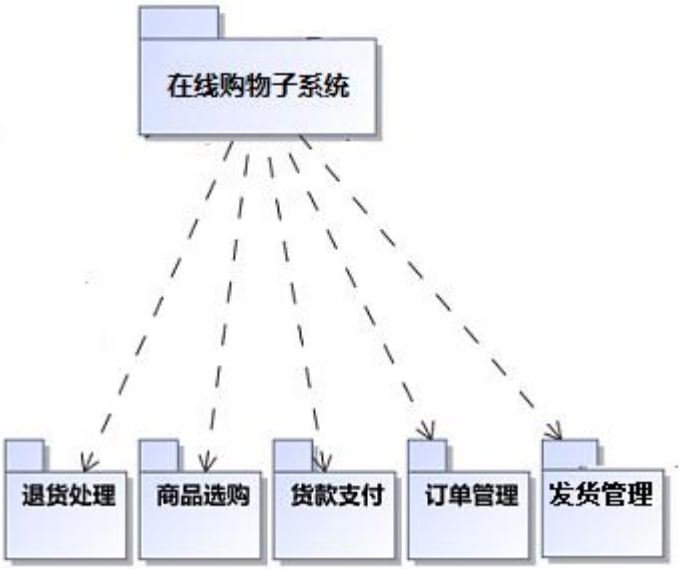


图 3-1 在线购物子系统功能包图

本节中，将对在线购物子系统功能包图中的各个功能模块进行具体设计，具体如下。

3.1.1 商品选购设计

（1）功能结构设计

<<使用 UML 包图给出子功能架构，例如>>

“商品选购”功能主要目的是为买家提供购买卖家提供的商品的功能；此外，该功能还负责将用户选中的商品放入购物车，对购物车进行管理。因此，商品选购模块主要涉及商品信息的查询获取以及购物车管理两个子模块。其中商品信息获取模块主要分为检索商品和查看商品详细信息两个组成部分。购物车管

理又分为将商品放入购物车和将商品从购物车移除两个部分，其包图如图 3-2 所示。

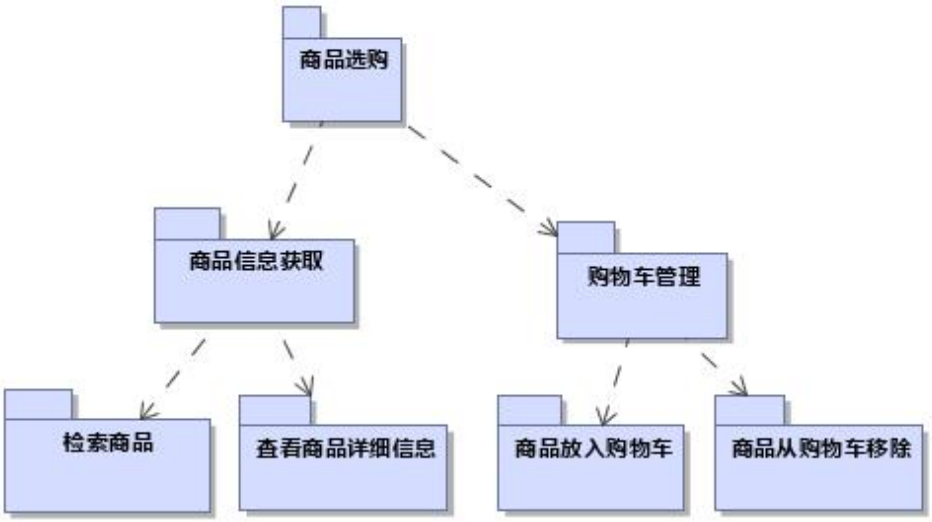


图 3-2 商品选购包图

（2）类图设计

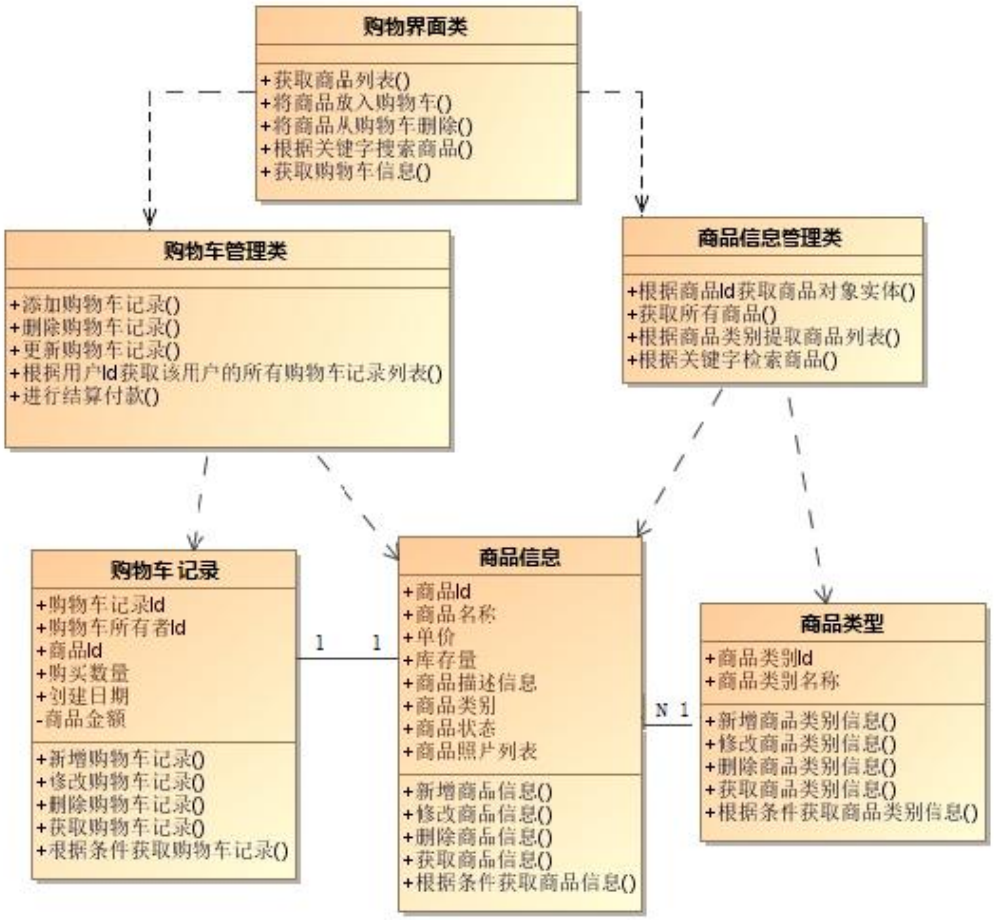


图 3-3 商品选购类图

如上图 3-3 所示是商品选购类图中，一共有 6 个类，其中：购物界面类主

要负责响应页面发出的商品信息获取请求，负责购物车的维护，它的执行依赖于购物车管理类和商品信息管理类；购物车管理类是购物车管理主功能类；商品信息管理类主要负责根据各类格式或条件获取商品的信息以及对商品类型的管理，依赖于商品信息类和购物车类两个实体类；商品信息类是商品信息实体类，具备单元信息维护功能；商品类别类是商品类别实体类，具备单元信息维护功能；购物车类是购物车实体类，具备单元信息维护功能。商品信息类与购物车类为多对多的关联关系；商品类型和商品信息为一对多的对应关系。

（3）顺序图设计

<<使用顺序图或协作图描述该功能（用例）所参与的对象，以及这些对象相互之间的动态消息联系；使用状态图用来描述一个特定的对象所有可能的状态,以及由于各种事件的发生而引起的状态之间的转移和变化。示例：>>

商品信息获取功能的顺序图如图 3-4 所示。

图 3-4 商品信息获取顺序图

购物车管理功能的顺序图如图 3-5 所示。

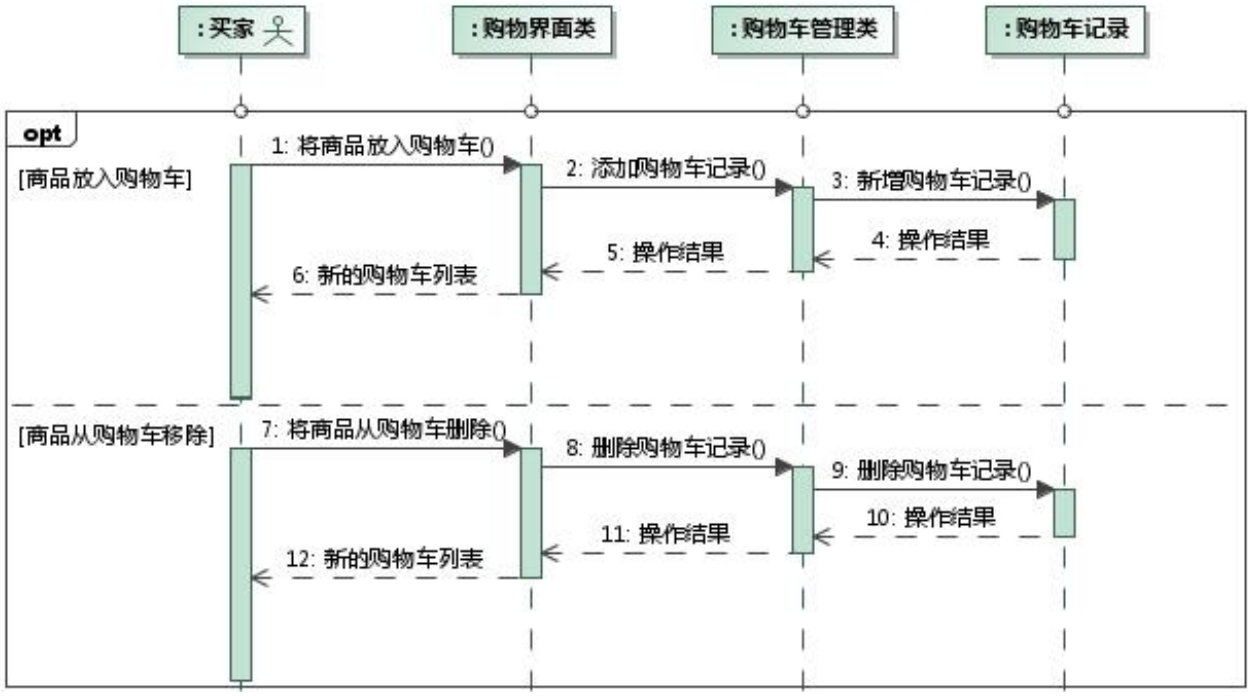


图 3-5 购物车管理顺序图

（4）核心处理流程设计

<使用活动图描述上面顺序图中所述核心操作的处理流程。在论文中，每个复杂的流程步骤都应该用一个活动图来进行描述，不能省略。>

针对上述顺序图，图 3-4 中“根据关键字搜索商品”和图 3-5 中“添加购物车记录”这两个方法还可以进一步细化，如图 3-6 和图 3-7 所示。

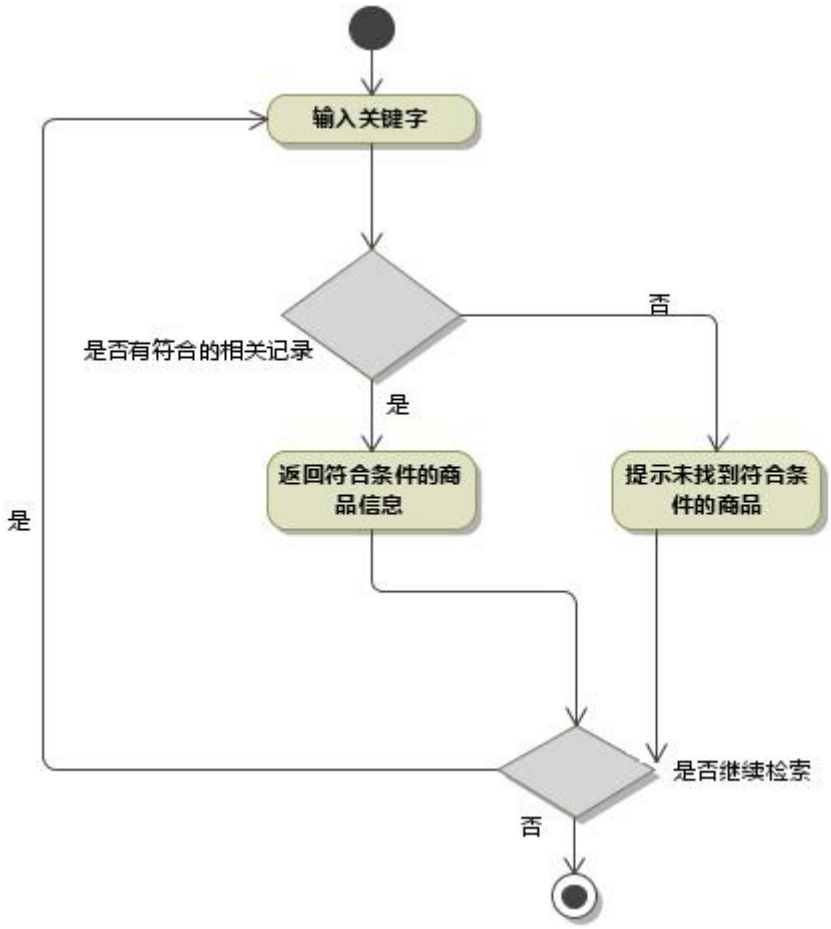


图 3-6 根据关键字搜索商品活动图

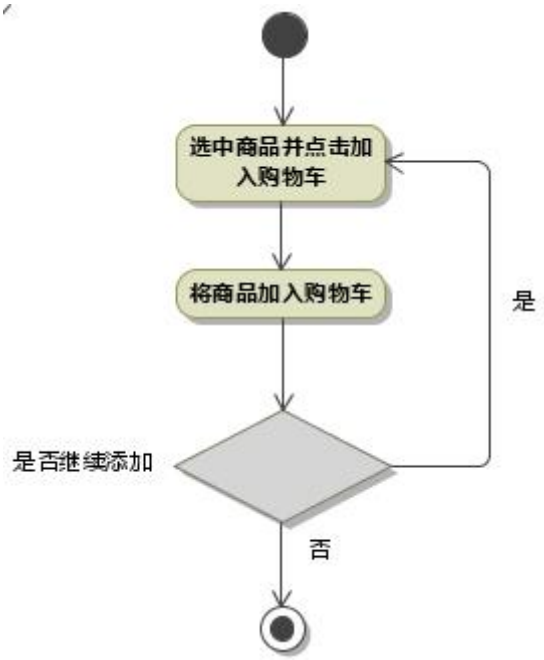


图 3-7 添加购物车记录活动图

3.1.2 货款支付设计

3.1.3 订单管理设计

3.1.4 发货管理设计

3.1.5 退货处理设计

3.2 数据库设计

<<根据实体类，给出软件系统主要表的设计，给出数据表（采用 Word 中的表格，不要截数据库管理工具中的界面图），例如>>

购物系统的主要表的结构如下所示。

（1）商品类别

保存商品类别的基本信息。如表 3-1 所示。

表 3-1 商品类别表

字段名	数据类型	长度	主键/外键	描述
CategoryId	Guid		**主键**	商品类别 ID
CategoryName	nvarchar(20)	20	否	商品类别名称

（2）商品信息

保存商品的基本信息。如表 3-2 所示。

表 3-2 商品信息表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	商品 ID
Name	nvarchar(150)	150	否	商品名称
Price	int		否	单价
Amount	int		否	库存量
Description	nvarchar(255)	255	否	描述信息
Photo	Binary		否	商品照片
Category	nvarchar(50)	50	否	商品类别
Status	nvarchar(10)	10	否	商品状态

（3）购物车记录

保存购物车记录的基本信息。如表 3-3 所示。

表 3-3 购物车记录表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	购物车记录 ID
OwnerId	Guid		**外键**	购物车所有者 ID
GoodId	Guid		**外键**	商品 ID
Quantity	int		否	购买数量
CreateDate	Datetime		否	创建日期
Money	int		否	商品总额

（4）订单项

保存订单项的基本信息。如表 3-4 所示。

表 3-4 订单项表

字段名	数据类型	长度	主键/外键	描述
OrderDetailId	Guid		**主键**	订单项 ID
OrderId	Guid		**外键**	订单 ID
GoodId	Guid		**外键**	商品 Id
Quantity	int		否	购买数量
Money	int		否	商品总额

（5）订单

保存订单的基本信息。如表 3-5 所示。

表 3-5 订单表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	订单 ID
UserId	Guid		**外键**	客户 ID
SumMoney	int		否	订单总额
OrderDate	Datetime		否	下单日期
PaymentStatues	nvarchar(10)	10	否	支付状态
GoodsDeliverStatus	nvarchar(10)	10	否	发货状态
EndDate	Datetime		否	结单日期
PaymentType	nvarchar(10)	10	否	支付方式

Status	nvarchar(10)	10	否	支付记录状态
PayTime	Datetime		否	支付时间
SumMoney	int		否	支付总额
UserId	Guid		否	支付人 Id

（6）用户信息

保存用户的基本信息。如表 3-6 所示。

表 3-6 用户信息表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	用户 ID
Name	nvarchar(10)	10	否	用户姓名
Password	nvarchar(20)	20	否	密码
Mail	nvarchar(50)	50	否	电子邮件
MobilePhone	nvarchar(50)	50	否	手机号码
Tel	nvarchar(50)	50	否	座机号码
Address	nvarchar(255)	255	否	发货地址
Sex	Byte		否	性别

（7）退货申请表

保存退货申请的基本信息。

表 3-7 退货申请表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	退货申请 ID
OrderId	Guid		**外键**	订单 ID
DealDate	Datetime		否	处理日期
ApplicantId	Guid		否	申请人 ID
Description	nvarchar(255)	255	否	问题描述
FeedbackResult	nvarchar(255)	255	否	反馈结果
Status	nvarchar(10)	10	否	退货申请状态
SubmitDate	Datetime		否	退货申请提交日期

（8）商品发货记录

保存发货记录的基本信息。

表 3-8 商品发货记录表

字段名	数据类型	长度	主键/外键	描述
Id	Guid		**主键**	发货记录 ID
OrderId	Guid		**外键**	订单 ID
DeliverDate	Datetime		否	发货日期
Status	nvarvhar(10)	10	否	发货记录状态
TransportCompany	nvarvhar(50)	50	否	货运公司名称
Route	nvarvhar(255)	255	否	物流信息
WayBillNumber	nvarvhar(20)	20	否	运单号

给出数据表之间的关系图（可用 Visio 画），主要以直观的方式体现表之间的关系。

第 4 章 模块实现

<<围绕**项目开发中所使用的技术**来**有选择性**描述相关模块的实现。提炼出一些便于介绍的内容展开本章的编写。如可以通过某个模块介绍分页，可以通过某个模块介绍 JDBC 的应用，通过另一个模块介绍 ajax 等，这样写起来比较清晰。

可以介绍前端界面的实现、JDBC、连接池、分页、ajax、Struts、文件上传、统计报表、访问控制等。>>

4.1 系统软件架构

4.1.1 系统开发框架

<<使用的工具、技术或框架注意标注引用文献出处！>>

本系统基于 SSM 的 JavaWeb 开发框架如图 4-1 所示。

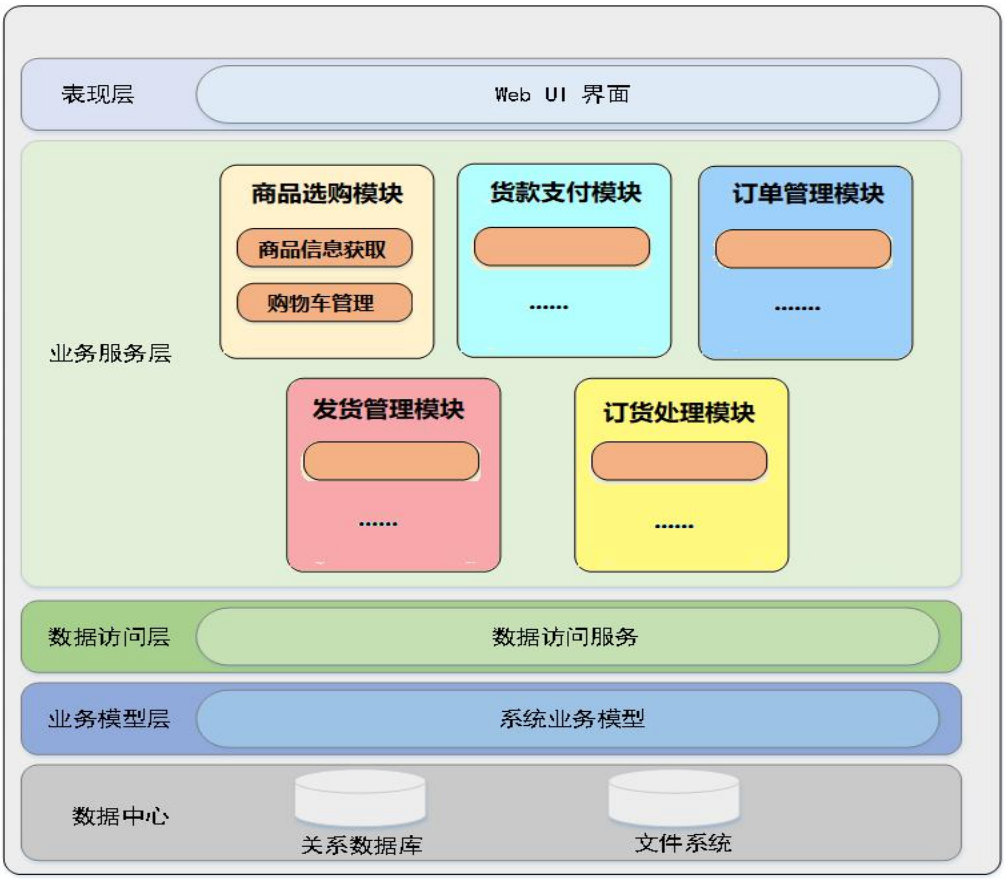


图 4-1 系统架构图

如图 4-1 所示的系统架构中，具体层次职责如下：

（1）表现层（View）：通过 html 页面实现；

（2）业务模型层（Model）：对应的是数据库中每个表格的实体；

（3）数据访问层（Dao）：用来访问数据库，对数据库进行操作，实现数据的持久化；

（4）业务服务层（Service）；引用对应的 Dao 数据库操作，封装表示层的请求，每个请求封装成 Service 类的方法；且该层提供相应的接口供 Rest(Controller)层调用；

（5）数据中心：指数据库和文件系统。

在线购物子系统实现的工程文件结构如图 4.2 所示。

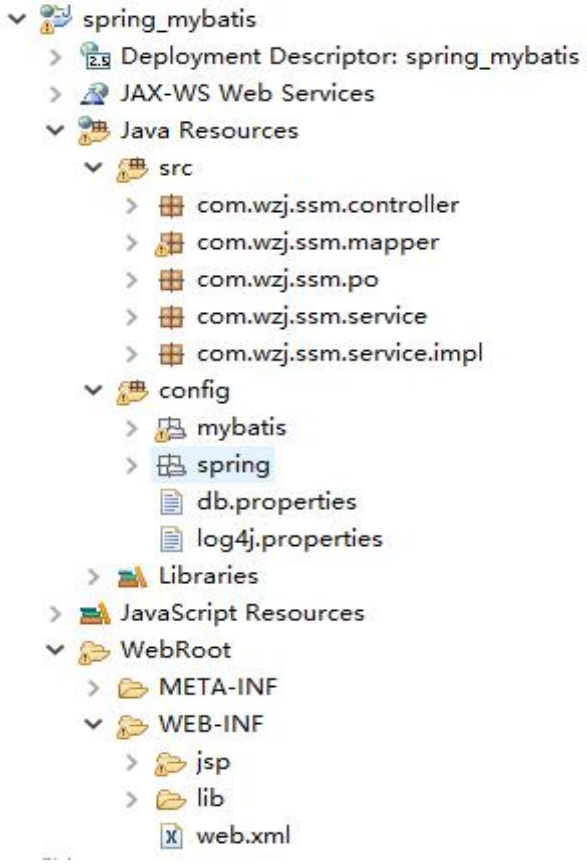


图 4-2 工程文件结构图

4.2.2 SSM 框架全局配置

① 配置 db.properties。该文件包括连接数据库需要的配置信息，内容如下。

```
jdbc.driver=org.gjt.mm.mysql.Driver
jdbc.url=jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8&useSSL=true
jdbc.username=xxx
jdbc.password=xxxxx
```

② Mybaits 与 Spring 的整合。包括 mybaits 的全局配置、spring 与 mybatis 整

合配置和 Spring 事务控制 3 个部分。

第 1 步，配置 mybaits 的全局配置文件 sqlMapConfig.xml：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
 "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- setting 全局配置参数，需要时再配置 -->
    <!-- 别名定义 -->
    <typeAliases>
        <!-- 批量扫描别名 -->
        <package name="com.wzj.ssm.po" />
    </typeAliases>
    <!-- 使用 mybatis 和 spring 整合包进行 mapper 扫描，这里不需要配置
    但需要遵循规范：mapper.java 和 mapper.xml 必须同名且在同一目录下
    <mappers>
        <mapper resource="com/wzj/ssm/mapper/ItemsMapper.xml" />
        <mapper resource="com/wzj/ssm/mapper/ItemsMapperCustom.xml" />
    </mappers> -->
</configuration>
```

第 2 步，spring 与 mybatis 整合配置文件 applicationContext-dao.xml，该文件中包括：数据源、sqlSessionFactory 和 mapper 扫描器三个方面的内容，核心代码如下：

```
    <!-- 1,配置数据源 -->
    <!-- 先加载 db.properties 文件 -->
    <context:property-placeholder location="classpath:db.properties" />
    <!-- 后配置属性 -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close" >
        <property name="driverClassName" value="${jdbc.driver}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
        <!--  maxActive:连接池支持的最大连接数，0 表示没有限制 -->
        <property name="maxActive" value="30" />
        <!-- maxIdle:连接池允许的最大空闲连接数，而不会被清除，0 表示没有限制 -->
        <property name="maxIdle" value="5" />
    </bean>
    <!-- 2,配置 sqlSessionFactory -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean" >
        <!-- 数据库连接池 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 加载 mybatis 全局配置文件 -->
        <property name="configLocation" value="classpath:mybatis/sqlMapConfig.xml" />
```

```
    </bean>
    <!-- 3,配置 mapper 扫描器 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer" >
        <!-- 扫描包路径，需要扫描多个包时，用半角逗号隔开 -->
        <property name="basePackage" value="com.wzj.ssm.mapper" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>
</beans>
```

第 3 步，配置 Spring 事务控制（applicationContext-transaction.xml），核心代码如下：

```
    <!-- 1,配置事务管理器 -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.
            DataSourceTransactionManager" >
        <!-- 加载在 applicationContext-dao.xml 中配置的 dataSource -->
        <property name="dataSource" ref="dataSource" />
    </bean>
    <!-- 2,通知 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager" >
        <tx:attributes>
            <!-- 传播行为 -->
            <tx:method name="save*" propagation="REQUIRED" />
            <tx:method name="delete*" propagation="REQUIRED" />
            <tx:method name="update*" propagation="REQUIRED" />
            <tx:method name="insert*" propagation="REQUIRED" />
            <tx:method name="find*" propagation="SUPPORTS" read-only="true" />
            <tx:method name="get*" propagation="SUPPORTS" read-only="true" />
            <tx:method name="select*" propagation="SUPPORTS" read-only="true" />
        </tx:attributes>
    </tx:advice>
```

③配置 springmvc（表现层）。springmvc 是 spring 的一个组件，只需配置好 springmvc 的属性，如处理器映射器、处理器适配器、视图解析器即可。具体如下：

第 1 步：配置 springmvc.xml 文件，核心配置代码如下：

```
    <!-- 扫描控制器 -->
    <context:component-scan base-package="com.wzj.ssm.controller"/>
    <!-- 使用此标签来加载注解的处理器映射器和处理器适配器 -->
    <mvc:annotation-driven/>
    <!-- 视图解析器 -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!-- jsp 文件路径前缀 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- jsp 文件路径后缀 -->
        <property name="suffix" value=".jsp" />
```

```
    </bean>
```

第 2 步：在 web.xml 中配置前端控制器，核心代码如下：

```
<!-- 前端控制器 -->
  <servlet>
    <servlet-name>spring_mybatis</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/springmvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>spring_mybatis</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>
```

4.2 商品选购实现

实现用户进行获取购物车信息，购物车管理。描述该模块的功能。

4.2.1 设计思想

商品选购模块，首先在 Model 层封装用户实体，在 Dao 层封装了用户实体数据的增删改查接口，在 Service 层封装表示层的请求（即用户请求）的接口，在 Service 层中调用相应的 Dao 层接口，用 Service 的具体实现类去实现这些接口。Controller 层引用相应的 Service 层。

4.2.2 操作界面

描述界面的组成（如何布局界面使界面更友好）并附图。

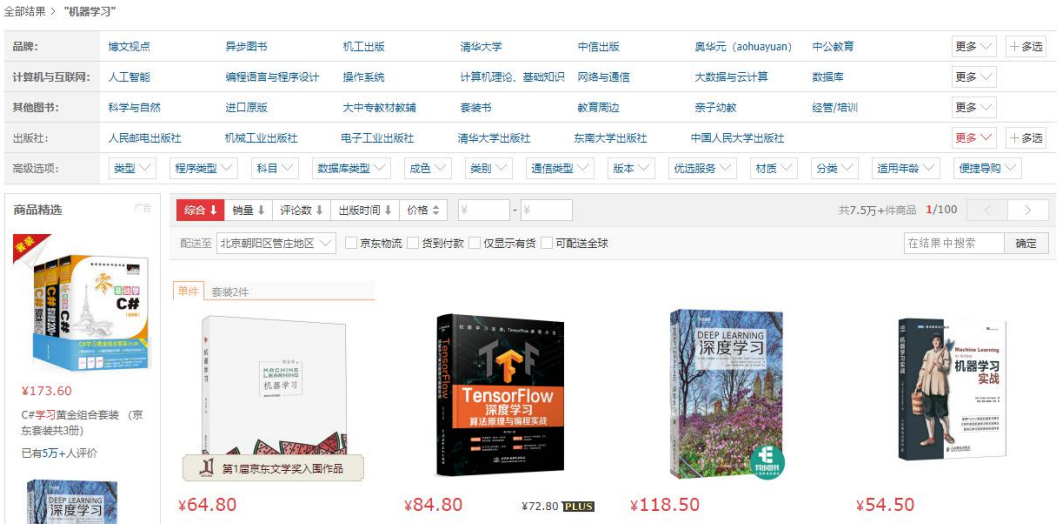


图 4-3 检索商品效果界面

图 4-4 查看商品详细信息界面

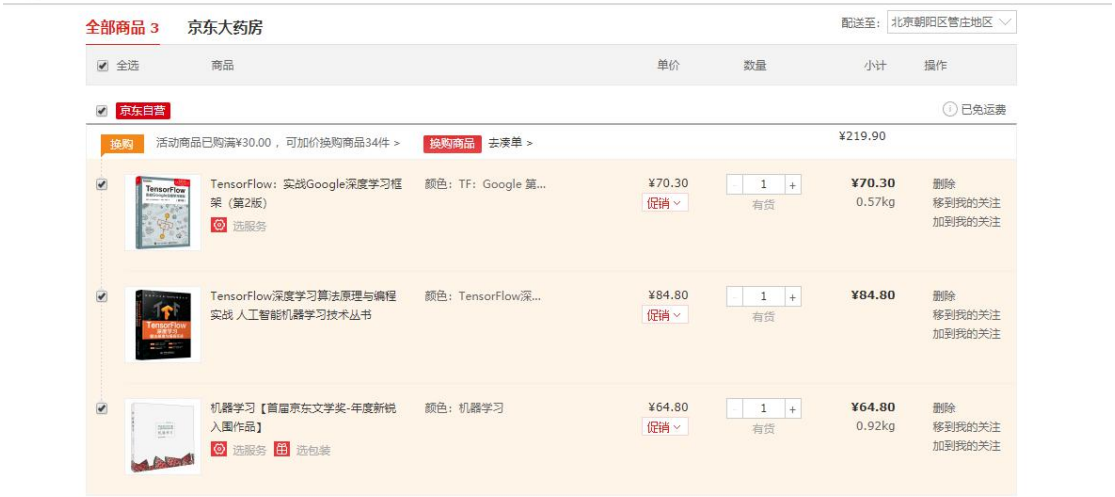


图 4-5 添加购物车效果界面

图 4-6 移除购物车效果界面

4.2.3 核心代码

<<围绕设计思想中提到的问题介绍你是如何应用相应代码（含配置）完成相应功能（如何进行商品选购）。

对代码要进行文字说明，特别是重要的语句、方法（如）要进行介绍，让读者能看懂这段代码是如何实现这个功能。一个功能实现的代码不能超过一页，次要代码采用省略号。>>

以检索商品为例，介绍相关程序的核心代码。

①View 层（表示层，以 Item.jsp 为例）代码，核心代码如下：

```
<form action="" method="get" >
    <table>
        <tr>
            <td align="center" >商品 id</td>
            <td align="center">商品名称</td>
            <td align="center">价格</td>
            <td align="center">数量</td>
            <td align="center">商品描述</td>
            <td align="center">上市时间</td>
        </tr>
        <%
        List<ItemsCustom> itemsList = (List<ItemsCustom>)request. getAttribute ("itemsList");
        for(ItemsCustom ic:itemsList){
        %>
        <tr>
            <td align="center"><%=ic.getId() %></td>
            <td align="center"><%=ic.getItemname() %></td>
```

```
                <td align="center"><%=ic.getPrice() %></td>
                <td align="center"><%=ic.getAmount() %></td>
                <td align="center"><%=ic.getDescription() %></td>
                <td align="center"><%=ic.getCreate_time() %></td>
            <%
            }
            %>
        </table>
</form>
```

②Controller 层（控制层）主要代码：

ItemsController.java

```
//商品综合查询信息列表控制器
@Controller
public class ItemsController {
    //使用注解注入 service
    @Autowired
    private ItemsService itemsService;
    @RequestMapping("/queryItems")
    public ModelAndView queryItems() throws Exception{
        //添加查询条件
        ItemsQueryVo itemsQueryVo = new ItemsQueryVo();
        ItemsCustom itemsCustom = new ItemsCustom();
        //价格 8000
        itemsCustom.setPrice(8000);
        //商品名字类似于‘iphone’
//      itemsCustom.setItemname("iphone");
        itemsQueryVo.setitemsCustom(itemsCustom);
        //调用 service 接口实现类
        List<ItemsCustom> itemsList = itemsService.findItemsList(itemsQueryVo);
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("itemsList", itemsList);
        //设置对应的 jsp 视图
        modelAndView.setViewName("items");
        return modelAndView;
```

```
}
```

③Service 层（服务层）主要代码：

第 1 步，定义 Service 接口 ItemsService.java；

第 2 步，编写 Service 实现类 ItemsServiceImpl.java，相关代码如下：

```
//使用注解方式注入 itemsMapperCustom
@Autowired
private ItemsMapperCustom itemsMapperCustom;
@Override
public List<ItemsCustom> findItemsList(ItemsQueryVo itemsQueryVo) throws
Exception {
    //通过 itemsMapperCustom 查询数据库
    return itemsMapperCustom.findItemsList(itemsQueryVo);
}
```

第 3 步，在 spring 容器中配置 service（applicationContext-service.xml），内容如下：

```
<bean  id="itemsService"  class="com.wzj.ssm.service.impl.ItemsServiceImpl"
/>
```

第 4 步，在事务控制（applicationContext-transaction.xml）添加配置：

```
<aop:config>
    <aop:advisor advice-ref="txAdvice" pointcut="execution(*
        com.wzj.ssm.service.impl.*.*(..))" />
</aop:config>
```

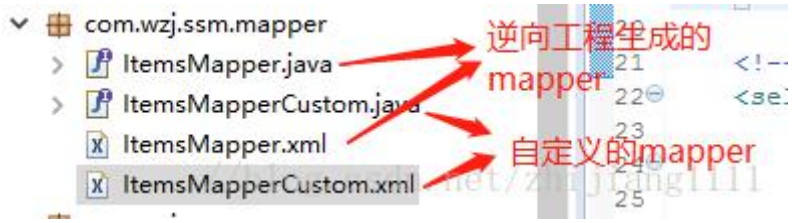
④DAO 层主要代码：

在 4.2.2 节中已经对整个项目的 Mybaits 做了全局配置，在此主要利用逆向工程生成 po 类和 mapper，然后自定义 mapper 进行综合查询。

第 1 步，逆向工程实现 po 类和 mapper.java、mapper.xml。一般，所生成的 po 类尽量不要去修改，如果需要添加一些数据库不需要的属性，可以创建对应 po 类的扩展类。而所生成的 mapper，只有单表的增删改查，由于我们要综合查询（多表查询），需要自定义。

po 类的结构：

mapper 配置的结构：



自定义的 **ItemsMapperCustom.xml** 包含商品信息的综合查询，核心代码如下：

```
<mapper namespace="com.wzj.ssm.mapper.ItemsMapperCustom" >
    <!-- 商品综合查询信息列表条件的 sql 片段 -->
    <sql id="query_items_where" >
        <if test="itemsCustom != null" >
            <if test="itemsCustom.price != null and itemsCustom.price != ''">
                AND items.price > #{itemsCustom.price}
            </if>
            <if test="itemsCustom.itemname != null and itemsCustom.itemname !=
                ''">
                AND items.itemname LIKE '%${itemsCustom.itemname}%'
            </if>
        </if>
    </sql>
    <!-- 商品综合查询信息列表 -->
    <select id="findItemsList" parameterType="itemsQueryVo"
resultType="itemsCustom">
        SELECT * FROM items
        <where>
            <include refid="query_items_where" />
        </where>
    </select>
</mapper>
```

自定义的商品综合查询信息列表的 mapper 接口：**ItemsMapperCustom .java**，核心代码如下：

```
public interface ItemsMapperCustom {
    public List<ItemsCustom> findItemsList(ItemsQueryVo itemsQueryVo) throws
    Exception;
}
```

4.2.4 测试

本模块的功能测试采用黑盒测试，设计测试用例。测试用例（Test Case）是对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略。测试用例设计和执行是测试工作的核心，好的测试用例能提高测试效率

和质量。

测试通过标准包括：①测试结果与测试用例中预期结果一致；②数据安全、正确。在测试过程中发现的软件缺陷，将缺陷提交到缺陷管理系统中，进行跟踪维护，直到该缺陷被解决时，才能关闭该缺陷。

在此以检索商品功能为例，设计相关测试用例，如表 4-1 所示。

表 4-1 商品选购模块之检索商品测试用例

功能模块名	检索商品		
用例编号	Cs—1		
测试优先级	高		
前置条件	可输入检索条件		
测试方法	等价类划分（对应还有“错误猜测法”、“边界值分析”等）		
测试目的	验证检索商品信息的结果		
用例编号	步骤	参数值	期望结果
Cs—3.1	手动输入想查找的商品名称，出现下拉条件列表框。点击框中某条记录，或点击框中右下方 关闭 按钮；	检索条件=“软件工程”	1. 搜索框下方弹出与“软件工程”相关的下拉条件列表框； 2. 单击条件列表框中第一条记录“软件工程导论”，页面会显示所有“软件工程导论”的商品信息； 3. 当点击下拉搜索框右下方 关闭 按钮，下拉框会关闭； 满足上述 3 个条件测试通过；
Cs—3.2	检索条件为空，直接点击 搜索 按钮	检索条件=“”	1. 页面显示所有商品信息； 满足上述两个条件测试通过
Cs—3.3	手动输入待查找的检索条件，然后点击 搜索 按钮	检索条件=“软件工程”	1. 页面上部出现“软件工程”相应的条件复选框； 2. 页面主体部分显示所有“软件工程”的商品信息并分页显示； 满足上述两个条件测试通过
Cs—3.4	手动输入待查找的检索条件，然后点击 搜索 按钮	检索条件=“我不知道我想要什么啊 111111111”	1. 页面上提示“抱歉，没有找到与“我不知道我想要什么啊 111111111”相关的商品”信息； 2. 提示信息下方显示“热销商品”信息； 满足上述两个条件测试通过

4.3 货款支付实现

4.3.1 设计思想

4.3.2 操作界面

4.3.3 核心代码

4.3.4 测试

4.4 订单管理实现

4.5 发货管理实现

4.6 退货管理实现

第 5 章 总结和展望

5.1 总结

对整个毕业设计工作的总结。历时多长时间、如何完成了毕业设计；采用什么技术完成了什么系统，系统具备什么功能，使用该系统有什么好处，系统还存在什么需要改进的地方；个人有什么体会，取得了什么进步

5.2 展望

针对不足，或针对前景（新技术、新应用等），展望未来还可以开展什么工作。

参考文献

<<参考文献 10 篇以上，其中 3 篇以上最新期刊上的学术论文，其中 1 篇以上外文参考文献。按论文中出现先后顺序引用>>

[1] 作者甲,作者乙.论文的标题 [J].出版地（如北京）：XXX 出版社，年，卷（期）:起始页-结束页.

[2] 作者甲,作者乙.网上资源的标题 [EB/OL].网址.年-月-日.

[3] 作者.学位论文的标题 [D].出版地：XXX 出版社，年.

[4] 作者甲,作者乙.书籍的标题 [M].出版地：XXX 出版社，年.

[5] ... ...

[6] .

[7]

[8]

[9]

[10]

[11]

[12]

[13]