



**本科毕业设计（论文）**

**基于JAVA的Web购物网站**

**学 院 计算机学院**

**专 业 软件卓越**

**年级班别 2014级（1）班**

**学 号 3114006535**

**学生姓名 王荣俊**

**指导老师 滕少华**

**2018年 5 月**

**基于JAVA的Web购物网站**

**王荣俊**

**计算机学院**

**摘 要**

随着社会的进步，人民生活水平的提高，电子商务的业务也迅速增长。该项目旨在模拟电子商城项目，建立一个通过软件和信息技术来达到在线购物的平台，提高卖家的售卖力，提升买家购物的便捷性，提高卖家产品的竞争力。

该项目的主要角色有两个，分为买家和卖家。对于买家来说，功能模块有注册，登录，修改个人信息，浏览商品，添加商品到购物车，管理收货地址，下单，修改订单状态，评价订单。对于卖家来说，功能模块有注册，登录，修改个人信息，管理商店，管理商品，修改订单状态，查看评论，统计商品销售情况。该项目程序后端的编写语言是使用java语言，数据库使用MySql数据库，前端使用jsp + vue + jquery + bootstrap，系统的架构是javaEE架构，使用Spring + SpringMVC + Hibernate框架。从而使项目具有良好的可伸缩性、灵活性和易维护性。

本文首先介绍了项目的背景和意义，概述项目内容，然后根据市场观察，提出项目的可行性分析和需求分析，在此基础上，进行系统的总体设计，数据库设计和系统结构设计。然后根据设计的内容一步一步实现各个功能。最后对系统进行总结和展望，并感谢所有在此过程中提供帮助的老师和同学。

**关键词：**Web购物网站，电子商城，电商，java语言，SSH框架

Abstract

With the progress of society and the improvement of people's living standard, the business of electronic commerce also grows rapidly. Analog electronic mall project, the project aims to establish a through software and information technology to achieve online shopping platform, improve the seller's sales, improve buyers shopping convenience, improve the competitiveness of the sellers products.

There are two main roles in the project, which are divided into buyers and sellers. For buyers, the function module has registered, login, modify personal information, browse merchandise, add to cart goods, management of receiving address, order, change order status, evaluate the order. For the seller, the function module has registration, login, modify personal information, manage the store, manage the goods, modify the status of the order, check the comments, and make statistics on the sales of the goods. The project procedures after the written language is to use the Java language, database using the MySql database, front end using JSP + vue + jquery + the bootstrap, architecture of the system is javaEE architecture, using Spring + + Hibernate for SpringMVC framework. Thus, the project has good scalability, flexibility and maintainability.

This article first introduces the background and significance of the project, an overview of the content of the project, and then according to the market observation, put forward the feasibility of the project analysis and demand analysis, on this basis, the system overall design, database design and system structure design. Then implement each function step by step according to the content of the design. Finally, summarize and prospect the system, and thank all the teachers and classmates who helped me in this process.

**Keywords**: Web shopping, Java language, SSH framework.

# 绪论

## 项目背景及意义

随着中国互联网和电子商务的飞速发展，社会的进步和消费的增长，各行各业蒸蒸日上，利益矛盾和竞争日益激烈，电商也不例外。除了业务上的竞争，对于顾客的竞争也不能落后，因此如何进一步提高服务质量，提升顾客的满意程度，改善运营管理水平和办事效率是一个企业保持竞争力的关键因素。在顾客就是上帝的服务行业(包括电商)，在时间就是金钱的今天，很多顾客对于在购物过程中将大量的毫无意义的时间耗费在货比三家地赶路，商品信息价格不透明，排队等待付款很不满意，因此，提供一个良好的交易环境已经为竞争的重要手段。

低效的货比三家，商品信息的不透明排队会大大降低买到优惠，合适商品的比率。人们为了买到比较优惠的商品，需要到多家店进行一一比较，既考验记忆力，又把大量的时间精力浪费在寻找店铺的过程中。此外，商品信息的不透明，没有统一的商品属性说明，也使顾客很难买到合适自己的商品。

虽然现在部分商店提供了送货上门的服务，能够电话预约，希望能够方便顾客购买商品，刺激消费，但是这并没有完全发挥它的最大作用。主要是因为存在一部分客户对其质量心存疑虑，怕送上门的与想象中不一致，维权又麻烦。

该项目旨在创建一个网上商城，建立一个通过软件和信息技术来达到公平，规范交易的电商平台，增加交易的透明度，交易时遵守既定流程，使交易更公平，更规范，提升顾客的购买满意度。

## 项目概述

该项目是Web购物网站，是一个买家购物，卖家管理商品的一个平台。

其主要功能是：卖家注册并登录，完善个人信息后，就可以创建多个商店，每个商品都有店铺名，描述信息，介绍图片，以及其下的商品。商店创建成功后就可以发布商品到这个商店中，并且定好商品的标题，类型，价格，描述，属性等基本信息。此时买家就可以浏览卖家发布的商品，查看商品的详细信息，包括商品评价，销量等。不过买家要添加商品到购物车或者对商品下单，就需要先注册登录并且完善个人信息后才能执行。买家下单后可以查看相应的订单信息，了解到订单的状态。

一旦买家下单，该商品所属的卖家就会收到通知，通知处理订单。等卖家发出商品后，需要修改订单的状态，告诉买家已经发货了。等到买家收货，就需要在系统上确认收货，本次交易就算是完成了。之后，买家可以对这一次购买进行评价，可以根据实际情况给好评，中评或者差评。之后其他买家就可以在商品详情页看到该评论。

当然，假如交易有问题，比如买家下单后由于某些原因不想买了，又或者货物运送过程中出现问题，又或者买家收到货后发现质量问题或者其他状况，买家可以申请退款。

# 系统分析

## 2.1可行性分析

### 2.1.1市场可行性

该系统是Web购物网站，这种网站虽然在市面上早就已经出现，但是仍有广大的实体店是没有进驻购物网站。在如此众多数量的实体店中，几乎每一个实体店都会有发展更多销售渠道的需求，所以该系统是非常有市场的，市场可执行性很高。

### 2.1.2技术可行性

程序后端的编写语言是使用 java 语言，数据库使用 MySql 数据库，前端使用 jsp + vue + jquery[5] + bootstrap，系统的架构是 javaEE 架构，使用 Spring + SpringMVC + Hibernate 框架。从而使项目具有良好的可伸缩性、灵活性和易维护性。

Java[1]是一种可以撰写[跨平台](http://baike.baidu.com/view/469855.htm" \t "_blank)应用程序的[面向对象](http://baike.baidu.com/view/125370.htm" \t "_blank)的程序设计语言。[Java](http://baike.baidu.com/subview/29/12654100.htm" \t "_blank) 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于[PC](http://baike.baidu.com/subview/13621/4882082.htm" \t "_blank)、[数据中心](http://baike.baidu.com/view/754520.htm" \t "_blank)、游戏控制台、科学[超级计算机](http://baike.baidu.com/view/296324.htm" \t "_blank)、移动电话和互联网，同时拥有全球最大的开发者专业社群。Java是功能完善的通用程序设计语言，可以用来开发可靠的、要求严格的应用程序。

Spring + SpringMVC + Hibernate[2] 框架系统从职责上分为四层：[表示层](http://baike.baidu.com/view/239615.htm" \t "_blank)、控制层、[业务逻辑层](http://baike.baidu.com/view/1030527.htm" \t "_blank)、[数据持久层](http://baike.baidu.com/view/1862970.htm" \t "_blank)，以帮助开发人员在短期内搭建结构清晰、可复用性好、[维护](http://baike.baidu.com/view/1123731.htm" \t "_blank)方便的[Web应用程序](http://baike.baidu.com/view/138452.htm" \t "_blank)。其中使用 SpringMVC 作为系统的整体基础架构，负责MVC的分离，利用Hibernate框架对[持久层](http://baike.baidu.com/view/198047.htm" \t "_blank)提供支持，Spring做管理，管理 SpringMVC 和 Hibernate。采用 Spring + SpringMVC + Hibernate 框架，不仅实现了视图、控制器与模型的彻底分离，而且还实现了[业务逻辑层](http://baike.baidu.com/view/1030527.htm" \t "_blank)与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。而且由于不同层之间[耦合度](http://baike.baidu.com/view/1599212.htm" \t "_blank)小，有利于团队成员并行工作，大大提高了开发[效率](http://baike.baidu.com/view/47610.htm" \t "_blank)。

MySQL[3]是目前最流行的[关系型数据库管理系统](http://baike.baidu.com/view/1450387.htm" \t "_blank)，在 WEB 应用方面MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。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。

JSP是一种跨平台的动态网页技术,在静态页面中嵌入Java代码片段,再由Web服务器中的JSP引擎来进行编译并执行嵌入的Java代码片段,生成的页面信息返回给客户端。可以实现与用户的交互,利于扩展,基于B/S技术的动态网页。

在B/S的系统中，用户可以通过浏览器向分布在网络上的许多服务器发出请求。B/S结构极大的简化了客户机的工作，客户机上只需安装、配置少量的客户端软件即可，服务器将担负更多的工作，对数据库的访问和应用程序的执行将在服务器上完成。

Hibernate[3]是一个将对象映射到数据库中的轻量且快速的ORM解决方案。它减轻开发人员处理低级别的数据库需求，同时节省开发时间。写SQL和解析查询结果是相当繁琐和耗时的任务。hibernate使开发人员远离这些java对象映射到数据库表（称为ORM，对象/关系映射）的繁琐操作，做到使用一个简单的面向对象的API对java对象进行存储、删除、查询、更新。为使hibernate更好地应用到实际项目情景中，对其进行以下拓展封装和优化：增加指定字段默认值属性、数据库升级封装优化、注解属性优化。

Vue[7] 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

Bootstrap[8] 是一个快速开发前端页面的框架，而 Bootstrap 的目标就是帮助web开发者更轻松的打造出功能丰富并且美观的UI界面，是一个非常成熟的框架。

### 2.1.3操作可行性

本系统会使用相对人性化的界面设计，尽量使使用者舒服的操作简单清晰的界面，尽量使用鼠标对界面进行操作。

## 2.2需求分析

### 2.2.1功能需求概述

#### 2.2.1.1买家

（1）注册：用户在登录页面输入手机号，密码，性别，昵称，姓名，用户身份（买家还是卖家）等个人信息进行注册。

（2）登录：用户输入手机号，密码，验证码进行登录。

（3）修改个人信息：登陆后可以对个人信息如昵称，姓名，密码，头像进行修改。

（4）首页：买家在首页可以看到商品类型列表以及商品列表。可以选择某个商品类型查看该类型的商品，也可以在搜索栏输入关键字查找感兴趣的商品。

（5）搜索商品：买家可以根据关键字，商品类型等查询条件查询出想要的商品。

（6）显示商品详情：用户点击进入某个商品的详情页，可以看到商品的详细信息，包括商品名称，描述，图片，价格，所属的商店，用户评价。

（7）添加商品到购物车：买家在商品详情页输入购买数量，点击添加购物车，就可以把商品放入到购物车中。

（8）下单：在商品详情页选好颜色和数量后，可以结账进行下单，也可以在购物车页面中的所有商品结账下单。

（9）管理收货地址：在下单页面需要填写收货地址。收货地址包括联系人姓名，联系人电话，地址这三个信息，买家可以添加收货地址，选择已有的收货地址。在我的收货地址页面中，可以添加收货地址，修改收货地址，设置默认收货地址。

（10）查看订单：查看订单信息，包括订单对应的商品，购买数量，收货地址，订单状态。

（11）确认收货：收到商品后，需要到对应的订单确认收货。

（12）评价订单：确认收货后，可以对本次购物进行评价。评价分为评价等级和评价内容。评价等级分为好评，中评，差评。评价内容是买家对本次购物感受以及商品情况的一段描述文字。

（13）申请退款：买家可以因为收不到货，不想买，买错了，商品不符合预期而申请退款。

#### 2.2.1.2卖家

（1）注册：用户在登录页面输入手机号，密码，性别，昵称，姓名，用户身份（买家还是卖家）等个人信息进行注册。

（2）登录：用户输入手机号，密码，验证码进行登录。

（3）修改个人信息：登陆后可以对个人信息如昵称，姓名，密码，头像进行修改。

（4）首页：卖家在首页可以看到自己已经创建的所有商店。

（5）添加商店：卖家在首页点击添加商店，可以输入商店名，商店描述，商店头像等信息进行创建。

（6）管理商品：卖家进入某个商店，可以查看该商店的所有商品，添加商品，修改商品信息，删除商品。添加商品时需要输入商品名称，描述，价格，图片集，分类。

（7）管理订单：卖家可以查看自己商品的订单，对未发货的订单进行发货，然后执行确认发货的操作。或者对已经申请退款的订单退款，然后执行确认退款操作。

### 2.2.2 功能基本流程

（1）注册登录

买家和卖家在注册登录的流程是一样的。如果没有帐号，就先注册，需要填入手机号，密码，昵称，姓名，性别等个人信息。注册成功后就可以使用该手机号和密码登录。登录之后可以修改个人信息。

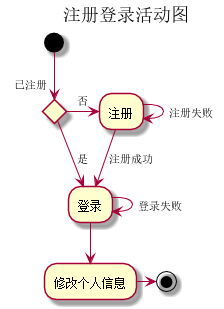


图2.1注册登录活动图

1. 买家购买商品

买家在登录后，可以

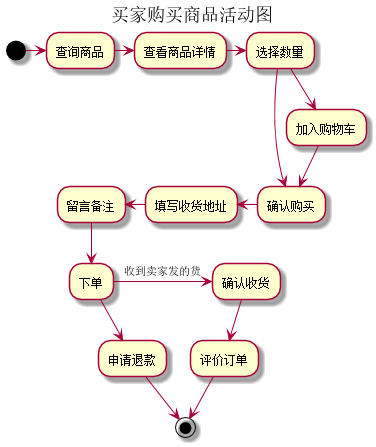


图2.2买家购买商品活动图

1. 卖家管理商品

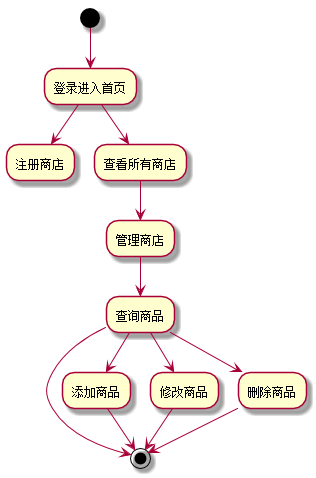


图2.3卖家管理商品活动图

1. 卖家处理订单

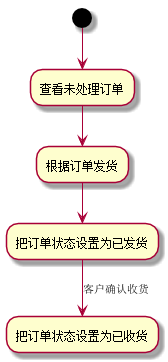


图2.4卖家处理订单活动图

### 2.2.3 功能用例描述

（1）买家用例图

买家可以注册登录，修改个人信息，根据关键字，商品分类等条件查询商品后，可以查看商品详细信息，添加到购物车，下单。下单时需要选择收货地址。收货地址可以添加，也可以选择已有的收货地址。下单后买家可以查询订单信息。在收到商品后，买家需要确认收货，然后可以评价订单。在下单后的整个时期，买家都可以申请退款。

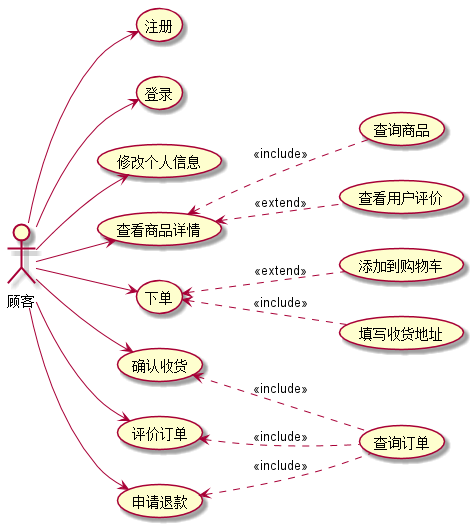


图2.5买家用例图

1. 卖家用例图

卖家注册登录后，可以创建商店，修改商店信息，删除商店（会同时删除商店下的所有商品）。进入商店页面后可以看到该商店下的所有商品。此时可以添加，删除商品。也可以进入商品详情页查看和修改商品信息。

卖家也可以处理相关的订单，比如发货后需要对订单进行确认发货的操作。买家申请退款时，如果已经把钱退回给买家，需要对订单进行确认退款操作。

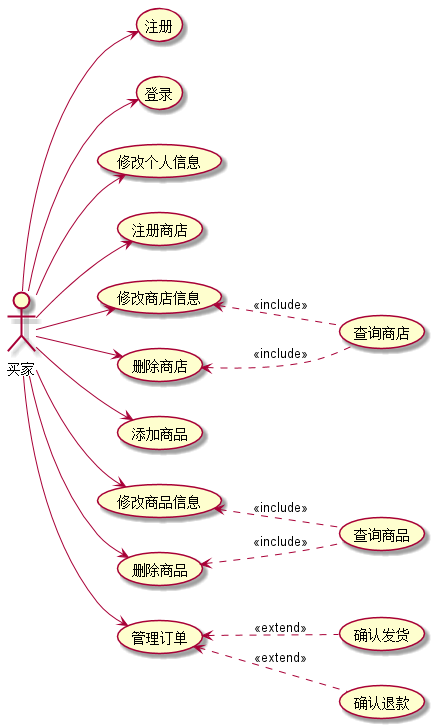


图2.6卖家用例图

### 2.2.4 用例详细文档

（1）注册登录

**表2.1 注册登录用例说明**

|  |
| --- |
| 用例名称：注册登录 |
| 参与者：买家、卖家 |
| 简要说明：买家、卖家执行注册、登录操作 |
| 前置条件：无 |
| 事件流：(1)基本流：   1. 用户到注册页面注册。 2. 用户使用已注册的帐号登录。 3. 登录后修改个人信息。 |
| 异常事件流：   1. 注册失败，一般是帐户已存在或信息填写错误。 2. 登录失败，一般是帐户或密码或验证码错误。 |
| 后置条件：无 |
| 注释：在登录页面中，需要选择身份。如果身份不对，即使帐号密码正确，也会登录失败 |

（2）买家购买商品

**表2.2 买家购买商品用例说明**

|  |
| --- |
| 用例名称：买家购买商品 |
| 参与者：买家 |
| 简要说明：买家购买商品 |
| 前置条件：已经登录 |
| 事件流：(1)基本流：   1. 根据关键字，商品类型等查询条件查询出想要的商品。 2. 查看商品的详细信息，包括商品名称，描述，图片，价格，所属的商店，用户评价。 3. 可选步骤：添加商品到购物车 4. 下单。下单时需要填写收货地址。 5. 查看订单状态。 6. 确认收货 7. 评价订单   (2)备选流：下单后，如果对本次交易不满，可以申请退款 |
| 异常事件流：无 |
| 后置条件：无 |

（3）卖家管理商品和订单

**表2.3 卖家管理商品和订单用例说明**

|  |
| --- |
| 用例名称：卖家管理商品和订单 |
| 参与者：卖家 |
| 简要说明：卖家管理商品和订单 |
| 前置条件：已经登录 |
| 事件流：(1)基本流：   1. 首页查看已经注册的所有商店 2. 添加商店，修改商店信息 3. 对某个商店下的商品进行增删查改 4. 修改自己商品相关订单的状态   (2)备选流：  无 |
| 异常事件流：无 |
| 后置条件：无 |

### 2.2.5 非功能性需求

非功能需求是指依一些条件判断系统运作情形或其特性，而不是针对系统特定行为的需求，我们的软件产品为了满足用户业务的需求就必须具有除功能需求以外的特性，它一般和系统的状态有关而与系统需要提供的功能无关，是功能需求的有力补充。一般软件产品的非功能需求包括系统的可用性、性能、可靠性、可支持性、其他等。

1、可用性

可用性是一个非常宽泛的概念，它泛指那些能让用户顺利使用系统的指标，包括易用性（易操作、易理解）、安全性（权限体系、访问限制）、兼容性（服务器、客户端的兼容度），等等。

易用性：本系统是以一个简单易用的网页展示，基于web浏览器访问，非常容易操作。

安全性：

兼容性：服务器使用tomcat，客户端使用浏览器，即B/S架构，兼容性方面没有什么问题。

2、性能需求。

系统的数据量不会太大，在数据库方面会使用缓存和索引等，性能是可以得到保障的。

3、可靠性

系统可以可靠运行。数据要按时进行备份。

4、可支持性

由于使用Java语言来编写此系统，对于跨平台是毫无问题的；但对于不同的浏览器的兼容问题是需要考虑的一个问题。

# 系统设计

## 3.1总体设计

### 3.1.1系统总体结构图

Web购物网站从用户身份区分，可以分为买方和卖方。本网站的主体是商品和订单。买方的主要角色是买家，主要的职责是购买商品。卖方的主要角色是卖家，主要的职责是管理商品。

其中买家可以注册登录，浏览商品，添加商品到购物车，提交订单，确认收货，申请退款，评价订单。卖家可以注册登录，管理商店信息，包括增删查改，也可以处理订单，比如查看订单，确认发货，确认退款。

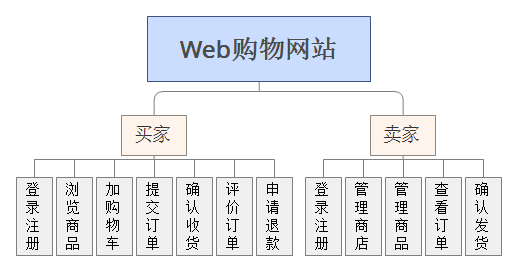


图3.1系统总体结构图

### 3.1.2开发运行环境

系统开发环境：windows

运行平台：windows

数据库：mysql 5.7

Web服务器：tomcat8

系统开发工具: Intellij Idea

系统开发技术: Spring + SpringMVC + Hibernate + Maven + Jsp + Vue + JQuery + Bootstrap

## 3.2数据库设计

### 3.2.1数据库的需求分析

依据项目的需求以及业务逻辑，可以设计如下数据表及其功能：

图片表：保存图片，主要的属性有图片id，图片类型，图片二进制数据。

买家表：保存买家个人信息，主要的属性有用户id，用户名，密码，性别，个人头像。

卖家表：保存卖家个人信息，主要的属性有用户id，用户名，密码，性别，个人头像。

商品分类表：保存商品的分类信息，主要的属性有分类id，分类名。

商品分类关系表：保存商品分类之间的从属关系，主要的属性有祖先分类外键id，子孙分类外键id，祖先与子孙距离。

商店表：保存商店信息，主要的属性有商店id，商店名称，商店描述，商店头像，商店所属的卖家id。

商品表：保存商品信息，主要的属性有商品id，商品名称，商品描述，商品分类，商品价格，商品所属的商店id。

商品图片关系表：保存商品与图片之间的关系，主要的属性有商品id，图片id。

购物车表：保存买家添加商品到购物车的记录，主要的属性有商品id，买家id。

收货地址表：保存买家的收货地址，主要的属性有收货地址id，买家id，收货人名称，收货人电话，收货人地址。

订单表：保存买家下单的信息，主要的属性有订单id，商品id，买家id，创建日期，购买数量，总价格，订单状态，买家下单时的备注留言。

评价表：保存买家对某个订单的评价，主要的属性有评价id，买家id，订单id，评价等级（好，中，差评），评价内容。

退款记录表：保存买家申请退款的记录，主要的属性有退款记录id，订单id，退款理由。

### 3.2.2数据库概念结构设计

1、买家，购物车，收货地址

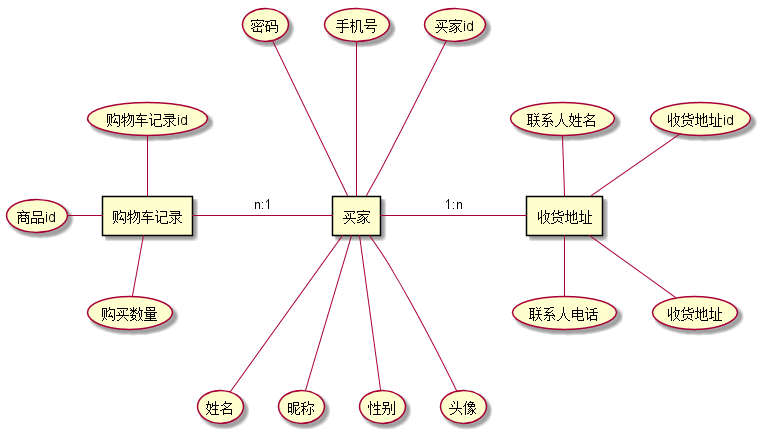


图3.1买家，购物车，收货地址实体E-R图

2、卖家，商店

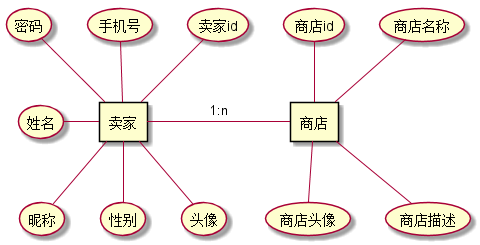


图3.2卖家，商店实体E-R图

1. 商品

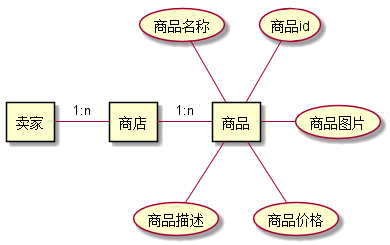


图3.3商品实体E-R图

1. 订单

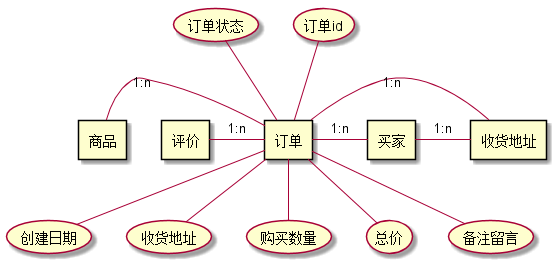


图3.4订单实体E-R图

### 3.2.3数据库逻辑结构设计

根据以上实体属性以及实体与实体之间的关系，我们可以设计各个表的字段值，以及各个表之间的关联，表的逻辑结构。从而建立一个数据库的逻辑结构模型，如图3.7所示。

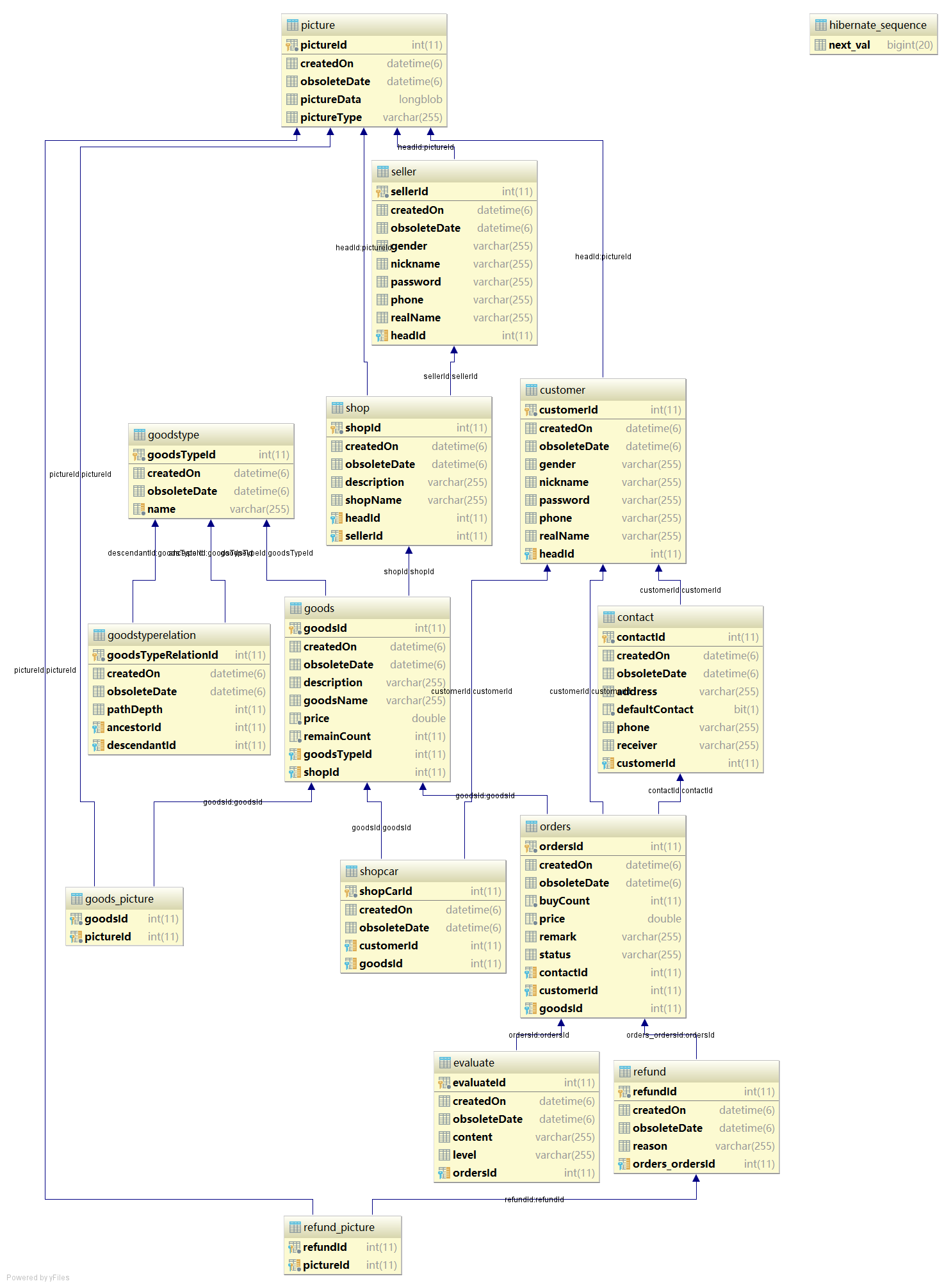


图3.5数据库逻辑结构图

### 3.2.4数据库物理结构设计

1、买家表

**表3.1 买家表字段说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| customerId | int |  | 主键，自动增长 | 买家id |
| phone | varchar | 50 | Unique，非空 | 手机号 |
| password | varchar | 50 | 非空 | 密码 |
| realName | varchar | 50 | 非空 | 姓名 |
| gender | varchar | 2 | 非空 | 性别 |
| Head | blob |  | 非空 | 头像 |

2、卖家表

**表3.2 卖家表字段说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| sellerId | int |  | 主键，自动增长 | 卖家id |
| phone | varchar | 50 | 唯一，非空 | 手机号 |
| password | varchar | 50 | 非空 | 密码 |
| realName | varchar | 50 | 非空 | 姓名 |
| gender | varchar | 2 | 非空 | 性别 |
| Head | blob |  | 非空 | 头像 |

3、商品表

**表3.3 商品表字段说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| goodsId | int |  | 主键，自动增长 | 商品id |
| goodsName | varchar | 20 | 唯一 | 商品名称 |
| description | varchar | 20 | 非空 | 商品描述 |
| price | varchar | 20 | 非空 | 商品价格 |
| goodsTypeId | varchar | 20 | 外键，非空 | 商品分类id |
| shopId | int |  | 外键，非空 | 商店id |

4、订单表

**表3.4 订单表字段说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| ordersId | int |  | 主键，自动增长 | 订单id |
| goodsId | int |  | 外键，非空 | 商品id |
| customerId | int |  | 外键，非空 | 买家id |
| buyCount | int |  | 非空 | 购买数量 |
| contactId | int |  | 外键，非空 | 收货地址id |
| remark | varchar | 50 |  | 备注留言 |
| status | varchar | 20 |  | 订单状态 |

5、图片表

**表3.5 图片表字段说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| pictureId | int |  | 主键，自动增长 | 图片id |
| pictureType | varchar | 10 | 非空 | 图片类型 |
| pictureData | Blob |  | 非空 | 图片二进制数据 |

6、评价表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 描述 |
| evaluateId | int |  | 主键，自动增长 | 评价id |
| ordersId | int |  | 外键，非空 | 订单id |
| content | varchar | 200 | 非空 | 评价内容 |
| level | varchar | 10 | 非空 | 评价等级（好评，中评，差评） |

### 3.2.5数据库物理结构实现

1、买家表

CREATE TABLE `customer` (

`customerId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`gender` varchar(255) DEFAULT NULL,

`nickname` varchar(255) DEFAULT NULL,

`password` varchar(255) DEFAULT NULL,

`phone` varchar(255) DEFAULT NULL,

`realName` varchar(255) DEFAULT NULL,

`headId` int(11) DEFAULT NULL,

PRIMARY KEY (`customerId`),

KEY `FKgersc38dith03ggnf7tskkutp` (`headId`),

CONSTRAINT `FKgersc38dith03ggnf7tskkutp` FOREIGN KEY (`headId`) REFERENCES `picture` (`pictureId`)

)

2、卖家表

CREATE TABLE `seller` (

`sellerId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`gender` varchar(255) DEFAULT NULL,

`nickname` varchar(255) DEFAULT NULL,

`password` varchar(255) DEFAULT NULL,

`phone` varchar(255) DEFAULT NULL,

`realName` varchar(255) DEFAULT NULL,

`headId` int(11) DEFAULT NULL,

PRIMARY KEY (`sellerId`),

KEY `FKrmxbv01wbaodl7hxqapw3idxw` (`headId`),

CONSTRAINT `FKrmxbv01wbaodl7hxqapw3idxw` FOREIGN KEY (`headId`) REFERENCES `picture` (`pictureId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

3、商品表

CREATE TABLE `goods` (

`goodsId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`description` varchar(255) DEFAULT NULL,

`goodsName` varchar(255) DEFAULT NULL,

`price` double NOT NULL,

`remainCount` int(11) NOT NULL,

`goodsTypeId` int(11) DEFAULT NULL,

`shopId` int(11) DEFAULT NULL,

PRIMARY KEY (`goodsId`),

KEY `FKehmg3k7j3l9rhnkoqyuh7tbac` (`goodsTypeId`),

KEY `FKlmbbumv9jpniv0mbdd8u2y0jy` (`shopId`),

CONSTRAINT `FKehmg3k7j3l9rhnkoqyuh7tbac` FOREIGN KEY (`goodsTypeId`) REFERENCES `goodstype` (`goodsTypeId`),

CONSTRAINT `FKlmbbumv9jpniv0mbdd8u2y0jy` FOREIGN KEY (`shopId`) REFERENCES `shop` (`shopId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

4、订单表

CREATE TABLE `orders` (

`ordersId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`buyCount` int(11) NOT NULL,

`price` double NOT NULL,

`remark` varchar(255) DEFAULT NULL,

`status` varchar(255) DEFAULT NULL,

`contactId` int(11) DEFAULT NULL,

`customerId` int(11) DEFAULT NULL,

`goodsId` int(11) DEFAULT NULL,

PRIMARY KEY (`ordersId`),

KEY `FKbnx6jjc9w9ybes1haph2dwdev` (`contactId`),

KEY `FK1afv963odd6tgw8i6fttjc0f` (`customerId`),

KEY `FKgxs45pb28fkvmv79dt7vmc920` (`goodsId`),

CONSTRAINT `FK1afv963odd6tgw8i6fttjc0f` FOREIGN KEY (`customerId`) REFERENCES `customer` (`customerId`),

CONSTRAINT `FKbnx6jjc9w9ybes1haph2dwdev` FOREIGN KEY (`contactId`) REFERENCES `contact` (`contactId`),

CONSTRAINT `FKgxs45pb28fkvmv79dt7vmc920` FOREIGN KEY (`goodsId`) REFERENCES `goods` (`goodsId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. 图片表

CREATE TABLE `picture` (

`pictureId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`pictureData` longblob,

`pictureType` varchar(255) DEFAULT NULL,

PRIMARY KEY (`pictureId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. 评价表

CREATE TABLE `evaluate` (

`evaluateId` int(11) NOT NULL,

`createdOn` datetime(6) DEFAULT NULL,

`obsoleteDate` datetime(6) DEFAULT NULL,

`content` varchar(255) DEFAULT NULL,

`level` varchar(255) DEFAULT NULL,

`ordersId` int(11) DEFAULT NULL,

PRIMARY KEY (`evaluateId`),

KEY `FK5lkjkfds614jc2m28fw1lny7n` (`ordersId`),

CONSTRAINT `FK5lkjkfds614jc2m28fw1lny7n` FOREIGN KEY (`ordersId`) REFERENCES `orders` (`ordersId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

## 3.3系统结构设计

### 3.3.1系统体系结构设计

#### 3.3.1.1网络结构模式

该系统的的后台管理系统使用B/S结构，即浏览器/服务器模式。它是C/S架构的一种改进，可以说是属于三层C/S架构主要利用不断成熟的浏览器技术，用通用浏览器就实现啦原来需要复杂专用软件才能实现的强大技术，并节约开发成本。

**表3.6 网络结构**

|  |
| --- |
| 第一层：客户端  表现层，用于界面引导，接受用户输入，并向服务器发送请求 |
| 第二层：应用服务器  业务逻辑层：执行业务逻辑，向数据库发送请求 |
| 第三层：数据库服务器  数据存储层：执行数据逻辑，运行SQL或者存储过程 |

第一层是浏览器，即客户端，只有简单的输入和输出功能，几乎不处理事务逻辑。客户不需要安装客户端，在浏览器就能浏览，因此它适合大众化，界面比较简洁好用。

第二层是WEB服务器，主要作用是信息传送。当用户想要访问数据库时，就会首先向WEB服务器发送请求，WEB服务器统一请求后会向数据库服务器发送访问数据库的请求，数据库就会通过SQL语句得到用户想要的数据。

第三层是数据库服务器，它存放着大量的数据。当数据库服务器收到了WEB服务器的请求后，会对SQL语句进行处理，并将返回的结果发送给WEB服务器，接下来，WEB服务器将收到的数据结果转换为HTML文本形式发送给浏览器，也就是我们打开浏览器看到的界面。

买家访问Web购物网站的数据过程，如图3.8所示。

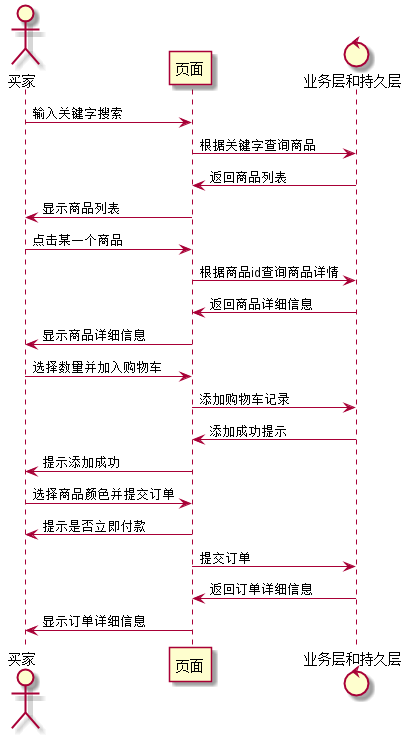


图3.6买家访问Web购物网站数据过程图

#### 3.3.1.2网络协议

一般因特网可以分为五层，[应用层](http://baike.baidu.com/subview/239619/16373356.htm" \t "_blank)、[传输层](http://baike.baidu.com/view/239605.htm" \t "_blank)、[网络层](http://baike.baidu.com/subview/239600/16505450.htm" \t "_blank)、[链路层](http://baike.baidu.com/subview/4874044/4874488.htm" \t "_blank)和[物理层](http://baike.baidu.com/subview/239585/239585.htm" \t "_blank)。

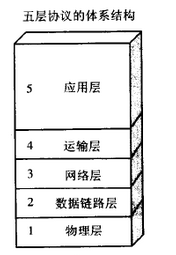


图3.7五层网络协议图

其中传输层负责为信源和信宿提供应用程序进程间的数据传输服务，这一层上主要定义了两个传输协议，传输控制协议即TCP和用户数据报协议UDP，而网络层负责将称为数据报([datagram](http://baike.baidu.com/subview/638347/638347.htm" \t "_blank))的网络层分组从一合主机移动到另一台主机

TCP（Transmission Control Protocol，传输控制协议）是面向连接的协议，也就是说，在收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来。

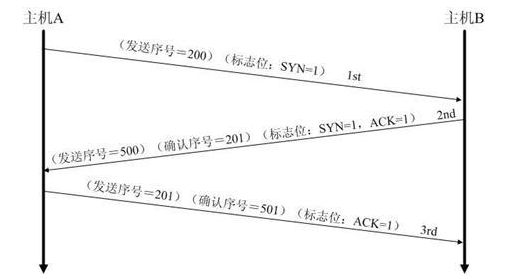


图3.8三次握手过程图

TCP三次握手过程：

1)、主机A通过向主机B 发送一个含有同步序列号的标志位的数据段给主机B ,向主机B 请求建立连接,通过这个数据段,  
主机A告诉主机B 两件事:我想要和你通信;你可以用哪个序列号作为起始数据段来回应我.  
 2)、主机B 收到主机A的请求后,用一个带有确认应答(ACK)和同步序列号(SYN)标志位的数据段响应主机A,也告诉主机A两件事:  
我已经收到你的请求了,你可以传输数据了;你要用哪佧序列号作为起始数据段来回应我  
 3)、主机A收到这个数据段后,再发送一个确认应答,确认已收到主机B 的数据段:"我已收到回复,我现在要开始传输实际数据了  
 这样3次握手就完成了,主机A和主机B 就可以传输数据了。

UDP是一个非连接的协议，传输的开始端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP把每个消息段放在队列中，应用程序每次从队列中读一个消息段。由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。

在取号叫号中我们需要服务器的响应，需要可靠安全不丢失数据，所以我们将采用TCP协议。

我们的客户端和服务端是采用socket进行连接的，所以需要采用socket的TCP编程模式来创建客户端和服务端。

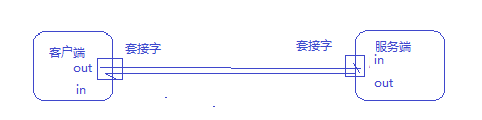


图3.9客户端与服务端Socket连接图

#### 3.3.1.3系统分层设计

系统可以分为两大部分，前端和后台。后台使用javaWeb实现，以Spring + SpringMVC + Hibernate 为基本框架，以MVC三层模型为基本模型，实现web页面对数据的访问。前端使用Html5实现，以 jsp + vue + jquery + bootstrap 为基本框架，以管理数据和状态代替DOM操作。

### 3.3.2系统功能模块设计

通过需求分析可以了解到，Web购物网站的基本功能包括注册登录，买家购买商品，卖家管理商品。其划分功能的功能模块结构如表3.7所示。

**表3.7 系统功能模块表**

|  |  |
| --- | --- |
| 功能类别 | 子功能 |
| 帐户管理 | 注册 |
| 登录 |
| 修改个人信息 |
| 买家购买商品 | 首页浏览商品 |
| 搜索商品 |
| 显示商品详情 |
| 添加商品到购物车 |
| 下单 |
| 管理收货地址 |
| 确认收货 |
| 评价订单 |
| 申请退款 |
| 卖家管理商品 | 管理商店 |
| 管理商品 |
| 管理订单 |

#### 3.3.2.1注册登录的流程

买家和卖家在注册登录的流程是一样的。如果没有帐号，就先注册，需要填入手机号，密码，昵称，姓名，性别等个人信息。注册成功后就可以使用该手机号和密码登录。登录之后可以修改个人信息。

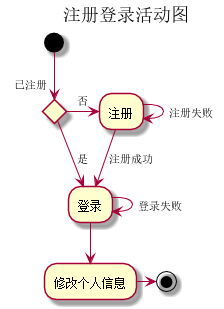


图3.10注册登录活动图

#### 3.3.2.2买家购买商品的流程

买家登录后，可以根据关键字，商品分类等条件查询商品后，查看商品详细信息，添加到购物车，下单。下单时需要选择收货地址。收货地址可以添加，也可以选择已有的收货地址。下单后买家可以查询订单信息。在收到商品后，买家需要确认收货，然后可以评价订单。在下单后的整个时期，买家都可以申请退款。

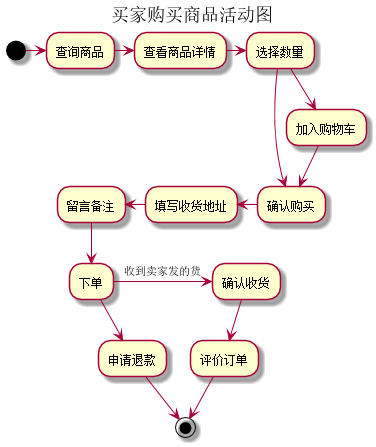


图3.11买家购买商品活动图

#### 3.3.2.3卖家管理商品的流程

卖家注册登录后，会在首页看到所有属于自己的商店，可以创建商店，修改商店信息，删除商店（会同时删除商店下的所有商品）。进入商店页面后可以看到该商店下的所有商品。此时可以添加，删除商品。也可以进入商品详情页查看和修改商品信息。

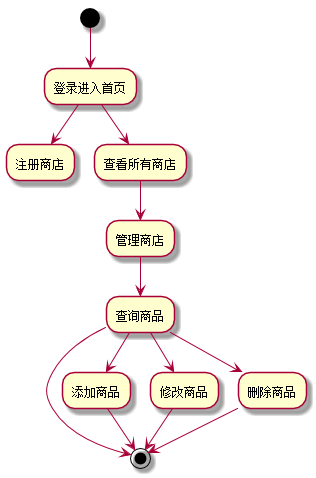


图3.12卖家管理商品活动图

#### 3.3.2.4卖家处理订单的流程

卖家在登录后，可以查询并处理相关的订单，比如发货后需要对订单进行确认发货的操作。买家申请退款时，如果已经把钱退回给买家，需要对订单进行确认退款操作。

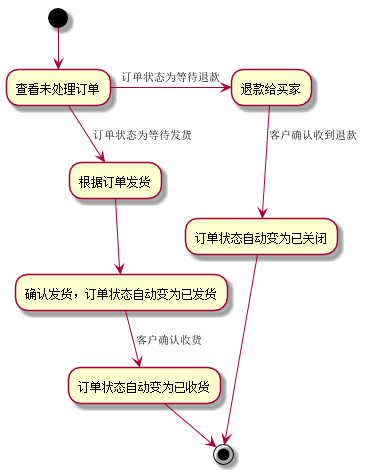


图3.13卖家处理订单活动图

# 系统实现

## 4.1买家相关功能的时序图

### 4.1.1买家购买商品时序图

买家登录后，可以根据关键字，商品分类等条件查询商品后，查看商品详细信息，添加到购物车，下单。下单时需要选择收货地址。收货地址可以添加，也可以选择已有的收货地址。下单后买家可以查询订单信息。在收到商品后，买家需要确认收货，然后可以评价订单。在下单后的整个时期，买家都可以申请退款。

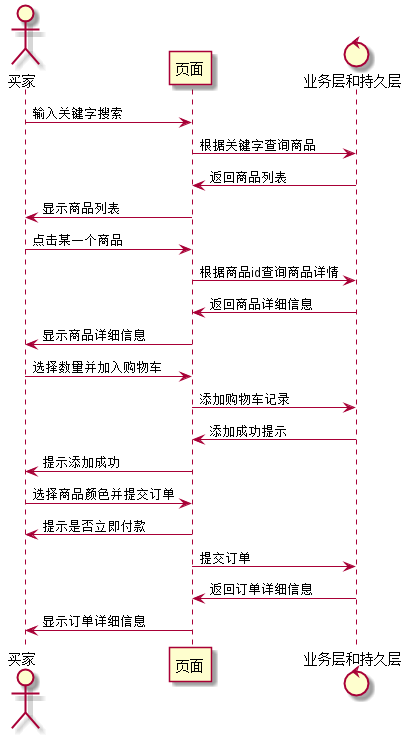


图4.1买家购买商品时序图

## 4.2卖家相关功能的时序图

### 4.2.1卖家管理商品时序图

卖家注册登录后，会在首页看到所有属于自己的商店，可以创建商店，修改商店信息，删除商店（会同时删除商店下的所有商品）。进入商店页面后可以看到该商店下的所有商品。此时可以添加，删除商品。也可以进入商品详情页查看和修改商品信息。

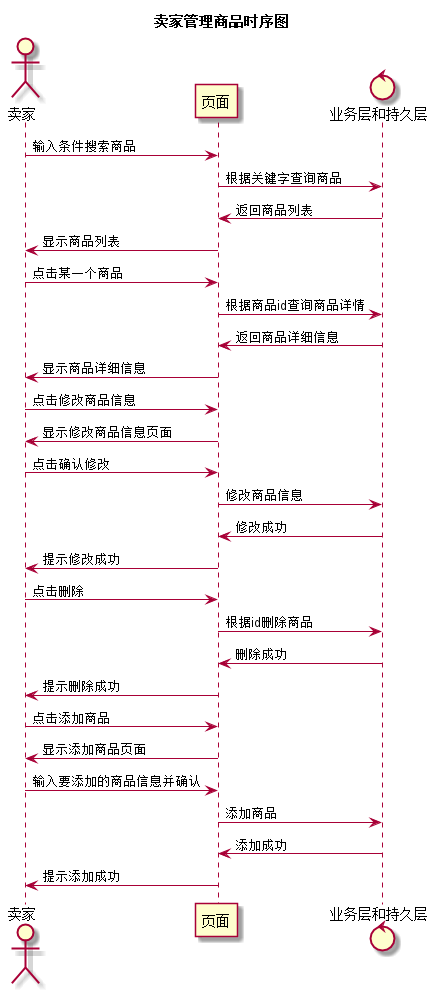


图4.2卖家管理商品时序图

### 4.2.2卖家处理订单时序图

卖家在登录后，可以查询并处理相关的订单，比如发货后需要对订单进行确认发货的操作。买家申请退款时，如果已经把钱退回给买家，需要对订单进行确认退款操作。

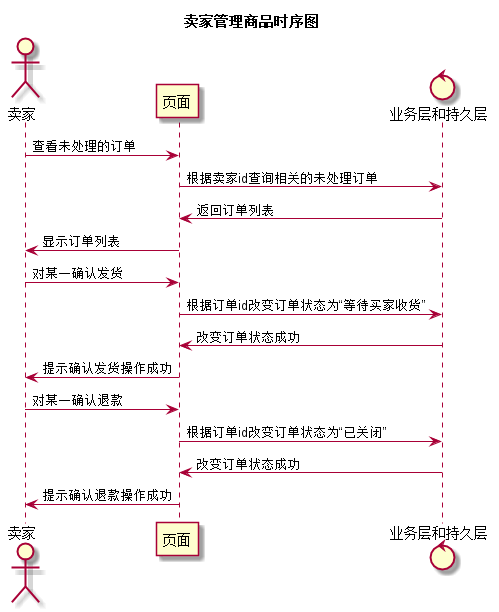


图4.3卖家处理订单时序图

## 4.3买家相关功能的实现

### 4.3.1买家首页

买家首页的界面如图4.4所示。

买家在首页可以根据关键字，商品分类进行查询商品。支持分页功能，可以在不同的子页中查看商品。支持排序功能，可以按照不同的条件进行排序，比如按价格由低到高排序，或者由高到低排序。

首页中的商品分页默认显示第1页，按照综合排序，搜索栏可以输入关键字搜索商品，分类栏可以筛选买家感兴趣的商品分类的商品。买家可以点击商品图片进入某个商品的详情页。

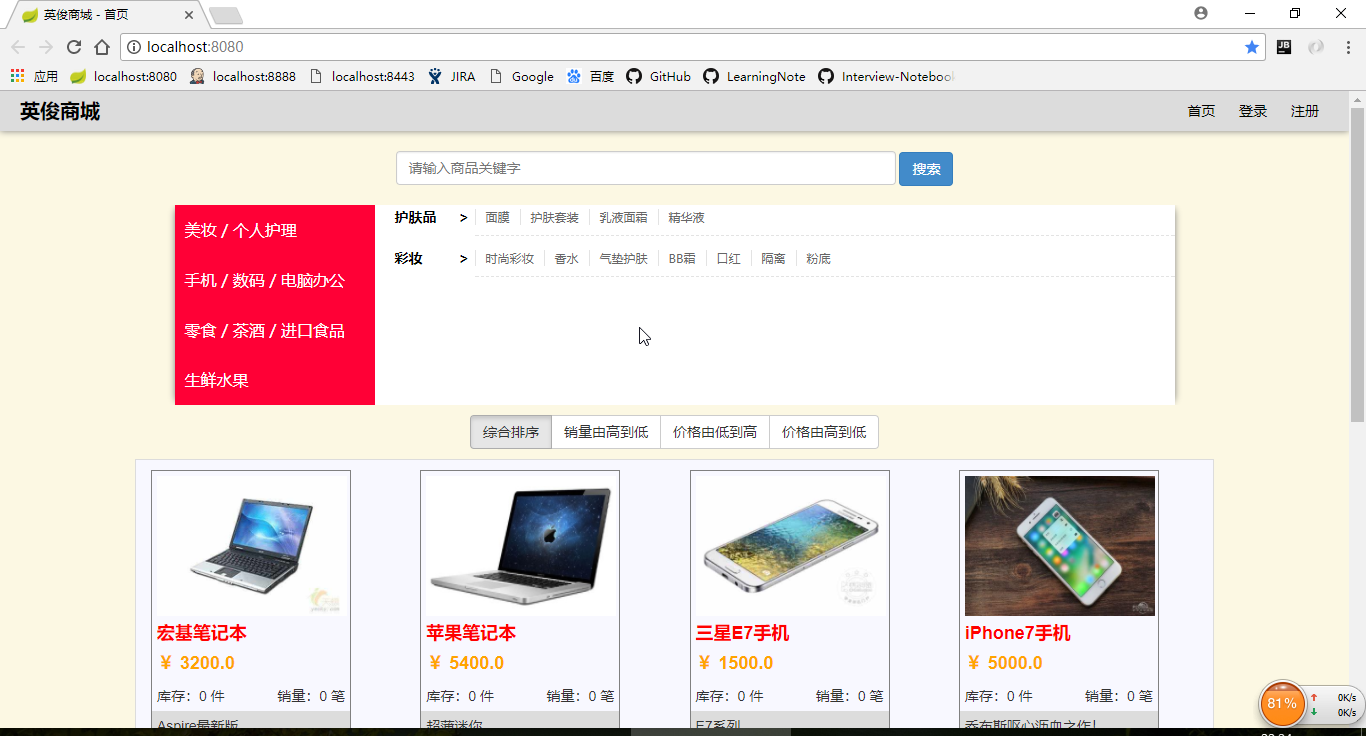


图4.4买家首页

相关的代码：

@Controller

public class IndexController extends BaseController {

@Resource

private GoodsDao goodsDao;

@Resource

private GoodsTypeDao goodsTypeDao;

@RequestMapping("/")

public String showIndex(

HttpServletRequest request,

HttpServletResponse response,

@RequestParam(defaultValue = "0") int pageIndex,

@RequestParam(defaultValue = "0") int sortType) throws ServletException, IOException {

if (getLoginSellerFromSession(request) != null) {

request.getRequestDispatcher("/sellerIndex").forward(request, response);

return null;

}

List<GoodsType> goodsTypeList = goodsTypeDao.queryAll();

List<Goods> goodsList = goodsDao.begin(12 \* pageIndex).count(12).sortType(sortType).queryAll();

long totalCount = goodsDao.queryCount(null);

request.getSession().setAttribute("goodsTypeList", goodsTypeList);

request.setAttribute("pageIndex", pageIndex);

request.setAttribute("totalCount", totalCount);

request.setAttribute("goodsList", goodsList);

request.setAttribute("sortType", sortType);

return "index";

}

}

### 4.3.2商品详情

商品详情页的界面如图4.5所示。

页面包含商品的基本信息如名称，价格，描述，分类。同时还有商店信息，买家可以点击“进店”查看本商店的所有商品，也可以点击联系卖家查看卖家的信息，包括卖家的姓名，电话。

下面是对于该商品的评价内容，有评价用户的信息，评价日期，评价内容，评价等级（好评，中评，差评），帮助买家识别商品，充分考虑商品是否适合自己。

最后是操作。填好购买数量后可以添加购物车，也可以点击“购买”直接下单。

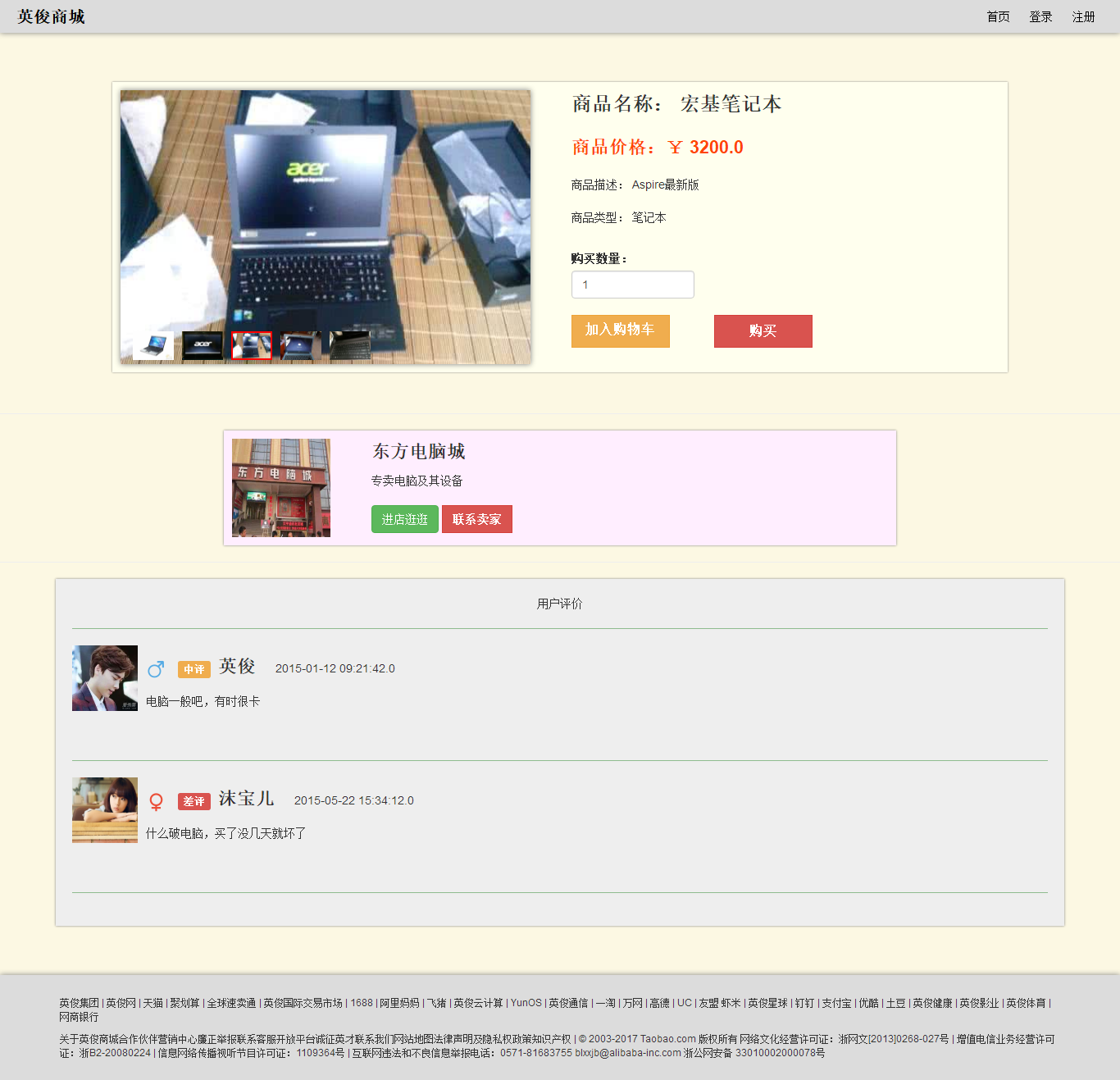


图4.5商品详情页

相关的代码：

@GetMapping("/goods/{goodsId}")

public String showGoodsInfo(HttpServletRequest request, HttpServletResponse response, @PathVariable Integer goodsId) {

Goods goods = goodsDao.queryById(goodsId);

List<Evaluate> evaluateList = evaluateDao.queryByGoodsId(goodsId);

request.setAttribute("goodsJson", GsonConverter.toJson(goods,

"GoodsAttrName.goods", "GoodsAttrValue.goodsAttrName", "Seller.shopList"));

request.setAttribute("evaluateListJson", GsonConverter.toJson(evaluateList,

"Orders.contact", "Orders.refund", "Goods.goodsAttrNames", "Seller.shopList"));

return "goods\_info";

}

### 4.3.3注册

用户在一开始使用本系统时，必须先注册。需要填写手机号，密码，昵称，姓名，性别和身份。注册成功后会自动跳到登录页面并填充注册时的手机号和密码。注册页会动态检查手机号是否可以注册，提高用户体验。



图4.6注册页

相关代码：

@Resource

private SellerDao sellerDao;

@Resource

private CustomerDao customerDao;

@Resource

private PictureDao pictureDao;

@Resource

private SessionFactory sessionFactory;// TODO 这个东西不利于测试，以后想办法移到dao层！

@PostMapping("/register")

public String register(HttpServletRequest request,

String phone,

String password,

String realName,

String nickname,

String gender,

String identity) throws IOException {

if (identity.equals("customer")) {

if (customerDao.queryByPhone(phone) != null) {

request.setAttribute("msg", "该手机号已注册");

return "register";

}

String picturePath = request.getServletContext().getRealPath("/admin/img/user\_default\_head.jpg");

FileInputStream fis = new FileInputStream(picturePath);

// TODO 以后OpenSessionInView不用之后，要改为openSession

Blob pictureData = sessionFactory.getCurrentSession().getLobHelper().createBlob(fis, fis.available());

Picture head = new Picture(Picture.PictureType.jpg, pictureData);

pictureDao.insert(head);

Customer customer = new Customer(phone, password, realName, nickname, "man".equals(gender) ? "男" : "女", head);

customerDao.insert(customer);

request.setAttribute("phone", phone);

request.setAttribute("password", password);

request.setAttribute("identity", identity);

return "login";

} else {

if (sellerDao.queryByPhone(phone) != null) {

request.setAttribute("msg", "该手机号已注册");

return "register";

}

String picturePath = request.getServletContext().getRealPath("/admin/img/user\_default\_head.jpg");

FileInputStream fis = new FileInputStream(picturePath);

// TODO 以后OpenSessionInView不用之后，要改为openSession

Blob pictureData = sessionFactory.getCurrentSession().getLobHelper().createBlob(fis, fis.available());

Picture head = new Picture(Picture.PictureType.jpg, pictureData);

pictureDao.insert(head);

Seller seller = new Seller(phone, password, realName, nickname, "男".equals(gender) ? "男" : "女", head);

sellerDao.insert(seller);

request.setAttribute("phone", phone);

request.setAttribute("password", password);

request.setAttribute("identity", identity);

return "login";

}

}

### 4.3.4登录

使用手机号和密码登录。同时还要选择身份，如果身份选错了，即时手机号密码都正确，也会登录失败。此外，登陆时还要输入验证码，验证是否为正常用户登录。



图4.7登录页

相关代码：

@Resource

private SellerDao sellerDao;

@Resource

private CustomerDao customerDao;

@PostMapping("/login")

public String login(HttpServletRequest request,

HttpServletResponse response,

@RequestParam String phone,

@RequestParam String password,

@RequestParam String identity,

@RequestParam String validateCode,

String autoLogin) throws IOException {

removeCookie(request);// 如果客户点击登录了，就移除之前的cookie，防止登录失败时变回cookie的内容。

// 如果验证码不匹配

if (!ImageCode.validateCode(request, validateCode)) {

request.setAttribute("msg", "验证码错误");

return "login";

}

if (identity.equals(C.SESSION\_CUSTOMER)) {// 如果是客户登录

Customer customer = customerDao.queryByPhone(phone);

if (customer != null && password.equals(customer.getPassword())) {

request.getSession().invalidate();

setLoginCustomerToSession(request, customer);

addCookie(response, phone, password, C.SESSION\_CUSTOMER, autoLogin);

// request.getRequestDispatcher("/").forward(request, response);

response.sendRedirect("/");

return null;

} else {

request.setAttribute("msg", "用户名或密码错误");

return "login";

}

} else {// 如果是卖家登录

Seller seller = sellerDao.queryByPhone(phone);

if (seller != null && password.equals(seller.getPassword())) {

request.getSession().invalidate();

setLoginSellerToSession(request, seller);

addCookie(response, phone, password, C.SESSION\_SELLER, autoLogin);

// request.getRequestDispatcher("/").forward(request, response);

response.sendRedirect("/");

return null;

} else {

request.setAttribute("msg", "用户名或密码错误");

return "login";

}

}

}

### 4.3.5购物车

可以查看在购物车中的商品，可以执行结算操作。



图4.8购物车页

相关代码：

@Resource

private GoodsDao goodsDao;

@Resource

private ShopCarService shopCarService;

@GetMapping

public String list(HttpServletRequest request, @RequestParam(defaultValue = "0") int pageIndex) {

Customer customer = getLoginCustomerFromSession(request);

List<Goods> goodsList = goodsDao.queryByCustomerId(customer.getCustomerId(), 12 \* pageIndex, 12);

int totalCount = goodsDao.queryCountByCustomerId(customer.getCustomerId());

request.setAttribute("pageIndex", pageIndex);

request.setAttribute("totalCount", totalCount);

request.setAttribute("goodsList", goodsList);

return "shop\_car";

}

### 4.3.6创建订单

确认好商品信息和价格，并填写收货地址和备注信息后，就可以付款下单了。



图4.9创建订单页

相关代码：

@PostMapping("/confirmOrders")

public String confirmOrders(HttpServletRequest request, int goodsId, int count) {

Customer customer = getLoginCustomerFromSession(request);

Goods goods = goodsDao.queryById(goodsId);

List<Contact> contactList = contactDao.queryByCustomerId(customer.getCustomerId());

request.setAttribute("customerJson", GsonConverter.toJson(customer));

request.setAttribute("goodsJson", GsonConverter.toJson(goods, "Goods.shop",

"GoodsAttrName.goods", "GoodsAttrValue.goodsAttrName", "Seller.shopList"));

request.setAttribute("contactListJson", GsonConverter.toJson(contactList));

request.setAttribute("count", count);

return "create\_orders";

}

### 4.3.7选择收货地址

买家在创建订单时，需要填写收货地址。可以选择已经存在的收货地址，也可以添加收货地址并选择。

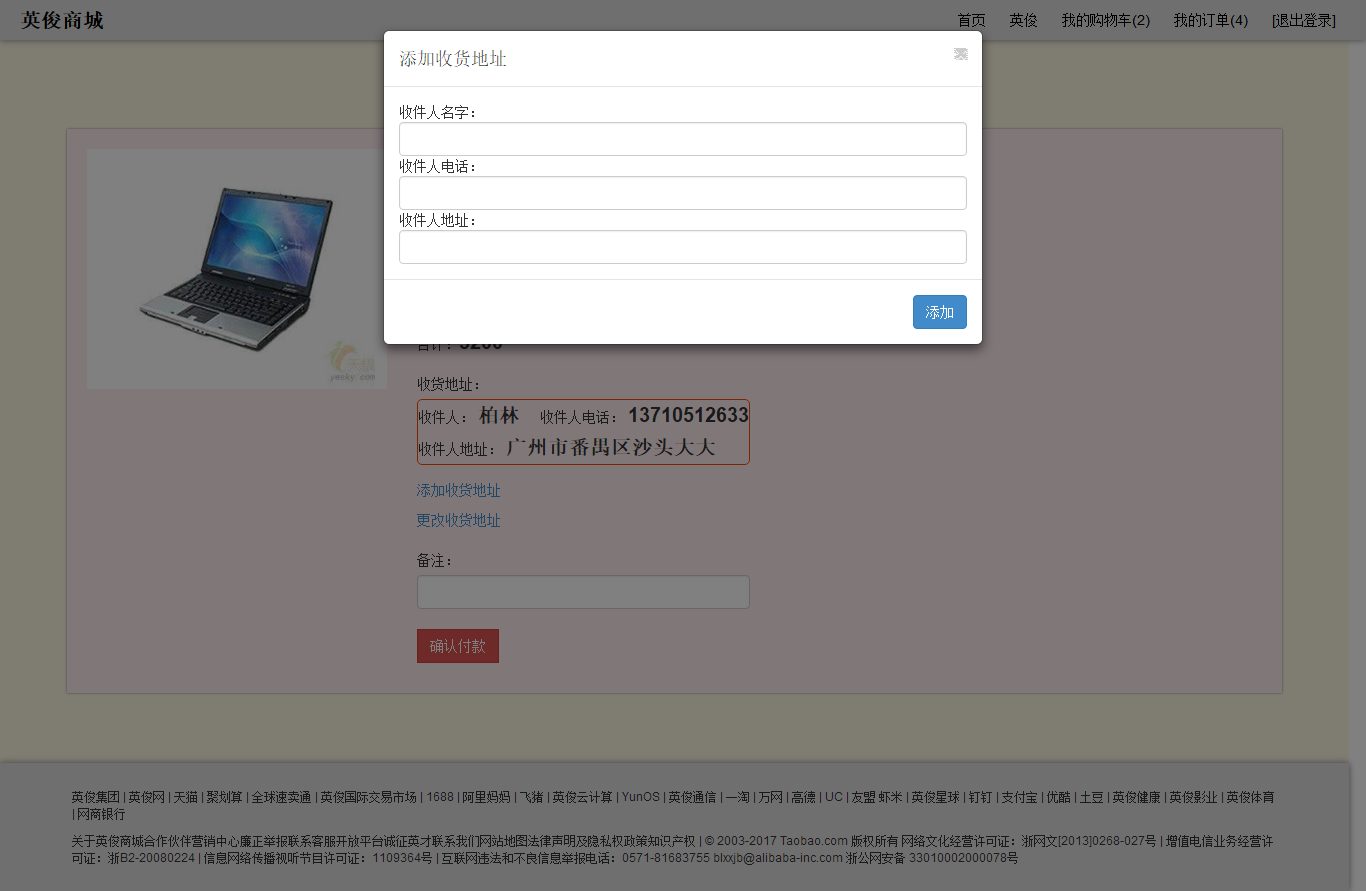


图4.10添加收货地址页

买家在创建订单时，需要填写收货地址。可以选择已经存在的收货地址，也可以添加收货地址并选择。



图4.8选择收货地址页

相关代码：

@Resource

private ContactDao contactDao;

@GetMapping

public List<Contact> list(@PathVariable int customerId) {

return contactDao.queryByCustomerId(customerId);

}

@PostMapping

public RequestStatus<Integer> add(@PathVariable int customerId,

@Valid Contact contact, BindingResult result) {

if (result.hasErrors()) {

throw new IllegalArgumentException(result.getAllErrors().get(0).getDefaultMessage());

}

contact.setCustomer(new Customer(customerId));

contactDao.insert(contact);

return RequestStatus.success(contact.getContactId());

}

### 4.3.8查看订单

买家可以在该页面查看自己的所有订单。订单按照创建日期由新到旧排序。每个订单都有对应的一个状态。不同的状态可以执行的操作也不一样。

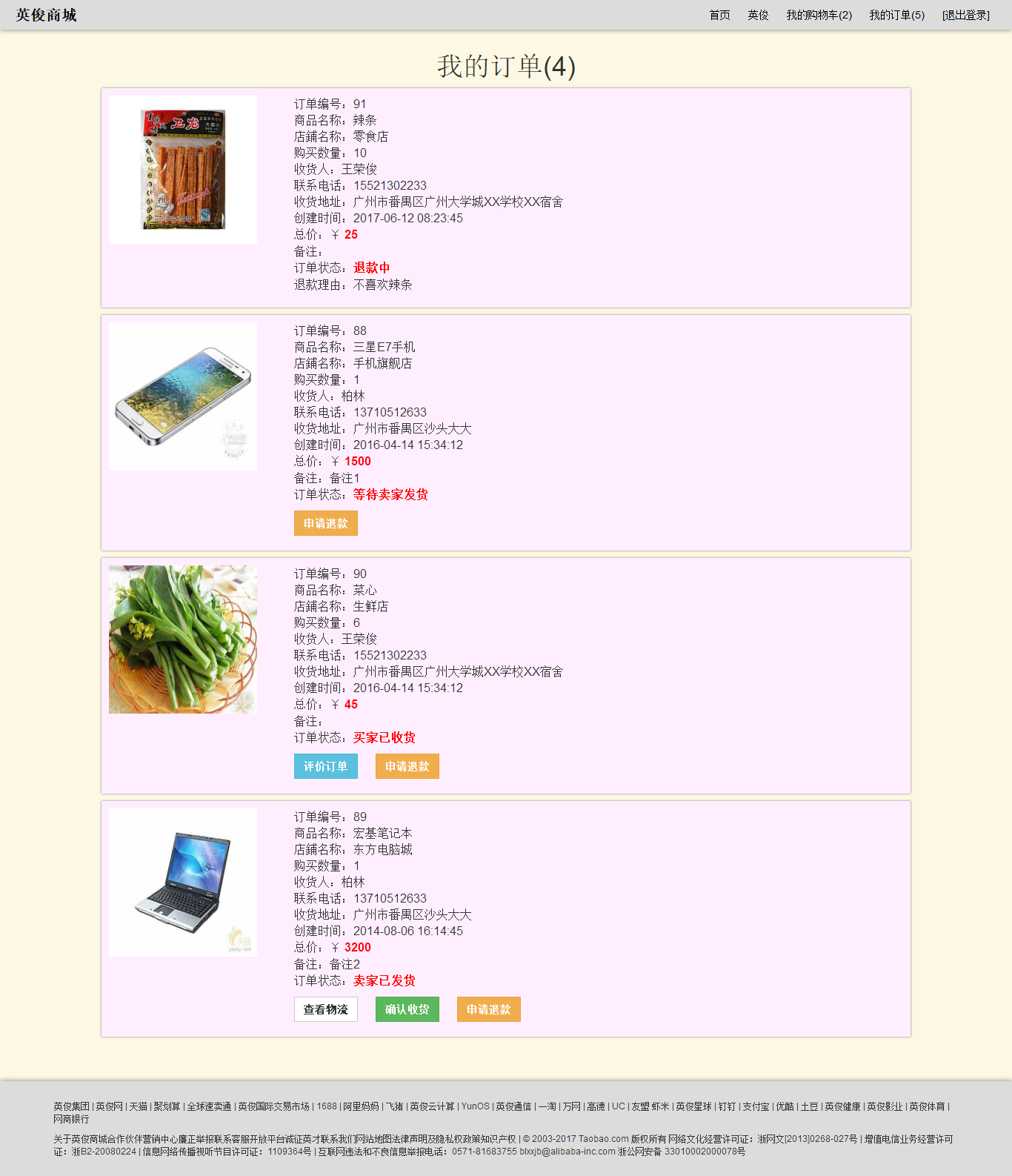


图4.11订单页

相关代码：

@GetMapping("/orders")

public String listOrders(HttpServletRequest request) {

int customerId = getLoginCustomerFromSession(request).getCustomerId();

Pager<Orders> pager = ordersController.listByCustomer(customerId, new PageParam("-createdOn"), createBR());

request.setAttribute("ordersCount", pager.getTotalCount());

request.setAttribute("ordersListJson", GsonConverter.toJson(pager.getDataList(),

"Shop.seller", "Refund.orders", "Goods.goodsAttrNames"));

return "customer\_orders\_list";

}

### 4.3.10评价订单

买家在确认收货后，可以对订单进行评价。填写评价内容，并且选择评价等级。等级相当于对本次购物的满意度，分为三个等级：好评，中评，差评。



图4.12订单评价页

相关代码：

/\*\*

\* 买家评价订单

\*/

@PutMapping("/{ordersId}/action/customerEvaluateOrders")

public RequestStatus customerEvaluateOrders(@PathVariable Integer ordersId,

@RequestParam String evaluateContent,

@RequestParam Evaluate.Level evaluateLevel) {

if (evaluateLevel == null) {

throw new IllegalArgumentException("evaluateLevel is null");

}

Orders orders = ordersDao.queryById(ordersId);

Orders.Status newStatus = getNewStatus(orders.getStatus(), "customerEvaluateOrders");

orders.setStatus(newStatus);

ordersDao.update(orders);

Evaluate evaluate = new Evaluate(orders, evaluateContent, evaluateLevel);

evaluateDao.insert(evaluate);

Map<String, Object> result = new HashMap<>();

result.put("newStatus", newStatus.toString());

return RequestStatus.success(result);

}

### 4.3.11申请退款

买家在下单后的整个过程，如果不想购买商品，都可以申请退款，但需要填写退款理由。系统有一些默认的退款理由。不过买家也可以写其他的理由。



图4.13订单申请退款页

相关代码：

/\*\*

\* 买家申请退款

\*/

@PutMapping("/{ordersId}/action/customerApplyForRefund")

@ReturnObjectToJsonIgnoreFields("Refund.orders")

public RequestStatus customerApplyForRefund(@PathVariable Integer ordersId,

@RequestParam String refundReason) {

Orders orders = ordersDao.queryById(ordersId);

Orders.Status newStatus = getNewStatus(orders.getStatus(), "customerApplyForRefund");

orders.setStatus(newStatus);

ordersDao.update(orders);

Refund refund = new Refund(orders, refundReason, null);

refundDao.insert(refund);

Map<String, Object> result = new HashMap<>();

result.put("newStatus", newStatus.toString());

result.put("refund", refund);

return RequestStatus.success(result);

}

## 4.2卖家相关功能的实现

### 4.2.1卖家首页

卖家在登录后，会自动跳转到首页。首页显示卖家所拥有的商店。可以修改个人信息，也可以点击进去查看商店信息和其下的商品，也可以创建商店。



图4.14卖家首页

### 4.2.2卖家创建商店

卖家在首页点击“创建商店”按钮后，会弹出输入框提示输入商店的信息，包括商店名称，描述。填写完后点击创建商店，首页就会多了新创建的商店。由于使用了ajax技术，所以不会引起页面刷新。

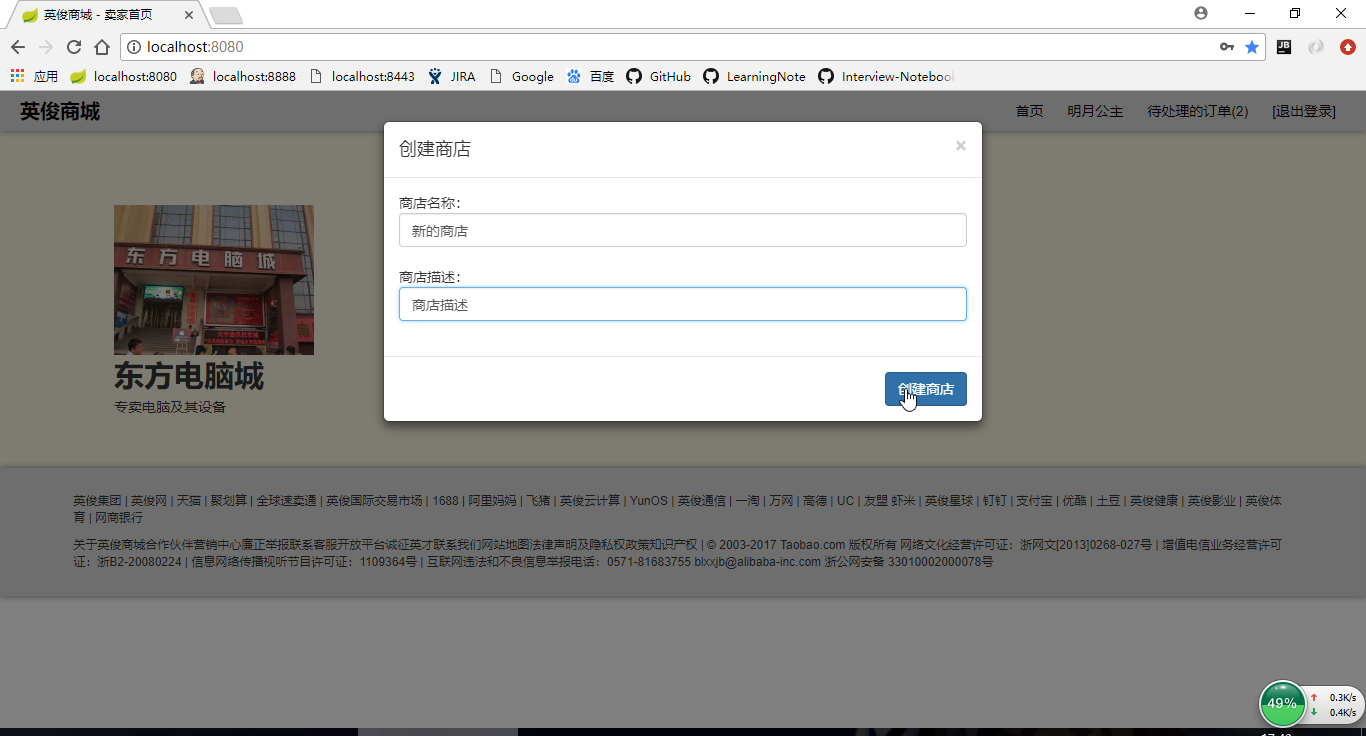


图4.15卖家首页-创建商店

# 总结与展望

## 5.1总结

本论文系统与电商的运营紧密相连。在系统的开发当中，通过对于实际流程的观察，制订了系统的需求分析和功能模块。实际的电商可以提供非常完善的服务和规范流程来促使购物的平稳进行。

在实际开发当中遇到了许多问题，基本通过网上查找和与同学交流解决了。在解决问题的过程当中，学会了解决问题的方法，练就了一个细心查找问题的所在的耐心。

在整个论文和系统开发过程当中，主要的工作结论如下。

1、从需求分析，系统设计和系统实现等方面，基本理解Web购物网站的研发一样，明确了系统设计的思想和过程。

2、结合电商平台的实际流程，通过自己的需求分析，提出系统的构建思路，确定了各个功能模块，并实现了一个电商平台大部分的核心功能。

3、结合电商的当前业务的流程和状况，收集各种业务的数据，完成后台数据管理和报表分析，有效的提升了分析数据的效率。

4、鉴于时间和技术的问题，系统还有部分功能没有达到预期的目的，还有待改善。

## 5.2展望

目前本系统只能在自己的个人电脑上运行，也就是说只有一台服务器提供服务。随着业务的持续增长，需要通过更有效的方式提高系统的效率与稳定性。在后续的开发中，将会考虑将分布式，负载均衡，数据热备份，微服务架构等功能加上去，进一步简化业务流程，提高系统效率与稳定性，保证系统的高可用和高并发。

# 参考文献

1. Java从入门到精通（第4版）,清华大学出版社
2. Spring实战（第4版），人民邮电出版社
3. 萨师煊，数据库系统概论[M]，北京高等教育出版社，2006
4. （美） Nicholas C. Zakas, JavaScript高级程序设计（第3版）,人民邮电出版社
5. （美）JonathanChaffer ，jQuery基础教程，人民邮电出版社
6. Eric A. Meyer, CSS权威指南, 美, 中国电力出版社
7. Vue官方学习文档：<https://cn.vuejs.org/v2/guide/>
8. Bootstrap中文网：https://v3.bootcss.com/