廣東財經大學
GUANGDONG UNIVERSITY OF FINANCE & ECONOMICS

学 年 论 文

院　　系：	信息学院
专　　业：	计算机科学与技术
班　　级：	18 计算机科学与技术 1 班
学　　号：	18251102141
姓　　名：	彭海涛
指导教师：	罗勇
提交日期：	2020 年 12 月 28 日

广财经大学教务处　制

姓名_____　　学年论文成绩_____

评语：

选题符合专业培养目标和专业特点，具有实际意义和实践价值。

该生设计和开发了一个二手商品交易网站。设计开发流程正确，选择的技术方案，所设计的系统结构和系统数据库均合理。整个设计与开发过程，表现出较强的查阅资料，利用所学知识解决实际问题的能力。测试表明，实现了同类系统的核心功能，界面友好，操作简单。体现了该生具有独立开发小型应用软件的一定能力。

撰写的设计文档结构清晰，行文较流畅，内容较充实，格式和图表较规范，较为全面地反映了整个设计流程。

作为学年论文，可以说该生完成了较大的工作量，是一篇较好的论文。但是该生的系统与目前已有的系统相比缺乏创新性与针对性，尚不具备独特的竞争力；尚需加深对同类系统的了解，使系统的某些功能更合理更方便；在测试方面不仅仅做功能测试，还需加强鲁棒性测试。建议在后续的毕业设计阶段进一步完善。

指导教师（签名）_____

年　月　日

说明：指导教师评分后，学年论文交院（系）办公室保存。

基于 B/S 架构的二手商品交易系统

目录

中文摘要

随着网络通讯和计算机应用技术的发展，人们在网上发布自己的闲置商品或者在网上购买二手商品已成为一种新的商品交易方式。

B/S 架构是目前最流行的系统架构之一。本文基于 B/S 架构设计与实现了二手商品交易系统。在需求分析与系统设计阶段使用了软件开发工具 PowerDesigner 对系统进行建模，在编码实现阶段使用了 Java、Servlet、JSP、MySQL 等技术构建了系统。系统在功能上划分为三层，即用户表现层（UIL）、业务逻辑层（BLL）和数据访问层（DAL），实现了同类系统的三大核心功能，即账户管理功能、购买商品功能、销售商品功能。

经过测试，基于 B/S 架构的二手商品交易系统运行正常，各模块功能均正确实现，而且界面友好，操作简单，基本实现了既定的开发目标。但系统在安全性和功能多样性上仍存在不足，需要继续深入研究以达到更好的应用效果。

关键词：二手商品交易系统 B/S 架构 Java MySQL

Abstrat

With the development of network communication and computer application technology, it has become a new second-hand trading method for people to publish their idle goods online or to buy second-hand goods online.

B/S architecture is one of the most popular system architectures. Based on B/S architecture, this paper designs and implements a second-hand trading system. The software development tool PowerDesigner is used to model the system in the requirement analysis and system design stage, and the Java、Servlet、JSP、MySQL technology is used to construct the system according to the model in the coding implementation stage. The system is divided into three layers, namely, user interface layer (UIL), business logic layer (BLL) and data access layer (DAL), which realize the three core functions of the similar system, that is, account Management function, purchasing function, selling function.

After testing, the second-hand trading system based on B/S architecture runs normally, each module function is realized correctly, and the interface is friendly, the operation is simple, and the established development goal is basically realized. However, the system still has shortcomings in security and functional diversity, so it needs to be further studied to achieve better application results.

Key words： Second hand trading system　B/S Architecture

Java　MySQL

1 绪论

1.1 项目背景

二手商品交易市场从古至今都一直存在，它是一种买卖旧物品的市场。对于旧物品的出售者来说，他们能通过二手商品市场来出售自己用不到的物品，来获得收益。对于旧货的购买者来说，他们能够花更少的金钱来获得自己需要的物品。二手商品交易市场使得一些弃置物品得到了重新利用，减少了弃置物品的数量，符合绿色发展的新理念。

近年来，随着社会的发展，二手商品交易市场有了很大的发展空间。具体体现在三个方面：第一，可进行二手交易的物品种类不断增加。可进行二手交易的传统的物品有书籍、古董、珠宝等，而现代的物品则有手机、电脑、汽车等。第二，整个社会的闲置物品的数量不断增多。以手机这类物品为例，每年换新手机的人很多，闲置的旧手机也很多了。这些被闲置的旧手机很多都是还可以使用的，完全可以放到二手商品市场去卖。第三，互联网的发展使得有了线上的二手商品交易市场，使得二手商品交易的范围可大到全国甚至全世界。

现在在互联网上已经有了一些二手商品交易系统，但大多数都是基于 C/S 架构的，亦或者是基于论坛的模式的。本文正是在这样的背景之下，设计与实现了一个基于 B/S 架构的二手商品交易系统。它满足了人们在网上进行二手商品交易的需求。

1.2 技术背景

1.2.1 B/S 架构

B/S 架构即浏览器/服务器架构模式，是伴随 Web 技术发展起来的一种新型架构模式。采用 B/S 架构的系统，数据库安装在服务器上，在服务器上实现系统功能，用户电脑上只要安装一个浏览器，可以减少用户电脑的负担。这是 B/S 架构被广泛使用的一个重要原因。B/S 架构系统的整个工作流程是用户通过浏览器发送请求，服务器接收请求、处理请求、发送响应、返回结果给浏览器。采用 B/S 架构设计系统维护成本低、实用性高，可以降低总体成本[1]。

1.2.2 MySQL 数据库

MySQL 是一个源代码相对开放的管理数据库平台，MySQL 数据库平台应用的是经常应用的管理数据库语言查询平台--（SQL）管理数据库。因为 MySQL 的代码是相对开放，所以无论是什么人都能在 General Public License 的同意后下载，并通过个性

化标准对其修改。MySQL 由于可靠性，速度和适应性而广受喜爱，许多人在不需要处理事故的状况下认为 MySQL 是管理最优的方法[2]。

MySQL 与 Java 的兼容性很好，而且 MySQL 的众多优点也能够满足本系统未来发展的需要，所以本文采用 MySQL 数据库来作为本系统的数据库。

1.2.3 Java 及其相关技术

Java 语言是一种面向对象的编程语言，它拥有丰富的类库，可以很容易的实现网络连接功能和数据库存取功能。另外，它还具有稳定性高、可移植性好等优点。所以，本文所设计的系统采用 Java 语言来开发[3]。

JDBC，即 Java 数据库连接，是 JavaEE 平台下 Java 和数据库连接的规范，由类和接口组成，可以为关系型数据库提供统一的访问。进行数据库中数据查询和数据库中数据更新[3]。

Servlet，即服务连接器，是用 Java 编写的服务器端程序。Servlet 运行在服务器端，是一种按照 Servlet 标准开发的类。Servlet 可以接收浏览器发送给服务器的请求，再将服务器的处理结果返回给浏览器，事务处理功能强大。与 JSP 类似都可以生成动态 Web 内容[3]。

JSP，即 Java 服务器页面，是一种动态网页技术标准。JSP 运行在服务器端，主要用于实现 Java Web 应用程序的界面。它可以接收浏览器发送的请求，将处理结果返回给浏览器按照请求生成动态 Web 网页[3]。

2 需求分析

软件系统的开发，需求分析是主要步骤之一。需求分析过程的详细和准确是系统设计和开发成功的前提和基础。需求分析需要研发人员和用户之间充分沟通，发挥各自领域的优势密切配合，将用户需要实现的内容准确地传递给研发人员，研发人员需要了解用户行业领域的专业知识，深刻理解需求文档并进行有效的沟通，从而确定软件系统设计与开发的目标。

2.1 功能分析

本人基于日常生活中进行二手商品交易的经验，通过与二手商品交易系统的用户进行沟通交流，并且参考了市场上已存在的二手商品交易系统，提炼出了二手商品交易系统需要实现的一些核心功能，这些功能可分为三大类：账户管理类、购买业务类、销售业务类。

2.1.1 账户管理类功能

(1) 注册功能

用户可通过注册功能，可填写相关信息来在本系统中创建一个属于自己的账户。注册是是整个二手商品交易业务的开始，注册后得到的账户相当于是进入本系统的入场券。

(2) 登录功能

用户通过使用登录功能，可填写账号和密码来登录到本系统。登录功能的作用有二：一是通过账号和密码验证登录的合法性，二是让系统知道接下来的服务对象是哪个用户。

(3) 查看我的账户信息功能

用户通过使用此功能可查看自己的账户信息，这些信息包含了注册时填写的所有信息（密码除外）。

(4) 修改我的账户信息功能

用户通过使用此功能，可更改自己的账户信息，使得该账户信息与自己的实际情况相一致。

(5) 修改密码功能

用户通过修改密码功能，可修改自己账户的登录密码。

(6) 查看我的钱包信息功能

每一个账户都与一个在本系统上的虚拟钱包相关联。这个钱包用户注册时自动创建的，余额初始化为 0，支付密码初始化为登录密码。用户通过使用此功能，可查看我的钱包信息。

(7) 修改支付密码功能

用户使用此功能可修改自己的虚拟钱包的支付密码。

2.1.2 购买业务类功能

（1）搜索商品功能

用户可在搜索框输入多个关键字，使用搜索功能。搜索功能根据用户输入多个的关键字，返回与之相关的商品列表页。商品列表页提供“查看详情”的链接。

（2）查看商品详情功能

用户在商品列表页点击“查看详情”时使用该功能，该功能向用户返回商品详情页。商品详情页向用户呈现商品的所有信息，并提供“查看商品评论”和“购买”链接给用户。

（3）查看商品评论功能

用户在商品详情页中点击“查看商品评论”时使用该功能，该功能会向用户返回该商品的评论查看页，该页面会展示关于该商品的所有评论。

（4）创建订单功能

用户在商品详情页点击“购买”链接时开始使用该功能，然后该功能会返回该商品的购买订单填写页，然后用户可在此页面填写购买数量和收货地址后点击“创建订单”链接来进一步使用该功能，最后该功能会在后台创建相应的订单，并跳转到该订单的订单详情页。

（5）查看购买订单列表功能

用户在首页点击“购买订单”链接时使用该功能，该功能会向该用户返回他的购买订单列表页。购买订单列表页呈现该用户的所有购买订单的简要信息，并提供各个订单的“查看详情”链接。

（6）查看订单详情功能

用户可在购买订单列表页点击“查看详情”链接来使用该功能，该功能会返回一个订单的订单详情页。订单详情页向用户呈现该订单的详细信息，并根据订单状态向用户提供不同的链接。当订单处于“待付款”状态时，该页面向用户提供“去付款”链接；当订单处于“待收货”状态时，该页面向用户提供“确认收货”链接；当订单处于“待评价”状态时，该页面向用户提供“去评价”链接。

（7）为订单付款功能

用户在订单详情页点击“去付款”链接时使用该功能，该功能首先返回一个页面要求用户填写支付密码。用户填写支付密码后点击“确认付款”，然后该功能会在后台检查支付密码是否正确。如果支付密码不正确，那么将要求用户再次填写支付密码；如果支付密码正确，则将用户虚拟钱包的余额减去订单的总金额，再将订单所购买商品的数量减去订单的购买数量，最后把订单的状态设置为“待发货”。

（8）对订单确认收货功能

用户在订单详情页点击“确认收货”时使用该功能，然后该功能返回一个确认页面，用户再次点击“确认收货”时，该功能会将订单状态设置为“代评价”，并返回该订单的订单详情页。

（9）对订单评价功能

用户在订单详情页点击“去评价”时，该功能首先返回一个订单评论填写页，用户可在此页面填写评价后点击“提交”，然后该功能会在后台保存该订单的评价并把订单的状态设置为“已成交”，最后返回该订单的订单详情页。

2.1.3 销售业务类功能

（1）查看我的商品列表功能

用户可在首页点击“我的商品”链接来使用该功能，该功能向用户返回我的商品列表页。这个页面向用户呈现他所销售的所有商品的简要信息，并提供“查看我的商品详情”和“发布新商品”功能的入口。

（2）查看我的商品详情功能

用户在我的商品列表页中点击“查看详情”时使用该功能，该功能向用户返回我的商品详情页，该页面向用户呈现一个商品的所有信息，并提供“修改商品信息”、“查看商品评论”和“删除该商品”功能的入口。

（3）发布新商品功能

用户可在我的商品列表页点击“发布新商品”链接来使用开始该功能，该功能会首先返回商品信息填写页，用户在该页面填写好新商品的各项信息之后可点击“提交”按钮，然后该功能会在后台数据库中创建该商品，并向用户返回该商品的我的商品详情页。

（4）修改商品信息功能

用户在我的商品详情页中点击“修改商品信息”时使用该功能，然后该功能会先返回一个商品信息修改页。这个页面是一个表单页面，表单中的各项信息是原来商品的信息，用户对其进行更改后可在表单下方点击“提交更改”按钮，最后该功能会更新后台数据库中该商品的信息，并返回该商品的我的商品详情页。

（5）查看商品评论功能

这里的查看商品评论功能与购买业务类中的查看商品功能是一样的。

（6）删除商品功能

用户在我的商品详情页中点击“删除该商品”时使用该功能，该功能会先返回一个确认删除商品页，用户在此页面中点击“确认删除”后，该功能才会将后台数据库中的该商品删除并返回我的商品列表页。

（7）查看售出订单列表功能

用户在首页中点击“售出订单”时使用该功能，该功能会返回一个售出订单列表页。该页面会呈现该用户的所有售出订单的摘要信息，并提供“查看详情”链接。

（8）查看售出订单详情功能

用户在售出订单列表页中点击“查看详情”时使用该功能，该功能返回一个售出订单的订单详情页。该页面呈现该售出订单的所有信息，并根据订单状态提供不同的链接。当订单状态处于“待发货”时，提供“去发货”链接；当订单状态处于“待回复”时，提供“回复评价”链接。

（9）为订单发货功能

用户在售出订单详情页中点击“去发货”时使用该功能，该功能首先要求用户填写物流后，用户填写好订单号并提交号，该功能在订单中写入该订单号并将订单状态设置为“待收货”，然后返回原来的售出订单详情页。

2.2 业务流程分析

2.2.1 购买业务流程分析

本系统的购买商品的业务流程如图 1 所示。

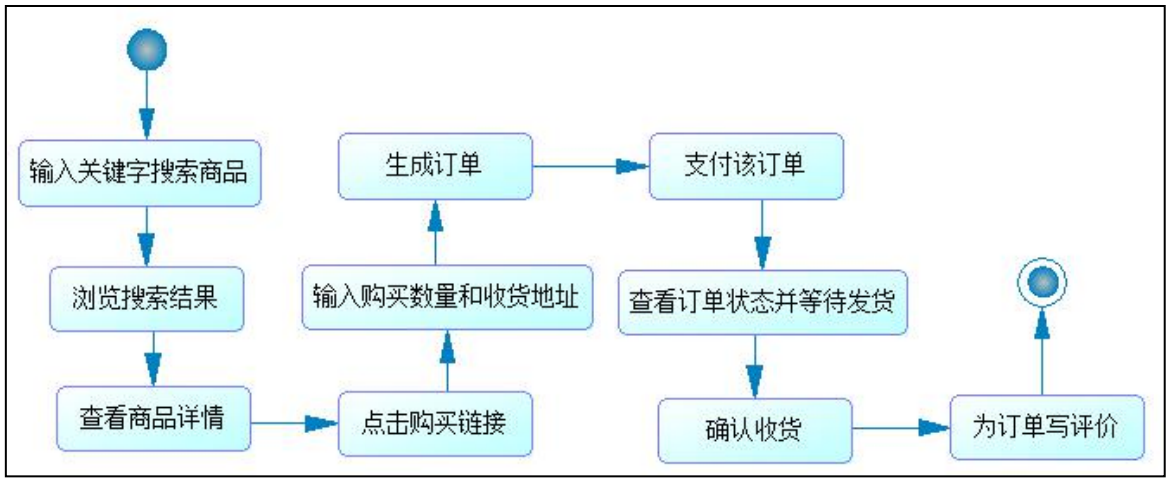

图 1 购买业务流程图

2.2.2 销售业务流程分析

本系统的销售商品的业务流程如图 2 所示。

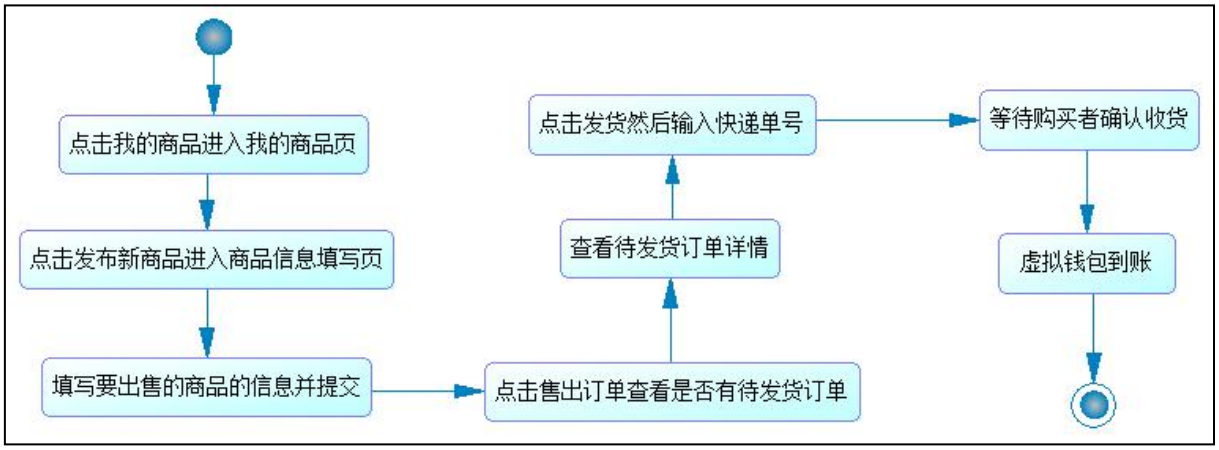

图 2 销售业务流程图

2.3 网页状态转换图

网页状态转换图是进行网站设计中必不可少的一种图，它能直观地展示一个网站由哪些网页组成，以及这些网页之间的相互联系。图 3 是本系统的网页状态转换图。

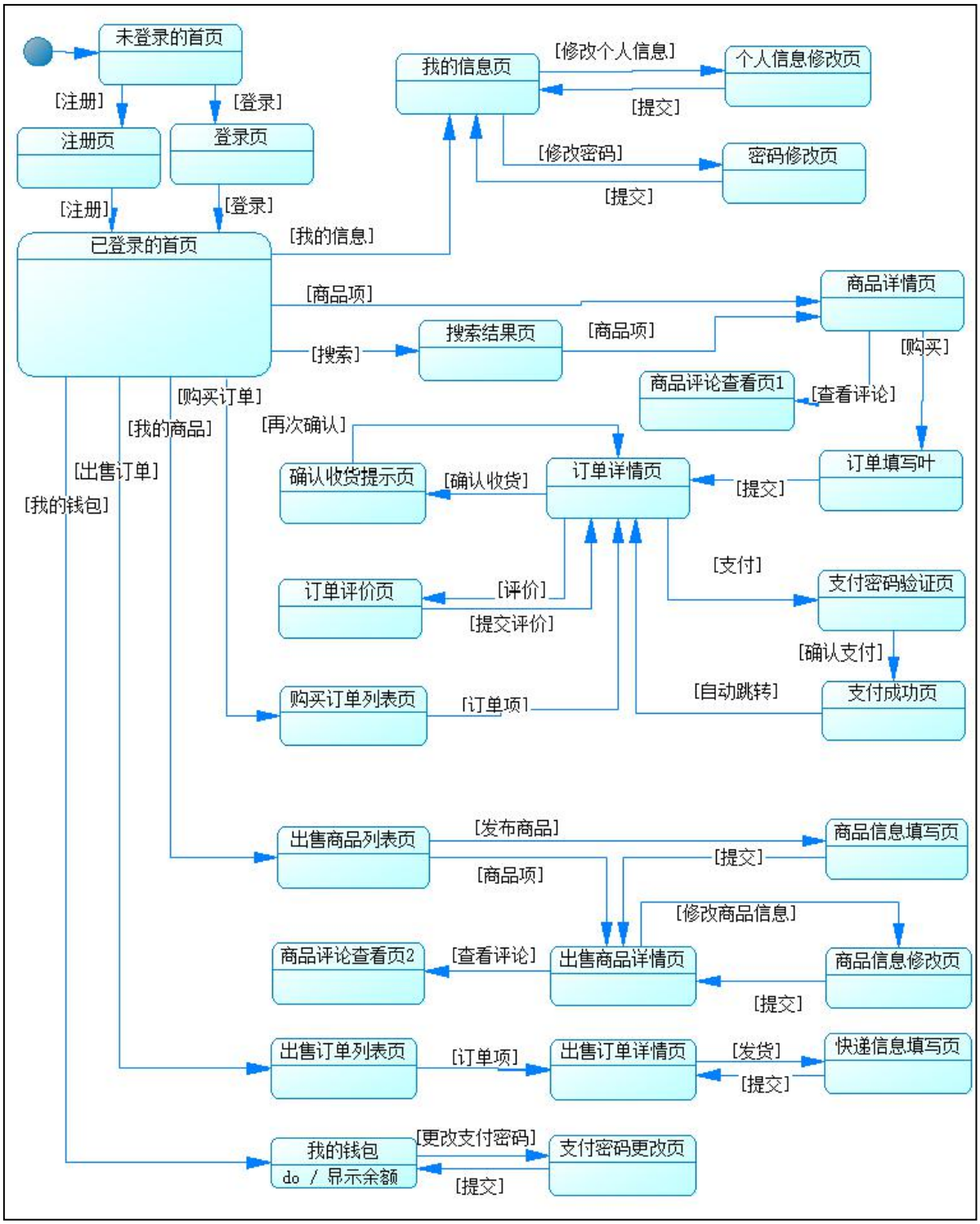

图 3 网页状态转换图

3 系统设计

3.1 系统总体结构设计

3.1.1 系统架构设计

本系统的整体架构为三层 B/S 架构。三层 B/S 架构指的是，整个系统仍采用浏览器/服务器模式，但把部署在服务器上的系统划分为三层来实现，通常把它划分为用户表现层（UIL）、业务逻辑层（BLL）、数据访问层（DAL）和实体类库（Model）。它们之间的关系如图 4 所示，图中箭头代表调用关系。

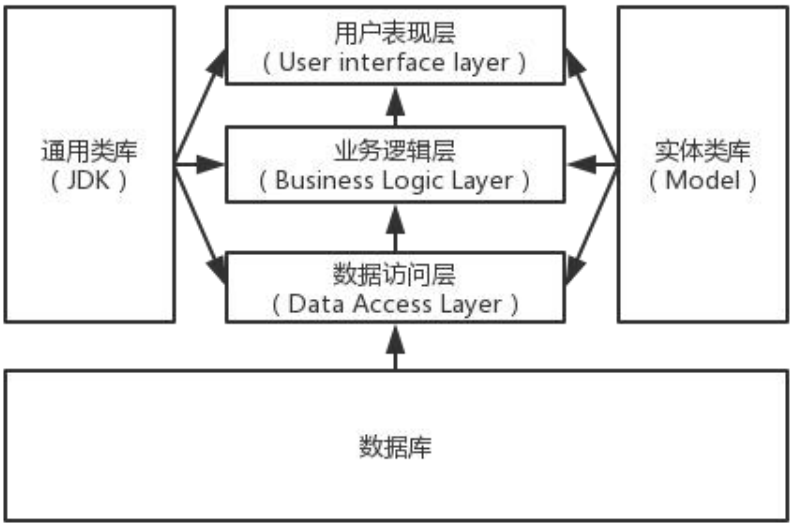

图 4 三层 B/S 架构图

在本系统中用户表现层使用 JSP 技术实现，业务逻辑层使用 Servlet 技术实现，数据访问层和实体类库使用 Java 技术实现。

3.1.2 系统模块划分

本系统在采用三层 B/S 架构的基础上，对用户表现层和业务逻辑层进行了模块划分，如图 5 所示，图中箭头代表调用关系。

本系统的工程目录结构如图 6 所示，它也体现了图 5 的结构。WebContent 目录下的所有 jsp 文件构成本系统的用户表现层，businessLogicLayer 目录下的所有文件构成本系统的业务逻辑层，dataAccessLayer 目录下的 Mysql.java 的是本系统的数据访问层，model 目录下的所有文件构成本系统的实体类库。

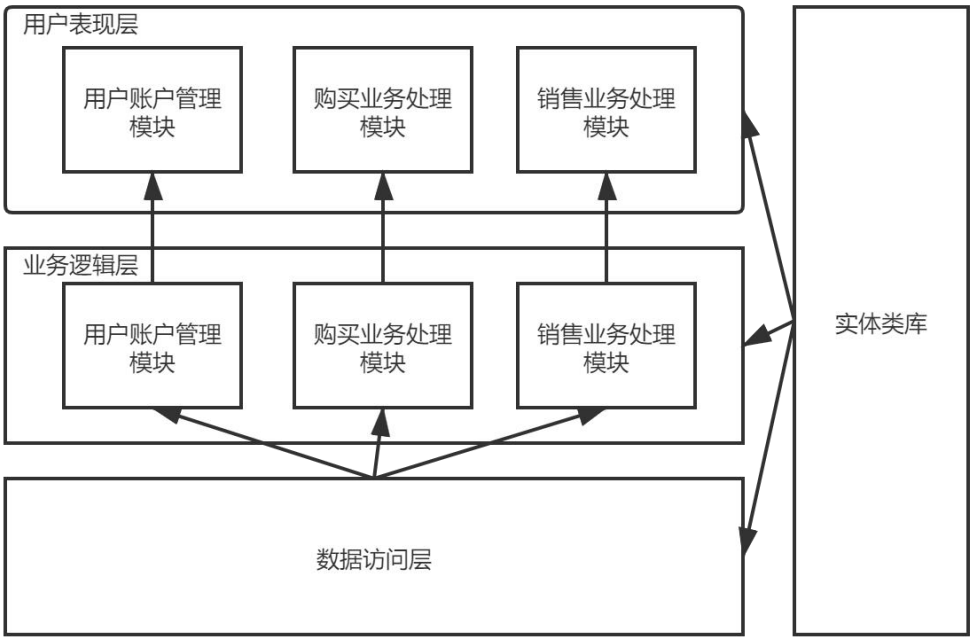

图 5 三层 B/S 架构下的系统模块划

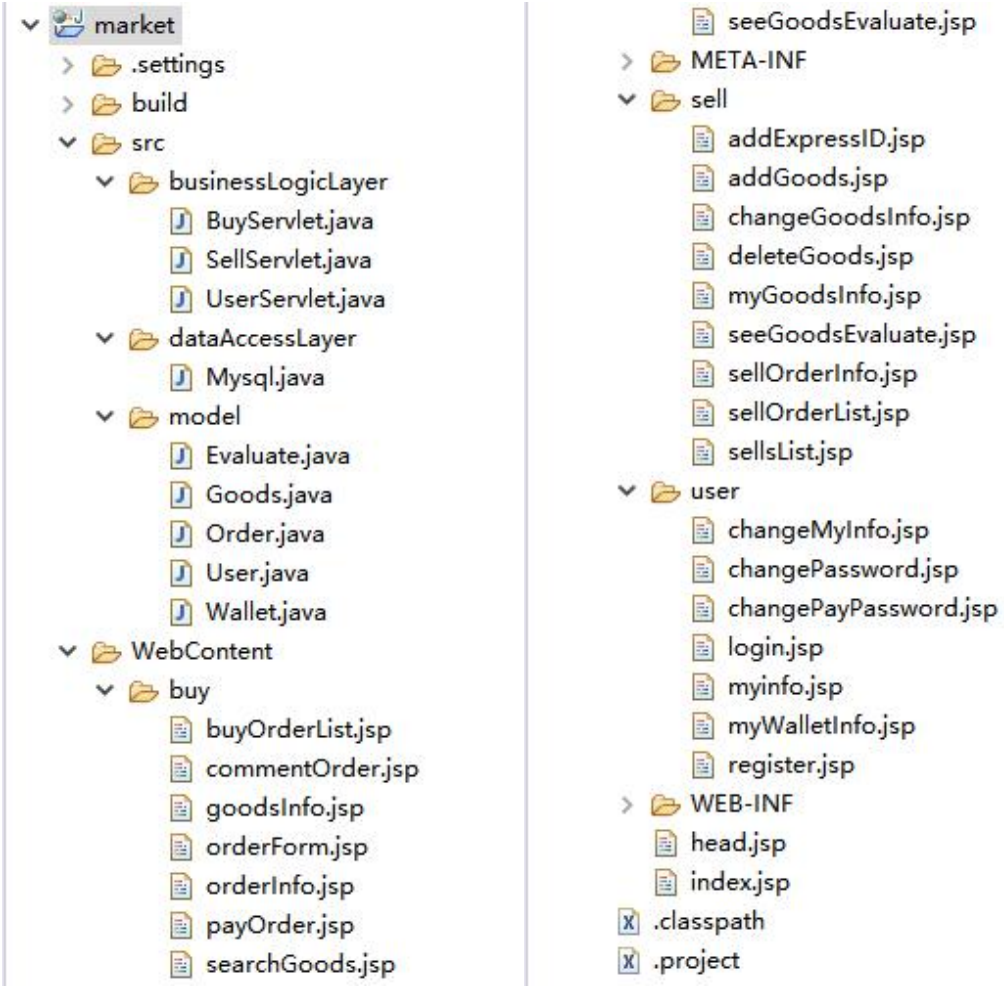

图 6 本系统的工程目录结构

3.2 系统模块详细设计

3.2.1 用户账户管理模块

用户账户管理模块在表现层有 7 个页面，这 7 个页面所产生的事务都由业务逻辑层的 UserServlet.java 来处理。而 UserServlet.java 需要使用公共的数据访问层和实体类模型来访问数据库并暂存数据。

该模块的表现层需实现的 7 个 jsp 页面分别是：注册页、登录页、我的信息页、修改我的信息页、修改密码页、我的钱包信息页、修改支付密码页。这些页面之间的关系可看图 3 的网页状态转换图。

该模块的业务逻辑层 UserServlet.java 需实现的功能有：注册功能、登录功能、查看我的账户信息功能、修改我的账户信息功能、修改密码功能、查看我的钱包信息功能、修改支付密码功能。

3.2.2 购买事务处理模块

购买事务处理模块在表现层有 9 个页面，这 9 个页面所产生的事务都由该模块的业务逻辑层 BuyServlet.java 来处理。而 BuyServlet.java 需要使用公共的数据访问层和实体类模型来访问数据库并暂存数据。

该模块的表现层需实现的 jsp 页面分别是：搜索商品结果页、商品详情页、商品评论查看页、订单填写页、订单详情页、订单支付页、订单评论页、确认收货页、购买订单列表页。这些页面之间的关系可看图 3 的网页状态转换图。

该模块的业务逻辑层 BuyServlet.java 需实现的功能有：搜索商品功能、查看商品详情功能、查看商品评论功能、创建订单功能、查看购买订单列表功能、查看订单详情功能、为订单付款功能、对订单确认收货功能、对订单评价功能。

3.2.3 销售事务处理模块

销售事务处理模块在表现层有 9 个页面，这 9 个页面所产生的事务都由该模块的业务逻辑层 SellServlet.java 来处理。而 SellServlet.java 需要使用公共的数据访问层和实体类模型来访问数据库并暂存数据。

该模块的表现层需实现的 jsp 页面分别是：我的商品列表页、出售商品详情页、新商品发布页、商品删除页、商品评论查看页、商品信息修改页、出售订单列表页、出售订单详情页、快递信息填写页。这些页面之间的关系可看图 3 的网页状态转换图。

该模块的业务逻辑层 SellServlet.java 需实现的功能有：查看我的商品列表功能、查看我的商品详情功能、发布新商品功能、修改商品信息功能、查看商品评论功能、删除商品功能、查看售出订单列表功能、查看售出订单详情功能、为订单发货功能

3.2.4 数据访问层和实体类库

数据访问层和实体类库是上面的三大模块都要使用的公共资源。数据访问层Mysql.java里的每一个方法代表一种数据库操作，比如：查询一个表的某些数据、往一个表中插入一些新数据、修改一个表中的某些数据等。实体类库里的实体类的个数与数据库里表的个数相同，其中每一个实体类的结构对应数据库中一张表的结构。

4 系统数据库设计

4.1 概念实体设计

根据二手商品交易系统的建设目标的分析，需要有以下的数据库实体：

(1) 用户实体：

用户实体用来存储用户的账号信息，它包括用户的手机号、昵称、密码、注册时间和收货地址这些字段。

(2) 用户钱包实体：

用户钱包实体用来存储用户的虚拟钱包的信息，它包括手机号、余额和支付密码。

(3) 商品实体：

商品实体用来存储商品的信息，它包括商品编号、拥有者编号、标题、标签、数量、价格、发布时间、发货地址和详细描述这些字段。

(4) 订单实体：

订单实体用来存储订单信息。当有一个用户购买另一个用户的商品时，就会产生一个订单。它包含订单编号、购买者编号、销售者编号、商品编号、购买数量、购买价格、订单总金额、订单状态、创建时间、结束时间、收货地址和快递单号这些字段。

(5) 评价实体：

评价实体用来存储评价信息。评价是对订单进行的，但会关联到商品。一个评价有这些字段：订单编号、商品编号、用户编号、评论时间、评分、评论。

4.2 概念数据模型

本系统的概念数据模型如图7所示。其中在实体集的名称后面标注为“（弱）”的实体集属于弱实体集。因为它们没有足够的属性以形成主码，所以称之为弱实体集。其它的有主码的实体集则称之为强实体集。一个弱实体集需要依赖于一个强实体集才有意义。在图7中，钱包（弱）实体集依赖于用户实体集，表示每一个用户拥有一个钱包；评论（弱）实体集依赖于订单实体集，表示每一个订单有一个或零个评论。

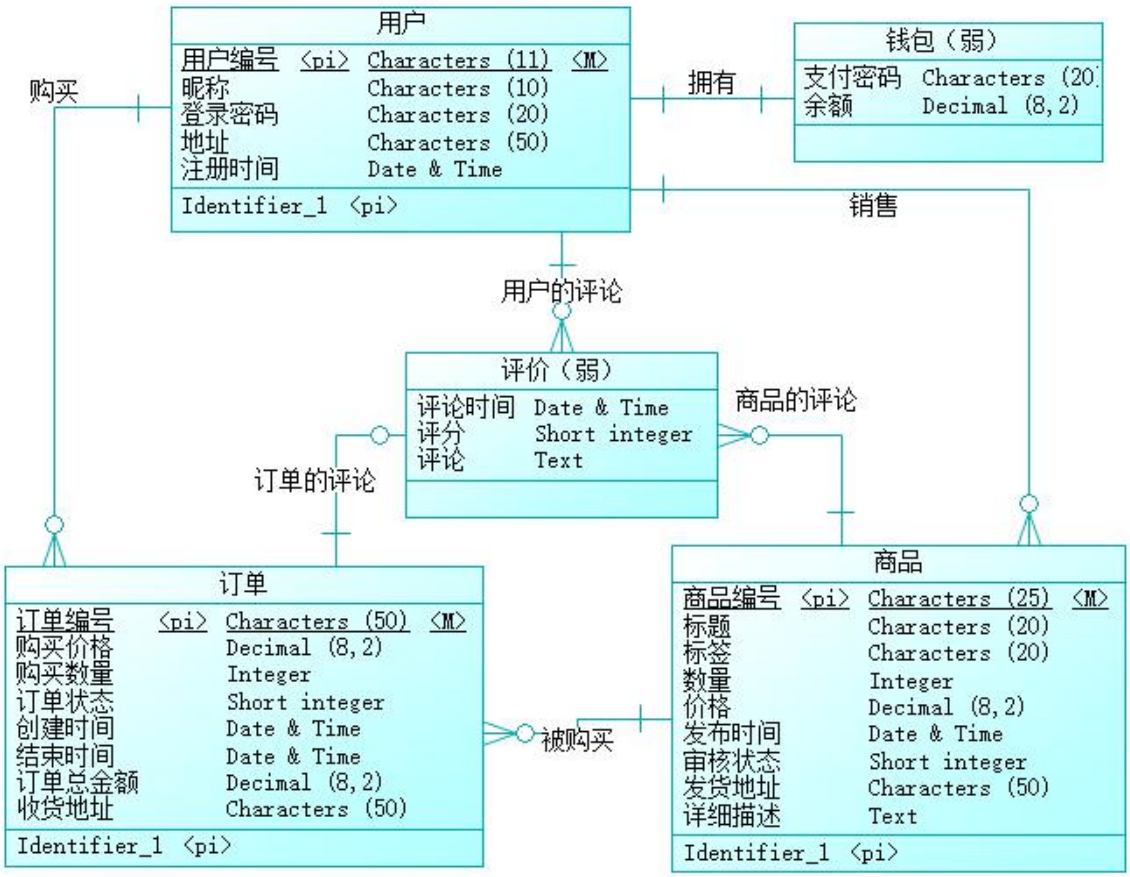

图 7 概念数据模型

4.3 数据表设计

1) 名称：用户表

数据表名：user

数据表用途：保存用户信息

数据来源：在用户注册时录入，在用户登录后可修改

表 1 用户表

字段名称	字段类型	字段意义	是否非空	是否键值	默认值	备注
userID	char(11)	用户编号	是	主键	无	手机号
nickname	char(10)	昵称	否		无	
password	char(20)	登录密码	是		无	
address	char(50)	地址	否		无	
registerTime	datetime	注册时间	否		无	

2) 名称：用户钱包表

数据表名：wallet

数据表用途：保存虚拟钱包信息

数据来源：在用户注册账号时由触发器自动创建，在发生交易时更改

表 2 用户钱包表

字段名称	字段类型	字段意义	是否非空	是否键值	默认值	备注
userID	char(11)	用户编号	是	主键 外键	无	
balance	decimal(8,2)	余额	是		0	
payPassword	char(20)	支付密码	是		无	

3) 名称：商品表

数据表名：goods

数据表用途：保存商品的各种信息

数据来源：用户发布商品时录入，登录后可更改属于自己的商品项

表 3 商品表

字段名称	字段类型	字段意义	是否非空	是否键值	默认值	备注
goodsID	char(25)	商品编号	是	主键	无	
ownerID	char(11)	拥有者编号	是	外键	无	参照用户表
title	char(20)	标题	否		无	
label	char(20)	标签	否		无	
number	int	数量	否		无	
price	decimal(8,2)	价格	否		无	
submitTime	datetime	发布时间	否		无	
fromAddress	char(50)	发货地址	否		无	
detail	text	详细描述	否		无	

4) 名称：订单表

数据表名：order

数据表用途：保存订单信息

数据来源：在用户购买商品时创建

表 4 订单表

字段名称	字段类型	字段意义	是否非空	是否键值	默认值	备注
orderID	char(50)	订单编号	是	主键	无	
buyerID	char(11)	购买者编号	是	外键	无	参照用户表
sellerID	char(11)	销售者编号	是	外键	无	参照用户表
goodsID	char(25)	商品编号	是	外键	无	参照商品表
buyNumber	int	购买数量	否		无	
buyPrice	decimal(8,2)	购买价格	否		无	
totalMoney	decimal(8,2)	订单总金额	否		无	

orderStatus	smallint	订单状态	否		无	
startTime	datetime	创建时间	否		无	
endTime	datetime	成交时间	否		无	
toAddress	char(50)	收货地址	否		无	
expressID	char(50)	快递单号	否		无	

5) 名称：评价表

数据表名：evaluate

数据表用途：保存评价信息

数据来源：在用户对订单进行评价时录入

表 5 评价表

字段名称	字段类型	字段意义	是否非空	是否键值	默认值	备注
orderID	char(50)	订单编号	是	主键 外键	无	参照订单表
goodsID	char(25)	商品编号	是	外键	无	参照商品表
userID	char(11)	评论者编号	是	外键	无	参照用户表
cmTime	datetime	评论时间	否		无	
stars	smallint	评分	否		无	
comment	text	评论	否		无	

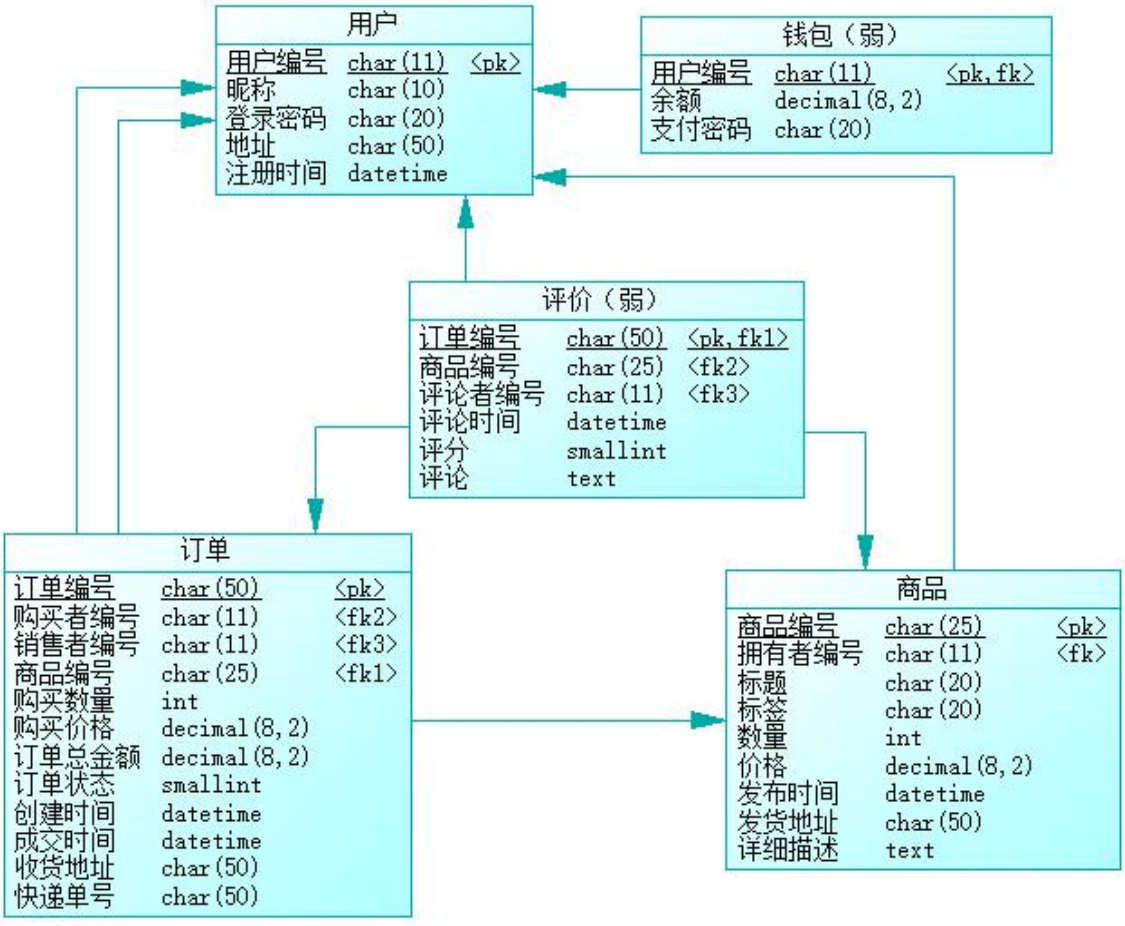

图 8 物理数据模型

4.4 物理数据模型

本系统的物理数据模型如图 8 所示。

在本系统的物理数据模型中，还定义了一个触发器，其代码如下：

```
create trigger createWallet after insert
on `user` for each row
begin
    insert into wallet values(new.userID, 0, new.password);
end;
```

这个触发器的作用是，当有新用户注册成功时，自动为其创建一个余额为 0 的钱包。

5 系统关键代码设计

5.1 数据访问层关键代码

这一层的代码大多数都是对数据库进行增删改查的代码，不同表的增删改查的代码是相似的。故下面只介绍对商品表进行增删改查的代码。但再那之前，有必要先介绍一个能执行任意 SQL 语句的函数，因为在本系统中所有的 SQL 语句最终都将传递给这个函数执行。5.1.1 是这个函数的介绍。

5.1.1 执行 SQL 语句的函数

```
//Temp 结构
private static class Temp {
    CachedRowSet crs;    //结果集
    int updateCount;       //更新计数
}

//执行一条传入的 sql，并返回一个 Temp 结构的数据
private static Temp execute(String sql) {
    Temp temp = new Temp();
    Connection conn = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Properties info = new Properties();
        info.setProperty("user", "root");
        info.setProperty("password", "8569");
        info.setProperty("useUnicode", "true");
        info.setProperty("characterEncoding", "utf8");
```

```
            conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/market", info);
            Statement stat = conn.createStatement();
            ResultSet rs = null;
            CachedRowSet crs = null;

            boolean isResult = stat.execute(sql);
            if(isResult) {
                rs = stat.getResultSet();
                RowSetFactory rsft = RowSetProvider.newFactory();
                crs = rsft.createCachedRowSet();
                crs.populate(rs);
                temp.crs = crs;
                temp.updateCount = -1;
            } else {
                temp.updateCount = stat.getUpdateCount();
                temp.crs = null;
            }
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            try {
                if(conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return temp;
    }
```

5.1.2 插入商品的函数

```
    public static boolean insertGoods(Goods goods) {
        String sql = "INSERT INTO `goods` VALUES ('"
                    +goods.getGoodsID()+"', '"
                    +goods.getOwnerID()+"', '"
                    +goods.getTitle()+"', '"
                    +goods.getLabel()+"', '"
                    +goods.getNumber()+"', '"
                    +goods.getPrice()+"', '"
                    +goods.getSubmitTime()+"', '"
                    +goods.getFromAddress()+"', '"
                    +goods.getDetail()+"');";
        int updateCount = Mysql.execute(sql).updateCount;
```

```
        if(updateCount == 1) return true;
        else return false;
    }
```

5.1.3 删除商品的函数

```
    public static boolean deleteGoods(String ID) {
        String sql = "DELETE FROM goods\r\n"
                + "WHERE goodsID='"+ID+"';";
        int updateCount = Mysql.execute(sql).updateCount;
        if(updateCount == 1) return true;
        else return false;
    }
```

5.1.4 修改商品的函数

```
public static boolean updateGoodsByID(String ID, Goods goods) {
        String sql = "UPDATE goods\r\n"
                + "SET\r\n"
                + "goodsID='"+goods.getGoodsID()+"',"
                + "ownerID='"+goods.getOwnerID()+"',"
                + "title='"+goods.getTitle()+"',"
                + "label='"+goods.getLabel()+"',"
                + "number='"+goods.getNumber()+"',"
                + "price='"+goods.getPrice()+"',"
                + "submitTime='"+goods.getSubmitTime()+"',"
                + "fromAddress='"+goods.getFromAddress()+"',"
                + "detail='"+goods.getDetail()+"'\r\n"
                + "WHERE goodsID='"+ID+"'";
        int n = Mysql.execute(sql).updateCount;
        if(n==1) return true;
        else return false;
    }
```

5.1.5 精准查询商品的函数

```
    public static Goods getGoodsByID(String ID) {
        Goods goods = null;
        String sql = "SELECT * FROM goods WHERE goodsID = '"+ID+"'";
        CachedRowSet crs = Mysql.execute(sql).crs;
        try {
            if(crs.next()) {
                String goodsID = crs.getString(1);
                String ownerID = crs.getString(2);
                String title = crs.getString(3);
                String label = crs.getString(4);
                int number = crs.getInt(5);
```

```
                double price = crs.getDouble(6);
                Timestamp submitTime = crs.getTimestamp(7);
                String fromAddress = crs.getString(8);
                String detail = crs.getString(9);
                goods = new Goods(goodsID, ownerID, title, label, number, price, submitTime,
fromAddress, detail);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return goods;
    }
```

5.1.6 模糊查询商品的函数

```
    public static Goods[] searchGoods(String keyword, int page) { //page>=0
        final int PAGESIZE = 10;
        String[] keywords = keyword.split("\\s+"); //提取出多个关键字
        int length = keywords.length;
        CachedRowSet crs = null;

        if(length>=1) {
            //以下，用交查询拼接多个 SQL 查询语句
            StringBuffer sql = new StringBuffer();
            for(int i=0; i<length; i++) {
                sql.append("(SELECT goodsID, title, label, price\r\n"
                        + "FROM goods\r\n"
                        + "WHERE title LIKE '%"+keywords[i]
                        +"%' OR label LIKE '%"+keywords[i]+"%')\r\n"
                        + "AS a" + i + "\r\n"
                        + "NATURAL JOIN\r\n");
            }
            sql.delete(sql.length()-14, sql.length());
            String sql2 = "SELECT goodsID, title, label, price\r\n"
                        + "FROM (\r\n" + sql.toString() + ")\r\n"
                        + "ORDER BY title LIMIT "+page*PAGESIZE+", "+PAGESIZE;
            crs = Mysql.execute(sql2).crs;
        } else { //如果没有关键字
            String sql = "SELECT goodsID, title, label, price\r\n"
                    + "FROM goods\r\n"
                    + "ORDER BY name\r\n"
                    + "LIMIT "+page*PAGESIZE+", "+PAGESIZE;
            crs = Mysql.execute(sql).crs;
        }
        Goods[] goods = new Goods[PAGESIZE];
```

```
        try {
            for(int i=0; crs.next(); i++) {
                goods[i] = new Goods();
                goods[i].setGoodsID(crs.getString(1));
                goods[i].setTitle(crs.getString(2));
                goods[i].setLabel(crs.getString(3));
                goods[i].setPrice(crs.getDouble(4));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return goods;
    }
    /*注释：
    假如上面的函数的 keyword 参数为“甲 乙”，page 参数为 0；那么该函数将拼接出如下的 SQL 语句来给 5.1.1 小节的 execute 函数执行。
    SELECT goodsID, title, label, price
    FROM (
    (SELECT goodsID, title, label, price
    FROM goods
    WHERE title LIKE '%甲%' OR label LIKE '%甲%')
    AS a0
    NATURAL JOIN
    (SELECT goodsID, title, label, price
    FROM goods
    WHERE title LIKE '%乙%' OR label LIKE '%乙%')
    AS a1
    )
    ORDER BY title LIMIT 0, 10
    */
```

5.2 业务逻辑层关键代码

业务逻辑层的代码的主要作用是：接收来自浏览器端的 HTTP 请求，并从中提取出有用的参数，在完成某种业务逻辑之后，使用这些参数去调用数据访问层的函数，然后再把数据访问层返回的结果存储在请求域（request）或会话域（session）中，最后把 HTTP 请求转发到用户表现层。下面将列举一些有关商品的一些业务逻辑层的代码作为例子。

5.2.1 搜索商品的函数

```
public void searchGoods(HttpServletRequest request, HttpServletResponse response) {
    //先从浏览器请求信息中获得相关数据
    String searchText = (String)request.getParameter("searchText");
```

```
        int page = Integer.parseInt(request.getParameter("page"));
        //然后使用数据访问层获得 goods 对象
        Goods[] goodss = Mysql.searchGoods(searchText, page);
        //最后设置一些属性，再转发到用户表现层
        request.setAttribute("searchText", searchText);
        request.setAttribute("page", page);
        request.setAttribute("goodss", goodss);
        try {
        request.getRequestDispatcher("buy/searchGoods.jsp").forward(request, response);
        } catch (ServletException | IOException e) {
            e.printStackTrace();
        }
    }
```

5.2.2 查看商品详情的函数

```
    public void myGoodsInfo(HttpServletRequest request, HttpServletResponse response) {
        String goodsID = request.getParameter("goodsID");
        Goods sellGoods = null;
        if(goodsID!=null) {
            sellGoods = Mysql.getGoodsByID(goodsID);
        }
        HttpSession session = request.getSession();
        session.setAttribute("sellsGoods", sellGoods);
        try {
        request.getRequestDispatcher("sell/myGoodsInfo.jsp").forward(request, response);
        } catch (ServletException | IOException e) {
            e.printStackTrace();
        }
    }
```

5.2.3 更改商品的函数

```
    public static void changeGoodsInfo(HttpServletRequest request, HttpServletResponse response) {
        String title = request.getParameter("title");
        if(title == null) {
            try {
    request.getRequestDispatcher("sell/changeGoodsInfo.jsp").forward(request, response);
            } catch (ServletException | IOException e) {
                e.printStackTrace();
            }
        } else {
            String label = request.getParameter("label");
            int number = Integer.parseInt(request.getParameter("number"));
            double price = Double.parseDouble(request.getParameter("price"));
            String fromAddress = request.getParameter("fromAddress");
```

```
                String detail = request.getParameter("detail");

                HttpSession session = request.getSession();
                Goods sellsGoods = (Goods)session.getAttribute("sellsGoods");
                sellsGoods.setTitle(title);
                sellsGoods.setLabel(label);
                sellsGoods.setNumber(number);
                sellsGoods.setPrice(price);
                sellsGoods.setFromAddress(fromAddress);
                sellsGoods.setDetail(detail);
                Mysql.updateGoodsByID(sellsGoods.getGoodsID(), sellsGoods);
                try {
response.sendRedirect("SellServlet?type=myGoodsInfo&goodsID="+sellsGoods.getGoodsID()+"");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
```

5.2.4 发布新商品的函数

```
    /*添加商品成功，则重定向到出售商品详情页，否则将通过请求转发携带错误信息返回原页面；
*/
    public void addGoods(HttpServletRequest request, HttpServletResponse response) {
        String hasData = request.getParameter("title");
        if(hasData == null || hasData.equals("")) {
            //请求域中没有新商品数据，转发请求到商品信息填写页
            try {
            request.getRequestDispatcher("sell/addGoods.jsp").forward(request, response);
            } catch (ServletException | IOException e) {
                e.printStackTrace();
            }
        } else {
            //请求域中有新商品数据，开始添加商品
            HttpSession session = request.getSession();
            User user = (User)session.getAttribute("user");

            Date date = new Date();
            String goodsID = user.getUserID() + date.getTime();
            String ownerID = user.getUserID();
            String title = request.getParameter("title");
            String label = request.getParameter("label");
            int number = Integer.parseInt(request.getParameter("number"));
            double price = Double.parseDouble(request.getParameter("price"));
            Timestamp submitTime = new Timestamp(date.getTime());
```

```
                String fromAddress = request.getParameter("fromAddress");
                String detail = request.getParameter("detail");

                Goods goods = new Goods(goodsID, ownerID, title, label, number, price, submitTime,
fromAddress, detail);
                if(Mysql.insertGoods(goods)) {
                    try {
            response.sendRedirect("SellServlet?type=myGoodsInfo&goodsID="+goodsID+"");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                } else {
                    request.setAttribute("message", "添加出错，请重试");
                    try {
            request.getRequestDispatcher("sell/addGoods.jsp").forward(request, response);
                    } catch (ServletException | IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
```

5.3 用户表现层关键代码

用户表现层的代码的主要作用是：接收由业务逻辑层转发的 HTTP 请求，并从中提取出数据，然后把数据放到网页中，最后向浏览器端返回网页。下面将列举一些有代表性的页面代码。

5.3.1 网页头部

在本系统中，几乎每一个网页都会通过静态包含来加载一段网页头部的代码。这段代码编写了一个位于网页头部的导航栏，这段代码位于 head.jsp 中，其代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="model.User"%>
<%-- 该文件不作为单独的网页页面，而是作为每个网页的头部，被其它网页包含 --%>
    <style>
        a:link {
            color: blue;
        }
        a:visited {
            color: blue;
        }
    </style>
```

```
<div id="head">
    <h2>基于 B/S 架构的二手商品交易系统</h2>
    <%
        User user2 = null;
        String nickname = null;
        user2 = (User)session.getAttribute("user");
        if(user2 != null) nickname = user2.getNickname();
        if(nickname != null) {
    %>
    <form name="seach" action="BuyServlet" method="get">
        <table>
            <tr>
                <td><a href="/market/">本站首页 </a></td>
                <td><a href="/market/UserServlet?type=myInfo"> 你好，<%= nickname %> </a></td>
                <td><a href="/market/UserServlet?type=myWalletInfo"> 我的钱包 </a></td>
                <td><a href="/market/SellServlet?type=sellsList&page=0"> 我的商品 </a></td>
                <td><a href="/market/BuyServlet?type=buyOrderList&page=0"> 购买订单 </a></td>
                <td><a href="/market/SellServlet?type=sellOrderList&page=0"> 售出订单 </a></td>
                <td><input type="text" name="searchText" placeholder="输入多个关键字以搜索商品，用空
格隔开" size=38><input type="submit" value="搜索"></td>
            </tr>
        </table>
        <input type="hidden" name="page" value="0">
        <input type="hidden" name="type" value="searchGoods">
    </form>
    <%
        } else {
    %>
    <form name="seach" action="BuyServlet" method="get">
        <table>
            <tr>
                <td><a href="/market/UserServlet?type=login"> 登录 </a></td>
                <td><a href="/market/UserServlet?type=register"> 注册 </a></td>
                <td> <input type="text" name="searchText" placeholder="输入多个关键字以搜索商品，用
空格隔开" size=38><input type="submit" value="搜索"></td>
            </tr>
        </table>
        <input type="hidden" name="page" value="0">
        <input type="hidden" name="type" value="searchGoods">
    </form>
    <% } %>
    <hr>
</div>
```

5.3.2 商品信息填写页

下面以商品信息填写页为例，介绍需要用户输入数据的一种页面。商品信息填写页（addGoods.jsp）中的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>发布新商品页</title>
</head>
<body>
    <%@ include file="/head.jsp" %>
    <form id="addGoods" action="SellServlet?type=addGoods" method="post">
    <table>
        <tr>
            <td>商品标题：</td>
            <td><input type="text" name="title"></td>
        </tr>
        <tr>
            <td>商品标签：</td>
            <td><input type="text" name="label"></td>
        </tr>
        <tr>
            <td>商品数量：</td>
            <td><input type="number" name="number"></td>
        </tr>
        <tr>
            <td>商品价格：</td>
            <td><input type="number" step="0.01" name="price"></td>
        </tr>
        <tr>
            <td>发货地址：</td>
            <td><input type="text" name="fromAddress"></td>
        </tr>
        <tr>
            <td>详细描述：</td>
            <td></td>
        </tr>
    </table>
    <textarea rows="10" cols="60" name="detail" form="addGoods"></textarea><br>
    <input type="submit" value="发布">
    </form>
</body>
</html>
```

5.3.3 搜索商品结果页

下面以搜索商品结构页为例，介绍一种用于展示数据的可翻页的页面。搜索商品结果页（searchGoods.jsp）的代码如下：

```
<%@page import="model.Goods" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>商品搜索页</title>
</head>
<body>
    <%@ include file="/head.jsp" %>
    <!-- searchGoods.jsp 用于显示关键词搜索商品的结果  -->
    <table>
        <tr>
            <td>商品编号</td>
            <td>商品标题</td>
            <td>商品标签</td>
            <td>商品价格</td>
        </tr>
    <%
    String searchText = (String)request.getAttribute("searchText");
    int page3 = (Integer)request.getAttribute("page");
    Goods[] goodss = (Goods[])request.getAttribute("goodss");
    for(int i=0; i<10 && goodss[i]!=null; i++) {
    %>
        <tr>
            <td><%= goodss[i].getGoodsID() %></td>
            <td><%= goodss[i].getTitle() %></td>
            <td><%= goodss[i].getLabel() %></td>
            <td><%= goodss[i].getPrice() %></td>
            <td><a href="BuyServlet?type=goodsInfo&goodsID=<%= goodss[i].getGoodsID() %>">
查看详情</a></td>
        </tr>
    <%
    }
    %>
    </table>
    <table>
        <tr>
            <td><a href="BuyServlet?type=searchGoods&searchText=<%=
```

```
searchText %>&page=<%= (page3-1)>=0? page3-1:0 %>">上一页</a></td>
                <td>当前页：<%= page3 %></td>
                <td><a href="BuyServlet?type=searchGoods&searchText=<%=
searchText %>&page=<%= (page3+1)>=0? page3+1:0 %>">下一页</a></td>
            </tr>
        </table>
</body>
</html>
```

6 系统的安装与运行

6.1 运行环境

操作系统：Window 10

JDK: jdk1.8.0_271

Web 服务器：apache-tomcat-8.5.59

数据库：mysql-5.7.31-winx64

6.2 系统安装

本节将介绍如何在 6.1 节描述的运行环境中安装本系统。本系统使用 Eclipse jee(4.7.3a)开发，并使用 Eclipse 将本项目打包成了一个 war 文件“market.war”。使用这个文件可以很方便的安装本系统。下面介绍安装本系统的详细步骤：

6.2.1 安装前题

① 已安装好 JDK、Tomcat、MySQL，且 MySQL 的 root 用户的密码设置为 8569。

② 拥有以下文件：

market.war（本系统的所有代码的打包文件）

market.sql（本系统所用数据库的备份文件）

mysql-connector-java-5.1.48-bin.jar（MySQL 的 JDBC 驱动文件）

6.2.2 安装本系统的数据库

① 打开系统命令行

② 切换到 market.sql 所在的目录，例如：

cd C:\Users\PLANET\Desktop\market

③ 使用 mysql 命令执行 market.sql 文件，例如：

mysql -uroot -p8569 < market.sql

④ 若在第三步时出现以下页面（有一个警告信息，但没有错误信息），则表示本

系统数据库安装成功。

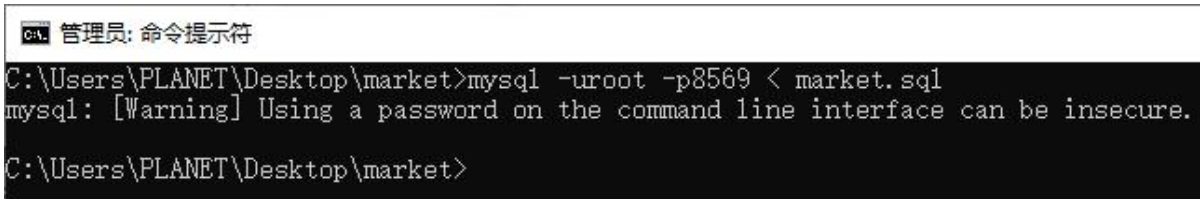

图 9 数据库安装成功图

6.2.3 安装 MySQL 驱动

① 将 mysql-connector-java-5.1.48-bin.jar 文件复制到 tomcat 的安装目录的 lib 目录下，如下图。

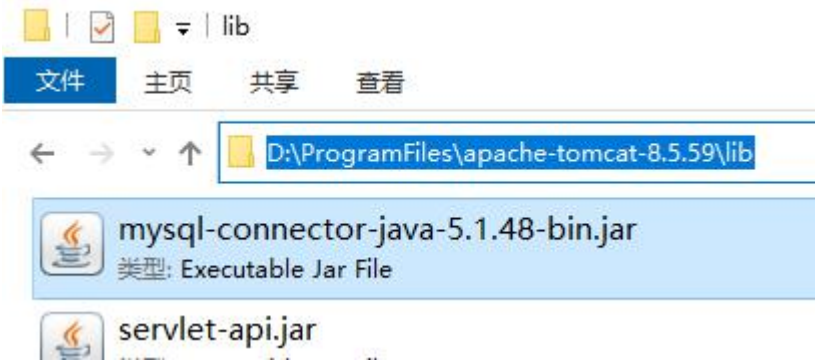

图 10 MySQL 驱动安装图

② 驱动安装完成。

6.2.4 安装本系统

① 将 market.war 文件复制到 tomcat 安装目录的 webapps 目录下，如下图。

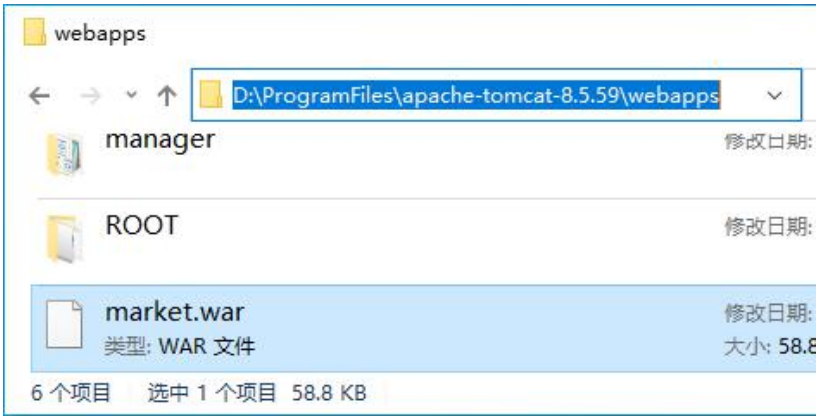

图 11 本系统安装图

② 这样，本系统就已经在 tomcat 上部署完毕了，接下来只要启动 tomcat 就能浏览器访问该系统了。

6.3 运行结果与测试结论

6.3.1 系统启动方法

首先，切换到 tomcat 安装目录的 bin 目录下，点击 startup.bat 启动 tomcat。出现下面的输出代表 tomcat 已经成功启动。

图 12 Tomcat 启动成功图

然后，在浏览器输入“http://127.0.0.1:8080/market/”即可访问本系统的首页。本系统的首页如图 13 所示。

图 13 系统首页图

6.3.2 综合测试

综合测试主要测试能否完成商品的二手商品交易流程。

测试方案：用两个不同的浏览器登录两个不同的账号，分别称为甲、乙。用甲扮演购买者，购买乙发布的商品。如果能够完成一次交易且系统不出现错误，则测试通过；否则，测试不通过。

下面开始测试：

图 14 用户信息（甲）

图 15 用户信息（乙）

图 16 甲浏览乙发布的商品

图 17 甲填写购买订单

图 18 甲创建了一个购买订单（待付款）

在图 18 的页面付款后，甲的钱包的余额会减少 129.0 元，这里不作展示。

用另一个浏览器，用乙扮演销售者的角色，对甲购买的订单发货

图 19 乙有一个待发货订单

图 20 乙已将订单发货

最后，用甲的账号，对购买订单进行确认收货

图 21 甲可进行确认收货

确认收货后，乙的钱包余额会增加 129.0 元，但这里不作展示。最后在图 22 所示页面进行订单评价。这样，一个商品的交易就完成了。

图 22 可对订单进行评价

6.3.3 测试结论

本系统能够正常运行，能够正常使用在 2.1 节的功能分析中描述的所有功能，能够完成在 2.2 节业务流程分析中的所有业务流程，能够为用户提供一个简单的线上二手商品交易平台。

本系统的缺点：安全性低、用户界面不太直观。例如，订单详情页只有商品的编号而没有商品的标题等信息。这就导致用户在查看订单时，不知道这个订单是什么商品的订单。

本系统的优点：界面简洁、响应迅速、结构完整、架构清析。

7 结束语

本文目的是针对二手商品交易的流程，设计一个 B/S 架构的二手商品交易平台，满足人们的二手商品交易需求，实现便捷的线上二手商品交易。开发建设一套完整的二手商品交易系统不是一蹴而就的，而且这是本人第一次尝试系统性设计，故本文设计的二手商品交易系统只实现了核心的业务流程，在很多方面还有完善空间。

在本系统的设计与实现过程中，出现过许多大大小小的问题，但后来都一一克服了。每当遇到一个问题，我都先试着独立解决，然后再去查资料和问老师。这期间，通过理论与实践结合，我学会了很多理论知识，也学会了很多实践知识，体会到了学习的乐趣。

感谢我的指导老师罗勇老师。有了罗老师在论文选题以及论文写作过程中的指导和督促，我才完成了本篇论文。

参 考 文 献

[1] 杨秀荣，任姚鹏. 基于 B/S 的办公自动化系统的研究与设计[J]. 廊坊师范学院学报(自然科学版)，2014，14(01)：36-38.

[2] 王丽娟,吴东明. 基于 MySQL 数据库实施完整性约束的研究[J]. 科技创新与应用,2019(02):72-73.

[3] 王佳瑶. 基于 B/S 架构的税务局办公自动化系统设计[D].沈阳工业大学,2020.

[4] 孙卫琴. Tomcat 与 Java Web 开发技术详解.第 2 版[M]. 电子工业出版社, 2008.