|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学院logo | 学院校名文字 | | | 计算机科学系logo |
|  |  | **课程大作业** |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **课程代号** | **：** | BK2240002 | | **课程名称** | **：** | Java Web应用程序开发 | | **学生姓名** | **：** | 林怡 | | **学号** | **：** | 20551120028 | | **班级** | **：** | 20大数据 | | **指导教师** | **：** | 吕绪洋 | | **提交日期** | **：** | 2022年 6 月 24 日 | | **综合得分** | **：** |  | |

**一、大作业要求**

1、创建一个站点，以自己姓名拼音（完整拼音，不要用缩写）+学号命名

2、写出完整的程序；

3、在关键语句后写上详尽的注释；

4、写上或粘贴上运行结果；

5、提交实验报告和程序代码压缩包，压缩包名与项目名相同；

6、每位学生必须独立完成所有实验环节。

**二、实验环境（工具、材料、硬软件）**

1、硬件平台：PC机

2、实验环境：Windows平台

3、操作系统：Windows 7/10

4、实验开发工具：JDK, Eclipse等

**三、具体内容及分析**

**校园二手交易网**

（1）主要实现系统登陆注册, 商品查看, 添加购物车, 模拟结算的功能, 能够方便高效同学进行二手物品的自定义发布信息与交易,提高商品的使用效率。

（2）运行环境和相关技术：

jdk1.8，tomcat，mysql，idea，eclipse等

（3）至少完成如下功能：

* 系统登陆注册模块, 能够完成注册登录
* 个人资料模块, 可以展示个人资料, 并查看个人的发布商品列表与已经购买的商品
* 商品发布模块, 新建二手物品信息, 提交后跳转到个人商品发布列表页显示。
* 购物车逻辑, 包含购物车逻辑, 对某个商品可以加入购物车并下单。
* 商品展示模块, 可以查询商品列表，搜索商品，并查看商品页详情。

**二、报告撰写要求：**

1、可按照系统概述（系统及功能进行介绍，包括使用到的开发技术和框架）、系统开发及运行环境、数据库设计（设计出对应的数据库表）、分模块介绍详细实现内容分小节介绍设计和实现过程。

2、图文结合，不要大面积的粘贴代码，可采用建模语言，如uml描述

3、建议在各功能实现部分，增加实现过程中所遇问题、如何解决等。

**三、报告提交要求**

提交：数据库文件+项目文件夹+课程设计文档打包（数据库导出为.sql文件）。

文档和压缩包均命名为：学号-姓名-班级-java web程序设计 -课程设计

请最晚于2022年7月10日前由学委收齐，上交。没有按时提交，导致无期末成绩的，责任自负。

**四、评分**

以题目1为例：

评分分为设计实现和报告撰写两部分。总分=50%代码+50%报告

1. 代码部分分数100

* 完成5个以上功能，至少给予80分基础分，完成功能越多，注释越全面或细节方面优化越好，按实际情况加分
* 无法正常运行，或功能较少在原评分基础上扣减至少5%，扣完为止。

1. 报告撰写部分分数100

* 基础分60分
* 大面积粘贴代码，分数至少扣减5%，反之给出较为详细的实现过程（如文字描述+uml建模）给予加分至少5%。
* 没有分析设计实现过程，分数至少扣减5%，反之加分
* 抄袭，直接0分计

注：1、题目一基本要求和数据库设计已给出，设计较为简单主要功能多为增删改查，要求源码打包提交，该文档中不要粘贴代码，“系统实现”章节必须介绍具体的技术实现，讨论详细实现环节（如选课部分，课时冲突的解决），可适当的粘贴效果图、代码截图。

目录

[一、项目介绍 6](#_Toc28972)

[1.1 项目背景 6](#_Toc15685)

[1.2 简介 6](#_Toc9600)

[二、需求分析 7](#_Toc12013)

[2.1 需求综述 7](#_Toc5436)

[2.1.1 功能需求 7](#_Toc16314)

[2.1.2 性能需求 11](#_Toc14450)

[2.1.3 校园需求分析 11](#_Toc29791)

[2.1.4 社会需求分析 11](#_Toc25162)

[2.2 数据流图 11](#_Toc14239)

[2.3 数据字典 15](#_Toc11877)

[2.3.1 数据元素 15](#_Toc12830)

[2.3.2 数据存储 15](#_Toc18867)

[2.3.3 数据流 16](#_Toc24871)

[2.3.4 数据处理 16](#_Toc27874)

[2.4 可行性分析 18](#_Toc19808)

[2.4.1 经济可行性分析 18](#_Toc29447)

[2.4.2 技术可行性分析 18](#_Toc30231)

[2.4.3 操作可行性分析 18](#_Toc29832)

[2.4.4 市场定位分析 19](#_Toc30534)

[三、 平台设计 20](#_Toc30224)

[3.1 平台功能设计 20](#_Toc8467)

[3.2 项目结构 21](#_Toc13458)

[3.3. 模块设计 22](#_Toc23663)

[3.4数据库设计 23](#_Toc7675)

[3.5 制定测试计划 24](#_Toc19810)

[3.6 审查和复查 25](#_Toc17590)

[四、主要界面及实现 26](#_Toc26894)

[4.1 主页面 26](#_Toc19520)

[4.1.1 主页面设计 27](#_Toc9322)

[4.1.2 主页面实现 28](#_Toc32597)

[4.2、登录、注册及找回密码界面 28](#_Toc19018)

[4.2.1 登录、注册及找回密码界面设计 30](#_Toc1146)

[4.2.2 登录、注册及找回密码界面实现 32](#_Toc14894)

[4.3、个人主页 35](#_Toc3062)

[4.3.1 个人主页设计 35](#_Toc7909)

[4.3.2 个人主页实现 36](#_Toc27254)

[4.4 订单页面即购物车页面 36](#_Toc25365)

[4.4.1 订单页面及购物车页面的设计 37](#_Toc28146)

[4.4.2 订单页面及购物车页面的实现 38](#_Toc6940)

[4.5 数据库设计 38](#_Toc26250)

[五、关键部分源代码 41](#_Toc5869)

[5.1、JSTL+EL页面展示源代码 41](#_Toc27504)

[5.2 个人主页面代码设计 44](#_Toc9167)

[5.2.1 修改密码 44](#_Toc25640)

[5.2.2 绑定手机和邮箱 46](#_Toc1327)

[5.3 登录注册页面 46](#_Toc24724)

[5.3.1 注册前端功能 46](#_Toc9442)

[5.3.2 注册后端功能 49](#_Toc24242)

[5.4、加入购物车以及立即购买 50](#_Toc16005)

[5.6 BootStrap 52](#_Toc21236)

[5.6 Junit单元测试 52](#_Toc9955)

[5.7 Druid数据库连接池 53](#_Toc12432)

[5.8 JdbcTemplate 54](#_Toc6025)

[六、总结 55](#_Toc22860)

[七、参考文献 56](#_Toc26605)

# 一、项目介绍

## **1.1 项目背景**

随着时代与科技的不断发展，各种资源的更新换代步伐逐渐加快，造成各种资源剩余和浪费，同时网络的飞速发展，为人们创造出了新的交易方式。而如何合理的利用这些“二次资源”，为其建立规范的交易平台市场，是当代互联网发展的一大方向。在学校生活的我们，难免会在网上购买物品，但有时候网上的物品太昂贵，而且自己也不怎么会使用该商品，比如为了跳绳比赛而购买的跳绳，自己可能买来都不会用。这时候，我们一般会去计院的小黑市群里问问学长学姐等群成员，看是否有人要出售自己用不上的物品。当然自己也可以在此群发布自己要售卖的东西。这种群里发布的方法，存在以下几个缺点：

* 不能及时发现群里的消息：当群里有自己需要的物品时，由于自己不能看手机而导致错过该消息。
* 看到的消息存在滞后性：可能自己看到人家售卖的物品时，该物品已经被先于你看到的同学购买了，这样你在联系消息发布者就会出现售空现象，不仅费事还浪费别人宝贵的时间。
* 无人理睬自己发布的消息：有时自己发布的收二手物品的消息迟迟没人回应，但自己又很想要采购该物品，就会比较麻烦，还可能遇到第一个、第二个问题导致一系列物品系统不同步、不一致的问题。

## 1.2 简介

为了解决物品信息的不一致和QQ群售卖带来的诸多问题，我决定开发一个用于我们学院的二手交易平台，其信息储存于远程数据库，利用浏览器通过网页的形式展示给用户，用户在网页上的操作就是对数据库的操作，会被实时修改到数据库中，这样就可以避免上面说的第二个问题消息的滞后性。由于系统是实时更新的，因此自己需要时打开网页就可以发现有无自己需要的商品，解决了第二个问题和第三个问题。

本项目就是一个简易的网络实时购物系统，类似于京东、淘宝等购物系统一样，都有基础的购物功能，当然，本项目还可以发布自己需要售卖的物品，这项功能不需要自己建立店铺，只要注册账号了就可以发布，这是京东等不存在的功能。本平台立足于解决高校二手物品“交易难”，“联系难”，“售后难”的问题，尝试建立一个统一的二手交易平台来促进校园内闲置物品的流通，规范二手物品的交易流程，让学生能够省钱省力省心。

本项目利用了HTML5+CSS3+JS+JQuery来完成前端的设计，前端设计遵循样式和结构相分离的原则，将HTML与JS和CSS等的分开存放于文件。本项目使用的数据库是MySQL数据库，使用该数据库有两个原因：一个是该数据库存在开源免费的版本，另一个就是本学期的数据库课程学习是以MySQL为例来讲解的，对MySQL比对SQL Server和Oracle等熟悉。本项目的后端使用的是容器工具是Tomcat8.0版本，本项目在该容器中运行，项目使用主要后端技术包括MVN三层设计模式也就是JSP+Servlet+JavaBean，此外还利用了Ajax+EL+JSTL+Druid+JdbcTemplate+qcloudsms等技术。平台主要分为前台与后台两大模块，前台主要功能有用户注册登录，浏览商品，查看商品详情，加入购物车，结算，查看订单等。后台主要功能有各张表的查看，增删查改等。目前设计并实现了五张基本表：用户表，商品分类表，商品表，订单项表，订单表。通过这些基本表，为整个二手物品购买流程提供了数据支持，以后还可以增加更多数据表来完善平台功能，甚至可以以此为基础上线商用整个平台。

# 二、需求分析

2.1 需求综述

校园里存在着很多二手商品，由于信息不够流通，可能仅仅只有园区内部分二手货物能进行交易，导致了很多仍然具有一定价值的二手商品囤积乃至当作废弃物处理，资源得不到利用。

校园二手交易系统立足于解决校园内闲置二手物品的出售和购买需求，用户通过学号登录后可以浏览，发布，购买商品等，管理员则负责对整个平台进行审查维护。

本系统主要处理的数据有商品信息表、用户信息表、订单信息、订单的完成状态和商品的分类。

2.1.1 功能需求

用户：使用学号注册账号，登录后可以浏览，搜索，查看商品，在找到心仪的商品后，可以选择加入购物车。在确定购买意向后，可以在购物车中进行结算（时间关系暂时不引入真实支付API）。收到货物后，可以对商品进行评价，或者售后。用户也可以在平台上出售闲置物品，填写相应信息并通过审核后即可上架。

数据管理员：输入管理账号和密码后，登录管理端界面。管理端分为商品分类管理，商品管理，订单管理，用户管理等。同时，管理端还应该看到整个平台的运行情况，包括日/月均销量，单量最高的商品信息，平台访问数等。关于管理功能的详细描述如下：

分类管理：对商品分类信息进行增删改。

商品管理：审核新上架商品是否合格，下架不合格商品，修改已上架商品。

订单管理：查看订单详情，删除不合理订单（恶意刷单）。

用户管理：查看用户详情，处理违反平台规定的用户。

本项目的基础功能主要包括登录、注册、找回密码、显示用户信息、修改密码、绑定手机或邮箱等等的功能，本项目独有的功能是加入购物车、立即购买、查看订单状态、发布售卖信息等。

* 登录、注册及找回密码功能

这三个功能在同一个前端页面，通过label标签来实现转换页面，登录只需输入账号密码即可完成登录，其中会检测用户的账号存在以及密码是否错误等信息。

注册中实现了一个发送手机验证码或者邮箱验证码的功能，发送邮箱验证码使用的是javax.mail里面封装的SimpleEmail类，该类可以发送纯文本的简单邮件，不能包含附件信息，由于发送验证码我们也不需要使用附件，因此使用该类就可以满足了。另外发送的服务器我使用的是我们学校的教育邮箱服务器，其地址为smtp.email.ncu.edu.cn，使用自己的教育邮箱发送验证码。手机验证码的发送使用的是腾讯提供的短信服务，利用qcloudsms中SmsSingleSender类发送手机验证码，将自己服务器的中的AppID、AppKey、templateID、签名等信息设置好后就可以实现发送手机验证码了。

找回密码功能就是一般的模式，先验证已绑定的手机或邮箱，验证成功后再验证新的手机或邮箱即可，其中发送验证码的方法和上面的方法是相同的。

* 显示用户信息

该功能会在用户登录后显示用户的信息，其中用户的信息是从后台的数据库读出来的，使用JSTL结合EL表达式来做出展示，其中提供了绑定手机和邮箱，修改用户名的操作。

* 查看订单信息

该功能是查看用户目前订单的状态，其中订单状态包括，未付款、待收货、待评价、已完成和我的售卖，其中的数据均是保存在数据库中，通过前端进行展示，其中由于数据会很多，因此设计了一个分页展示的功能。其具体实现是使用了SQL语句DQL中的分页查询来实现的。

* 加入购物车以及立即购买

加入购物车功能是每一个购物系统都必须存在的，可以在商品页面加入购物车，由于是二手购物平台，因此售卖的物品肯定都是单类的，不存在一个商品有不同的型号之分，因此加入购物车时仅需选择数量即可，加入购物车不会限制数量，在购买下单时，会检测数量是否小于库存，如果小于会给用户提示信息。

* 锚点链接

为了实现页面内部的跳转，使用a标签，其中href为要跳转本页元素的id即可，这样就可以实现首页侧边导航栏的导航效果。

项目页面主要需求如下：

主页面：

1、完成商品的显示以及加入购物车和立即购买

2、加入购物车或立即购买时会检测是否有用户登录，如果当前无用户登录则提示登录

3主页面存在导航栏和锚点链接

登录注册页面：

1、主要完成登录注册以及找回密码的功能

2、注册和找回密码要使用邮箱或者手机接收验证码

购物车页面 ：

1、显示该用户的购物车信息

2、在购物车界面，要能删除购物车信息和改变数量等信息

个人主页：

1、显示当前用户的个人信息

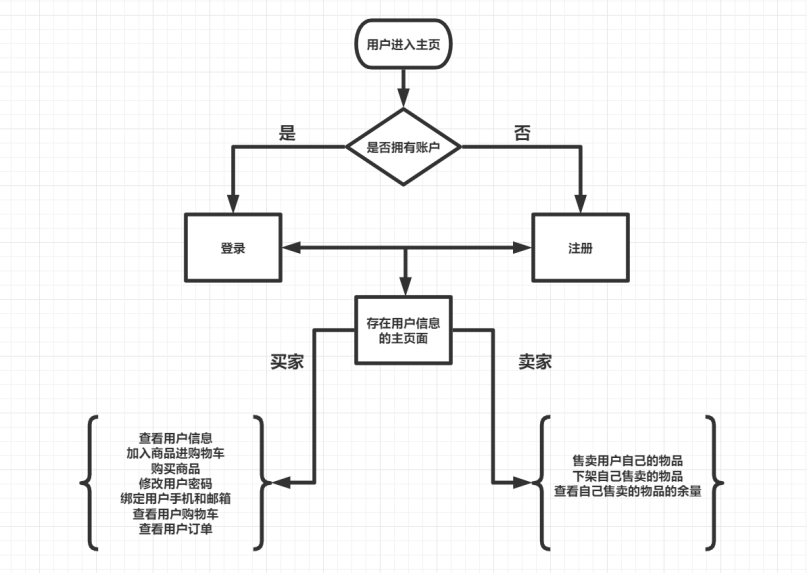
2、要能自己修改密码和绑定邮箱或者手机

订单页面：

1、显示订单

2、提供改变订单状态的功能，比如确认收货、取消订单等

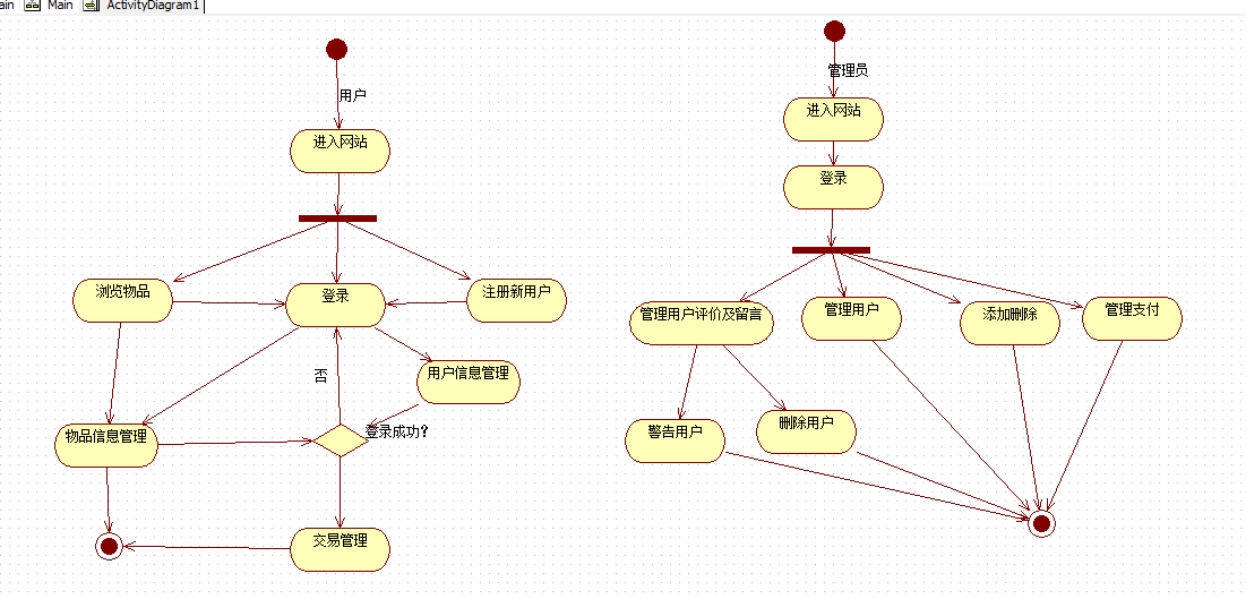
项目逻辑功能如下图：



逻辑功能图



导航模型



活动图

2.1.2 性能需求

性能需求指定系统需满足定时约束或容量约束，本系统主要因为涉及的数据库规模较小，响应时间也较短，完全可以做到实时相应，体验较好。

2.1.3 校园需求分析

从校园市场容量看，校园市场具有很大的开发潜力。大学生在校四年会买下很多东西，一些东西例如生活用品、用过的旧书等几乎对所有学生来说都是必要的，但大多数都被毕业生低价出售给废品收购者甚至扔掉，能获得馈赠的在校生非常少，在供给信息极端匮乏的情况下，在校生只能去购买全新的商品。当学生有物品闲置不用时，通常只能将之束之高阁，而有此需要的同学在无法获得信息的情况下，也只能去购买全新的商品，増大了经济压力并且不利于资源的节约。

通过校园二手交易系统，可以为临毕业同学解决实际困难，处理带不走的闲置物品，同时，也使需要的同学以较低的代价获得需要的物品，并有效的解决了资源浪费的问题。

2.1.4 社会需求分析

二手商品是学生比较青睐的廉价商品，网站设计应着重突出实用和廉价。网上交易为学生提供一个供需平台，学生可以将自己不用的东西放在平台上，也可以在平台上购买自己需要的东西。建立二手交易市场的主要目标是：

1、为学生处理不需要但仍有剩余价值的物品

2、为学生提供廉价、实用的二手商品

3、净化校园环境，提供规范统一的二手交易平台。

2.2 数据流图

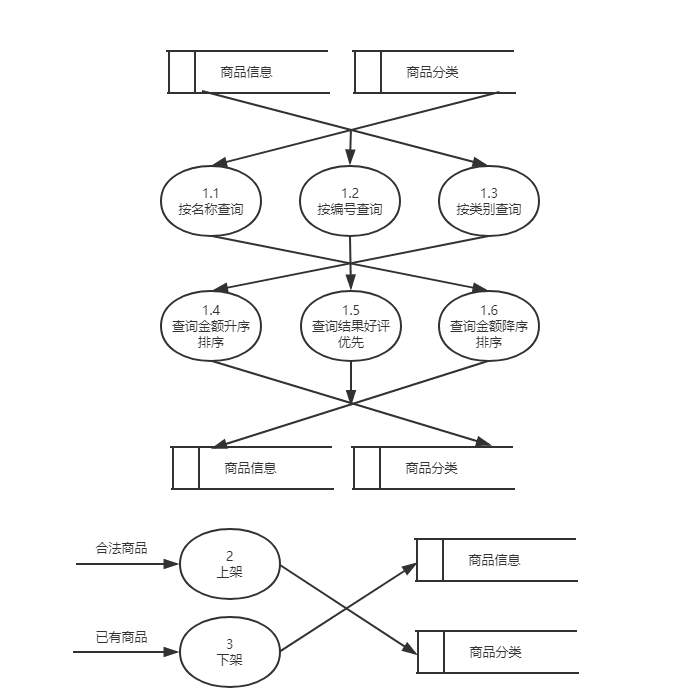
数据流图

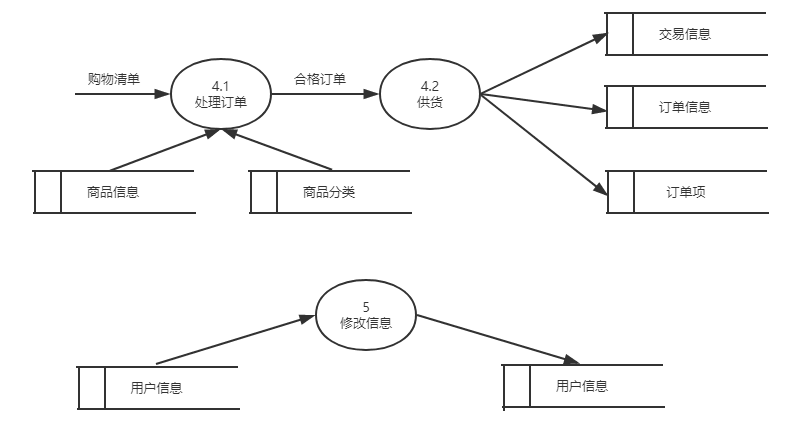


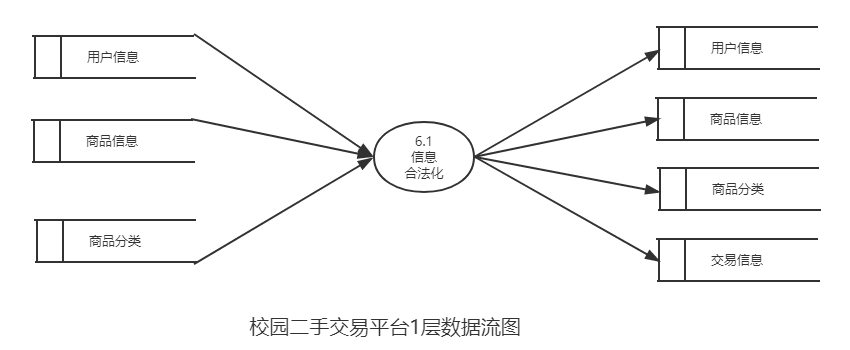
校园二手交易平台顶层数据流图



校园二手交易平台0层数据流图







平台HIPO图

2.3 数据字典

2.3.1 数据元素

数据元素是不可再分的数据单位，一般而言，包括如下内容：数据项描述＝｛数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系，数据项之间的联系｝.其中，后两项定义了数据的完整性约束条件，用于数据校验。

本系统的数据元素具体详见下文的数据库表。

2.3.2 数据存储

数据存储是数据结构停留或保存的地方.也是数据流的来源和去向之一，可以是手工文档或手工凭单，也可以是计算及文档.一般而言，应包括如下内容：｛数据存储名，说明，编号，输入的数据流，输出的数据流，组成（数据结构），数据量，存取频度，存取方式｝。

名称：商品信息表  
描述：记录商品的各种基本信息  
编号：D1  
组成：商品编号，商品名称，价格，图片，类别，数目，商品是否下架，备注

名称：商品分类  
描述：各种商品属性的分类  
编号：D2  
组成：生活日用品，家用电器，数码电子产品，图书音像，运动户外，衣物鞋帽

名称：用户信息表  
描述：记录用户的基本信息  
编号：D3  
组成：学号，用户密码，姓名，昵称，性别，电话，邮箱，地址，支付账号

名称：订单信息表  
描述：记录订单的基本信息  
编号：D4  
组成：订单编号，金额，订单状态，收货人信息，下单时间

名称：订单项  
描述：每一个订单中的订单项信息

编号：D5  
组成：商品编号，订单编号，购买数量，小计

2.3.3 数据流

数据流是数据结构在系统内传输的路径.对数据流的描述通常包括如下内容：｛数据流名，说明，编号，数据流来源，数据流去向，组成（数据结构）｝。

2.3.4 数据处理

处理过程的具体处理逻辑通常在详细设计过程中用判定表或判定树来描述.在数据字典中，只无原则描述处理过程的说明性信息，通常包括以下内容：｛处理过程名，说明，输入数据流，输出数据流，处理｝

编号：P1

名称：系统管理

描述：管理员对系统管理维护

输入数据：D1,D2,D3

输出数据：D1,D2,D3,D4

编号：P2

名称：商品出售上架

描述：商品对出售商品的描述和更改

输入数据：D1,D2

输出数据：D1,D2

编号：P3

名称：商品购买

描述：买家对商品的购买记录

输入数据:D1,D2

输出数据：D4,D5,D6

编号：P4

名称：商品查询

描述：完成对所需商品的查询

输入数据：D1,D2

输出数据：D1,D2

编号：P5

名称：用户信息更改

描述：用户对于自己信息的更改

输入数据：D3

输出数据：D3

2.4 可行性分析

2.4.1 经济可行性分析

经济可行性是指可以使用的资源的可能性（资源包括人力资源、自然资源和资金条件）。任何一项公共政策都存在一个争取公共经济资源的问题。经济可行性包括两个方面的内容：一是某一备选方案占有和使用经济资源的可能性，进而实现政策目标的可能性；二是实施某一政策方案所需花费的成本和取得的收益。

校园二手交易平台前期作为课程设计的任务，在自然资源、资金条件方面没有任何需求，只需要开发者投入较多的时间设计完善系统即可。平台开发完成后，如果后期打算投入使用，则只需花费较少的资金和人力资源进行平台的维护。

总的来说，该系统对自然资源没有任何要求，开发完成后只需要管理员进行平台的维护，对人力和资金的需求也不大，在经济方面是完全可行的。

2.4.2 技术可行性分析

校园二手交易平台开发的目的是给学校的师生们处理闲置物品、购买二手商品提供一个操作简单，使用便捷的平台。该系统的功能包括用户商品查询，上架，下架，购买，以及管理员审查维护等功能，因此，一个简洁明了的操作环境是必需的。该平台与用户交互界面的开发涉及到HTML，CSS，JavaEE等方面的知识，用于搭建该交易平台的网页和布局，构造一个使用起来较为方便，人人都容易上手的界面。把各个功能标注于明显位置上，方便使用者点击。同时本系统使用MySQL对后台的数据进行增、删、改、查等操作，使得数据完整性得以保证，同时使数据操作简单化。

2.4.3 操作可行性分析

本系统界面清晰，操作简单，使用起来非常方便。前端网页开发完成后，用户可以非常轻松地通过网页使用该平台，进行商品的购买，发布等操作。而平台管理员只需要经过简单培训便可掌握该平台的具体维护功能。

该平台对使用者和管理员的技术要求较低，在操作可行性方面也是可行的。

2.4.4 市场定位分析

目前上网已经成为大学生活不可或缺的一部分，在校大学生平均上网率将近100%。二手商品是学生比较青睐的廉价商品。校园二手交易网站为在校大学生提供一个供需平台，学生可以将自己不用的东西放在网上，也可在网上找到自己需要的东西。物美价廉，达到双赢。校园二手交易平台定位于面向广大学生，网站提供详细的商品信息、简便快捷的操作过程。综上所述，构建校园网上二手商品交易网站在市场上、技术上都是可行的。因而决定开展二手商品交易网站系统开发。

# 平台设计

# 3.1 平台功能设计

整个系统是比较简单的，其设计主要采用MVC三层设计模式。

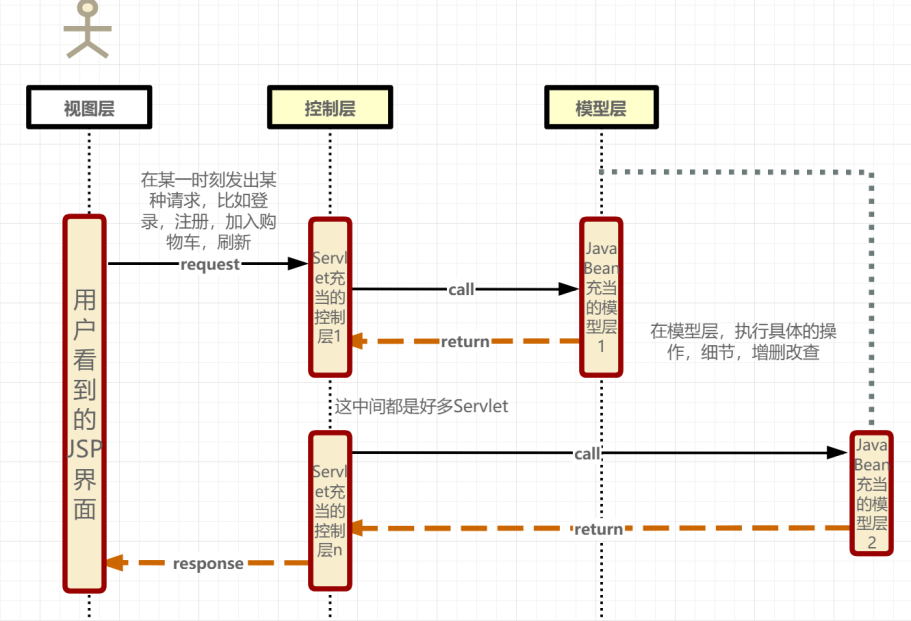
M即Model模型是指模型表示业务规则。在MVC的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

V即View视图是指用户看到并与之交互的界面。比如由HTML元素组成的网页界面，或者软件的客户端界面。MVC的好处之一在于它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，它只是作为一种输出数据并允许用户操作的方式。

C即Controller控制器是指控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

本项目使用JSP+Servlet+JavaBean来完成MVC设计，其中JavaBean充当模型层、Servlet充当控制层、JSP充当视图层，三层各自联系，又具有一定的独立性。

项目设计的架构图如下：



项目时序图

用户在前端看到的仅是JSP等的视图，此时用户处于视图层，在视图中用户可以提交自己的请求，修改密码，登录，加入购物车等等，其目的就是为了和后台数据库打交道，实现数据交换。接着会通过Ajax技术发送一个同步或者异步请求给后端的Servlet，这时就来到了控制层，每一个功能会对应一个Servlet，Servlet控制层所控制的逻辑大致分为三步，第一步就是设置响应头格式和编码为UTF-8，第二步就是如果存在参数就获取参数的值，第三件步就是调用对应的Dao层进行操作，此时就来到了模型层，该层负责实现用户具体的需求，比如注册一个账户就是在数据库中插入一条满足数据完整性的记录，再比如登录就是查询用户表中存在存在对应账号和密码的记录，等等这些功能都是动过模型层来实现的。

## **3.2 项目结构**

本项目的结构分为两个大类，第一大类就是webapps下面的JSP和样式脚本代码，这些代码主要负责组织前端的视图层界面。第二大类就是后台的Java代码，对于第二大类主要分为四个部分：

* Beans

这个包下的类都对应于数据库中的一个实体，其中属性和数据库的字段名一一对应，其目的就是为了方便自己编程不会把字段弄错，最重要的就是后期使用JdbcTemplate模板时可以直接将查询的数据封装为一个类对象，而不需要自己在去一个一个封装，既费时又耗力，还容易出错。

* Daos

这个包下的类就是充当项目模型层的类，该包下的类会实现具体的需求，通过操作数据库完成自己的任务，最后将执行结果返回给控制层即可。

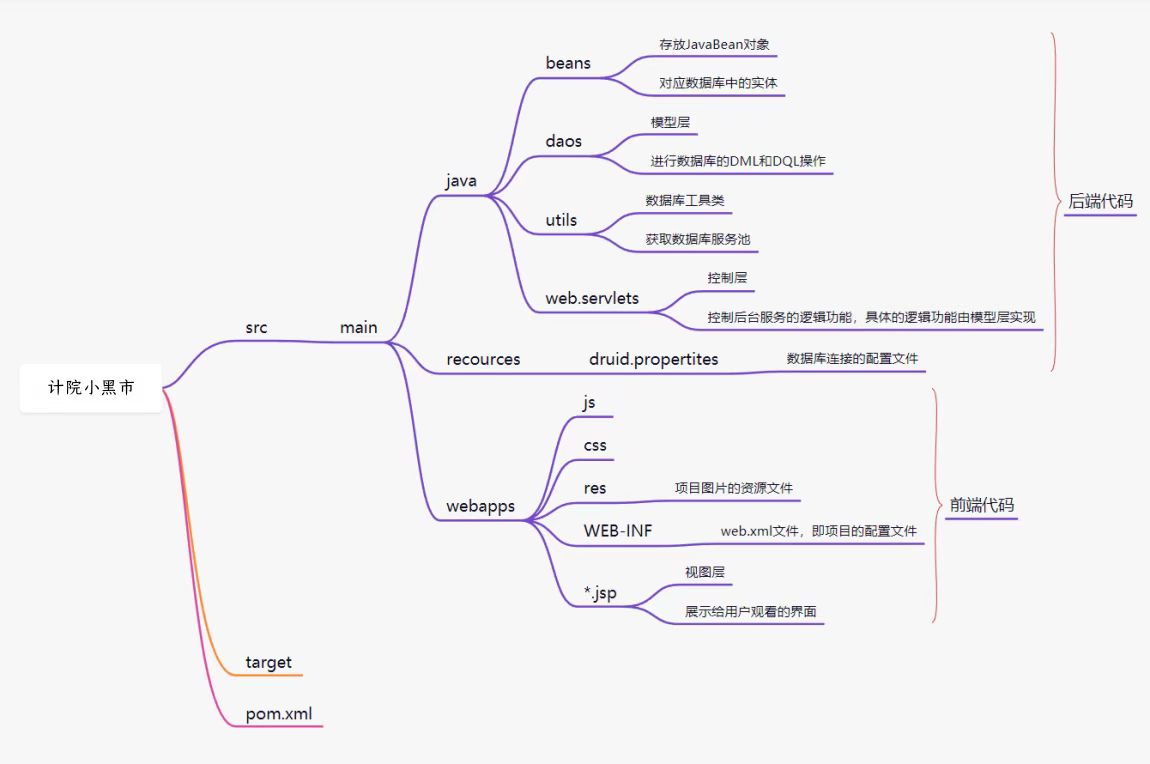
* Servlets

该包下的类主要是充当控制层的Servlet类，用于接收视图层用户的请求， 获取其参数，然后调用模型层的方法执行，等待返回结果，最后将得到 的返回结果通过响应的方式响应给视图层，将得到的数据交给视图层做 显示。

* Uitls

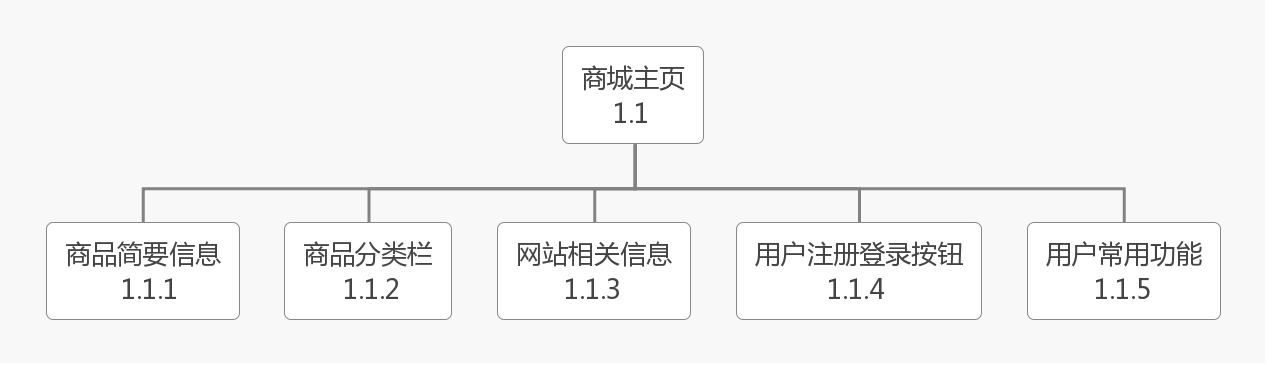
该包下存放的是用户获取数据库连接的工具类，其中数据库连接的技术使用的阿里巴巴提供的一种Druid连接池的技术，最后将使用JdbcTemplate来操作数据库，这个模板里面封装了许多的方法，包括获取连接到释放连接。

项目的工程结构图如下所示：



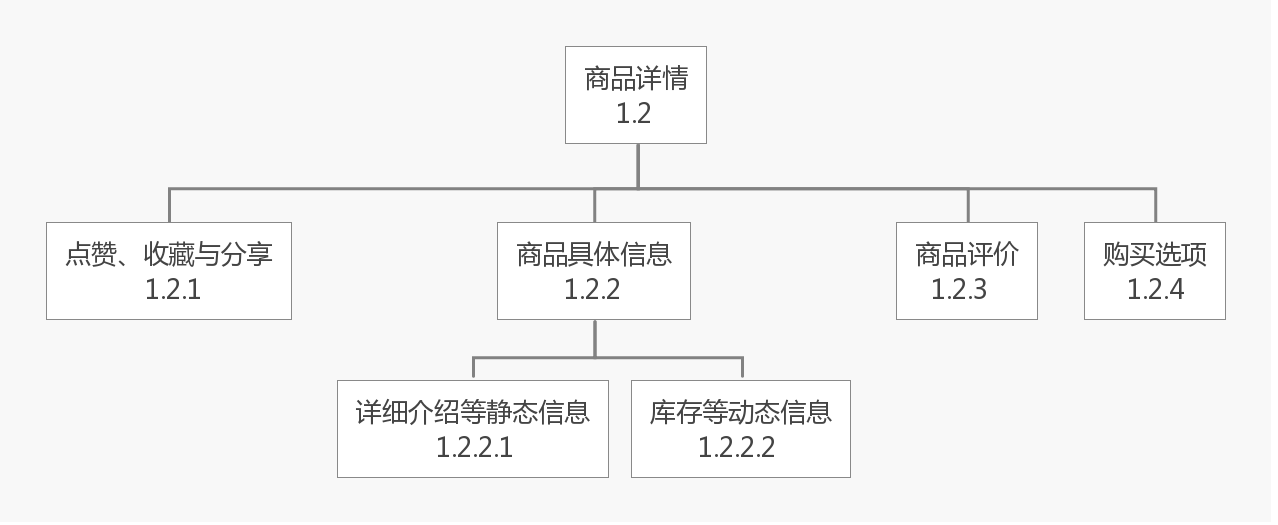
3.3. 模块设计

商城主页模块设计。商城主页是用户第一次进入网站时看到的界面，主要功能为展示商品的简要信息、商品分类栏、网站相关信息、用户注册登录、用户常用功能入口等。商品简要信息主要用来向用户展示二手物品并吸引用户打开，主要包括商品照片、名称、价格、出售者。商品分类栏主要展示商品的分类。网站相关信息用来展示该网站的简介、开发者等信息。用户管理入口包括登录与注册按钮。用户常用功能包括购物车、订单、地址、收藏等的快捷入口。综上，商城主页模块的HIPO图如下：



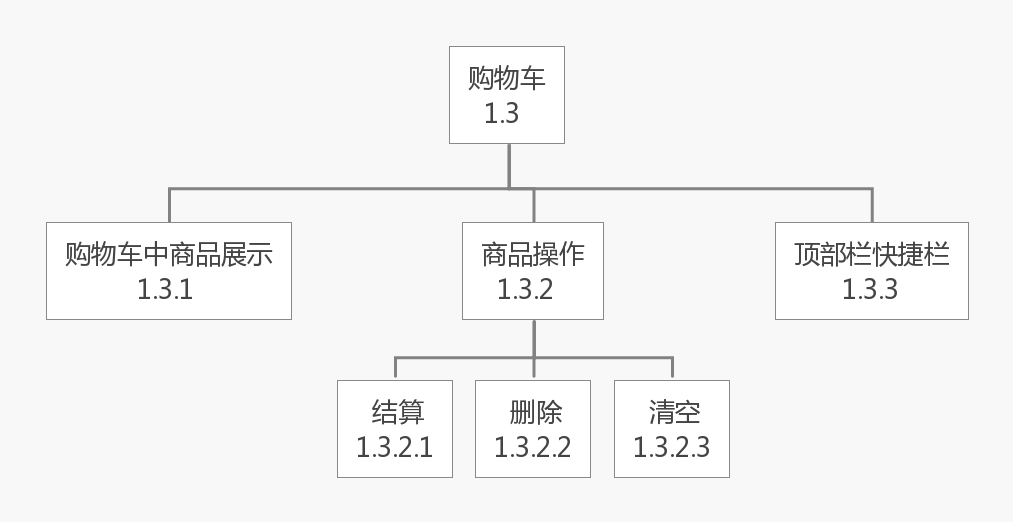
商城主页HIPO图

商品详情模块设计。商品详情界面应该充分展示商品的各种信息以及购买前的准备。首先商品的名称、图片、价格、库存、详细介绍都应该具备，这里因为库存和价格时较易变动的，所以将其归为“动态商品信息”。用户看到喜欢的商品，可以进行一些交互，比如点赞、收藏、分享等操作。对商品的评价也是必不可少的，这样可以让用户与出售者，用户与用户多了交流的渠道。用户在看完商品详情后可能会购买，所以我们应该做好购买前的准备，比如地址的添加、商品数量和附加服务的确认。



商品详情HIPO图

购物车模块设计。本平台认为用户必须把商品加入购物车才能购买，事实上很多购物系统也是这样设计的，这样做会避免很多逻辑和功能上的混乱。购物车功能首先应该能清楚的展示用户购物车中商品的信息，还应该包含一些对商品的操作，比如删除购物车中的商品，清空购物车等，以及最后的结算。最后应该添加一个顶部栏，用以返回主页和进行快捷操作。



购物车HIPO图

订单管理模块设计。订单界面应该展示用户以及购买的订单情况，包括商品的配送状态，以及订单号等信息，当然也需要一个顶部栏返回主页。（功能非常简单明了的没有绘制HIPO图，下同）

用户系统模块设计。用户系统主要包括用户的注册和登录。

后台数据显示模块设计。后台管理员/维护人员在登录后，可以看到平台的相关数据信息，本平台在设计的时候主要展示了使用的5张数据表。包括分类表、用户表、商品表、订单项表、订单表。

后台数据增删查改模块设计。对后台数据的操作主要有：数据的添加、数据的删除、数据的查找、数据的更改。

3.4数据库设计

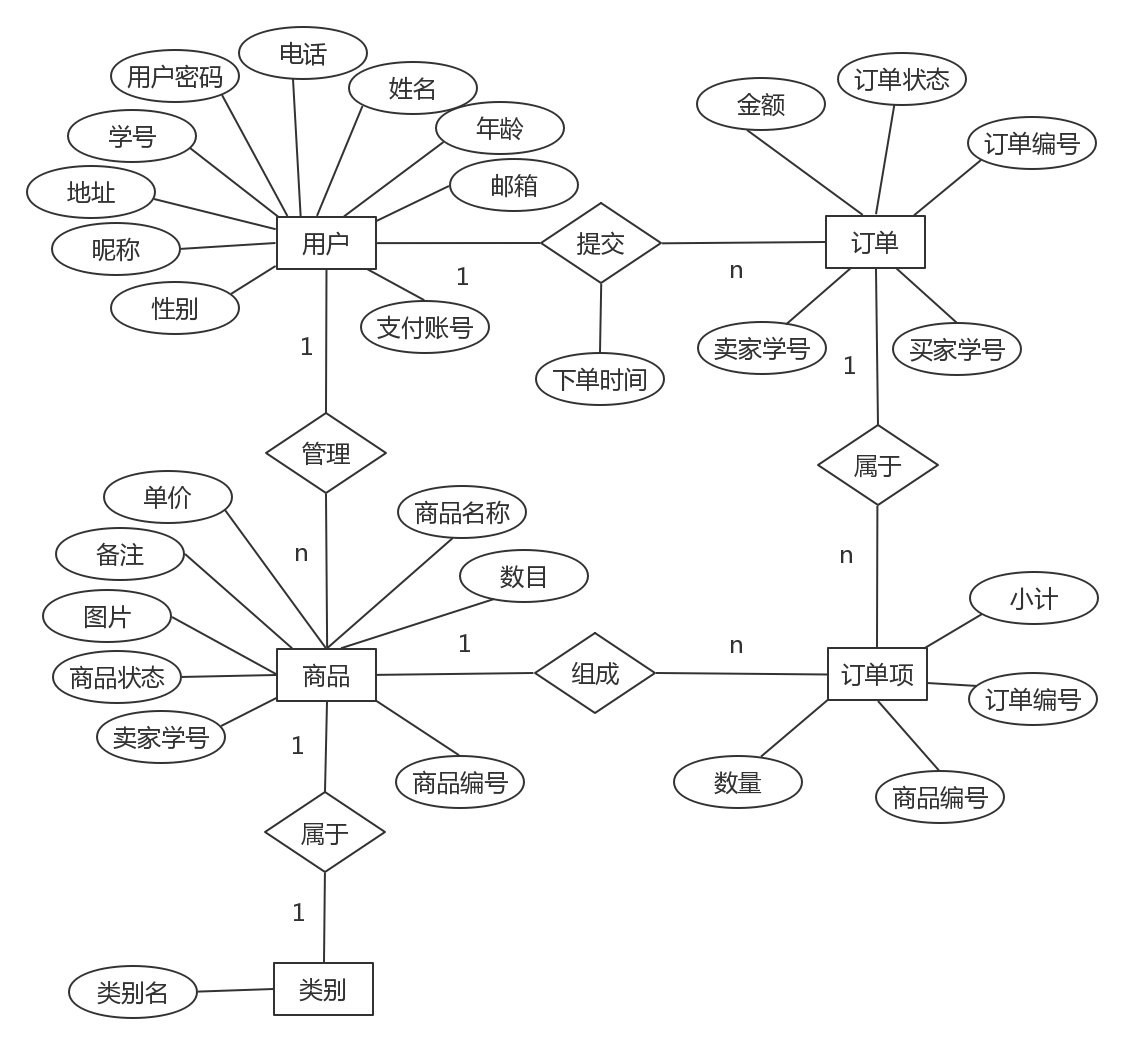
用户（学号，密码，昵称，姓名，性别，年龄，地址，邮箱，手机号，支付账号）

商品（商品编号，商品名称，商品图片，类别名，数目，单价，备注，状态，卖家学号）

商品类别（类别名）

订单（订单编号，买家学号，卖家学号，金额，订单状态，下单时间）

订单项（订单编号，商品编号，数量，小计）



实体联系图

3.5 制定测试计划

在软件开发的早期阶段考虑测试问题，能促使软件设计人员在设计时注意提高软件的可测试性。

在总体设计时，平台的设计细节还没有确定下来，但仍然可以列出一些可能的测试计划:

1. 用户登录和注册的测试：用户输入的数据格式、大小变化，某些选项空缺等是否会达到正确的结果，同名用户应该怎么处理。
2. 商品详情界面：若库存为0时是否能下单，地址确认等问题。
3. 购物车：用户购物车中商品过多时应该怎么做。
4. 订单界面：订单号的唯一性确定。

3.6 审查和复查

最后应该对总体设计的结果进行严格的技术审查，在技术审核通过之后再由使用部门的负责人从管理角度进行复审。

在经过审查后，总体设计符合要求，进入详细设计阶段。

# 四、主要界面及实现

## 4.1 主页面

商品主页面设计如下，包括商品的详情展示：图片、名称、价格、库存、评价、服务等，配送地址、选择购买数量、收藏或分析以及下单操作等。



商品的详情页面设计

### **4.1.1 主页面设计**

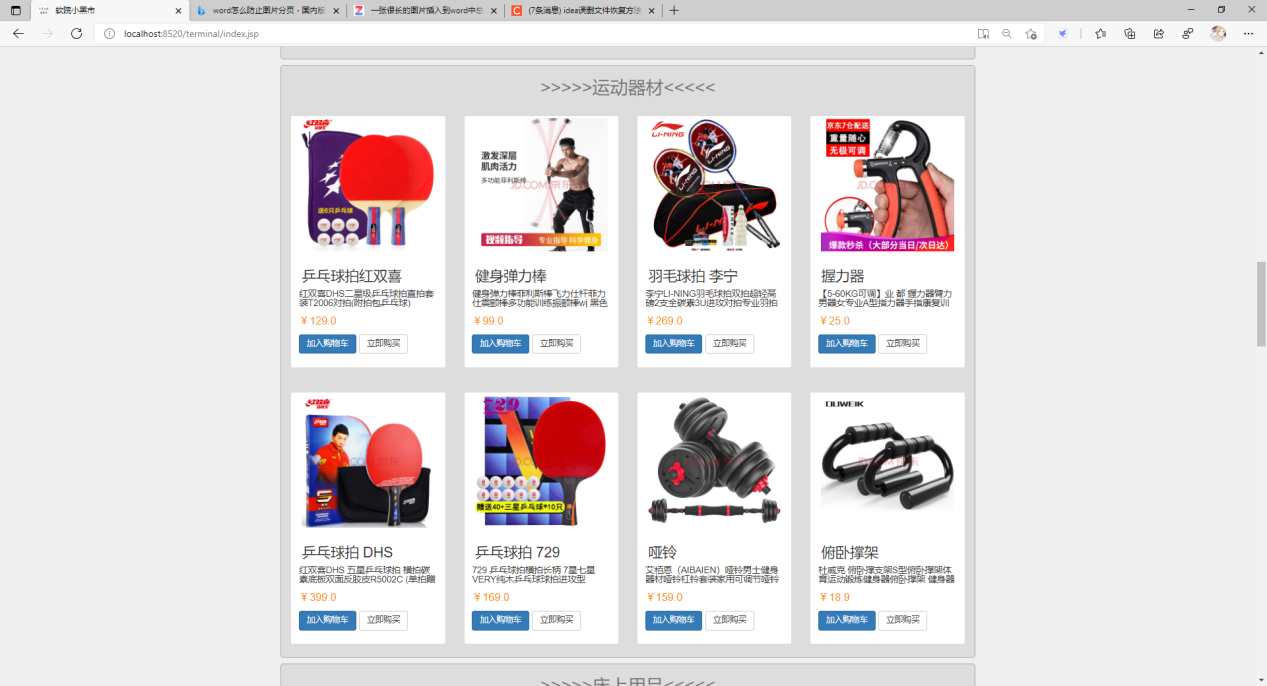
主页面主要分为四个部分，最上面是系统的导航条，中间部分是整个主页面的核心部分，分为页面导航和商品主页面，最下边部分是底部的信息，包括版权信息和联系方式等。

这次主页面都是从后台数据库读取出来的，这是和半期大作业的最本质的区别。从页面读取出来的数据是采用EL和JSTL表达式展示出来的，主要使用的是JSTL的forEach标签和if标签。

主页面的界面如下：



主页面顶端即导航部分



商品部分

另外，如果对应的商品没有库存了，则会不显示该类商品，在侧边的导航栏中会显示缺货，如下：



缺货显示

可以看到，其他物品类商品下方显示的是缺货，并且后面的商品列表中并无其他物品类商品。

### **4.1.2 主页面实现**

主页面使用到的技术主要是HTML5+CSS3+JavaScript的前端技术用于布局出界面的样式，其中HTML5用于搭布局框架，CSS3用于给框架添加样式，JavaSc用于实现按钮等的点击事件，并与后端进行交互。

主要面中的内容使用的是JSTL和EL结合从数据库生成的，其中EL表达式读取的内容来自request的域对象中。具体数据的读出如下：

首先，存在一个Servlet名为InitIndexServlet，这个Servlet的url为inedx.jsp，注意，这不是真正的jsp，而是一个Servlet，没有一个真正的jsp文件名为index.jsp，在这个Servlet中就会调用数据库操作方法执行查询商品，将查询的商品存储于request的域对象中，接着转发到main.jsp，这个文件就是主页面的前端文件，在此main.jsp中使用EL和JSTL就可以将域对象中的数据显示出来。

## **4.2、登录、注册及找回密码界面**

系统的登录和注册设计如下，首先注册账号，根据注册流程图判断是否注册成功。接着可进行登录，登录流程如下图登录流程图所示。



登录流程图



注册流程图

### 4.2.1 登录、注册及找回密码界面设计

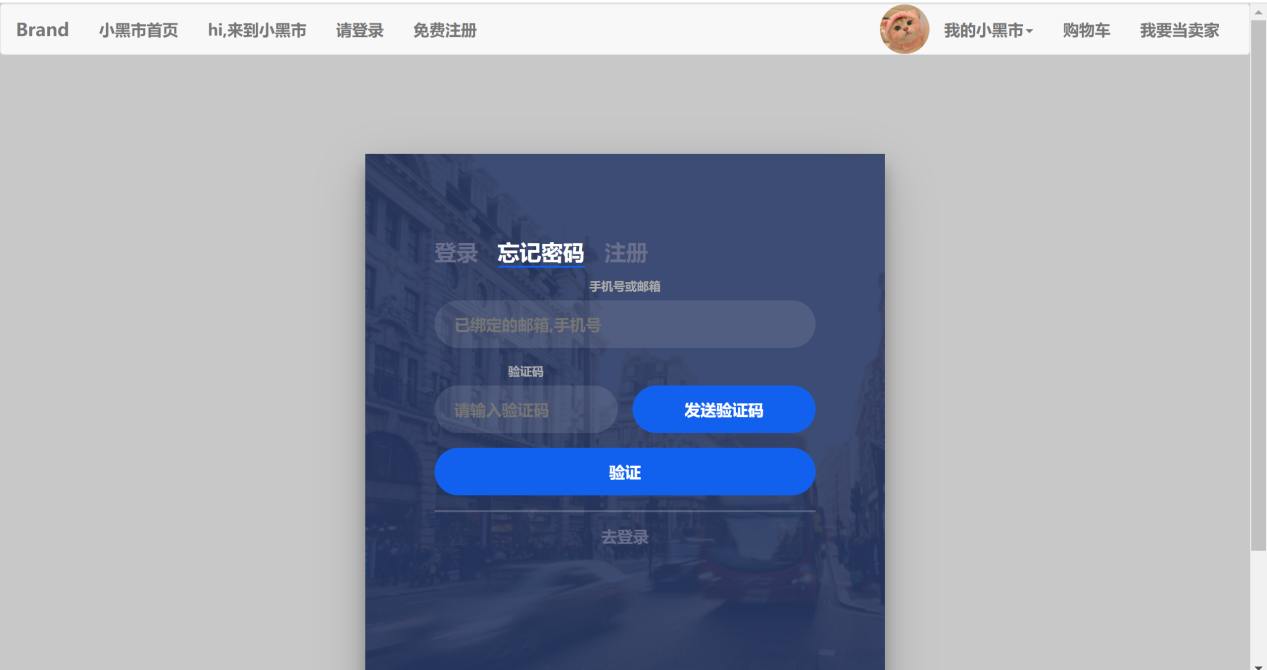
登录、注册、及找回面都是在一个页面中，这三个功能复用了一个页面，减少了页面数量。登录页面需要用户输入账号、密码再点击登录按钮就可以登录了；注册页面会让用户填写一个邮箱或者电话号码，再通过所填邮箱或电话号码接收到的验证码是否正确来确定注册是否会执行。



登录页面



注册页面



找回密码

这三个界面通过小框顶部的按钮来实现切换，并且使用了transform使其沿着y轴旋转，有一种动态特效的感觉。

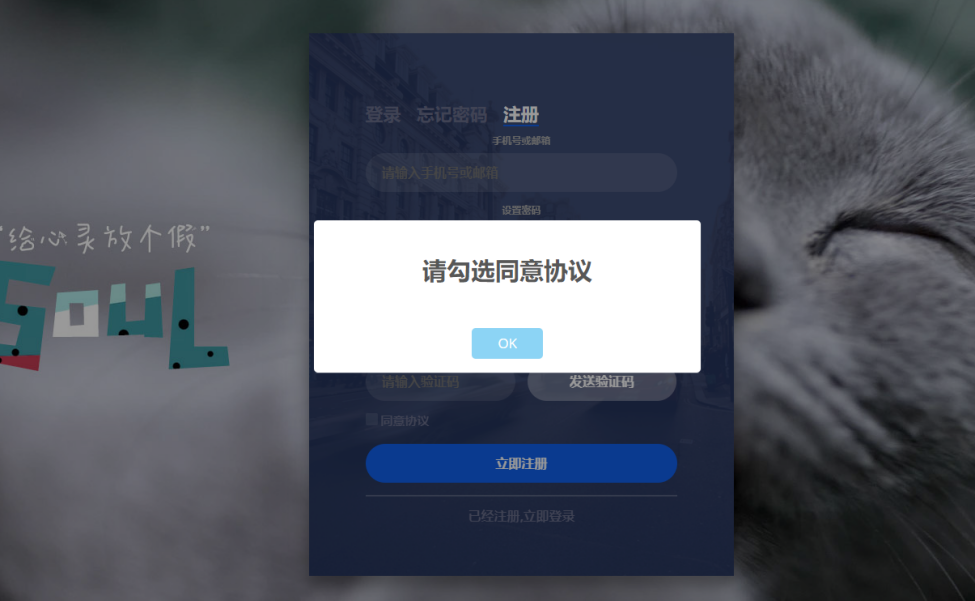
### 4.2.2 登录、注册及找回密码界面实现

前端实现主要是布局和CSS的书写，这里就不在赘述。

后端实现注册的话，首先通过JS从前端获取文本框里输入的内容，并对内容进行检验，分别是否勾选同意协议、邮箱格式是否正确、密码是否为弱密码，两次密码是否一致，这些前端就会做好检测，最后点击发送按钮调用JQuery的Ajax方法发送异步请求，从后端发送验证码到手机或者邮箱。用户再输入自己收到的验证码，最后验证码验证成功方可成功注册。

这里就会涉及到如果用户手机或者邮箱已经注册了，是否还可以继续注册，答案肯定是不可以的，因此在立即注册之前，应该先使用Ajax判断该手机号或邮箱是否已经注册过，如是，则弹出提示用户。

下面演示几张检测失败的图片：



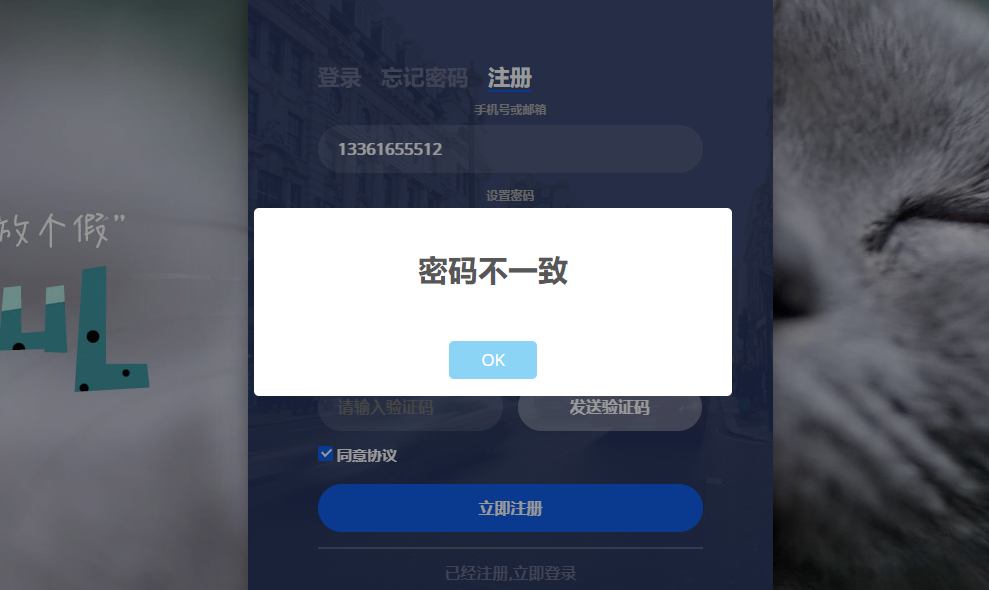
未勾选同意协议



邮箱格式不正确

可以看到，用户输入的是一串英文且没有服务器地址，因此这既不是手机号，也不会是邮箱地址，所以弹出该弹框提示用户。

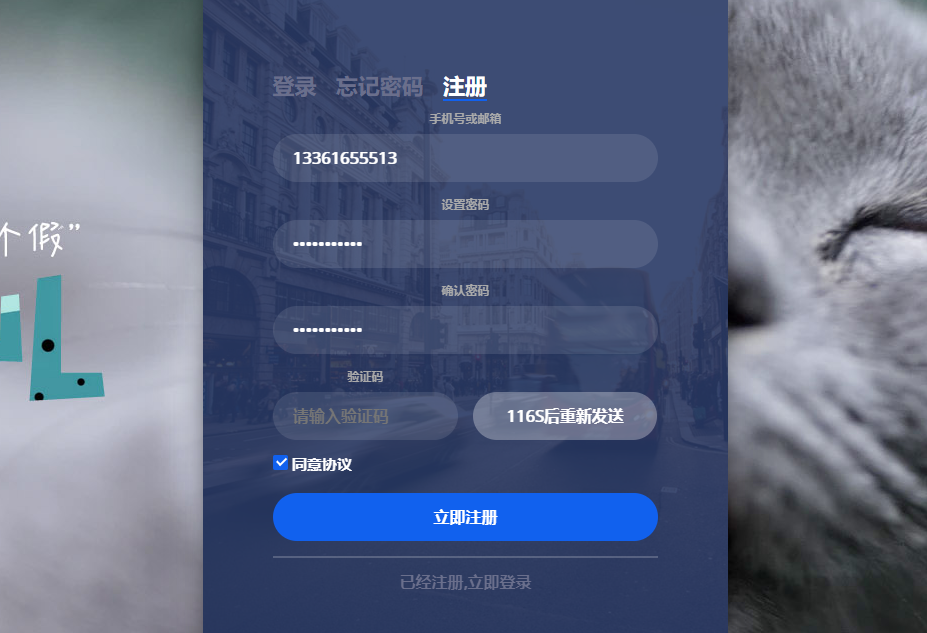
这里检测邮箱或者手机号是否正确使用的是正则表达式，这样既方便准确，又能简化代码的书写。



密码不一致



弱密码



发送成功会存在一个倒计时

这个倒计时就是使用定时器来实现的，每隔一秒执行一个方法，方法中对计数的值减一，再设置该按钮的内容即可。每次启动定时器时，设置一个有效变量使其值为1，代表此时验证码是有效的，当数值减为0的时候，让该变量置为0，代表此时验证码已经过期，并清掉定时器，同时设置按钮内容为重新发送。



验证码过期



显示重新发送

## **4.3、个人主页**

### **4.3.1 个人主页设计**

个人主页的设计比较简洁，主要是用于展示个人信息的，另外还可以修改密码和绑定手机和邮箱账户。

该页面的展示如下：



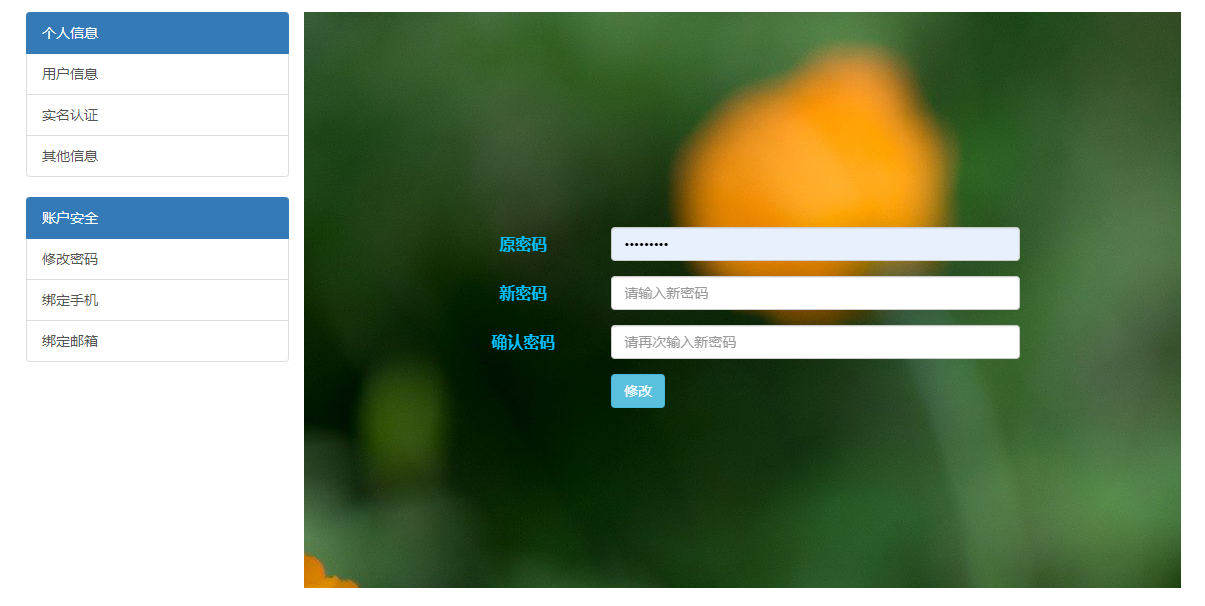
个人主页

个人主页左边是信息列表，右边是具体的信息，其中有的信息可以修改，有的信息不可以修改。

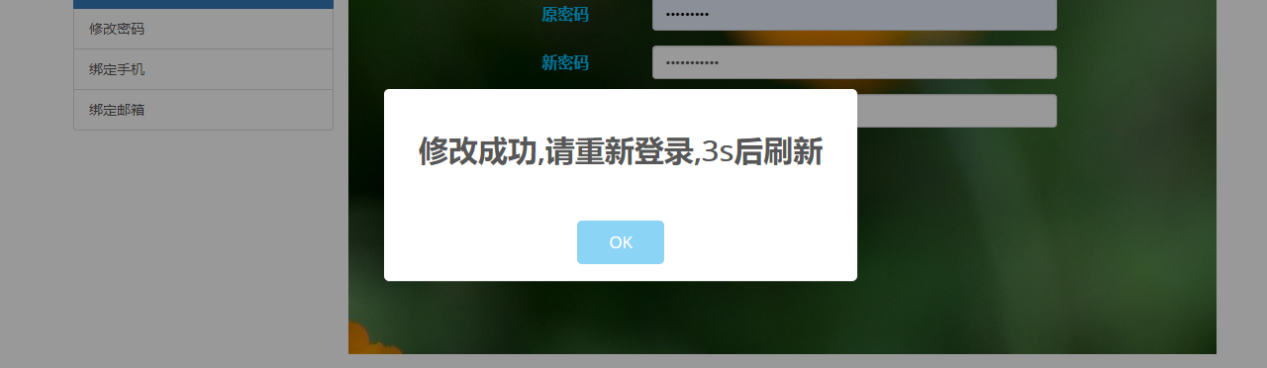
### **4.3.2 个人主页实现**

个人主页的信息会根据用户不同显示该用户对应的信息，其思路就是登录时在后端使用session域存储用户信息，在该页面使用EL表达式做出展示，EL表达式有一个好处，那就是对于没有用户登录或者用户对应信息不存在时，EL表达式输出的是空白，而不是null，这就比较符合常识，不存在的属性不显示。

修改密码会判断当前密码是否正确，其次检测新密码是否为弱密码，最后检测新密码和确认密码是否一致，这些检测都成功之后，就会使用Ajax发送到后台进行修改数据库处理。修改成功后，会提示用户重新登录。



修改密码



修改密码成功

## **4.4 订单页面即购物车页面**

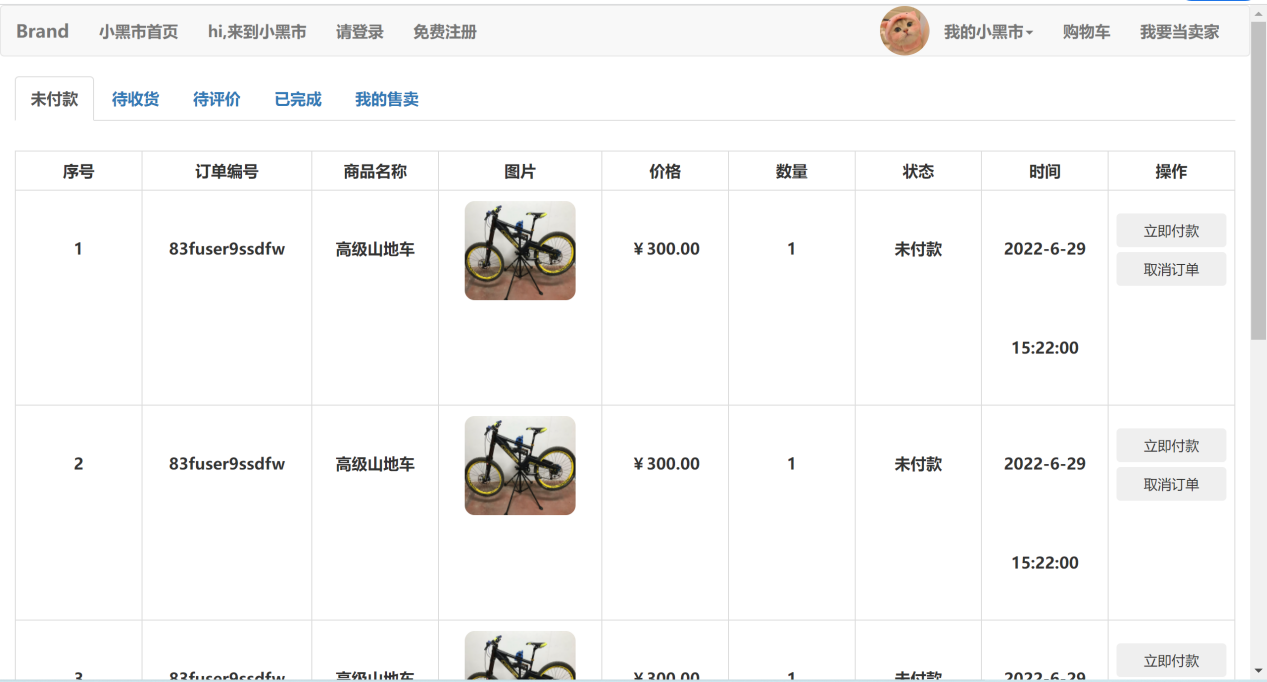
订单管理页面设计如下，包括订单时间，订单号，订单总价，订单中的商品信息，收货人，订单状态，以及进行售后服务等。用户可登录进行查看个人订单。



订单管理页面

### **4.4.1 订单页面及购物车页面的设计**

订单和购物车设计得也比较简单，主要是使用表格作为展示，其中由于内容可能很多，因此还涉及了分页查询的功能，最下面就是分页条，会根据当前的内容来进行自动的展示分页数。其页面展示如下：



商品订单页面分页

我们可以看到，此时订单中只有三页，通过点击数字或者箭头都可以切换到下一页。点击上面的导航条还可以切换到其他类别的订单。

### **4.4.2 订单页面及购物车页面的实现**

实现读出数据主要是通过查询数据库将数据封装为JavaBean对象，然后封装为List存入request的域中，转发到订单前端页面进行展示。在jsp页面中使用EL+JSTL表达式进行数据的展示。这项功能是比较容易实现的，和主页面几乎一致。

下面主要介绍本页面的分页实现方法。其主要是利用数据库的分页查询limit关键字，在后端进行分页查询，而不是前端。在前端每次点击数字或者下一页箭头都记录此次数据库查询的初始值，limit查询参数有两个，第一个是从第几个开始查询，第二个是查询几个，第二个参数会固定不变，即每一页展示的数目5行，第一个参数则是刚才传入的参数。传入Servlet中后，通过Dao对象，调用数据库查询，最后将查询结果存入request域对象转发即可。同时将页面的信息保存到域对象中，比如当前是第几页，当前的顶部标签是哪一个，整个订单一共多少页等重要数据。在转发到前端后，首先要做得就是初始化刚才后端传过来的数据，然后调用选择页面和顶部标签的choose()和change()方法，注意这两个方法是自己实现的方法，调用这两个方法后就可以将刚才的数据展示出来了。

4.5 数据库设计

**4.5.1 对数据库进行设计**

根据数据库设计分析，要实现上面的所有功能模块，共需要设计五个表，它们分别是：用户表、商品表、类别表、订单表、订单项表，具体如下表。

表1 用户表（USER）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 数据长度 | 允许为空 | Primarykey |
| sid | 学号 | varchar | 20 | N | Y |
| sname | 姓名 | varchar | 20 | N | N |
| password | 密码 | varchar | 16 | N | N |
| sex | 性别 | varchar | 4 | Y | N |
| age | 年龄 | int | / | Y | N |
| address | 地址 | varchar | 50 | N | N |
| email | 邮箱 | varchar | 20 | N | N |
| phone | 手机号 | varchar | 11 | N | N |
| payid | 支付账号 | varchar | 20 | N | N |
| nickname | 昵称 | varchar | 20 | Y | N |

表2 商品表（COMMODITY）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 数据长度 | 允许为空 | Primarykey |
| cid | 商品编号 | varchar | 20 | N | Y |
| cname | 商品名称 | varchar | 40 | Y | N |
| cpicture | 商品图片 | varchar | 80 | Y | N |
| cnum | 数目 | int | / | Y | N |
| cprice | 单价 | float | / | Y | N |
| cstate | 状态 | varchar | 10 | Y | N |
| category | 类别名 | varchar | 20 | Y | F |
| remarks | 备注 | varchar | 100 | Y | N |
| seller | 卖家学号 | varchar | 20 | Y | F |

表3 类别表（CATEGORY）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 数据长度 | 允许为空 | Primarykey |
| category | 商品类别 | varchaer | 20 | N | Y |

表4 订单表（ORDER）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 数据长度 | 允许为空 | Primarykey |
| oid | 订单编号 | varchar | 20 | N | Y |
| buyer | 买家学号 | varchar | 20 | Y | F |
| seller | 卖家学号 | varchar | 20 | Y | F |
| sum | 订单金额 | float | / | Y | N |
| ostate | 订单状态 | varchar | 10 | Y | N |
| time | 下单时间 | datetime | / | Y | N |

表5 订单项表（OrderItems）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 描述 | 数据类型 | 数据长度 | 允许为空 | Primarykey |
| oid | 订单编号 | varchar | 20 | N | Y and F |
| cid | 商品编号 | varchar | 20 | N | Y and F |
| qyt | 数量 | int | / | Y | N |
| subtotal | 小计 | float | / | Y | N |

# **五、关键部分源代码**

## 5.1、JSTL+EL页面展示源代码

这类代码主要是用于展示，在项目中经常使用，这里就以主页面的JSTL+EL为例进行说明。代码如下：

<!-- 床上用品 -->  
<div class="row goods-class">  
 <c:forEach items="${beddingList}" var="mer" varStatus="s">  
 <c:if test="${s.index==0}">  
 <div class="row">  
 <h2 class="col-xs-12 good-title">  
 &gt;&gt;&gt;&gt;&gt;床上用品&lt;&lt;&lt;&lt;&lt;  
 </h2>  
 </div>  
 </c:if>  
 <c:if test="${s.index<8}">  
 <div class="col-xs-3 goods-items">  
 <div class="thumbnail">  
 <img src=".${mer.merRes}" width="224" height="200">  
 <div class="caption">  
 <h3 class="merName" id="bed1-${s.index}">${mer.merName}</h3>  
 <div class="merDes">${mer.merDes}</div>  
 <p id="bed2-${s.index}">￥${mer.merPrice}</p>  
 <p>  
 <button name="${s.index}" class="btn btn-primary" role="button"  
 onclick="addToCart('bed',this)" style="outline: none">加入购物车  
 </button>  
 <button name="${s.index}" style="outline: none" tag="${mer.merID}" class="btn btn-default" role="button"  
 onclick="buyItem('bed',this)">  
 立即购买  
 </button>  
 </p>  
 </div>  
 </div>  
 </div>  
 </c:if>  
 </c:forEach>  
</div>

当该类商品没有商品时，上述黄色的部分，不会被执行，因为集合为空故下面的展示也不会被解析，因此商品缺货时，商品类别的标题不会被显示。

另外由于每个商品都具有加入购物车和立即购买按钮，因此为了区别这个按钮，点击按钮时能够区别其对应的商品，这里使用了EL结合<c:forEach>标签的循环状态给对应商品的信息取id或者自定义属性，见上述代码蓝色部分所示。 这样做的目的就是点击按钮时，获取按钮的name属性的值，这个值就是循环状态的下标，肯定每个商品不会一样，这个值也是商品id的后缀，就是为了区别不同的id。按钮的点击事件onclick="addToCart('bed',this)"*，*里面存在两个参数，第一个就是商品的类别，第二个就是按钮的this对象，用于获取按钮的name属性，结合jquery就可以唯一确定点击按钮对应的商品。代码如下：

function addToCart(str, obj) {  
 $("#cover908").css("display", "block");  
 $(".chooseCnt").css("display", "block");  
 let index = $(obj).attr("name");  
 //alert(index);  
 merID = $(obj).attr("tag");  
 //alert(merID);  
 let id1, id2;  
 switch (str) {  
 case "book":  
 id1 = "#book1-" + index;  
 id2 = "#book2-" + index;  
 break;  
 case "pen":  
 id1 = "#pen1-" + index;  
 id2 = "#pen2-" + index;  
 break;  
 case "sport":  
 id1 = "#sport1-" + index;  
 id2 = "#sport2-" + index;  
 break;  
 case "bed":  
 id1 = "#bed1-" + index;  
 id2 = "#bed2-" + index;  
 break;  
 case "clo":  
 id1 = "#clo1-" + index;  
 id2 = "#clo2-" + index;  
 break;  
 case "shuma":  
 id1 = "#shuma1-" + index;  
 id2 = "#shuma2-" + index;  
 break;  
 case "other":  
 id1 = "#other1-" + index;  
 id2 = "#other2-" + index;  
 break;  
 }  
 merName = $(id1).html();  
 merPrice = $(id2).html();  
 merPrice = merPrice.substring(1, merPrice.length);  
 console.log(merName + " " + merPrice);  
}

可见，首先通过传过来的this对象获取按钮的name属性，获取的name属性就是商品在该类别的索引，之后通过传过来的另一个参数选择具体的某类商品，组合成商品的id，这时同类商品的id前缀相同，通过后缀区分出不同的商品，这时就可以对该类商品进行自己想要的操作了，只需通过id得到标签的值，再传入后端操作即可。

该项目未登录时顶部不会显示用户的头像，当用户登录后会显示用户对应的头像。这部分也是通过JSTL显示的，其具体代码如下：

<c:if test="${isLogin==true}">  
 <li><img src="./res/headimg/${user.user\_id}.jpg" alt="" width="50" height="50" style="border-radius: 25px;"></li>  
</c:if>

既然，展示的数据是从request的域对象中读取的，那么接下来我们就看看后端的操作，后端Servlet是和JSP打交道的类，其主要工作仅三步，读取参数，执行操作，返回结果。初始化主页面的InitIndexServlet代码如下：

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入了初始化index.jsp...");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
  
 //调用Dao层查询  
 MerchandiseDao merchandiseDao = new MerchandiseDao();  
 //查找各类书籍  
 List<Merchandise> booksList = merchandiseDao.findAbout("各类书籍");  
 req.setAttribute("booksList",booksList);  
 //查找文具用品  
 List<Merchandise> stationeryList = merchandiseDao.findAbout("文具用品");  
 req.setAttribute("stationeryList",stationeryList);  
 //查找运动器材  
 List<Merchandise> sportsList = merchandiseDao.findAbout("运动器材");  
 req.setAttribute("sportsList",sportsList);  
 //查找床上用品  
 List<Merchandise> beddingList = merchandiseDao.findAbout("床上用品");  
 req.setAttribute("beddingList",beddingList);  
 //查找穿搭物品  
 List<Merchandise> clothesList = merchandiseDao.findAbout("穿搭物品");  
 req.setAttribute("clothesList",clothesList);  
 //查找数码物品  
 List<Merchandise> digitalList = merchandiseDao.findAbout("数码物品");  
 req.setAttribute("digitalList",digitalList);  
 //查询其他物品  
 List<Merchandise> othersList = merchandiseDao.findAbout("");  
 req.setAttribute("othersList",othersList);  
 req.getRequestDispatcher("main.jsp").forward(req,resp);  
}

可以看到，分为不同的类别查询，存入不同的域对象，在转发到main.jsp中进行展示即可，另外，本项目Servlet的路径配置使用的是index.jsp，注意这不是一个JSP页面，这是一个Servlet，因此项目一运行，该Servlet就会被执行，最后转发到一个main.jsp的页面进行展示。

这里存在一个巧妙之处，那就是转发具有一个十分重要的特点，那就是转发属于一次请求，不会过多浪费资源，另外重要的是转发其URL统一资源定位符是不会发生改变的，简而言之就是web浏览器上的网址栏不会发生变化，但是显示的内容却是main.jsp的内容。这样做的好处就是下次刷新时，还是会执行Servlet并转发到main.jsp显示，因此当后端更改数据库时，前端只需要刷新一下即可同步过来，不用等待重新部署。

## **5.2 个人主页面代码设计**

个人主页面的代码展示JSTL和EL进行展示的，思路和上面说的方法几乎一致，这里主要介绍个人主页的另一个功能修改密码以及绑定手机和邮箱。

### 5.2.1 修改密码

修改密码设计的前端操作就是检测新密码是否一致以及其是否为弱密码，主要是通过JS进行检测的，其检测代码如下：

function checkPwd() {  
 //获取密码  
 let password = $("#new-password").val();  
 if (password.length < 8) { //密码小于八位不合格  
 return false;  
 }  
 //密码是否为纯数字  
 for (let i = 0; i < password.length; i++) {  
 if (password[i] < '0' || password[i] > '9') {  
 return true;  
 }  
 }  
 return false;  
}

主要检测是否为8位数以下以及是否为纯数字，检测成功就会发送到后端进行进一步操作。

进入后端后主要进行的操作就是先检测原密码是否正确，也就是一次查询，修改数据库中的密码字段，也就是一次修改，因此修改密码在后端进行的操作就分为一次查询和一次修改。

其代码如下：

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入修改密码Servlet...");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
  
 //获取参数  
 String email = req.getParameter("email");  
 String old\_password = req.getParameter("old\_password");  
 String new\_password = req.getParameter("new\_password");  
 System.out.println("email: "+email+",old\_password: "+old\_password+",new\_password: "+new\_password);  
  
 //验证登录  
 UserDao userDao = new UserDao();  
 User user = userDao.login(email, old\_password);  
 if(user==null){  
 out.print("原密码错误");  
 return;  
 }  
 //修改密码  
 boolean b = userDao.modifyPassword(email, new\_password);  
 if(b){  
 out.print("修改成功,请重新登录");  
 req.getSession().removeAttribute("user");  
 req.getSession().removeAttribute("isLogin");  
 }else{  
 out.print("服务器错误,代码9821");  
 }  
}

### 5.2.2 绑定手机和邮箱

绑定手机和邮箱用到的主要技术就是发送邮件和手机验证码技术，这里只给出代码，至于发送手机验证码以及邮箱验证码使用的SMS和SimpleEmail技术将在后面陆续介绍。

绑定手机和邮箱对应Servlet的代码如下：

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入修改手机Servlet...");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
  
 //获取参数  
 String user\_id = req.getParameter("user\_id");  
 String email\_phone = req.getParameter("email\_phone");  
 System.out.println(user\_id+","+email\_phone);  
  
 //调用dao层修改  
 UserDao userDao = new UserDao();  
 boolean b = userDao.modifyEmailPhone(user\_id, email\_phone);  
 User user = userDao.findUserByUID(user\_id);  
 req.getSession().setAttribute("user",user);  
 out.print(b);  
 System.out.println("modifyPhone: "+b);  
}

具体的实现细节这里不做过多的解释说明，将会在下一章详细说明。

## 5.3 登录注册页面

登录注册功能其实就是在数据库中添加一条满足完整性约束的用户记录，在前端会检测一部分的用户自定义完整性，比如非空等等。提示用户密码或者账户等信息不能为空。

### 5.3.1 注册前端功能

其前端的检测也主要是JS实现的检测，分为邮箱手机号格式与密码等的检测。

* 邮箱手机号的前端检测

/\*检测邮箱和手机\*/  
function checkEmail\_phone() {  
 let sign\_ep\_tag = false;  
 let email\_phone = $("#sign-email-phone").val();  
 //先检测是不是电话  
 let reg = /^1[3458]\d{9}$/;  
 sign\_ep\_tag = reg.test(email\_phone);  
 if (sign\_ep\_tag) return sign\_ep\_tag;  
 //再检测是不是邮箱  
 reg = /^\w+@[a-zA-Z0-9]{2,10}(?:\.[a-z]{2,4}){1,3}$/;  
 return reg.test(email\_phone);  
}

此处的检测使用的不是遍历检测，使用的名为正则表达的检测方法，我们可以看到红色部分正则表达式1为/^1[3458]\d{9}$/，其中**“/”**是JS中正则表达式的起始符和结束符，^表示限定开头，1表示第一位必须是1，这是手机号共同的特征，（注意这里的手机号指三大运营商移动、联通、电信提供的手机号），[3458]表示其手机号第二位必须是3、4、5、8中的某一位，\d{9}表示后面的9位必须是数字，最后的$表示限定结尾，^和$的主要作用就是是JS不匹配包含手机号的数字串，比如5655651531010265116514，虽然包含了满足手机号正则的字符串，但是由于使用了限定开头和结尾，因此不会匹配成功。

* 密码的前端检测

/\*检测密码\*/  
function checkPwd() {  
 //获取密码  
 let password = $("#sign-password").val();  
 if (password.length < 8) { //密码小于八位不合格  
 return false;  
 }  
 //密码是否为纯数字  
 for (let i = 0; i < password.length; i++) {  
 if (password[i] < '0' || password[i] > '9') {  
 return true;  
 }  
 }  
 return false;  
}

这里检测密码没有使用正则表达式，就是普通的JS代码逐位判断。

* 检测协议勾选

function checkAgree() {  
 return $("#sign-agree").prop("checked");  
}

只有这三项前端JS做的检测通过了才会向后端发起一个请求，请求会分为两次发送，第一次发送为检测该手机号或者邮箱是否被注册过，其因为不用刷新页面，因此使用的是Ajax请求，代码如下：

/\*提示信息\*/  
valid = true;  
interval = setInterval(function () {  
 time--;  
 $("#sign-sendCode").html(time + "s后重新发送");  
 if (time == 0) {  
 time = 120;  
 clearTimeout(interval);  
 valid = false;  
 $("#sign-sendCode").html("重新发送");  
 }  
}, 1000);  
//最后一个检测--邮箱是否被注册过  
$.post("checkEmailServlet", {  
 email\_phone: $("#sign-email-phone").val()  
}, function (date) {  
 //回调函数  
 if (date == "false") {  
 $.post("sendCodeServlet", {  
 email\_phone: $("#sign-email-phone").val()  
 }, function (date) {  
 //回调函数  
 checkCode = date;  
 // alert(checkCode);  
 }, "text");  
 } else {  
 alert("该手机或邮箱已被注册,请直接登录或者找回密码");  
 }  
}, "text");

以上代码红色部分是实现点击发送倒计时功能的，其主要作用是提示用户验证码有效期的剩余时长。

下面的就是发送给后端进行用户邮箱和手机号检测的代码。

* 发送注册请求给后端

/\*注册按钮的事件\*/  
function sign() {  
 //获取邮箱或手机号  
 let email\_phone = $("#sign-email-phone").val();  
 //获取密码  
 let password = $("#sign-password").val();  
 //获取用户输入的验证码  
 let checkCode\_ = $("#sign-checkCode").val();  
 if (valid && checkCode\_ == checkCode) {  
 //关闭按钮的点击事件  
 document.getElementById("sign").disabled = true;  
 //发送注册请求  
 $.post("signServlet", {  
 email\_phone: email\_phone,  
 password: password  
 }, function (date) {  
 if (date == "true") {  
 alert("注册成功!");  
 } else {  
 alert("服务器错误,代码8214");  
 }  
 }, "text");  
 document.getElementById("sign").disabled = false;  
 } else if (valid) {  
 alert("验证码错误");  
 } else {  
 alert("验证码过期");  
 }  
}

完整性检测和重复检测完成后，就会发送验证码，用户填写正确的验证码后就会发送到后端进行注册。至此前端的事情就已经干完了，只需等待后端响应成功后返回即可。

### 5.3.2 注册后端功能

用户点击注册后通过Ajax发送一个请求到Servlet执行插入数据库功能，这部分操作主要是由Servlet控制，由Dao层进行具体的操作。代码如下：

@Override  
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入注册服务器:");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
 //获取传来的数据  
 String email\_phone = req.getParameter("email\_phone");  
 String password = req.getParameter("password");  
 String remember = req.getParameter("remember");  
 System.out.println("开始注册 : ");  
 //调用dao层注册  
 UserDao userDao = new UserDao();  
 System.out.println("new 了对象");  
 System.out.println("email\_phone: "+email\_phone+",passowrd: "+password+",remember: "+remember);  
 boolean b = userDao.signUser(email\_phone, password);  
 if(b&&remember.equals("true")){  
 //记住密码  
 }  
 out.print(b);  
}

可以看到，其主要是调用Dao层的方法执行具体的注册功能，下面展示UserDao层的signUser()方法：

public boolean signUser(String email\_phone, String password) {  
 //判定邮箱还是手机  
 String column = email\_phone.indexOf('@') == -1 ? "mobile\_phone" : "email";  
  
 int update = template.update("insert into user(user\_id," + column + ",password) values(?,?,?)"  
 , new Object[]{generateUserID(), email\_phone, password});  
  
 return update > 0;  
}

我们可以看到，其中也不是自己的代码，而是使用一种名为JdbcTemplate的技术封装的方法update()去操作的数据库，该技术下一章重点讲解。

## 5.4、加入购物车以及立即购买

加入购物车主要也是后端操作数据库，其中加入购物车不用检测库存，在购买时会检测库存是否足够，并且通过数据库触发器实现库存自动减少。

其中触发器的创建语句如下：

DROP TRIGGER IF EXISTS check\_merCnt;

DELIMITER $$

CREATE TRIGGER check\_merCnt

BEFORE INSERT

ON orderstatus

FOR EACH ROW

BEGIN

IF new.cnt>(SELECT merCount FROM merchandise WHERE merID=new.merID) THEN

signal SQLSTATE '45000' SET message\_text = '库存不足';

END IF;

END$$

DELIMITER ;

DROP TRIGGER IF EXISTS auto\_upmerCnt;

DELIMITER $$

CREATE TRIGGER auto\_upmerCnt

AFTER INSERT

ON orderstatus

FOR EACH ROW

BEGIN

UPDATE merchandise SET merCount=merCount-new.cnt WHERE merID=new.merID;

END$$

DELIMITER ;

第一个触发器是购买前判断库存的，第二个触发器是购买后自动更新库存的，在后端捕获到这个异常就会输出库存不足的信息。

这两个功能的前端主要是发出Ajax请求，交给后端进行数据的处理，操作数据库。后端代码如下：

* 加入购物车后端代码

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入加入购物车...");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
  
 //获取参数  
 String user\_id = req.getParameter("user\_id");  
 String merName = req.getParameter("merName");  
 Double merPrice = Double.parseDouble(req.getParameter("merPrice"));  
 Integer cnt = Integer.parseInt(req.getParameter("cnt"));  
 System.out.println("user\_id : " + user\_id + ",merName : " + merName + ",merPrice : "  
 + merPrice + ",cnt : " + cnt);  
  
 //调用Dao层  
 MerchandiseDao merchandiseDao = new MerchandiseDao();  
 boolean b = merchandiseDao.addToCart(user\_id, merName, merPrice, cnt);  
 System.out.println(b);  
 out.print(b);  
}

具体的Servlet功能是依靠Dao层实现的，这里暂不介绍Dao层如何操作数据库。

* 立即购买后端代码

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 System.out.println("进入BuyServlet...");  
 //设置数据格式  
 req.setCharacterEncoding("utf-8");  
 resp.setContentType("text/html;charset=UTF-8");  
 resp.setCharacterEncoding("utf-8");  
 PrintWriter out = resp.getWriter();  
  
 //获取参数  
 String user\_id = req.getParameter("user\_id");  
 String status = req.getParameter("status");  
 String merID = req.getParameter("merID");  
 Integer cnt = Integer.parseInt(req.getParameter("cnt"));  
  
 //调用Dao层  
 OrderDao orderDao = new OrderDao();  
 boolean b = orderDao.buyItem(user\_id, merID, status, cnt);  
 out.print(b);  
}

到这里，基本的功能就介绍完毕了，下面会详细介绍使用的拓展技术，也就是老师上课未曾讲过，但是十分好用的技术。

## 5.6 BootStrap

本次项目的前端布局，大多数是使用的BootStrap前端框架3版本，这样做的好处就是，不用自己麻烦的书写样式，在别人的框架下进行修改也是比较符合实际开发的需求。所谓的框架可以看成一个半成品软件，程序员只需在这款软件上加入自己的构思和想法即可开发自己的应用或者网页。

Bootstrap的布局主要是使用类来定义样式，比如按钮，表单等等，另外，有一个很有趣的地方，那就是，Bootstrap支持响应式开发，所谓响应式开发就是在不同的设备上显示的布局不完全相同，这样就只需要开发一套代码，通过响应式布局实现在不同设备手机、iPad、笔记本、台式电脑上展示，提高了代码的复用性。

其响应式布局主要依赖的就是Bootstrap的栅格系统，其将浏览器分为行，每行分为12个格子，通过类名row定义行，通过类名col-xx-num定义列，其中xx表示设备的代码，其取值可以是xs、sm、md、lg分别表示手机、iPad、笔记本、台式机的屏幕尺寸适应，num表示每行占多少个格子，其取值为1到12。

## 5.6 Junit单元测试

单元测试这个概念应该大家都比较熟悉，在开发应用中，会经历测试阶段，测试主要分为黑盒测试和白盒测试，本项目主要使用的就是白盒测试，所谓黑白盒的区别就在于是否可以看到源代码，而白盒测试就是在已知源代码的情况下进行的测试。

我们都知道，Java的代码必须在主函数运行，因此我们为了完成白盒测试不得不新建一个test包，在里面书写具有主方法的类进行测试，但是这样我们会发现，不仅效率高，还会比较麻烦，因为每测一个功能就要新建一个类，后期测试完还要删掉测试类，十分繁琐。

因此，本项目使用Junit单元测试，其使用的包在pom.xml中引入，通过给测试方法加Test注解即可运行，其中测试方法必须满足三个条件

* 测试方法必须是public修饰的方法
* 测试方法的返回值必须是void
* 测试方法的参数必须为空

满足这三个基本条件的方法就可以添加Test注解进行测试。

下面给出一个添加了Test注解的测试方法：

@Test  
public void test() {  
 int size = getSize("DXtFzTM6wQ", "未付款");  
 System.out.println(size);  
}

## 5.7 Druid数据库连接池

数据库操作上，我使用的是MySQL数据库，并且使用了阿里巴巴提供的数据库连接池技术Druid。该技术可以方便的管理数据库的连接，只需要通过配置文件配置，通过读取配置文件内容就可以新建数据库连接池，每次需要获取连接就只需要从池中取出一个即可。

Druid数据库连接池的配置文件Druid.propertites如下：

driverClassName=com.mysql.cj.jdbc.Driver  
url=jdbc:mysql://127.0.0.1:3306/salesystem  
username=sale  
password=sale\_admin  
initialSize=5  
maxActive=10  
maxWait=2000

其中，driverClassName就是MySQL驱动的全类名；url就是连接数据库的统一资源定位符；username和password就是数据库的账号以及密码；initialSize是初始化服务池的连接数，maxActive是数据库池最大的连接数，maxWait是最长等待时间2s。

下面是MySQL工具类的代码，其主要就是获取数据库连接池的DataSource类对象。

/\*\*  
 \* 数据库连接工具  
 \* 包含获取连接，获取池，释放连接功能  
 \*/  
public class MySQLUtils {  
 private static DataSource dataSource=null;  
 /\*\*  
 \* 静态代码块，初始化静态成员  
 \* \*/  
 static {  
 try {  
 //加载配置文件  
 Properties properties = new Properties();  
 properties.load(MySQLUtils.class.getClassLoader().getResourceAsStream("druid.properties"));  
 //获取数据源  
 dataSource = DruidDataSourceFactory.createDataSource(properties);  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 /\*\*  
 \* 获取连接池的方法  
 \* @return 返回连接池  
 \*/  
 public static DataSource getDataSource(){  
 return dataSource;  
 }  
}

红色部分使用静态代码块初始化静态成员dataSource，其主要是读取配置文件，交给Druid数据源工厂从而获取数据源。

绿色代码部分是获取数据源，以便在后面的JdbcTemplate中使用。

## **5.8 JdbcTemplate**

我们都知道本项目主要是和数据库进行操作，为了和数据库更好的操作，我们使用了JdbcTemplate封装的一些方法，该模板封装了查询和更新等方法，从获取数据库连接到释放连接都封装好了。

JdbcTemplate的初始化：

private JdbcTemplate template = new JdbcTemplate(**MySQLUtils.getDataSource()**);

JdbcTemplate对象的构造方法接受一个数据库池的数据源，这个就是上述的dataSource，因此通过MySQLUtils获取数据源即可。

下面介绍本项目使用到的几个封装好的实用的方法。

* 更新方法update()

int update = template.update("insert into user(user\_id," + column + ",password) values(?,?,?)"  
 , new Object[]{generateUserID(), email\_phone, password});

以这个例子来看，我们可以看到，其第一个参数就是需要执行的SQL语句，其中参数可以使用？代替；第二个参数是一个Object的数组，其数组内容就是刚才问号依次对应的内容；其返回值就是执行完这条更新语句所影响到的行数。

* 查询方法query()

List<User> userList = template.query("select \* from user where " + column + " = ?",  
 new BeanPropertyRowMapper<User>(User.class),  
 new Object[]{email\_phone});

以这个例子为例，我们看到可以将查询结果封装为JavaBean的List对象。需要注意的是需要满足两点要求：

* JavaBean对象的属性要和表中的字段一一对应，特别是字段类型
* JavaBean必须具有getter和setter方法

我们可以看到，template.query()的第一个参数同样是一条SQL语句，参数使用？代替；第二个参数就是一个BeanPropertyRowMapper对象，就是这个对象将数据库的内容读出转化为JavaBean的List对象，这个类是一个泛型，其模板应该是对应的JavaBean，其构造方法需要接收一个类的字节码文件，于是我们传入对应JavaBean的class字节流；第三个参数就是和问号对应的参数列表，其为Object的数组。

# 六、总结

本次项目的完成，学会了如何开发一个Java Web的应用程序，明确了开发流程和步骤。

这个项目走了一遍web的开发流程，包括产品功能设计、数据库表的设计、页面代码设计、代码规范、网页功能操作流程、git协同开发等等，涵盖了开发的大部分流程，是一个不错的练手学习项目。整体功能和代码逻辑还是比较清晰的，主要是按照功能来分页面，按照功能的相关性把页面连接起来。核心围绕着增删查改这些操作，以及简单的权限管理。

首先，开发之前自己要做好需求分析，定向自己的产品将会是一个什么样的效果；做好需求分析之后就应该进入数据库的设计，其中分为概念模型的设计以及逻辑模型的转化，也就是画出ER图以及将其转化为对应的关系。接着就是进行前端页面的开发，其主要难点就是布局。前端开发完成后可进行后端的开发。最后可以对自己的项目进行移植性测试，或者部署到服务器找同学测试其效果。

对于后端的开发，整个设计模式就是MVC的设计模式，即Model模型层、View视图层、Controller控制层；其中Model层主要是JavaBean充当，View层主要是JSP页面充当，Controller主要是Servlet充当。当JSP展示的页面收到用户的一个点击操作时，将其首先传递给处理这个事件的Servlet，由Servlet控制调用具体的Dao层进行处理。

本次项目学会了Java Web的开发基本知识，能够基本的开发一个简易的交易系统或者管理系统，激发了自己的学习兴趣。

本次项目熟练使用了Servlet、JSP、MySQL、JDBC、Druid、JDBCTemplate、EL、JSTL等等相关技术，对这些技术有了初步的认识，明白其功能和使用方法。在最后的文档书写过程中，将自己的代码与书中的理论相结合，对软件工程学有了更加深刻的理解。

# 七、参考文献

[1]张继军,董卫,王婷婷.Java Web应用开发技术与案例教程[M].机械工业出版社:北京,2019.4:1-320.

[2]BootStrap.BootStrap中文网[EB/OL].https://www.bootcss.com/,.

[3]Maven.mvnrepository[EB/OL].https://mvnrepository.com/,.

[4]腾讯云.[EB/OL].https://cloud.tencent.com/?fromSource=gwzcw.2212127.2212127.2212127&utm\_medium=cpd&utm\_id=gwzcw.2212127.2212127.2212127,.

[5]zhiguo98.Druid数据库连接池[EB/OL].https://blog.csdn.net/yanzhiguo98/article/details/88981134,.

[6]https://www.bilibili.com/video/BV1qv4y1o79t

[7]https://www.bilibili.com/video/BV1s5411H7Kz?t=2572

[8]高星宇. 校园二手交易平台的设计与实现[J]. 福建电脑，2020, 36（2）：87-89.

[9]郑建华. 校园二手商品交易平台的设计与实现[J]. 计算机光盘软件与应用，2012（9）：183-184.

[10]刘芸琦. 九江职业大学二手商品交易系统设计与实现[D]. 电子科技大学，2014.

[11]贺欢. 大学生二手交易平台的现状调查与前景预测—以阜阳师范大学为例[J]. 产业，2021（8）：93-95.