

**NANCHANG UNIVERSITY**

**JavaEE web应用技术**



**题 目:** 软院小黑市购物系统

**专 业：** 软件工程

**班 级：** 软件工程1910班

**完成人数：** 1

**成 员：** 王俊洪

**学 号：**  8002119319

**任课教师：** 孙孟标

**完成时间：** 2021年6月15日

**说 明**

1. 本任务书由任课教师填写，下达到学生。
2. 完成任务后，任课教师需填写小结表。
3. 任务书、学生成绩与学生完成后的大作业（纸质和电子两种）一并报送各教学研究部审核后转教务办。
4. 另附一份全班大作业总结。

|  |
| --- |
| **大作业的要求和内容：（包括题目选择范围、技术要求、递交时间、考核方法等）**  一、实验目的  通过综合运用Html、CSS、JavaScript、JSP、JavaBean、Servlet、JDBC访问数据库、数据库设计等技术设计与实现一个自选项目，进一步理解和掌握Java Web编程开发的基本技术以及熟练使用开发工具。  运用SSH或SSM等框架技术进行设计亦可。  二、实验内容和要求  要求学生自定选题，可成立设计小组，原则上最多3人1组，完成需求分析，软件设计以及程序的编写、调试和测试。  设计和程序完成后，要求根据指定的格式要求，独自完成设计报告的撰写，撰写内容包括：  1、根据课设写出用户基本需求。（需求分析）  2、根据需求对系统进行分析与设计，并画出系统的用例图、时序图、程序结构图。  3、对系统中设计的关键算法进行设计，并用算法流程图或伪代码描述其步骤。  4、数据库设计  5、程序实现关键代码或主要组件的API。  6、准备足够的数据对设计的系统进行测试。  7、本人在设计团队中承担的工作描述，根据自己的工作分工，报告可以有不同的侧重点。  强调：同一小组成员的报告撰写不能相同，因为分工不同，报告中重点写自己承担的任务。另外，报告中不需要代码的大段粘贴，只需书写程序的关键代码或主要组件的API说明，运行测试图最多两张,设计报告严禁抄袭，否则按不及格论.  三、课程设计报告提交  书面报告、电子版报告和程序等文件请通过学习委员交给指导老师。  课程设计报告Word文件命名格式：学号+姓名+课程设计题目名.doc,程序或者其他文件压缩命名格式：学号+姓名+课程设计程序.rar。  提交时间：2021年6月18日。  四、实验报告排版要求  撰写报告应简明扼要，文理通顺，章节层次分明，图表清晰准确，8000-10000字即可即可，最多不能超过20页。不要为了凑字数而加大字号和行距。  字体字号：  文章题目用“黑体小二号字”，居中排列。  文中大标题用“宋体四号字”。  文中小标题和整个文章用“宋体小四号字”。  行距1.5倍。  页边距：上 2.5cm，下 2.5cm, 左 2.5cm(其中装订线0.5), 右 2cm,  对称页边距，单面A4打印。  要求有3篇以上的参考文献。  文献格式：  引用报刊文章的语句，依次注明作者姓名、篇名、报刊名称、年月日或期数、页码。  引用经典著作中的语句，要按作者姓名、书名（或篇名）、集名、卷数、版本（出版社名、出版年份）、页码的先后顺序注明。  如果是网络上的文章，给出作者或者网站名，文章名，网址  示例（字体5号宋体）：[1]王少华 蒋毅等编著，《ASP 基础教程》，国防工业出版社，2001 年5 月 第一版  成绩：  教 师 签 名：  教研部负责人：  学 生 姓 名： 王俊洪 |

目录

[一、项目介绍 1](#_Toc13928)

[1.1 项目背景 1](#_Toc14798)

[1.2 简介 1](#_Toc25882)

[二、分析与设计 2](#_Toc17065)

[2.1 功能分析 2](#_Toc19413)

[2.2 功能设计 4](#_Toc18998)

[2.3 项目结构 5](#_Toc17174)

[三、主要界面及实现 6](#_Toc30560)

[3.1 主页面 6](#_Toc19698)

[3.1.1 主页面设计 6](#_Toc18264)

[3.1.2 主页面实现 8](#_Toc12165)

[3.2、登录、注册及找回密码界面 8](#_Toc22220)

[3.2.1 登录、注册及找回密码界面设计 8](#_Toc20415)

[3.2.2 登录、注册及找回密码界面实现 10](#_Toc11777)

[3.3、个人主页 12](#_Toc14231)

[3.3.1 个人主页设计 12](#_Toc28340)

[3.3.2 个人主页实现 13](#_Toc19187)

[3.4 订单页面即购物车页面 14](#_Toc31200)

[3.4.1 订单页面及购物车页面的设计 14](#_Toc2253)

[3.4.2 订单页面及购物车页面的实现 15](#_Toc5343)

[四、关键部分源代码 16](#_Toc25164)

[4.1、JSTL+EL页面展示源代码 16](#_Toc4234)

[4.2 个人主页面代码设计 19](#_Toc16373)

[4.2.1 修改密码 19](#_Toc18996)

[4.2.2 绑定手机和邮箱 21](#_Toc19747)

[4.3 登录注册页面 21](#_Toc24000)

[4.3.1 注册前端功能 22](#_Toc9307)

[4.3.2 注册后端功能 24](#_Toc22816)

[4.4、加入购物车以及立即购买 25](#_Toc2883)

[五、使用的拓展技术 27](#_Toc1269)

[5.1 BootStrap 27](#_Toc9022)

[5.2 Junit单元测试 28](#_Toc13112)

[5.3 Druid数据库连接池 28](#_Toc23547)

[5.4 JdbcTemplate 30](#_Toc19467)

[5.5 SmsSingleSender及腾讯云短信服务 31](#_Toc8065)

[5.6 部署到真实服务器 32](#_Toc21024)

[六、总结 33](#_Toc11122)

[6.1 遇到的困难 33](#_Toc9054)

[6.2 个人总结 34](#_Toc27043)

[七、参考文献 34](#_Toc8107)

# 一、项目介绍

## 1.1 项目背景

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，我们一般会去软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。

# 二、分析与设计

## 2.1 功能分析

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图：



图1 项目需求图

项目逻辑功能如下图：

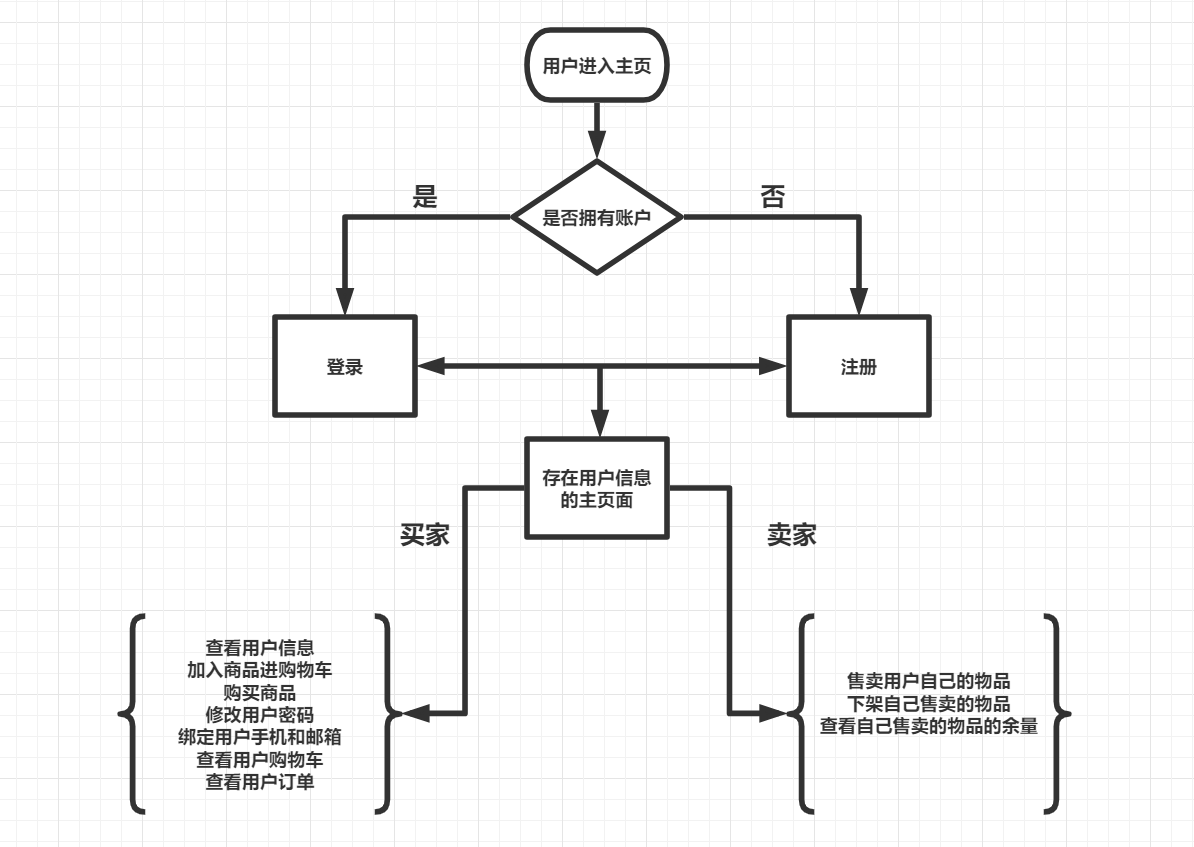


图2 逻辑功能图

## 2.2 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：

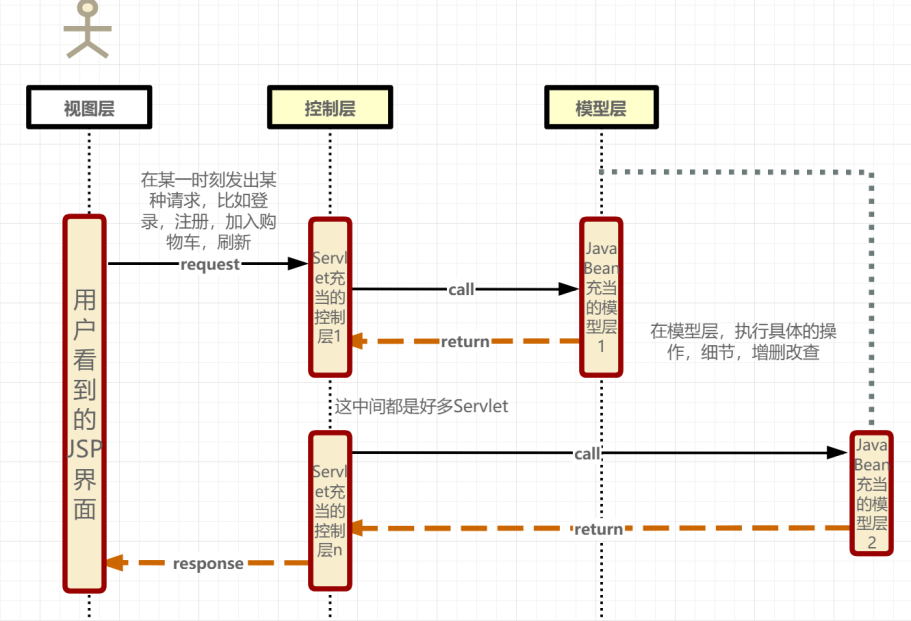


图3 项目时序图

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。

## 2.3 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：

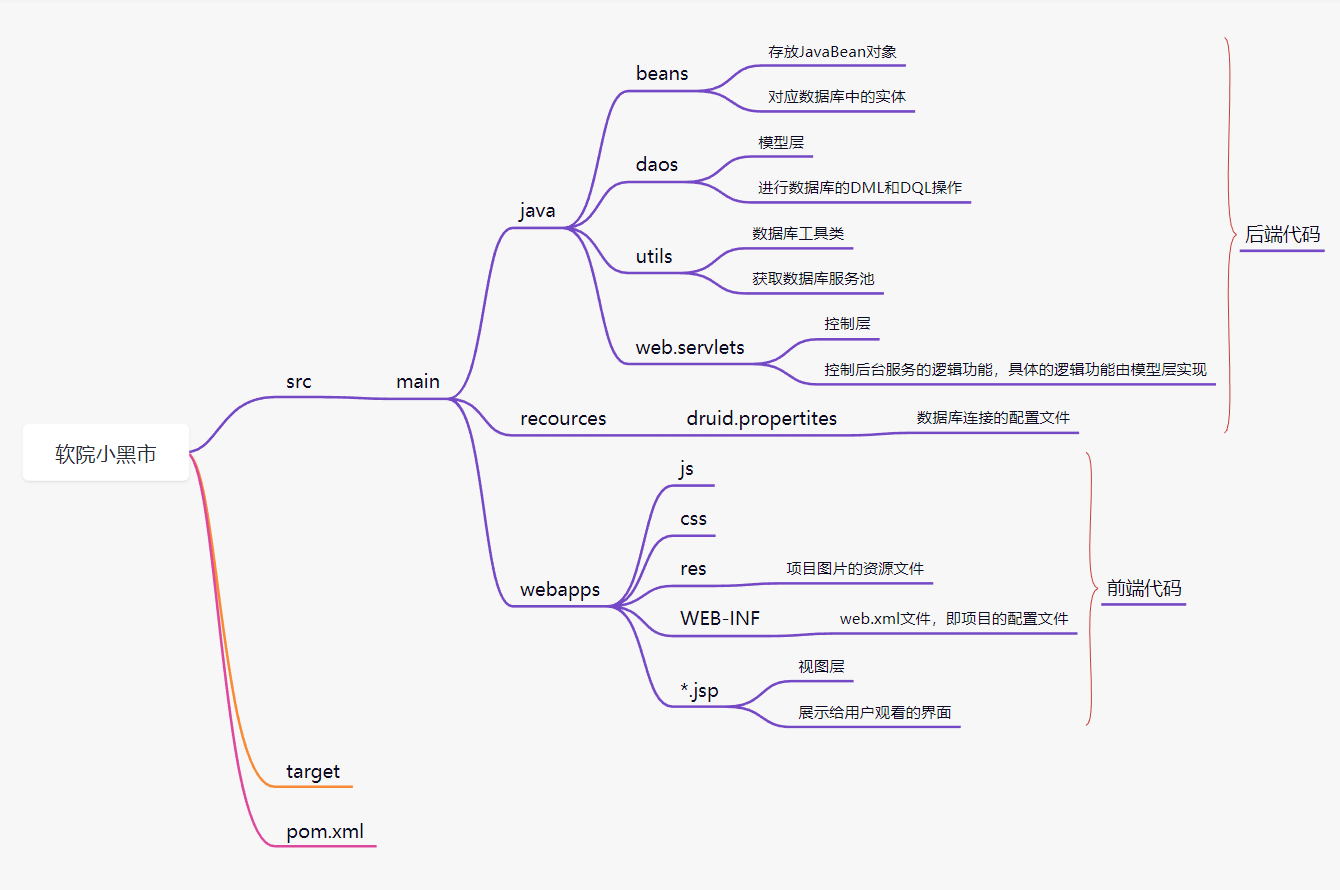


图4 工程结构图

# 三、主要界面及实现

## 3.1 主页面

### 3.1.1 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：

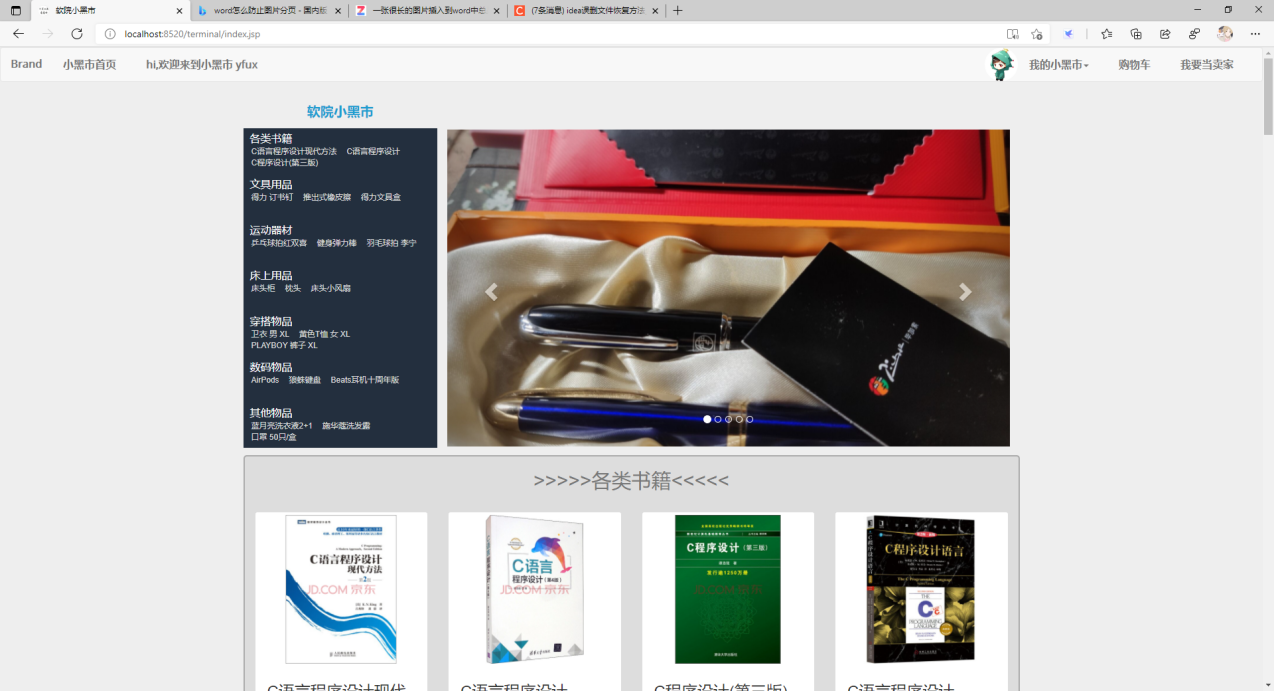


图5 主页面顶端即导航部分

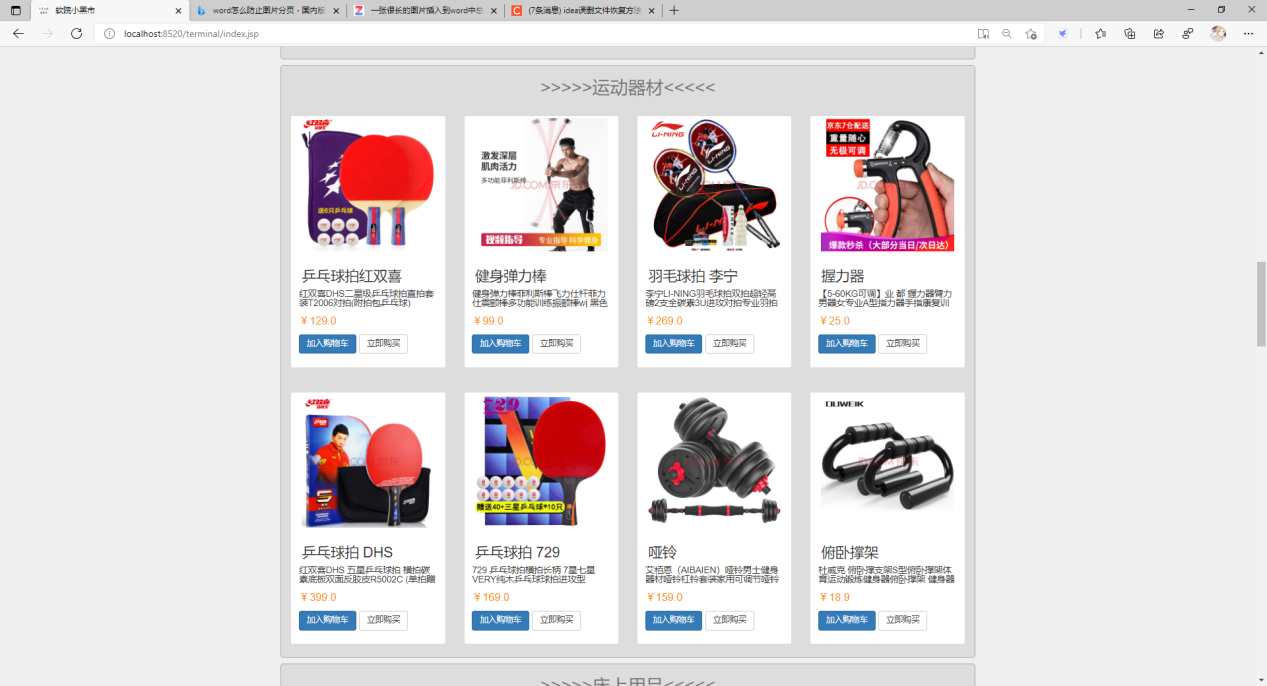


图6 商品部分



图7 主页面底部

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：



图8 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。

### 3.1.2 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。

## 3.2、登录、注册及找回密码界面

### 3.2.1 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。

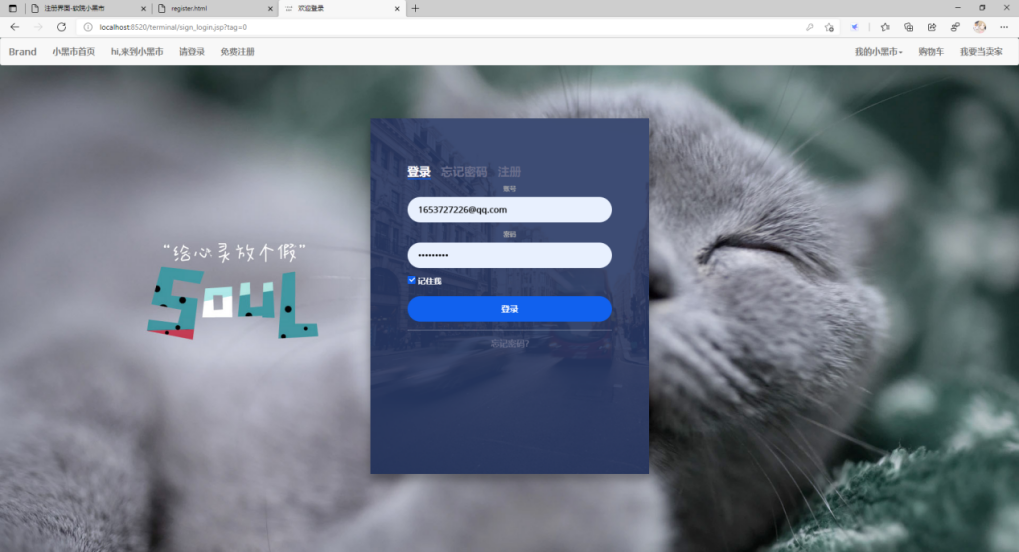


图9 登录页面

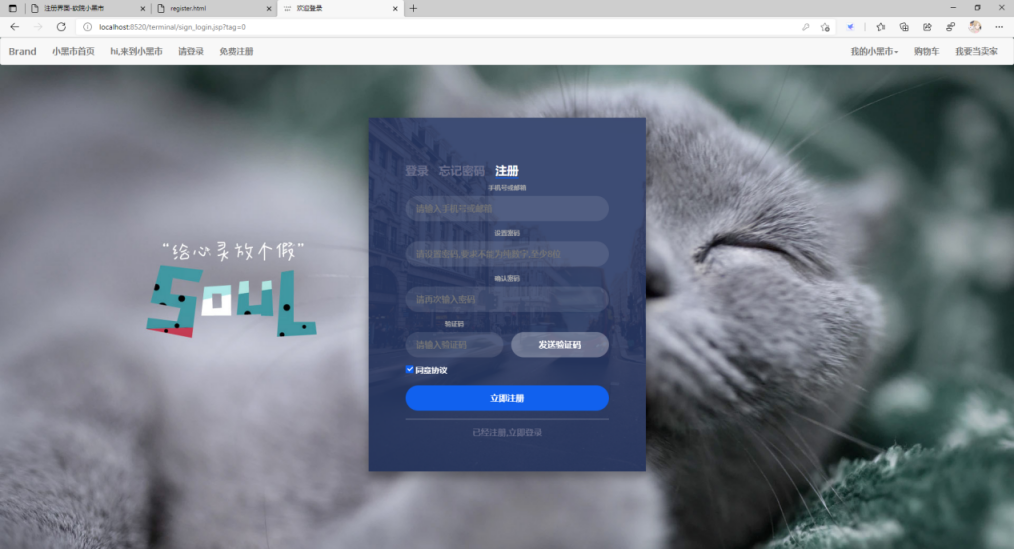


图10 注册页面

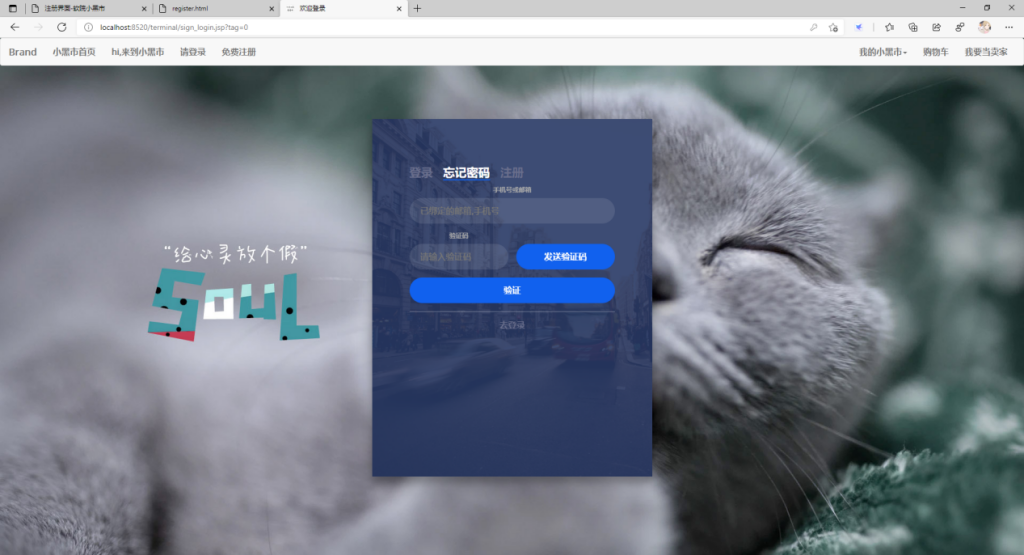


图11 找回密码

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。

### 3.2.2 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：

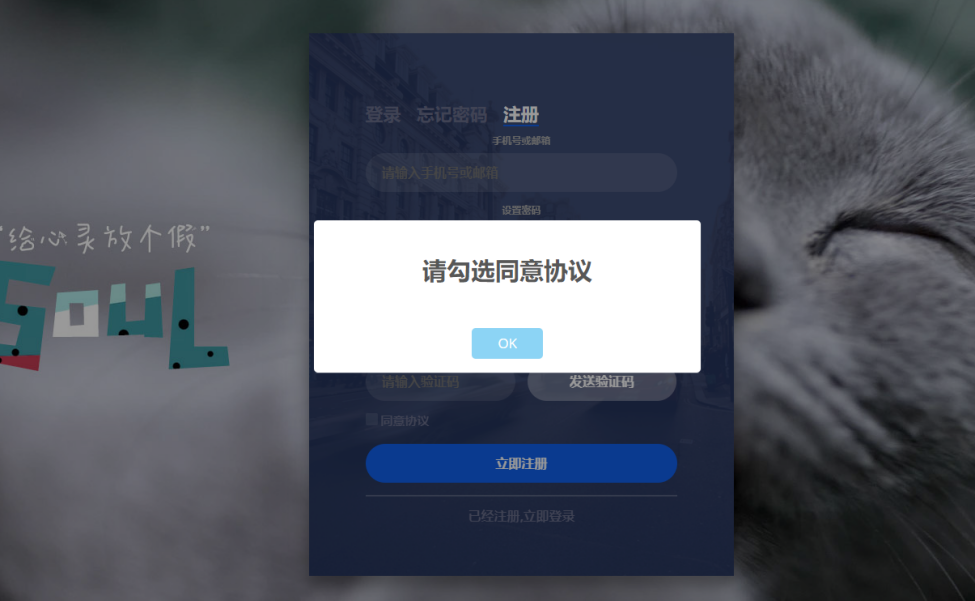


图12 未勾选同意协议



图13 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。

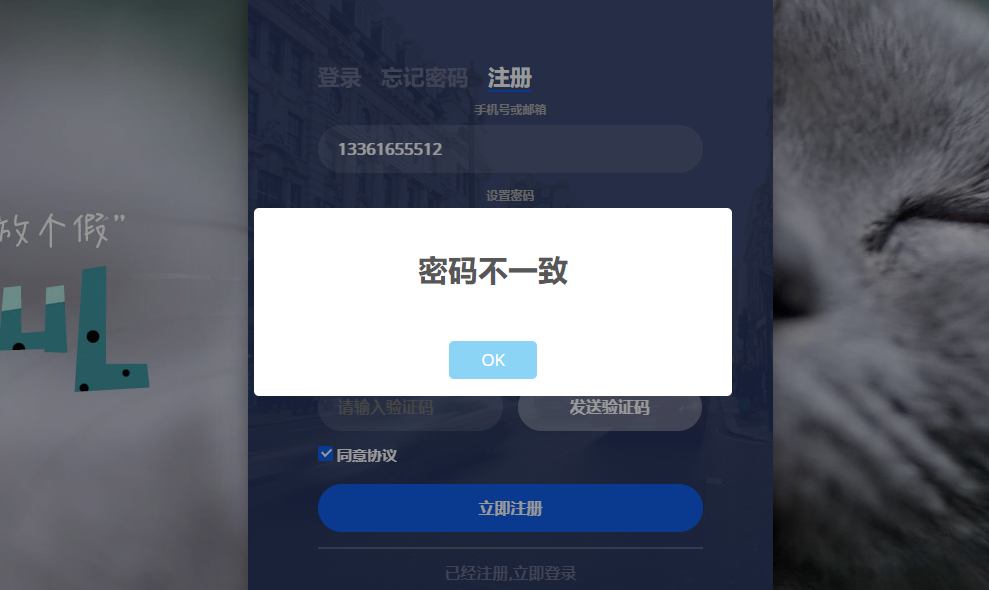


图14 密码不一致



图15 弱密码

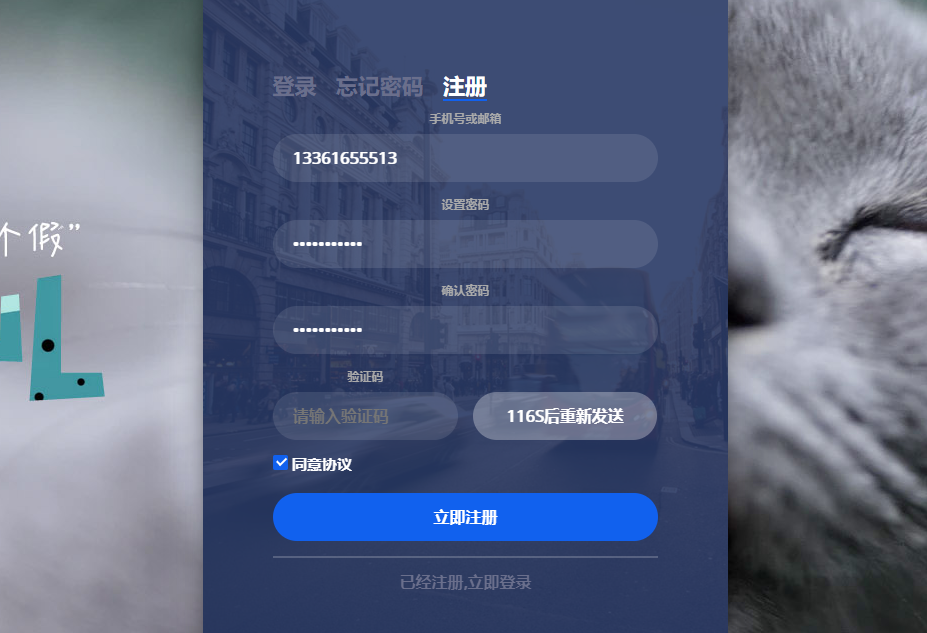


图16 发送成功会存在一个倒计时

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。



图17 验证码过期



图18 显示重新发送

## 3.3、个人主页

### 3.3.1 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：

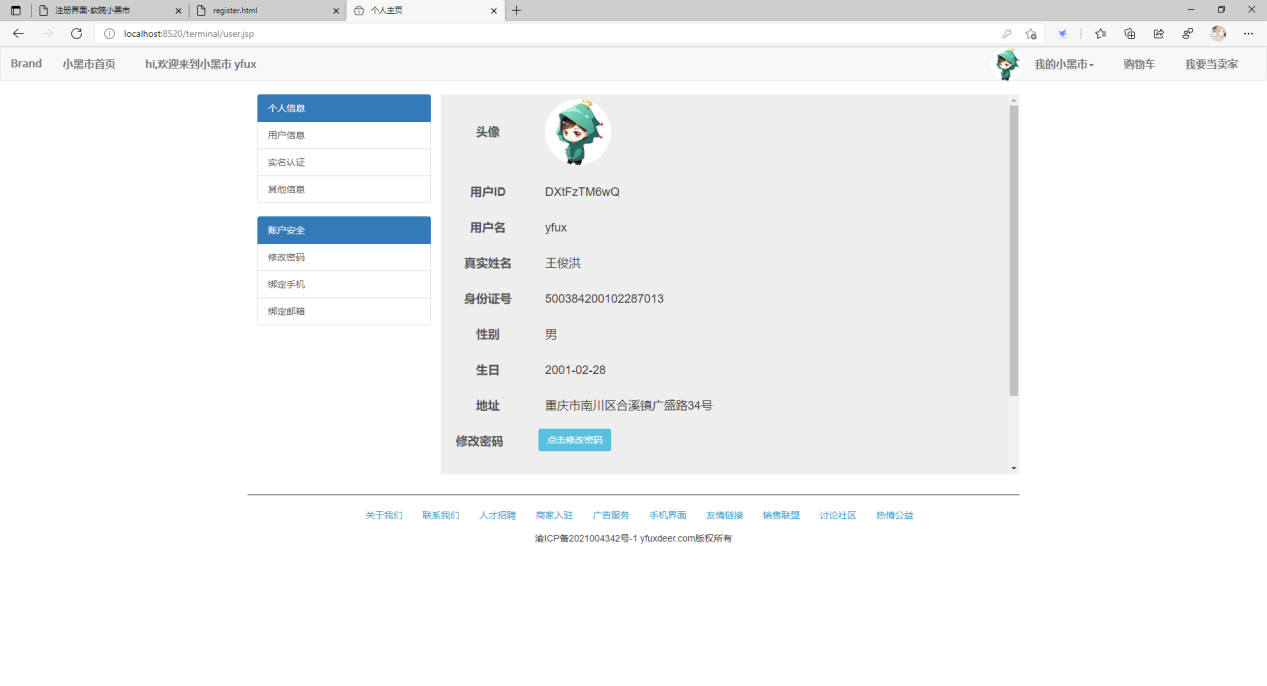


图19 个人主页

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。

### 3.3.2 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。

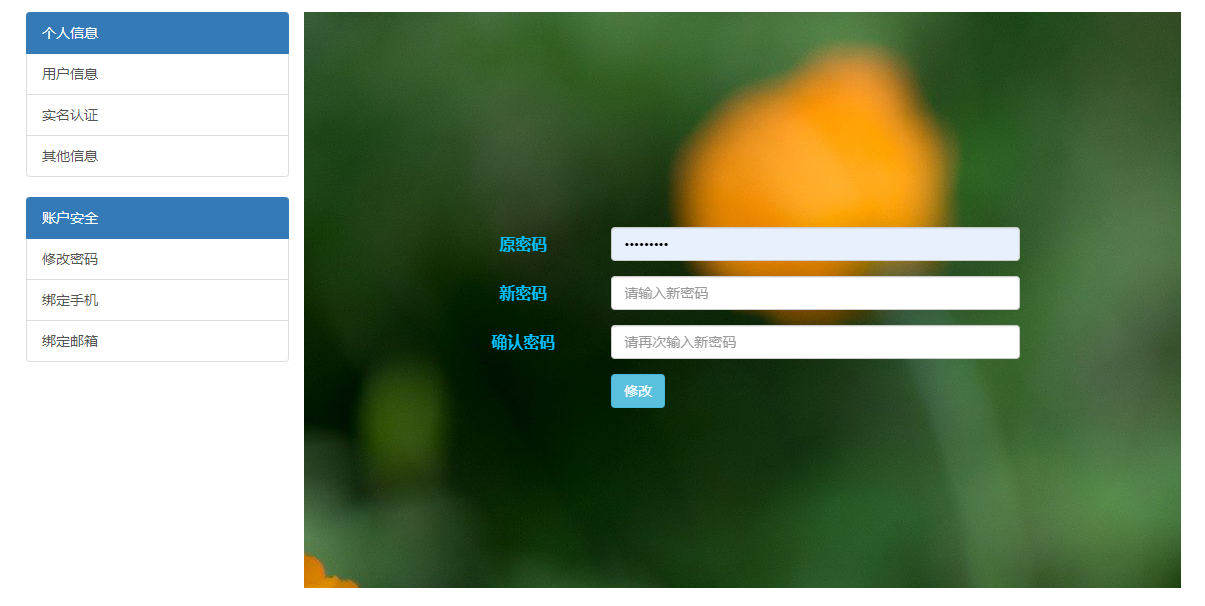


图20 修改密码

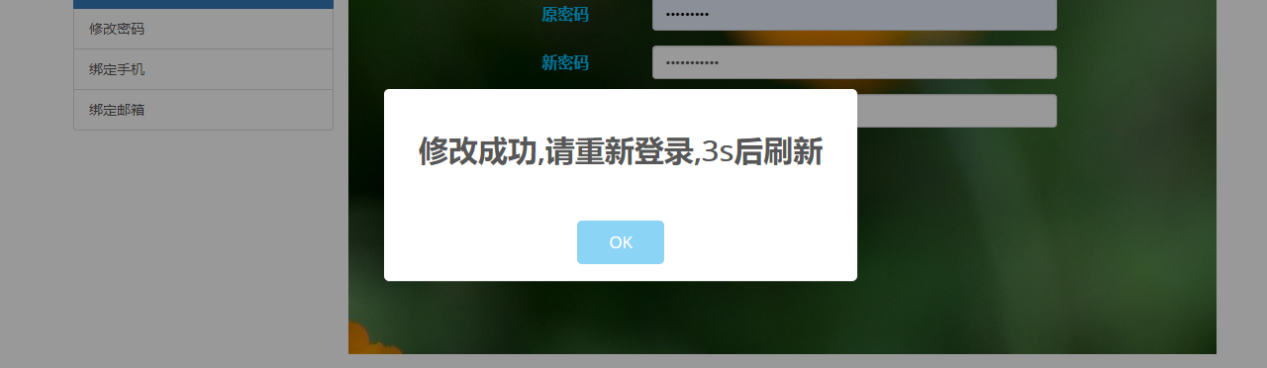


图21 修改密码成功

## 3.4 订单页面即购物车页面

### 3.4.1 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：

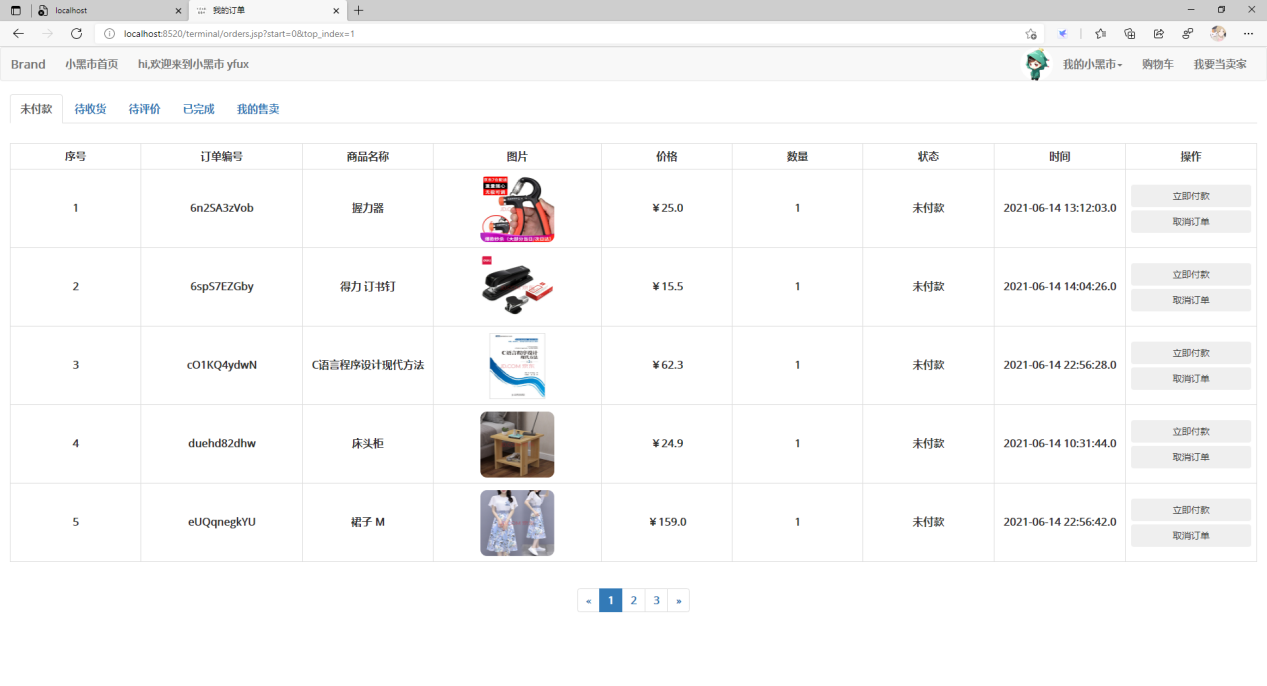


图22 商品订单页面分页

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。

### 3.4.2 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。

# 四、关键部分源代码

## 4.1、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。

## 4.2 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。

### 4.2.1 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");  
 }  
}

### 4.2.2 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。

## 4.3 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。

### 4.3.1 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565565*15310102651*16514，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。

### 4.3.2 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。

## 4.4、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。

# 五、使用的拓展技术

## 5.1 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。

## 5.2 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);  
}

## 5.3 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。

## 5.4 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。

## 5.5 SmsSingleSender及腾讯云短信服务

另外，本项目为了实现注册发送验证码，与腾讯云注册了发送短信的服务，该服务免费赠送一百条短信，用于学习这一百条短信已经足够了。

下面是腾讯云短信服务的控制台：

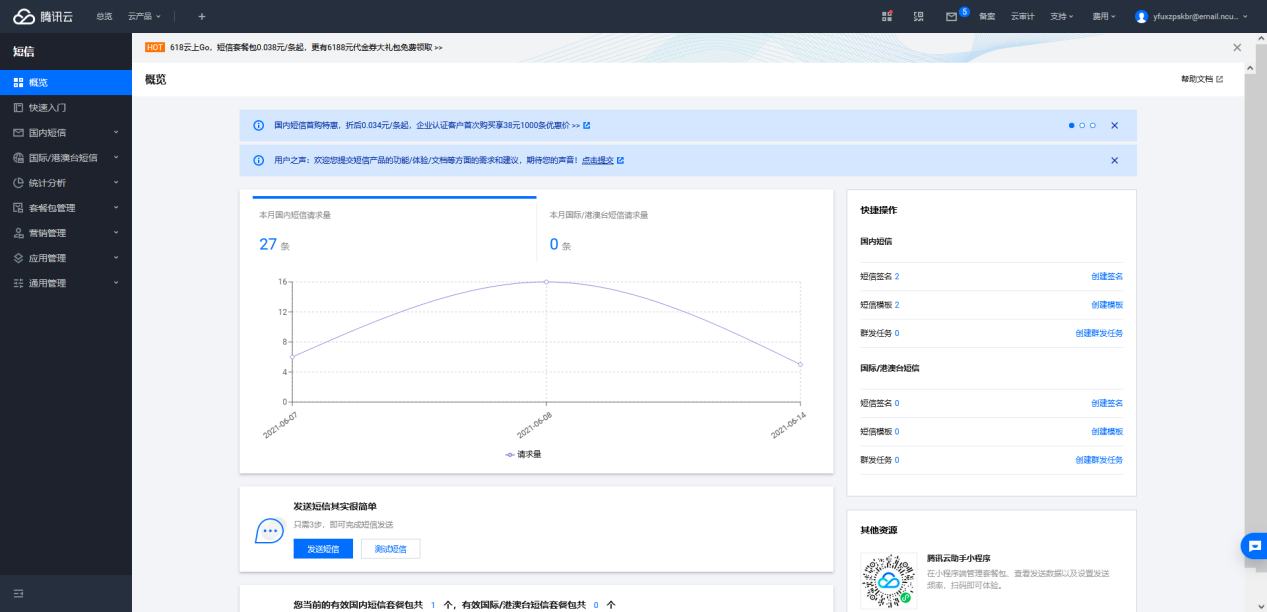


图23 腾讯云短信控制台

腾讯云提供了短信的发送接口，该接口使用的包的pom.xml配置如下：

<dependency>  
 <groupId>com.github.qcloudsms</groupId>  
 <artifactId>qcloudsms</artifactId>  
 <version>1.0.6</version>  
</dependency>

该功能使用前需要先腾讯云申请，通过审核之后便可以发送验证码。

主要的代码如下：

private boolean sendPhone(String mobile, long code) {  
 int appID = 1400517293;  
 String appKey = "02ae3cfde868c1b632fd9f3e21209406";  
// appKey = "";  
 int templateID = 947528;  
 String smsSign = "yfuxTest公众号";  
 try {  
 String[] params = {Long.toString(code), "2"};  
 System.out.println("验证码为 : " + code);  
 SmsSingleSender sender = new SmsSingleSender(appID, appKey);  
 SmsSingleSenderResult result = sender.sendWithParam("86", mobile,templateID, params, smsSign, "", "");  
 System.out.println(result);  
 if (result.errMsg.equals("OK"))  
 return true;  
 return false;  
 } catch (Exception e) {  
 e.printStackTrace();  
 return false;  
 }  
 }

使用之前需要先腾讯申请AppID和appKey，以及模板id和签名，最后通过*SmsSingleSenderResult result = sender.sendWithParam("86", mobile,templateID, params, smsSign, "", "");*发送验证码于手机，其中第一个参数是国际区号，中国是86；第二个参数是要发送的手机，这个将从前端传递过来；第三个参数是模板id，由腾讯云提供，用户进行申请，期间会进行审核；第四个参数是携带的参数，即验证码和有效时间；第四个参数是申请的签名，最后两个是附加参数，可以不使用。

## 5.6 部署到真实服务器

个人曾购买腾讯云学生服务器以及域名，并进行了域名和服务器的绑定，在此基础上，将这次的项目部署到服务器中，由于学生云服务器处理器不怎么多，因此访问速度可能不是特别快，其访问的网址为：<http://www.yfuxdeer.com:8520/terminal/>。

具体部署之前的服务器购买、域名购买、域名认证审核以及服务器绑定域名进行域名解析的工作这里就不再赘述，可以参考网上的教程。

具体的部署步骤如下：

1. 在本地找到tomcat的webapps目录，因为IDEA部署的项目都会自动保存在该目录下：

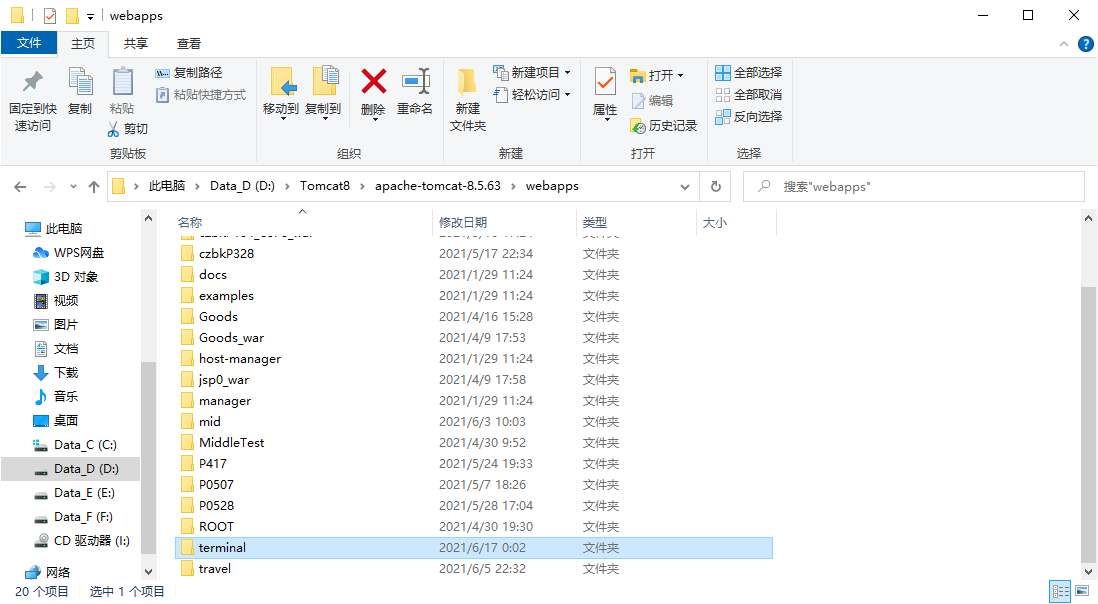


图24 本地terminal项目

1. 使用工具将这个文件夹传输到服务器对应的webapps目录下即可，CentOS7安装tomcat后webapps的绝对路径为/usr/shared/tomcat/webapps，将本机的terminal文件夹传到服务器的webapps下即可。这里使用的传输工具是FileZilla。具体的传输过程可以参考网上的诸多教程，这里不再赘述。
2. 完成这两步工作之后，重启服务器的tomcat服务器即可，此时就可以通过域名进行访问了。

# 六、总结

## 6.1 遇到的困难

本次开发是自己一个人独立书写一个功能相对完整的系统，期间也遇到了许多的困难。下面主要介绍记忆犹新的困难。

* 前端的布局

刚开始着手时，被前端布局挡住了去路，一时没有想法，想着模仿其他网络商城的前端界面，但是觉得模仿的过于逼真就会难以完成后端的实现，因此放弃了这个想法。知道在网上看到了一个前端框架Bootstrap，于是决定采用这个框架进行前端开发，因为这个框架看上去清新简洁，所以布局的页面就比较呈简约风格，于是就开启了我的前端书写。

* 样式的缓存

由于上次体会了外部样式表会因为缓存原因，有时候可能不起作用，因此本次使用的是内部样式，每次刷新都会重新加载，就避免了因为缓冲造成的书写的样式不同步的问题。通常解决这个问题的主要方法有：

* + 在引用处加上一个参数*?v=new Date()*，该参数的含义就是每次让参数v等于当前的时间，因为每次刷新的时间不可能相同，当浏览器发现新的页面参数和原来不一样时就认为页面存在了更新，就不会使用缓存的内容。
  + 第二个方法就是删掉程序的target目录，让其下次部署时重新生成target目录。
* 错误使用路径

在开发时，在jsp页面没有使用完成的虚拟目录，导致*location.href*的地址找不到，这个问题的出现原因就是没有弄明白什么时候使用绝对路径，什么时候使用相对路径。

在解决这个问题后，总结如下：

* 在客户端发起的传向服务器端的请求都是用绝对路径，其可以通过EL表达式获取${pageContext.request.contextPath}获取虚拟目录，在使用/依次获取服务器的资源路径。比如常见的重定位就需要使用完成的路径。
* 在服务器端发起的请求可以使用相对路径，比如转发时就是用相对路径，当然也可是使用绝对路径。

## 6.2 个人总结

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。

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，因为自己的项目比较简单，因此ER图便省略了。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。还学会了使用服务器进行短信验证码的发送，这是一个十分重要的技能，因为不管任何系统，都会涉及到注册登录的短信验证码的发送。

最后，感谢我们的Java Web孙老师的呕心沥血的教育和培养，在今后，一定继续学习这方面相关技术，了解更多的开发框架和模板。

下面我的安排就是学习Spring这套框架，希望能够从中学到许多东西。

# 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]腾讯云.服务器IP如何绑定域名(基于腾讯云)?[EB/OL].https://cloud.tencent.com/developer/article/1511605,.

[9]CSDN.Filezilla使用教程[EB/OL].https://blog.csdn.net/wy\_091112/article/details/82145456,.