



华南理工大学
South China University of Technology

JavaWeb

网络应用开发

学 院:	计算机科学与工程学院
专业班级:	18 网络工程
姓 名:	欧阳学强
学 号:	201830590222
指导老师	布社辉
时 间:	2021. 6. 1

一、网上商城系统需求分析报告	3
1、 引言	3
1.1 系统概述	3
2、 需求	3
2.1 需求概述	3
2.2 需求详细说明	4
2.3 开发框架	5
3、 时间安排	5
二、网上商城系统设计报告	6
1、 数据库设计	6
1.1 管理员 admin	6
1.2 用户 user	6
1.3 商品 good	6
1.4 登录日志 register_log	6
1.5 浏览记录 browse_log	7
1.6 购物车 cart	7
1.7 订单（购买记录）order	7
1.8 操作日志 log	8
2、 后端设计	8
2.1 设计思想	8
2.2 用例图	8
2.3 类图	9
3、 前端设计	10
3.1 美观性:	10
3.2 稳定性:	10
三、网上商城系统实现报告	11
1、 构建数据库	11
1.1 商户	11
1.2 用户	11
1.3 商品	11
2、 配置 xml 文件	12
2.1 导入对应 jar 包并配置 applicationContext.xml	12
2.2 导入对应 jar 包并配置 springmvc-config.xml 文件	13
2.3 导入对应 jar 包并配置 mybatis-config.xml 文件	13
2.4 连接 MySql 数据库，导入对应 jar 包并配置 jdbc.properties	13
3、 创建 Entity、Mapper、Dao 和 Service 层	13
4、 创建 Controller 层	14
4.1 创建 adminController	14
4.2 创建 userController	18
5、 项目部署（包含部署地址以及源代码地址）	24
四、网上商城系统测试报告	25
1、 用户测试	25
2、 商户测试	28

一、网上商城系统需求分析报告

1、引言

1.1 系统概述

21 世纪的今天，科技发展日新月异，网上购物也随之成为一种潮流。为了跟上时代发展的步伐，我们需要开发一套完善的系统，为商品卖家与买家提供更加快捷，安全的服务。由此我们决定开发一套网上商城系统。

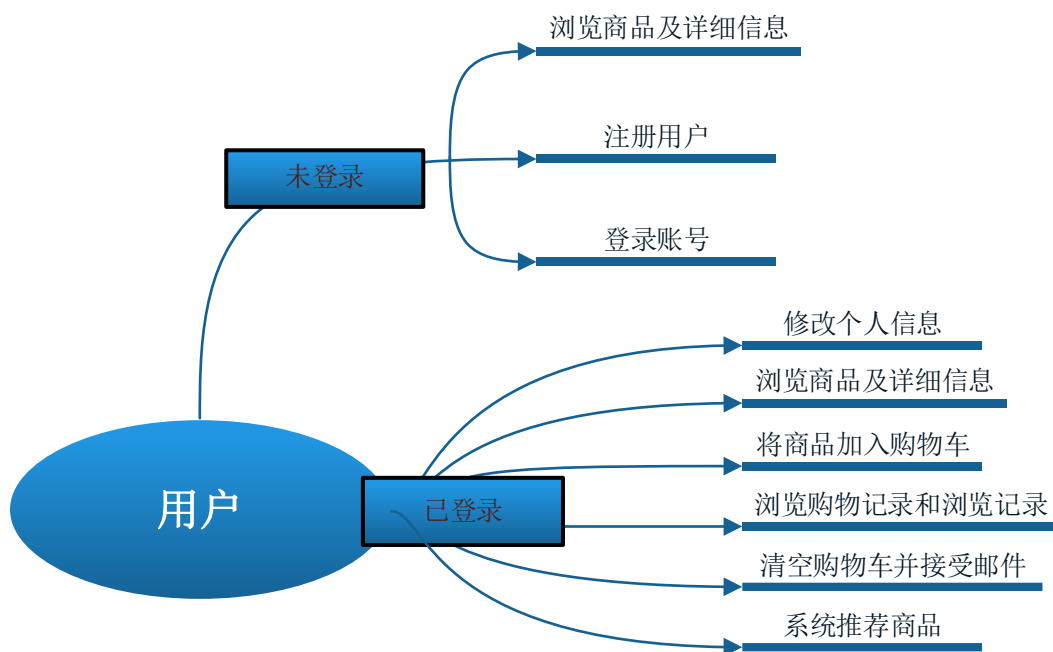
本文档面向开发人员，便于开发人员了解本系统所需要的功能。

2、需求

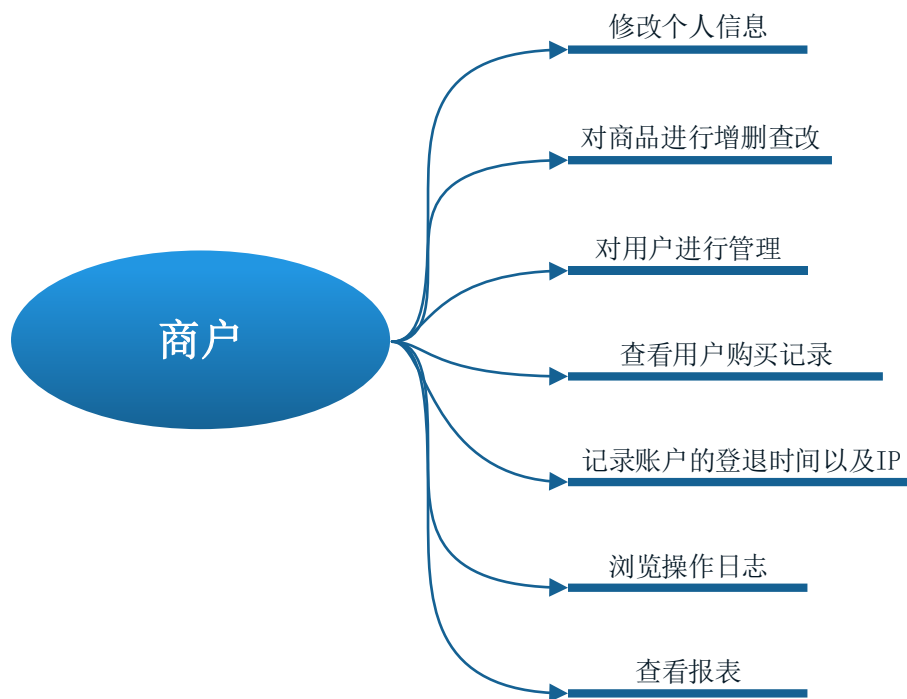
2.1 需求概述

2.1.1 目标

软件的开发意图是便于商品买卖双方在线上环境下进行交易，目标用户是商品买卖双方。主要功能如下
用户基本功能：



商户（同时也是管理员）基本功能：



2.1.1.2 运行环境

JDK 版本: 12

服务器系统: Windows 10

数据库系统: MySQL

Web 应用服务器: Tomcat 9

集成开发环境: Eclipse 2019

浏览器: Edge 86

2.1.1.3 关键点

- 为确定用户付款后，商户准时发货，需要在用户付款后给用户发送一条确认信息。
- 能够获取到登录用户的真实 IP 地址。
- 为了方便用户体验，需要设置有购物车功能
- 需要支持图片上传并浏览的功能
- 支持关键字查询

2.2 需求详细说明

2.2.1 功能描述

用户界面:

- 未登录用户打开网站首页时，首页头部是网站的导航栏，分别对应各类商品以及注册和登录选项，而首页主题部分将展示一张海报；
- 未登录用户可以点击导航栏中的不同种类商品，从而跳转到对应页面进行浏览，并且可

以点开商品进行详细信息的浏览，但是不能执行加入购物车功能；

- 未登录用户可以点击注册选项，填写信息后进行注册；
- 未登录用户可以点击登录选项，凭借注册账号进行登录；
- 登录用户的导航栏，将会多出浏览记录，购买记录，购物车以及修改信息的功能；
- 登录用户可以按照不同类别浏览商品，并且加入购物车；
- 登录用户可以清空购物车并付款，付款后会接受到商户发送的邮件。

商户（同时也是管理员）界面：

- 登录功能；
- 对商品关键词的查询；
- 对商品的增加，删除，修改和删除功能；
- 对用户的删除功能；
- 查看用户的购物记录；
- 查看对商品的操作日志；
- 查看用户和商户的登录退出信息；
- 商品的销售报表。

2.3 开发框架

使用 MVC 三层架构，并在 SSM 框架下进行搭建。

3、时间安排

5 月 3 号	完成需求分析报告
5 月 4 号到 5 月 6 号	完成系统的整体模块设计，并完成系统设计报告
5 月 7 号	完成数据库的代码编写，并编写系统实现报告
5 月 8 号到 10 号	完成初步的功能，并补充系统实现报告
5 月 11 号到 12 号	第一次对系统进行 debug 测试并修改，并编写测试报告
5 月 13 号	完成在服务器上的部署，再次补充系统实现报告
5 月 14 号	第二次对系统进行 debug 测试并修改，并补充测试报告

二、网上商城系统设计报告

1、数据库设计

1.1 管理员 admin

(同时也是商户)

字段名	类型	说明
id	int	自增项 id
username	string	用户名
password	string	密码

1.2 用户 user

字段名	类型	说明
id	int	自增项 id
username	string	用户名
password	string	密码
name	string	姓名
phone	string	电话
address	string	地址

1.3 商品 good

字段名	类型	说明
id	int	自增项 id
name	string	商品名
cover	string	封面图片路径
image1	string	图片 1 路径
image2	string	图片 2 路径
price	float	价格
intro	string	简介, 不超过 100 字
stock	int	库存
type_id	int	商品种类, 取值 1, 2, 3, 4

1.4 登录日志 register_log

字段名	类型	说明
id	int	自增项 id
user_id	int	用户 id

user_name	string	用户名
person_type	int	用户类型, 1 商户, 2 用户
ip	string	ip 地址
operate_type	int	操作类型, 1 登录, 2 退出
time	date	操作时间

1.5 浏览记录 browse_log

字段名	类型	说明
id	int	自增项 id
user_id	int	用户 id
username	string	用户名
good_id	int	商品 id
good_name	string	商品名
good_type	int	商品种类
time	date	浏览时间

1.6 购物车 cart

字段名	类型	说明
id	int	自增项 id
user_id	int	用户 id
good_id	int	商品 id
amount	int	数量
price	float	单价

1.7 订单（购买记录）order

字段名	类型	说明
id	int	自增项 id
user_id	int	用户 id
username	string	用户名
good_id	int	商品 id
good_name	string	商品名
price	float	商品单价
amount	int	商品数量
pay_type	int	支付方式, 1 货到付款, 2 微信, 3 支付宝
phone	string	电话
address	string	地址
time	date	购买时间

1.8 操作日志 log

字段名	类型	说明
id	int	自增项 id
user_id	int	用户 id
username	string	用户名
ip	string	ip 地址
operate_object	int	操作对象, 1 用户, 2 商品
operated_id	int	被操作对象 id
operated_name	string	被操作对象名字
content	string	操作内容, 增删查改
time	date	操作时间

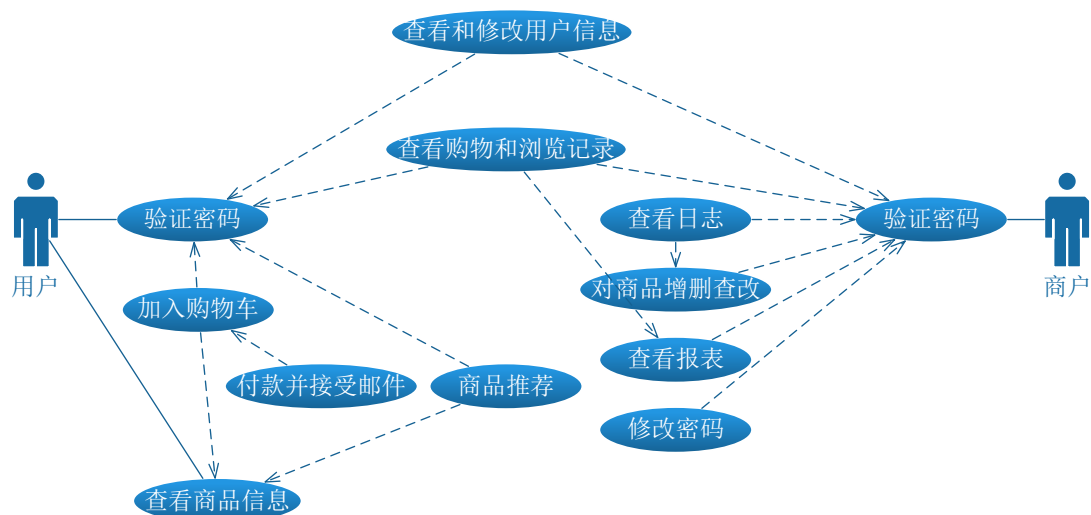
2、后端设计

2.1 设计思想

在 MVC 三层框架下, 使用 SSM 框架

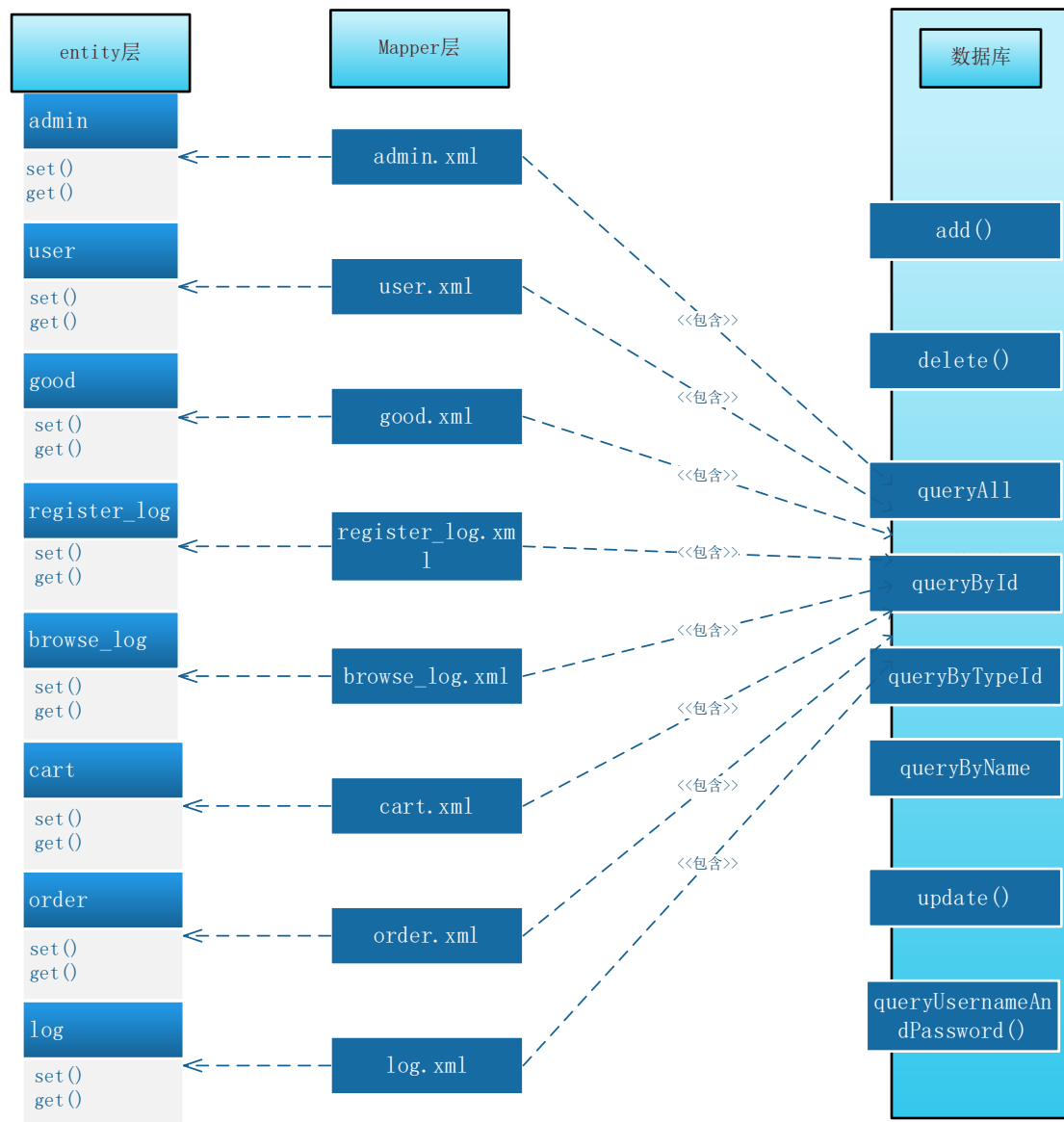
- 创建 spring 项目, 导入 jar 包, 配置 xml 文件, 配置数据库连接文件
- 构建 entity 实体类, 并实现 set 和 get 方法
- 构建 mapper.xml, 用于连接数据库进行增删查改
- 构建 dao 层, 调用 mapper 对数据的增删查改
- 构建 service 层, 传入数据给 dao 层, 用于增删查改
- 构建 controller 层, 用于连接前端和后端, 进行数据交换

2.2 用例图

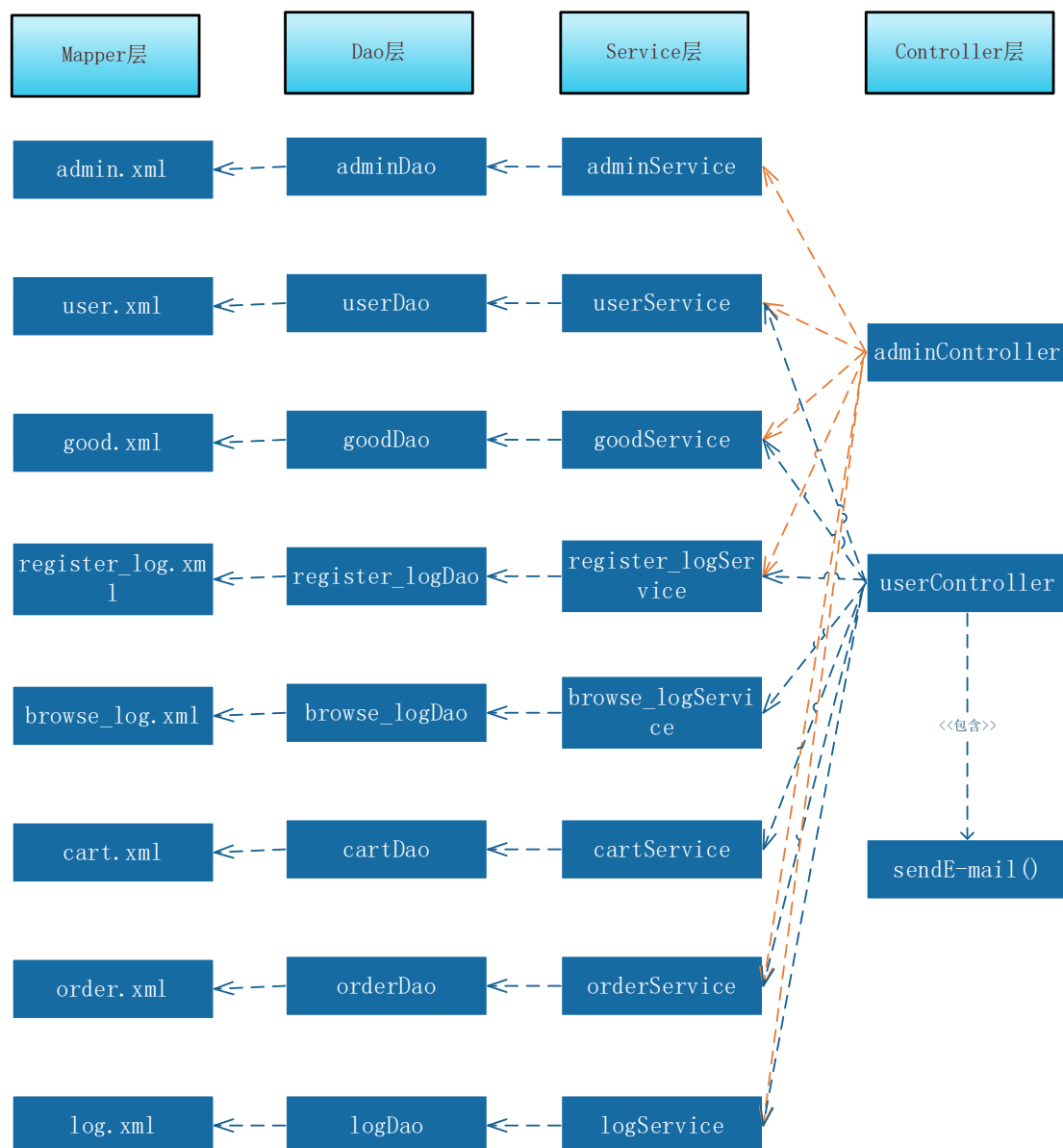


2.3 类图

2.3.1 mapper 层与数据库之间的连接



2.3.2 mapper、dao、service、controller 层之间的关系



3、前端设计

3.1 美观性:

套用网上前端模板

3.2 稳定性:

- 能够控制用户输入的格式
- 在用户和登录时，账号密码必填；
- 在商户增加和修改商品价格时设定为浮点型；
- 在商户增加和修改商品数量时设定为整数型；
- 在用户未登录时，不允许显示购物车，用户等条目，并且不允许执行加入购物车操作，

并给出提示。

三、网上商城系统实现报告

1、 构建数据库

由于创建数据库的过程是一个重复的过程，所以下面只列举三种表的创建，其余表依据设计报告同理可以得出。

1.1 商户

(同时也是管理员)，同时插入一对账号密码

```
create table admin(  
id int AUTO_INCREMENT not null primary key,  
username varchar(20),  
password varchar(20));  
insert into admin(username,password) values('zs','123456');
```

1.2 用户

```
create table user(  
id int AUTO_INCREMENT not null primary key,  
username varchar(20),  
password varchar(20),  
name varchar(20),  
phone varchar(20),  
address varchar(20));
```

1.3 商品

```
create table goods(  
id int AUTO_INCREMENT not null primary key,  
name varchar(20),  
cover varchar(20),  
image1 varchar(20),  
image2 varchar(20),  
price int,  
intro varchar(20),  
stock int,  
type_id int);
```

2、 配置 xml 文件

首先新建一个 spring 项目

2.1 导入对应 jar 包并配置 applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans >
    <!-- 配置注解扫描范围 -->
    <context:component-scan base-package="com.service" /
    <!-- 引用数据库配置文件 -->
    <context:property-placeholder location="classpath:jdbc.properties" />
    <!-- 配置数据源 - 使用tomcat连接池 -->
    <bean id="dataSource" class="org.apache.tomcat.jdbc.pool.DataSource"
destroy-method="close">
        <property name="driverClassName" value="${jdbc.driver}" /><!-- 数据库驱动 -->
        <property name="url" value="${jdbc.url}" /> <!-- 连接URL -->
        <property name="username" value="${jdbc.username}" /> <!-- 数据库用户名 -->
        <property name="password" value="${jdbc.password}" /> <!-- 数据库密码 -->
        <!-- 连接池参数配置 -->
        <property name="initialSize" value="1" /><!-- 连接器启动时创建的初始连接数。默认为10 -->
        <property name="maxActive" value="5" /><!-- 同时能分配的活跃连接的最大数目。默认为100 -->
        <property name="minIdle" value="1" /><!-- 空闲连接的最小数目。默认为initialSize: 10 -->
        <property name="maxIdle" value="5" /><!-- 空闲连接的最大数目。默认为maxActive: 100 -->
        <property name="testOnBorrow" value="true" /><!-- 从池中借出对象之前，是否对其进行验证。默认值为false。如果对象验证失败，将其从池中清除，再接着去借下一个。注意：为了让 true 值生效，validationQuery参数必须为非空字符串。为了实现更高效的验证，可以采用validationInterval -->
        <property name="validationQuery" value="select 1" /><!-- 在将池中连接返回给调用者之前，用于验证这些连接的SQL 查询。如果指定该值，则该查询不必返回任何数据，只是不抛出SQLException 异常。默认为 null -->
        <property name="validationInterval" value="30000" /><!-- 为避免过度验证而设定的频率时间值（以秒计）。最多以这种频率运行验证。如果连接应该进行验证，但却没能在此间隔时间内得到验证，则会重新对其进行验证。默认为30000（30秒） -->
    </bean>
    <!-- 配置MyBatis SqlSessionFactory -->
    <bean id="sqlSessionFactory"
```

```

class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="mapperLocations" value="classpath:com/mapper/*.xml"
/><!-- 兼容xml方式 -->
    <property name="configLocation" value="classpath:mybatis-
config.xml"/><!-- 配置mybatis全局配置文件 -->
</bean>
<!-- 配置Dao所在包 Mybatis会动态创建实现类 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory" />
    <property name="basePackage" value="com.dao" /><!-- 兼容注解方式 -->
</bean>
<!-- 配置事务管理器 -->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>
<!-- 使用注解定义事务 -->
<tx:annotation-driven transaction-manager="transactionManager"/>
</beans>

```

2.2 导入对应 jar 包并配置 springmvc-config.xml 文件

原理与 applicationcontext.xml 相似，请参见源代码

2.3 导入对应 jar 包并配置 mybatis-config.xml 文件

原理与 applicationcontext.xml 相似，请参见源代码

2.4 连接 MySQL 数据库，导入对应 jar 包并配置 jdbc.properties

```

jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/new_schema?useUnicode=true&characterEn
coding=UTF-8&serverTimezone=GMT%2B8
jdbc.username=root
jdbc.password=*****

```

3、 创建 Entity、Mapper、Dao 和 Service 层

mybatis 的逆向工程，可以通过相应插件，自动生成 mybatis 数据库连接的一些文件。基本步骤如下：

- 下载相应工具，即 mybatis-generator-core
- 编写 generatorConfig.xml 文件
- 命令行模式，进入 lib 文件夹
- 输入

```
java -jar mybatis-generator-core.jar -
configfile generatorConfig.xml -overwrite
```

- 生成 entity、mapper 和 dao 层

而 service 层就是整合 dao 层中的功能，基本无太多改动。

4、 创建 Controller 层

4.1 创建 adminController

首先 import 相应的包，并通过注解@Autowired 的方式调用 service 层。然后依据前端的不同请求，从数据库获取信息，并返回给前端。

4.1.1 登录

判断账号密码是否正确，并将登录记录加入到登录日志。

获取 IP 地址时应该防止用户使用代理 IP。

```
@RequestMapping("/login")
public String login(Admins admin, HttpServletRequest request,
HttpSession session) {
    if (adminService.checkUser(admin.getUsername(),
admin.getPassword())) { // 登录成功
        session.setAttribute("username", admin.getUsername());
        String ip = request.getHeader("x-forwarded-for");
        if (ip != null && ip.length() != 0
&& !"unknown".equalsIgnoreCase(ip)) {
            // 多次反向代理后会有多个ip值，第一个ip才是真实ip
            if( ip.indexOf(",")!=-1 ){
                ip = ip.split(",")[0]; }
        }
        if (ip == null || ip.length() == 0 ||
"unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("Proxy-Client-IP"); }
        if (ip == null || ip.length() == 0 ||
"unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("WL-Proxy-Client-IP"); }
        if (ip == null || ip.length() == 0 ||
"unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("HTTP_CLIENT_IP"); }
        if (ip == null || ip.length() == 0 ||
"unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("HTTP_X_FORWARDED_FOR"); }
        if (ip == null || ip.length() == 0 ||
"unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("X-Real-IP"); }
        if (ip == null || ip.length() == 0 ||
```

```

"unknown".equalsIgnoreCase(ip)) {
    ip = request.getRemoteAddr(); }
    Registers register = new Registers();
    register.setUserId(1);
    register.setUserName(admin.getUsername());
    register.setPersonType("管理员");
    register.setIpAddress(ip);
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");//设置日期格式
    register.setSystemtime(df.format(new Date()));
    register.setOperateType("登录");
    registerService.insertRegister(register);
    return "redirect:index";
}
request.setAttribute("msg", "账号或密码错误!");//登录失败
return "/admin/login.jsp";}

```

4.1.1.2 修改密码

```

@RequestMapping("/adminReset")
public String adminReset(@RequestParam("passwordNew") String
passwordNew, @RequestParam("id") int id,
    @RequestParam("username") String username,
    @RequestParam("password") String password,
    HttpServletRequest request) {
    if (adminService.checkUser(username, password)) {
        Admins admin = new Admins();
        admin.setPassword(passwordNew);
        admin.setId(id);
        admin.setUsername(username);
        adminService.update(admin);
        request.setAttribute("msg", "修改密码成功!请重新登录");
        return "/admin/login.jsp";
    } else {
        request.setAttribute("msg", "原密码错误!请重新输入");
        return "/admin/admin_reset.jsp";
    }
}

```

4.1.1.3 展示订单

```

@RequestMapping("/showOrder")
public String showOrder(HttpServletRequest request) {
    List<Orders> orderlist = orderService.getList();
    request.setAttribute("orderList", orderlist);
    return "/admin/order_list.jsp";
}

```

```
}
```

4.1.4 增加商品

并将操作记录加入到操作日志

```
@RequestMapping("/newGood")
public String newGood(Goods good, HttpServletRequest request,
    @RequestParam("file") MultipartFile file,@RequestParam("file1")
MultipartFile file1,@RequestParam("file2") MultipartFile file2) throws
IOException {
    InputStream input = file.getInputStream();//io
    String fileName = file.getOriginalFilename();
    OutputStream out = new FileOutputStream("路径"+fileName);
    InputStream input1 = file1.getInputStream();//io
    String fileName1 = file1.getOriginalFilename();
    OutputStream out1 = new FileOutputStream("路径"+fileName1);
    InputStream input2 = file2.getInputStream();//io
    String fileName2 = file2.getOriginalFilename();
    OutputStream out2 = new FileOutputStream("路径"+fileName2);
    byte[] bs =new byte[1024];
    int len = -1;
    while((len = input.read(bs)) != -1) {
        out.write(bs, 0, len); }
    out.close();
    input.close();
    good.setCover(fileName);
    byte[] bs1 =new byte[1024];
    int len1 = -1;
    while((len1 = input1.read(bs1)) != -1) {
        out1.write(bs1, 0, len1); }
    out1.close();
    input1.close();
    good.setImage1(fileName1);
    byte[] bs2 =new byte[1024];
    int len2 = -1;
    while((len2 = input2.read(bs2)) != -1) {
        out2.write(bs2, 0, len2); }
    out2.close();
    input2.close();
    good.setImage2(fileName2);
    goodService.insertGood(good);
    Logs log = new Logs();
    log.setUserId(1);
    log.setUserName("zs");
    log.setIpAddress(ip);
}
```



```

        log.setOperateObject("商品");
        log.setOperateId(0);
        log.setOperateName(good.getName());
        log.setContent("增加");
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        log.setSysTime(df.format(new Date()));
        logService.insertLog(log);
        request.setAttribute("msg", "新加商品成功");
        return "/admin/new_good.jsp";
    }

```

4.1.5 删除商品

并且将操作记录加入到操作日志

```

@RequestMapping("/deleteGood")
public String deleteGood(@RequestParam("id") int id, HttpServletRequest
request) {
    Logs log = new Logs();
    log.setUserId(1);
    log.setUserName("zs");
    log.setIpAddress(ip);
    log.setOperateObject("商品");
    log.setOperateId(id);
    Goods good = new Goods();
    good = goodService.queryGoodById(id);
    log.setOperateName(good.getName());
    log.setContent("删除");
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    log.setSysTime(df.format(new Date()));
    logService.insertLog(log);
    goodService.deleteGood(id);
    return "showGood";
}

```

4.1.6 通过名字查询商品

```

@RequestMapping("/queryGood")
public String queryGood(@RequestParam("name") String name,
HttpServletRequest request) {
    List<Goods> goodlist = goodService.getListByName(name);
    request.setAttribute("goodList", goodlist);
    request.setAttribute("msg", "查询商品结果如下");
    return "/admin/good_list.jsp";
}

```

4.1.7 通过编号查询商品

与通过名字查询类似，参见源代码。

4.1.8 展示商品

与展示订单类似，参见源代码。

4.1.9 修改商品

与增加商品类似，参见源代码

4.1.10 展示用户

与展示订单类似，参见源代码。

4.1.11 删除用户

并将操作记录加入到操作日志

与删除商品类似，参见源代码

4.1.11 通过名字查询用户

与通过名字查询商品类似，参见源代码。

4.1.12 展示操作日志

与展示订单类似，参见源代码。

4.1.13 展示登录记录

与展示订单类似，参见源代码。

4.2 创建 userController

首先 `import` 相应的包，并通过注解 `@Autowired` 的方式调用 `service` 层。然后依据前端的不同请求，从数据库获取信息，并返回给前端。

4.2.1 用户注册

```
public String register( Users user, HttpServletRequest request){
    int flag = userService.insertUser(user);
    if(flag==1) {
        request.setAttribute("msg", "注册成功");
        return "/index/login.jsp";
    }else {
        request.setAttribute("msg", "用户名重复! 注册失败");
        return "/index/register.jsp";}}}
```

4.2.2 用户登录

类似操作已经在 `admincontroller` 中展示

4.2.3 修改用户信息

类似操作已经在 `admincontroller` 中展示

4.2.4 展示商品信息

类似操作已经在 `admincontroller` 中展示

4.2.5 依据类型展示商品

类似操作已经在 `admincontroller` 中展示

4.2.6 展示商品详细信息

```
@RequestMapping("/detail")
public String detail(@RequestParam("goodid") int goodid,
    HttpSession session,HttpServletRequest request){
    Goods goods = goodService.queryGoodById(goodid);
    request.setAttribute("good", goods);
    Users user = new Users();
    user = (Users) session.getAttribute("user");
    if (user!=null) {
        Histories history = new Histories();
        history.setUserId(user.getId());
        history.setUserName(user.getUsername());
        history.setGoodId(goods.getId());
        history.setGoodName(goods.getName());
        history.setGoodType(goods.getTypeId());
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");//设置日期格式
        history.setSystime(df.format(new Date()));
        historyService.insertHistory(history);
    }
    return "/index/detail.jsp";
}
```

4.2.7 购物车功能

即查看购物车内商品

```
@RequestMapping("/cart")
public String cart(@RequestParam("userid") int userid, HttpSession
session, HttpServletRequest request) {
    List<Items> items = itemService.getItemListByOneId(userid);
    int allPrice = 0;
    if(items==null) {
        request.setAttribute("msg", "空空如也");
    }else {
```

```

        int allamount = 0;
        for(Items item:items) {
            int goodid = item.getGoodId();
            Goods goods = goodService.queryGoodById(goodid);
            item.setGood(goods);
            int price = item.getPrice();
            int amount = item.getAmount();
            allPrice += price * amount;
            allamount += amount;
        }
        session.setAttribute("count", allamount); //购物车数量
        session.setAttribute("items", items);
        session.setAttribute("allprice", allPrice); //购物车总价
    }
    return "/index/cart.jsp";
}

```

4.2.8 加入购物车

如果此时购物车中有该商品，则该商品数量+1，并且跳转到购物车页面，如果没有，则新加购物车。

```

@RequestMapping("/buy")
public String buy(@RequestParam("goodid") int goodid,
    @RequestParam("userid") int userid,
    HttpSession session, HttpServletRequest request){
    //将购物车记录加入到数据库
    Goods goods = goodService.queryGoodById(goodid);
    Users users = userService.queryUserById(userid);
    if (goods .getStock() <= 0) {
        return "empty";
    }
    Items item = new Items();
    item = itemService.getItemByTwoId(users.getId(), goods.getId());
    if(item==null) {
        Items newitem = new Items();
        newitem.setAmount(1);
        newitem.setOrderId(users.getId());
        newitem.setGoodId(goods.getId());
        newitem.setPrice(goods.getPrice());
        itemService.insertItem(newitem);
        //输出购物车内商品数量
        List<Items> items = itemService.getItemListByOneId(userid);
        int amount = 0;
        for(Items i:items) {
            amount += i.getAmount();}
    }
}

```

```

        session.setAttribute("count", amount);
        return "getGood";
    }else {
        int n = item.getAmount();
        n = n+1;
        item.setAmount(n);
        itemService.updateItem(item);
        //输出购物车内商品数量
        List<Items> items = itemService.getItemListByOneId(userid);
        int amount = 0;
        for(Items i:items) {
            amount += i.getAmount();
        }
        session.setAttribute("count", amount);
        return "cart";
    }
}

```

4.2.9 减少购物车

商品数量-1, 如果购物车商品数量此时为 0, 则删除该商品。

```

@RequestMapping("/lessen")
public String lessen(@RequestParam("goodid") int goodid,
    @RequestParam("userid") int userid,
    @RequestParam("id") int id){
    Items item = new Items();
    item = itemService.getItemByTwoId(userid, goodid);
    int n = item.getAmount();
    n = n-1;
    if(n==0) {
        itemService.deleteItem(id);
    }else {
        item.setAmount(n);
        itemService.updateItem(item);
    }
    return "cart";
}

```

4.2.10 删除购物车

```

@RequestMapping("/delete")
public String delete(@RequestParam("goodid") int goodid,
    @RequestParam("userid") int userid,
    @RequestParam("id") int id){
    Items item = new Items();
    item = itemService.getItemByTwoId(userid, goodid);
    itemService.deleteItem(id);
    return "cart";
}

```

4.2.11 提交订单付款

```
@RequestMapping("/save")
public String save(@RequestParam("userid") int userid,
    @RequestParam("pay") int pay,
    HttpSession session, HttpServletRequest request){
    List<Items> items = itemService.getItemListByOneId(userid);
    for(Items i:items) {
        Orders order = new Orders();
        Users user = userService.queryUserById(i.getOrderid());
        Goods good = goodService.queryGoodById(i.getGoodid());
        order.setAddress(user.getAddress());
        order.setAmount(i.getAmount());
        order.setGoodId(good.getId());
        order.setGoodName(good.getName());
        if(pay==1) {
            order.setPaytype("支付宝");
        }
        if(pay==2) {
            order.setPaytype("微信");
        }
        if(pay==3) {
            order.setPaytype("货到付款");
        }
        order.setPhone(user.getPhone());
        order.setPrice(good.getPrice());
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");//设置日期格式
        order.setSystime(df.format(new Date()));
        order.setUserId(user.getId());
        order.setUserName(user.getName());
        orderService.insert(order);
    }
    request.setAttribute("msg", "付款成功");
    return "/index/payok.jsp";
}
```

4.2.12 发送邮箱

由于阿里云不开发 25 端口，而 25 端口用于发送邮件，所以在服务器上无法进行发送邮箱功能，不过在测试时，我会采用自己电脑进行演示。

```
@RequestMapping("/mail")
public String mail(@RequestParam("mail") String mail,
    HttpServletRequest request){
    try {
```

```

//设置发件人
String from = "*****@qq.com";
//设置收件人
String to = mail;
//设置邮件发送的服务器，这里为QQ邮件服务器
String host = "smtp.qq.com";
//获取系统属性
Properties properties = System.getProperties();
//设置系统属性
properties.setProperty("mail.smtp.host", host);
properties.put("mail.smtp.auth", "true");
//获取发送邮件会话、获取第三方登录授权码
Session session = Session.getDefaultInstance(properties, new
Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication()
    {
        return new PasswordAuthentication(from, "*****");
    }
});
Message message = new MimeMessage(session);
//防止邮件被当然垃圾邮件处理，披上Outlook的马甲
message.addHeader("X-Mailer", "Microsoft Outlook Express
6.00.2900.2869");
message.setFrom(new InternetAddress(from));
message.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));
//邮件标题
message.setSubject("Are you ok购物商城");
BodyPart bodyPart = new MimeBodyPart();
bodyPart.setText("亲，您的货物已经发出，请注意查收哦。");
Multipart multipart = new MimeMultipart();
multipart.addBodyPart(bodyPart);
message.setContent(multipart);
Transport.send(message);
System.out.println("mail transports successfully");
} catch (Exception e) {
    e.printStackTrace();
}
return "/index/payok.jsp";
}

```

4.2.13 查看订单

类似操作已经在 admincontroller 中展示

4.2.14 查看浏览记录

类似操作已经在 `admincontroller` 中展示

4.2.15 推荐系统

根据用户的最近浏览信息，获取用户最近浏览的商品类别，从而推荐同种类别商品。

```
@RequestMapping("/recommend")
public String recommend(@RequestParam("userid") int userid,
    HttpSession session, HttpServletRequest request) {
    List<Histories> history = historyService.getListById(userid);
    int typeid;
    if(history==null) {
        typeid=1;
    }
    else {
        typeid = history.get(0).getGoodType();
    }
    List<Goods> goodlist = goodService.queryGoodByTypeId(typeid);
    request.setAttribute("goodList", goodlist);
    return "/index/goods.jsp";}
```

5、 项目部署

- 1- 注册阿里云账号
- 2- 购买轻量应用服务器(服务器系统为Windows 2016 Server,IP地址为116.62.105.78)
- 3- 为服务器设置用户密码。
- 4- 通过远程桌面连接登录服务器账号，进行远程操控。
- 5- 在阿里云中打开 8080 端口用于网页访问(由于阿里云不允许 25 端口的打开，所以服务器端的网站无法执行发送邮件的功能)
- 6- 在服务器上安装 JDK12 并配置环境
- 7- 在服务器上安装 Tomcat9 并配置环境
- 8- 将写好的项目打包成 war 包
- 9- 点击 `statup.sh` 启动服务器上的 Tomcat9，并将 war 包放在 Tomcat 下的 webapps 文件夹下
- 10- 部署位置 `116.62.105.78:8080/ssmdemo/index/index.jsp`
- 11- 将源代码上传到 GitHub，地址为：
`https://github.com/ouyangxueqiang/mallplus`

四、网上商城系统测试报告

首先，进入 116.62.105.78:8080/ssmdemo/index/index.jsp，即商城主页。

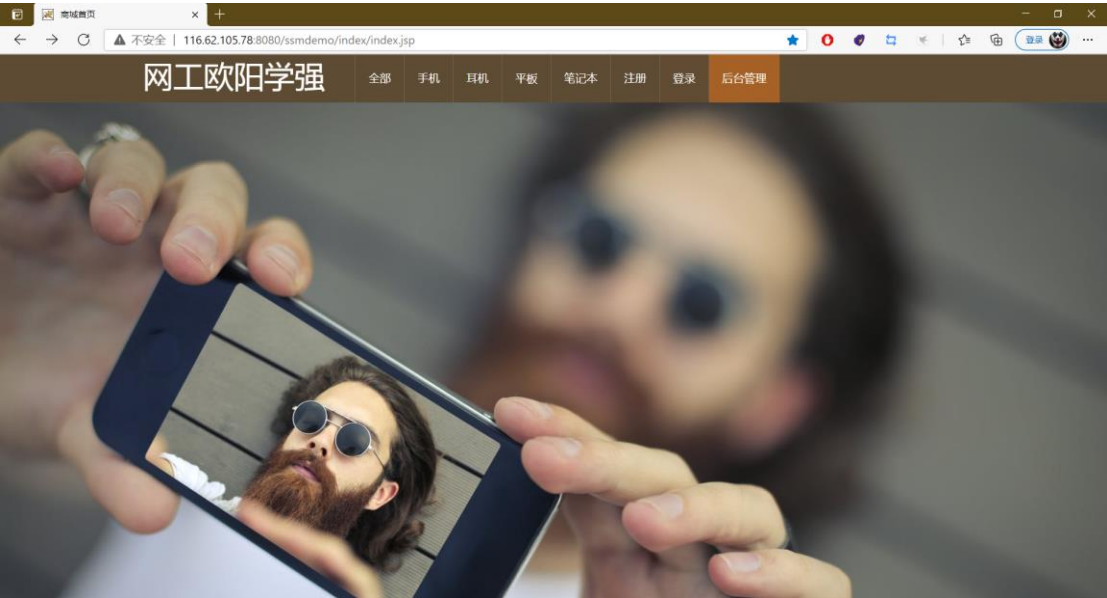


图 1、主页

用户账号：zs，用户密码：123456

商户账号：zs，商户密码：123456

1、用户测试

如果用户没有登录，依然可以浏览商品，但当加入购物车时，系统会提醒需要登录：

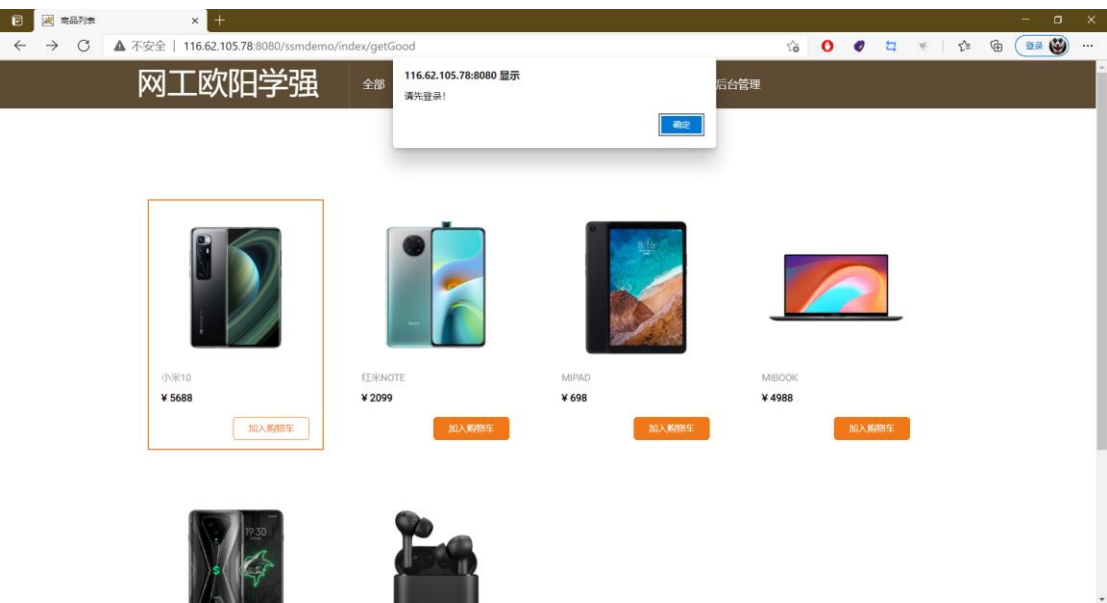


图 2、未登录加入购物车

用户登录界面：

用户登录

用户名 *

zs

密码 *

123456

登录

图 3、用户登录界面

登录成功后，可以查看商品，以及推荐商品、还有浏览记录以及订单记录：



图 4、登录用户的推荐页面

登录成功后，可以修改个人信息：

网工欧阳学强

全部 手机 耳机 平板 笔记本 推荐 我的订单 浏览记录 退出 后台管理

2 zs

修改信息

用户名 *

zs

密码 *

123456

收货人 *

欧阳学强

收货电话 *

15080950585

收货地址 *

广州

修改

图 5、用户修改个人信息界面

登录成功后，可以将商品加入购物车，之后确定商品数量，当数量为 0，自动删除：



图 6、购物车界面

加入购物车后，可以付款清空购物车：

确认收货信息

收货人:欧阳学强
收货电话:15080950585
收货地址:广州

若信息有误，请修改我的信息。

选择支付方式

支付金额: 1396



图 7、付款界面

付款成功后，可以输入邮箱，从而收到发货提醒，但由于阿里云服务器 25 端口关闭，导致服务器上网站无法发送邮件，所以使用我自己电脑演示该功能，得出下列结果：



图 8、发送邮件界面



图 9、接受邮件结果

2、商户测试

点击后台管理，登录商户账号。

点击订单管理，可以查看订单详情，并获得销售额：

后台管理

订单管理

商品管理

用户管理

操作日志

登录记录

修改密码

退出

全部订单

销售额为116736元

商品名称	商品单价	商品数量	付款方式	下单时间	地址	电话	姓名	用户编号
小米10	5688	1	货到付款	2021-05-16 20:31:36	广州	19927525330	欧阳学强	1
mibook	4988	1	货到付款	2021-05-16 20:31:36	广州	19927525330	欧阳学强	1
小米10	5688	2	支付宝	2021-05-16 20:41:40	广州	19927525330	欧阳学强	1
小米10	5688	1	支付宝	2021-05-16 21:01:24	广州	19927525330	欧阳学强	1
mibook	4988	1	微信	2021-05-16 21:03:43	广州	19927525330	欧阳学强	1
mibook	4988	1	支付宝	2021-05-16 21:09:46	广州	19927525330	欧阳学强	1
mibook	4988	1	货到付款	2021-05-16 21:30:51	广州	19927525330	欧阳学强	1
mibook	4988	2	货到付款	2021-05-22 22:50:14	广州	15080950585	欧阳学强	1

图 10、订单报表

点击商品管理，点击新加商品：

后台管理	订单管理	商品管理	用户管理	操作日志	登录记录	修改密码	退出
------	------	------	------	------	------	------	----

全部商品	新加商品	查询商品
------	------	------

商品名称	红米耳机
封面	<input type="button" value="选择文件"/> 红米耳机.jpg
图片1	<input type="button" value="选择文件"/> 红米耳机1.jpg
图片2	<input type="button" value="选择文件"/> 红米耳机2.jpg
价格	99
简介	超耐操
库存	444
类型	耳机
<input type="button" value="提交增加"/>	

图 11、增加商品界面

点击商品管理，在指定商品处点击删除：

2	红米note			<div>116.62.105.78:8080 显示 确定删除?</div>	666	手机	删除
3	mipad			698 大屏幕更清晰	888	平板	删除
4	mibook			4988 机制性价比	666	笔记本	删除
5	黑鲨			4988 游戏手机，腾讯优化	777	手机	删除
6	小米耳机 airpro			213 降噪，蓝牙	1000	耳机	删除
7	红米耳机			99 超耐噪	444	手机	删除
8	redmi耳 机			230 经典	333	耳机	删除

图 12、删除商品界面

点击商品管理，点击查询商品，输入关键字“米”，可以获得以下结果：

后台管理

订单管理

商品管理

用户管理

操作日志

登录记录

修改密码

退出

全部商品

新加商品

查询商品


ID	名称	封面	图片1	图片2	价格	简介	库存	类型
1	小米10				5688	水桶机	999	手机
2	红米note				2099	超长续航	666	手机
6	小米耳机airpro				213	降噪，蓝牙	1000	耳机

图 13、按关键字查询商品界面

另外商户登录后，可以查看操作日志和登录记录。

1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:21:55
1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:22:03
1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:22:36
1	zs	120.238.248.183	商品	0	redmi耳机	增加	2021-05-22 23:23:26
1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:27:19
1	zs	120.238.248.183	商品	8	redmi耳机	删除	2021-05-22 23:27:41
1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:29:03
1	zs	120.238.248.183	商品	7	红米耳机	修改	2021-05-22 23:29:26

图 14、日志记录界面

点击修改密码:

后台管理

订单管理

商品管理

用户管理

操作日志

登录记录

修改密码

退出

用户名

zs

原密码

新密码

提交修改

图 15、商户修改密码界面