**深 圳 大 学**

本 科 毕 业 论 文（设计）

题目: **基于zookeeper的分布式电器购物商城的搭建与实现**

姓名:  **赵楷佳**

专业: **软件工程**

学院: **计算机与软件学院**

学号: **2013150399**

指导教师: **张席**

职称： **副教授**

2017年 4 月 26 日

摘要

纵观传统的电商系统，作为大物件的电器行业，也出现了类似于国美，苏宁等这样的大集团公司。但就家乡地区来说，在网上购物的需求没有想象中像大城市那样的巨大。因为在电器这个销售行业中，其中最难解决的还是售后服务，与运输过程出现的质量问题。

本系统开发的总体需求任务，是为农村用户群提供高度客户化和互联网化的浏览器或手机界面。引导用户去直接上互联网浏览以及购买，并且解决保修的距离问题。系统购物的web端，简约风格，支持电脑，手机端微信端进行购买。后台管理系统，通过权限的管理，以及资源的管理，对产品，用户订单，以及保修单进行维护和管理。最后，通过三个不同的tomcat和JNDI数据源管理对这两个系统和同一个数据库连接在一起。

针对家乡地区电器实体店的现状，方便家里更好的管理订单和维修单等，开发了这样一个集购买，维修，以及管理为一体的两个web端应用。运用zookeeper注册中心以及RPC远程调用方式实现，满足这种现象的需求。

关键词：电子商务，JNDI，J2EE，网上商店

第一章 绪论

1.1 选题的依据与意义

在淘宝，天猫，京东等大型网络电子商务网站出现之后，人们的消费观念也发生了深刻的变化，足不出户的购物方式已经走进了人们的生活，变成无可取代的一种生活方式，实体市场并不一定是唯一购物的地点。就大城市而言，快递行业也已经普及到了每个角落，每一个楼层。就电器行业来讲，国美集团，苏宁集团也已经成为了行业巨头。人们可以非常轻松的通过网络途径就可以购买商品并且送货到家，并且提供安装和售后问题的服务。最大限度的降低了人们对于实体市场的依赖性以及耦合性。但就家乡地区的发展情况而言，苏宁，国美还没有普及到这些地方，他们对于电器的采购依旧是通过传统的实体店购买，然后进行送货上门和安装。在天猫，国美，苏宁等大型的电器销售平台购买的人，是相对比较少的，因为他们怕快递过程的商品质量问题，担心售后服务的问题。对于这个现象而言，是有需求做一个结合自己的实体店和网络，结合实体店以及解决电器行业保修困难的情况，推一款适用于区域性电器购物的网络应用。

这款应用，应该体现出区域性以及个体性，需要有实体门店的个体网络应用。物流的方式可以通过自己实体店的配送，出现这样一个做区域性电子商务的想法，是为了简化家乡地区电器销售批发以及售后服务的区域性流程，为广大的用户带来方便，实现随时买，随时送，随时换，随时售后服务的宗旨。

1.2 国内外研究现状

互联网的广泛应用和J2EE技术的愈发成熟，也因为在各类电商大巨头的成功例子的基础上。确定了网上购物的热门性和可行性。在美国，日本这些信息化高速发达的国家，网络商店的电子商城更是发展非常迅速，在一些传统的商品上，美国有世界级超一流的零售商，如科罗格，亚马逊，沃尔玛凯马特等纷纷跻身于网络经商的行列中。就电器行业来讲，涉及的服务要比传统的商品要多，会出现售后服务，退换货很麻烦的问题等。在美国，电器行业的电子商务模式，依旧是全国性质的销售平台，没有细化到某个区域。例如ebay，newegg等平台。因此，对于区域性的购物网上应用，具有一定的开发价值。

根据我的初步统计，在家乡片区，在30公里的范围内，电器销售商超过20家，电器商品的样式大多大同小异，价格差距不大，大多数通过租商铺进行电器销售的个体经营户，对商品售卖的价格会相应的抬高，因为需要解决商铺租金的问题。通常都是客户自己上门到个体经营店，去买商品，并且达成购买协议后，进行配送安装等服务。他们的经营方式都是一样的，就是通过线下进行销售，没有和现如今非常炙手可热的互联网结合起来。对于回头客，以及很熟悉的用户群，他们只需要打个电话，商户直接配送即可。为了解决以及方便用户的选购，改变这种单一的电器购物行为，能够更好的利用线上，使自己的竞争地位更加处于上风。因此，自己通过制作一款亲民简约的用户软件系统，预示着另外一种区域性电器销售模式的改进，以及对保修问题的解决提供了互联网环境。同时，也可以对商品货物进行可控的管理。及时的察觉哪些商品卖的最多，并且也可以计算出商品的销售量。对于一个个体的经营户来说，这样的模式可以省去很多线下的订单管理，密密麻麻的订单不需要通过纸上来体现，通过互联网的记录就能达到相同的效果，并且查找非常简单，不需要找订单通过手动，纸张去管理。极大的方便了个体经营户的管理形式。并且保修的过程得到了简化，维修的过程也能得到简化，不需要用手机纸张对维修人的地址和电话进行记录，而是通过网上填单就能达到相同的效果。这样的系统虽远远比不上正式庞大的国美，苏宁。但是却能在区域性发生一定的化学反应。

第二章 J2EE与zookeeper分布式技术概述与配置

2.1 J2EE框架平台技术介绍

J2EE对于目前很多现有的应用程序集成有着强有力的支持，J2EE平台简化了企业解决方案的开发，部署和管理的相关问题。从典型的J2EE三层结构剖析，其有三层重要的结构，分为表现层，中间层(即控制层，业务逻辑层)和数据服务层。这三层结构将数据的访问，业务规则及合法性校验等工作放在中间层进行处理，客户端不直接与数据库进行交互，而是通过组件与中间层进行连接，再由中间层与数据库进行交互。

表现层：目前最传统的表现层结构为JSP技术，java可通过Spring的JSTL语言或者是EL表达式，将从后台处理得到的数据在JSP层进行解析之后返回到前台。

Web层，中间层，控制层：控制层框架常用的有struts，SpringMVC，springMVC还可以集成freemarker模板引擎，基于模板和数据来生成输出文本。

Service层(业务逻辑层)：负责实现业务逻辑，业务逻辑层以DAO层为基础，通过对DAO组件进行包装，完成系统所需要的业务逻辑，包括数据的逻辑处理等。

DAO层(数据持久化层)：负责与持久化对象进行交互，该层封装了JDBC或ODBC等连接数据库组件，将数据库的每个记录转化为ORM对象，封装了数据的增删改查的操作。

其中Spring框架贯穿了整个中间层，将Web层，DAO层，Service层以及PO无缝的整合在一起，因此Spring是Java框架中最核心的成员，它负责了对象的生存周期等业务，将对象通过单态模式或者工厂模式，在ApplicationContext统一进行管理。

在严格的分层结构中，控制层依赖于业务逻辑层，但不会与任何业务逻辑组件耦合，业务逻辑层依赖于DAO层，也不会与任何具体的DAO组件进行耦合，并且采用面向接口的编程方式。引入接口需要Spring进行管理，也能通过AOP切面对业务进行中间干预。

2.2 分布式服务框架Zookeeper，dubbo介绍

Zookeeper框架是一个分布式的应用程序协调服务软件，是Google的Chubby的一个开源实现，同时是Hadoop和Hbase的重要组件。它为分布式的应用提供一致性的服务，包括配置维护，域名服务，分布式同步，组服务等。本系统应用到Zookeeper系统，整合alibaba开源dubbo框架进行开发。

Zookeeper的核心是原子广播，这个机制保证各个server之间的同步。实现这个机制的协议是Zab协议，Zab协议有两种模式，他们分别是恢复模式(选主)和广播模式(同步)，当服务启动或运行过程中，出现服务奔溃，Zab进入恢复模式，当领导者被选举出来，且大多数Server完成了和leader的状态同步以后，恢复模式就结束。状态同步保证了leader和Server具有相同的系统状态。每个Server在工作过程中有三种状态：

* LOOKING：当前Server不知道leader是谁，正在搜寻
* LEADING：当前Server即为选举出来的leader
* FOLLOWING：leader已经选举出来，当前Server与之同步

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。Alibaba的一个开源组件，其中节点的角色包含：

* Provider: 暴露服务的服务提供方。
* Consumer: 调用远程服务的服务消费方。
* Registry: 服务注册与发现的注册中心。
* Monitor: 统计服务的调用次调和调用时间的监控中心。
* Container: 服务运行容器。

出现dubbo框架是随着互联网的发展，网站应用的规模不断扩大，常规的垂直应用架构无法应对需求，因此出现分布式服务架构，dubbo提高了业务复用和整合分布式服务框架RPC。

Dubbo框架的设计一共划分为10层，分别为：

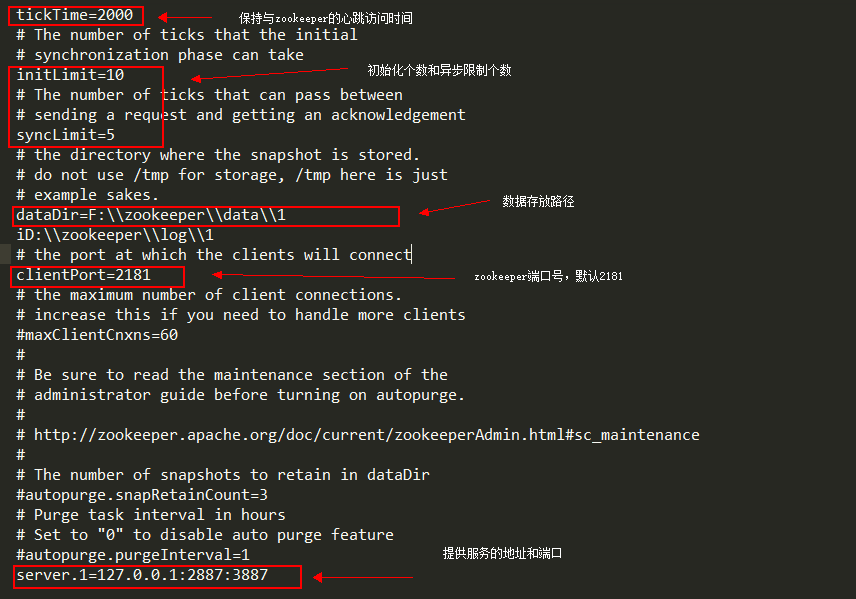
* 服务接口层(service):该层是实际业务逻辑相关的，根据服务提供方生产者和服务消费方的业务设计对应的接口和实现。
* 配置层（Config）：对外配置接口，以ServiceConfig和ReferenceConfig为中心，可以直接new配置类，也可以通过spring容器解析配置生成配置类。
* 服务代理层（Proxy）：服务接口的透明代理，生成服务的客户端Stub和服务器端Skeleton，以ServiceProxy为中心，扩展接口为ProxyFactory。
* 服务注册层（Registry）：封装着服务地址的注册与发现，以服务的URL为中心，扩展接口为RegistryFactory、与Registry和RegistryService。可能没有服务注册中心，此时服务提供方生产者直接暴露服务。
* 集群层（Cluster）：封装多个提供者的路由信息及负载均衡，并桥接到注册中心，以Invoker为中心，扩展接口为Directory、Cluster、Router和LoadBalance。将多个服务提供方生产者组合为一个服务提供方，实现对服务的消费方来透明其暴露的接口服务，只需要与一个服务提供方进行交互。
* 监控层（Monitor）：RPC调用次数和调用时间监控，以Statistics为中心，扩展接口为MonitorFactory、Monitor和MonitorService。
* 远程调用层（Protocol）：封将RPC调用，以Invocation和Result为中心，扩展接口为Protocol、Invoker和Exporter。Protocol是服务域，它是Invoker暴露和引用的主功能入口，它负责Invoker的生命周期管理。Invoker是实体域，它是Dubbo的核心模型，其它模型都向它靠扰，或转换成它，它代表一个可执行体，可向它发起invoke调用，它有可能是一个本地的实现，也可能是一个远程的实现，也可能一个集群实现。
* 信息交换层（Exchange）：封装请求响应模式，同步转异步，以Request和Response为中心，扩展接口为Exchanger、ExchangeChannel、ExchangeClient和ExchangeServer。
* 网络传输层（Transport）：抽象mina和netty为统一接口，以Message为中心，扩展接口为Channel、Transporter、Client、Server和Codec。
* 数据序列化层（Serialize）：可复用的一些工具，扩展接口为Serialization、 ObjectInput、ObjectOutput和ThreadPool。

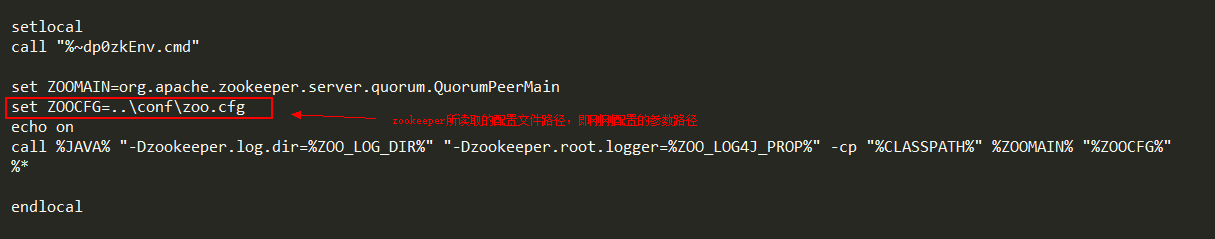
Dubbo提供的注册中心有Zookeeper注册中心，Redis注册中心，simple注册中心，Multicast注册中心，本系统采用Zookeeper注册中心。无论是服务提供方还是服务消费者，他们都需要对服务调用的实际状态进行有效的监控，从而改进服务的质量。

2.3 分布式服务框架Zookeeper，dubbo配置

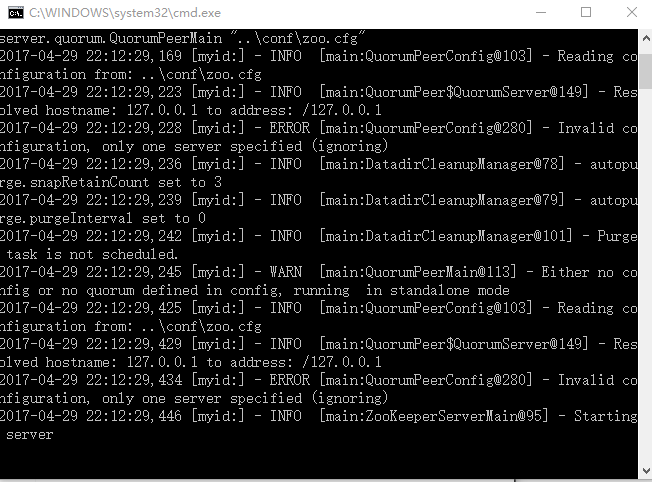
2.3.1 Zookeeper的安装和配置

1. Windows操作系统下的安装和配置:

* <https://zookeeper.apache.org/> 到该链接地址中下载Zookeeper的安装包并解压，本系统采用的是zookeeper 3.4.8版本进行开发。
* 打开conf目录，将zoo\_sample.cfg赋值一个，更改名字为zoo.cfg
* 
* 到bin目录下修改zkServer.cmd文件如下图



* 启动Zookeeper的zkServer.cmd文件

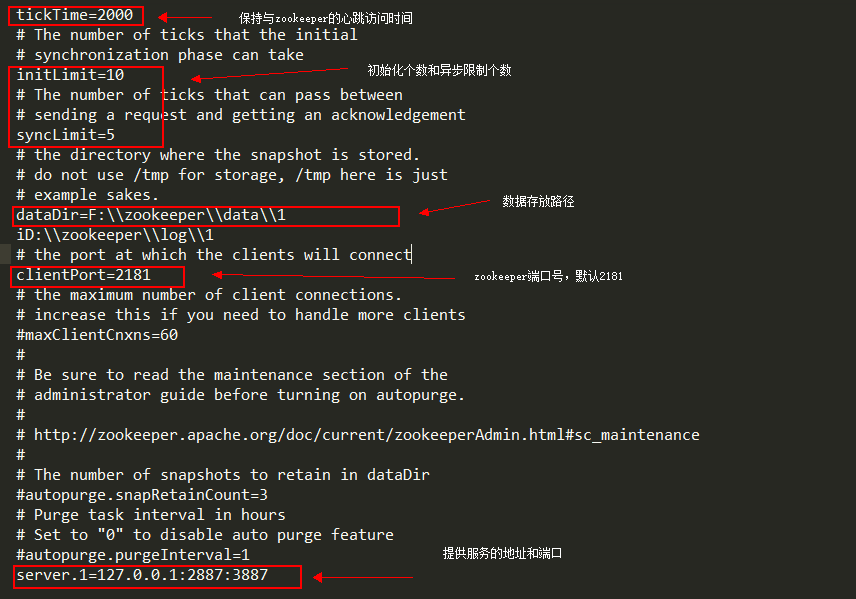


2. Linux操作系统下的安装和配置:

* 创建 /usr/local/Lauren/zookeeper 文件夹：
* mkdir -p /usr/local/Lauren/zookeeper
* 进入到 /usr/local/Lauren/zookeeper 目录中：
* cd /usr/local/Lauren/zookeeper
* wget <https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-3.4.9/zookeeper-3.4.9.tar.gz，下载3.4.9>安装包
* 解压缩 zookeeper-3.4.9.tar.gz ，tar -zxvf zookeeper-3.4.9.tar.gz
* 进入到 /usr/local/Lauren/zookeeper/zookeeper-3.4.9/conf 目录中：

cd zookeeper-3.4.9/conf/

* 复制 zoo\_sample.cfg 文件的并命名为为 zoo.cfg，cp zoo\_sample.cfg zoo.cfg
* 用 vim 修改zoo.cfg文件



* dataDir=/usr/local/Lauren/zookeeper/zookeeper-3.4.9/data,并且在这个目录中创建这个文件夹
* dataLogDir=/usr/local/Lauren/zookeeper/zookeeper-3.4.9/logs，并且在这个目录中创建这个文件夹
* 保存并关闭 zoo.cfg 文件:
* 进入到 /usr/local/Lauren/zookeeper/zookeeper-3.4.9/bin 目录中：
* 用 vim 打开 /etc/ 目录下的配置文件 profile：
* vim /etc/profile
* export ZOOKEEPER\_HOME=/usr/Lauren/services/zookeeper/zookeeper-3.4.9/

export PATH=$ZOOKEEPER\_HOME/bin:$PATH

export PATH

* 使 /etc/ 目录下的 profile 文件即可生效：source /etc/profile
* 启动 zookeeper 服务：zkServer.sh start
* 如打印如下信息则表明启动成功：

ZooKeeper JMX enabled by default

Using config:/usr/local/Lauren/zookeeper/zookeeper-3.4.9/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

查询 zookeeper 状态： zkServer.sh status

* 关闭 zookeeper 服务：zkServer.sh stop

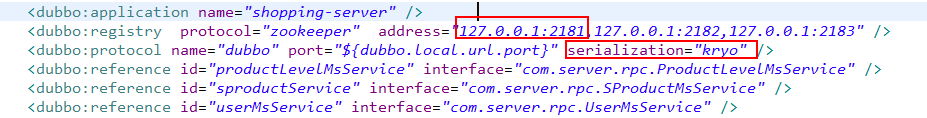
2.3.2 spring与dubbo zookeeper的整合

服务端的配置：





消费端的配置：



2.4 nginx服务器的介绍与配置

Nginx是一个高性能的反向代理HTTP服务器。聚焦高性能，高并发，低内存消耗问题，并且为多种web服务器提供功能特性，如负载均衡，缓存，访问控制，以及带宽控制等。Nginx是目前互联网最流行的开源web服务器软件。他提供了其必要的关键功能并且用于方便将下列功能从应用层剥离到更高效的边缘web服务器层：长连接处理、并发、SSL、静态内容、缓存压缩、连接和请求限速，以及HTTP媒体流等。Nginx服务器同时也允许直接整合memcached、Redis或者其他的NoSQL解决方案，增强为处理大规模并发用户的性能。

Nginx通过工作进程来分配任务，其不为每个连接派生进程或线程，而是由worker进程进行监听共享套接字来接受新请求，并且会使用高效的循环来不断的处理数千个连接，这个工作由操作系统的内核机制来完成，不使用仲裁器或分发器来分发连接。

Nginx在运行时会产生多个进程，一个master进程以及多个worker进程，master进程主要负责读取校验配置文件，创建绑定关闭套接字，启动，终止维护所配置的数目的worker进程，不中断的刷新配置文件，不中断服务升级程序，重新打开日志文件以及编译嵌入Perl脚本。Worker进程则接受处理来自客户端的连接，提供反向代理和其他nginx所具备的功能以及过滤功能，worker进程是web服务器实际执行者。

Centos平台的配置与安装：

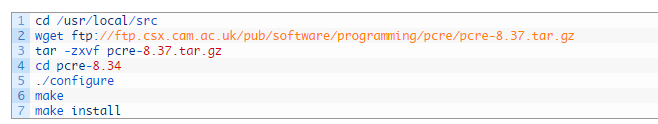
1. 安装make

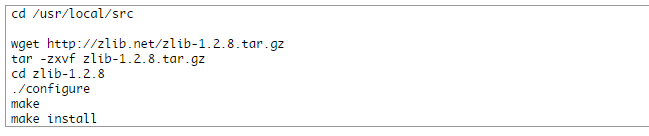
yum -y install gcc automake autoconf libtool make

2. 安装g++

yum install gcc gcc-c++

3. 安装PCRE库

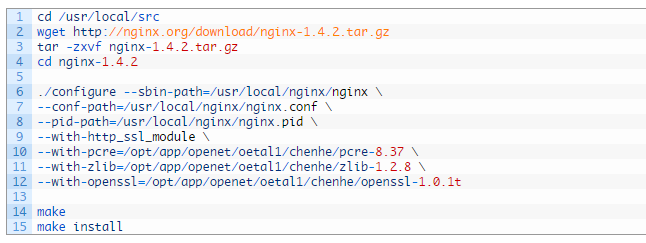
4. 安装zlib库



5. 安装ssl



6. 安装nginx





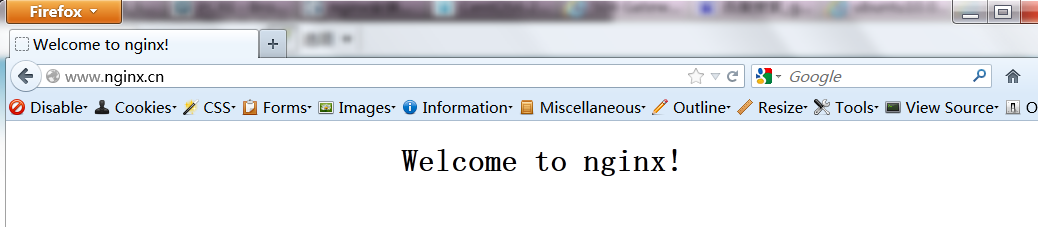
6. 启动

确保Centos系统的80端口没有被其他程序占用，运行/usr/local/nginx/nginx 命令来启动 Nginx。

查询80端口占用可通过



7.成功



第三章 区域性电器商城系统的总体需求分析

3.1 区域性电器商城行业的基本需求

传统销售和购物方式以及个体商户管理复杂是开发个体商户网上购物电器商城的主要原因。家乡地区的个体电器销售商，都是通过实体线下进行电器的销售，从用户的角度来讲，需要到实体店去选购，或者是打电话订购，并且电器的性能也要通过口头的介绍来让用户明白电器的性能，以及应该选哪款商品。从个体经营商户的角度出发，对于订单的管理非常的复杂。在某种情况下，有时候客户的订单电话需要对电话进行录音，或者是纸张进行记录收货的地址，产品的型号等等信息。而且订单都是通过纸张证明。保修的日期也需要额外的进行维护。假设用户买的商品，在保修期一年时间内，出现了产品问题，那么商户还要去找相应的订单，查看其购买的时间，是否在保修期内，或者是需要用户提供相应的依据证明。显然会有很多歧义。所以开发区域性个体电器商户的系统，从用户的角度，或者商户的角度都起到了一个方便管理和简化流程的作用。这一系统的主要优势在于：

首先，从竞争的角度来讲，在方圆30公里以内的所有电器商户中，他们的销售模式都是单一的，倘若出现一个与互联网相结合的系统供用户使用，让用户足不出户也能买到商品，对于固定回头客或者用户群来讲，只需要简单的操作就能买到商品，无疑降低了很多复杂的流程。连电器最后的保修维护都不需要用户。

其次，从个体经营商户角度来讲，运用这个系统可以随时随地的查询到订单的情况，查询某商品订单是否超过保修期，并且方便对回头客用户群的信息进行管理。省去了很多纸张的记录流程。另外一方面，网站可以通过资源和角色的管理，关闭某个管理员的权限或者不看哪个页面都可以通过设置来完成。

在目前电器商城的盛行之下，如出现国美，苏宁等大型的网上购物商城。但是其没有覆盖到农村地区，也很难解决一方销售的问题。因此这个系统对于个体销售户，建立了具有可移植性，可扩展性，方便易用性的网上购物系统。

3.2 区域性电器商城系统的总体设计

区域性电器商城系统，其开发主要包含前端应用程序的开发和后台数据服务以及数据库的建立和维护三个方面。对于前端开发，要求应用程序具备功能完备，并且风格简约，容易操作和使用的特点。而对于后台数据服务，则要求数据的逻辑，以及获取的速度。并且包括并发性等等。对于第三方面的数据库而言，则要保证数据的一致性，完整性和安全性。

对于区域性电器商城系统，主要考虑其功能的完整性，商城提供商品的详情展示，购物等功能，并且适配手机端，用户直接通过手机进行操作购买即可。并且还为用户提供了，购买之后的商品评价，保修管理，以及不定时的推出新品和热搜活动，方便与用户进行交互。

本系统开发的总体任务，是为了给用户提供客户化易使用的手机浏览器界面。引导用户查找，浏览并购买所需的商品。本系统分为两个子系统，第一个系统为用户商城系统，系统模块分别是商品浏览搜索模块，注册登录模块，购物车模块，购买结算模块，个人订单管理模块，商品评论模块，保修或维修模块。第二个子系统为个体商户管理系统，该系统功能模块分别是登录模块，角色管理模块，权限管理模块，用户订单管理模块，用户信息管理模块，产品管理模块，资源管理模块，用户报修管理模块。

用户商城子系统功能界面如下图3-1所示



图3-1 用户商场子系统模块图

商城系统是通过后台管理系统，对相应的产品进行管理，展示到用户进行购买和查看。

商品浏览搜索模块：包括商品的详情浏览，评论浏览，以及产品的详细参数浏览等，同时可以进行商品的搜索，适配商品名称或者类型或者品牌的搜索等，准确定位到商品信息服务上面来。

注册登录模块：用户可以随便在商城系统中查看各类商品等，购买或查看自己信息的时候，需要进行登陆，登录分为两种方式，一种是手机验证码登录，一种是普通的密码登录，都可以登陆到商城系统进行商品的购买。用户的账号，通过用户注册，注册时需要提供真实的手机号码，收到验证码验证正确之后才能注册成功。

购物车管理模块：用户需要在登录之后，对满意需要购买的产品进行加入购物车，等到需要进行结算的时候，点击我的购物车模块进行买单。买单完购物车自动清空。

购物结算模块：用户在登录之后，可以直接对某个商品发起购买，并微信支付生成后台订单号，或者是通过加入购物车的方式，等到商品凑足再一次性进行购买，同样调用微信支付功能。

我的订单和地址管理模块：在购物之前需要对用户的地址进行管理，用户可以添加自己的常用地址和联系人信息，在购买的时候需要指定默认的地址进行购买。并且购物完之后可以查看到订单的状态以及买的商品的信息等。

评论模块：用户购买之后，可以对购买交易完成之后的产品，进行质量的评价，这些评价会到商品详情的评价模块可供用户查看。

保修维修模块：用户可以在购买之后凭借订单进行商品的售后服务，填写保修单，即可让商户知道这件商品需要保修，也可以在不买商品的情况下直接填写维修单对其进行维修发起。

商户后台管理系统功能界面如下图3-2所示



图3-2 商户管理系统模块图

商户管理系统主要是对用户的信息以及产品，订单，资源进行统一的管理的系统。

产品管理模块：对新上映进行销售的商品进行上传，包括对其型号，商品名称，品牌管理，类型管理和图片，图片详情进行添加，以及对添加成功的商品进行修改删除，以及让其上热搜，上新品区等产品管理功能。

登录模块：登录的账号密码采用MD5对账号和密码进行合起加密，并且将登陆的ip和log进行记录，本系统不提供注册的功能，而且防止恶意的多次登陆，并且将账户分成角色化和资源可视化进行管理。

角色管理模块：将角色分为互联网的角色模式进行管理，分别有产品经理，产品专员，管理员，超级管理员。超级管理员可以统一对其下面所有的角色进行管理，包括资源的可视管理和角色分配管理等。

权限管理模块： 超级管理员拥有对一切角色的权限管理功能，对资源进行权限化的管理，超级管理员可以为角色指定哪些资源他们有权限进行访问，哪些资源他们无法访问到等等控制流。

用户信息管理模块：对其在系统中注册的用户进行信息的可视和管理，并且可以知道用户的常用地址默认地址等信息，方便管理。

用户订单管理模块：商户可以对用户发起的订单进行及时的查看，收到订单之后修改其状态通知用户订单的改变。并且会显示到用户购买的数量金额，产品型号，收货的联系人姓名，手机号码，详细地址等。

保修维修管理模块：商户查询到用户发起的订单保修或者是自己其他的物品进行维修申请，倘若是订单保修，则会有订单号，通过订单号可以查询到其购买商品的时间确保其是否在保修期内，另外没有订单号的维修单则是用户自己发起的商品维修，同样填写和显示需要维修或保修服务的联系人电话，姓名地址等信息。

3.3 区域性电器商城系统的总体业务流程图

区域性电器商城系统首先提供用户的随时浏览功能，若未登录的用户只能浏览产品的详情，评论，商品信息，新品，热搜，专题列表等信息。倘若用户需要对商品进行购买行为，加入购物车，或者是对个人信息等进行查阅，则需要通过登录才能进行，没有账号登陆的用户需要注册账号，登录和注册采用手机验证码操作，保证用户的真实性，每个手机号码只能注册一次。登录后的用户可以对商品进行购买，评价，订单管理，保修管理，收藏商品，管理自己的账号等信息。同时商品查找方面，采用关键字搜索，商品类型搜索，商品品牌搜索进行匹配。同时，本系统对用户购物车的信息进行保存，即使用户退出系统，下次再登陆系统也能看到其收藏以及加入到购物车的商品信息，购买之后进行清空。流程业务图如3-3



图3-3 商城总体业务流程图

商户管理系统首先通过账号登陆，系统根据它的等级或者权限加载相应的菜单项，而账户的权限以及它所需加载的菜单设置通过超级管理员进行设置管理。登录进去的账户可以对模块进行管理，分为几大模块，分别是产品管理，计划管理，账户管理，系统设置。

产品管理：品牌管理，产品管理，类别管理，其中品牌管理是对产品的品牌进行添加，包括logo上传，产品管理对商品的信息进行添加，包括详情图片上传等，类别管理目前分成7大类，分别是电视，冰箱，洗衣机，空调，厨房用品，浴室用品，其他。

计划管理：订单管理，维修管理，将用户的订单和维修单进行统一的管理。

账户管理：用户列表，用户收货地址管理，将用户的信息进行管理，以及查阅。

资源设置：操作日志，用户管理，角色管理，资源管理。操作日志记录了商户登录的情况，用户管理管理可以使用该系统的账号，角色管理是指超级管理员对其账户进行权限和角色的分配，资源管理是超级管理员对账户进行资源的分配操作。流程业务图如3-4所示：



图3-4 商户后台管理系统业务流程图

3.4 本章小结

本章主要介绍了区域性电器商城的基本需求，着重介绍了本论文所设计的系统前，后台，以及电器商城系统，商户后台管理系统的系统前后业务流程图。

第四章 系统软件环境架构与数据库设计

4.1 区域性电器商城和商户后台管理系统的环境

区域性电器商城要求：具有IOS或android系统的智能手机，拥有safari浏览器或者主流QQ浏览器，chrome浏览器等手机浏览器等。电脑处理器且满足以下最低要求的计算机：最低64MB内存，最小4GB硬盘，鼠标，键盘，且需要主流浏览器，如QQ浏览器，chrome浏览器，IE(9.0或者更高版本)，firefox浏览器等。

商户后台管理系统要求：具有电脑处理器且满足以下最低要求的计算机：最低64MB内存，最小4GB硬盘，鼠标，键盘，且需要主流浏览器，如QQ浏览器，chrome浏览器，IE(9.0或者更高版本)，firefox浏览器等。

WEB服务器硬件要求：Centos LINUX环境，内存不低于600MB，具备2M带宽，网络支持WEB服务软件tomcat，数据库服务器软件mysql。

4.2 区域性电器商城和商户后台管理系统的设计

4.2.1 数据库系统的基本概念

数据库是按照数据结构的方式，对数据进行组织，存储和管理的仓库，随着信息技术和市场的发展，oracle公司的mysql，oracle数据库，还有DB2, SQLServer等关系型数据库，已经慢慢的应用到各个领域。银行用的数据库大多都是DB2，解决了不仅仅只是存储和管理数据，而是能够转变成用户所需要的各种数据模型。数据库系统，是由数据库机器管理软件所组成的，它是一个实际可运行的存储介质，通过硬盘文件存储，成为为维护和应用系统提供数据服务的软件系统，是存储介质，处理对象和管理系统的集合体。

数据库的基本结构分为三个层次，从不同的角度分析这三个层次，分别是1.概念数据层，这一层指出了每个数据之间的逻辑联系，是存储记录的集合。2.物理数据层，它是数据库的最内层，是物理存储设备上实际存储的介质。概念数据层所能表示的是一个或一些特定用户所使用的数据集合，数据库会在不同的层次之间进行联系映射和转换。

4.2.2 数据库表总体概念结构设计

数据库管理软件：mysql 5.7版本

数据库分为两个库，1. shopping-manager 2. shopping-if

4.2.2.1 区域性电器商城系统的数据库设计

表4-1 区域性电器商城系统数据库的所有表名和功能

|  |  |  |
| --- | --- | --- |
| 表编号 | 表名称 | 中文说明 |
| 1 | t\_address | 用户地址表 |
| 2 | t\_application\_setting | 系统设置表 |
| 3 | t\_band | 品牌管理表 |
| 4 | t\_favor | 用户收藏表 |
| 5 | t\_marketcar | 用户购物车表 |
| 6 | t\_order | 用户订单表 |
| 7 | t\_product | 商品信息表 |
| 8 | t\_product\_res | 商品图片信息表 |
| 9 | t\_repair | 保修或维修表 |
| 10 | t\_smslog | 短信通知信息表 |
| 11 | t\_sort | 商品分类表 |
| 12 | t\_talk | 评论信息表 |
| 13 | t\_user | 用户信息表 |

表4-2 商户后台管理系统数据库的所有表名和功能

|  |  |  |
| --- | --- | --- |
| 表编号 | 表名称 | 中文说明 |
| 1 | SYS\_LOG | 操作日志表 |
| 2 | SYS\_RESOURCE | 资源表 |
| 3 | SYS\_ROLE | 角色表 |
| 4 | SYS\_ROLE\_RESOURCE | 角色-资源关联表 |
| 5 | SYS\_USER | 用户表 |
| 6 | SYS\_USER\_ROLE | 用户-角色关联表 |
| 7 | t\_application\_setting | 系统设置表 |

4.3.3 数据库表逻辑结构的设计

实体表的表名命名规则为“t\_”+该表的英文含义，若是复合单词，第一个字母需要大写。数据库中主键的命名规则为：“PK\_” + 表名 +“\_”+ 字段名，如（FK\_T\_USER\_ID，字母全部用大写字母），其中字段类型为int(整型)，dateTime(日期型)，varchar(可变的字符串型)，char(字符型)，text(文本型)，bigint(大整型)，decimal(小数点的大整型)。

4.3.3.1 区域性商城系统数据表逻辑设计

本系统所使用的13个数据库表。涵盖了用户操作的所有所需设计，其中有用户地址表，系统设置表，品牌管理表，用户收藏表，用户购物车表，用户订单表，商品信息表，商品图片信息表，保修或维修表，短信通知信息表，商品分类表，评论信息表，用户信息表。接下来一一介绍各个表的表结构。

表4-3 t\_address---用户地址表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户地址表 | | 英文名称 | t\_address | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | user\_name | varchar | 20 | 是 |  | 用户名 |
| 3 | user\_id | varchar | 50 | 是 |  | 用户关联id |
| 4 | user\_address | varchar | 255 | 是 |  | 用户地址 |
| 5 | user\_phone | varchar | 11 | 是 |  | 用户手机 |
| 6 | status | tinyint | 4 | 是 | 0 | 地址状态 |

用户地址表为用户所填录的地址信息表，包括主键id，用户姓名，发起交易的用户id，用户的收货地址，用户的联系电话，以及对应的状态status，0 表示不是默认地址，1表示默认地址，每个user\_id的用户只能拥有一个默认地址的设置。

表4-4 t\_application\_setting ---系统设置表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 系统设置表 | | 英文名称 | t\_application\_setting | | |
| 主键 | FID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | FID | int | 11 | 是 |  | 主键ID |
| 2 | FKEY | varchar | 50 | 是 |  | 设置KEY |
| 3 | FVALUE | varchar | 200 | 是 |  | 设置值 |
| 4 | FDESC | varchar | 50 | 是 |  | 描述 |
| 5 | FCACHED | Int | 1 | 是 | 1 | 是否启用缓存 |

系统设置表设置了系统相关的配置信息，包括加解密的key以及系统的固定配置，FKEY是程序通过该key名获取到FVALUE对应的属性，FDESC是对这个记录的描述，FCACHED表示是否开启了缓存。系统设置是通过hibernate结构来进行访问，获取系统的各种参数，参数的修改只需要修改数据库即可，这样子去除了很多耦合性。

表4-5 t\_band ---品牌管理表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 品牌管理表 | | 英文名称 | t\_band | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | band\_name | varchar | 50 | 是 |  | 品牌名字 |
| 3 | band\_id | varchar | 50 | 是 |  | 品牌id，唯一 |

品牌管理表为商品的品牌管理表，包括主键id，品牌姓名，以及品牌id，品牌id为程序生成的长度大约为30的唯一序列号，该字段设置了unique唯一索引。记录着商品的品牌信息。

表4-6 t\_ favor---用户收藏表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户收藏表 | | 英文名称 | t\_favor | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | user\_id | varchar | 50 | 是 |  | 关联的用户id |
| 3 | prod\_id | varchar | 50 | 是 |  | 关联的产品id |

用户收藏表为用户登录之后，对商品进行收藏，以便下次直接通过我的收藏列表找到对应商品的用户管理表，包含主键id，user\_id为关联对应的用户唯一id，prod\_id为关联对应的商品唯一id，此表所关联查询的信息直接通过user\_id和prod\_id进行关联查询，去除了数据的冗余性。

表4-7 t\_marketcar---用户购物车表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户购物车表 | | 英文名称 | t\_marketcar | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | user\_id | varchar | 50 | 是 |  | 关联的用户id |
| 3 | prod\_id | varchar | 50 | 是 |  | 关联的产品id |
| 4 | prod\_count | int | 11 | 是 | 0 | 加入购物车时的数量 |

用户购物车表为用户登录后对商品进行加入购物车操作的统一管理表，其中包含了主键id，user\_id为关联的user\_id，prod\_id为关联的商品id，通过user\_id查找到其加入购物车的商品集合，再根据查找的商品id查询对应的信息，达到购物车统一管理的目的。

表4-8 t\_product\_res ---商品资源表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 商品资源表 | | 英文名称 | t\_ product\_res | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | res\_id | varchar | 50 | 是 |  | 资源id |
| 3 | res\_parent\_id | varchar | 50 | 是 |  | 父级资源 |
| 4 | res\_seq | Int | 11 | 是 |  | 排序号 |
| 5 | prod\_id | varchar | 50 | 是 |  | 对应的商品id |

商品资源表为商品对应的图片资源管理表，默认情况下，一个商品具有四张图片上传，第一张为产品头像，第二三四张为产品详情图片。其中包含主键id，res\_id为该图片对应的唯一资源id，res\_parent\_id对应的父级资源id，res\_seq表示其的显示顺序，prod\_id

为对应的商品id，这样设计的目的在于，寻找图片的时候，首先通过父级资源id查询到父级文件夹，再通过对应的资源id查找到对应的图片信息。这样子的设计会减除很多冗余情况，也能够直接找到对应的图片资源。

表4-9 t\_order---用户订单表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户订单表 | | 英文名称 | t\_order | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | user\_id | varchar | 50 | 是 |  | 关联的用户id |
| 3 | prod\_id | varchar | 50 | 是 |  | 关联的产品id |
| 4 | order\_id | varchar | 50 | 是 |  | 唯一订单号 |
| 5 | status | tinyint | 4 | 是 | 0 | 订单状态 |
| 6 | talk\_status | tinyint | 4 | 是 | 0 | 评论状态 |
| 7 | buy\_count | int | 11 | 是 | 0 | 购买的数量 |
| 8 | money\_sum | decimal | 12 2 | 是 |  | 总金额 |
| 9 | attach\_address | bigint | 20 | 是 |  | 关联的收获地址id |
| 10 | create\_time | datetime | 0 | 是 |  | 订单创建时间 |

用户订单表为用户登录后对商品进行加入购物车操作之后的结算，或者是对商品发起直接购买行为的统一管理表，其中包含了主键id，user\_id为关联的user\_id，prod\_id为关联的商品id，order\_id为程序生成的长度约为30的唯一订单号码，status记录着商户是否对这笔订单察觉并做出处理，talk\_status为该订单完成后是否被评论，buy\_count表示用户购买此产品的个数，money\_sum表示购买结算的总金额，attach\_address为购买时指定的默认地址，以及create\_time创建订单的时间，方便之后的保修功能。

表4-10 t\_product ---商品信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户订单表 | | 英文名称 | t\_order | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | code\_id | varchar | 50 | 是 |  | 产品型号 |
| 3 | prod\_id | varchar | 50 | 是 |  | 唯一id |
| 4 | prod\_name | varchar | 255 | 是 |  | 商品名称 |
| 5 | prod\_detail | Text | 0 | 是 |  | 商品详情 |
| 6 | prod\_type\_name | varchar | 20 | 是 |  | 种类名称 |
| 7 | prod\_prize | decimal | 10 2 | 是 |  | 商品单价 |
| 8 | prod\_free\_time | varchar | 10 | 是 |  | 商品保修期 |
| 9 | prod\_sum | Int | 11 | 是 |  | 商品库存量 |
| 10 | prod\_sell\_sum | int | 11 | 是 | 0 | 销售量 |
| 11 | status | tinyint | 4 | 是 | 0 | 商品状态 |
| 12 | create\_time | datetime | 0 | 是 |  | 创建时间 |
| 13 | band\_id | varchar | 50 | 是 |  | 对应品牌id |
| 14 | prod\_type\_code | int | 11 | 是 |  | 对应的类别编号 |
| 15 | is\_new | tinyint | 4 | 是 | 0 | 是否新品 |
| 16 | is\_hot | tinyint | 4 | 是 | 0 | 是否热搜 |
| 17 | comment\_count | int | 11 | 是 | 0 | 评论个数 |
| 18 | sort\_id | varchar | 50 | 是 |  | 对应类别id |

商品信息表，是对商城信息进行管理的信息记录。其中包括标记着由程序生成的唯一商品id，商品的名称，商品详情，种类名称，商品单价，保修期，库存量，销售量，对应的品牌，以及是否为新品和热搜模块的商品状态进行记录。也是商城中最核心的一张表结构。

表4-11 t\_repair---用户维修信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户维修信息表 | | 英文名称 | t\_repair | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | order\_id | varchar | 50 | 否 |  | 订单id |
| 3 | prod\_id | varchar | 50 | 否 |  | 商品id |
| 4 | user\_id | varchar | 50 | 是 |  | 用户id |
| 5 | user\_address | varchar | 255 | 是 |  | 用户地址 |
| 6 | user\_name | varchar | 255 | 是 |  | 用户姓名 |
| 7 | user\_phone | varchar | 11 | 是 |  | 用户手机号码 |
| 8 | what\_problem | varchar | 255 | 是 |  | 故障原因 |
| 9 | status | tinyint | 4 | 是 | 0 | 状态 |
| 10 | create\_time | datetime | 0 | 是 |  | 创建时间 |
| 11 | repair\_id | varchar | 50 | 是 |  | 唯一维修号 |

用户维修信息表为用户按照相应订单发起商品售后服务的统一管理表，除了订单发起的保修操作，普通的要求维修的信息也同样记录在这张表结构中，其中包含主键id，程序生成的唯一维修号，以及管理商户对其更改并反馈到用户的状态。

表4-12 t\_sort---商品分类表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 商品分类表 | | 英文名称 | t\_sort | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | sort\_id | varchar | 50 | 否 |  | 分类id |
| 3 | sort\_name | varchar | 255 | 否 |  | 分类名称 |

商品分类表为商品分类表，包括主键id，sort\_id为程序生成的长度大约为30的唯一id，sort\_name表示分类名，目前系统的分类分成7大类，分别为电视，冰箱，洗衣机，空调，厨房用品，浴室用品，其他7大类，统一管理着电器商品的分类别。

表4-13 t\_smslog---验证码信息表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 验证码信息表 | | 英文名称 | t\_smslog | | |
| 主键 | id | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | id | bigint | 20 | 是 |  | 主键ID |
| 2 | user\_phone | varchar | 50 | 是 |  | 手机号码 |
| 3 | msg\_info | varchar | 255 | 是 |  | 讯息内容 |
| 4 | msg\_code | varchar | 255 | 是 |  | 手机验证码 |
| 5 | status | tinyint | 4 | 是 | 0 | 状态 |
| 6 | create\_time | datetime | 0 | 是 |  | 创建时间 |

验证码信息表记录了每个用户注册或者登录时采用手机验证码验证的方式，对第三方接口调用返回的信息进行记录以便对用户输入的验证码信息进行匹配，查找的方式一般是通过手机号码最近的一条消息记录进行验证码的验证。其中包含主键id，所验证的手机号码，讯息内容，手机验证码，状态，创建的时间等信息。

4.2.2.1 商户后台管理系统的数据库设计

表4-14 SYS\_RESOURCE---资源表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 资源表 | | 英文名称 | SYS\_RESOURCE | | |
| 主键 | RES\_ID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | RES\_ID | bigint | 20 | 是 |  | 主键ID |
| 2 | RES\_PARENT\_ID | bigint | 50 | 是 |  | 父资源ID |
| 3 | RES\_TYPE | int | 1 | 是 | 1 | 资源类型 |
| 4 | RES\_NAME | varchar | 255 | 是 |  | 资源名称 |
| 5 | RES\_ICON | varchar | 50 | 是 | 0 | 资源ICON |
| 6 | RES\_URL | varchar | 350 | 否 |  | 资源URL |
| 7 | RES\_SEQ | int | 4 | 是 | 1 | 同一级别排列顺序 |
| 8 | RES\_STATUS | int | 1 | 是 | 1 | 状态 |

资源表管理网站的资源，包括在操作中所添加的链接的名称，还有菜单的信息，以及菜单的顺序排序等，这是管理整个网站目录的最核心的一个表结构。

表4-15 SYS\_ROLE---角色表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 角色表 | | 英文名称 | SYS\_ROLE | | |
| 主键 | ROLE\_ID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | ROLE\_ID | bigint | 20 | 是 |  | 主键ID |
| 2 | ROLE\_NAME | varchar | 55 | 是 |  | 角色名称 |
| 3 | ROLE\_STATUS | int | 1 | 是 | 1 | 状态 |

角色表为管理网站的角色，本网站按照互联网的机制沿用三种角色，分别为产品经理，产品专员，管理员三个角色进行管理。其中ROLE\_STATUS表示该角色是否要沿用，是否有效。

表4-16 SYS\_ROLE\_RESOURCE---角色-资源关联表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 角色-资源关联表 | | 英文名称 | SYS\_ROLE\_RESOURCE | | |
| 主键 | ROLE\_ID，RES\_ID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | ROLE\_ID | bigint | 20 | 是 |  | 角色ID |
| 2 | RES\_ID | bigint | 20 | 是 |  | 资源ID |

角色-资源关联表为管理网站的角色与其对应所能查看到的资源进行关联，本网站通过超级管理员的角色对整个系统的角色和对应的资源进行分配，分配哪些角色可以查看哪些对应的资源。

表4-17 SYS\_USER\_ROLE---用户-角色关联表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 角色-用户关联表 | | 英文名称 | SYS\_USER\_ROLE | | |
| 主键 | ROLE\_ID，USER\_ID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | ROLE\_ID | bigint | 20 | 是 |  | 角色ID |
| 2 | USER\_ID | bigint | 20 | 是 |  | 用户ID |

角色-用户关联表，是管理用户和其对应角色的指派关联，记录哪些用户属于哪个角色，其中ROLE\_ID为角色ID，USER\_ID为对应的用户ID。

表4-18 SYS\_USER---用户表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 表名 | 用户表 | | 英文名称 | SYS\_USER | | |
| 主键 | USER\_ID | | **外键** | 无 | | |
| 序号 | **名称** | **类型** | **大小** | **必填字段** | **默认值** | **备注** |
| 1 | USER\_ID | bigint | 20 | 是 |  | 角色ID |
| 2 | USER\_NAME | varchar | 20 | 是 |  | 用户名 |
| 3 | PHONE | varchar | 20 | 否 |  | 手机号码 |
| 4 | USER\_PWD | varchar | 255 | 是 |  | 用户密码 |
| 5 | REAL\_NAME | varchar | 255 | 是 |  | 真实姓名 |
| 6 | IS\_SUPER\_ADMIN | varchar | 255 | 是 | 2 | 是否为超级管理员 |

用户表为网站所有能登录参与管理的人员账号密码，其中IS\_SUPER\_ADMIN字段表示其是否为超级管理员，拥有管理全部角色和资源的角色，默认2为不是超级管理员，1为是超级管理员。USER\_PWD记录着密码，密码通过MD5加密，加密的介质为用户名和用户的密码进行联合加密。

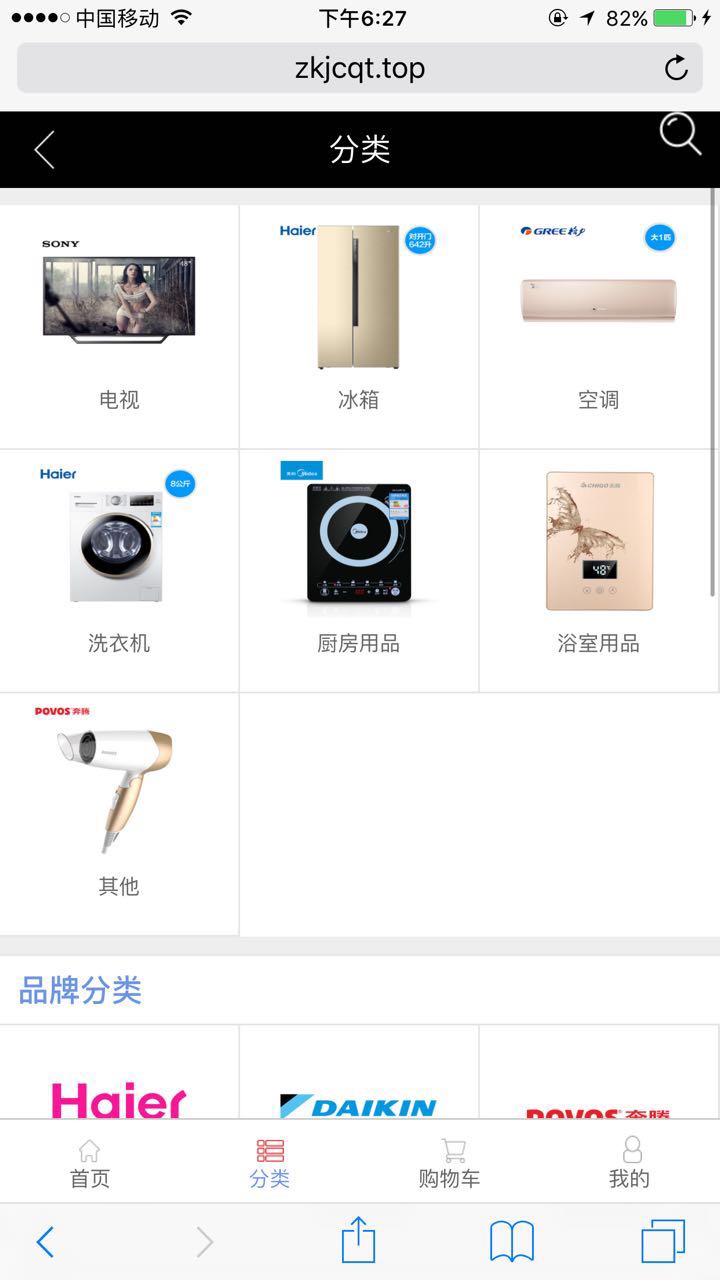
4.4 本章小结

本章主要介绍了数据库系统设计的基础知识，注重介绍了区域性电器商城系统和商户后台数据管理系统所涉及的基本数据表的明细设计以及基本功能详细设计介绍。

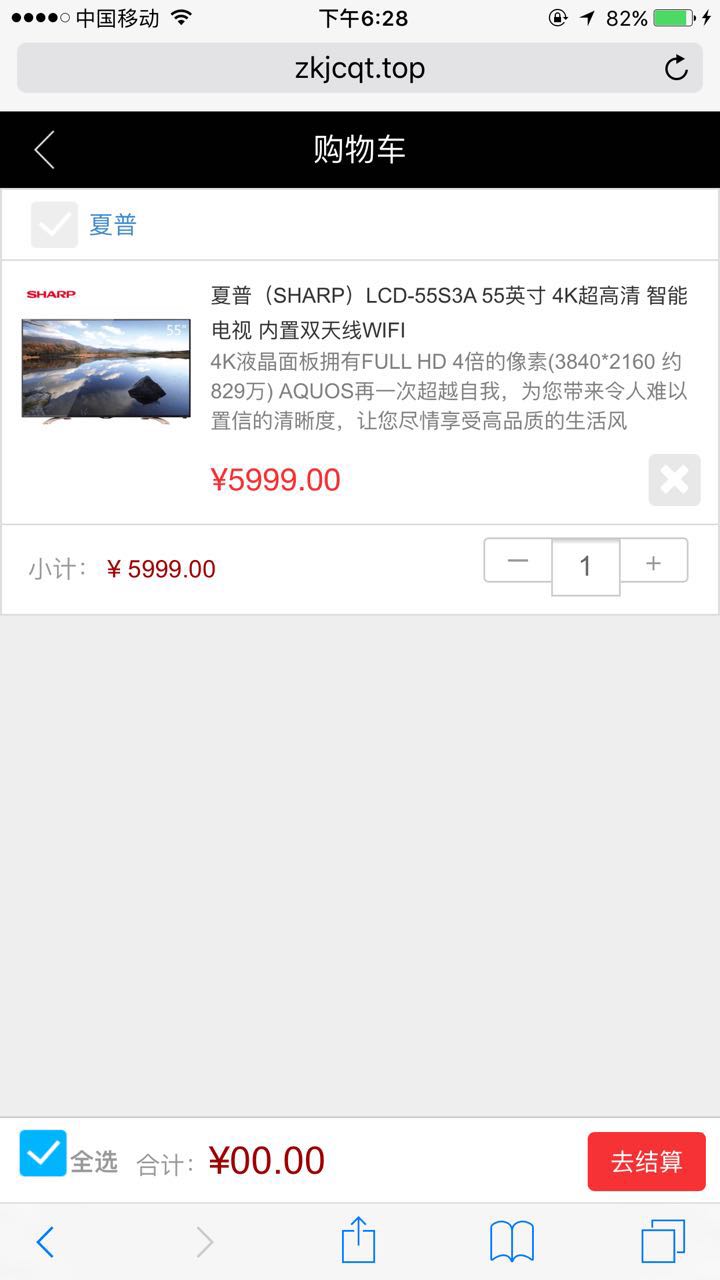
第五章 系统具体功能的设计与实现

5.1 区域性电器商城系统主要模块的设计与实现

系统采用手机端页面的设计，包括热门，新品，专题列表以及品牌列表按钮。分别查看热门商品，新上市的商品，以及活动列表，品牌分类。并且包括最上方的搜索功能，匹配名字模糊搜索，以及类别，品牌的查询。主页分为7个区进行产品的展示，1F电视区，2F 冰箱区，3F空调区，4F洗衣机区，5F厨房用品，6F浴室用品，以及7F其他物品。底部有模拟手机的四个按钮，为首页，分类，购物车，我的页面。引导用户操作。

5-1 主页 5-2 分类

5-3 购物车页面 5-4 我的页面

5.1.1 商品模块

本系统商城通过产品id号查询商品的资料详情，包括商品的名称，颜色，库存量，详情图片，型号，图文详情，以及评价内容。在商品详情页面中，加入了手机端的惯用浏览手法，采用左滑右滑的方式查看商品的图文详情，商品参数以及对应的评价。同时有收藏和加入购物车，立即购买的功能，加入购物车，收藏，立即购买必须在已经登录的情况下进行。否则会被重定向到登录页面。商品的列表，通过缓加载的方式，下拉加载更多，为用户节省不必要的流量开支，并且对商品以销量，评论数，价格，默认情况进行排序。提高用户的购物体验。

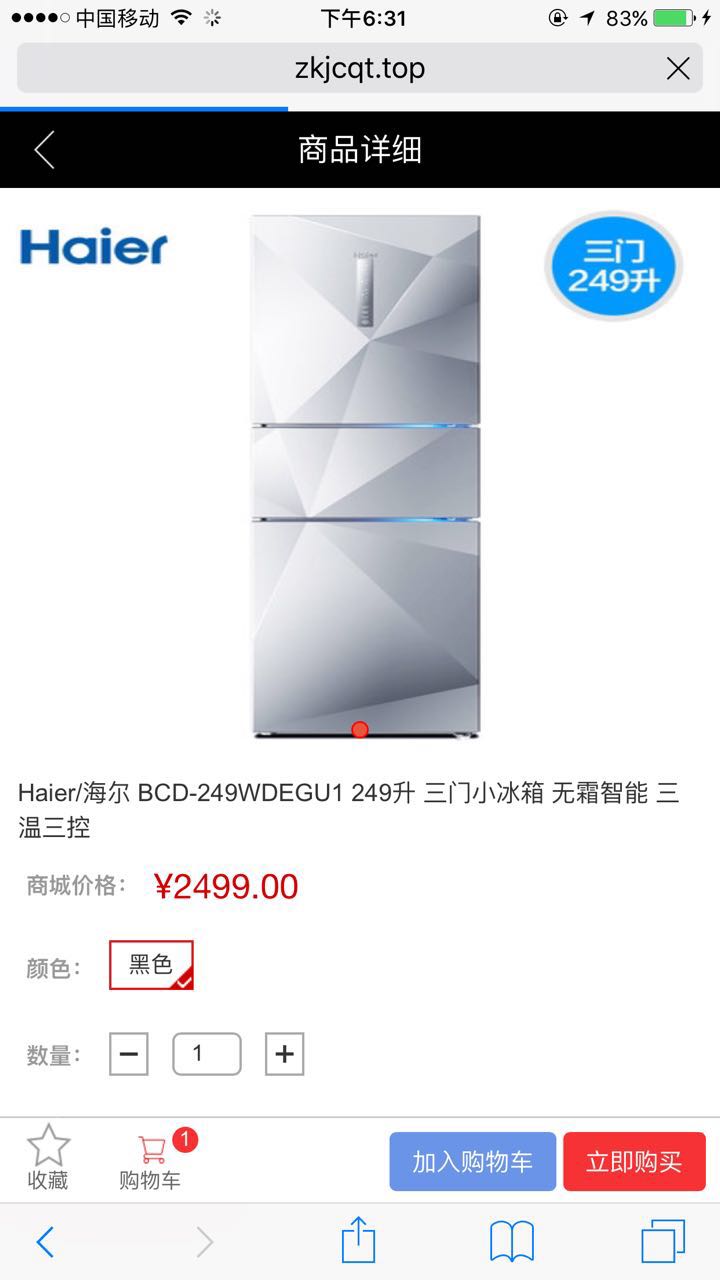
 

图5-5 商品详情页 图5-6 商品其他详情

5.1.2 用户模块

用户模块包含了用户的订单管理，保修单管理，地址管理，收藏夹管理，评论管理，注销功能，这些管理模块都称为用户模块，需要用户在登录之后才能访问到，否则会被重定向到登录页面。

保修单管理包含了保修或者维修单的管理，状态通过图片来判断，倘若后台人员收到了用户的保修或者维修申请，进行处理之后，用户通过标志进行辨认，当为“勾号”时，表示后台人员已经收到。

订单管理包含了用户的全部订单，以及已经完成的订单，正在进行交易的订单等等。

地址管理作用主要是用户管理自己的收货地址。用户在购物时，会被检测是否已经添加了收货地址，倘若没有，则要进行设置，才能进行购物。并且每个号码可以添加无限地址，但是只能有一个默认收货的地址，在地址管理中可以进行更改。

收藏夹管理是指用户在商品浏览时对商品进行收藏的动作，方便之后直接购买。

评论管理是指用户在订单完成之后，等商品的使用等进行评价，评价后会被反应到商品详情中。

注销功能是指用户退出本系统。

图5-7 订单列表 图5-8 维修列表

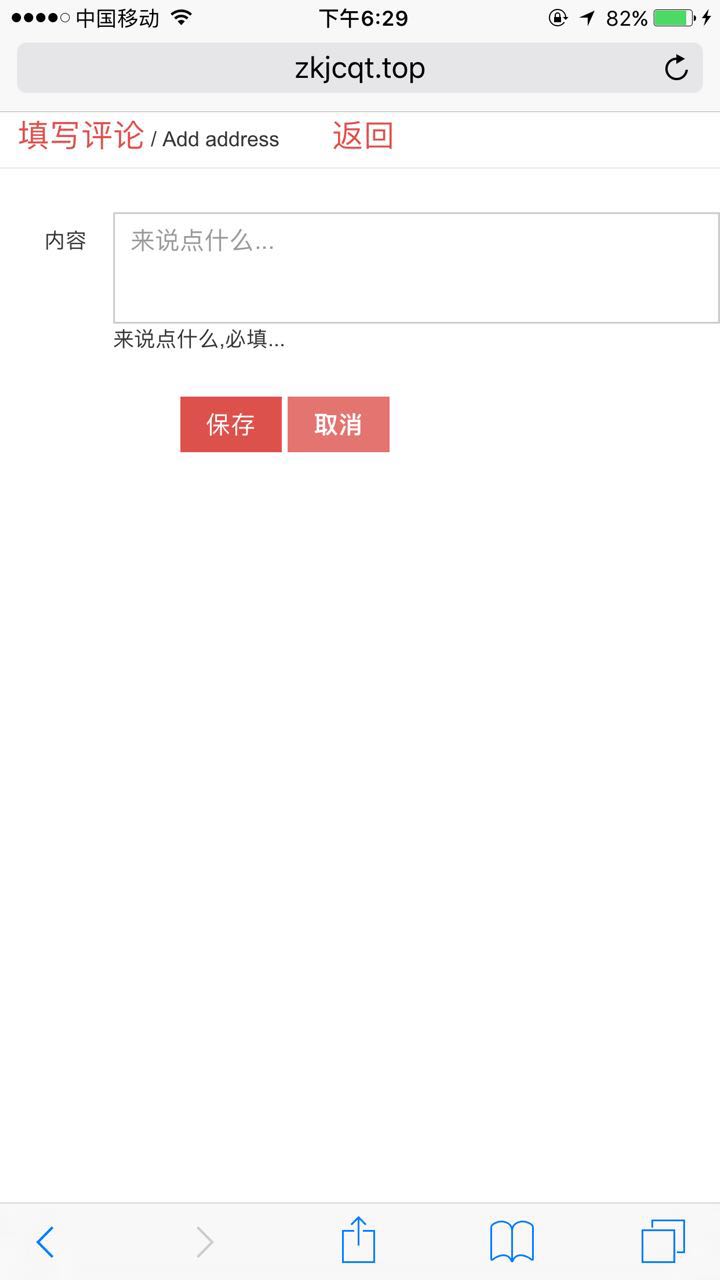
 

图5-9 保修单填写 图5-10 评论填写

5.1.3 短信验证模块

系统采用手机验证码的方式，进行手机号码的注册和登录。确保注册和登录的是有实名手机号码的用户。

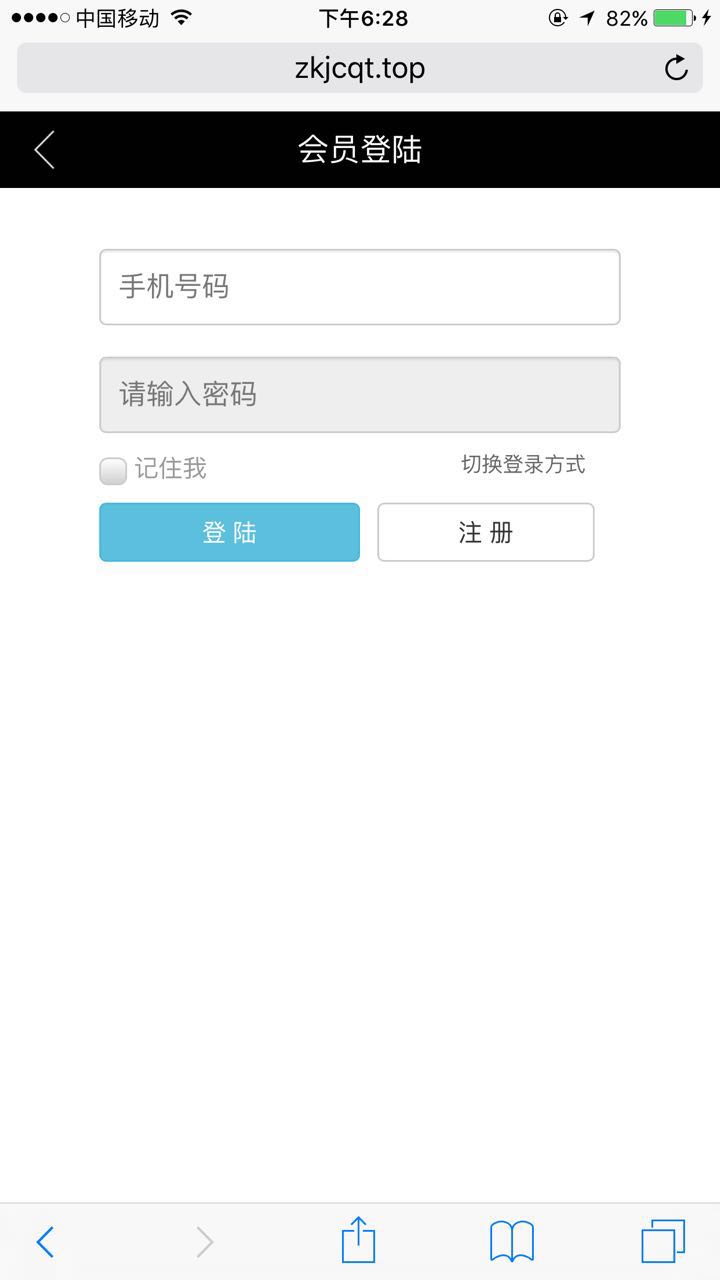
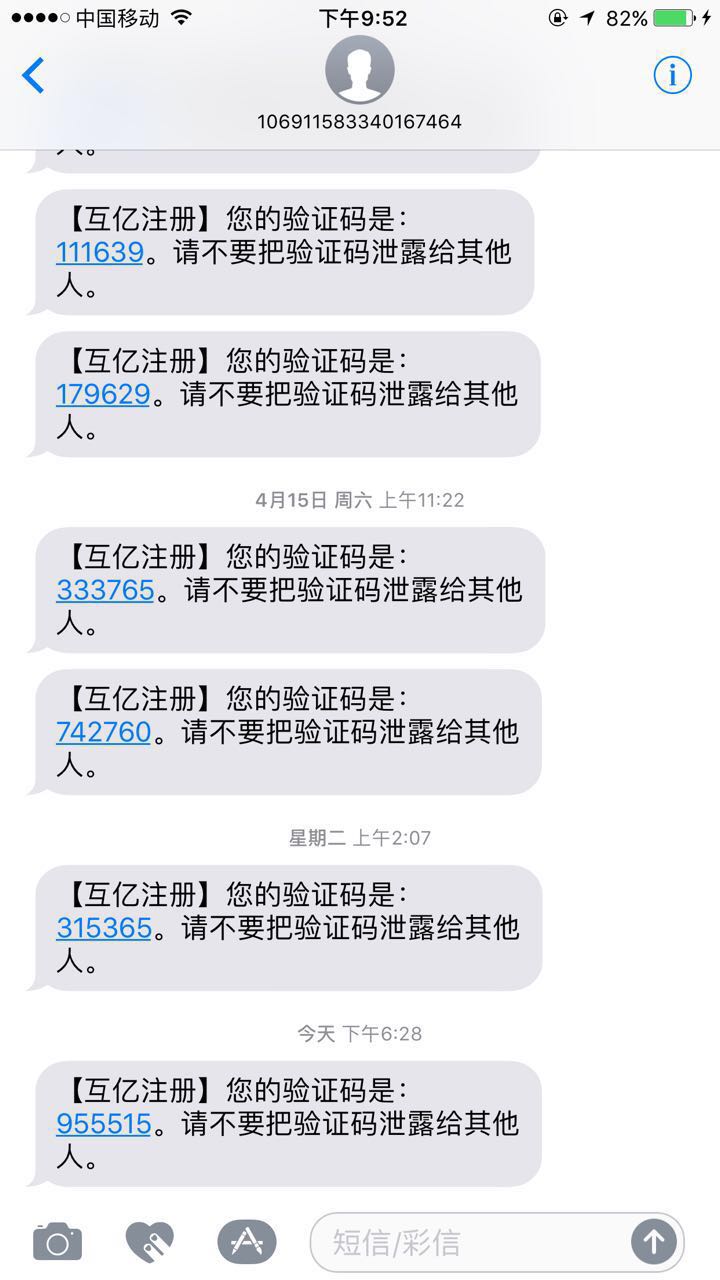
 

图5-11 登录注册 图5-12 验证码

5.1.4 交易模块

本系统采用模拟微信付款，由于第三方支付或者是银联支付需要工商号或者公司备案，因此先做模拟支付。支付成功之后返回一个唯一订单号码，并且在后台系统生成了一个订单，由商户的后台管理人员进行确认。

图5-13 商品价格 图5-14 微信模拟支付



图5-15 支付成功返回订单号

5.2 商户后台管理系统主要模块的设计与实现

5.2.1 商户登录主页模块

系统的登录模块的密码是将用户名和密码进行MD5联合加密存到数据库中，没有提供注册接口。并且登录信息通过cookie进行记录，有效期为7天。有效期过后自行去除。系统对登录的ip进行记录显示。



图5-15 商户后台管理系统登录页面

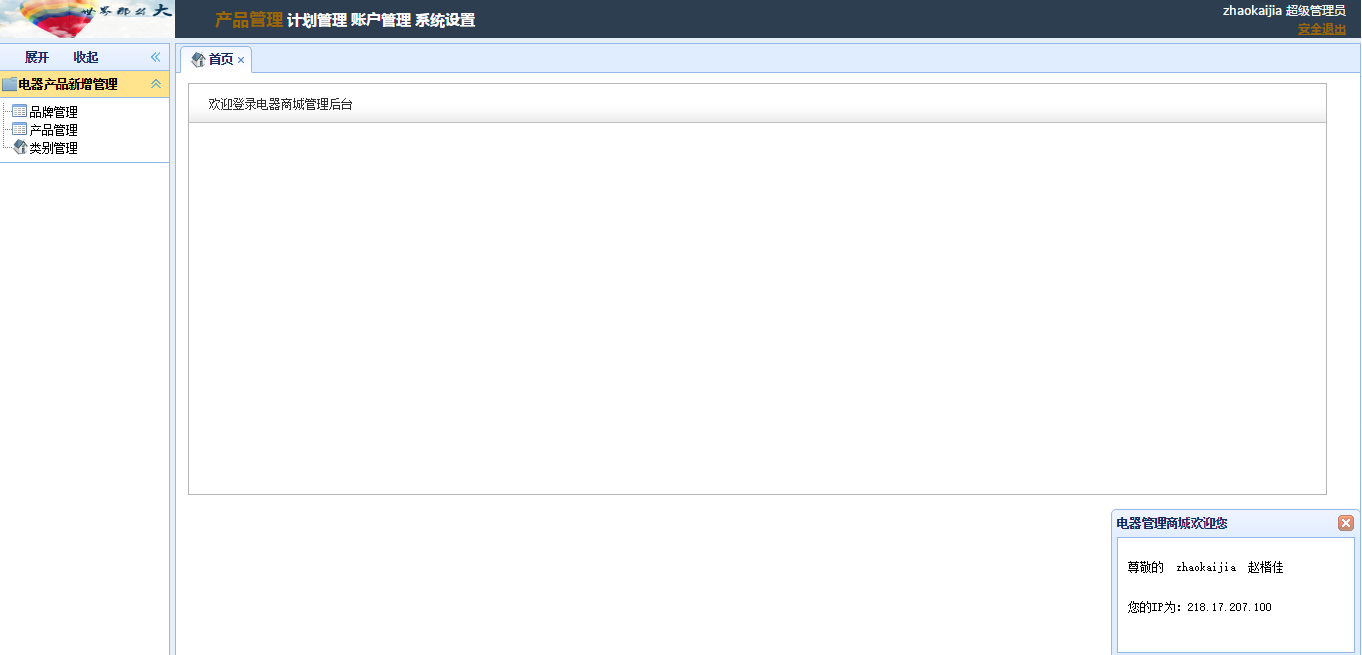


图5-16 主页

5.2.2 产品管理模块

系统的产品管理模块分成三个子模块，分别为品牌管理，产品管理，类别管理模块。

品牌管理模块为商户提供品牌的添加以及条件查询，添加包括了品牌的名称以及品牌的logo，这里添加的属性会反馈到电器购物商城的分类页面。

产品管理模块为商户提供新产品的上线，上新品，上热搜，删除，更新，添加操作。上线的产品会被优先反应到电器购物商城主页面，按照时间排序。上新品的商品会进行新品模块专区，上热搜的商品会到热搜产品区域。商品的添加通过添加商品的名称，商品型号，商品的详情图片等资料进行上传。页面全部采用分页数据，附加条件搜索，加载条。

类别管理是管理商品的分类，目前主要归为7大类，分别为电视，冰箱，洗衣机，空调，厨房用品，浴室用品，其他。

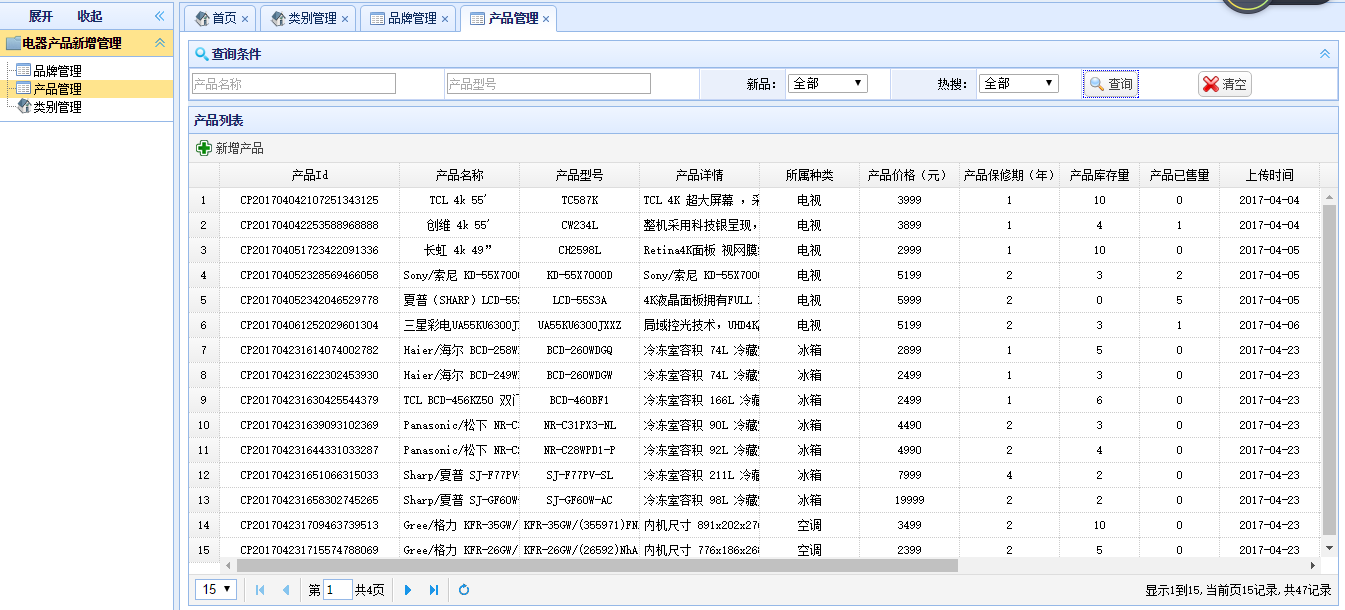


图5-17 产品管理

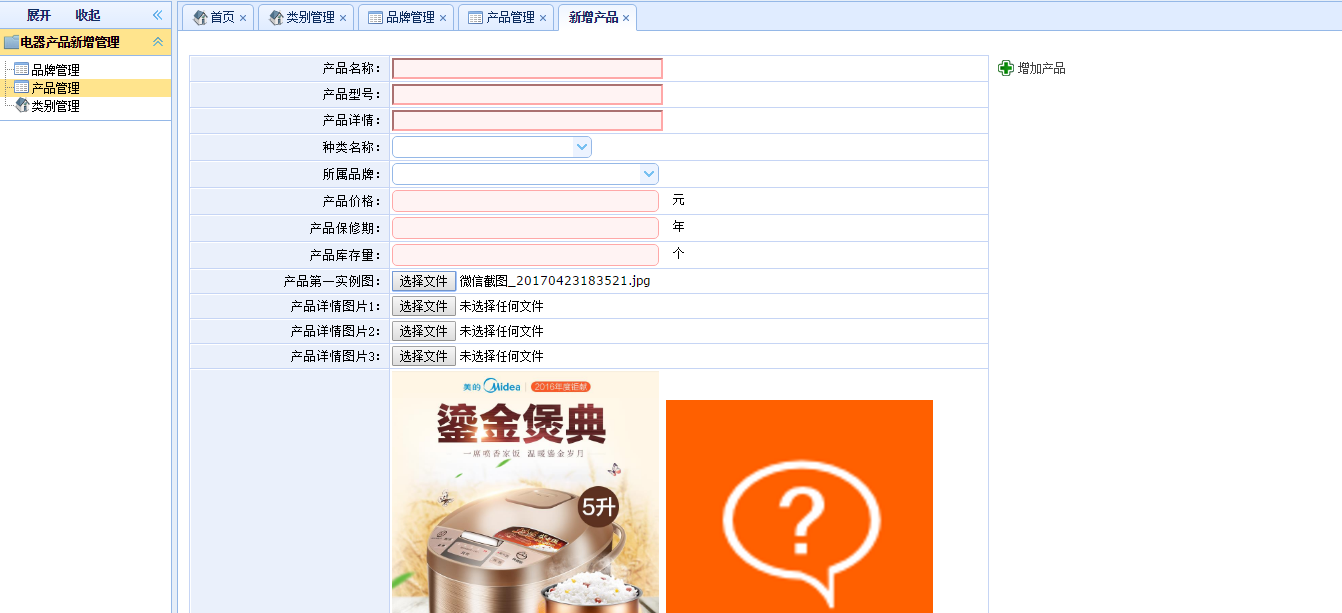


图5-18 产品增加

5.2.3 计划管理模块

本系统的计划管理模块分为订单管理，维修管理模块。订单管理模块主要是各个用户在电器购物商城中所发起的订单，管理员进行维护。包括订单的状态更正，以及能够查询到收获的地址。维修管理模块，管理用户在电器购物商城发起的维修或者保修单，商户在后台进行维护，包括地址的记录以及状态的改变。

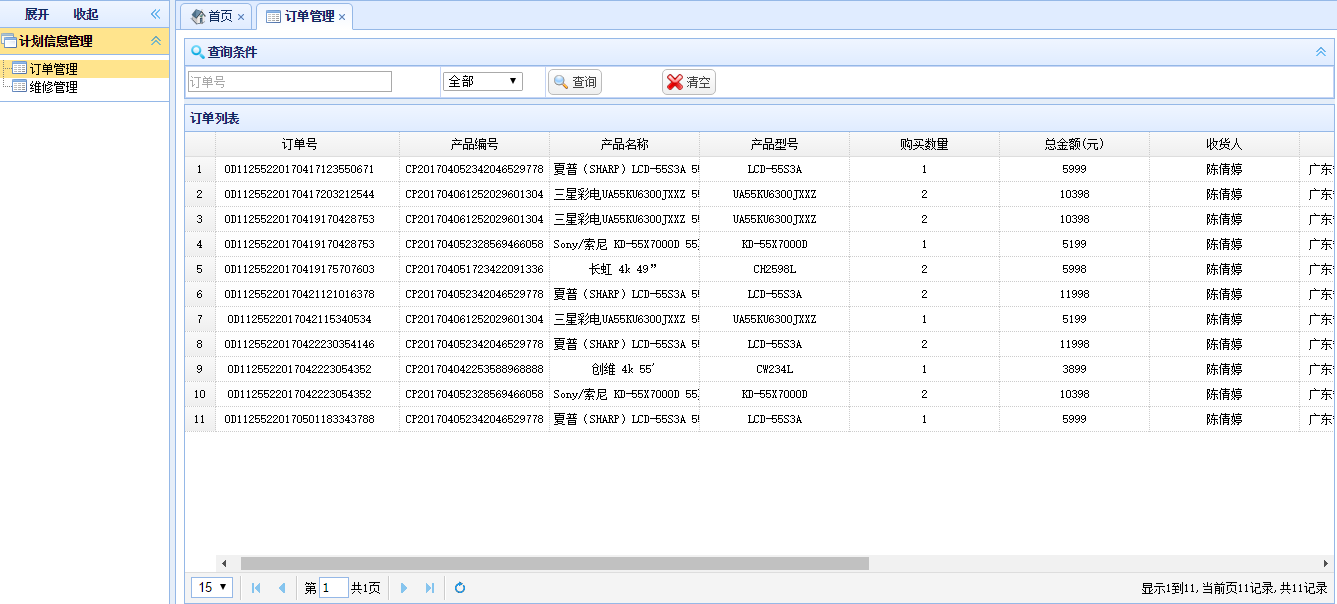


图5-19 计划管理模块

5.2.4 账户管理模块

本系统的账户管理模块分为用户列表以及用户收货管理模块，用户列表模块管理在电器购物商城注册的用户信息全部显示在这里。用户收货管理模块显示着用户设置的收货地址，方便后台管理员维护和查询。

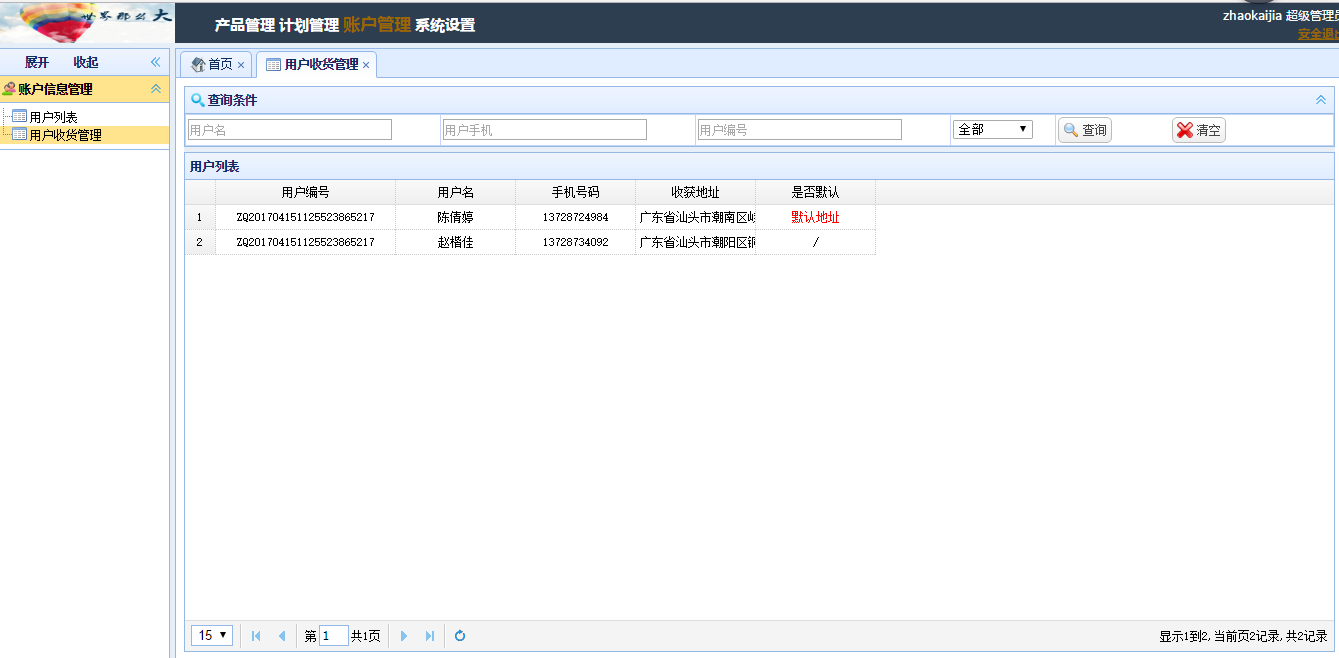


图5-20 账户管理模块

5.2.5 系统设置模块

本系统的系统设置模块包括了用户管理模块，角色管理模块，资源管理模块。用户管理模块是为该系统增加用户信息的模块，角色管理是超级管理员为该系统的角色进行授权以及管理其角色的变动和密码的修改。资源管理模块是指超级管理员对整个系统的资源进行管理。包括资源的添加修改删除等操作。

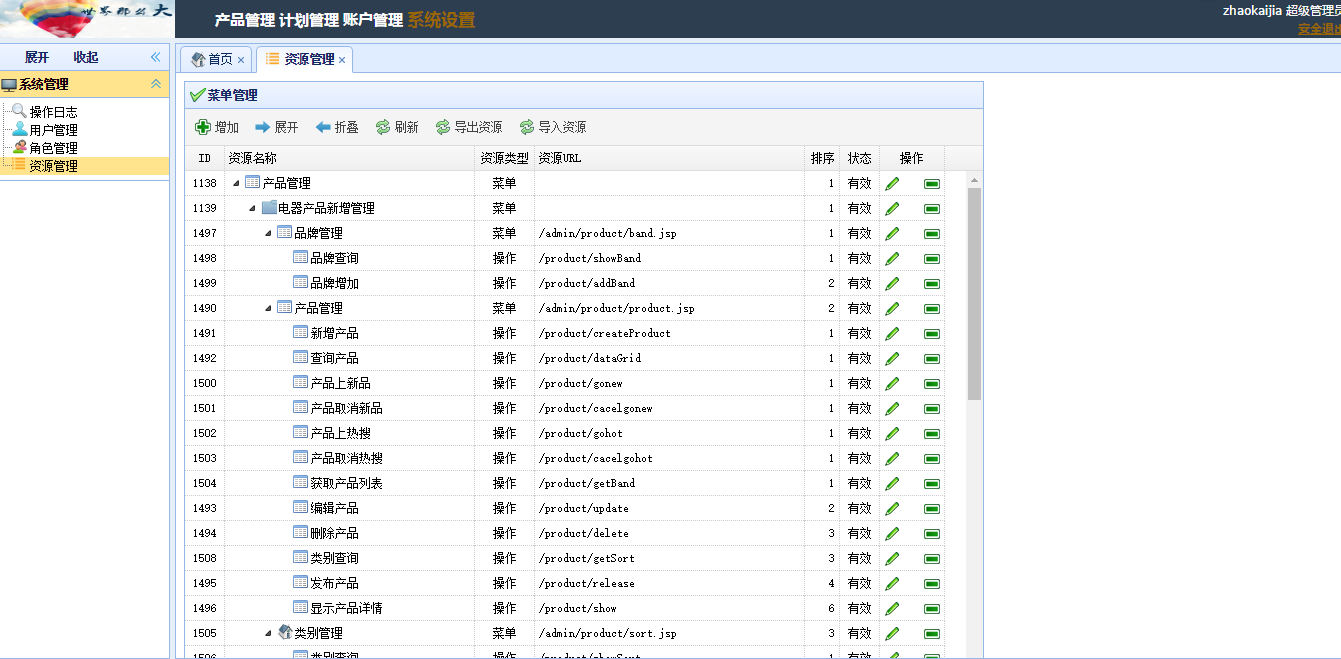


图5-21 系统设置模块

5.3 本章小结

本章主要描述两个内容，一是电器购物系统的模块架构设计，包括其每个模块的截图以及每个模块的详细介绍。二是商户管理后台的模块结构设计以及其每个模块的截图和每个模块的详细介绍。

第六章 系统测试

6.1 组件测试

6.1.1 zookeeper组件的测试

Zookeeper注册中心的正常与否，影响到项目是否能够正常的运行，含括两个系统的api接口是否可以正常被调用，或者数据是否能安全送达或者是更新，查询。Zookeeper测试windows开发侧，开启时如图6-1所示，表示zookeeper运行完成。接着运行服务端程序，将程序的接口通过dubbo框架组件注入在zookeeper注册中心上。如图6-2所示。图中红色表示接口被注册到zookeeper上，该图只显示了一部分接口。

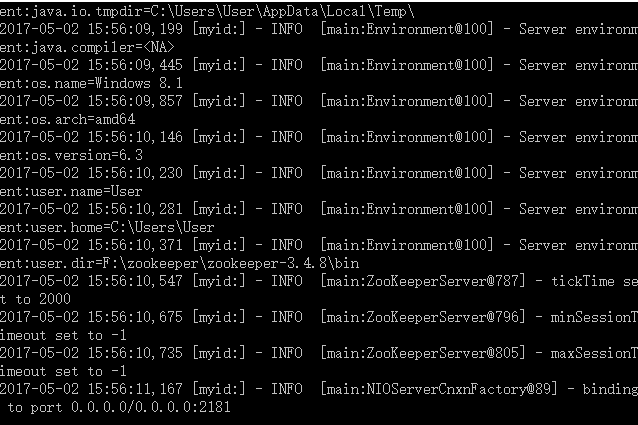


图6-1 zookeeper启动页面

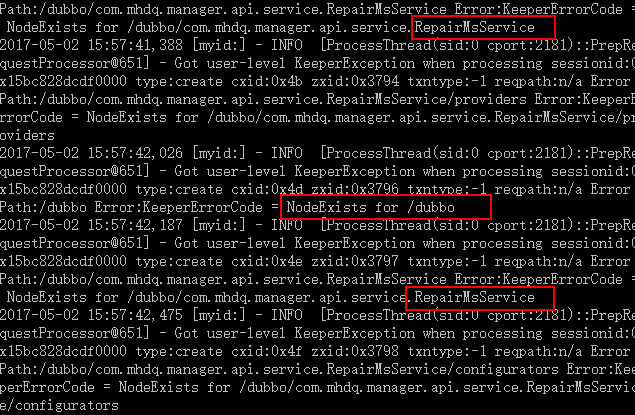
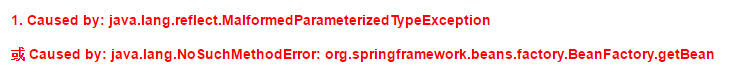
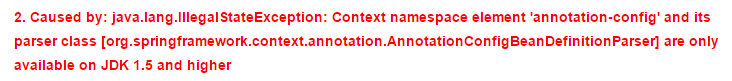


图6-2 zookeeper接口注册

6.1.2 zookeeper常见错误



原因：dubbo启动时，报这个错误，因为dubbo依赖spring 2.5.6.SEC03这个库文件，假设项目中有spring 2.5.6 REALSE版本，则会造成包冲突。则在maven依赖中去掉spring的依赖即可。



错误原因：因为使用了 jdk1.8, 而dubbo 依赖 Spring 2.5 ，而Spring2.5仅仅支持 jdk1.5 到 jdk 1.7, 所以jdk 1.8 不被识别，所以报错。



错误原因：造成这种情况的原因是spring context 初始化两次导致dubbo端口被占用

修改 tomcat 的 server.xml。将autoDeploy="true" deployOnStartup="true" 全部改为false

6.2 性能测试

本系统的性能测试采用WebBench工具，在centos5系统中，对两个系统分别进行测试。

WebBench是压力测试的一个重要工具，由Lionbridge公司团队开发，其测试处在相同的硬件上，不同服务的性能以及不同硬件上同个服务的运行状况。它能够反应每秒钟的相应的请求数量和每秒钟传输的数据量。本测试先以腾讯主页，www.qq.com作为测试对象，如图6-3所示。



图6-2 webbench测试QQ首页

每秒钟响应请求数：1888 pages/min，每秒钟传输数据量8018651 bytes/sec.

请求：1888次成功，0次失败

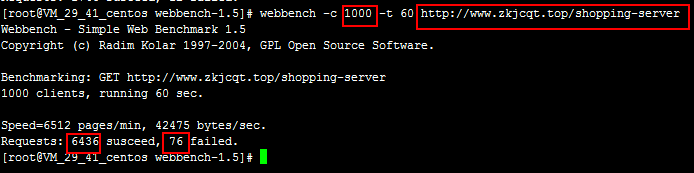
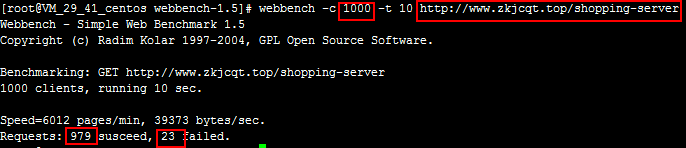


图6-3 电器商城系统测试

测试电器购物商城系统，采用1000个并发，为时60s，得出结论如下，证明系统的并发请求到千级别。根据图6-3所示。每秒钟响应请求数：6512 pages/min，每秒钟传输数据量42475 bytes/sec. 请求：6436次成功，76次失败。



测试后台管理系统，采用1000个并发，为时10s，得出结论如下，证明系统的并发请求到千级别。根据图6-3所示。每秒钟响应请求数：6012 pages/min，每秒钟传输数据量39373 bytes/sec. 请求：979次成功，23次失败。

6.3 本章小结

本章主要对电器商城购物系统以及商户后台管理系统进行测试。分别为zookeeper注册中心的测试，以及对应的性能测试，还有常见错误的描述。首先是对zookeeper注册中心的启动测试，以及项目启动接口注册是否正常进行测试。对在使用zookeeper的时候出现的常用错误进行叙述和总结。另外对这两个系统进行了压力性能测试，通过WebBench工具实现简单并发请求。

第七章 总结和展望

7.1 本论文研究总结

本论文研究开发了两个系统，一个是电器购物商城系统，一个是商户后台管理系统。对于前者要求该应用程序拥有功能完备，易操作，易使用等特点。对于后者，则要求数据的统一性，以及系统的安全性，可扩展性。

电器购物商城系统具有一般商城拥有的基本功能，比如购物，保修等功能。后台管理系统则采用扩展性和管理性较强的方式，分成角色管理，权限管理，资源管理等，对于人员，权限，资源的管理具有针对性较强的作用。

本论文就系统的功能设计，结构设计，数据设计，交互设计，以及相关技术的介绍，测试进行了详细的叙述，展示了系统的每一个细节。

7.2 未来展望

开发这两个系统，主要的目的是方便家里的生意能够走上用户群管理的互联网化。希望系统可以经过不断的完善，能够真正的上线并投入使用。并且真正的具备安全性，能够扛住并发压力，恶意攻击等。系统的安全性还不完备，还需要进一步完善。

参考文献

1. 李子骅. Redis入门指南. 北京：人民邮电出版社, 2013.
2. 陈昊鹏、王浩、姚建平. JAVA核心技术卷1. 北京：机械工业出版社, 2008: 198-325.
3. 陈昊鹏、王浩、姚建平. JAVA核心技术卷2. 北京：机械工业出版社, 2008: 234-458.
4. 陈雄华、林开雄. Spring 3.x 企业应用开发实战. 北京：清华大学出版社, 2012.
5. 拉德. 深入解析Spring MVC与Web Flow. 人民邮电出版社, 2008.
6. 林昊. 分布式Java应用. 电子工业出版社, 2010.
7. 张宴. 实战Nginx:取代Apache的高性能Web服务器. 北京：电子工业出版社, 2010.
8. Chen, shiping, Greenfield, paul.Qos evaluation of JMS:An empirical approach.Proceedings of the 37th Annual Hawaii International Conference on System Sciences,v37,2004,p4385-4394.
9. Bruno, Eric J. Building a lightweight JMS provider. Dr.Dobb’s Journal,v29,n6,December,2005,p958-961.
10. URL: http://www.cnblogs.com/ASPNET2008/p/5622005.html.
11. URL: http://boy00fly.iteye.com/blog/1103586.
12. Jeffrey A. Hoffer，University of Dayton，Joey F.George.Modern Systems Analysis and Design，Third Edition. Pearson Education，2003：66-67.
13. [美]盖尔兹,陈晓燕译.实用 J2EE 应用程序体系结构[M].北京: 清华大学出版社, 2007, (1): 5-26.
14. 胡雪婧. 基于ZooKeeoer的分布式系统的消息发送机制的设计与实现[D]. 吉林大学，2016.
15. 刘庆典，李川. 一个基于Redis架构的分布式图计算系统设计[D]. 四川大学，2016.
16. 蒲晓阳. 基于Nginx和Redis高并发web服务负载均衡的研究[D]. 中国人民银行西安分行，2016.
17. 欧阳昉.网上商城购物系统研究与开发[D].电子科技大学,2007.
18. 李红陛.基于J2EE技术的网上商城系统设计[D].河北科技大学,2010.DOI:10.7666/d.y1933629.
19. 吴康.基于J2EE的新型分布式网络商店的设计与实现[D].电子科技大学,2010.

致谢

这几个月以来，我一直在努力完成毕业设计以及论文的攒写，期间遇到了很多难以解决的错误，并且也请求过多人的帮助

首先要衷心感谢我的毕业设计导师------张席老师，对张席老师的了解，是从上他的java设计课程开始的，觉得老师是一个讲课认真，知识针对性很强，教学经验丰富的老师。在毕业设计期间，老师给我的论题提供了非常多的建议，在我对项目进行开发的过程中，遇到的问题都能请教他，并会及时的回复。老师主张的是，遇到问题，直接见面聊解决，不要只是通过电话。正是因为老师的悉心教导以及负责，我才能够顺利的完成毕业设计的开发，以及论文攒写，并顺利完成学业，使我的业务开发能力有了大幅度的提升。

同时，也要感谢其他教导过我的老师，以及每一位帮助过我的软件工程系的同学，在他们的帮助下解决了许多困难的问题。

最后，还要感谢我的爸妈，有他们在背后默默的支持以及关心，让我能够有信心去面对未来发生的一切困难。

Throughout the traditional electricity business system, as a large object of the electrical industry, there have been similar to the United States, Suning and other large group companies. But the hometown area, the demand for online shopping did not imagine the big city as big as that. Because the electrical industry in this sales industry, one of the most difficult to solve or after-sales service, and the transport process of quality problems.

The overall demand task developed by the system is to provide a highly customized and Internet-based browser or mobile interface for rural users. Guide users to go directly to the Internet to browse and buy, and solve the problem of warranty distance. System shopping web side, simple style, support the computer, mobile phone side WeChat to buy. Background management system, through the management of authority, and resource management, product, user orders, and warranty maintenance and management. Finally, through the three different tomcat and JNDI data source management of the two systems and the same database together.

Home appliances for the home of the status quo, to facilitate the home better management orders and maintenance orders, such as the development of such a set to buy, repair, and management as one of the two web applications. Using zookeeper registry and RPC remote call to achieve, to meet the needs of this phenomenon.