湖南科技大学计算机科学与工程学院

WEB 编程课程设计报告

题目：_____东东购物网站_____

学　　号：__1505040117__
姓　　名：__李东_____
指导老师：___张世文_____
完成时间：__2018.6.21__

目录

一、需求分析

1.1、网站的功能

电子商务网站的 功能主要包括用户的注册、登录、购物和新闻发布这几个主要方面，还包括所售商品的维护。

1.1.1、用户登录、注册功能

（1）注册功能。

这是用于第一次进入网站，并有兴趣在本网站购买东西的顾客而设的，具有提醒第一次进入本站的用户注册的功能。另外用户注册页面应有一个介绍和解释本站详细功能的说明，并附上同意与否的按钮和一个用于提示用户填写详细资料的表，并划分出是否为必填或其他。同时还需要有基本的判断功能，对用户填写的资料的正确与否进行判断，然后返回相应的信息，还能够把用户提供的信息返回，建立一个数据库，并把这些信息写入数据库。

（2）登录功能。

这是为老顾客而设的，其目的是为顾客创造一个友好的环境，并且让他知道自己上次离站的时间等信息，是一个简单用于登录、判断用户填写正确与否的程序。如果登录成功，则再把这次信息写入数据库，重新更新数据库信息，并启动其他线程以便于用户购物。

（3）查看、修改用户信息功能。

当用户对自己注册的信息不满意时，系统必须有对此项的支持。

（4）修改密码。

这是出于对安全型的考虑，经常地更换密码可以提高安全性。

(5) 管理收货地址功能

这是出于对购物方便方面的考虑，储存了多个地址以后方便了下订单的时候选择正确的收货人和收货地址。

1.2、购物功能

（1）物品浏览：

当顾客进入网站，首先应当看到推荐商品或最近更新的商品，同时对商品的类型进行适当的划分。对商品还要有详细的介绍，对大类中的小类也应当把它们的属性特点向顾客介绍清楚。

（2）商品购买：

当顾客看中某样商品时，就应当启动购买功能，在小类型下所属的商品列表中应当有购买功能，购买完所有商品后，系统应当给出购物清单，在清单中应列出所购物品及数量，没见物品的价格和购物总价格，还应当有完整的确认系统，不仅包括放入购物车的确认，而且还包括购物与否的确认。

（3）动态新闻发布功能：

动态新闻发布功能和商品最近更新的发布是类似的，它们构建数据库的形式和程序的实现大致相同。

（4）商品热卖推荐功能

商品热卖推荐功能是根据页面的点击次数来显示的，点击次数多就会上。

（5）商品搜索功能

商品种类和数量多了以后有商品名字搜索的功能会对购物体验造成良好的影响

1.3、管理员管理功能

（1）管理商品功能

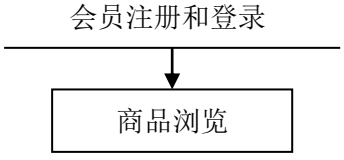
添加、修改，编辑商品内容

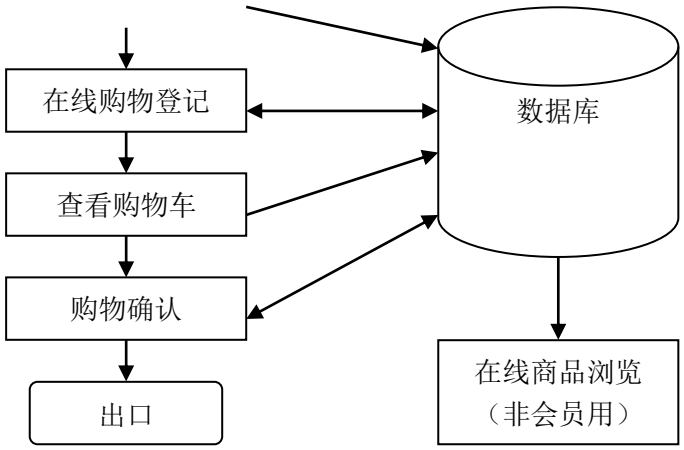
（2）修改订单状态功能。

能够将订单处理状态改变，比如改为发货。

二、网站的结构设计

该电子商务网站主要包括了首页，用户中心，购物中心，新闻中心几个模块，它们是互相联系的，其中新闻中心的建立与购物中心的建立是类似的。对于整个网站，顾客的购物流程可以简单地表示如下：





1、首页

首页上应当能实现以下功能：有各功能的链接，要包括必要的内容，没必要把内容都包含在首页中，而且网站的整体风格要一致。首页里包括了购物中心，新闻中心的一些内容。

2、用户中心

建立一个好的、功能全面的用户中心不仅能方便管理员对顾客的管理，而且还能为用户提供友好的界面，以吸引更多的顾客。

在用户中心应能实现以下功能：

（1）页面能单独用来链接到注册登录功能，并且，将集中把信息返回到这里。因此，只要写入一些简单的信息，用于提示用户登录注册，并且做链接到登录注册页面的超链接。在写返回信息时，也只用简单的话提示用户注册或登录成功或提示哪项填写不正确。

（2）要有让用户修改注册信息和密码的页面，修改注册信息的页面设计和注册使用到的页面相同，密码的修改则应另建一个页面。

3、购物中心

购物中心包括三个方面：商场（商品浏览）区域，商品热卖区域和购物确认区域。

（1）在商场可以浏览所有商品的属性及详细资料；商品的浏览设有对大类的链接，也设置

了小类的链接和十大热门商品的排列，这样可以让用户对自己所在区域及商品出售状况一目了然，同时还设置对各商品的详细说明画面。

（2）商品热卖区有对商品的卖出情况的排名，让用户对商品的售出情况有一个大致的认识，具有导购的作用。

（3）购买确认区在首页中可通过“查看购物车”的链接到达该区，通过设置表单和用户最终确认功能来实现。

4、新闻中心

与购物中心很类似，是发布最新信息（包括企业新闻、社会新闻、国内新闻和国际新闻）的场所。

5、后台管理功能

主要提供对网站所出售商品的管理功能，包括对商品的新增、修改和删除，订单状态的修改。

三、数据库概要设计与详细设计

数据库采用了 Mysql5.7，数据库名是 Dshop，数据库的格式为 UTF8，放于远程服务器并对同学数据库没装好的同学提供服务。

3.1、数据库概要设计

根据需求，我创建了 7 个表。其中包括了
address 表，用来保存用户名地址信息。
category 表，用来保存商品的种类。
orders 表，用来保存商品的订单数据
item 表，orders 的级联表，用来保存一个订单多个商品的保存情况。
member 表，用来处理用户的信息，管理员权限设置是 member 表的一个属性
news 表，用来储存新闻信息
product 表，用来储存商品信息

3.2、数据库详细设计

3.2.1 商品类目表

表名：category

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
name	varchar(32)		类目名称
alias	varchar(32)		类目别名（英文或拼音）
p_id	int	外键，引用本表主键	父类目编号
order_weight	Int		排序权重值：默认为10

3.2.2.会员信息表

表名：member

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
mobile	varchar(32)		会员手机号
openid	varchar(32)		微信用户唯一标识
pwd	varchar(32)		密码

real_name	varchar(32)		真实名
nick_name	varchar(32)		昵称
email	varchar(128)		邮箱号
gender	tinyint		性别
register_time	datetime		注册时间

3.2.3.新闻表

表名：news

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
title	varchar(32)		标题
thumbnail	varchar(32)		主配图片
content	longtext		内容
hits	int(11)		点击数
top	tinyint(1)		是否轮播图，默认值false
pub_time	datetime		发布时间

3.2.4.订单表

表名：orders

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
number	varchar(64)		订单号
buyer_id	int(11)		所属买家会员编号
total_amount	decimal(9,2)		商品总数量
total_price	decimal(9,2)		总的费用
payment_price	decimal(9,2)		实付的费用
remark	varchar(255)		买家留言
contactr	varchar(32)		收货人
mobile	varchar(32)		联系电话
street	varchar(255)		详细地址
zipcode	varchar(32)		邮编
create_time	datetime		下单时间
payment_time	datetime		支付时间
delivery_time	datetime		发货时间

end_time	datetime		完成时间
status	int(11)		订单状态

3.2.5.商品表

表名：product

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
name	varchar(255)		商品名称
cate_id	int(11)		所属类目编号
thumbnail	varchar(128)		主配图片
inventory	int(11)		库存数量
sales_volume	int(11)		售出数量
price	decimal(9,2)		定价
sale_price	decimal(9,2)		售价
detail_description	mediumtext		详情富文本描述
selling_description	varchar(255)		卖点富文本描述
create_time	datetime		添加时间
sale_time	datetime		开售时间

3.2.6.订单级联表

表名：item

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
order_id	int(11)		所属订单编号
product_id	int(11)		商品编号
amount	int(11)		单品数量
total_price	decimal(9,2)		单品总价
payment_price	decimal(9,2)		单品实付的总价

3.2.7.地址表

表名：address

字段名	类型	约束	说明
id	int	主键	编号，自动增长列
contact	varchar(32)		收货人
mobile	varchar(32)		联系电话
street	varchar(32)		详细地址
zipcode	varchar(32)		邮编
mbr_id	int(11)		所属会员
default_value	tinyint(1)		是否为所属会员的默认收货地址

四、详细设计

4.1、实体类设计

实体类一共有 8 个，其中 7 个是和数据表一样的实体类，另外是一个分页器的实体类。实体类的编写基本如下：

Item 实体类

```
package entity;

import java.io.Serializable;
import java.math.BigDecimal;

/**
 * 订单项
 * @author morzlee
 */
public class Item implements Serializable {
        private static final long serialVersionUID = -5513015465167003750L;
        private Integer id;
        private Integer order_id;/* '所属订单编号', */
        ……
        private BigDecimal payment_price;/* '单品实付的总价' */
        private Product product;/* '所购商品' */
        public Integer getId() {
                return id;
```

```
        }
        public void setId(Integer id) {
                this.id = id;
        }
        ……
        public void setProduct(Product product) {
                this.product = product;
        }
        @Override
        public String toString() {
                return "Item [id=" + id + ", order_id=" + order_id + ",
product_id="+ product_id + ", amount=" + amount + ", total_price="+
total_price + ", payment_price=" + payment_price+ ", product=" + product +
"]";
        }
}
```

4.2、service 类

service 类一般是对数据库操控的操作类，我实现了一些对数据库操作的一些接口，我只要在 servlet 里创立对应的 service 类然后调用这些接口就可以对数据库进行操作，其中这些操作都是基于实体 entity 类对数据库进行操作，因为这些 service 写法大多类似，故只写一个，会在写明所有的 service 具有的接口方法。

Product 实体类

```
package service;

import ……


/**
 * 商品的业务逻辑类
 * @author morzlee
 */
public class ProductService {
        private QueryRunner qr = new QueryRunner();
        private ScalarHandler<Long> scalarHandler = new
ScalarHandler<Long>();
        private BeanHandler<Product> beanHandler = new
BeanHandler<Product>(Product.class);
        private BeanListHandler<Product> beanListHandler = new
BeanListHandler<Product>(Product.class);
```

```
        public Product save(Product product) throws RuntimeException{
                String sql = "INSERT INTO product(name, cate_id,
thumbnail,inventory,"+"sales_volume,price,sale_price,detail_description,
"+"selling_description,create_time,sale_time)"+"VALUES(?,?,?,?,?,?,?,?,?
,?,?)";

                Object[] params = {product.getName(),
product.getCate_id(),product.getThumbnail(), product.getInventory(),
product.getSales_volume(), product.getPrice(),product.getSale_price(),
product.getDetail_description(),product.getSelling_description(),
product.getCreate_time(),product.getSale_time()};
                Connection conn = null;
                try{
                        conn = DbHelper.getConn();
                        conn.setAutoCommit(false);
                        Long id = qr.insert(conn, sql, scalarHandler,
params);
                        product.setId(id.intValue());
                        DbUtils.commitAndCloseQuietly(conn);
                }catch(Exception e){
                         DbUtils.rollbackAndCloseQuietly(conn);
                         throw new RuntimeException(e);
                }
                return product;
        }
        public Page<Product> findAll(int number, int size) throws
RuntimeException{
                Page<Product> page = new Page<>(number, size);

                String sql = "SELECT COUNT(id) FROM product ";
                String sql2 = "SELECT * FROM product ORDER BY id DESC
LIMIT ?,?";

                Connection conn = null;
                try{
                        conn = DbHelper.getConn();

                        Long temp = qr.query(conn, sql, scalarHandler);
                        if(temp != null && temp.longValue() > 0){
                                page.setTotalElements(temp.longValue());

                                Object[] params = {Integer.valueOf((number
- 1) * size),
                                                Integer.valueOf(size)};
```

```
                                List<Product> list = qr.query(conn, sql2,
beanListHandler, params);
                                page.setItems(list);
                        }
                }catch(Exception e){
                        throw new RuntimeException(e);
                }finally{
                        DbUtils.closeQuietly(conn);
                }
                return page;
        }
```

剩余代码太多，剩余的方法就不贴出来了

4.3、web 类

4.3.1、controll 类

其中分别有 member 权限才有的页面控制 servlet 还有不登陆所有人都可用的页面 servlet。列出 2 个 servlet，其中一个是共有页面，另外一个是 member 页面的 Addressaddservlet：

```
package web.controller.member;

import entity.Address;
import entity.Member;
import service.AddressService;

import ……;

/**
 * 会员的新增地址
 */
@WebServlet("/member/address/add")
public class AddressAddServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                //获取请求的参数数据
                String contact = request.getParameter("contact");
                String mobile = request.getParameter("mobile");
```

```
                String street = request.getParameter("street");
                String zipcode = request.getParameter("zipcode");
      String default_value=request.getParameter("default_value");
                //把散装数据封装成对象
                Address address = new Address();
                address.setContact(contact);
                address.setMobile(mobile);
                address.setStreet(street);
                address.setZipcode(zipcode);
                boolean dv = Boolean.valueOf(default_value);
                address.setDefault_value(dv);

                //设置这个要新增的地址所属的会员 ID
                Member mbr =
(Member)request.getSession().getAttribute("curr_mbr");
                address.setMbr_id(mbr.getId());
                //业务逻辑处理
                AddressService service = new AddressService();
                service.save(address); //保存新增的地址,address 的 ID 就有值
                //处理默认地址的问题
                if(dv){
                service.updateDefault(mbr.getId(), address.getId());
                }
                //跳转(重定向)
                //理解：Servlet 中的两种跳转方式：重定向、请求分派
                response.sendRedirect(request.getContextPath() +
"/member/address/list");

        }
        protected void doPost(HttpServletRequest request,
HttpServletResponse response)
                        throws ServletException, IOException {
                doGet(request, response);
        }

}
```

Addtocardservlet:

```
package web.controller;

import entity.Product;
import ……;

/**
```

```
 * 用于处理AJAX提交“添加商品到购物车”的请求的Servlet
 */
@WebServlet("/addToCart")
public class AddToCartServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private ProductService service = new ProductService();
	@SuppressWarnings("unchecked")
	protected void doGet(HttpServletRequest request,
HttpServletResponse response)
			throws ServletException, IOException {
		String idStr = request.getParameter("id");
		String numStr = request.getParameter("num");
		Integer id = Integer.valueOf(idStr);
		Integer num = Integer.valueOf(numStr);
		Product prod = service.findOne(id);
		//购物车是存放在session中的
		HttpSession session = request.getSession();
		Map<Product, Integer> cart = (Map<Product,
Integer>)session.getAttribute("cart");
		if(cart == null){
			cart = new ConcurrentHashMap<Product, Integer>();
			session.setAttribute("cart", cart);
		}
		//购物车,用商品对象作为key，用数量给value存放到Map
		//先把购物车中获取是否已经存在当前要购买的商品
		Integer oldNum = cart.get(prod);
		if(oldNum != null){
			cart.put(prod, Integer.valueOf(oldNum.intValue()
+ num.intValue()));
		}else{
			cart.put(prod, num);
		}
		//step3: 响应文本
		response.setContentType("text/html;charset=UTF-8");
		PrintWriter out = response.getWriter();
		out.print("ok");
		out.flush();
	}
	protected void doPost(HttpServletRequest request,
HttpServletResponse response)
			throws ServletException, IOException {
		// TODO Auto-generated method stub
		doGet(request, response);
	}
```

```
}
```

4.3.2、filter 类

Filter 类用来处理判断用户是否有登陆，如果有登陆就有权限进入用户中心购买结账的页面，没有就跳到登陆和注册界面。

SecurityFilter：

```
package web.filter;
import ……;
/**
 * 对会员的操作进行安全验证的过滤器
 * @author Administrator
 */
@WebFilter("/member/*")
public class SecurityFilter implements Filter {
        @Override
        public void destroy() {
        }
        @Override
        public void doFilter(ServletRequest req, ServletResponse res,
FilterChain chain)
                        throws IOException, ServletException {

                HttpServletRequest request = (HttpServletRequest)req;
                HttpServletResponse response = (HttpServletResponse)res;
                HttpSession session = request.getSession();

                Member mbr = (Member)session.getAttribute("curr_mbr");
                if(mbr != null){ //登录后的会员发起的请求，通过
                        chain.doFilter(request, res);
                }else{ //未登录的会员发起的请求，要阻止
                        session.setAttribute("msg", "会员的相关操作，需要登
录！");
                        response.sendRedirect(request.getContextPath() +
"/member_login.jsp");
                }
        }
        @Override
        public void init(FilterConfig arg0) throws ServletException {
        }
```

```
}
```

4.3.3、listener 类

```
package web.listener;
import ……;
/**
 * 自定义的应用上下文监听器<br/>
* @author morzlee
 */
@WebListener
public class MyServletContextListener implements ServletContextListener {
        @Override
        public void contextDestroyed(ServletContextEvent arg0) {
        }
        //本方法会在 Web 程序部署成功后，立即执行。一般是用来对本程序中的数据做一些初始化工作
        @Override
        public void contextInitialized(ServletContextEvent event) {
                //放置 web 应用上下文根路径
                ServletContext context = event.getServletContext();
                String ctx = context.getContextPath(); //JSP 页面的链接的基准路径

                //往 ServletContext 作用域里存放一个名为 ctx 的变量。
                context.setAttribute("ctx", ctx);

                //JSP 页面中要显示的后台动态上传的图片的基准路径
                context.setAttribute("pic_base", ctx + "/img/");

                //加载所有的类目列表数据
                CategoryService service = new CategoryService();
                List<Category> list = service.findAll();
                context.setAttribute("cates", list);
        }
}
```

4.3.4、tag 类

自定义标签类，用来分页

4.4、Service 接口设计

4.4.1、AddressService 接口设计

接口名字	接口备注
save(Address address)	地址保存
update(Address address)	地址更新
delete(Integer id)	地址删除
findByMember(Integer mbr_id)	获取指定会员的所有地址
findOne(Integer id)	获取指定编号的地址

4.4.2、CategorySetvice 接口设计

接口名字	接口备注
save(Category cate)	类型保存
update(Category cate)	类型更新
delete(Integer id)	类型删除
findOne(Integer id)	寻找类型
findAll()	列出所有类型

4.4.3、MemberService 接口设计

接口名称	接口备注
Member save(Member member)	新增会员
void delete(Integer id)	删除会员
void update(Member member)	更新会员
Member findOne(Integer id)	根据 ID 获取该会员对象
Member findByMobile(String mobile)	根据手机号获取该会员对象
List<Member> findAll()	获取所有会员的列表
long count()	统计会员总数量
Page<Member> findByPager(int number, int size)	根据页数查找

4.4.4、NewsService 接口设计

接口名称	接口备注
News save(News news)	新增新闻
void update(News news	更新新闻
void delete(Integer id)	删除新闻

News findOne(Integer id)	根据 ID 获取该新闻对象
void count()	获取新闻数量
Page<News> findTopByPublic(int number, int size)	获取已经发布的轮播新闻列表
Page<News> findByPublic	获取已经发布的新闻列表
List<News> findAll()	获取所有新闻对象

4.4.5、OrdersService 接口设计

接口名称	接口备注
Orders save(Orders orders)	新增订单
void update(Orders news)	更新订单
void delete(Integer id)	删除订单
Orders findOne(Integer id)	获取指定编号的订单
Page<Orders> findByBuyer(Integer buyer_id, int number, int size)	获取指定会员的订单分页列表
Page<Orders> findAccumulating(int number, int size)	获取进行中的订单分页列表
void updateStatus(Integer id, int status)	更新订单的状态
Page<Orders> findByStatus(int status, int number, int size)	获取指定状态的订单分页列表

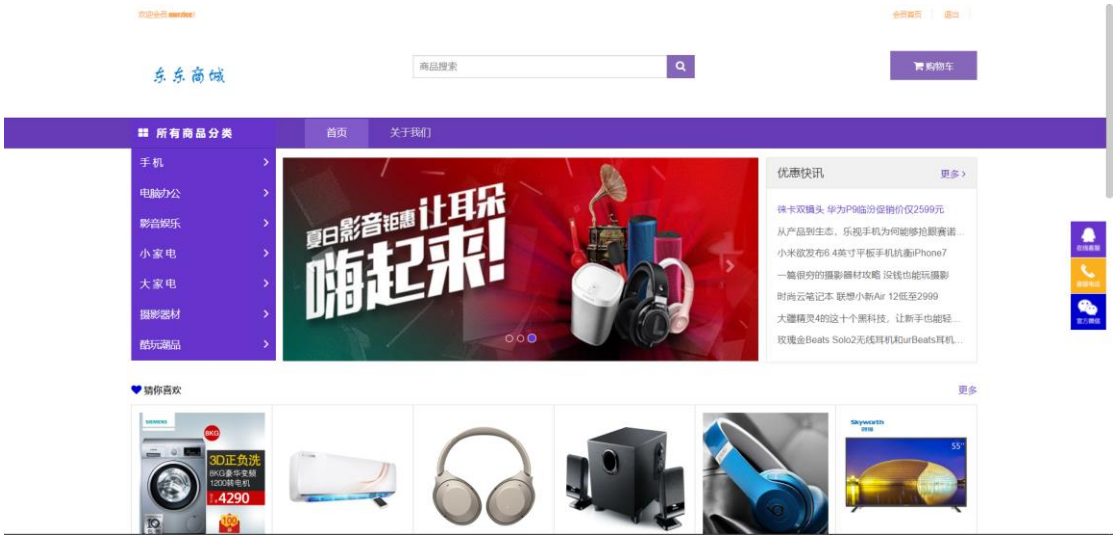
4.4.6、ProductService 接口设计

接口名字	接口备注
save(Product cate)	商品新增
update(Product cate)	商品更新
delete(Integer id)	类型删除
findOne(Integer id)	寻找某个编号的产品
long count()	计算商品数量
Page<Product> findBySubCategory(Integer cate_id, int number, int size)	获取指定的二级类目下的产品分页列表
Page<Product> findHot(int number, int size)	获取热门产品分页列表
Page<Product> findHotTopCategory(Integer cate_id, int number, int size)	获取指定一级类目下的热门产品分页列表
Page<Product> findHotSubCategory(Integer cate_id, int number, int size)	获取指定二级类目下的热门产品分页列表
Page<Product> findByLikeName(String name, int number, int size)	根据产品标题模糊查询，返回分页列表
Page<Product> findByTopCategory(Integer cate_id, int number, int size	获取指定一级类目下的所有产品分页列表
findAll()	寻找所有产品

五、界面设计

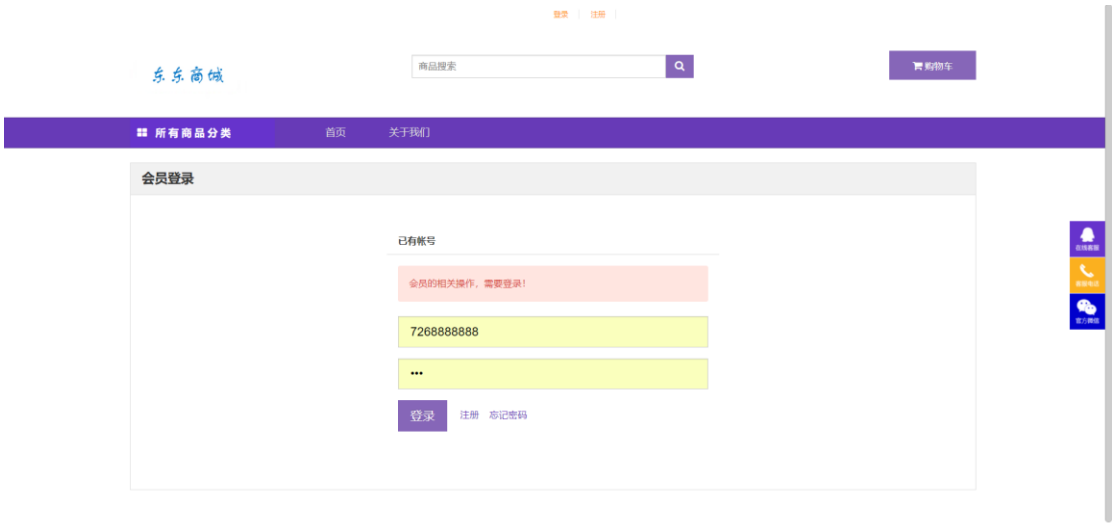
界面我主要分为主页、注册/登陆页面、用户页面、产品详情页面、购物车页面、结算页面、新闻详情页面、搜索结果页面，每个页面都包括了我的基本页面，将页面布局设置的比较合理。

5.1、主页

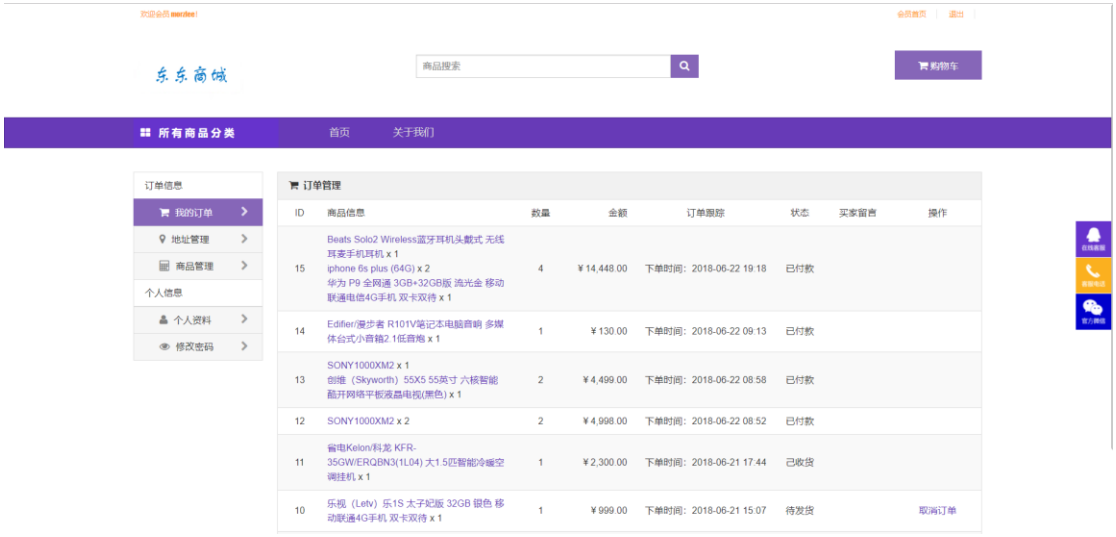


主页界面由很多个页面拼接而来，其中包括.icd_button.jsp,icd_link.jsp,icd_top.jsp,icd_meta.jsp

5.2、注册/登陆页面



5.3、用户中心



其他界面效果也差不多，就不继续展示了，整个网站已经放到互联网，链接（https://shop.lidongspace.cn/Lshop_war）

六、心得体会

编写一个 Web 首先的先决定用什么方式实现，然后将整个网站的大部分功能先确定下来。

接下来确定自己的前端要使用的框架，虽然可以不用框架，但是框架能够在短时间内给我们 能力将页面做的更好看，我本来打算用 bootstrap 框架，最后选择了一个基于 bootstrap 框架深度定制的框架 ZUI，给我编写程序编写很大的便利，然后就是学会使用 jQuery 这些框架，能够让我们对 jsp 页面进行数据解析、处理带来很大的便利，在有就是要对你心中的购物网站进行重构，看哪些模块是可以单独独立出来，在另外的页面只要包含这些模块就可以使用，这样会让你的网站保持一个完整统一的样式，而且也更规范，更容易修改。

后端建议多看看别人的代码可以帮助自己写出更高效更结构清晰的代码。例如创建一个实体，然后再以这个实体为对象进行操作，会使你的数据更安全，也让代码更加清晰。还学会将数据传到 servlet 后调用自己写的服务接口然后对数据库进行操作，这样保证了对数据库操作的统一，不容易出错，而且比每次写入数据都重新写一次 sql 语句工作量大大降低，这样虽然前期可能没有别人做的快，但是后期你的代码显得很整洁，也易于维护，无论是添加新功能还是修改。