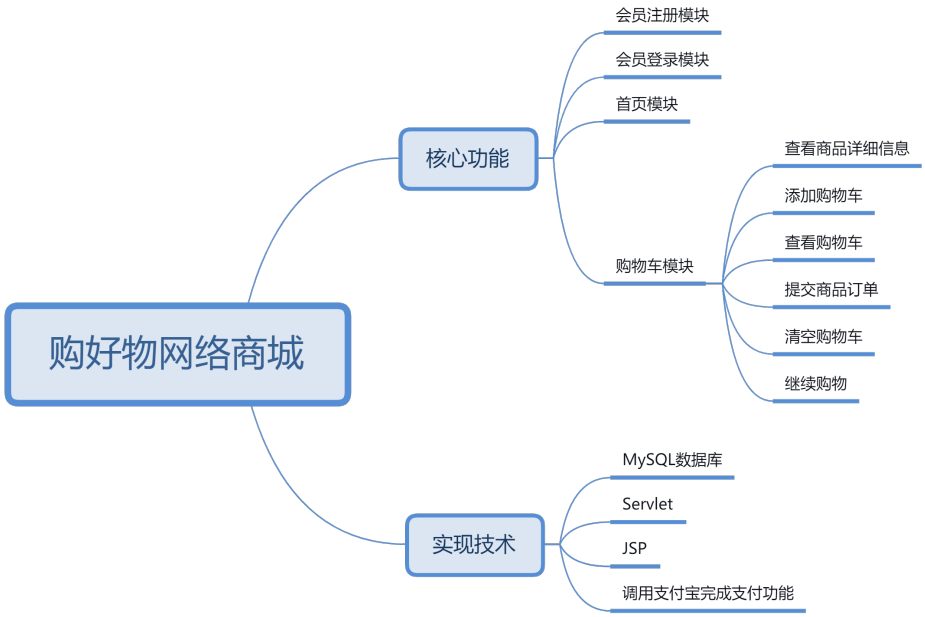
第 2 章

购好物网络商城

—— Servlet + JSP + MySQL

随着互联网技术的高速发展，电子商务已经成为现代生活不可或缺的重要组成部分，各类购物网站不断涌现。购物网站是买卖双方用于交易商品的电子商务平台。卖家可以在此平台上开设网店，发布出售商品的信息，低成本、高效率地为消费者提供商品及服务；买家可以在此平台上选择并购买喜欢的商品，足不出户地享受购物的快乐。本章将使用 Java Web 开发中的 MySQL 数据库、Servlet、JSP 等关键技术开发一个购物网站项目——购好物网络商城。

本项目的核心功能及实现技术如下：



2.1 开 发 背 景

与传统的线下购物相比，网上购物的优点十分突出，它不受时间和地点的限制，消费者可以随时随地浏览和购买商品，这种便捷性使得它越来越受到消费者的青睐。加之配套的物流服务与网上支付技术越来越完善，网上购物不仅能让消费者十分便捷地购买到心仪的商品，并且能让消费者享受到送货上门的优质服务，极大程度地提升了消费者的购物体验。如今，网上购物已经成为一种主流的购物形式，各大购物网站都在积极优化网站的功能，力求为用户提供更好的购物服务。为此，本章使用 Java Web 技术开发一个名为“购好物网络商城”的购物网站。

购好物网络商城将实现以下目标：

☑ 首页展示商品类别和主要商品；

- ☑ 商品详情页展示某件商品的详细信息；
- ☑ 支持会员注册和会员登录的功能；
- ☑ 支持购物车功能，通过购物车可以完成支付和提交订单操作。

2.2 系 统 设 计

2.2.1 开发环境

本项目的开发及运行环境如下：

- ☑ 操作系统：推荐 Windows 10、11 及以上，兼容 Windows7（SP1）。
- ☑ 开发工具：IntelliJ IDEA。
- ☑ 开发语言：Java EE。
- ☑ 数据库：MySQL 8.0。
- ☑ Web 服务器：Tomcat 9.0 及以上版本。

2.2.2 业务流程

启动项目后，购好物网络商城的首页会被自动打开。在购好物网络商城的首页上，用户既可以浏览热门商品、最新上架的商品和折扣商品，又可以通过导航栏切换页面，进而去浏览图书类、家电类、服装类和电子类的对应商品。用户注册或登录会员后，不仅可以把喜欢的商品加入购物车，而且能够在购物车中完成支付和提交订单操作。购好物网络商城的业务流程如图 2.1 所示。

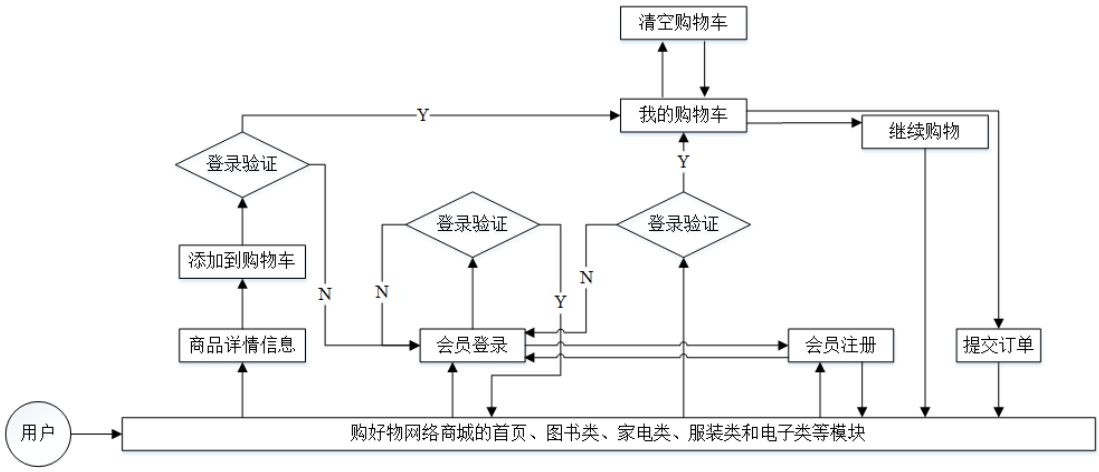


图 2.1 购好物网络商城的业务流程图

说明

购好物网络商城具有一个导航栏，其中包括首页、图书类、家电类、服装类和电子类 5 个模块，这 5 个模块的设计过程都大同小异。限于本书篇幅，这里仅讲解首页模块的设计过程。读者可以参考首页模块的设计过程，自行根据提供的源码学习其他模块的设计过程。

2.2.3 功能结构

本项目的功能结构已经在章首页中给出。作为一个网上购物网站，本项目实现的具体功能如下：

- ☑ 首页模块：展示热门商品、最新上架的商品和折扣商品，以及通过导航栏链接商品分类页面。
- ☑ 会员注册模块：允许新用户注册，成为购好物网络商城的会员。
- ☑ 会员登录模块：验证用户是否是购好物网络商城的会员，并允许会员登录。
- ☑ 购物车模块：用户通过该模块可以实现查看某件商品的详细信息、把喜欢的商品添加到购物车、查看购物车、完成支付并提交订单、清空购物车、继续购物等操作。

2.3 技 术 准 备

2.3.1 技术概览

- ☑ MySQL 数据库：数据库（Database）就是一个存储数据的仓库。通过数据库管理系统，可以有效地存储、组织和管理数据。MySQL 就是这样的一个关系型数据库管理系统（RDBMS），它可以称得上是目前运行速度最快的 SQL 语言数据库管理系统。本章将使用 JDBC 技术操作 MySQL 数据库，如使用 Class 类的静态方法 forName(String className)加载指定数据库的驱动程序。代码如下：

```
public Connection conn = null;                                          //数据库连接对象
public Statement stmt = null;                                  //Statement 对象，用于执行 SQL 语句
public ResultSet rs = null;                                             //结果集对象
private static String dbClassName = "com.mysql.cj.jdbc.Driver";         //驱动类的类名
private static String dbUrl = "jdbc:mysql://127.0.0.1:3306/db_shop?user=root&password=root"
        + "&useUnicode=true&characterEncoding=UTF-8&useSSL=false&serverTimezone=Asia/Shanghai"
        + "&zeroDateTimeBehavior=CONVERT_TO_NULL&allowPublicKeyRetrieval=true";   //访问 MySQL 数据库的路径
public static Connection getConnection() {
        Connection conn = null;                                         //声明数据库连接对象
        try {                                                           //捕捉异常
                Class.forName(dbClassName).newInstance();               //装载数据库驱动
                conn = DriverManager.getConnection(dbUrl);              //获取数据库连接对象
        } catch (Exception ee) {                                        //处理异常
                ee.printStackTrace();                                   //输出异常信息
        }
        return conn;                                                    //返回数据库连接对象
}
```

- ☑ Servlet 技术：Servlet 实质上就是按 Servlet 规范编写的 Java 类，它采用 Java 语言编写，具有 Java 语言的优点。与 Java 语言的区别是，Servlet 对象对 Web 应用进行了封装，提供了 Servlet 对 Web 应用的编程接口，以处理相应的 HTTP 请求，如处理提交数据、会话跟踪、读取和设置 HTTP 头信息等。例如，本章使用 Servlet 技术完成了会员登录模块的设计。
- ☑ JSP 技术：JSP 是一种动态网页技术标准，JSP 页面通常是由 Java 代码片段和 JSP 指令标识构成的。在接收到用户请求时，服务器首先会处理 Java 代码片段，然后生成处理结果返回客户端，客户端将呈现相应的页面效果。本章的会员注册、会员登录、首页、购物车等模块所呈现的页面效果均是采用 JSP 技术设计的。

有关 MySQL 数据库等基础知识以及开发工具 IntelliJ IDEA 在《Java 从入门到精通（第 7 版）》中有详细的讲解，对这些知识不太熟悉的读者可以参考该书对应的内容。Servlet 和 JSP 这两个技术在本书第 1 章

的“技术准备”中有详细的介绍，对这两个技术不太熟悉的读者可以参考本书第 1 章对应的内容。下面将对“调用支付宝完成支付操作”技术进行必要介绍，以确保读者可以顺利完成本项目。

2.3.2 调用支付宝完成支付操作

在购物车页面中，单击“结账”按钮，会弹出支付对话框，在该对话框中，扫描二维码将调用支付宝完成支付操作。实现此功能的基本步骤如下：

- ☑ 注册支付宝企业账户：进入支付宝开发平台（蚂蚁金服开放平台）。注册成为“商家”，选择“企业账户”，按照向导进行操作即可。
- ☑ 完成支付宝实名认证：注册支付宝企业账户后，会要求用户进行实名认证。准备以下资料后，单击“企业实名信息填写”按钮，按照向导操作即可完成实名认证。
  - ➢ 营业执照影印件；
  - ➢ 对公银行账户，可以是基本户或一般户；
  - ➢ 法定代表人的身份证影印件。

说明

如果是代理人，除以上资料外，还需要准备代理人的身份证影印件和企业委托书，必须盖有公司公章或者财务专用章。

- ☑ 申请支付套餐：支付宝提供了多种支付套餐。一般情况下，选择“即时到账”套餐，该套餐可以让用户在线向购物平台的支付宝账号支付资金，并且交易资金即时到账。申请“即时到账”套餐后，会进入审核阶段。通常情况下，2~5 天会有申请结果。
- ☑ 生成与配置密钥：在调用支付宝完成支付操作时，需要提供商户的私钥和支付宝的公钥，使用支付宝提供的一键生成工具即可获取。
- ☑ 下载 Demo：前面的工作准备就绪后，就可以测试支付功能了。这时，可以下载支付宝开发平台提供的即时到账交易接口的 Demo，根据 Demo 中的说明进行测试即可。

2.4 数据库设计

2.4.1 数据库概述

本项目采用的数据库名称为 db_shop，数据库包含 7 张数据表，如表 2.1 所示。

表 2.1 购好物网络商城的数据库结构

数 据 库 名	表 名	表 说 明
db_shop	tb_goods	商品信息表
	tb_manager	管理员信息表
	tb_member	会员信息表
	tb_order	订单信息主表
	tb_order_detail	订单信息明细表
	tb_subtype	商品小分类信息表
	tb_supertype	商品大分类信息表

2.4.2 数据表设计

在本项目中发挥主要作用的数据表有 tb_supertype、tb_subtype、tb_goods、tb_order、tb_order_detail、tb_manager 和 tb_member，下面将详细介绍这 7 张数据表的结构设计。

☑ tb_supertype（商品大分类信息表）：主要用于保存商品大分类信息，也就是父分类，其结构如表 2.2 所示。

表 2.2 tb_supertype 表结构

字段名称	数据类型	长度	是否主键	说明
ID	INT		主键	ID 号
TypeName	VARCHAR	50		分类名称

☑ tb_subtype（商品小分类信息表）：主要用于保存商品小分类信息，也就是子分类，其结构如表 2.3 所示。

表 2.3 tb_subtype 表结构

字段名称	数据类型	长度	是否主键	说明
ID	INT		主键	ID 号
superType	INT			父类 ID 号
TypeName	VARCHAR	50		分类名称

☑ tb_goods（商品信息表）：主要用于保存商品信息，其结构如表 2.4 所示。

表 2.4 tb_goods 表结构

字段名称	数据类型	长度	是否主键	说明
ID	BIGINT		主键	商品 ID
typeID	INT			类别 ID
goodsName	VARCHAR	200		商品名称
introduce	TEXT			商品简介
price	DECIMAL	19(4)		定价
nowPrice	DECIMAL	19(4)		现价
picture	VARCHAR	100		图片文件
INTime	DATETIME			录入时间
newGoods	INT			是否新品，1 为是，默认 0
sale	INT			是否特价，1 为是，默认 0
hit	INT			浏览次数

☑ tb_order（订单信息主表）：主要用于保存订单的概要信息，其结构如表 2.5 所示。

表 2.5 tb_order 表结构

字段名称	数据类型	长度	是否主键	说明
OrderID	BIGINT		主键	订单编号
bnumber	SMALLINT			品种数
username	VARCHAR	15		用户名

续表

字段名称	数据类型	长　度	是否主键	说　明
recevieName	VARCHAR	15		收货人
address	VARCHAR	100		收货地址
tel	VARCHAR	20		联系电话
OrderDate	DATETIME			订单日期
bz	VARCHAR	200		备注

☑ tb_order_detail（订单信息明细表）：主要用于保存订单的详细信息，其结构如表 2.6 所示。

表 2.6　tb_order_detail 表结构

字段名称	数据类型	长　度	是否主键	说　明
ID	BIGINT		主键	ID 号
orderID	BIGINT			与 tb_Order 表的 OrderID 字段关联
goodsID	BIGINT			商品 ID
price	DECIMAL	19(4)		价格
number	INT			数量

☑ tb_manager（管理员信息表）：主要用于保存管理员信息，其结构如表 2.7 所示。

表 2.7　tb_manager 表结构

字　段　名	数据类型	长　度	是否主键	说　明
ID	INT		主键	ID 号
manager	VARCHAR	30		管理员名称
PWD	VARCHAR	30		密码

☑ tb_member（会员信息表）：主要用于保存会员信息，其结构如表 2.8 所示。

表 2.8　tb_member 表结构

字段名称	数据类型	长　度	是否主键	说　明
ID	INT		主键	ID 号
userName	VARCHAR	20		账户名称
trueName	VARCHAR	20		真实姓名
passWord	VARCHAR	20		密码
city	VARCHAR	20		所在城市
address	VARCHAR	100		地址
postcode	VARCHAR	6		邮编
cardNO	VARCHAR	24		证件号码
cardType	VARCHAR	20		证件类型
grade	INT			等级
Amount	DECIMAL	19(4)		预存金额
tel	VARCHAR	20		联系电话
email	VARCHAR	100		邮箱
freeze	INT			是否被冻结，1 为是，默认 0

2.5 数据库公共类的编写

数据库公共类 ConnDB 被保存在 com.tools 包中，其中包含了 5 个成员变量，代码如下：

```
public class ConnDB {
    public Connection conn = null;                                  //数据库连接对象
    public Statement stmt = null;                                   //Statement 对象，用于执行 SQL 语句
    public ResultSet rs = null;                                     //结果集对象
    private static String dbClassName = "com.mysql.cj.jdbc.Driver"; //驱动类的类名
    private static String dbUrl = "jdbc:mysql://127.0.0.1:3306/db_shop?user=root&password=root"
        + "&useUnicode=true&characterEncoding=UTF-8&useSSL=false&serverTimezone=Asia/Shanghai"
        + "&zeroDateTimeBehavior=CONVERT_TO_NULL&allowPublicKeyRetrieval=true"; //访问 MySQL 数据库的路径
}
```

在 ConnDB 类中，编写用于连接数据库的 getConnection()方法，用于根据指定的数据库驱动获取数据库连接对象。如果连接失败，则输出异常信息。该方法返回一个数据库连接对象。getConnection()方法的代码如下：

```
public static Connection getConnection() {
    Connection conn = null;                                 //声明数据库连接对象
    try {                                                   //捕捉异常
        Class.forName(dbClassName).newInstance();           //装载数据库驱动
        conn = DriverManager.getConnection(dbUrl);          //获取数据库连接对象
    } catch (Exception ee) {                                //处理异常
        ee.printStackTrace();                               //输出异常信息
    }
    return conn;                                            //返回数据库连接对象
}
```

在 ConnDB 类中，编写用于更新数据的 executeUpdate()方法，该方法的返回值为 int 型的整数，表示更新数据的行数。executeUpdate()方法的代码如下：

```
public int executeUpdate(String sql) {
    int result = 0;                                         //更新数据的记录条数
    try {                                                   //捕捉异常
        conn = getConnection();                             //获取数据库连接
        //创建用于执行 SQL 语句的 Statement 对象
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        result = stmt.executeUpdate(sql);                   //执行 SQL 语句
    } catch (SQLException ex) {                             //处理异常
        result = 0;                                         //指定更新数据的记录条数为 0，表示没有更新数据
        ex.printStackTrace();                               //输出异常信息
    }
    try {                                                   //捕捉异常
        stmt.close();                                       //关闭用于执行 SQL 语句的 Statement 对象
    } catch (SQLException ex1) {                            //处理异常
        ex1.printStackTrace();                              //输出异常信息
    }
    return result;                                          //返回更新数据的记录条数
}
```

在 ConnDB 类中，编写用于查询数据的 executeQuery()方法。在该方法中，首先调用 getConnection()方法获取数据库连接对象，然后通过该对象的 createStatement()方法创建一个 Statement 对象，并且调用该对象的 executeQuery()方法执行指定的 SQL 语句，进而实现查询数据的功能。executeQuery()方法的代码如下：

```
public ResultSet executeQuery(String sql) {
    try {                                                   //捕捉异常
```

```
            conn = getConnection();                                    //获取数据库连接
            //创建用于执行 SQL 语句的 Statement 对象
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
            rs = stmt.executeQuery(sql);                               //执行 SQL 语句
        } catch (SQLException ex) {                                    //处理异常
            ex.printStackTrace();                                      //输出异常信息
        }
        return rs;                                                     //返回查询结果
}
```

在 ConnDB 类中，编写用于关闭数据库连接的 close()方法。在该方法中，首先关闭结果集对象，然后关闭 Statement 对象，最后再关闭数据库连接对象。close()方法的代码如下：

```
public void close() {
        try {                                                          //捕捉异常
            if (rs != null) {
                rs.close();                                            //关闭结果集对象
            }
            if (stmt != null) {
                stmt.close();                                          //关闭 Statement 对象
            }
            if (conn != null) {
                conn.close();                                          //关闭数据库连接对象
            }
        } catch (Exception e) {                                        //处理异常
            e.printStackTrace(System.err);                             //输出异常信息
        }
}
```

2.6 会员注册模块设计

会员注册模块主要用于实现新用户成为购好物网络商城会员的功能。在会员注册页面中，用户需要先根据实际情况填写当前用户的信息，再单击“同意协议并注册”按钮。程序会自动验证用户输入的会员信息是否唯一。如果唯一，就把填写完成的会员信息保存到数据库中，完成会员注册的操作；否则给出提示。会员注册页面的效果如图 2.2 所示。

图 2.2 会员注册页面

2.6.1 会员模型类的编写

会员模型类 Member 被保存在 com.model 包中，其中包含会员 ID、账户、真实姓名、密码、所在城市、地址、邮编、证件号码、证件类型、联系电话、邮箱、新密码等属性。此外，还要为这些属性添加 Getters 和 Setters 方法。这样，不仅可以控制对上述属性的访问和修改，还可以使代码更加安全，并且可以维护。创建会员模型类 Member 的代码如下：

```
public class Member {
    private Integer ID = Integer.valueOf("-1");                    //会员 ID 属性
    private String username = "";                                  //账户属性
    private String truename = "";                                  //真实姓名属性
    private String pwd = "";                                       //密码属性
    private String city = "";                                      //所在城市属性
    private String address = "";                                   //地址属性
    private String postcode = "";                                  //邮编属性
    private String cardno = "";                                    //证件号码属性
    private String cardtype = "";                                  //证件类型属性
    private String tel = "";                                       //联系电话属性
    private String email = "";                                     //邮箱属性
    private String newPwd = "";                                    //新密码

    public String getNewPwd() {
        return newPwd;
    }
    public void setNewPwd(String newPwd) {
        this.newPwd = newPwd;
    }
    public Integer getID() {
        return ID;
    }
    public void setID(Integer iD) {
        ID = iD;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getTruename() {
        return truename;
    }
    public void setTruename(String truename) {
        this.truename = truename;
    }
    public String getPwd() {
        return pwd;
    }
    public void setPwd(String pwd) {
        this.pwd = pwd;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getPostcode() {
        return postcode;
    }
    public void setPostcode(String postcode) {
```

```
        this.postcode = postcode;
    }
    public String getCardno() {
        return cardno;
    }
    public void setCardno(String cardno) {
        this.cardno = cardno;
    }
    public String getCardtype() {
        return cardtype;
    }
    public void setCardtype(String cardtype) {
        this.cardtype = cardtype;
    }
    public String getTel() {
        return tel;
    }
    public void setTel(String tel) {
        this.tel = tel;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

2.6.2 会员数据库操作接口及其实现类的编写

会员数据库操作接口 MemberDao 被保存在 com.dao 包中，其中定义一个 insert()方法（用于保存会员信息）、一个 select()方法（用于查询会员信息）、一个 update()方法（用于修改会员信息）和一个 delete()方法（用于删除会员信息）。需要注意的是，这里只定义了方法，没有实现方法。会员数据库操作接口 MemberDao 的代码如下：

```
public interface MemberDao {
    public int insert(Member m);                    //保存会员信息
    public List select();                           //查询会员信息
    public int update(Member m);                    //修改会员信息
    public int delete(Member m);                    //删除会员信息
}
```

创建接口后，必须实现该接口。在 com.dao 包中创建一个 MemberDao 接口的实现类，类名为 MemberDaoImpl。在实现 MemberDao 接口中的各个方法前，需要创建数据库连接类的对象和字符串操作类的对象。代码如下：

```
private ConnDB conn = new ConnDB();                 //创建数据库连接类的对象
private ChStr chStr = new ChStr();                  //创建字符串操作类的对象
```

在 MemberDaoImpl 类中，使用 Insert into 语句实现用于保存会员信息的 insert()方法；使用 select 语句实现用于查询会员信息 select()方法；使用 update 语句实现用于修改会员信息 update()方法；使用 delete 语句实现用于删除会员信息 delete()方法。MemberDaoImpl 类的代码如下：

```
@Override
public int insert(Member m) {
    int ret = -1;                                   //用于记录更新记录的条数
    try {                                           //捕捉异常
```

```
            if(m.getUsername()!=null){
                String sql = "Insert into tb_Member (UserName,TrueName,PassWord,City,address,postcode,"
                        + "CardNO,CardType,Tel,Email) values('"
                        + m.getUsername() + "','" + chStr.chStr(m.getTruename()) + "','"
                        + chStr.chStr(m.getPwd()) + "','" + chStr.chStr(m.getCity()) + "','" + chStr.chStr(m.getAddress())
                        + "','" + chStr.chStr(m.getPostcode()) + "','" + m.getCardno() + "','"
                        + chStr.chStr(m.getCardtype()) + "','" + chStr.chStr(m.getTel()) + "','"+ m.getEmail()
                        + "')";                                          //用于实现保存会员信息的 SQL 语句
                ret = conn.executeUpdate(sql);                           //执行 SQL 语句实现保存会员信息到数据库
            }else{
                ret = 0;                                                 //表示注册失败
            }
        } catch (Exception e) {                                          //处理异常
            e.printStackTrace();                                         //输出异常信息
            ret = 0;                                                     //设置变量的值为 0，表示保存会员信息失败
        }
        conn.close();                                                    //关闭数据库的连接
        return ret;                                                      //返回更新记录的条数
}

@Override
public List select() {
        Member form = null;                                              //声明会员对象
        List list = new ArrayList();                                     //创建一个 List 集合对象，用于保存会员信息
        String sql = "select * from tb_member";                          //查询全部会员信息的 SQL 语句
        ResultSet rs = conn.executeQuery(sql);                           //执行查询操作
        try {                                                            //捕捉异常
            while (rs.next()) {
                form = new Member();                                     //实例化一个会员对象
                form.setID(Integer.valueOf(rs.getString(1)));            //获取会员 ID
                list.add(form);                                          //把会员信息添加到 List 集合对象中
            }
        } catch (SQLException ex) {                                      //处理异常
        }
        conn.close();                                                    //关闭数据库的连接
        return list;
}

//执行删除操作
public int delete(Member m) {
        String sql = "delect from tb_member where ID=" + m.getID();
        int ret = conn.executeUpdate(sql);
        conn.close();
        return 0;
}

//执行修改操作
public int update(Member m) {
        int ret = -1;
        try {
            String sql = "update tb_member set TrueName='" + chStr.chStr(m.getTruename()) + "',UserName='"
                    + chStr.chStr(m.getUsername()) + "',PassWord='" + chStr.chStr(m.getNewPwd()) + "',City='"
                    + chStr.chStr(m.getCity()) + "',address='" + chStr.chStr(m.getAddress()) + "',postcode='"
                    + chStr.chStr(m.getPostcode()) + "',CardNO='" + chStr.chStr(m.getCardno()) + "',CardType='"
                    + chStr.chStr(m.getCardtype()) + "',Tel='" + chStr.chStr(m.getTel()) + "',Email='"
                    + chStr.chStr(m.getEmail()) + "' where ID=" + m.getID();
            ret = conn.executeUpdate(sql);
            System.out.println(sql);
        } catch (Exception e) {
            e.printStackTrace();
            ret = 0;
        }
```

```
        conn.close();                                                    //关闭数据库连接
        return ret;
}
```

2.6.3 会员注册页面的编写

本项目采用一个 JSP 文件作为会员注册页面，这里指定的 JSP 文件是位于 WebContent/front 下的 register_deal.jsp。在 register_deal.jsp 文件中添加代码，用于创建 ConnDB、MemberDaoImpl 和 Member 类的对象，并且通过“<jsp:setProperty name="member" property="*"/>”对 Member 类的所有属性进行赋值，用于获取用户填写的注册信息。代码如下：

```
<%-- 创建 ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
<%-- 创建 MemberDaoImpl 类的对象 --%>
<jsp:useBean id="ins_member" scope="page" class="com.dao.MemberDaoImpl" />
<%-- 创建 Member 类的对象，并对 Member 类的所有属性进行赋值 --%>
<jsp:useBean id="member" scope="request" class="com.model.Member">
        <jsp:setProperty name="member" property="*" />
</jsp:useBean>
```

判断输入的账号是否存在，如果存在则给予提示，否则调用 MemberDaoImpl 类的 insert()方法，将填写的会员信息保存到数据库中。代码如下：

```
<%
        request.setCharacterEncoding("UTF-8");                           //设置请求的编码为 UTF-8
        String username = member.getUsername();                          //获取会员账号
        ResultSet rs = conn.executeQuery("select * from tb_Member where username='"
        + username + "'");
        if (rs.next()) {                                                 //如果结果集中有数据
                out.println("<script language='javascript'>alert('该账号已经存在，请重新注册！');"
                                + "window.location.href='register.jsp';</script>");
        } else {
                int ret = 0;                                             //记录更新记录条数的变量
                ret = ins_member.insert(member);                         //将填写的会员信息保存到数据库
                if (ret != 0) {
                        session.setAttribute("username", username);      //将会员账号保存到 Session 中
                        out.println("<script language='javascript'>alert('会员注册成功！');"
                                        + "window.location.href='index.jsp';</script>");
                } else {
                        out.println("<script language='javascript'>alert('会员注册失败！');"
                                        + "window.location.href='register.jsp';</script>");
                }
        }
%>
```

2.7 会员登录模块设计

会员登录模块主要用于验证用户是否为购好物网络商城的会员，并允许会员登录商城，进行后续相关操作。在会员登录页面中，填写会员账户、密码和验证码（如果验证码看不清楚，可以单击验证码图片刷新验证码），单击“登录”按钮，即可完成会员登录的操作，如图 2.3 所示。如果没有输入账户、密码或者验证码，程序都将给予提示。此外，账户、密码或者验证码输入错误，程序也将给予提示。

图 2.3　会员登录页面

2.7.1　会员登录页面的编写

本项目采用一个 JSP 文件作为会员登录页面，这里指定的 JSP 文件是位于 WebContent/front 下的 login.jsp。在 login.jsp 中，没有业务逻辑，主要使用 标签规划页面区域。login.jsp 的代码如下：

```
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title>登录-购好物网络商城</title>
<link rel="stylesheet" href="css/mr-01.css" type="text/css">
</head>

<body>
	<div id="mr-mainbody" class="container mr-mainbody">
		<div class="row">
			<!-- 主体内容 -->
			<div id="mr-content" class="mr-content col-xs-12">
				<div class="login-wrap" style="margin-bottom: 60px; margin-top: 50px">
					<div style="max-width: 540px; margin: 0 auto;">
						<a href="index.jsp" title="点击返回首页"><img src="images/51logo.png"></a>
					</div>
					<div class="login">
						<div class="page-header" style="pause: 0px;"> <h1 class="login_h1">
						会员登录</h1> </div>
						<!-- 会员登录表单 -->
						<form action="login_check.jsp" method="post" class="form-horizontal">
							<fieldset>
								<div class="form-group">
									<div class="col-sm-4 control-label">
										<label id="username-lbl" for="username" class="required">
										账户 ： </label>
									</div>
									<div class="col-sm-8">
```

```
                                        <!-- 账户文本框 -->
                                        <input type="text" name="username" id="username" value=""
                                          size="38" class="validate-username required"
                                          required="required" autofocus="">
                                    </div>
                                </div>
                                <div class="form-group">
                                    <div class="col-sm-4 control-label">
                                        <label id="password-lbl" for="password" class="required">
                                        密码 ： </label>
                                    </div>
                                    <div class="col-sm-8">
                                        <!-- 密码文本框 -->
                                        <input type="password" name=PWD id="password" value=""
                                        class="validate-password required" size="38" maxlength="99"
                                        required="required" aria-required="true">
                                    </div>
                                </div>
                                <div class="form-group">
                                    <div class="col-sm-4 control-label">
                                        <label id="password-lbl" for="password" class="required">
                                        验证码 ： </label>
                                    </div>
                                    <div class="col-sm-8" style="clear: none;">
                                        <!-- 验证码文本框 -->
                                        <input type="text" name="checkCode" id="checkCode" value=""
                                        class="validate-password required" style="float: left;"
                                            title="验证码区分大小写" size="18" maxlength="4"
                                            required="required" aria-required="true">
                                            <!-- 显示验证码 -->
                                            <img src="../CheckCode" name="checkCode"
                                            onClick="myReload()" width="116" height="43"
                                            class="img_checkcode" id="img_checkCode" />
                                    </div>
                                </div>
                                <div class="form-group">
                                    <div class="col-sm-offset-4 col-sm-8">
                                    <button type="submit" class="btn btn-primary login">登录</button>
                                    </div>
                                </div>
                                <div class="form-group"
                                    style="border-top: 1px solid #D9D9D9; margin: 20px;">
                                    <label
                                        style="float: right; color: #858585; margin-right: 40px;
                                        margin-top: 10px; font-size: 14px;">没有账户？ <a
                                        href="register.jsp">立即注册</a></label>
                                </div>
                            </fieldset>
                        </form>
                    </div>
                </div>
            </div>
            <!-- //主体内容 -->
        </div>
    </div>
    <script language="javascript">
        //刷新验证码
        function myReload() {
```

```
                document.getElementById("img_checkCode").src = document.getElementById("img_checkCode").src
                + "?nocache=" + new Date().getTime();
            }
        </script>
</body>
</html>
```

在向会员登录页面添加验证码时，引用了 com.tools 下的用于生成验证码的 CheckCode.java 文件。在该文件中，包含一个 getRandColor()方法，该方法通过随机生成 RGB 颜色中的 r 值、g 值和 b 值获取随机颜色。getRandColor()方法代码如下：

```
public Color getRandColor(int s, int e) {
    Random random = new Random();
    if (s > 255)
        s = 255;
    if (e > 255)
        e = 255;
    int r = s + random.nextInt(e - s);                    //随机生成 RGB 颜色中的 r 值
    int g = s + random.nextInt(e - s);                    //随机生成 RGB 颜色中的 g 值
    int b = s + random.nextInt(e - s);                    //随机生成 RGB 颜色中的 b 值
    return new Color(r, g, b);
}
```

在 CheckCode.java 文件中，还包含一个用于生成验证码的 Servlet 的 service()方法，该方法用于处理来自客户端的请求（生成验证码），并将处理结果（验证码的图片）返回客户端。service()方法的代码如下：

```
public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setHeader("Pragma", "No-cache");
    response.setHeader("Cache-Control", "No-cache");
    response.setDateHeader("Expires", 0);
    //指定生成的响应是图片
    response.setContentType("image/jpeg");
    int width = 116;                                      //指定验证码的宽度
    int height = 33;                                      //指定验证码的高度

    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();                     //获取 Graphics 类的对象
    Random random = new Random();                         //实例化一个 Random 对象
    Font mFont = new Font("宋体", Font.BOLD, 22);         //通过 Font 构造字体
    g.fillRect(0, 0, width, height);                      //绘制验证码背景
    g.setFont(mFont);                                     //设置字体
    g.setColor(getRandColor(180, 200));                   //设置颜色
    //画随机的线条
    for (int i = 0; i < 100; i++) {
        int x = random.nextInt(width - 1);
        int y = random.nextInt(height - 1);
        int x1 = random.nextInt(3) + 1;
        int y1 = random.nextInt(6) + 1;
        g.drawLine(x, y, x + x1, y + y1);                 //绘制直线
    }
    /************************** 画一条折线 ********************************/
    //创建一个供画笔选择线条粗细的对象
    BasicStroke bs = new BasicStroke(2f, BasicStroke.CAP_BUTT, BasicStroke.JOIN_BEVEL);
    Graphics2D g2d = (Graphics2D) g;                      //通过 Graphics 类的对象创建一个 Graphics2D 类的对象
    g2d.setStroke(bs);                                    //改变线条的粗细
    g.setColor(Color.GRAY);                               //设置当前颜色为预定义颜色中的灰色
    int lineNumber = 4;                                   //指定端点的个数
    int[] xPoints = new int[lineNumber];                  //定义保存 x 轴坐标的数组
```

```
    int[] yPoints = new int[lineNumber];                    //定义保存 x 轴坐标的数组
    //通过循环为 x 轴坐标和 y 轴坐标的数组赋值
    for (int j = 0; j < lineNumber; j++) {
        xPoints[j] = random.nextInt(width - 1);
        yPoints[j] = random.nextInt(height - 1);
    }
    g.drawPolyline(xPoints, yPoints, lineNumber);           //绘制折线
    /*************************************************************************/
    String sRand = "";
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
        char ctmp = (char) (random.nextInt(26) + 65);       //生成 A~Z 的字母
        sRand += ctmp;
        Color color = new Color(20 + random.nextInt(110), 20 + random.nextInt(110), 20 + random.nextInt(110));
        g.setColor(color);                                  //设置颜色
        /** **随机缩放文字并将文字旋转指定角度* */
        //将文字旋转指定角度
        Graphics2D g2d_word = (Graphics2D) g;
        AffineTransform trans = new AffineTransform();
        trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
        //缩放文字
        float scaleSize = random.nextFloat() + 0.8f;
        if (scaleSize > 1f)
            scaleSize = 1f;
        trans.scale(scaleSize, scaleSize);                  //进行缩放
        g2d_word.setTransform(trans);
        /** ********************* */
        g.drawString(String.valueOf(ctmp), width / 6 * i + 23, height / 2);

    }
    //将生成的验证码保存到 Session 中
    HttpSession session = request.getSession(true);
    session.setAttribute("randCheckCode", sRand);
    g.dispose();
    ImageIO.write(image, "JPEG", response.getOutputStream());
}
```

2.7.2 生成验证码的编写

在配置一个用于生成验证码的 Servlet 时，主要是通过<servlet>标记先配置 Servlet 文件，再通过<servlet-mapping>标记配置一个映射路径，用于使用该 Servlet。因此，在本项目的 WebContent/WEB-INF/web.xml 文件中添加如下代码：

```
<servlet>
        <servlet-name>CheckCode</servlet-name>
        <servlet-class>com.tools.CheckCode</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>CheckCode</servlet-name>
        <url-pattern>/CheckCode</url-pattern>
</servlet-mapping>
```

打开会员登录页面 login.jsp，使用<img>标记显示验证码，并且调用 myReload()方法，实现单击该验证码图片时重新获取一个验证码图片功能。代码如下：

```
<img src="../CheckCode" name="img_checkCode" onClick="myReload()" width="116"
         height="43" class="img_checkcode" id="img_checkCode" />
```

2.7.3 编写会员登录处理页

在实现会员登录的功能时，除了会员登录页面，还需要给表单设置一个处理页 login_check.jsp（位于 WebContent/front 下），该处理页用来将会员输入的账户和密码与数据库中的数据进行匹配，并给出提示。

在 login_check.jsp 文件中，导入 java.sql 包中的 ResultSet 类，并且创建 ConnDB 类的对象。代码如下：

```
<%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.sql.ResultSet"%>
<%-- 创建 ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
```

获取当前用户输入的账号和密码，并将其与数据库中保存的账户和密码进行匹配，根据匹配结果给予相应的提示，并跳转到指定页面。代码如下：

```
<%
String username = request.getParameter("username");                          //获取账户
String checkCode = request.getParameter("checkCode");                        //获取验证码
if (checkCode.equals(session.getAttribute("randCheckCode").toString())) {
    try {                                                                    //捕捉异常
        ResultSet rs = conn.executeQuery("select * from tb_Member where username='" + username + "'");
        if (rs.next()) {                                                     //如果找到相应的账号
            String PWD = request.getParameter("PWD");                        //获取密码
            if (PWD.equals(rs.getString("password"))) {                      //如果输入的密码和获取的密码一致
                //把当前的账户保存到 session 中，实现登录
                session.setAttribute("username", username);
                response.sendRedirect("index.jsp");                          //跳转到前台首页
            } else {
                out.println(
        "<script language='javascript'>alert('您输入的用户名或密码错误，请与管理员联系!');"
                        +"window.location.href='login.jsp';</script>");
            }
        } else {
            out.println(
                "<script language='javascript'>alert('您输入的用户名或密码错误，或您的账户"+
               "已经被冻结，请与管理员联系!');window.location.href='login.jsp';</script>");
            }
        } catch (Exception e) {                                              //处理异常
            out.println(
                    "<script language='javascript'>alert('您的操作有误!');"
                  +"window.location.href='login.jsp';</script>");
        }
        conn.close();                                                        //关闭数据库连接
    } else {
        out.println("<script language='javascript'>alert('您输入的验证码错误!');history.back();</script>");
    }
%>
```

2.8 首页模块设计

当用户访问购好物网络商城时，首先进入的就是首页。在首页中，用户既可以浏览热门商品、最新上架的商品和折扣商品，又可以浏览图书类、家电类、服装类和电子类的商品。购好物网络商城首页的效果如图 2.4 所示。

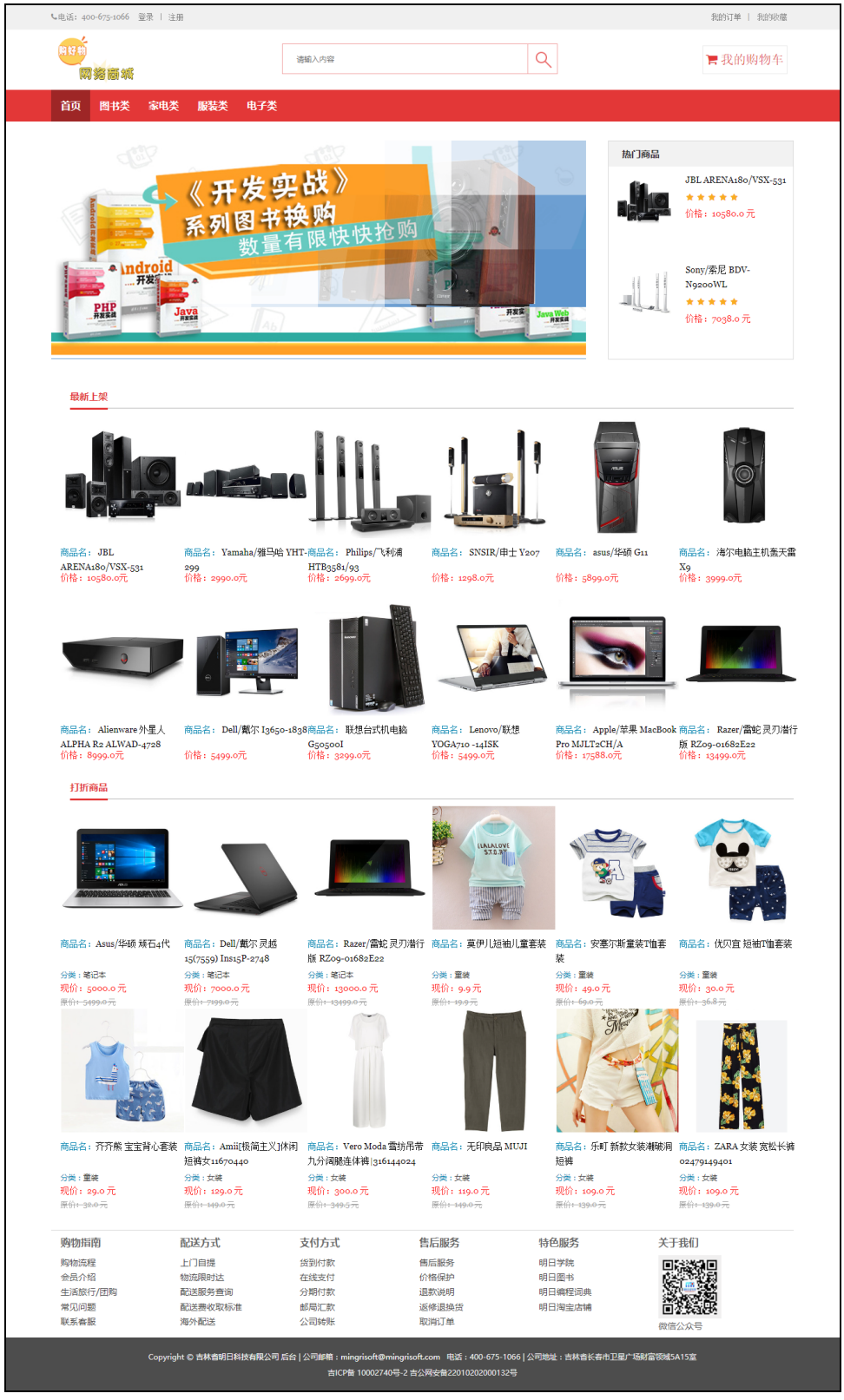


图 2.4　购好物网络商城的首页

2.8.1 实现显示最新上架商品的功能

本项目采用一个 JSP 文件用于显示最新上架的商品，这里指定的 JSP 文件是位于 WebContent/front 下的 index.jsp。在 index.jsp 文件中，因为在实现查询最新上架商品时，需要访问数据库，所以需要导入 java.sql.ResultSet 类并创建 com.tools.ConnDB 类的对象。代码如下：

```
<%@ page import="java.sql.ResultSet"%>                          <%-- 导入 java.sql.ResultSet 类 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />  <%-- 创建 com.tools.ConnDB 类的对象 --%>
```

在 index.jsp 文件中，调用 ConnDB 类的 executeQuery()方法执行 SQL 语句，用于从数据表中查询最新上架商品，这里需要编写一个连接查询的 SQL 语句。另外，还需要定义保存商品信息的变量。代码如下：

```
<%
    /* 最新上架商品信息 */
    ResultSet rs_new = conn.executeQuery(
            "select t1.ID, t1.GoodsName,t1.price,t1.picture,t2.TypeName "
            +"from tb_goods t1,tb_subType t2 where t1.typeID=t2.ID and "
            +"t1.newGoods=1 order by t1.INTime desc limit 12");      //查询最新上架商品信息
    int new_ID = 0;                                                  //保存最新上架商品 ID 的变量
    String new_goodsname = "";                                       //保存最新上架商品名称的变量
    float new_nowprice = 0;                                          //保存最新上架商品价格的变量
    String new_picture = "";                                         //保存最新上架商品图片的变量
    String typeName = "";                                            //保存商品分类的变量
%>
```

在 index.jsp 文件中，通过循环显示从数据库获取的 12 条最新上架商品信息，将获取的商品信息显示到页面的“最新上架商品展示区”。代码如下：

```
<%
while (rs_new.next()) {                                              //设置一个循环
    new_ID = rs_new.getInt(1);                                       //获取最新上架商品的 ID
    new_goodsname = rs_new.getString(2);                             //获取最新上架商品的商品名称
    new_nowprice = rs_new.getFloat(3);                               //获取最新上架商品的价格
    new_picture = rs_new.getString(4);                               //获取最新上架商品的图片
    typeName = rs_new.getString(5);                                  //获取最新上架商品的类别
%>
```

2.8.2 实现显示打折商品的功能

用于显示打折商品的 JSP 文件也是位于 WebContent/front 下的 index.jsp。在 index.jsp 文件中，调用 ConnDB 类的 executeQuery()方法执行 SQL 语句，用于从数据表中查询打折商品，这里也需要编写一个连接查询的 SQL 语句。另外，还需要定义保存商品信息的变量。代码如下：

```
/* 打折商品信息 */
ResultSet rs_sale = conn.executeQuery(
            "select t1.ID, t1.GoodsName,t1.price,t1.nowPrice,t1.picture,t2.TypeName "
            +"from tb_goods t1,tb_subType t2 where t1.typeID=t2.ID and t1.sale=1 "
            +"order by t1.INTime desc limit 12");                    //查询打折商品信息
int sale_ID = 0;                                                     //保存打折商品 ID 的变量
String s_goodsname = "";                                             //保存打折商品名称的变量
float s_price = 0;                                                   //保存打折商品的原价格的变量
float s_nowprice = 0;                                                //保存打折商品的打折后价格的变量
String s_introduce = "";                                             //保存打折商品简介的变量
String s_picture = "";                                               //保存打折商品图片的变量
```

在 index.jsp 文件中，通过循环显示从数据库获取的 12 条打折商品信息，将获取的商品信息显示到页面

的“打折商品展示区”。代码如下：

```
<%
        while (rs_sale.next()) {                                    //设置一个循环
                sale_ID = rs_sale.getInt(1);                        //获取打折商品的 ID
                s_goodsname = rs_sale.getString(2);                 //获取打折商品的商品名称
                s_price = rs_sale.getFloat(3);                      //获取打折商品的原价
                s_nowprice = rs_sale.getFloat(4);                   //获取打折商品的现价
                s_picture = rs_sale.getString(5);                   //获取打折商品的图片
                typeName = rs_sale.getString(6);                    //获取打折商品的类别
%>
```

2.8.3 实现显示热门商品的功能

热门商品是指商城中点击率最高的商品，这里将获取并显示两件商品。用于显示热门商品的 JSP 文件依然是位于 WebContent/front 下的 index.jsp。在 index.jsp 文件中，调用 ConnDB 类的 executeQuery()方法执行 SQL 语句，用于从数据表中查询点击率最高的两件商品，这里需要编写一个倒序排列的 SQL 语句。另外，还需要定义保存商品信息的变量。代码如下：

```
/* 热门商品信息 */
ResultSet rs_hot = conn
                .executeQuery("select ID,GoodsName,nowprice,picture "
                +"from tb_goods order by hit desc limit 2");        //查询热门商品信息
int hot_ID = 0;                                                     //保存热门商品 ID 的变量
String hot_goodsName = "";                                          //保存热门商品名称的变量
float hot_nowprice = 0;                                             //保存热门商品价格的变量
String hot_picture = "";                                            //保存热门商品图片的变量
```

在 index.jsp 文件中，通过循环显示从数据库获取的两条热门商品信息，将获取的商品信息显示到页面的“热门商品展示区”。代码如下：

```
<%
        while (rs_hot.next()) {                                     //设置一个循环
                hot_ID = rs_hot.getInt(1);                          //获取商品 ID
                hot_goodsName = rs_hot.getString(2);                //获取商品名称
                hot_nowprice = rs_hot.getFloat(3);                  //获取商品价格
                hot_picture = rs_hot.getString(4);                  //获取商品图片
%>
```

2.9 购物车模块设计

会员登录购好物网络商城后，使页面跳转至“我的购物车”页面的方式有两种：一种是单击首页右上角的“我的购物车”按钮；另一种是在某件商品的详细信息页面上单击“添加到购物车”按钮。在“我的购物车”页面上，会员能够看到已经被添加到购物车的商品信息。

会员如果想要为购物车里的商品提交订单，那么需要首先填写物流信息，然后单击“我的购物车”页面上的“结账”按钮，接着用支付宝扫描对话框中的二维码进行支付，再单击对话框上的“支付”按钮完成支付操作，最后程序会提交订单并在对话框中显示订单号。

会员如果想要继续购物，就单击“我的购物车”页面上的“继续购物”按钮，页面会跳转至购好物网络商城的首页。

会员如果想要清空购物车，就单击“我的购物车”页面上的“清空购物车”按钮，这时在“我的购物车”页面上显示的商品详细信息会被删除。在清空购物车后，会员可以单击“购物车为空”页面上的“继续购物”

按钮，将页面跳转至购好物网络商城的首页。

2.9.1 购物车商品模型类的编写

在 com.model 包中，创建一个购物车商品模型类 Goodselement。在该类中，添加 3 个公有类型的属性，分别表示商品 ID、当前价格和数量。代码如下：

```
package com.model;

public class Goodselement {
        public int ID;                                          //商品 ID
        public float nowprice;                                  //当前价格
        public int number;                                      //数量
}
```

2.9.2 实现查看商品详细信息的功能

会员登录购好物网络商城后，单击首页上的任何商品名称或者商品图片，都会显示该商品的详细信息页面，如图 2.5 所示。在显示某件商品详细信息的页面上，会员可以浏览该商品的名称、当前价格、商品描述等信息。下面将介绍查看某件商品详细信息的页面的实现过程。

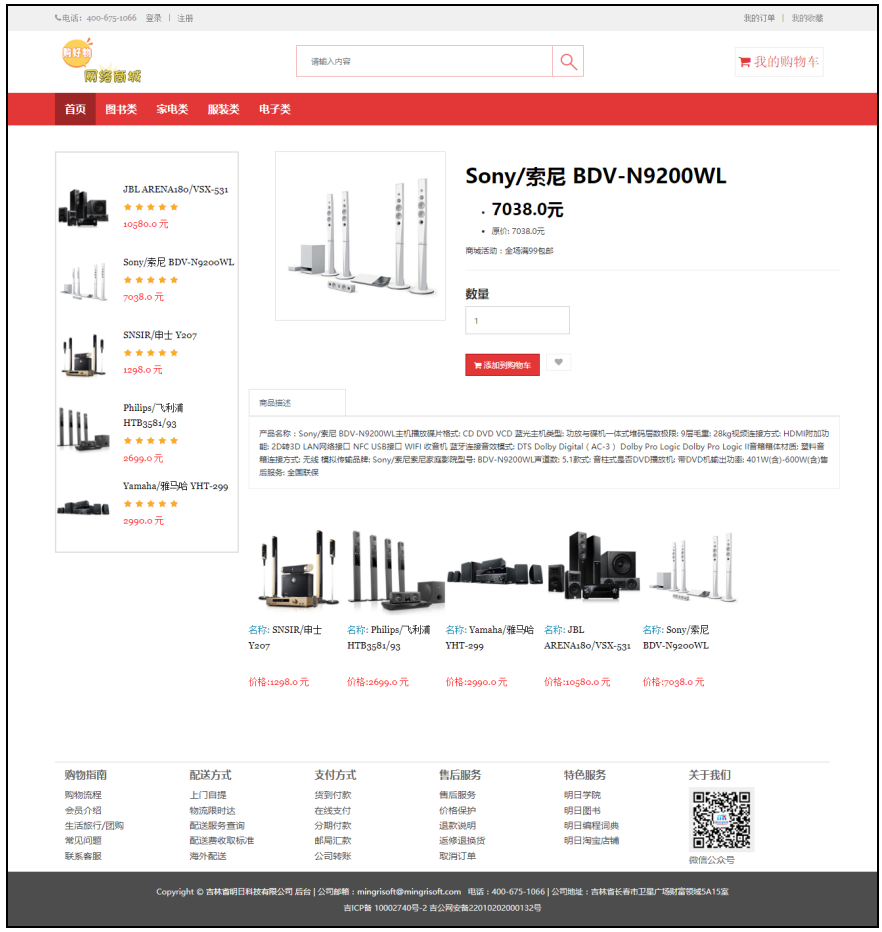


图 2.5 商品的详细信息页面

WebContent/front 下的 goodsDetail.jsp 文件用于显示某件商品的详细信息。在 goodsDetail.jsp 文件中，导入 java.sql 包中的 ResultSet 类，并且创建 ConnDB 类的对象。代码如下：

```
<%@ page import="java.sql.ResultSet"%>                    <%-- 导入 java.sql.ResultSet 类 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />    <%-- 创建 com.tools.ConnDB 类的对象 --%>
```

在 goodsDetail.jsp 文件中，根据获取的商品 ID 查询商品信息。具体方法：首先获取商品 ID，然后根据该商品 ID 从数据表中获取所需的商品信息，如果找到对应的商品，则将商品信息保存到相应的变量中，最后关闭数据库连接。代码如下：

```
<%
    int typeSystem = 0;                                          //保存商品类型 ID 的变量
    int ID = Integer.parseInt(request.getParameter("ID"));       //获取商品 ID
    if (ID > 0) {
        ResultSet rs = conn.executeQuery("select ID,GoodsName,Introduce,nowprice,picture, "
        + " price,typeID from tb_goods where ID=" + ID);         //根据 ID 查询商品信息
        String goodsName = "";                                   //保存商品名称的变量
        float nowprice = (float) 0.0;                            //保存商品现价的变量
        float price = (float) 0.0;                               //保存商品原价的变量
        String picture = "";                                     //保存商品图片的变量
        String introduce = "";                                   //保存商品描述的变量
        if (rs.next()) {                                         //如果找到对应的商品信息
            goodsName = rs.getString(2);                         //获取商品名称
            introduce = rs.getString(3);                         //获取商品描述
            nowprice = rs.getFloat(4);                           //获取商品现价
            picture = rs.getString(5);                           //获取商品图片
            price = rs.getFloat(6);                              //获取商品原价
            typeSystem = rs.getInt(7);                           //获取商品类别 ID
        }
        conn.close();                                            //关闭数据库连接
%>
```

2.9.3 实现添加购物车的功能

如图 2.5 所示，在某件商品的详细信息页面上单击“添加到购物车”按钮后，该商品将被添加至购物车，页面会跳转至“我的购物车”页面，如图 2.6 所示。下面将介绍添加购物车的实现过程。

在 goodsDetail.jsp 文件中，通过“添加到购物车”按钮的 onclick 属性，调用自定义的 JavaScript 函数 addCart()，用于验证商品数量是否合法。如果不合法，则给出提示，并且返回某件商品的详细信息页面；如果合法，就将该商品添加到购物车。代码如下：

```
<script src="js/jquery.1.3.2.js" type="text/javascript"></script>
<script type="text/javascript">
    function addCart() {
        var num = $('#shuliang').val();                          //获取输入的商品数量
        //验证输入的数量是否合法
        if (num < 1) {                                           //如果输入的数量不合法
            alert('数量不能小于 1！');
            return;
        }
        //调用添加购物车页面，实现将该商品添加到购物车
        window.location.href="cart_add.jsp?goodsID=<%=ID%>&num="+num;
    }
</script>
```

说明

在上面的代码段中，cart_add.jsp 是用于将商品添加到购物车的处理页面，后面的问号“？”用于标识它之后是要传递的参数，多个参数间用“ & ”分隔。

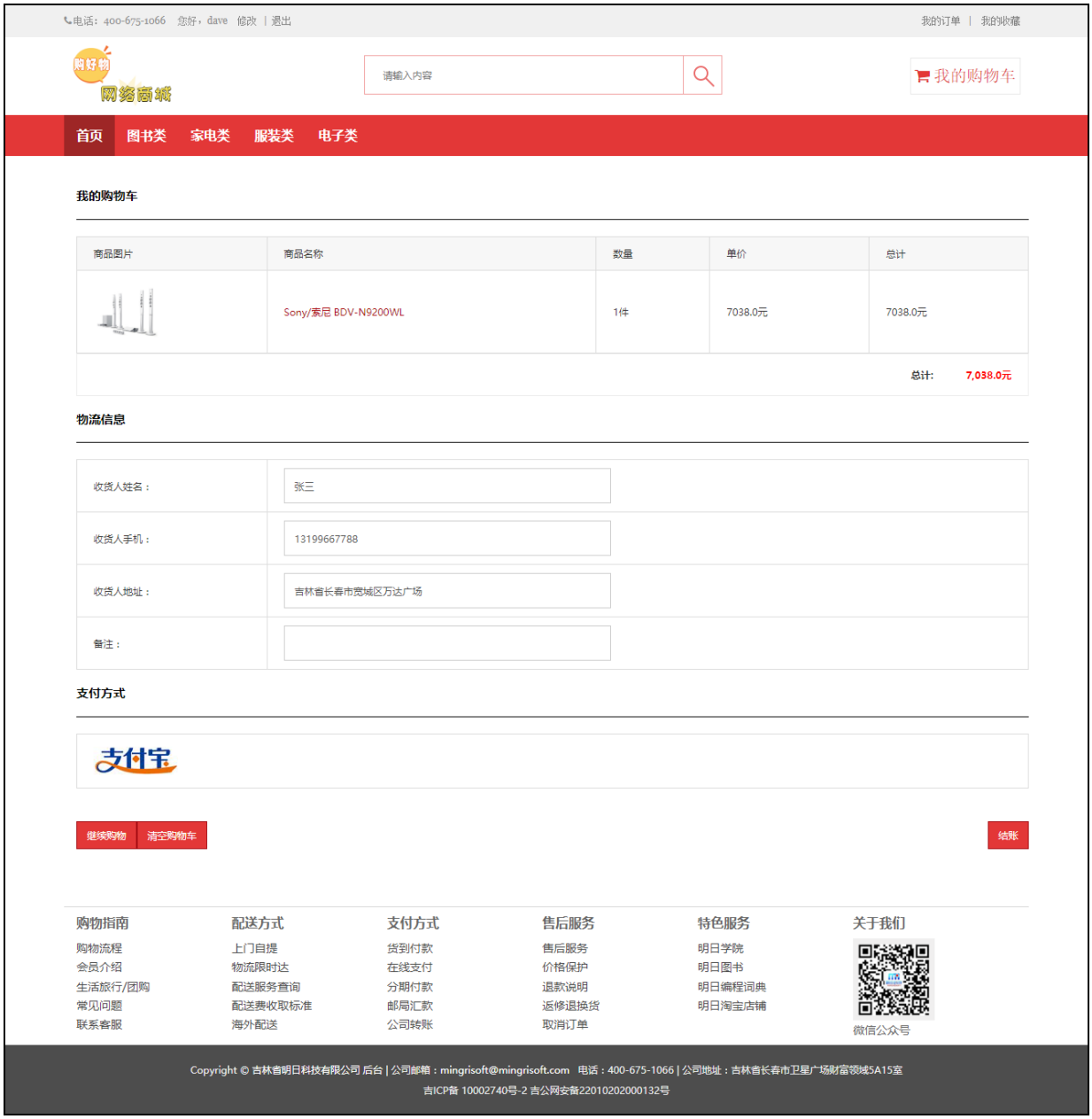


图 2.6 “我的购物车”页面

WebContent/front 下的 cart_add.jsp 文件用于显示“我的购物车”页面。在 goodsDetail.jsp 文件中，导入 java.sql 包中的 ResultSet 类、向量类，以及购物车商品模型类，并且创建 ConnDB 类的对象。

```
<%@ page import="java.sql.ResultSet"%>                    <%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.util.Vector"%>                      <%-- 导入 Java 的向量类 --%>
<%@ page import="com.model.Goodselement"%>                <%-- 导入购物车商品模型类 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB"/>  <%-- 创建 ConnDB 类的对象 --%>
```

在 cart_add.jsp 文件中，为了实现添加到购物车功能，首先获取会员账号和商品量，并判断会员是否登录。如果没有登录，则重定向到会员登录页面进行登录。然后将商品基本信息保存到购物车商品模型类的对象 mygoodselement 中。接着把该商品添加到购物车中。最后使页面跳转到“我的购物车”页面。代码如下：

```
<%
    String username=(String)session.getAttribute("username");        //获取会员账号
    String num = (String) request.getParameter("num");               //获取商品数量
    //如果没有登录，将跳转到登录页面
    if (username == null || username == "") {
        response.sendRedirect("login.jsp");                          //重定向页面到会员登录页面
        return;                                                      //返回
    }
    int ID = Integer.parseInt(request.getParameter("goodsID"));      //获取商品 ID
    String sql = "select * from tb_goods where ID=" + ID;            //定义根据商品 ID 查询商品信息的 SQL 语句
    ResultSet rs = conn.executeQuery(sql);                           //根据商品 ID 查询商品
    float nowprice = 0;                                              //定义保存商品价格的变量
    if (rs.next()) {                                                 //如果查询到指定商品
        nowprice = rs.getFloat("nowprice");                          //获取该商品的价格
    }
    //创建保存购物车内商品信息的模型类的对象 mygoodselement
    Goodselement mygoodselement = new Goodselement();
    mygoodselement.ID = ID;                                          //将商品 ID 保存到 mygoodselement 对象中
    mygoodselement.nowprice = nowprice;                              //将商品价格保存到 mygoodselement 对象中
    mygoodselement.number = Integer.parseInt(num);                   //将购买数量保存到 mygoodselement 对象中
    boolean Flag = true;                                             //记录购物车内是否已经存在所要添加的商品
    Vector cart = (Vector) session.getAttribute("cart");             //获取购物车对象
    if (cart == null) {                                              //如果购物车对象为空
        cart = new Vector();                                         //创建一个购物车对象
    } else {
        //判断购物车内是否已经存在所购买的商品
        for (int i = 0; i < cart.size(); i++) {
            Goodselement goodsitem = (Goodselement) cart.elementAt(i);          //获取购物车内的一个商品
            if (goodsitem.ID == mygoodselement.ID) {                 //如果当前要添加的商品已经在购物车中
                //直接改变购物数量
                goodsitem.number = goodsitem.number + mygoodselement.number;
                cart.setElementAt(goodsitem, i);                     //重新保存到购物车中
                Flag = false;                                        //设置标记变量 Flag 为 false，代表购物车中存在该商品
            }
        }
    }
    if (Flag)                                                        //如果购物车内不存在该商品
        cart.addElement(mygoodselement);                             //将要购买的商品保存到购物车中
    session.setAttribute("cart", cart);                              //将购物车对象添加到 session 中
    conn.close();                                                    //关闭数据库的连接
    response.sendRedirect("cart_see.jsp");                           //重定向页面到查看购物车页面
%>
```

2.9.4 实现查看购物车的功能

如图 2.6 所示，已经被添加到购物车的商品显示在“我的购物车”页面的上部，会员可以查看购物车中商品的图片、名称、数量、单价、总计（价格）等信息。下面将介绍查看购物车功能的实现过程。

WebContent/front 下的 cart_see.jsp 文件用于查看购物车。在 cart_see.jsp 文件中，导入查看购物车所能用到的类（主要包括保存结果集的 ResultSet 类、向量类、格式化数字的类，以及购物车商品模型类），并且创建 ConnDB 类的对象。代码如下：

```
<%@ page import="java.sql.ResultSet"%>                              <%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.util.Vector"%>                                <%-- 导入 Java 的向量类 --%>
<%@ page import="java.text.DecimalFormat"%>                         <%-- 导入格式化数字的类 --%>
<%@ page import="com.model.Goodselement"%>                          <%-- 导入购物车商品的模型类 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />     <%-- 创建 com.tools.ConnDB 类的对象 --%>
```

在 cart_see.jsp 文件中，需要判断当前用户是否登录。如果没有登录，则进入登录页面进行登录；否则，

获取购物车对象。代码如下：

```
<%
    String username = (String) session.getAttribute("username");          //获取会员账号
    //如果没有登录，将跳转到登录页面
    if (username == "" || username == null) {
        response.sendRedirect("login.jsp");                               //重定向页面到“会员登录”页面
        return;                                                           //返回
    } else {
        Vector cart = (Vector) session.getAttribute("cart");              //获取购物车对象
        if (cart == null || cart.size() == 0) {                           //如果购物车为空
            response.sendRedirect("cart_null.jsp");                       //重定向页面到“购物车为空”页面
        } else {
%>
```

在 cart_see.jsp 文件中，使用循环遍历购物车对象，获取需要显示的商品信息。代码如下：

```
<%
    float sum = 0;
    DecimalFormat fnum = new DecimalFormat("#,##0.0");                    //定义显示金额的格式
    int ID = -1;                                                          //保存商品 ID 的变量
    String goodsname = "";                                                //保存商品名称的变量
    String picture = "";                                                  //保存商品图片的变量
    //遍历购物车中的商品
    for (int i = 0; i < cart.size(); i++) {
        Goodselement goodsitem = (Goodselement) cart.elementAt(i);        //获取一个商品
        sum = sum + goodsitem.number * goodsitem.nowprice;                //计算总计金额
        ID = goodsitem.ID;                                                //获取商品 ID
        if (ID > 0) {
            ResultSet rs_goods = conn.executeQuery("select * from tb_goods where ID=" + ID);
            if (rs_goods.next()) {
                goodsname = rs_goods.getString("goodsname");              //获取商品名称
                picture = rs_goods.getString("picture");                  //获取商品图片
            }
            conn.close();                                                 //关闭数据库的连接
    }
%>
```

2.9.5 实现商品订单提交功能

如图 2.6 所示，单击“我的购物车”页面右下角的“结账”按钮，将弹出“支付”对话框，如图 2.7 所示。

图 2.7 “支付”对话框

首先使用支付宝扫描对话框中的二维码进行支付，然后单击对话框右下角的“支付”按钮完成支付操作，程序会提交订单并显示订单号，如图 2.8 所示。下面将介绍商品订单提交的实现过程。

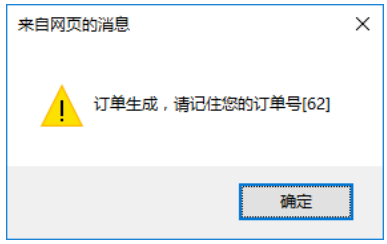


图 2.8　订单生成

用于提交订单的 JSP 文件是 WebContent/front 下的 cart_order.jsp。在 cart_order.jsp 文件中，导入 java.sql 包中的 ResultSet 类、向量类，以及购物车商品模型类，并且创建 ChStr 类和 ConnDB 类的对象。代码如下：

```
<%@ page import="java.sql.ResultSet"%>                    <%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.util.Vector"%>                      <%-- 导入 Java 的向量类 --%>
<%@ page import="com.model.Goodselement"%>                <%-- 导入购物车商品模型类 --%>
<jsp:useBean id="chStr" scope="page" class="com.tools.ChStr" />   <%-- 创建 ChStr 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />   <%-- 创建 ConnDB 类的对象 --%>
```

在 cart_order.jsp 文件中，为了实现提交订单的功能，首先判断购物车是否为空，不为空时，判断会员账户是否合法，只有会员账户合法，才能够提交订单。在提交订单信息时，需要分别向数据库的订单主表和订单明细表插入数据。代码如下：

```
<%
    if (session.getAttribute("cart") == "") {                          //判断购物车对象是否为空
        out.println(
                "<script language='javascript'>alert('您还没有购物!');"
                +"window.location.href='index.jsp';</script>");
    }
    String Username = (String) session.getAttribute("username");       //获取输入的账户名称
    if (Username != "") {
        try {                                                          //捕捉异常
            ResultSet rs_user = conn.executeQuery("select * from tb_Member where username='"
            + Username + "'");
            if (!rs_user.next()) {                          //如果获取的账户名称在会员信息表中不存在（表示非法会员）
                session.invalidate();                       //销毁 session
                out.println(
                        "<script language='javascript'>alert('请先登录后，再进行购物!'); "
                        +"window.location.href='index.jsp';</script>");
                return;                                                //返回
            } else {                                                   //如果会员合法，则提交订单
                //获取输入的收货人姓名
                String recevieName = chStr.chStr(request.getParameter("recevieName"));
                //获取输入的收货人地址
                String address = chStr.chStr(request.getParameter("address"));
                String tel = request.getParameter("tel");              //获取输入的电话号码
                String bz = chStr.chStr(request.getParameter("bz"));   //获取输入的备注
                int orderID = 0;                                       //定义提交订单 ID 的变量
                Vector cart = (Vector) session.getAttribute("cart");   //获取购物车对象
                int number = 0;                                        //定义提交商品数量的变量
                float nowprice = (float) 0.0;                          //定义提交商品价格的变量
                float sum = (float) 0;                                 //定义商品金额的变量
                float Totalsum = (float) 0;                            //定义商品件数的变量
                boolean flag = true;                                   //标记订单是否有效，为 true 表示有效
                int temp = 0;                                          //保存返回自动生成的订单号的变量
                int ID = -1;
```

```
                    //插入订单主表数据
                    float bnumber = cart.size();
                    String sql = "insert into tb_Order(bnumber,username, recevieName,address, "
                            +"tel,bz) values("+ bnumber + ",'" + Username + "','" + recevieName
                            + "','" + address + "','" + tel+ "','" + bz + "')";
                    temp = conn.executeUpdate_id(sql);                          //保存订单主表数据
                    if (temp == 0) {                                            //如果返回的订单号为 0，表示不合法
                        flag = false;
                    } else {
                        orderID = temp;                                         //把生成的订单号赋值给订单 ID 变量
                    }
                    String str = "";                                            //保存插入订单详细信息的 SQL 语句
                    //插入订单明细表数据
                    for (int i = 0; i < cart.size(); i++) {
                        //获取购物车中的一个商品
                        Goodselement mygoodselement = (Goodselement) cart.elementAt(i);
                        ID = mygoodselement.ID;                                 //获取商品 ID
                        nowprice = mygoodselement.nowprice;                     //获取商品价格
                        number = mygoodselement.number;                         //获取商品数量
                        sum = nowprice * number;                                //计算商品金额
                        str = "insert into tb_order_Detail (orderID,goodsID,price,number)"
                                +" values(" + orderID + ","+ ID + "," + nowprice + ","
                                + number + ")";                                 //插入订单明细的 SQL 语句
                        temp = conn.executeUpdate(str);                         //保存订单明细
                        Totalsum = Totalsum + sum;                              //累加合计金额
                        if (temp == 0) {                                        //如果返回值为 0，表示不合法
                            flag = false;
                        }
                    }
                    if (!flag) {                                                //如果订单无效
                        out.println("<script language='javascript'>alert('订单无效');"
                                +"history.back();</script>");
                    } else {
                        session.removeAttribute("cart");                        //清空购物车
                        out.println("<script language='javascript'>alert('订单生成，请记住您"
                                +"的订单号[" + orderID
                        + "]');window.location.href='index.jsp';</script>");    //显示生成的订单号
                    }
                    conn.close();                                               //关闭数据库连接
                }
            } catch (Exception e) {                                             //处理异常
                out.println(e.toString());                                      //输出异常信息
            }
        } else {
            session.invalidate();                                               //销毁 session
            out.println(
                    "<script language='javascript'>alert('请先登录后，再进行购物!');"
                    +"window.location.href='index.jsp';</script>");
        }
%>
```

2.9.6 实现清空购物车功能

会员在登录购好物网络商城并把喜欢的商品添加到购物车后，就可以在“我的购物车”页面上查看购物车中的商品信息了。会员如果想要清空购物车，就单击“我的购物车”页面左下角的“清空购物车”按钮，这时在“我的购物车”页面上显示的商品详细信息就会被删除，如图 2.9 所示。下面将介绍清空购物车的实现过程。

图 2.9　购物车为空页面

WebContent/front 下的 cart_clear.jsp 文件用于清空购物车。在 cart_clear.jsp 文件中，只包含了一个脚本程序。该脚本程序的作用有两个：一个是移除 Session；另一个是将页面跳转到“购物车为空”页面。代码如下：

```
<%
    session.removeAttribute("cart");                    //移除 session
    response.sendRedirect("cart_null.jsp");             //转到购物车为空页面
%>
```

用于显示“购物车为空”页面的 JSP 文件是位于 WebContent/front 下的 cart_null.jsp。如图 2.9 所示，“购物车为空”页面中的内容相对简单：一个是显示“您的购物车为空！”的信息；另一个是显示“继续购物”按钮。代码如下：

```
<!-- MAIN CONTENT -->
<div id="mr-content" class="mr-content col-xs-12">
    <div id="mrshop" class="mrshop common-home">
        <div class="container_oc">
            <div class="container_oc">
                <div class="breadcrumb"></div>
            </div>
            <div class="row">
                <div id="content_oc" class="col-sm-12" style="min-height:300px;">
                    <h1>我的购物车</h1>
                    <div class="table-responsive cart-info" style="margin-bottom:50px;">您的购物车为空！</div>
                    <div class="buttons">
                        <div class="pull-left">
                            <a href="index.jsp" class="btn btn-primary btn-default">继续购物</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

2.9.7　实现继续购物功能

如图 2.6 和 2.9 所示，在“我的购物车”页面和“购物车为空”页面上，都有“继续购物”按钮。不论是在“我的购物车”页面上，还是在“购物车为空”页面上，会员单击“继续购物”按钮后，页面都会跳转至购好物网络商城的首页。因此，不仅这两个“继续购物”按钮的作用是完全相同的（即跳转页面），而且

这两个“继续购物”按钮的实现代码也是完全相同的。代码如下：

```
<div class="pull-left">
        <a href="index.jsp" class="btn btn-primary btn-default">继续购物</a>
</div>
```

2.10 项 目 运 行

通过前述步骤，设计并完成了“购好物网络商城”项目的开发。下面运行本项目，以检验我们的开发成果。如图 2.10 所示，在 IntelliJ IDEA 中，单击快捷图标，即可运行本项目。

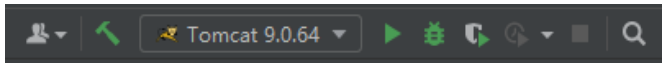


图 2.10 IntelliJ IDEA 的快捷图标

成功运行该项目，会自动打开如图 2.11 所示的“购好物网络商城”首页。在首页上，用户注册或登录会员后，单击首页上的任何商品名称或者商品图片，都会显示该商品的详细信息页面。在某件商品的详细信息页面上单击“添加到购物车”按钮后，该商品将被添加至购物车。用户可以在购物车中查看商品的图片、名称、数量、单价、总计（价格）等信息。用户单击“我的购物车”页面上的“结账”按钮，完成支付操作，同时程序会提交订单并生成订单号。用户单击“我的购物车”页面左下角的“继续购物”按钮，页面会跳转至首页。用户单击“我的购物车”页面左下角的“清空购物车”按钮，购物车中的商品就会被删除。这样，我们就成功地检验了该项目的运行。

图 2.11 成功运行项目后进入首页

Servlet 和 JSP 是 Java Web 开发中的两个重要概念，二者之间的交互主要通过表单、超链接、重定向（redirect）等方式进行。JSP 页面可以通过 HTML 表单收集用户输入的数据，这些数据通过 HTTP 请求发送到 Servlet。Servlet 通过 request.getParameter()方法获取这些参数值，并进行相应的处理。这种方式允许 JSP 页面与用户进行交互，并将数据传递给 Servlet 进行处理。JSP 页面中的超链接可以指向其他 JSP 页面或 Servlet。当用户点击超链接时，会根据链接的目标进行页面跳转。如果链接指向 Servlet，则 Servlet 处理请求并返回响应给客户端；如果链接指向 JSP 页面，则直接加载该 JSP 页面。重定向是通过 response.sendRedirect()方法实现的，它会结束当前请求并发送一个新的请求到指定的 URL。这种方式下，原始请求的属性不会保留在新请求中，适用于需要改变网页地址或引导用户到另一个完全不同的页面的情况。通过上述方式，JSP 和 Servlet 可以有效地进行交互，实现动态 Web 页面的生成和数据处理。

2.11 源 码 下 载

虽然本章详细地讲解了如何编码实现“购好物网络商城”项目的各个功能，但给出的都是代码片段，而非源码。为了方便读者学习，本书提供了完整的项目源码，扫描右侧二维码即可下载。

源码下载