小型网上书店系统的设计与实现

摘　要

伴随着网络的发展，电子商务的不断完善，网上书店是近年来随着网络技术的发展而产生的一种新型的书店形式。与传统书店相比，网上书店拥有许多的优势。网上书店的建立可以大大减少图书销售中的中间环节，节省大量的人力、物力，并且能够提供更多的书目信息。另外，网上购书的读者不会再受地域的限制，而是遍及世界各地，这也就极大限度地扩大了出版物的发行范围。正是由于这些优势，网上书店才能得以在短时间内迅速发展起来。网上书店的兴起，实际上是电子商务在图书业发展的必然结果，它使传统的图书销售业发生了根本性的变革，同时也使传统的购书方式发生了根本性的变化。

本文结合面向对象的分析思想，采用时下流行的 Struts 框架，选择 JAVA 语言作为开发工具，MySQL 作为后台数据库，设计实现了一个小型网上书店系统。系统功能主要包括用户登录、用户信息的添加、修改、网上选书、购书、产生订单等功能。测试表明，本系统的开发基本达到预定目标，具有一定的应用价值。

关键词：网上书店　MySQL　Java　Eclipse　Struts 框架

Design and Implementation of Small Online Bookstore

Abstract

Internet has provided excellent opportunities for development for E-commerce. It gives us enormous economic benefits and the promotion of social productive. E-commerce become new economic growth point. Further development of E—commerce based on Interact is changing and enrich the tradition of enterprise management and operation. In the new environment，the traditional business model cannot meet the requirements of economic globalization. Modem enterprises must have a fast response to customers' demand and real-time business deal with the supply chain capabilities and the ability of collaborative business partners on commerce. So some companies have to carry out e-commerce activities and set up their own e-commerce website. Electronic commerce is entered from only several small enterprises to government agencies. To adapt to the development for new market needs in use of e-business integration enterprise resource and optimize business processes.

In this paper, we combined with object-oriented design thought, use of nowadays popularity Struts framework, Java language as a development tool, MySQL as backend database. We designed and implemented the online bookstore systems. The corresponding functions are realized mainly such as user login, user's information adding, deleting, and picking and buying books, produce order online. Experiment results shows the system basically achieves the predetermined goal and has the certain application value.

Key Words：Online bookstore; MySQL; Java; Eclipse; Struts framework

目　录

1.引言

1.1 课题背景与意义

自从进入互联网时代以来，网络以其前所未有的速度改变着人们的生活方式，改变着人们的价值观念。如果说这是一次经济革命，它比工业革命所带来的影响价值更深入彻底得多，它将传统经济推向了无形的虚拟空间。地球村即真实地体现了经济全球化的趋势。美国未来学家阿尔温·托夫勒曾预言：“电脑网络的建立与普及将彻底改变人类生存及生活的模式。控制与掌握网络的人就是人类未来命运的主宰。”今天，网络的确正在深刻地冲击着人们的生活模式，出版发行业作为人类重要的经济产业更不能漠视甚至回避网络时代的到来。

网络经济模式正以前所未有的迅猛势头席卷我们生活的各个领域：随着电子商务的日益成熟，网上书店应运而生，并以其方便、快捷等一系列优点冲击着传统的图书发行产业，这既是机遇又是挑战。网络时代的今天，谁能更好的利用好这柄双刃剑，必将成为未来图书出版发行业新的主宰者。我国由于网络技术起步较晚，电子商务体系还很不完善，与欧美等国有较大差距。

网上书店作为电子商务网站的一种。网上书店是近年来随着网络技术的发展而产生的一种新型的书店形式。与传统书店相比，网上书店拥有许多的优势。网上书店的建立可以大大减少图书销售中的中间环节，节省大量的人力、物力，并且能够提供更多的书目信息。另外，网上购书的读者不会再受地域的限制，而是遍及世界各地，这也就极大限度地扩大了出版物的发行范围。正是由于这些优势，网上书店才能得以在短时间内迅速发展起来。网上书店的兴起，实际上是电子商务在图书业发展的必然结果，它使传统的图书销售业发生了根本性的变革，同时也使传统的购书方式发生了根本性的变化。

本课题在深入了解了网上购物系统流程的情况下，以 Java，MySQL 为工具，采用面向对象的软件开发方法，采用软件工程的基本步骤进行了系统分析，设计和实现了一个小型的网上书店系统。

1.2 产品介绍

网上书店系统主要是实现网上选书、购书、产生订单等功能的系统。一个典型的网上商城一般都需要实现商品信息的动态提示、购物车管理、客户信息注册登录管理、订单处理等模块。通过本系统软件，能帮助客户利用浏览器快速方便的进行网上购物，而网站管理员则可以方便管理会员信息，书籍入库和进行订单处理，使网上购书方便，快捷，安全。

客户可通过 IE 或其他浏览器浏览书目信息，可通过作者姓名，书名或者 ISBN 号来检索书籍。可以在网上进行注册，成为会员，并且可以随时修改除客户号意外的所有客户信息。系统采用会员制，会员采用唯一的客户标识来标识身份。具有购物车功能，可以增加，修改书的数量，取消已选书籍，等等。能够生成订单，查看当前和以往订单，可通过信息反馈系统跟书店员工进行交流。

后台数据库用 MySQL；系统具备一定的安全性和可靠性。

使用 Hibernate+Struts2.0 框架，具有 SSO（单点登陆）功能。

2. 需求分析与开发工具

2.1 需求分析

表 2-1 需求分析表

功能类别	功能名称、标识符	描述
用户登录	用户登录	系统用户登录系统
用户注册	用户注册	用户注册系统
用户信息管理	个人信息维护	修改个人信息、密码
	账户充值	向账户上充值
	信息反馈	与管理员进行信息交流
管理员功能	查询所有用户联系方式	查询所有用户联系方式
	查询某用户联系方式	查询指定用户联系方式
	删除用户	删除用户
	确认发货	确认订单已发货
	信息反馈	对用户留言进行信息反馈
网上书店系统	购买书籍	购买书籍
	管理购物车	对购车进行管理
	查阅订单	查阅自己订单
安全退出	安全退出	安全退出，清空登陆信息

2.2 开发工具

（1）Eclipse

（2）Tomcat 6.0

（3）MySQL 5.5

3. 系统设计

3.1 系统的结构图

图 3-1 是系统的整体模块结构图，对系统的整体模块进行的一个总的划分。

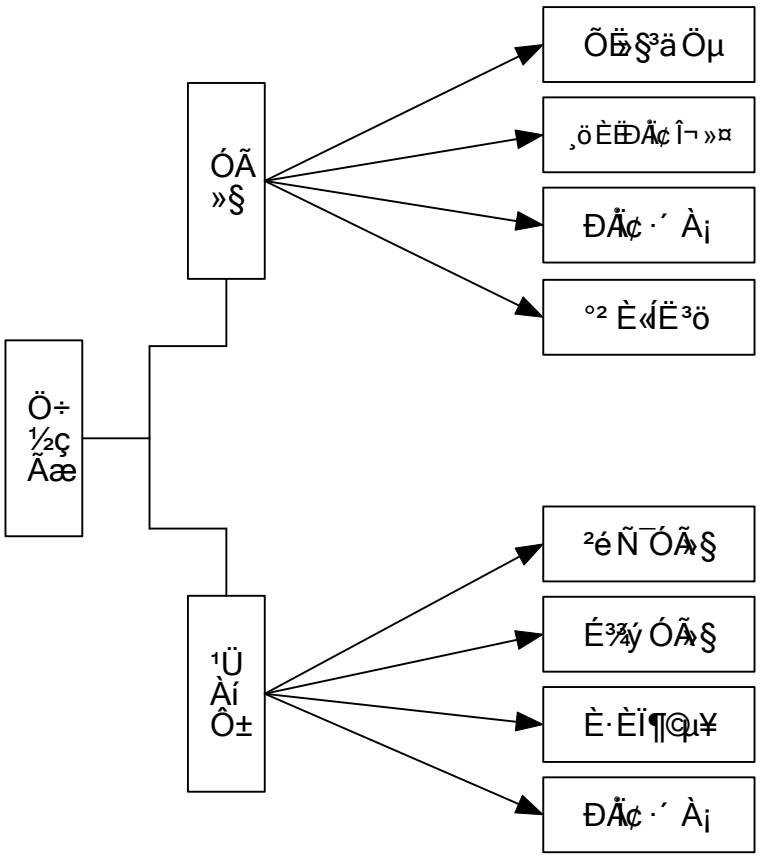

图 3-1 系统总体模块图

图 3-2 是登录模块的功能划分图，对登录模块的功能进行一个详细的划分

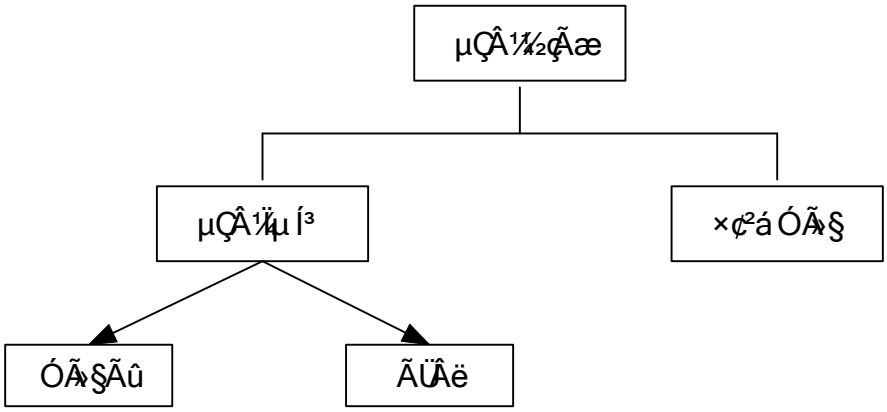

图 3-2 登录模块图

图 3-3 是用户信息管理模块图，对用户信息模块的功能进行一个详细的划分。

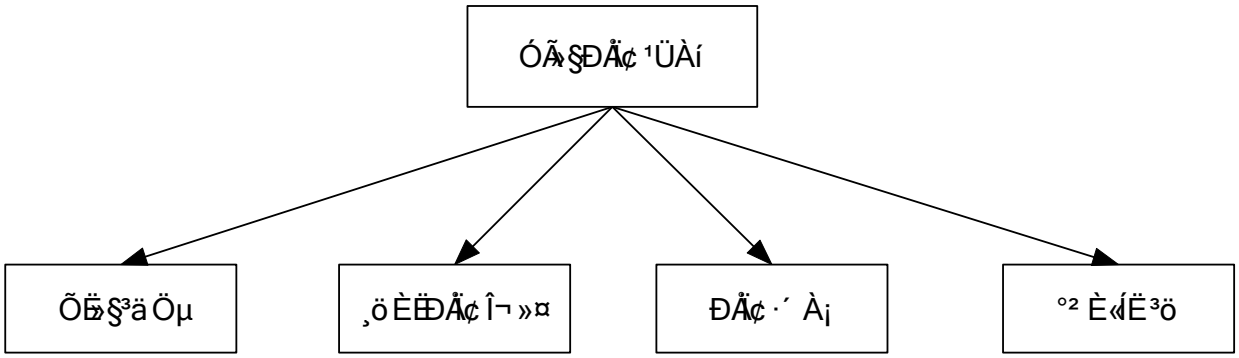

图 3-3 用户信息管理模块图

图 3-4 管理员模块图，对管理员可以操控的功能进行一个详细的划分。

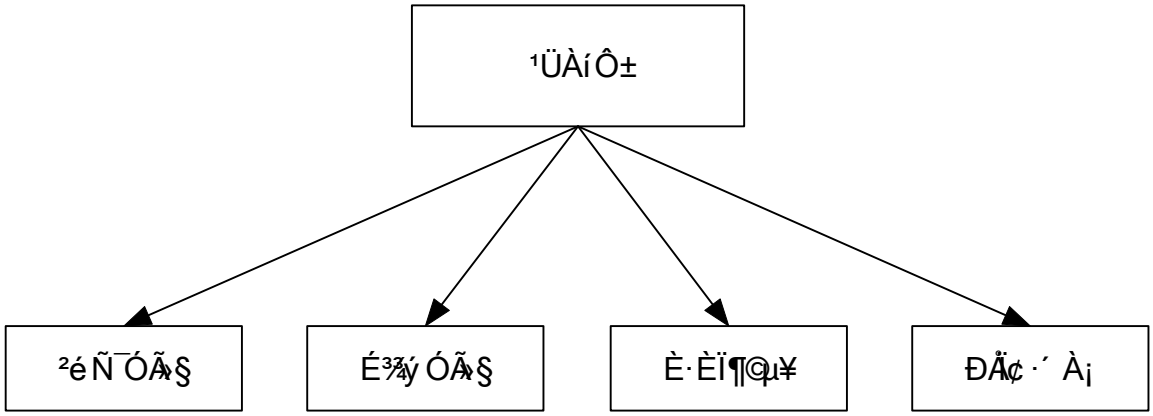

图 3-4 管理员模块图

图 3-5 是书店管理模块图，对书店管理模块的功能进行一个详细的划分。

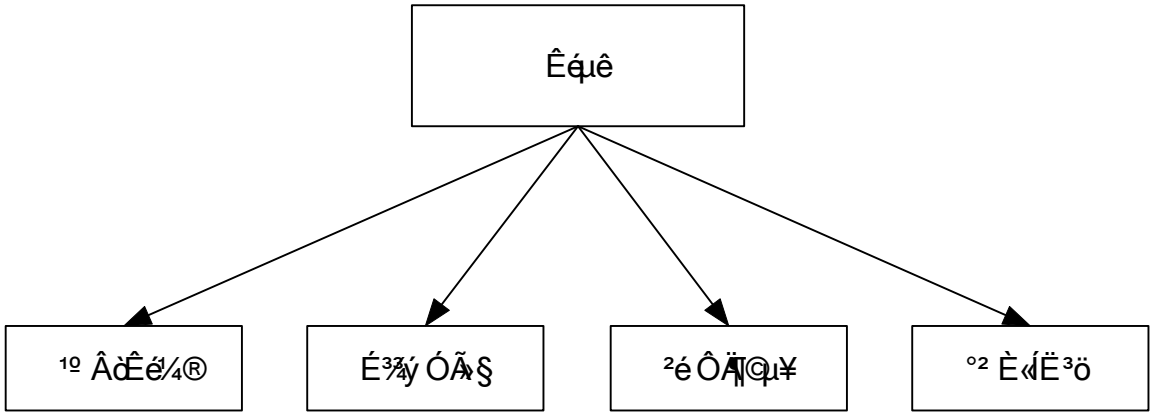

图 3-5 书店管理模块图

3.2 基本功能模块概述

3.2.1 登录模块：

1）模块描述

进入该系统的登陆对话框

2）功能

登陆系统、注册用户

3）性能

登陆：输入用户明和密码，判断用户名和密码是否正确来确定是否登陆系统

注册：注册新的用户

3.2.2 用户信息管理模块：

1）模块描述

用户信息管理

2）功能

主要功能：账户充值，个人信息维护，信息反馈，安全退出，主页面可以显示用户的部分信息，修改信息时也能看到个人信息。

3）性能

账户充值：给用户账户进行充值。

个人信息维护：修改维护个人信息，如密码、联系方式等。

信息反馈：与管理员交流的平台，相当于留言板功能。

3.2.3 管理员模块：

1）模块描述

查询所有用户或者指定用户联系方式、删除用户、信息反馈系统、

2）功能

查询所有用户或者指定用户联系方式、删除用户、信息反馈系统

3）性能

查询所有用户联系方式：查看到所有用户的联系方式。

查询指定用户联系方式：查看到指定用户的联系方式。

删除用户：删除指定用户。不能删除管理员自己。

3.2.4 网上书店模块

1）模块描述

使用网上书店功能。

2）功能

购买书籍，查阅订单，管理购物车，安全退出。

3）性能

购买书籍：查阅书籍列表，将所想要购买的书籍加入购物车。

查阅订单：查阅自己的订单，包括已发货和未发货的。

管理购物车：查阅自己的购物车，可以对购物车中书籍进行数量增减或者删除。

安全退出：安全退出，清空登陆信息。

3.3 数据库设计

MySQL 是一个快速的、多线程、多用户和健壮的 SQL 数据库服务器。MySQL 服务器支持关键任务、重负载生产系统的使用，也可以将它嵌入到一个大配置(mass-deployed)的软件中去。

MySQL 是一个数据库管理系统,一个数据库是一个结构化的数据集合。它可以是从一个简单的销售表到一个美术馆、或者一个社团网络的庞大的信息集合。如果要添加、访问和处理存储在一个计算机数据库中的数据，你就需要一

个像 MySQL 这样的数据库管理系统。从计算机可以很好的处理大量的数据以来，数据库管理系统就在计算机处理中和独立应用程序或其他部分应用程序一样扮演着一个重要的角色。

MySQL 是一个关系数据库管理系统,关系数据库把数据存放在分立的表格中，这比把所有数据存放在一个大仓库中要好得多，这样做将增加你的速度和灵活性。“MySQL”中的 SQL 代表“Structured Query Language”（结构化查询语言）。SQL 是用于访问数据库的最通用的标准语言，它是由 ANSI/ISO 定义的 SQL 标准。SQL 标准发展自 1986 年以来，已经存在多个版本：SQL-86，SQL-92，SQL:1999，SQL:2003，其中 SQL:2003 是该标准的当前版本。

MySQL 是开源的，开源意味着任何人都可以使用和修改该软件，任何人都可以从 Internet 上下载和使用 MySQL 而不需要支付任何费用。如果你愿意，你可以研究其源代码，并根据你的需要修改它。MySQL 使用 GPL(GNU General Public License，通用公共许可)，在 hpt://www.fsf.org/licenses 中定义了你在不同的场合对软件可以或不可以做什么。如果你觉得 GPL 不爽或者想把 MySQL 的源代码集成到一个商业应用中去，你可以向 MySQL AB 购买一个商业许可版本。

MySQL 服务器是一个快的、可靠的和易于使用的数据库服务器如果这是你正在寻找

的，你可以试一试。MySQL 服务器还包含了一个由用户紧密合作开发的实用特性集。你可以在 MySQL AB 的 http://www.mysql.com/it-resources/benchmarks/上找到 MySQL 服务器和其他数据库管理系统的性能比较。

MySQL 服务器原本就是开发比已存在的数据库更快的用于处理大的数据库的解决方案，并且已经成功用于高苛刻生产环境多年。尽管 MySQL 仍在开发中，但它已经提供一个丰富和极其有用的功能集。它的连接性、速度和安全性使 MySQL 非常适合访问在 Internet 上的数据库。

MySQL 服务器工作在客户/服务器或嵌入系统中,MySQL 数据库服务器是一个客户/服务器系统，它由多线程 SQL 服务器组成，支持不同的后端、多个不同的客户程序和库、管理工具和广泛的应用程序接口(APIs)。

MySQL 也可以是一个嵌入的多线程库，你可以把它连接到你的应用中而得

到一个小、快且易于管理的产品。

3.3.1 数据库逻辑设计

图 3-6 是系统的整体 E-R 图，对系统的数据库的逻辑进行设计。

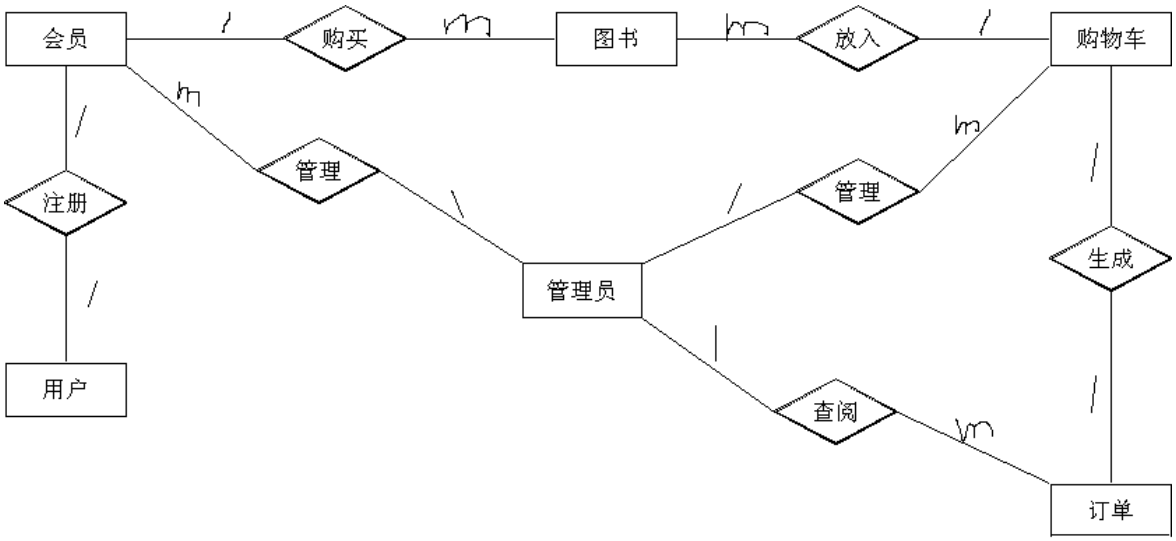

图 3-6 系统整体 E-R 图

3.3.2 数据库物理设计

表 3-1 是用户属性表，对用户的一些属性进行定义。

表 3-1 用户属性表

名	类型	长度	描述
Id	int	8	ID
Name	varchar	20	用户名
Password	varchar	20	密码
admin	varchar	8	是否管理员身份
money	Float	20	账户余额
Tel	varchar	20	联系方式

表 3-2 是购物车属性表，对购物车的一些属性进行定义。

表 3-2 购物车属性表

名	类型	长度	描述
Id	Int	8	ID

Bookname	Varchar	20	书名
Username	Varchar	20	购物车所属用户名
value	Float	20	单价
number	int	20	数量
Totalvalue	Float	20	总价

表 3-3 是订单属性表，对订单的一些属性进行定义。

表 3-3 订单属性表

名	类型	长度	描述
Id	Int	8	ID
Bookname	Varchar	20	书名
Username	Varchar	20	订单所属用户名
value	float	20	单价
number	int	20	数量
totalvalue	float	20	总价
time	varchar	20	订单时间
send	varchar	10	是否发货
Sendtime	Varchar	20	发货时间

表 3-4 是信息反馈表，对信息反馈系统的一些属性进行定义。

表 3-4 信息反馈表

名	类型	长度	描述
Id	Int	8	ID
sub	Varchar	50	信息主题
inf	varchar	200	信息内容
username	varchar	20	留言用户名
inftime	varchar	20	信息留言时间
back	varchar	10	管理员是否反馈
backinf	varchar	200	反馈内容
Backtime	Varchar	20	反馈时间

4. 系统的具体实现

4.1 登录模块

4.1.1 功能分析

这个模块是用来登录系统用的，可以通过该模块进入系统，注册新用户。

4.1.2 关键代码

该段代码是验证用户登录时验证用户信息的，判断是否存在该用户名，并判断该用户的密码是否正确。

```
@SuppressWarnings("serial")
public class Login extends ActionSupport{
    private String username;     //获取用户名
    private String password;     //获取密码
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }

    @SuppressWarnings("unchecked")
    public String login()
    {
        int us=0;     //作为登陆成功与否的判断参数
        try{
            SessionFactory sf= new Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            String sql="from User";   //使用 Hibernate 连接用户数据库
            Query query=ses.createQuery(sql);
```

```
                @SuppressWarnings("rawtypes")
                List list=query.list();
                User usr=new User();
                  for (int i=0;i<list.size();i++)
                  {
                    usr=(User)list.get(i);
                    if
((username.equals(usr.getName()))&&(password.equals(usr.getPassword())))      //判
断验证
                    {
                        @SuppressWarnings("rawtypes")
                        List list1 = new ArrayList();
                        HttpSession                session                =
ServletActionContext.getRequest().getSession();
                        session.setAttribute("role",username);      //为拦截器提供
session 参数
                        session.setAttribute("money",usr.getMoney());
                        list1.add(usr);
                        HttpServletRequest                request                =
ServletActionContext.getRequest();
                        request.setAttribute("list1",list1);     //用 session 传 list1
                        us=1;     //验证通过
                    }
                  }
                ses.close();
                if (us==0)   //验证不通过
                {
                    return "Loginfailed";
                }
            }
            catch(HibernateException e)
            {
                e.printStackTrace();
                System.out.println("服务器出错!");
                return "Failed";
            }
            return "SUCCESS";
        }
    }
```

该代码是用户注册的，要求用户名和密码必须填写。

```
@SuppressWarnings("serial")
public class Reg extends ActionSupport{
    private String username;
```

```
    private String password;
    private String tel;
    private String error;


    public String getTel() {
        return tel;
    }

    public void setTel(String tel) {
        this.tel = tel;
    }

    ……

    public String reg () throws UnsupportedEncodingException
    {
        if((username.equals(""))||(username==null))     //判断用户名是否填写
        {
            error = "未输入用户名";
            return "Regerror";
        }
        else if((password.equals(""))||(password==null))   //判断密码是否填写
        {
            error = "未输入密码";
            return "Regerror";
        }
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();
            //获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from User";   //连接 User 表
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
              User usr=new User();
              User usr1=new User();
              for (int i=0;i<list.size();i++)
            {
                usr1=(User)list.get(i);
                if (username.equals(usr1.getName()))   //判断用户名是否存在
                {
```

```
                    error = "用户名已存在";
                    return "Regerror";
                }
            }
            usr.setName(new      String(username.getBytes("ISO8859-1"),"utf-
8"));
            usr.setPassword(password);
            if(!((tel.equals(""))||(tel==null)))
            {
                usr.setTel(tel);
            }
            else
            {
                tel="未填写";
            }
            usr.setAdmin("no");
            ses.save(usr);
            tx.commit();
        ses.close();
    }
    catch(HibernateException e)
    {
        e.printStackTrace();
        System.out.println("服务器出错!");
        return "Failed";
    }
    return "SUCCESS";
  }
}
```

4.1.3 功能截图

图 4-1　登录界面截图

4.2 用户信息管理模块

4.2.1 功能分析

该模块是用来维护用户信息，可以进行账户充值，个人信息维护，信息反馈，安全退出，主页面可以显示用户的部分信息，修改信息时也能看到个人信息。

4.2.2 关键代码

该代码实现的是个人信息维护的功能。

```
public String modify()
    {
        if
(((newpassword.equals(""))||(newpassword==null))&&((newpasswordtwo.equals(""))|
|(newpasswordtwo==null))&&((newtel.equals(""))||(newtel==null)))
        {
            return "GIVEUP";
        }
        else if((oldpassword==null)||(oldpassword.equals("")))
        {
```

```
            return "Nooldpass";
        }
        else if(!(newpassword.equals(newpasswordtwo)))
        {
            return "Newpassworderror";
        }
        String username;
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();
        //获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from User";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            HttpSession session = ServletActionContext.getRequest().getSession();
            username = (String) session.getAttribute("role");
            User usr=new User();
            for (int i=0;i<list.size();i++)
            {
                usr=(User)list.get(i);
                if (username.equals(usr.getName()))
                {
                    if (oldpassword.equals(usr.getPassword()))
                    {
                        usr.setPassword(newpassword);
                    }
                    else return "Olderror";
                    if (!((newtel.equals(""))||(newtel==null)))
                    {
                        usr.setTel(newtel);
                    }
                    ses.save(usr);
                }
            }
            tx.commit();
            ses.close();
        }
        catch(HibernateException e)
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
```

```
                return "Failed";
            }
            return "SUCCESS";
        }
}
```

4.2.3 功能截图

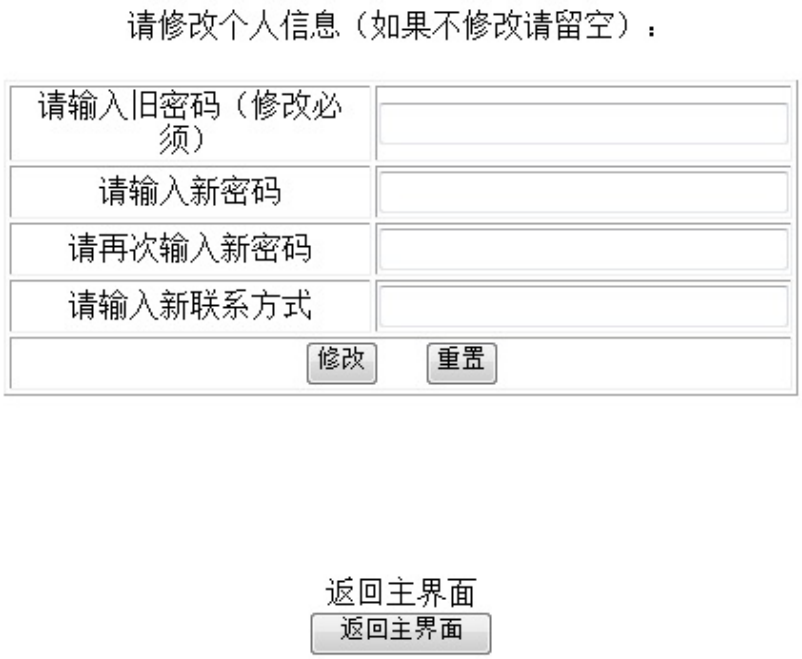

图 4-2　修改个人信息界面截图

4.3 管理员模块

4.3.1 功能分析

这个模块是用来查询所有用户或者指定用户的联系方式、删除用户、信息反馈的系统，由管理员来进行登录和操作。

4.3.2 关键代码

该代码实现的是信息反馈的功能。

```
@SuppressWarnings("unchecked")
    public String backinf2 () throws UnsupportedEncodingException
    {
        if((backinf.equals(""))||(backinf==null))
        {
            error = "未输入内容";
            return "Error";
        }
        try{
            Date now = new Date();
            DateFormat d1 = DateFormat.getDateInstance();
            String time = d1.format(now);
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();
             //获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from Inf";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            @SuppressWarnings("rawtypes")
            List list1 = new ArrayList();
            Inf inf1=new Inf();
            for (int i=0;i<list.size();i++)
            {
                inf1=(Inf)list.get(i);
                if (id==inf1.getId())
                {
                    inf1.setBackinf(new String(backinf.getBytes("ISO8859-
1"),"utf-8"));
                      inf1.setBack("Yes");
                      inf1.setBacktime(time);
                      ses.save(inf1);
                      list1.add(inf1);
                      break;
                }
            }
              HttpServletRequest request = ServletActionContext.getRequest();
             request.setAttribute("list1",list1);
              tx.commit();
            ses.close();
        }
        catch(HibernateException e)
```

```
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
            return "Failed";
        }
        return "SUCCESS";
    }
}
```

该代码实现的是查询所有用户的功能。

```
@SuppressWarnings("serial")
public class Lookall extends ActionSupport {

    public String look()
    {
        try{
          SessionFactory sf= new
Configuration().configure().buildSessionFactory();//获取 hibernate 配置
          Session ses=sf.openSession();//开启连接
          //Transaction tx=ses.beginTransaction();//开启事务
          String sql="from User";
          Query query=ses.createQuery(sql);
          @SuppressWarnings("rawtypes")
          List list=query.list();
            HttpServletRequest request = ServletActionContext.getRequest();
          request.setAttribute("list",list);
          ses.close();
         }
         catch(HibernateException e)
         {
          e.printStackTrace();
          System.out.println("服务器出错!");
            return "Failed";
         }
        return "SUCCESS";
    }
}
```

该代码实现的是删除用户的功能。

```
@SuppressWarnings("serial")
public class Delete extends ActionSupport{
    private String username;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String delete()
    {
        if (username.equals("xy"))
        {
            return "Deleteadminerror";
        }
        int user=0;
        try{
            SessionFactory sf= new Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from User";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            User usr=new User();
            for (int i=0;i<list.size();i++)
            {
                usr=(User)list.get(i);
                if (username.equals(usr.getName()))
                {
                    user=1;
                    ses.delete(usr);
                    break;
                }
            }
            if (user==0)
            {
                tx.commit();
                ses.close();
                return "Nousername";
```

```
            }
            tx.commit();
            ses.close();
        }
        catch(HibernateException e)
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
            return "Failed";
        }
        return "SUCCESS";
    }
}
```

该代码实现的是确认发送订单的功能。

```
@SuppressWarnings("serial")
public class Send extends ActionSupport {

    @SuppressWarnings("unchecked")
    public String send()
    {
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            //Transaction tx=ses.beginTransaction();//开启事务
            String sql="from Orderform";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            @SuppressWarnings("rawtypes")
            List list1=new ArrayList();
            Orderform orderform=new Orderform();
            for (int i=0;i<list.size();i++)
            {
                orderform=(Orderform)list.get(i);
                if (orderform.getSend().equals("No"))
                {
                    list1.add(orderform);
                }
            }
            HttpServletRequest request = ServletActionContext.getRequest();
            request.setAttribute("list1",list1);
```

```
            ses.close();
        }
        catch(HibernateException e)
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
            return "Failed";
        }
        return "SUCCESS";
    }
}
```

4.3.3 功能截图

图 4-3　管理员删除用户 信息界面截图

4.4 网上书店模块

4.4.1 功能分析

这个模块是网上书店，可以实现用户购买书籍，查阅订单，管理购物车，安全退出等功能。

4.4.2 关键代码

该代码实现的是购买《ACM 程序设计竞赛基础教程》这本书的功能。

```
@SuppressWarnings("serial")
public class Cart1 extends ActionSupport{
    private int number;

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    @SuppressWarnings("unchecked")
    public String cart1() throws UnsupportedEncodingException
    {
        String username;
        String bookname="ACM 程序设计竞赛基础教程";
        int us=0;
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from Shoppingcart";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            @SuppressWarnings("rawtypes")
            List list1 = new ArrayList();
            HttpSession session = ServletActionContext.getRequest().getSession();
            username = (String) session.getAttribute("role");
            Shoppingcart shoppingcart=new Shoppingcart();
```

```
            for (int i=0;i<list.size();i++)
            {
                shoppingcart=(Shoppingcart)list.get(i);
                if
((username.equals(shoppingcart.getUsername()))&&(bookname.equals(shoppingcart.
getBookname())))
                {
                    shoppingcart.setNumber(shoppingcart.getNumber()+number);

    shoppingcart.setTotalvalue(shoppingcart.getTotalvalue()+number*25);

                    ses.save(shoppingcart);
                    us=1;
                    break;
                }
                if (username.equals(shoppingcart.getUsername()))
                {
                    list1.add(shoppingcart);
                }
            }
            if (us==0)
            {
                Shoppingcart shoppingcart1=new Shoppingcart();
                shoppingcart1.setBookname(new      String(bookname.getBytes("utf-
8"),"utf-8"));
                shoppingcart1.setUsername(new       String(username.getBytes("utf-
8"),"utf-8"));
                shoppingcart1.setNumber(number);
                shoppingcart1.setValue(25);
                shoppingcart1.setTotalvalue(25*number);
                ses.save(shoppingcart1);
                list1.add(shoppingcart1);
            }
            HttpServletRequest request = ServletActionContext.getRequest();
             request.setAttribute("list1",list1);
            tx.commit();
            ses.close();
        }
        catch(HibernateException e)
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
            return "Failed";
        }
```

```
            return "SUCCESS";
        }
}
```

该代码实现的是增加购物车中某书籍数量的功能。

```
@SuppressWarnings("serial")
public class Nummodify extends ActionSupport{
    private int id;
    private int number;
    private String success;

    public String getSuccess() {
        return success;
    }

    public void setSuccess(String success) {
        this.success = success;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public String nummodify()
    {
        int us=0;
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
```

```
                String sql="from Shoppingcart";
                Query query=ses.createQuery(sql);
                @SuppressWarnings("rawtypes")
                List list=query.list();
                Shoppingcart shoppingcart=new Shoppingcart();
                for (int i=0;i<list.size();i++)
                {
                    shoppingcart=(Shoppingcart)list.get(i);
                    if (shoppingcart.getId()==id)
                    {
                        shoppingcart.setNumber(shoppingcart.getNumber()+number);

    shoppingcart.setTotalvalue(shoppingcart.getTotalvalue()+number*shoppingcart.g
etValue());
                        ses.save(shoppingcart);
                        us=1;
                        break;
                    }
                }
                if (us==0)
                {
                    tx.commit();
                    ses.close();
                    return "Failed";
                }
                tx.commit();
                ses.close();
            }
            catch(HibernateException e)
            {
                e.printStackTrace();
                System.out.println("服务器出错!");
                return "Failed";
            }
            success="增加数量成功！";
            return "SUCCESS";
        }
}
```

该代码实现的是查阅订单的功能。

```
@SuppressWarnings("serial")
public class Shoppingcart1 extends ActionSupport{
```

```
    @SuppressWarnings("unchecked")
    public String shoppingcart1() throws UnsupportedEncodingException
    {
        String username;
        try{
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();
//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from Shoppingcart";
            Query query=ses.createQuery(sql);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            @SuppressWarnings("rawtypes")
            List list1 = new ArrayList();
            HttpSession session = ServletActionContext.getRequest().getSession();
            username = (String) session.getAttribute("role");
            Shoppingcart shoppingcart=new Shoppingcart();
            for (int i=0;i<list.size();i++)
            {
                shoppingcart=(Shoppingcart)list.get(i);
                if (username.equals(shoppingcart.getUsername()))
                {
                    list1.add(shoppingcart);
                }
            }
            HttpServletRequest request = ServletActionContext.getRequest();
             request.setAttribute("list1",list1);
            tx.commit();
            ses.close();
        }
        catch(HibernateException e)
        {
            e.printStackTrace();
            System.out.println("服务器出错!");
            return "Failed";
        }
        return "SUCCESS";
    }
}
```

该代码实现的是提交订单的功能。

```
public String submit1() throws UnsupportedEncodingException
    {
        int us=0;
        String username;
        String send="未发货";
        float money;
        try{
            HttpSession session = ServletActionContext.getRequest().getSession();
             username = (String) session.getAttribute("role");
             money = (Float) session.getAttribute("money");
            SessionFactory sf= new
Configuration().configure().buildSessionFactory();//获取 hibernate 配置
            Session ses=sf.openSession();//开启连接
            Transaction tx=ses.beginTransaction();//开启事务
            String sql="from Shoppingcart";
            String sql1="from User";
            Query query=ses.createQuery(sql);
            Query query1=ses.createQuery(sql1);
            @SuppressWarnings("rawtypes")
            List list=query.list();
            @SuppressWarnings("rawtypes")
            List list1=query1.list();
            User user = new User();
            for (int j=0;j<list1.size();j++)
            {
                user=(User)list1.get(j);
                if (username.equals(user.getName()))
                {
                    break;
                }
            }
            for (int i=0;i<list.size();i++)
            {
                Shoppingcart shoppingcart=new Shoppingcart();
                shoppingcart=(Shoppingcart)list.get(i);
                if
((shoppingcart.getId()==id)&&(shoppingcart.getUsername().equals(username)))
                {
                    if (money<shoppingcart.getTotalvalue())
                    {
                        error = "账户余额不足!";
                        tx.commit();
                        ses.close();
```

```
                    return "Error";
                }
                Date now = new Date();
                DateFormat d1 = DateFormat.getDateInstance();
                String time = d1.format(now);
                Orderform orderform = new Orderform();
                orderform.setBookname(shoppingcart.getBookname());
                orderform.setNumber(shoppingcart.getNumber());
                orderform.setUsername(new
String(shoppingcart.getUsername().getBytes("utf-8"),"utf-8"));
                orderform.setValue(shoppingcart.getValue());
                orderform.setTotalvalue(shoppingcart.getTotalvalue());
                orderform.setSend("No");
                orderform.setTime(time);
                orderform.setSendtime(new   String(send.getBytes("utf-8"),"utf-
8"));
                user.setMoney(user.getMoney()-shoppingcart.getTotalvalue());
                session.setAttribute("money",user.getMoney());
                ses.save(user);
                ses.save(orderform);
                ses.delete(shoppingcart);
                us=1;
                break;
            }
        }
        if (us==0)
        {
            tx.commit();
            ses.close();
            error = "该书籍代码不存在！";
            return "Error";
        }
        tx.commit();
        ses.close();
    }
    catch(HibernateException e)
    {
        e.printStackTrace();
        System.out.println("服务器出错!");
        return "Failed";
    }
    success="该订单提交成功！";
    return "SUCCESS";
}
```

}

4.4.3 功能截图

这是您的所有订单，请查阅！

用户名：xy	书名：Athena	单价：24.0	数量：1	总价：24.0	订单时间：2008-07-02	是否发送：yes	发送时间：2008-07-30
用户名：xy	书名：Popular	单价：11.0	数量：2	总价：22.0	订单时间：2010-06-14	是否发送：yes	发送时间：2010-07-01
用户名：xy	书名：Hello,friends	单价：37.0	数量：1	总价：37.0	订单时间：2010-10-09	是否发送：yes	发送时间：2010-10-27
用户名：xy	书名：网络规划设计师教程	单价：96.0	数量：4	总价：384.0	订单时间：2011-5-7	是否发送：yes	发送时间：2011-5-7
用户名：xy	书名：ACM程序设计竞赛基础教程	单价：25.0	数量：3	总价：75.0	订单时间：2011-5-7	是否发送：No	发送时间：未发货

返回主界面

返回主界面

图 4-5　订单界面截图

5. 结语

毕业设计现在已经接近尾声了，回想起来，感觉收获颇多。我这次毕业设计的任务是开发一个小型的网上书店，它用到的开发工具是 Eclipse+Tomcat，采用的数据库是 MySQL，经过几个月的学习和实践，网上书店系统基本开发完成，其功能符合系统用户的基本需求，实现了用户注册，登录，基本信息的修改，后台管理等功能。系统使用简洁明快的界面风格设计，具备了友好性，灵活性和可靠性，实现了预期目标和功能。但鉴于毕业设计时间短，本人基础差，再则毕业实习占用了大量时间，系统还有许多不尽如人意的地方。数据库设计过程中表的结构和字段的设计还不够优化，因而本系统的运行效率不是很高。这也是系统需要完善的地方之一。

通过毕业设计，让我感受了软件开发的整个过程。毕业设计不仅是对我在大学所学知识的一个综合运用，也是一次增长知识和经验的好机会，同时也使我学会了许多处理、解决问题的方法，大大提高了自己的动手能力，为即将正式走上工作岗位打下了良好的基础。

参考文献

[1] （美）霍斯特曼，科奈尔.Java2 核心技术卷 1：基础知识. 机械工业出版社. 2008.

[2] （美）布朗森（Bronson,G.J.）著. Java 程序设计基础. 北京大学出版社. 2005.

[3] 孙卫琴编著. Java 面向对象编程. 电子工业出版社. 2006.

[4] 唐汉明. 深入浅出 MySQL 数据库开发优化与管理维护. 人民邮电出版社. 2008.

[5] （日）青野雅树编. 基于 Java 的计算机图形学. 科学出版社. 2004.

[6] 于雨编著. Java 语言程序设计. 辽宁科学技术出版社. 2006.

[7] 王健，张金波主编. Java 程序设计实训教程. 海洋出版社. 2005.

[8] 孙卫琴编著. Java 面向对象编程. 电子工业出版社. 2006.

[9] （英）福塔 著，刘晓霞，钟鸣 译. MySQL 必知必会. 人民邮电出版社. 2009.

[10] （美）贝尔 著，杨涛 等译. 深入理解 MySQL. 人民邮电出版社. 2010.

[11] 江义华, 林彩瑜. Java 完美经典(M). 中国铁道出版社. 2004.

[12] 林建素, 孟康健. Eclipse 开发学习笔记(M). 电子工业出版社. 2008.