第3章 在线图书销售管理系统

本章作为课程设计的案例，使用 ASP.NET 环境中的脚本语言 C# 设计一个图书销售管理信息系统的简单应用程序。本例涉及的功能比较少，有兴趣的读者可以在此基础上自行设计，增加一些其他的功能。

3.1 需求分析

近几年来，计算机和网络技术有了快速的发展和应用，商业销售方式从传统的店铺经营逐步发展到网络经营，顾客购买方式也从店铺购买逐步发展到网上购物。在线图书销售管理系统也随着网上购物的浪潮应运而生。

3.1.1 系统现状

在线图书销售管理系统对于网上图书的销售管理和图书购买是非常重要的。现在许多商业销售部门都有自己的销售管理系统。用户可以在因特网上查询自己所需要的购物信息，足不出户就可以了解各方面的信息，进行网上交易，再通过物流公司就可以达到远程购物的目的。通过远程登录图书销售管理系统查询出自己所需要的图书的详细信息并提交购买信息，这样既方便用户，同时也方便了销售人员的销售管理。

在线图书销售管理系统是因特网上常见的销售管理系统，它的一个基本作用就是为图书销售部门提供发布所销售图书信息的平台。使用 ASP.NET 的 Web 开发平台能够生成企业级 Web 应用程序所需的服务，提供一种新的编程模型和结构，用于生成更安全、可伸缩和稳定的应用程序，而使用 SQL Server 数据库将减轻管理人员的工作量，使系统便于维护和管理。

3.1.2 用户需求

对于图书销售企业来说，利用现代计算机网络和通信技术、数据库技术实现供应、销售等相关业务管理、共享数据资源，使业务办理过程网络化、电子化，这样能够进一步畅通销售管理、有效地实现快速销售、大大提高工作效率。

在线图书销售管理系统使用因特网的优势实现在线图书销售管理，主要实现会员注册、会员信息管理、图书信息管理、订单信息管理等功能。

3.2 系统功能分析

根据图书销售的基本要求，本系统的功能分为管理员、普通用户和会员 3 类。管理员负责系统维护；普通用户只具有浏览网站的权限；会员则可以实现购买功能。为了使问题简单

化，本课程设计只讨论系统管理员和会员两类用户。

3.2.1 系统功能概述

根据在线图书销售管理系统的需求，本系统主要完成以下功能。

- 注册功能：该功能是为了让普通用户成为会员而设立的。
- 会员登录功能：会员登录后才可以实现通过购物车购买图书的功能。
- 购物车功能：若会员对某本图书感兴趣，可以将该图书放入自己的购物车，和超市中的购物车一样，目的是方便记载会员购买的商品信息。
- 图书信息查找功能：用户可以直接搜索所需的图书信息，当图书信息数量很多时该功能对用户来说是非常方便的。
- 个人中心：方便会员查看和修改个人信息。
- 图书信息分类列表：图书一般会有很多种，为了分门别类而使得这项功能非常有用。当用户需要某种类型的图书时只需要使用该功能就可以看到所有属于该类的图书信息。
- 订单查询功能：该功能是为了方便查询会员的所有订单情况，从而及时地将订单上的货物邮寄给会员。
- 添加修改图书信息：该功能是为了对网站图书信息进行维护而设立的。

根据不同的用户需求，本章所介绍的在线销售管理系统主要完成以下两个功能区。

1. 用户功能区

根据需求，用户可以完成以下操作：

- 用户进行注册；
- 用户浏览图书信息；
- 用户查找图书信息；
- 用户选择购买图书信息；
- 用户提交购买图书订单信息；
- 用户修改个人资料信息；
- 用户填写意见信息。

2. 管理员功能区

- 管理员浏览用户购买图书信息；
- 管理员添加新图书信息；
- 管理员修改、删除图书信息；
- 管理员浏览用户意见信息；
- 管理员核查购买图书费用信息。

3.2.2 系统功能模块设计

在线图书销售管理系统的各功能模块如图 2.3.1 所示。

(1) 用户注册模块：此模块要求购买图书者必须首先进行会员注册，成为本系统的合法用户。用户在注册模块中主要完成登录账号、登录密码、信用卡账号、信用卡密码、姓名、身份证号、性别、家庭地址、联系电话和手机号等初始信息的填写。

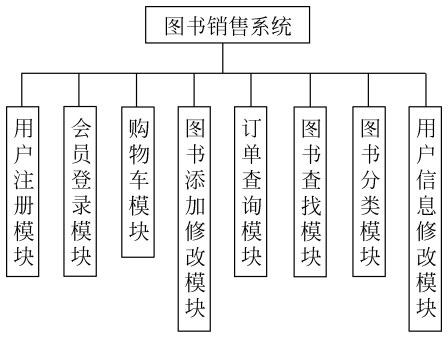

图 2.3.1　系统功能模块图

(2) 会员登录模块：此模块包括会员登录和检查会员登录信息功能，主要负责根据用户所输入的登录账号和登录密码判断该用户是否合法。

(3) 购物车模块：此模块的功能是将会员的购书信息放入购物车中，其中包括购物车编号、书名、每种书的数量、购买日期、每种书的总价、图书单价、国际标准书号、电子邮箱(会员账号)。

(4) 图书添加修改模块：此模块的功能是系统管理员在后台对新图书信息进行添加、对图书信息进行修改和对旧图书信息进行删除。

(5) 订单查询模块：此模块的功能是管理员通过查看会员的订单了解会员的购书信息，从而及时地将图书邮寄给相应会员。

(6) 图书查找模块：此模块的功能是用户通过访问图书信息表快速地查询到自己感兴趣的图书信息。

(7) 图书分类模块：此模块的功能是用户按分类查询图书信息表中的图书信息，例如“人文社科类”“自然科学类”“艺术美育类”等类图书信息。

(8) 用户信息修改模块：此模块的功能是会员登录系统后修改自己的信息。

该系统的主要功能如下：

- 管理员管理功能：负责整个系统的后台管理。
- 管理员添加、修改和删除图书信息功能。
- 会员查询指定图书信息功能。
- 会员购买图书信息的提交功能。
- 管理员/会员退出系统功能。

该系统主要分为以下两大功能模块：

1) 前台系统功能模块

前台系统功能模块主要是涉及会员操作，会员负责整个系统的前台操作，如图 2.3.2 所示。

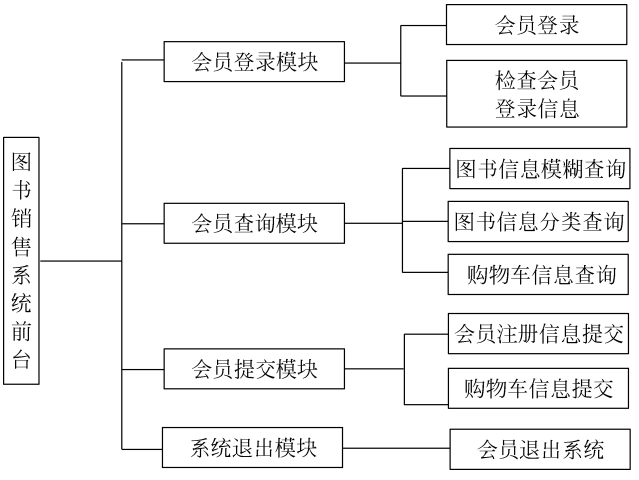

图 2.3.2　前台系统功能模块图

2）后台系统功能模块

后台系统功能模块主要涉及操作员操作，管理员负责整个系统的后台管理，如图 2.3.3 所示。

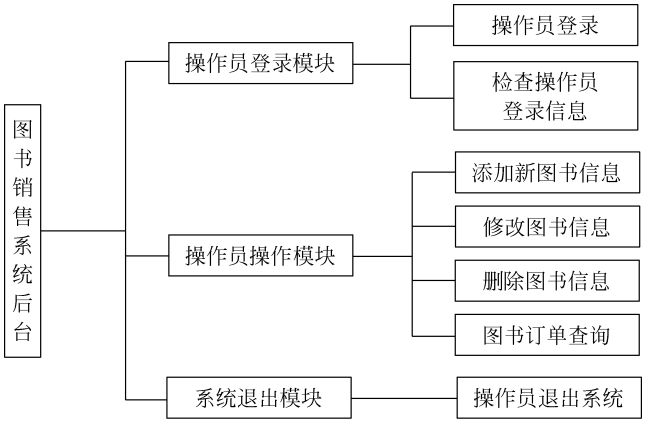

图 2.3.3　后台系统功能模块图

3.3　系统总体设计

系统总体设计是指关于对象系统的总体机能以及和其他系统相关的方面的设计，包括基本环境要求、用户界面的基本要求等。

3.3.1　总体系统流程图

通过会员的前台操作和管理员的后台操作来完成在线图书销售管理系统的总体结构流程。总体系统流程如图 2.3.4 所示。

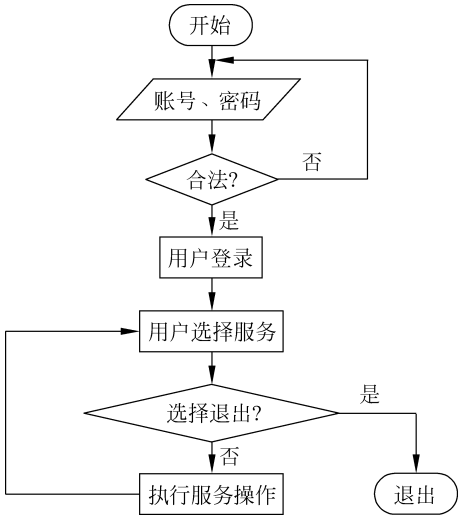

图 2.3.4　总体系统流程图

3.3.2 前台系统结构

会员前台操作主要完成用户登录、浏览图书信息、购买图书的流程信息,其结构如图 2.3.5 所示。

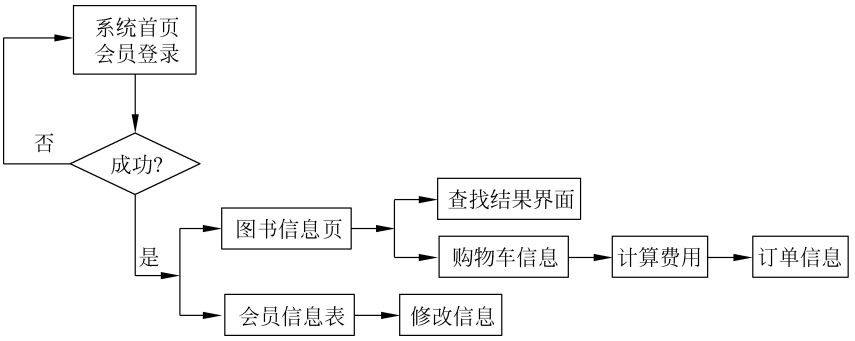

图 2.3.5 前台系统结构图

3.3.3 后台系统结构

管理员后台操作主要完成管理员登录、添加新图书信息、删除旧图书信息、查询订单信息和查看意见箱信息。其结构如图 2.3.6 所示。

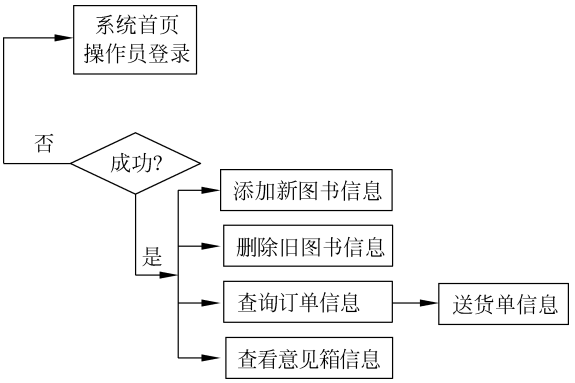

图 2.3.6 后台系统结构图

3.4 数据库设计

数据库设计是指根据用户的需求在某一具体的数据库管理系统上设计数据库的结构和建立数据库的过程。

3.4.1 数据库的概念设计

根据概念结构设计的步骤先进行局部概念设计,然后对各个局部概念进行综合。

1. 局部概念设计

确定系统的局部概念设计范围,为讨论简单,这里只给出各个实体的局部 E-R 模型,如图 2.3.7 所示。

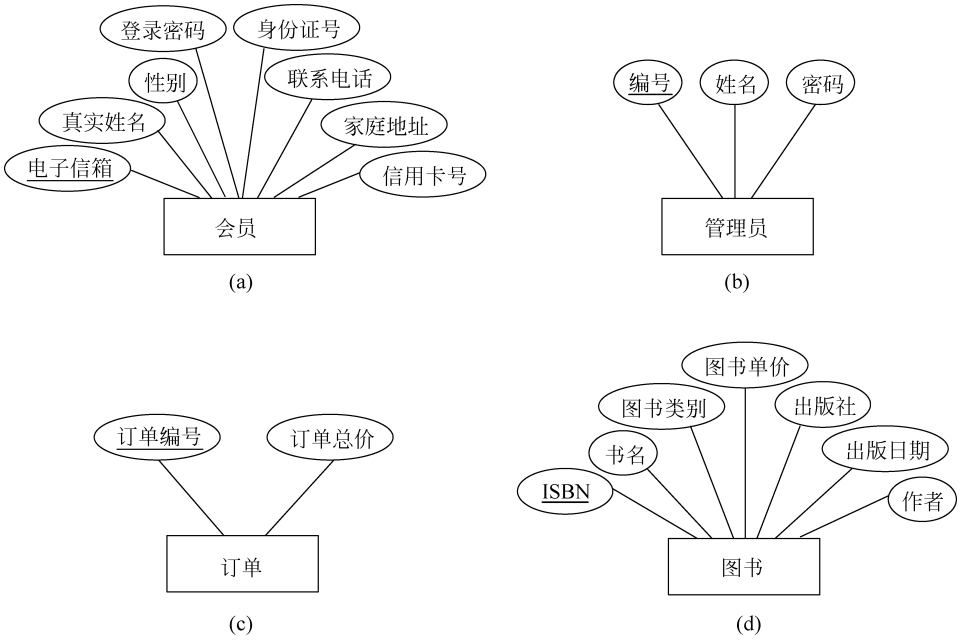

图 2.3.7　各个实体的局部 E-R 模型

2. 全局概念结构设计

综合各实体的局部 E-R 模型图形成如图 2.3.8 所示的全局 E-R 图。

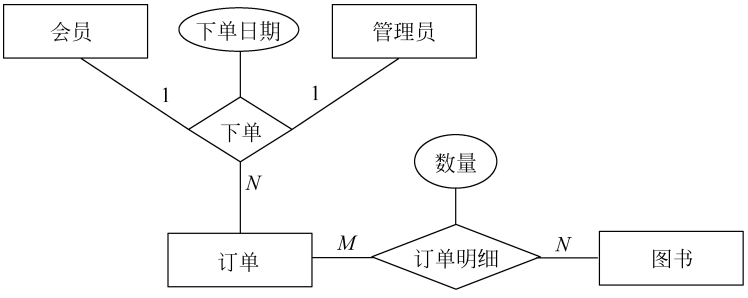

图 2.3.8　全局 E-R 模型图

3.4.2　数据库的逻辑设计

数据库的逻辑设计就是将概念设计阶段设计的 E-R 模型转化为关系模式，可分为下面两个步骤进行。

1. 将实体转化为关系模式

会员关系模式：会员(电子邮箱，真实姓名，性别，登录密码，身份证号，联系电话，家庭地址，信用卡号)

管理员关系模式：管理员(编号，姓名，密码)

订单关系模式：订单(订单编号，下单日期，订单总价)

图书关系模式：图书(ISBN，图书名，图书类别，图书单价，出版社，出版日期，作者)

2. 将联系转化为关系模式

在概念设计阶段共设计两个联系，一个是下单联系，它是一个 1∶1∶M 的三元联系，可以

将其放到 N 端实体转化为的关系模式上,另一个为订单与图书之间的 M ∶ N 的联系,必须将它转化为一个新的关系模式,结果如下。

订单关系模式:订单(订单编号,下单日期,订单总价,电子邮箱,管理员编号)

订单明细关系模式:订单明细(订单编号,图书编号,数量)

3.4.3 数据库的物理设计

1. 会员表(Member)

会员信息包括电子邮箱、真实姓名、性别、登录密码、身份证号、联系电话、家庭地址、信用卡号,如表 2.3.1 所示。

表 2.3.1 会员信息表

字 段 名	字段描述	字段类型	备 注
Email	电子邮箱	varchar(50)	主键
TrueName	真实姓名	varchar(20)	—
Sex	性别	char(2)	—
Password	登录密码	varchar(20)	—
IDNumber	身份证号	varchar(20)	—
Telephone	联系电话	varchar(15)	—
Address	家庭地址	varchar(50)	—
CreditCard	信用卡号	varchar(50)	—

2. 管理员表(Administrator)

管理员信息包括编号、姓名、密码,如表 2.3.2 所示。

表 2.3.2 管理员信息表

字 段 名	字段描述	字段类型	备 注
AdminNo	编号	varchar(20)	主键
Name	姓名	varchar(20)	—
Password	密码	varchar(20)	—

3. 图书表(Book)

图书信息包括 ISBN、图书名、图书类别、图书单价、出版社、出版日期、作者,如表 2.3.3 所示。

表 2.3.3 图书信息表

字 段 名	字段描述	字段类型	备 注
ISBN	ISBN	varchar(50)	主键
BookName	图书名	varchar(50)	—
BookType	图书类别	varchar(20)	—
BookPrice	图书单价	float	—
Publisher	出版社	varchar(50)	—
PublishDate	出版日期	datetime	—
Author	作者	varchar(20)	—

4. 订单表(Order)

订单信息包括订单编号、下单日期、电子邮箱、管理员编号、订单总价,如表 2.3.4 所示。

表 2.3.4　订单信息表

字　段　名	字 段 描 述	字 段 类 型	备　　注
OrderID	订单编号	int	主键,标识,从 1000 开始
OrderDate	下单日期	datetime	—
Email	电子邮箱	varchar(50)	外键,标识客户
AdminNo	管理员编号	varchar(20)	外键,标识管理员
OrderTotal	订单总价	float	—

5. 订单明细表(OrderDetail)

订单明细信息包括订单明细编号、订单编号、ISBN、数量,如表 2.3.5 所示。

表 2.3.5　订单明细信息表

字　段　名	字 段 描 述	字 段 类 型	备　　注
OrderDetailID	订单明细编号	int	主键,标识,从 1 开始
OrderID	订单编号	int	外键
ISBN	ISBN	varchar(50)	外键
Amount	数量	int	—

3.5　应用程序设计

使用应用程序设计用户界面和访问数据库,用户界面是用户控制与使用系统的工具和手段,友好、易用的界面有助于用户对数据库中数据的操作。

3.5.1　系统设计总体思路

该系统采用多层结构实现,所有数据访问层代码放在 DataAccess 目录下,所有业务层代码放在 Business 目录下,所有表示层放在 UI 目录下。

该系统的页面设计采用层叠样式表(CSS),在本系统中所有页面共同调用一个 CSS 文件,该文件放在 CSS 目录下,文件名为 Style.css。

```
.bolder{  font-weight: bolder;}
.red{  color: #ff0000;}
.left{  text-align: left;}
.center{  text-align: center;}
.right{  text-align: right;}
.header{  background-color: #eeeeee;  height: 30px;}
/*Table*/
table.t01{  width: 800px;  border: 1px solid #a0a0a0;background-color: #dfe8f7;  border-
collapse: collapse;}
table.t02{  width: 400px;  border: 1px solid #a0a0a0;background-color: #dfe8f7;border-
collapse: collapse;}
/*TD*/
td{  padding: 3px;  border: 1px solid #a0a0a0;}
td.td100{  width: 100px;  padding: 3px;  border: 1px solid #a0a0a0;}
td.td300{  width: 300px;  padding: 3px;  border: 1px solid #a0a0a0;}
td.td03{  width: 30%;  text-align: right;  padding: 3px;  border: 1px solid #a0a0a0;}
td.td07{  width: 70%;  text-align: left;  padding: 3px;  border: 1px solid #a0a0a0;}
input.bu01{height: 24px;width: 75px;  text-align: center;}
input.in01{border: #ffffff outset;font-size: 12px;width: 98%;border-width: 0px 0px 1px 0px;
  background-color: #dfe8f7;text-align: left;}
```

```
input.in02{border: #ffffff outset;font - size: 12px;  width: 200px;  border - width: 0px 0px
1px 0px;  background - color: #dfe8f7;text - align: left;}
A:link{color: #0000ff;  border: 0;  text - decoration: none;text - align: left;}
A:visited{color: #0000ff;border: 0;text - decoration: none;text - align: left;}
A:active{  color: #ff0000;border: 0;  text - decoration: none;text - align: left;}
A:hover{  color: #ff0000;border: 0;  text - decoration: none;text - align: left;}
A.a01:link{color: #0000ff;border: 0;  text - decoration: none;text - align: left;}
A.a01:visited{  color: #0000ff;border: 0;  text - decoration: none;text - align: left;}
A.a01:active{  color: #ff0000;border: 0;  text - decoration: none;text - align: left;}
A.a01:hover{  color: #ff0000;border: 0;  text - decoration: none;text - align: left;}
p.p01{margin: 4 0 8 0;  text - align: center;}
```

该系统中多次在页面中弹出对话框,在 ASP.NET 中未提供这个功能,因此扩展了 Page 类,使 Page 具有弹出对话框的功能。该扩展类放在 Util 目录下,代码如下:

```
namespace BookSales.Util
{
    public static class PageExtensions
    {
        ///< summary >
        ///服务器端弹出 alert 对话框
        ///</summary >
        ///< param name = "str_Message">提示信息,例子: "请输入您的姓名!"</param >
        ///< param name = "page"> Page 类</param >
        public static void Alert(this Page page, string str_Message)
        {
            page.ClientScript.RegisterStartupScript(page.GetType(), "", "< script > alert('" +
str_Message + "');</script >");
        }

        ///< summary >
        ///服务器端弹出 alert 对话框
        ///</summary >
        ///< param name = "str_Message">提示信息,例子: "请输入您的姓名!"</param >
        ///< param name = "str_CtlNameOrPageUrl">获得焦点控件 Id 值,例如 txt_Name,或者将要跳转
的页面</param >
        ///< param name = "page"> Page 类</param >
           public static void Alert(this Page page, string str_Message, string str_
CtlNameOrPageUrl)
        {
            if (str_CtlNameOrPageUrl.IndexOf(".") >= 0)
            {
        //如果 str_CtlNameOrPageUrl 里有"."说明为地址
        page.ClientScript.RegisterStartupScript(page.GetType(), "", "< script > alert('" + str_
Message + "');self.location = '" + str_CtlNameOrPageUrl + "';</script >");
            }
            else
            {
        page.ClientScript.RegisterStartupScript(page.GetType(), "", "< script > alert('" + str_
Message + "');document.forms(0)." + str_CtlNameOrPageUrl + ".focus(); document.forms(0)." +
str_CtlNameOrPageUrl + ".select();</script >");
            }
        }
            }
        }
```

该系统提供访问数据库的通用类放在 DataAccess 目录下的 SqlHelper.cs 文件中。

```
public class SqlHelper
{
```

```
    static string strConn;
    static SqlHelper()
    {
       strConn = System.Configuration.ConfigurationManager.ConnectionStrings["strConn"].
ConnectionString;
    }
    ///<summary>
    ///执行更新语句
    ///</summary>
    ///<param name="strSql"></param>
    ///<returns></returns>
    public static void ExecuteNonQuery(string strSql)
    {
        SqlConnection objConn = new SqlConnection(strConn);
        SqlCommand objCmd = new SqlCommand(strSql, objConn);
        try
        {
            objConn.Open();
            objCmd.ExecuteNonQuery();
        }
        catch (Exception e)
        {
            throw e;
        }
        finally
        {
            objCmd.Dispose();
            objCmd = null;
            objConn.Close();
            objConn = null;
        }
    }
    ///<summary>
    ///查找单个数据
    ///</summary>
    ///<param name="strSql"></param>
    ///<returns></returns>
    public static object ExecuteScalar(string strSql)
    {
        object ret = null;
        SqlConnection objConn = new SqlConnection(strConn);
        SqlCommand objCmd = new SqlCommand(strSql, objConn);
        try
        {
            objConn.Open();
                    ret = objCmd.ExecuteScalar();
        }
        catch (Exception e)
        {
        throw e;
        }
        finally
        {
            objCmd.Dispose();
            objCmd = null;
            objConn.Close();
            objConn = null;
        }
        return ret;
    }
```