|  |  |  |  |
| --- | --- | --- | --- |
| **学生学号** |  | **成绩** |  |



管理信息系统实训A训练报告书

**实 训 名 称** **管理信息系统实训A**

**开 课 学 院 管理学院**

**指 导 教 师 姓 名**

**学 生 姓 名**

**学 生 专 业 班 级 信管1502 \_**

**2017-2018学年 第 2 学期**

目录

[1 绪论 1](#_Toc519243769)

[1.1 系统开发背景 1](#_Toc519243770)

[1.2 课题研究内容 1](#_Toc519243771)

[1.3 论文组织结构 2](#_Toc519243772)

[2 相关技术介绍 3](#_Toc519243773)

[2.1 C/S模式 3](#_Toc519243776)

[2.2 强大的C#开发语言 3](#_Toc519243777)

[2.3 Visual studio开发工具 4](#_Toc519243778)

[2.4 SQL server数据库管理系统 4](#_Toc519243779)

[2.5 本章小结 5](#_Toc519243780)

[3 系统需求分析 6](#_Toc519243781)

[3.1 项目背景 6](#_Toc519243783)

[3.2 需求分析 6](#_Toc519243784)

[3.2.1 项目目标 6](#_Toc519243785)

[3.2.2 关键涉众及待解决问题 6](#_Toc519243786)

[3.2.3 项目成功标准 6](#_Toc519243787)

[3.3 业务分析 6](#_Toc519243788)

[3.3.1 分析业务流程 7](#_Toc519243789)

[3.4 本章小结 9](#_Toc519243790)

[4 系统设计 10](#_Toc519243791)

[4.1 系统设计原则 10](#_Toc519243796)

[4.2 系统技术架构设计 10](#_Toc519243797)

[4.3 系统功能结构设计 11](#_Toc519243798)

[4.3.1 模块划分原则 11](#_Toc519243799)

[4.3.2 功能模块划分 11](#_Toc519243800)

[4.4 系统数据库设计 12](#_Toc519243801)

[4.4.1 概念结构设计 12](#_Toc519243802)

[4.4.2 逻辑结构设计 14](#_Toc519243803)

[4.5 本章小结 18](#_Toc519243804)

[5 系统实现 19](#_Toc519243805)

[5.1 顾客登陆浏览模块的实现 19](#_Toc519243807)

[5.1.1 界面实现 19](#_Toc519243808)

[5.1.2 代码实现 20](#_Toc519243809)

[5.2 购物车模块设计 35](#_Toc519243810)

[5.2.1 界面实现 36](#_Toc519243811)

[5.2.2 代码实现 36](#_Toc519243812)

[5.3 结算模块设计 43](#_Toc519243813)

[5.3.1 修改收货地址 43](#_Toc519243814)

[5.3.2 代码实现 44](#_Toc519243815)

[5.4 后台管理模块设计 50](#_Toc519243816)

[5.4.1 订单管理 50](#_Toc519243817)

[5.4.2 会员管理 51](#_Toc519243818)

[5.4.3 商品管理 52](#_Toc519243819)

[5.4.4 公告管理 53](#_Toc519243820)

[5.5 本章总结 55](#_Toc519243821)

[6 参考文献 56](#_Toc519243822)

# 绪论

## 系统开发背景

如今，互联网已经无处不在，无论是人们的购物，出行，还是消费等日常活动都大量依赖于网络，网店就应运而生，对商家来说，网店可以免去一大笔的实体店的装修费用，省去一大笔成本，对消费者来说就意味着可以用更低的价格买到心仪的产品。

但是，在网店管理的过程中，有大量的信息，如用户信息，商品信息，还有各种促销活动的打折信息，这些信息部分是在一定的时间段内保持不变的，而另一部分则是时时变动的。针对这些海量的数据，传统人工的方式已经不能够满足商品管理消息及时更替的要求。同时，现在是一个人工智能的时代，利用用户已有的消费记录来预测用户接下来可能会产生的购物需求，可以提高商家的销售量。

现在的互联网技术已经十分成熟，计算机的算力已经非常强大，数据存储技术也越来越完善，因此在硬件都已经充分准备好的情况下，开发一个在线购物的网站系统是一个非常必要的且有意义的任务。

## 课题研究内容

本课题旨在开发一个在线购物商城，以方便买家购物、卖家发布商品，管理促销活动，管理用户信息，并生成对用户的个性化建议。

本课程设计的主要内容包括:

1. 系统功能的设计，包括系统的功能结构和业务流程。
2. 基于对在线购物流程的分析，详细讲解数据库的设计过程和原理。在讨论的全部过程当中，从技术的角度出发，详细介绍C/S模式、C#开发平台、Visual Studio2017 开发工具以及Sql server数据库在购物系统中的详细使用。
3. 对主要的各个功能模块进行剖析，讲解，分析各个模块的设计原则，实现方法以及功能需求。从买方和卖方对系统的不同要求出发，以最小的代价，最大的功能来满足网站的需求，切切实实方便卖家的商品发布，用户信息管理需求，同时为买家提供便捷的购物体验，最终为买卖双方带来实际的效率和效益。
4. 本系统所涉及到的知识点以及在开发过程中所得到的心得体会。

## 论文组织结构

本文总共分为六个部分。

第一章绪论。通过大量文献的阅读，对目前互联网的应用趋势做了总结，指出了传统购物方式在信息管理方面存在的一些问题，指明了当前计算机算力提升对解决购物信息的一个有效的管理方式。

第二章开发技术介绍。对在线购物平台的开发技术进行介绍和说明，主要包括C/S模式、C#开发语言、Visual Studio 2017开发工具以及SQL server数据库管理系统。

第三章系统需求分析。将需求分析划分为项目背景、需求分析、业务分析和非功能性需求四个步骤，通过各种UML图由浅入深、全面透彻地对在线购物商城进行需求描述。

第四章是系统设计。根据需求分析将系统划分为若干模块，完成数据库设计。

第五章是系统代码的实现。对本系统所有功能模块的代码实现进行较详细的补充，给出具体的操作方式。

# 相关技术介绍



## C/S模式

C/S模式其实也可以称作C/S结构，C和S分别代表Client和Server，这种模式是20世纪80年代末慢慢出现和成长的一种模式，其突出的优势是开发较为简便。

本在线购物系统在C/S模式下可以划分为两层结构，分别是客户端、服务端。

1. 客户端：客户端提供着用户与系统的交互功能，负责接收用户通过软件提交的业务请求，在对业务请求进行简单的处理后，将请求发送给服务器，最后将数据处理的结果反馈给终端用户。如果从用例的角度分析，客户端就扮演着“边界”的角色。一般来说，软件与数据库服务器可以进行双向通信。
2. 数据库服务器：数据库服务器主要承担系统数据的存储和管理业务，根据“边界”传入的数据操作指令，完成对数据库的增删改查等各类操作，并将数据处理结果回传给软件控制台。

## 强大的C#开发语言

C#语言是微软开发的一套面向对象的编程语言，他与早先所接触到的C语言及后续发展语言有很大的不同：面向对象的开发而不是面向过程的开发。面向对象的编程风格具有高效的特点，因为有大量前人已经写好的类，方法可以直接调用，开发者不需要再自己造轮子，不要再去做一些前人已经做过的工作，并且面向对象的继承，多态的特点使代码具有很高的可重用性，价值有微软这一个庞大的公司的支撑，利用C#开发，可以说是站在巨人的肩膀上了。

C#语言从本身的命名可以猜测其收到了C语言的影响，事实上也确实如此，C#属于C语言这个大家族。微软在设计这个语言的时候，其目的之一是与Oracle公司的Java语言抗衡，因此C#参考了大量的Java的设计规则，并在此基础上做了个性化的定制。在面向对象语言面， “指针” 这一功能被受到了严格的限制（当然C#中也是有指针存在的），C#中的 “指针” 只能够指向已知的变量，当它没有指着任何一个东西的时候，就会意味着它被回收。

最后，面向对象语言基于封装的特性，加之C#语言背靠微软这个庞大的帝国，C#开发平台有大量的封装好的了方法可以供开发者直接调用。放在不同的命名空间中，借助Visual Studio这个强大的编译器，只要输入了“using”这个关键字，编译器就能够自动的去找相应的包并提示下载。只要开发者合理有效的使用这些写好的包，就能够快速高效的完成一个庞大的项目。

## Visual studio开发工具

Microsoft Visual Studio（简称VS）是美国微软公司的开发工具包系列产品。VS是一个基本完整的开发工具集，它包括了整个软件生命周期中所需要的大部分工具，如UML工具、代码管控工具、集成开发环境 (IDE) 等等。所写的目标代码适用于微软支持的所有平台，包括Microsoft Windows、Windows Mobile、Windows CE、.NET Framework、.NET Compact Framework和Microsoft Silverlight 及Windows Phone。

Visual Studio是目前最流行的Windows平台应用程序的集成开发环境。最新版本为 Visual Studio 2017 版本，基于.NET Framework 4.5.2。

## SQL server数据库管理系统

SQL server数据库是一个微软开发的关系型数据库系统，使用结构化查询语言（SQL）进行着数据库的增删改查等数据库管理，被企业广泛使用，其排名仅次于Oracle输入据库。

SQL server功能非常强大，安装它的用户们可以是大型的企业，也可以是刚刚开始学习SQL server的学生。并且，微软为开发人员提供了免费的教育版，非常方便

1. SQL server数据库提供了可视化操作界面，开发者可以通过界面来对数据库进行操作了，这对初学者提供了非常大的便利。
2. 真正的客户机/服务器体系结构。
3. 图形化用户界面，使系统管理和数据库管理更加直观、简单。
4. 丰富的编程接口工具，为用户进行程序设计提供了更大的选择余地。
5. SQL Server与Windows NT完全集成，利用了NT的许多功能，如发送和接受消息，管理登录安全性等。SQL Server也可以很好地与Microsoft BackOffice产品集成。
6. 具有很好的伸缩性，可跨越从运行Windows 95/98的小型电脑到运行Windows 2000的大型多处理器等多种平台使用。
7. 对Web技术的支持，使用户能够很容易地将数据库中的数据发布到Web页面上。
8. SQL Server提供数据仓库功能，这个功能只在Oracle和其他更昂贵的DBMS中才有。

## 本章小结

本章对在线购物商城开发技术进行概括性的介绍，包括有C/S模式、C#开发语言、Visual studio2017开发环境，SQL server数据库管理系统，以及本系统的功能和旨在解决的问题。

# 系统需求分析



## 项目背景

购物网站已经和人们的日常生活密不可分，只要有网络和相关设备就可以随时选购自己需要的商品，并且可以享受商品送货上门的服务。购物网站为买卖双方交易提供了便捷的平台，卖家可以在网站上发布器想出售的商品的信息，买家可以从中选择并购买自己需要的物品。本系统使用基于C#语言的asp.net和SQL server技术开发一在线网站购物系统。

## 需求分析

### 项目目标

本项目希望搭建一个在线购物的网上平台，买家可以浏览商品，添加商品到购物车，购买商品；卖家可以发布自己的商品，更新自己的商品信息，管理用户的信息。

### 关键涉众及待解决问题

通过项目目标的分析，关键的涉众可以很清晰的分为两类：买家，买家，如表 3-1所示。

表 3-1 系统涉众分析

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 涉众 | 涉众说明 | 期望 |
| Administrator | 卖家 | 发布商品，制定价格，发布促销活动，管理用户订单信息； | 能够预测可能将来会对什么物品产生购买的欲望，精准推送 |
| Customer | 买家 | 购物；选择心仪的商品， | 更加高效的找到自己想购买的商品 |

### 项目成功标准

满足关键涉众的要求，尽量实现涉众群体的期望，完成基本的在线购物平台的实现。

## 业务分析

### 分析业务流程

流程图着重描述处理过程，它的主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系。

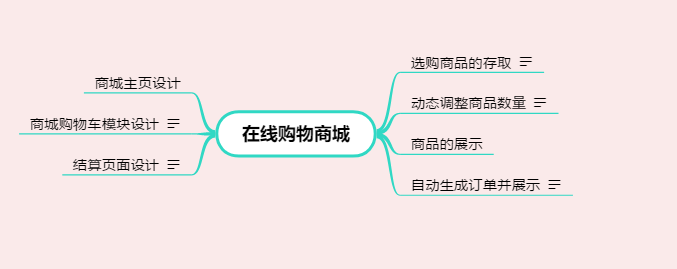
除了流程图，还有一种在面向对象开发过程中经常使用的图：活动图。活动图用来描述业务用例如何实现的运行流程，业务流程可以涵盖执行任务的主角和他们进行的一系列操作，为主角生成一个可以使用的工件。

活动图与流程图有着一些非常显著的区别：

1. 活动图是面向对象的，而流程图是面向过程的。
2. 流程图着重描述处理过程，它的主要控制结构是顺序、分支和循环，各个处理过程之间有严格的顺序和时间关系；活动图描述的是对象活动的顺序关系所遵循的规则，它着重表现的是系统内各个角色的行为，而非系统的处理过程

为了更加有效的展示本项目的开发过程，本报告既使用流程图，又使用活动图。

本系统的开发细节如下图。

图3.3.1-1—开发细节图

业务流程图



图3.3.1-2-业务流程图

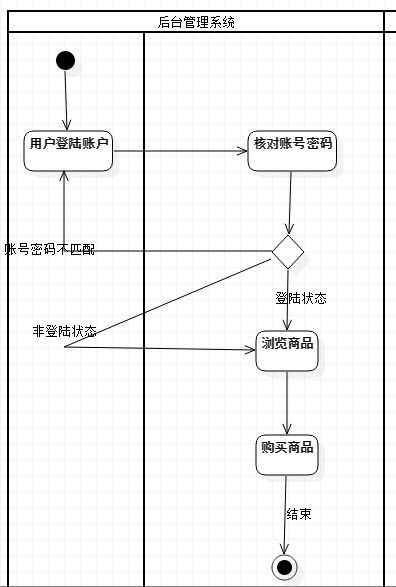


图3.3.1-2-顾客活动图

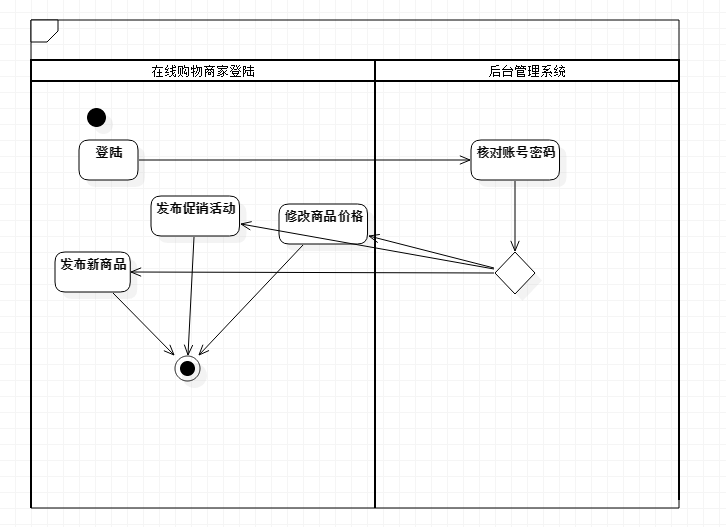


图3.3.1-2-管理员活动图

## 本章小结

本系统在设计的时候参考了现在流行的购物网站的设计，此部分主要对本在线购物系统做了详实的分析，分别从项目背景、需求分析、业务分析入手，结合了流程图、UML活动图尽量尽量详尽的阐述分析了系统的功能需求，设计需求，为后面的数据库设计，代码实现提供了理论指导。

# 系统设计



## 系统设计原则

以下是本在线购物商城设计原则的简要介绍：

1. 简单易用原则

本系统旨在为在线购物的顾客提供良好便捷的购物体验，为卖家提供简单实用的发布产品体验，因此在人机交互上，始终秉持简洁，易用，友好的原则，力图让使用者学习使用系统的时候零成本。

1. 稳定可靠原则

稳定可靠是网站运行的一个前提保障，数据终于一切，本系统使用了冗余备份原则，结束强大的SQL server数据库保证数据库的安全行，及时出现了数据故障，也能够在能够接受的时间内恢复。

1. 开销最小原则

在开发的过程中，不能一味的追求性能的堆砌而忽视了成本，本系统在开发的过程中，时时考虑开销最小原则，比如在保证数据安全的情况下减少数据库的冗余，在保证代码稳定的情况下最大化重用代码，本系统处处考虑了系统的开销，使得网站的性能最优化。。

## 系统技术架构设计

本系统的技术架构图如图 4-1.1所示。



图 4‑1.1 系统架构图

## 系统功能结构设计

### 模块划分原则

从开发者的角度来看，用户们多样的需求转化为计算机可以理解的编程语言，是我们进行系统功能结构设计的原因。对于某个功能模块的划分就是对用户整体的功能性需求进行分解的过程——最后的结果，是划分成为多个容易实现的小部分。即满足“高内聚，低藕合”的设计思想。本系统的模块划分原则如下：

1. 功能模块划分时要从逻辑关系着手。在分析的过程中，要将类似或对立的逻辑处理功能放在一个子系统或模块当中。
2. 功能模块划分时要从时间关系着手。如果可以在同一时间上实现的功能，就限定使用该模块，而不考虑到用其他的功能模块来实现，以而提高内聚性。
3. 功能模块划分时要从职能关系权限划分入手。在线购物系统非常强调不同职能的权限——普通客户是绝对不能发布产品的，而管理员也不能向用户的购物车里面添加商品。

### 功能模块划分

结合需求实际，划分结果如图 4-2所示。

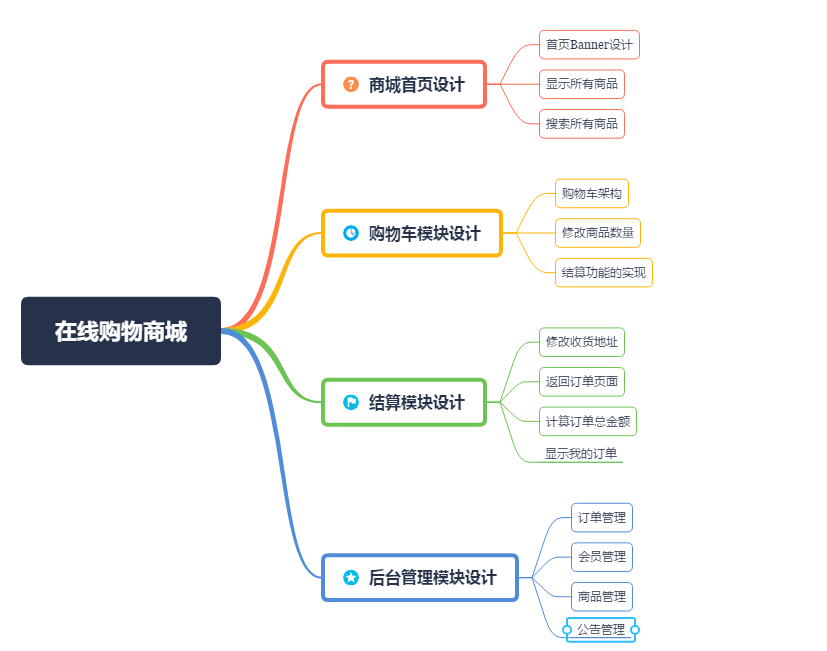


图 4‑2.1 系统功能模板划分

## 系统数据库设计

### 概念结构设计

概念结构设计是整个数据库设计的关键，我们可以依此对用户需求进行综合、归纳与抽象，形成具体的DBMS的概念模型。在线购物商城系统E-R模型如图 4-3所示，各实体属性图如图 4-4至图 4-7所示。



图 4.4.1-1 在线购物商城E-R模型



图 4.4.1-2 管理员实体属性图



图 4.4.1-3 评论实体属性图



图 4.4.1-4 用户实体属性图

### 逻辑结构设计

将概念设计阶段的E-R图转换成关系模式，设计出“在线购物系统”的逻辑结构。

#### 关系设计

地址（ID，地址描述，快递电话，快递姓名）

活动（ID，活动名称，内容，时间，评论数）

商品（ID，种类ID，商品名称，商品简介，价格，现价，上次购买时间，是否为新品）

管理员（ID，姓名，密码）

顾客（ID，用户名，真实姓名，所在城市，地址，邮编，证件号码，，电话号码，邮箱）

订单（ID，用户名，支付方式，邮递方式，订单产生时间，是否加急，其他）

订单详情（ID,订单ID，商品ID，价格，数量）

子分类（ID，子类ID，子类名称）

将关系模式转换为具体的关系数据模型（表结构）。本系统在SQL server数据库管理系统中共设计8张表，分别是用户地址表，商城活动表，商品表，管理员表，顾客表，订单表，订单详情表，子分类表，表结构如所示。

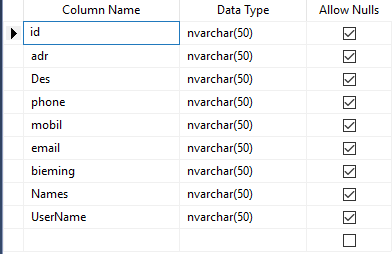


图 4.4.1-5 用户地址表

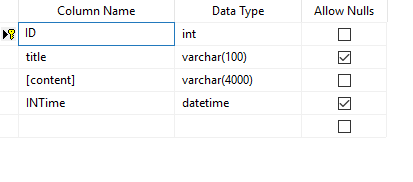


图 4.4.1-6 商城活动表

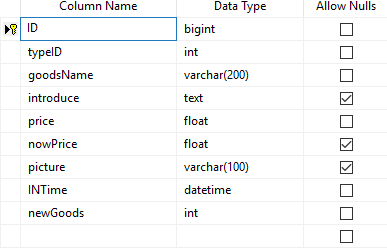


图4.4.1-7商品表

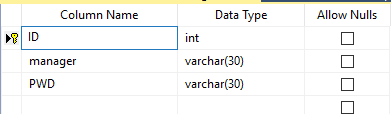


图 4.4.1-8 管理员表

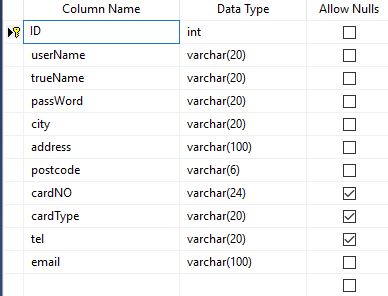


图 4.4.1-9 顾客表

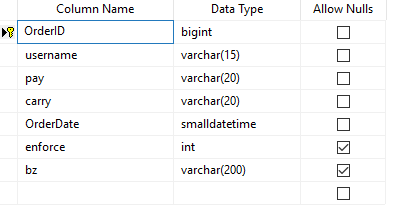


图 4.4.1-10 订单表

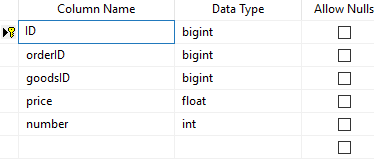


图 4.4.1-11 订单详情表

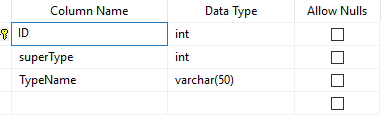


图 4.4.1-12 子分类表

## 本章小结

本章对在线购物商城数据库的设计做出了非常详细的分析与阐述。从功能模块划分、数据库的逻辑设计，到结合表格、ER图，属性图，本章细致入微地对在线购物系统进行设计，成功完成了代码前的所有预备工作。

# 系统实现



## 顾客登陆浏览模块的实现

### 界面实现

如图5-1，商城首页实现。当用户访问商城首页时，首先进入的便是前台页面。前台页面设计的美观程度将直接影响用户的购买欲望，因此前台也页面的美观程度至关重要。



图 5.1.1-1（1） 商城首页



图 5.1.1-1（2） 商城首页

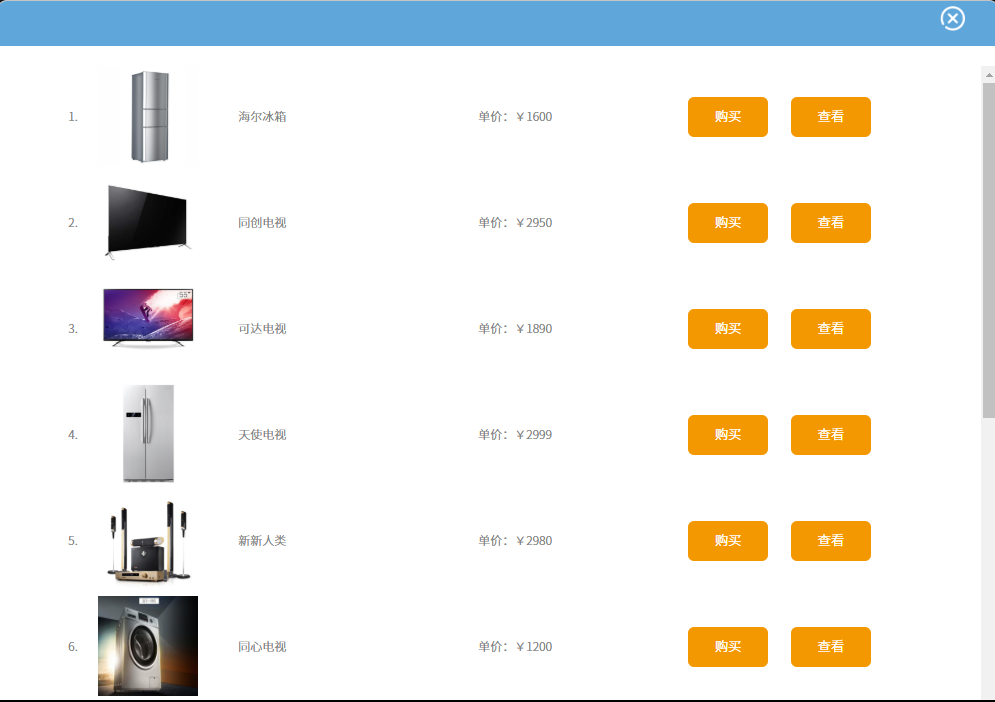


图 5.1.1-1（3） 商城首页

### 代码实现

（为了节省篇幅，此处只展示核心代码）

#### 首页Banner设计

<div style="background-image: url(image/KCM/NET电子商务网站首页\_05.gif); width: 790px;

height: 172px;">

<div style="width: 100%; height: 20px;">

<!—显示欢迎语句-->

<div style="color: #fff; width: 260px; float: left; margin-top: 3px;">HI，欢

迎来到 DIANZISHANGMU 电子商务中心</div>

<!—显示退出功能-->

<div id="outBtn" style="color: #fff; width: 60px; float: right; margin-top: 3px;

cursor: pointer;" onclick="Out()">退出</div>

<!—显示登录功能-->

<div id="loginBtn" style="color: #fff; width: 60px; float: right; margin-top:

3px; cursor: pointer;" onclick="ClickLogin()">登录</div>

<!—显示用户名-->

<div id="userNameBtn" style="color: #fff; width: 60px; float: right; margin-top:

3px; cursor: pointer;" onclick="User()">用户名</div>

<!—显示注册-->

<div id="regBtn" style="color: #fff; width: 60px; float: right; margin-top: 3px;

cursor: pointer;" onclick="ClickReg()">注册</div>

<!—显示我的订单-->

<div id="dingdanBtn" style="color: #fff; width: 60px; float: right; margin-top:

3px; cursor: pointer;" onclick="ClickOrder()">我的订单</div>

<!—显示购物车-->

<div id="carBtn" style="color: #fff; width: 60px; float: right; margin-top: 3px;

cursor: pointer;" onclick="shoppC()">购物车</div>

<!—显示客服中心-->

<div id="kefuBtn" style="color: #fff; width: 60px; float: right; margin-top:

3px; cursor: pointer;" onclick="ceneter()">客服中心</div>

</div>

<div style="width: 100%; height: 50px;">

<div style="width: 220px; height: 50px; float: left; margin-top: 60px;

margin-left: 45px;">

<div style="color: #fff; font-size: 20px; font-weight:

900;">DIANZISHANGWU</div>

<div style="color: #fff; font-size: 20px; font-weight: 900;">电子商务中心</div>

</div>

<!—显示查询文本框-->

<div style="width: 500px; height: 50px; float: right; margin-top: 60px;">

<input id="strSelect" type="text" style="height: 40px; width: 320px; float:

left; font-size: 15px;" placeholder="请输入搜索信息：例如“电视，彩电，洗衣机”" />

<input type="button" value="搜索" style="float: left; height: 46px; width:

100px; border: none; font-size: 18px; color: white; background-color: rgb(255, 106, 0)"

onclick="Clickselect()" />

</div>

<div style="width: 500px; height: 20px; float: right; margin-top: 0px;">

<div style="float: left; color: white; width: 110px;">辉伟等离子</div>

<div style="float: left; color: white; width: 110px;">新书推荐</div>

<div style="float: left; color: white; width: 110px;">特价五折</div>

<div style="float: left; color: white; width: 110px;">电视</div>

</div>

</div>

</div>

#### 轮播图展示



图5.1.2.2-1轮播图

首页的Banner下面会议轮播图的方式展示商城的最新活动及热门商品，轮播图的具体功能为：鼠标经过轮播图时，下面会显示四张页面，用户可以单击选择图片。

<div style="width: 420px; height: 270px; float: left;">

<!--ͼƬѭ۷-->

<div id="lb" style="width: 420px; height: 270px; position: relative; overflow:

hidden; text-overflow: clip;">

<ul>

<li>

<img src="OrderImg/1.jpg" /></li>

<li>

<img src="OrderImg/2.jpg" /></li>

<li>

<img src="OrderImg/3.jpg" /></li>

<li>

<img src="OrderImg/4.jpg" /></li>

</ul>

<div class="imgbanner">

<div class="imgbannerDiv">

<img src="OrderImg/1.jpg" />

<img src="OrderImg/2.jpg" />

<img src="OrderImg/3.jpg" />

<img src="OrderImg/4.jpg" />

</div>

</div>

</div>

</div>

#### 显示全部商品分类

图5.1.2.3-1显示全部商品分类

在购物商城的首页会显示所有的商品分类，用户可以根据分类查看制定类型的商品列表。商品分类导航如图所示。

//获取类型

function GetType() {

var pd = { "t": "SelectType" }; //设置需要向后台传递的值

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

var dataobj = eval("(" + data.status + ")");//格式化后台传递过来的值

$('#lx').empty(); //清空显示类型区域的内容

var tagtypeStr = '-99'; //当前类型

var tagstr = ''; //当前文本

var o = 1;

$.each(dataobj.root, function (i, item) {

var title = item.TypeName; //显示文本

var values = item.ID; //值

var typeStr = item.superType; //类型

if (tagstr == '') {

tagtypeStr = typeStr;

}

if (tagtypeStr == typeStr) {

tagstr += "<font

style='cursor:pointer;margin-left:18px;margin-top:20px;display:block;float:left'

onclick='ClickType(" + values + ")'>" + title + "</font>";

o++;

if (o == 4) {

tagstr += "<br />";

o = 1;

}

}

else {

$('#lx').append('<div style="margin-left: 1px;

font-size: 14px; color: rgb(92, 92, 92)">' + tagstr + '</div>');

tagtypeStr = '-99';

tagstr = '';

}

});

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

/// <summary>

/// 查询类型

/// </summary>

/// <param name="context"></param>

public void SelectType(HttpContext context)

{

string sql = "select tb\_subtype.\*,tb\_supertype.TypeName as ClassName from

tb\_subtype,tb\_supertype where tb\_subtype.superType=tb\_supertype.ID";//sql语句

DataSet dssuper = DataBase.ReDataSet(sql); //执行sql语句

DataTable dt = dssuper.Tables[0]; //提取查询出来的表

if (dt.Rows.Count > 0)

{

string json = f.ToJson(dt); //把表格式化成json格式的字符串

json = json.Replace("\"", "\\\""); //替换非法字符

context.Response.Write("{\"status\":\"" + json + "\"}");

}

else

{

context.Response.Write("{\"status\":\"-1\"}");

}

}

#### 显示边栏促销和公告

 在购物商城的首页中，在轮播图的右侧会显示公告，如图所示。

图5.1.2.4-1 公告

//获取公告

function GetBBS() {

var pd = { "t": "GetBBs" }; //传递到后台的参数

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

var dataobj = eval("(" + data.status + ")"); //格式化数据

$.each(dataobj.root, function (i, item) {

var html = '<div style="margin-top: 15px; margin-left: 10px;cursor:

pointer;white-space:nowrap; text-overflow:ellipsis;overflow:hidden;"

onclick="clickBBs(\'' + item.ID + '\')"><font style="color:#ff3300">【促销】</font> '

+ item.title + '</div>';

$('#gg').append(html); //把数据显示到前台

});

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

/// <summary>

/// 查询公告

/// </summary>

/// <param name="context"></param>

public void GetBBs(HttpContext context)

{

string sql = "select top(7) \* from tb\_bbs order by INTime desc"; //sql语句

DataSet dssuper = DataBase.ReDataSet(sql); //执行sql语句返回一组表

DataTable dt = dssuper.Tables[0]; //提取其中的第一张表

if (dt.Rows.Count > 0)

{

string json = f.ToJson(dt); //格式化表返回到前台

json = json.Replace("\"", "\\\"");

context.Response.Write("{\"status\":\"" + json + "\"}");

}

else

{

context.Response.Write("{\"status\":\"-1\"}");

}

}

#### 搜索所有商品

搜索商品功能主要是通过Clickselect()方法实现的，该方法为Javascript方法，主要通过Ajax技术访问到tools文件夹中的Handler.ashx文件，在Handler.ashx文件中执行完搜素操作之后，Clickselect()方法对返回到前台的JSON数据进行解析，并显示在使用EjecteIdent()方法弹出的select.html页面之中，代码如下：

图5.1.2.5-1搜索框

//单击搜索

function Clickselect() {

var str = $('#strSelect').val(); //获取文本框中的数据

var pd = { "t": "SetSelectStr", "str": str }; //设置传递到后台的数据

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

EjectIdent('select.html', '', '1000', '700'); //弹出一个页面

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

<body>

<!--显示搜索-->

<div style="width:425px;margin:20px auto;height:50px;">

<!--显示搜索文本框-->

<input id="strSelect" type="text" style="height: 40px; width: 320px; float: left;

font-size: 15px;" placeholder="请输入搜索信息：例如“电视，彩电，洗衣机”" />

<input type="button" value="搜索" style="float: left; height: 46px; width: 100px;

border: none; font-size: 18px; color: white; background-color: rgb(255, 106, 0)"

onclick="Clickselect()" />

</div>

<div id="goods" style="margin-top:20px; overflow:auto; height:600px;clear:both;">

<!--显示搜索到的内容-->

</div>

<!--单击产品的时候显示出来的产品详细信息-->

<div id="cen" style="margin-left:150px;

position:absolute;background-color:#e5e5e5;left:190px;top:150px;border:1px solid

#cccccc;width:300px;height:400px;">

<input type="hidden" id="proid" />

<!—关闭按钮-->

<div

style="float:right;cursor:pointer;margin-top:10px;margin-right:10px;font-size:12px;"

onclick="Close()">关闭</div>

<div style="width:200px;margin:auto;margin-top:20px;">

<img src="" id="proImg" style="height:200px;width:200px;margin:auto;" /></div>

<!—产品内容-->

<div style="line-height:40px;margin-left:50px;">

<div>产品名称：<font id="proName"></font></div>

<div>原价：<font id="proPrice"

style="text-decoration:line-through;"></font></div>

<div>现价：<font id="pronowPrioce"></font></div>

<div id="des"></div>

<!—购买按钮-->

<div><input type="button"

style="width:100px;margin-left:40px;height:30px;background-color:#ff6a00;border:none

;color:#fff;" value="购买" onclick="buyCen()" /></div>

</div>

</div>

</body>

//根据商品类型获取商品

function GetDetile() {

var pd = { "t": "Select" }; //设置传递到后台的参数

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

var dataobj = eval("(" + data.status + ")"); //格式化数据

$.each(dataobj.root, function (i, item) {

var html = '<div style="height:100px;clear:both;margin-left:50px;">'

+

' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:20px;wi

dth:20px;">' + (i + 1) + '.</div>' +

' <div style="float:left;

margin-left:10px;"><img src="' + item.picture + '" style="height:100px;width:100px;"

/></div>' +

' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;wi

dth:200px;">' + item.goodsName + '</div>' +

' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;wi

dth:80px;">单价：￥' + item.price + '</div>' +

' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:130px;"

>' +

' <input type="button" value="购买"

style="background-color:#ff6a00;color:#fff;border:none;height:40px;width:80px;"

onclick="buy(\'' + item.ID + '\')" />' +

' <input type="button" value="查看"

style="background-color:#ff6a00;color:#fff;border:none;height:40px;width:80px;margin

-left:20px;" onclick="look(\'' + item.ID + '\')" />' +

' </div>' +

' </div>';

$('#goods').append(html); //向页面添加内容

});

}

else {

$('#goods').html('<div

style="font-size:28px;font-weight:900;text-align:center;margin-top:200px;">没有任何产

品</div>');

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

#### 显示最新上架的商品

在购物商城的首页可以显示最新上架的所有商品，该部分的布局代码如下（下面代码中的注释代码是显示商品的模板）

<body>

<div id="goods" style="margin-top:20px; overflow:auto; height:670px;">

<!--<div style="height:100px;clear:both;margin-left:50px;">

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:20px;">

1.</div>

<div style="float:left; margin-left:10px;"><img src="image/父亲节1.jpg"

style="height:100px;width:100px;" /></div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

超级变形洗衣机</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

单价：￥100</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:130px;"

>

<input type="button" value="购买"

style="background-color:#ff6a00;color:#fff;border:none;height:40px;width:80px;"

onclick="buy('')" />

<input type="button" value="查看"

style="background-color:#ff6a00;color:#fff;border:none;height:40px;width:80px;margin

-left:20px;" />

</div>

</div>-->

</div>

<div id="cen" style="margin-left:150px;

position:absolute;background-color:#e5e5e5;left:190px;

top:150px;border:1px solid #cccccc;width:300px;height:400px;">

<input type="hidden" id="proid" />

<!—关闭按钮-->

<div

style="float:right;cursor:pointer;margin-top:10px;margin-right:10px;font-size:12px;"

onclick="Close()">关闭</div>

<div style="width:200px;margin:auto;margin-top:20px;"><img src="" id="proImg"

style="height:200px;width:200px;margin:auto;" /></div>

<div style="line-height:40px;margin-left:50px;">

<!—产品名称-->

<div>产品名称：<font id="proName"></font></div>

<!—产品原价-->

<div>原价：<font id="proPrice"

style="text-decoration:line-through;"></font></div>

<!—产品现价-->

<div>现价：<font id="pronowPrioce"></font></div>

<div id="des"></div>

<!—购买按钮-->

<div><input type="button"

style="width:100px;margin-left:40px;height:30px;background-color:#ff6a00;border:none

;color:#fff;" value="购买" onclick="buyCen()" /></div>

</div>

</div>

</body>

## 购物车模块设计

在购物网站中，会员登陆后，单击某商品可以进入到显示商品的详细信息页面，在该页面中，单击“购买”按钮，即可将该商品添加到购物车，购物车效果如图所示。

### 界面实现

图 5.2.1-1 购物车实现

### 代码实现

#### 购物车架构

购物车中的商品使用xml文件进行保存，该文件中主要存储商品的单据号，商品类别号，商品名称，商品介绍，商品价格，商品图片，加入时间，是否新加入信息。Xml文件结构如下：

<?xml version="1.0" standalone="yes"?>

<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"

xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">

<NewDataSet>

<Table diffgr:id="Table1" msdata:rowOrder="0">

<ID>3</ID>

<typeID>1</typeID>

<goodsName>海尔节能冰箱</goodsName>

<introduce>海尔节能冰箱经济型!!!</introduce>

<price>1600</price>

<nowPrice>1600</nowPrice>

<picture>goodimg/1.jpg</picture>

<INTime>2006-03-29T12:41:44+08:00</INTime>

<newGoods>1</newGoods>

<count />

</Table>

</NewDataSet>

</diffgr:diffgram>

定义一个godataset()方法，使用哈希表存储购物车中的商品编号和数量。

public void godataset(string id) //向Hashtable表中购物车中商品的id和数量

{

if (Session["bus"] == null) //是否存在产品

{

Hashtable ht = new Hashtable(); //设置一张表来储存产品

ht.Add(id, 1); //添加产品

Session["bus"] = ht;

dafill(ht); //把表进行储存

}

else

{

Hashtable ht = (Hashtable)Session["bus"]; //获取所有的产品

if (ht[id] == null)

{ht[id] = 1; //设置数量为1

}

else

{ht[id] = (int)ht[id] + 1; //数量加1

}

Session["bus"] = ht;

dafill(ht);

}

}

在后台的Handler.ashx文件中定义一个GetshopCar()方法，用来获取购物车中存储的所有商品信息，实现该方法，首先需要加载xml文件存储的数据格式化格式为JSON，代码如下：  
/// <summary>

/// 获取购物车

/// </summary>

/// <param name="context"></param>

public void GetshopCar(HttpContext context)

{

Class2.godset.WriteXml("F:\\aa.xml", XmlWriteMode.DiffGram);

try

{

if (Class2.godset.Tables[0].Rows.Count > 0)

{

DataTable dt = Class2.godset.Tables[0]; //获取查询到的购物车内容

foreach (DataRow dr in dt.Rows)

{

dr["count"] = GetCount(dr["ID"].ToString()); //获取物品的数量

}

string json = f.ToJson(dt); //格式化查询到的数据

json = json.Replace("\"", "\\\"");

context.Response.Write("{\"status\":\"" + json + "\"}");

}

else

{

context.Response.Write("{\"status\":\"-1\"}");

}

}

catch

{

}

}

#### 购物车计算数量和小计

在前台页面中显示购物车中的所有商品时，会同时显示购买物品的数量，该功能是通过GetCount()方法实现的，代码如下

/// <summary>

/// 获取购物车的数量

/// </summary>

/// <param name="context"></param>

public string GetCount(string id)

{

string count =

((System.Collections.Hashtable)(HttpContext.Current.Session["bus"]))[id].ToString();

return count;

}

#### 利用“+”和“-”修改商品数量

在购物车中，用户可以对所有购买商品的数量进行修改，单击“+”，增加商品数量，单击“-”，减少商品数量。实现代码如下：

//减

function reduce(a, id) {

//减少数量

var next = $(a).next('font'); //获取此标记的下一个font标记

if (next.html() != "0") { //判断有没有下一个标记

next.html(parseInt(next.html()) - 1);

}

//减少钱数

var price = parseFloat($(a).parent().prev('div').html()); //单价

var size = parseInt(next.html()); //个数

var amount = price \* size; //计算小计

$(a).parent().next('div').html('小计 ' + amount + ' （元）');//设置小计

allAmount -= price;

$('#allamount').html('总价：￥' + allAmount);

var pd = { "t": "UpdateNum", "id": id, "type": "2" }; //传递到后台

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

//加

function add(a, id) {

var prev = $(a).prev('font'); //增加数量

prev.html(parseInt(prev.html()) + 1); //增加钱数

var price = parseFloat($(a).parent().prev('div').html()); //单价

var size = parseInt(prev.html()); //个数

var amount = price \* size;

$(a).parent().next('div').html('小计 ' + amount + ' （元）');

allAmount += price;

$('#allamount').html('总价：￥' + allAmount);

var pd = { "t": "UpdateNum", "id": id, "type": "1" }; //调用后台方法

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

/// <summary>

/// 修改购物车数量

/// </summary>

/// <param name="context"></param>

/// <returns></returns>

public void UpdateNum(HttpContext context)

{

Class2 sh = new Class2();

string id = HttpContext.Current.Request["id"]; //产品id

string type = HttpContext.Current.Request["type"]; //修改类型，有可能加，有可能减

if (HttpContext.Current.Session["wo"] == null)

{

context.Response.Write("{\"status\":\"-1\"}"); //请登录

}

else

{

if (type == "1")

{ sh.godataset(id); //增加一个

}

else

{sh.godatasetjs(id); //减少一个

}

context.Response.Write("{\"status\":\"0\"}");

}

}

#### 删除选中商品

如果不再像购买某商品，可以对齐执行删除操作。在购物车中单击“删除”按钮即可。

//删除物品

function DeleteGoods() {

if ($('.ckClass:checked').length == 0)

{

return;

}

if (confirm('是否删除所选?')) { //弹出提示

var check = $('.ckClass:checked').each(function () {

var id = $(this).val();

var pd = { "t": "SetXml", "id": id }; //向后台传递的参数

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

//获取金额

var amoun = $($('#car' +

id).children('div').get(4)).html().replace('小计', '').replace('（元）', '');

allAmount -= parseFloat(amoun);

$('#allamount').html('总价：￥' + allAmount); //设置总价

$('#car' + id).remove();

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

});

}

}

## 结算模块设计

### 修改收货地址

在购物车中单击“去结算”按钮，跳转到结算页面，在该页面中输入收货人信息，支付方式和备注后，单击“提交订单”按钮，即可完成订单，如图5.3.1-1所示。



图 5.3.1-1 用户界面

### 代码实现

#### 购物车架构

在结算页面中，用户可以对收货地址进行修改，该功能实现需要同时用到前台Javascript方法和后台逻辑处理方法。在前台定义一个AddAdr()方法，该方法首先获取收货人的姓名，地址，详细地址，手机号码，固话号码等信心，并对这些信息进行验证。如果验证功能通过，则奖这些信息一JSON格式传递到后台的处理文件，后台的处理为念对其进行分析和处理，并将结果传递到前台。

function AddAdr() {

var Names = $('#Names').val(); //收货人姓名

var adr = $('#adr').val(); //收货人地址

var Des = $('#Des').val(); //详细地址

var phone = $('#phone').val(); //手机

var mobil = $('#mobil').val(); //电话

var email = $('#email').val(); //邮箱

var bieming = $('#bieming').val(); //备注

var userName = $('#userName').val(); //用户名

if (Names == '') {

alert('收货人不能为空'); //弹出提示

return;

}

if (Des == '') {

alert('详细地址不能为空'); //弹出提示

return;

}

if (phone == '') {

alert('手机号不能为空'); //弹出提示

return;

}

//向后台传递的参数

var pd = { "t": "AddAdr", "Names": Names, "adr": adr, "Des": Des, "phone": phone,

"mobil": mobil, "email": email, "bieming": bieming, "userName": userName };

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

closeCeng(); //关闭页面

GetAdr(); //获取地址信息

}

else {

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

/// <summary>

/// 添加地址

/// </summary>

/// <param name="context"></param>

public void AddAdr(HttpContext context)

{

string adr = HttpContext.Current.Request["adr"]; //收货人地址

string Des = HttpContext.Current.Request["Des"]; //详细信息

string phone = HttpContext.Current.Request["phone"]; //电话

string mobil = HttpContext.Current.Request["mobil"]; //手机电话

string email = HttpContext.Current.Request["email"]; //邮箱

string bieming = HttpContext.Current.Request["bieming"]; //备注

string Names = HttpContext.Current.Request["Names"]; //姓名

string userName = HttpContext.Current.Session["wo"].ToString(); //登录人

string sql = "delete Adr where UserName='" + userName + "'; insert into

Adr(id,adr,Des,phone,mobil,email,bieming,Names,UserName) values('" +

Guid.NewGuid().ToString() + "','" + adr + "','" + Des + "','" + phone + "','" + mobil

+ "','" + email + "','" + bieming + "','" + Names + "','" + userName + "')";

strcon.Close(); //关闭链接

strcon.Open(); //打开链接

SqlCommand comm = strcon.CreateCommand();

comm.CommandText = sql; //设置sql语句

SqlTransaction myTransaction = strcon.BeginTransaction(); //开始一个事务

comm.Transaction = myTransaction;

try

{

comm.ExecuteNonQuery(); //执行sql语句

myTransaction.Commit(); //提交事务

}

catch

{

myTransaction.Rollback(); //回滚事务

}

finally

{

strcon.Close();

}

context.Response.Write("{\"status\":\"0\"}");

}

#### 显示我的订单

单击购物商城首页的“我的订单”按钮，可以查看自己的历史订单，该部分使用DIV+CSS进行布局，代码如下。

<body>

<div id="goods" style="margin-top:20px; overflow:auto; height:670px;">

<!--<div style="height:100px;clear:both;margin-left:50px;">

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:20px;">

1.</div>

<div style="float:left; margin-left:10px;"><img src="image/父亲节1.jpg"

style="height:100px;width:100px;" /></div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

超级变形洗衣机</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

单价：￥100</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

数量：</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

支付方式</div>

<div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

购买时间</div>

</div>-->

</div>

</body>

function GetDetile() {

var pd = { "t": "GetOrderList" }; //向后台传递的参数

$.ajax({

type: "post",

url: "tools/Handler.ashx",

data: pd,

dataType: "json",

success: function (data) {

if (data.status != "-1") {

var dataobj = eval("(" + data.status + ")");//格式化数据

$.each(dataobj.root, function (i, item) {

var html = '<div

style="height:100px;clear:both;margin-left:50px;">' +

' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:20px;">

1.</div>' + ' <div style="float:left; margin-left:10px;"><img src="' + item.picture

+ '" style="height:100px;width:100px;" /></div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

' + item.goodsName + '</div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

单价：￥' + item.nowPrice + '</div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

数量：' + item.number + '</div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

' + item.pay + '</div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

' + item.OrderDate + '</div>' + ' <div

style="float:left;line-height:100px;font-size:12px;color:#808080;margin-left:40px;">

' + (item.enforce == '0' ? "未执行" : "已执行") + '</div>' + ' </div>';

$('#goods').append(html); //把数据设置到标签中

});

}

else {

$('#goods').html('<div

style="font-size:28px;font-weight:900;text-align:center;margin-top:200px;">没有购买过

任何东西</div>');

}

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

}

});

}

## 后台管理模块设计

后台管理模块主要用来对商品，订单，会员和公告信息进行管理，其效果如图5.15所示。



图5.4-后台管理

### 订单管理

在订单管理页面中，主要显示商城中所有的订单，该功能是通过一个自定义的dsdatabind()方法实现的。在该方法中，从订单表中获取所有的数据库，并绑定到GridView控件中进行显示。另外，在对Gridview进行绑定时，需要指定绑定的主键字段为订单号。代码如下：

protected void dsdatabind()//绑定GridView1数据

{

//要执行的查询所有订单的语句

string newque = "select

a.orderid,a.username,b.truename,a.pay,a.carry,a.orderdate,case enforce when 0 then '

未执行' else '已执行' end as zx from tb\_order a left outer join tb\_member b on

a.username=b.username ";

SqlDataAdapter datap = new SqlDataAdapter(newque, strcon);

datap.Fill(dset); //获取查询到的内容

GridView1.DataSource = dset; //把数据设置到前台控件中

this.GridView1.DataKeyNames = new string[] { "orderid" };//设置控件的主键

GridView1.DataBind(); //把数据绑定到控件中

strcon.Close(); //关闭数据链接

}

### 会员管理

在会员管理页面中，管理员可以对网站已经注册的会员进行查看和删除。定义dsdatabind()方法，该方法从tb\_member数据表中获取所有的注册会员信息，并将查询结果绑定到Grid View控件中进行展示。

protected void dsdatabind() //绑定GridView1数据

{

strcon.Open(); //打开连接

string newque = "SELECT \* FROM tb\_member"; //查询所有会员

SqlDataAdapter datap = new SqlDataAdapter(newque, strcon);

datap.Fill(dset);

GridView2.DataSource = dset; //设置数据源

this.GridView2.DataKeyNames = new string[] { "id" }; //设置主键

GridView2.DataBind();

strcon.Close();

}

### 商品管理

在商品管理页面中，管理员可以对商品信息进行查看，编辑和删除。

protected void dsdatabind() //绑定GridView1数据

{

string newque = "SELECT \*, CASE newgoods WHEN 1 THEN '是' ELSE '否' END AS nngoods

FROM tb\_goods"; //查询所有的商品

GridView1.DataSource = DataBase.ReDataSet(newque);

this.GridView1.DataKeyNames = new string[] { "id" };

GridView1.DataBind(); //绑定到前台GridView控件

}

protected void Button1\_Click(object sender, EventArgs e)

{

string goodsname = TextBox1.Text.ToString(); //设置商品名称

string introduce = TextBox4.Text.ToString(); //设置商品描述

double price = Convert.ToDouble(TextBox3.Text); //设置商品原价

double nowprice = Convert.ToDouble(TextBox5.Text); //设置商品现在价格

string picture = TextBox2.Text.ToString(); //设置商品图片

int newgoods; //标识是否为新上架商品

if (RadioButtonList1.SelectedIndex == 0)

{

newgoods = 0;

}

else

{

newgoods = 1;

}

strcon.Open(); //打开数据库连接

//定义更新商品信息的SQL语句

string ingo = "update tb\_goods set goodsname='" + goodsname + "',introduce='" + introduce

+ "',price='" + price + "',nowprice='" + nowprice + "',picture='" + picture + "',newgoods='"

+ newgoods + "',intime='" + System.DateTime.Now + "' where id="+mes;

SqlCommand goodsin = new SqlCommand(ingo, strcon); //生成命令执行对象

goodsin.ExecuteNonQuery(); //执行商品更新操作

strcon.Close(); //关闭数据库连接

Response.Write("<script language='javascript'>alert('修改成功！

');location='goodsmanage.aspx'</script>");

}

protected void GridView1\_RowDeleting(object sender, GridViewDeleteEventArgs e)

{

strcon.Open(); //打开数据库连接

//获取要删除的商品编号

string id = this.GridView1.DataKeys[e.RowIndex].Value.ToString();

//定义执行删除指定商品的SQL语句

SqlCommand comm = new SqlCommand( "delete from tb\_goods where id="+id, strcon);

comm.ExecuteNonQuery(); //执行删除商品操作

dsdatabind();

strcon.Close(); //关闭数据库连接

}

### 公告管理

在公告管理页面中，管理员可以对网站的公告信息进行查询，编辑和删除。

定义dsdatabind()方法，该方法用来从tb\_BBS数据表中获取所有的公告信息，并将查询结果绑定到GridView中进行显示，代码如下。

protected void dsdatabind() //绑定GridView1数据

{

strcon.Open();

string newque = "SELECT \* FROM tb\_BBS"; //获取所有公告信息的SQL语句

SqlDataAdapter datap = new SqlDataAdapter(newque, strcon); //执行sql语句

datap.Fill(dset); //填充数据到表中

GridView1.DataSource = dset; //绑定数据到控件的数据源中

this.GridView1.DataKeyNames = new string[] { "id" }; //设置主键

GridView1.DataBind(); //绑定数据源到控件

strcon.Close(); //关闭数据库连接

}

触发GridView控件的RowUpdating事件，该事件中，首先获取公告ID，然后根据该ID执行修改公告信息的操作。然后重新对数据进行绑定。

protected void GridView1\_RowUpdating(object sender, GridViewUpdateEventArgs e)

{

strcon.Open(); //打开数据库连接

string uid = this.GridView1.DataKeys[e.RowIndex].Value.ToString();//获取主键

GridViewRow row = this.GridView1.Rows[e.RowIndex]; //获取每行的数据

string ingo = "update tb\_BBS set title='" +

((TextBox)(row.Cells[0].Controls[0])).Text.ToString() + "',content='" +

((TextBox)(row.Cells[1].Controls[0])).Text.ToString() + "' where id=" +uid;//sql语句

SqlCommand goodsin = new SqlCommand(ingo, strcon); //执行sql语句

goodsin.ExecuteNonQuery(); //获取执行受影响的行

strcon.Close(); //关闭数据库连接

GridView1.EditIndex = -1;

dsdatabind(); //绑定公告

}

触发GridView控件的RowDeleting事件，该事件中，首先获取公告ID,然后根据该ID执行公告信息的删除操作，并重新对GridView控件进行数据绑定。代码如下：

protected void GridView1\_RowDeleting(object sender, GridViewDeleteEventArgs e)

{

strcon.Open(); //打开数据库连接

//获取要删除的公告编号

string id = this.GridView1.DataKeys[e.RowIndex].Value.ToString();

//定义执行删除指定公告的SQL语句

SqlCommand comm = new SqlCommand("delete from tb\_BBS where id=" + id, strcon);

comm.ExecuteNonQuery(); //执行删除公告操作

strcon.Close(); //关闭数据库连接

dsdatabind();

}

## 本章总结

下面通过一章概述图，对本章的框架结构进行总结。

# 参考文献

[1] 《精通C#（第6版）》. 人民邮电出版社. 2013-7

[2] 《UML大战需求分析》. 张传波.2015

[3] 《数据库原理》. 清华大学出版社.2011-11-16