摘 要

随着互联网的发展，网民数量在极剧增长，商家看好网络市场，纷纷投入资金进行网上商城的建设，网上购物环境日渐成熟。

网上购物系统，是在网络上建立一个虚拟的购物商场，避免了挑选商品的烦琐过程，使您的购物过程变得轻松、快捷、方便，很适合现代人快节奏的生活；同时又能有效的控制商场运营的成本，开辟了一个新的销售渠道。可对会员非会员同时进行购物管理，实现标准购物车功能（分为修改、继续购物、清空、结算四个状态），可对购物车在结算之前任意步骤进行查询和修改，购物过程支持网上结算，购买者可依据订单号查询订单状态（已收到订单、已收到货款、已发货、已送达收货人等状态），后台设置管理员维护界面，可在首页设定打折商品或推荐商品，可设定会员购买折扣，可对订单状态进行跟踪和管理（修改状态、删除订单）可查询当日新增加订单和所有订单。

网上购物系统一般是基于 B/S 模式的。这样的结构在客户端不需要安装额外的软件就可以使用，只需要一个浏览器就可以使用系统了(一般操作系统自带)。B/S（Browser/Server，浏览器/服务器）模式又称 B/S 结构。它是随着 Internet 技术的兴起，对 C/S 模式应用的扩展。在这种结构下，用户工作界面是通过 IE 浏览器来实现的。B/S 模式最大的好处是运行维护比较简便，能实现不同的人员，从不同的地点，以不同的接入方式（比如 LAN, WAN, Internet/Intranet 等）访问和操作共同的数据。

然而现在大多数搭建网上购物系统的方法对于将来的维护和扩展造成了麻烦，急需一种简单，清晰，可维护性和可扩展性十分便利的框架来搭建网上购物系统。这个框架就是轻量级架构。

本文以网上购物系统为案例，探讨如何应用轻量级架构实现中小型基于 B/S 结构系统的设计、实现方法，具体完成了系统需求分析、数据库设计、数据流程设计、界面以及部分模块的具体实现。同时，也对相关的理论及轻量级架构应用前景作了较为全面的阐述。

关键词：网上购物；B/S 模式；C/S 模式 轻量级架构

Abstract

With the development of the Internet, Sharp growth in the number of Internet users is extremely, Businessmen about network market, Having invested in building Internet Mall, Online shopping environment increasingly mature.

Online shopping system is the creation of a virtual network of shopping malls, Avoid the cumbersome process of selection, Make your shopping process becomes easy, fast, convenient, Well suited to the fast pace of modern life; At the same time effective control center operation costs, Opened a new sales channels. Member of the non-member can simultaneously manage shopping, Achieving the standard shopping car functions (divided into modifications, continue shopping, house, clearing four state), Can in a shopping cart before clearing and modification of arbitrary steps enquiries, Shopping process online clearing support, Enquiries can be made on the basis of purchase orders of state orders (orders received and has received the purchase price, has delivery, and has served on the recipient state), Background of managers maintaining interface, In the first set of commodities may be discounted or recommending products, Member may purchase a discount, To tracking and management of state orders (modified state, delete orders) may increase orders and enquiries on all new orders.

Online shopping system is generally based on B/S model. Such structures in the client side does not need to install additional software to be used, Only needs a browser

can use the system (generally operating system built). B/S (Browser/Server) model also named B/S structure, It is with the rise of Internet technology, C/S models for the expansion. In this structure, users inteface depend on IE browser interface to achieve, B/S model greatest advantage is relatively simple maintenance operation, To achieve different personnel from different locations, using different access methods (such as LAN, WAN, Internet/Intranet) visits and operate common data.

But mostly to put up the systematic method of online shopping cause the trouble to maintenance and expansion in future now. It is to need one simple, clear, maintainability, and expansibility convenient frame come to put up the system of online shopping. This frame is a lightweight frame.

This text is example for online shopping system. Talking about how the lightweight frame based on B/S structural system designed implementation method. Successed in system analysis, database design, data process design, user interface. especially, And in-depth analysis of its strengths and weaknesses and prospects for application.

Keywords: online shopping; B/S Model; C/S Model; lightweight frame

独创性声明

本人声明，所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽本人所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教学机构的学位或证书而使用过的材料。与我一起工作的同志对本研究所做的任何贡献已在论文中作了明确的说明并表示了谢意。

本人签名：夏楚

日期：2006年7月15日

关于论文使用授权的说明

本人完全了解北京交通大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。论文中所有创新和成果归北京交通大学软件学院所有。未经许可，任何单位和个人不得拷贝。版权所有，违者必究。

本人签名：_____

日期：2006 年 1 月 15 日

第1章 绪 论

1.1 选题背景

电子商务是指采用数字化电子方式进行商务数据交换和开展商务业务活动。电子商务系统是涉及商务活动的各方，包括商店、消费者、银行或金融机构、信息公司或证券公司和政府等，利用计算机网络技术全面实现在线交易电子化的过程。电子商务系统的关键在于完全实现在线支付功能，所以为了顺利完成整个交易过程，需要建立电子商务服务系统、通用的电子交易支付方法和机制，还要确实保证参加交易各方和所有合作伙伴都能够安全可靠地进行全部商业活动。

由于电子商务是在 Internet 等网络上进行的，因此，网络是电子商务最基本的构架；电子商务还强调要使系统的软件和硬件、参加交易的买方、卖方、银行或金融机构、厂商、企业和所有合作伙伴，都要在 Internet、Intranet、Extranet 中密切结合起来，共同从事在网络计算环境下的商业电子化应用。

随着互联网的发展，网民数量在极剧增长，商家看好网络市场，纷纷投入资金进行网上商城的建设，网上购物环境日渐成熟。

现在人们天天从收音机、电视、报纸和网络上听到看到的电子商务概念老百姓普遍认为是指“网上购物”－通过 WEB 技术将产品、服务和信息销售给顾客。

网上购物有如下优势：

1. 送货上门方便
2. 价格便宜
3. 买到本地所缺物品
4. 节省体力和时间
5. 商品品种较多

6. 比传统购物效率高

网上购物系统，是在网络上建立一个虚拟的购物商场，避免了挑选商品的烦琐过程，使您的购物过程变得轻松、快捷、方便，很适合现代人快节奏的生活；同时又能有效的控制商场运营的成本，开辟了一个新的销售渠道。可对会员非会员同时进行购物管理，实现标准购物车功能（分为修改、继续购物、清空、结算四个状态），可对购物车在结算之前任意步骤进行查询和修改，购物过程支持网上结算，购买者可依据订单号查询订单状态（已收到订单、已收到货款、已发货、已送达收货人等状态），后台设置管理员维护界面，可在首页设定打折商品或推荐商品，可设定会员购买折扣，可对订单状态进行跟踪和管理（修改状态、删除订单）可查询当日新增加订单和所有订单。

网上购物系统一般是基于 B/S 模式的。这样的结构在客户端不需要安装额外的软件就可以使用，只需要一个浏览器就可以使用系统了(一般操作系统自带)。B/S（Browser/Server，浏览器/服务器）模式又称 B/S 结构。它是随着 Internet 技术的兴起，对 C/S 模式应用的扩展。在这种结构下，用户工作界面是通过 IE 浏览器来实现的。B/S 模式最大的好处是运行维护比较简便，能实现不同的人员，从不同的地点，以不同的接入方式（比如 LAN, WAN, Internet/Intranet 等）访问和操作共同的数据。

开发网上购物系统的方案有很多，然而一些方法对于系统的可维护性、可扩展性较差，如开发中需求发生变化，需要修改多处代码，或者系统要添加新功能将会变得非常困难。

基于上述原因，基于 B/S 模式的轻量级架构具有旧的架构无可比拟的优势，用这个框架开发一个网上平台来让用户购买商品，这个平台就是网上购物系统。本论文以用户模块为例阐述了系统的详细设计和轻量级框架的实现，为今后的研发工作奠定了良好的基础。

1.2 电子商务的发展与现状

1.2.1 我国电子商务的发展状况

1998 年,是世界的“电子商务年”，新成立的信息产业部提出：推进国民经济信息化，要重点抓好企业信息化、金融电子化和电子商务这三个方面的工作。企业信息化是基础，金融电子化是保证，电子商务是核心。一场有关电子商务研究和讨论的“电子商务热”随之在国内掀起,我国的电子商务已进入到起步阶段。金桥工程的实施，推动了我国信息基础设施建设步伐，促进了我国因特网的普及和应用，为电子商务的实施打下了一定的物质基础。金卡工程的实施,推动了我国一些商业银行的电子化进程,为电子商务的开展打下了基础.从某种意义上来说，金卡工程本身就是电子商务在我国的应用试点，并取得了显着的成效。截止到 1997 年底，首批 12 个试点省市全部实现了自动柜员机 ATM 与销售点终端机 POS 的同城跨行(工、农、中、建、交等各商业银行)联网运行和信用卡业务的联营，这中间包括了电子数据交换 EDI、电子转账 EFT 的实际应用，金卡工程的建设为实现网上支付与资金清算提供了很好条件。比如，上海市商业增值网已连入金卡网络，这使得全市近百家大型商户建立了计算机管理系统，并与金卡网络相连；此外，中小型商场和超市、连锁店普遍采用了收款机，可全面受理信用卡，初步具备了发展电子商务所需要的基本条件。金贸工程是电子商务在经贸流通领域的应用工程，也是我国电子贸易体系建设的一项试点工程.商品交换是商品经济社会永恒的主题，研究市场经济，研究商品交易的学问是每一个企业在商品经济社会中求生存、图发展的必修课。金贸工程就是帮助企业，特别是帮助我们的国有大中型企业进行改革，走出困境，学会利用

现代电子信息技术手段管理企业，研究市场，学会经营贸易，开创商品交易新的模式的一项计算机应用系统工程。市场竞争的规律，其最终结果就是优胜劣汰。面对日益激烈、残酷的市场竞争，特别是面对国外跨国公司的竞争，我们的国有企业往往处于劣势。究其原因，除市场经济的“阅历”比较浅以外，还有两个重要的原因：一是观念上的问题，有些企业的领导干部，在市场经济的环境下，还没有把“贸易”，也就是“市场”放在生死攸关的重要位置上，还在等上级或别人来保护和援救；二是手段上的问题。现在很多企业，一直在用非常原始落后的方式经营，推销产品。国家经贸委和信息产业部共同推出的金贸工程，就是要引导帮助企业运用全新的观念和方式进行运作，给每一个企业提供一个用先进的信息技术手段进行平等贸易竞争的环境。金贸工程的建设，对我国大中型企业的深化改革，对于我国大型企业走向国际市场将会起到积极的推动作用。

1.2.2 我国网上购物的发展状况

网上购物可以被认为是电子商务的一部分。广义上讲,电子商务是指一种依托现代信息技术和网络技术，集金融电子化，管理信息化，商贸信息网络化为一体，旨在实现物质流，资金流，与信息流和谐统一的新型贸易方式，是贸易过程的电子化，网络化。简单的理解，就是利用电子技术进行商业行为。按应用领域划分，电子商务有以下几种模式：1.企业(Business)对消费者(Consumers or Customers)，也称商业机构对个人用户即B2(to)C；2.企业对企业，也称商家对商家即 B2B;3.企业对政府机构(Government)即 B2G;4.消费者对政府机构即 C2G；5.网上拍卖等个人行为即C2C。B2C模式相当于现实生活中的"商场"或"专卖店"，商业机构利用先进的通信和计算机网络的三维图形技术，把现实

的商业街搬到网上，并通过建立网站，在线发布信息和提供数据库检索向用户介绍和销售产品；消费者使用浏览器进行诸如浏览，购买，定单发送，支付操作；最后由商家将产品送到消费者手中。C2C 比较类似于现实生活中的"小商品批发市场"，网站提供数据库检索和一定的安全保障，收取一定的费用，商品信息的上载和交易的协商都由作为独立个体的"买家"和"卖家"完成，一个网站中同时存在数目众多的个体经营者，网站只起一个现实中"市场管理者"的作用。

我国自 1991 年起先后在海关，外贸，交通航运的部门开展 EDI(即 Electronic Data Interchange，电子数据交换，将业务文件以标准化，规范化的文件格式采用电子化方式，通过网络系统在计算机应用系统与计算机应用系统之间，直接进行信息业务的交换与处理。) 的应用，1993 年启动金卡，金关，金税过程，1996 年外贸部成立中国国际电子商务中心；1997 年出现网上书店，网上购物及中国商品订货系统；1998 年 7 月中国商品交易与市场网站正式运行，北京，上海启动电子商务工程。1998 年 3 月 6 日下午 3:30，国内第一笔 INTERNET 网上电子商务交易成功，中央电视台的王轲平先生通过中国银行的网上银行服务，从世纪互联公司购买了 10 小时的上网机时。3 月 18 日世纪互联和中国银行在京正式宣布了这条消息。事隔不久，满载价值 166 万元的 COMPAQ 电脑的货柜车，从西安的陕西华星公司运抵北京海星凯卓计算机公司，这是在中国商品交易中心的网络上生成的中国第一份电子商务合同。由此开始，因特网电子商务在中国从概念走入应用。1999 年兴起政府上网，企业上网。

1999 年底，正是互联网高潮来临的时候，国内诞生了 300 多家从事电子商务的网络公司。到 2000 年，变成了 700 家，而 2001 年，人们还有印象的只剩下三四家。随后网上购物经历了一个比较漫长的"寒冬时期"。随着经济的发展，网上购物逐渐重放异彩，越来越多的人开始参与网上购物，我国的网上购物环境已经有了

很大的改善。

网上购物的优势是能够降低交易成本，因此，应针对人们对电子商务的不同需求因人制宜，因地制宜。特别是传统购物方式对于高收入阶层，行动能力弱群体以及交通，商业不发达地区的消费者时间成本较高，网上购物更有市场潜力。对于网上购物体系存在的问题，应加强网络安全建设，建设较为严密的信用评价体系，网上支付体系和现代物流配送体系，还必须尽快制定适应约束网上交易行为的法律。虽然我国的网上购物体系还不够成熟，还存在这样那样的问题，但它的发展符合经济学原理和人们的需要，只要对症下药，一定可以逐渐发展完善，在竞争激烈的现代商务竞争中占据稳固地位。

1.3 章节安排

在本论文中作者主要完成了以下几方面的工作：系统需求分析；数据库设计，包括数据库 table 的创建，及其建立各表之间的关系；新闻模块、用户管理模块的设计，及其程序实现等。

本论文章节安排如下：

第一章先给出了电子商务的概念，并分析了目前电子商务的发展状况。

第二章介绍了数据库的概念及其本系统中所使用的数据库，并指出什么是轻量级框架及各层所用的框架，并列出轻量级框架所具有的优势。

第三章以网上购物系统为项目背景概要阐述了系统的总体需求，包括它的功能要求和运行要求。

第四章重点介绍了系统的数据库设计和新闻及用户模块流程图。

第五章详细说明了轻量级构架在本系统中用户模块中的具体

实现。

第六章对全文作出了总结，并根据轻量级架构的优点指出了其应用前景。

第2章 基础理论

2.1 数据库

2.1.1 数据库的基本概念

（1）数据库的基本概念

数据库是一组相关数据的集合，它将数据储存在一个特定的地方，以方便对数据做增加、删除、修改与查询的处理。其最大特色在于：可以对特定的数据类型有着较好的搜索以及排序的算法，来管理不同类型的数据。因此通过数据库，数据可以得到有效的管理以及空间的分配。

（2）数据库的基本结构。

A、系统结构

数据库系统的体系结构，是数据库系统的一个总的框架。尽管实际的数据库系统的软件产品多种多样，支持不同的数据模型，使用不同的数据库语言，建立在不同的操作系统之上，数据的存储结构也各不相同，但绝大多数数据库管理系统在总的体系结构上都具有三级模式结构的结构特征,它们是：概念模式、外模式和内模式。

数据库管理系统提供模式描述语言(模式 DDL)来严格地表示模式所包含的内容。用模式 DDL 写出的一个数据库逻辑定义的全部语句，称为数据库的模式。模式是对数据库结构的一种描述，它是装配数据的一个框架。外模式由外描述语言来描述，用外模式 DDL 写出的一个用户数据视图的逻辑定义的全部语句，称为此用户的外模式。内模式是全体数据库数据的内部表示或者底层描

述，定义数据的存储方式和物理结构。例如记录是按照顺序存储，按照树结构存储，或者用 hash 方法存储，数据压缩存储，是否加密。它是数据的存储结构的具体定义。通常用内模式数据描述语言来描述和定义。

B、存储结构

数据库的存储结构不同于一般文件系统的存储结构。数据库结构的特点是各种记录型之间彼此有联系，数据是结构化的。数据的存储结构不仅涉及每种记录型的记录如何存储，而且反映了各种记录型之间的联系。

联系的存储结构的实现方法主要有：邻接法、链接法、位图法、目录法，在关系数据库中用外来码实现概念记录之间的联系。邻接法是指用连续的物理顺序表示记录之间的联系的方法；链接法是指用指针实现记录之间的联系的方法；目录法是把链接法中用的指针从记录中分离出来，单独组成指针目录，用指针目录表示记录之间的联系。

2.1.2 SQL Server2000 数据库

1、SQL Server2000 数据库的概念及特点

Microsoft® SQL Server™ 2000 由一系列产品组成，不仅能够满足最大的数据处理系统和商业 Web 站点存储数据的需要，还能为个人或小企业提供易于使用的数据存储服务。数据库组件是基于结构化查询语言 (SQL) 的可伸缩的关系数据库，集成了对 Internet 应用程序的可扩展标记语言 (XML) 支持。

如今，企业或政府部门对数据存储的要求非常复杂。以下是一些示例：

联机事务处理 (OLTP) 系统必须能够同时处理上千份订单。

越来越多的公司正在实施将大型 Web 站点作为一种商业途径，顾

客可以通过网络输入订单、联系服务部门和获取产品信息，而许多过去必须与雇员联系才能完成的任务也可以通过网络来处理。这些网站需要安全的、与 Web 紧密集成的数据存储。

有些组织对重要业务，例如人力资源规划、生产资源规划和库存控制采用现成的软件包。这些系统要求数据库能够存储大量的数据和支持众多的用户。

有些组织的用户必须在没有联网的情况下继续工作。例如，正在旅行的销售代表或地区视察员因移动而中断了与网络的连接。这些用户必须使其笔记本或膝上型电脑中的数据与公司系统的当前数据保持同步，与网络断开连接，在现场记录工作结果，然后重新与公司网络连接，将自己的现场工作结果合并到公司数据存储中。

管理人员和市场营销人员需要对公司数据中记录的趋势作更为复杂的分析。他们需要可靠的联机分析处理 (OLAP) 系统，这些系统能够很容易地通过 OLTP 数据生成，并支持复杂的数据分析。独立的软件供应商 (ISV) 必须能够通过专门针对个人或小型工作组而开发的应用程序来分配数据存储能力。这意味着对于购买了该应用程序的用户而言，数据存储机制必须是透明的。这就要求数据存储系统可由应用程序配置，然后系统自身可以自动调整，用户不需要专门的数据库管理员不间断地监视和调整应用程序。

- **Internet 集成。**

SQL Server 2000 数据库引擎提供完整的 XML 支持。它还具有构成最大的 Web 站点的数据存储组件所需的可伸缩性、可用性和安全功能。SQL Server 2000 程序设计模型与 Windows DNA 构架集成，用以开发 Web 应用程序，并且 SQL Server 2000 支持 English Query 和 Microsoft 搜索服务等功能，在 Web 应用程序中包含了用户友好的查询和强大的搜索功能。

- **可伸缩性和可用性。**

同一个数据库引擎可以在不同的平台上使用，从运行

Microsoft Windows® 98 的便携式电脑，到运行 Microsoft Windows 2000 数据中心版的大型多处理器服务器。SQL Server 2000 企业版支持联合服务器、索引视图和大型内存支持等功能，使其得以升级到最大 Web 站点所需的性能级别。

➢ 企业级数据库功能。

SQL Server 2000 关系数据库引擎支持当今苛刻的数据处理环境所需的功能。数据库引擎充分保护数据完整性，同时将管理上千个并发修改数据库的用户的开销减到最小。SQL Server 2000 分布式查询使您得以引用来自不同数据源的数据，就好象这些数据是 SQL Server 2000 数据库的一部分，同时分布式事务支持充分保护任何分布式数据更新的完整性。复制同样使您得以维护多个数据复本，同时确保单独的数据复本保持同步。可将一组数据复制到多个移动的脱接用户，使这些用户自主地工作，然后将他们所做的修改合并回发布服务器。

➢ 易于安装、部署和使用。

SQL Server 2000 中包括一系列管理和开发工具，这些工具可改进在多个站点上安装、部署、管理和使用 SQL Server 的过程。SQL Server 2000 还支持基于标准的、与 Windows DNA 集成的程序设计模型，使 SQL Server 数据库和数据仓库的使用成为生成强大的可伸缩系统的无缝部分。这些功能使您得以快速交付 SQL Server 应用程序，使客户只需最少的安装和管理开销即可实现这些应用程序。

➢ 数据仓库。

SQL Server 2000 中包括析取和分析汇总数据以进行联机分析处理 (OLAP) 的工具。SQL Server 中还包括一些工具，可用来直观地设计数据库并通过 English Query 来分析数据。

2、系统结构：

1）SQL Server2000 的系统数据库

在 SQL Server2000 安装过程中，创建了四个系统数据库，这四

个系统数据库是运行 SQL Server2000 的基础，建立在这四个系统数据库中的表格定义了运行和使用 SQL Server 的规则，这个四个系统数据库分别是 master,tempdb,model,msdb。

- **master 数据库**

master 数据库记录了 SQL Server2000 所有的服务器级系统信息，所有的注册用户和密码，和所有的系统设置信息。还记录了用户定义数据库的存储位置和初始化信息。

- **tempdb 数据库**

tempdb 数据库记录了所有的临时表格、临时数据和临时创建的存储过程。存放的所有信息都是临时的，每当断开连接时，所有的临时表格和临时的存储过程都将被自动丢弃。

- **model 数据库**

model 数据库是用户建立新数据的模版，它包含将复制到用户数据库中去的系统表。

用户可以根据需要，将数据对象（如：自定义数据类型等）建立在 model 数据库中，每次建立新的数据库时，都将自动用有这些对象。

每次 SQL Server2000 启动时都将以 model 数据库为模版重建 tempdb 数据库，一旦删除 model 数据库，SQL Server2000 将无法使用。

- **msdb 数据库**

该数据库被用来排除故障。

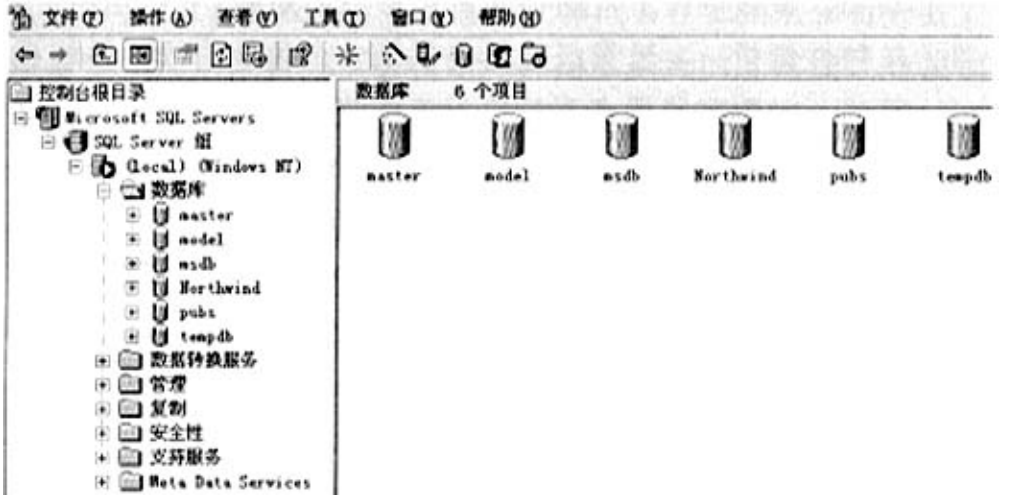
说明：数据库数据文件扩展名为 mdf，日志文件扩展名为 ldf

2）存储过程

系统存储过程是预先经过编译的 SQL 语句的集合，使用系统存储过程可以方便地查看有关数据库和数据库对象的相关信息。系统存储过程都存储在 master 数据库中，所有系统存储过程的名字都以 sp_开始，下划线后是这个系统存储过程的功能简介。

3）示例数据库

pubs 数据库是模仿一个图书出版公司建立的数据库模型。Northwind 数据库是模仿一个虚拟的贸易公司的数据库模型。如图所示：



2.2 轻量级架构

什么是轻量级架构，没有明确的定义。它具有重量级架构不具备的优势。重量级架构的典型代表就是 EJB，EJB 提供了一系列“重量级”企业级服务，并可以让你开发的组件可以很好的集成 EJB 容器所提供的企业级服务，如 JTA 等。然而对于全面的 EJB 容器，虽然给了我们看起来完整的服务策略，但是，它也给我们带来了许多负面效果。

EJB 缺点：

1. 部署复杂，运行缓慢
2. 内在服务多，启动慢
3. 规则特多，空间很小
4. 测试(调试)困难

轻量级容器的共同特征包括：

1. 基于 POJO 的编程——轻量级容器不具侵犯性。它不强迫执行任何 API。
2. 生命周期管理——轻量级容器管理放入其中的对象的生命周期。最低限度下，它们实例化并销毁对象。
3. 依赖性解析——轻量级容器提供了一个普通的依赖性解析策略。多数容器现在支持称为依赖注入的策略。还有一些支持 Java 2 平台企业版（J2EE）风格的策略，称之为服务定位。
4. 一致的配置——轻量级容器是一个便于提供一致配置服务的位置。
5. 服务关联——轻量级容器提供一种将服务与容器中的对象相关联的方法

其实，就算用 Java 建造一个不是很烦琐的 web 应用，也不是件轻松的事情。 在构架的一开始就有很多事情要考虑。 从高处看，摆在开发者面前有很多问题：要考虑是怎样建立用户接口？在哪里处理业务逻辑？ 怎样持久化的数据。 而这三层构架中，每一层都有他们要仔细考虑的。 各个层该使用什么技术？ 怎样的设计能松散耦合还能灵活改变？ 怎样替换某个层而不影响整体构架？应用程序如何做各种级别的业务处理(比如事务处理)？

构架一个 Web 应用需要弄明白好多问题。 幸运的是，已经有不少开发者已经遇到过这类问题，并且建立了处理这类问题的架构。 一个好架构具备以下几点： 减轻开发者处理复杂的问题的负担（“不重复发明轮子”)； 内部有良好的扩展； 并且有一个支持它的强大的用户团体。 好的构架一般有针对性的处理某一类问题，并且能将它做好（Do One Thing well)。 然而，你的程序中有几个层可能需要使用特定的架构，已经完成的 UI(用户接口) 并不代表你也可以把你的业务逻辑和持久逻辑偶合到你的 UI 部分。举个例子， 你不该在一个 Controller(控制器)里面写 JDBC 代码作为你的业务逻辑， 这不是控制器应该提供的。 一个 UI 控制器应该

委派给在 UI 范围之外的轻量级组件。 好的架构应该能指导代码如何分布。 更重要的是，架构能把开发者从编码中解放出来，使他们能专心于应用程序的逻辑（这对客户来说很重要）。

如何建立你的架构，并且怎样让你的各个应用层保持一致？如何整合架构以便让每个层在以一种松散耦合的方式彼此作用而不用管低层的技术细节？这对我们来说真是一种挑战。 这里讨论一个整合架构的策略(使用 3 种受欢迎的开源架构) ：表示层我们用 Struts； 业务层我们用 Spring；而持久层则用 Hibernate。 你也可以用其他 FrameWork 替换只要能得到同样的效果。

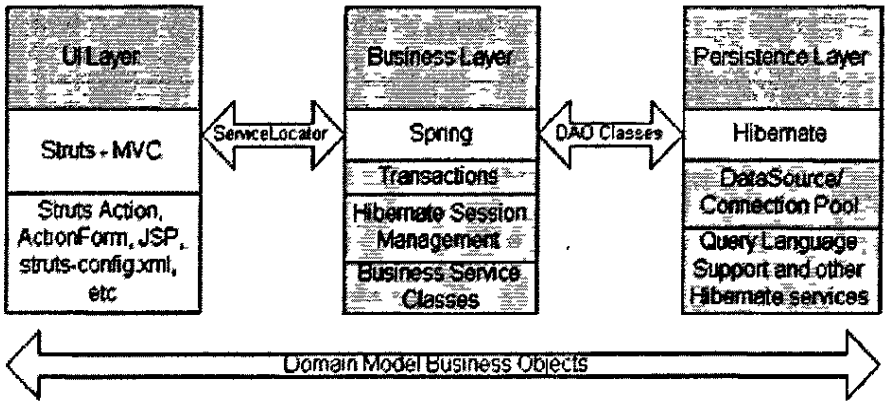


图 2.1 框架组合示意图

系统分为三层：

- UI 层：借助 Struts 实现
- 业务层：借助 SpringFramework 进行业务组件的组装关联。
- 数据持久层：借助 Hibernate 实现

采用这样的三层架构有三方面原因：

1. 通过成熟的开源产品实现各层，同自己编写代码实现，相比之下能缩短开发周期，且架构所用到的开源产品均有很广泛的用户群，经受过实践的考验，质量和性能更有保障。
2. 层与层之间松散耦合，增加代码重用率。
3. 各层分工明确，这样也利于团队的明确分工。

2.2.1 UI 层的框架

UI 是 User Interface 的缩写，这一层是面向用户的界面，是用户与系统之间交互的媒介。如，用户在界面发送请求， 系统接收请求，进行处理，然后通过界面将结果呈现于用户。这一过程包括了用户动作、数据传递、界面显示。大家熟悉的 MVC 模式就是将这三者分离，减少三者耦合。

我们在该层借助了 Struts 来实现：

1. 用 ActionForm 类封装与用户互动的数据元素。
2. 用 Action 类实现业务逻辑、动作处理、链接转向。实现 MVC 中的 C。
3. 借助 Struts 标签来完成数据呈现。实现 MVC 中的 V。

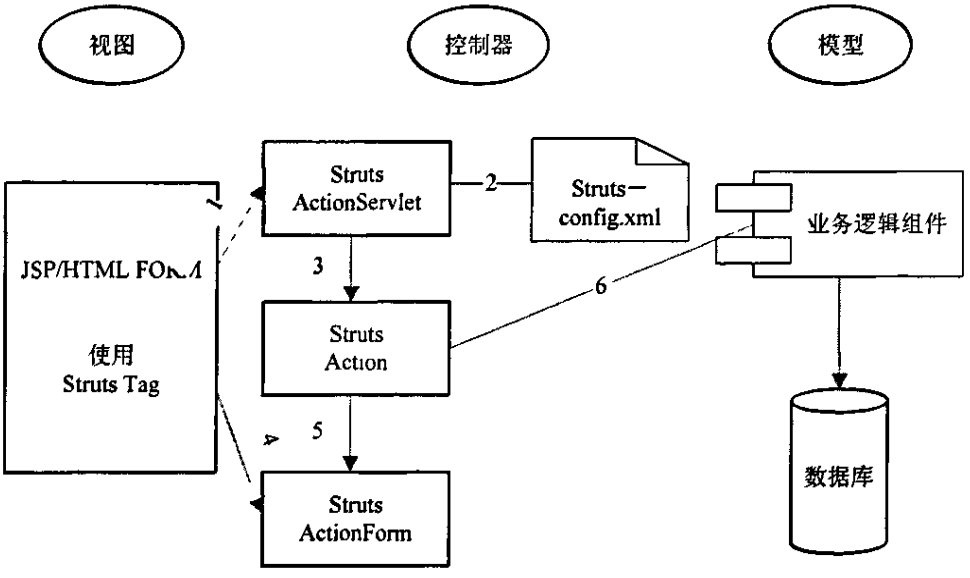


图 2.2 struts 框架

在图 2.2 中有关数字表述如下：

“1”所有浏览器请求都被提交给 StrutsAction Servlet 处理。
“2” StrutsAction Servlet 根据 struts-config.xml 文件中预先配置好的设置，选择应该将请求转到哪个 StrutsAction 子类。
“3”将请求数据传递到适当的 Action 请求处理器。
“4”当用户提交表单时，一个类型为适当的或配置好的 StrutsActionForm 的子类将被创建，并被填入表单中相应的数据。
“5” StrutsAction 子类可以从 StrutsForm 子类中获取数据，用以调用业务逻辑代码。
“6” StrutsAction 子类调用业务逻辑组件，完成业务功能。

使用 Struts 主要出于以下几方面考虑：

1. Struts 将业务数据、页面显示、动作处理进行分离，这有利各部分的维护。
2. Struts 采用 Front Controller 模式来实现动作处理，让所有的动作请求都是经过一个统一入口， 然后进行分发。这样方便我们在入口中加入一些全局控制代码的实现，如安全控制、日志管理、国际化 编码等。
3. 通过 Struts 提供的 ActionForm 封装 web form 中的元素，使重用 web 表单成为可能。
4. 借助 Struts Validator 框架帮助完成 web 层的验证工作，通常情况下我们不用再去为每个 web 页面写其验证代码，只需通 过配置即可实现。这也减少了我们的开发量，由于验证代码的集中管理，也为维护带来便利。

2.2.2 业务层的框架

在实际的项目开发中，每个领域都会有自己独特的业务逻辑，

正因为这样，致使项目中代码高度耦合，原本有可能被重用的代码 或功能，因为与具体的业务逻辑绑定在一块而导致很难被重用。因此我们将实现这些具体逻辑的代码抽取出来分为单独的一层， 其目的是希望通过层，来降低它与系统其他部分的耦合度。

现实中世界是变化的，既然该层实现的是现实中具体的业务逻辑，那该层的实现代码不可避免的会发生变更。怎样让该层适应 最大的变化，做到最小的改动?通常我们在编码的时候会尽量考虑到同一业务多种实现的兼容和可扩展的能力。因此我们在 该层借助了 Spring，通过依赖注入、AOP 应用、面向接口编程，来降低业务组件之间的耦合度，增强系统扩展性。

Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式

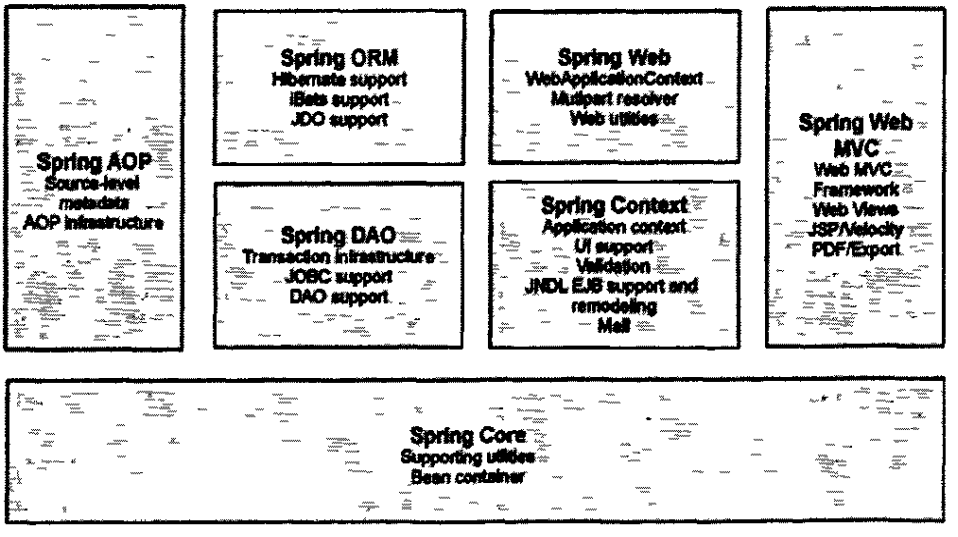


图 2.2 Spring 框架的 7 个模块

2.2.3 持久化层的框架

开发中与数据库进行数据交互必不可少，通常我们归为 CRUD

（添加、读取、修改、删除），这些操作占据了系统开发中大部分的时间， 同时我们还需要考虑与数据库交互的性能问题，如连接池、数据缓存等等。因此该层实现我们借助了 Hibernate。Hibernate 是一个 ORM 工具，它不仅仅是实现了数据库访问性能优化和与数据库交互的常用操作（CRUD），还将数据表与对象进行了 关联，让我们可以脱离数据表，而直接针对对象来与数据库交互，我们不再需要用字符串去描述表中字段，不再需要一个个”+“号去 组装 Sql 语句。这使得编码中可书写性提高。

Hibernate 是一个功能强大，可以有效地进行数据库数据到业务对象的 O/R 映射方案。Hibernate 推动了基于普通 Java 对象模型，用于映射底层数据结构的持久对象的开发。通过将持久层的生成自动扩展到一个更大的范围，Hibernate 使开发人员专心实现业务逻辑而不用分心于繁琐的数据库方面的逻辑，同时提供了更加合理的模块划分的方法。

第3章 系统需求分析

3.1 系统的总体需求

3.1.1 总体目标及任务

与其他商务一样，网上购物的总体目标是使顾客满意，希望他们再次惠顾。而要使网上的顾客满意，掌握网上销售的技巧就尤为重要。与传统的销售方式不同，顾客在网上购物时只能根据网站上提供的内容来进行选择，如果他们觉得不满意就会马上离开。因此，创建合适的销售内容和提供完善的客户服务方式是进行网上销售获得成功的关键因素。要把握客户的心理，使用客户满意的文本、图像及交互式对话去创建销售内容，可使自己的网站在众多站点中脱颖而出。综合运用一些网上的资源及一些进行网上销售的技巧，从而可以最少的投入去获得最大的收益。

3.2 系统功能需求分析

3.2.1 用户模块

1. 进行注册，填写相应信息，包括：（必填）用户名、密码、邮箱；其他信息（可选）真实姓名，地址、电话等
2. 对自己的信息进行修改（除用户名外）
3. 进行登陆后，可以将所选商品放入购物车，可以对商品进行评论

4. 选购商品（要求在登陆后）（见后详细）
5. 放入、查看购物车（要求在登陆后）（见后详细）
6. 查看新闻

3.2.2 选购商品模块

1. 页面包括
一级页面：首页
包括货品分类导航条，用户操作导航条，用户登录或信息区块，商城热讯区块，分类导航区块，搜索区块，统计信息区块，热卖商品区块，人气商品区块，推荐商品区块，最新商品区块，精彩推荐区块，页头区块，页尾区块。

二级页面：货品分类页（列出一类商品）
包含子分类描述，货品分类导航条，用户操作导航条，用户登录或信息区块，商城热讯区块，分类导航区块，子目录列条/商品条目。

三级页面：产品详细介绍（列出一件商品）
包含子分类描述，货品分类导航条，用户操作导航条，用户登录或信息区块，商城热讯区块，分类导航区块，基本信息，商品简介，相关商品，相关评论
新闻列表页以及新闻查看页：
查看新闻
功能：
1. 通过以上页面察看商品
2. 登陆后 放入购物车

3.2.3 购物车模块

1. 客户察看所选购的商品-
2. 对所选购的商品进行管理（删除）
3. 继续购买（跳回选购页面）
4. 确认所选商品后，填写相关的送货信息， 用户名、地址、电话（多个：固定电话、手机、其他电话；至少填一个）、联系人（考虑和用户名不同）、邮编、送货方式等有用信息
5. 生成订单

3.2.4 管理员模块

1. 超级管理员：可以对其他管理员管理，及所有操作
2. 客户管理员：
   1)修改客户信息，
   2)删除客户
   3)修改客户为 VIP
3. 商品管理员：
   A.商品信息管理
      1. 添加新商品
      2. 加数量
      3. 修改所属区块（货架）
      4. 修改商品其他信息
   B.新闻信息管理
      1. 添加新闻（设置新闻类别）
      2. 修改删除新闻

3． 添加新闻类别

C.发布公告

1．发布及时公告（放在 application）

D.商品评论信息管理

1． 删除部分评论信息

E.订单管理：

对订单进行管理（删除订单），对 订单的送货信息进行修改

对订单执行发货（选做：考虑打印订单）

完成的订单，标志完成；对订单执行时间 进行纪录。

3.3 系统要求

3.3.1 功能要求

1. 用户注册要验证输入信息的合法性，如：用户名只能是字母和数字的组合并不得超过 16 位，电话必须是数字等
2. 用户查看自己的所有订单 可以删除未发货的订单。其他订单可以查看所处状态
3. 密码忘记后可以取回：通过注册时填写的信息，找回密码的提问、答案。未登陆时显示找回密码按钮

找回密码时：填写用户名、提问、答案，返回密码（最好发至邮箱）

4. 会员要分级别。级别不同，折扣率不同
5. 管理员分权限。权限不同工作不同，

1）超级管理员：可以注册新管理员，删除管理员，以及其他

所有工作

2）商品管理员：对商品信息进行管理（修改商品所有信息、商品所属区块、折扣；新闻管理；订单管理；公告管理，）

3）客户管理员：对客户进行管理，（删除会员，修改会员级别）

3.3.2 页面信息要求

页面信息内容：

1) 商品信息：

商品 ID、商品名称、商品类别、照片、商品产地、品牌、所属区块、市场价、会员折扣、VIP 会员折扣、简单介绍、详细说明、添加时间、商品总数量、已下订单待送货商品数量等

2) 商品分类包括：

a)按品牌：联想、Sony、Appple、爱国者、奈克、波导、IBM、三洋、四通等

b)按类别：电脑、笔记本、摄象机、掌上电脑、打印机、生活用品等

c)按区块（货架）：热卖商品、人气商品、推荐商品、最新商品、精彩商品、推荐商品、促销商品等

3) 订单信息：订单号、客户名、订货时间、配送方式、联系人、电话、移动电话（至少填写一个电话）、送货地址、订单状态、完成时间、订单总价格、邮编、商品名称、商品数量、成交价格(考虑多种商品) 等

4) 订单配送方式：如：货到付款、邮政快递等

5) 订单状态：如：删除、生成、送货、完成等

6) 新闻信息：发布时间、主题、内容、新闻类别 ID、新闻图片等

7) 新闻分类：体育、电脑、娱乐、汽车、生活等

8) 评论信息：被评论商品、发布时间、发布人(客户)、内容等
9) 所有级别、分类、状态、级别、配送方式都要求可以添加
10) 订单号：使用，日期时分秒随机数组合（20051030183659fgtw）
11) 执行订单时，注意修改商品数量

3.3.3 系统开发环境

PC 机（P4 2G/512M 内存/30G 硬盘)
数据库软件（SQL Server2000）
中间件（TOMCAT 5.0）
操作系统（Windows2000）
开发工具（JBuilder2006、Eclipse）
建模工具（Microsoft Office Visio2003）

3.3.4 运行要求

硬件要求：
PC 机 CPU： 奔腾 III 800
内存：128M
硬盘：10G

软件要求：
操作系统：Windows2000、Windows XP
浏览器：Internet Explorer6.0

第4章 系统设计

根据以上的需求分析结果，我们将进行系统设计，包括建立相应的数据表，模块数据流程设计、页面设计等。

4.1 数据库设计

4.1.1 数据库表的设计

1. 客户信息表

UserInfo

字段名	类型	约束	名称	说明
userId	整形，自动增长	非空，主键	客户 ID	PK
userName	字符型(50)	非空	客户名	
userPass	字符型(50)	非空	密码	
userRealName	字符型(50)	空	真实姓名	空
userTel	字符型(50)	空	电话	空
userCellPhone	字符型(50)	空	移动电话	空
userEmail	字符型(50)	空	E-mail	空
userAddress	字符型	空	地址	空

	(100)			
userLevel	整形	非空	级别	0 普通会员 1 VIP会员

2. 管理员信息表

EmployeeInfo

字段名	类型	约束	名称	说明
employeeID	整形,自动增长	非空,主键	管理员 ID	PK
employeeName	字符型(50)	非空	管理员名	
employeePass	字符型(50)	非空	密码	
dutyID	整形	非空	权限 ID	FK

3. 管理员权限表

DutyInfo

字段名	类型	约束	名称	说明
dutyID	整形,	非空,主键	权限 ID	PK
duty	字符型(50)	非空	权限	3 种

4. 商品类别表

ProductSort

字段名	类型	约束	名称	说明
productSortID	整形,自动增长	非空,主键	商品类别 ID	PK
productSortName	字符型(50)	非空	商品类别名称	

5. 商品基本信息

ProductInfo

字段名	类型	约束	名称	说明
productID	整形,自动增长	非空,主键	商品 ID	PK
productName	字符型(50)	非空	商品名称	
productSortID	整形	非空	商品类别 ID	FK
productImg	字符型(50)	非空	照片	
productManufactorer	字符型(50)	非空	商品产地	
ProductBrandID	整型	非空	品牌 I D	Fk
ProductFieldID	整型	非空	所属区块 ID	FK
productPrice	Float	非空	市场价	
productAgio	Float	默认 1	会员折扣	
ProductAgioVIP	Float		VIP 会员折扣	
productIntrod	字符型(50)	非空	简单介绍	
productDetail	字符型(200)	非空	详细说明	
productAmount	整形	非空	商品总数量	
ProductOrderAmount	整形	默认为 0	待发商品数量	
AddTime	日期形	默认系统时间	添加时间	

6. 商品区块表

ProductField

字段名	类型	约束	名称	说明
productFieldId	整型，自动增长	非空	所属区块ID	Pk
ProductFieldName	字符型(50)	非空	所属区块名称	

7. 商品品牌表

ProductBrand

字段名	类型	约束	名称	说明
productBrandID	整型，自动增长	非空	所属品牌ID	Pk
ProductBrandName	字符型(50)	非空	所属品牌名称	

8. 订单表

OrderInfo

字段名	类型	约束	名称	说明
orderid	字符型(50)	非空,主键	订单ID	时间毫秒数
userId	整形	非空	客户ID	FK
orderTime	日期形	默认系统时间	订货时间	
deliverTime	日期形	默认为空	要求送货时间	

DeliverWayID	Int	非空	配送方式	FK
userRealName	字符型(50)	非空	联系人	
telPhone	字符型(50)	任意一个	电话	任意一个
cellPhone	字符型(50)		移动电话	
deliverAddress	字符型(50)	非空	送货地址	
statusID	整形	非空	订单状态	Fk，默认 1
fulfillTime	日期形	默认为空	完成时间	
orderTotle	Float	默认为空	商品总价格	
post	字符型(50)	非空	邮编	

9. 订单配送方式表

DeliverWayType

字段名	类型	约束	名称	说明
deliverWayID	整型,自动增长	非空	配送方式 ID	Pk
deliverWayName	字符型(50)	非空	配送方式名称	

10. 订单状态表

OrderStatusType

字段名	类型	约束	名称	说明
statusID	整型,自动增长	非空	订单状态 ID	Pk
statusName	字符型(50)	非空	订单状态名称	

11. 订货表详细信息

IndentInfo

字段名	类型	约束	名称	说明

indentedID	整形,自动增长	非空,主键	ID	PK
orderId	字符型(50)	非空	订单 ID	FK
productID	整形	非空	商品名称 ID	FK
productAmount	整形	非空	商品数量	
sellPrice	Float	非空	成交价格	

12. 新闻信息表

newsInfo

字段名	类型	约束	名称	说明
newsId	整型，自动增长	非空	Id	
newsPostTime	日期	非空	发布时间	
newsTitle	字符型(50)	非空	主题	
newsContent	字符型(500)	非空	内容	
newsTypeID	整形	非空	新闻类别 ID	FK
newsPrice	字符型(50)	空	新闻图片	

13. 新闻类别表

newsType

字段名	类型	约束	名称	说明
newsTypeID	整形,自动增长	非空,主键	新闻类别 ID	PK
newsType	字符型(50)	非空	新闻类别	

14. 会员评论

FeedBack

字段名	类型	约束	名称	说明
feedBackID	整型,自动增长	非空	Id	PK
productID	整型	非空	被评论商品id	FK
feedBackPostTime	日期	非空	发布时间	
userID	整型	非空	发布人(客户)ID	FK
feedBackContent	字符(200)	非空	内容	

4.1.2 数据库关系模型图

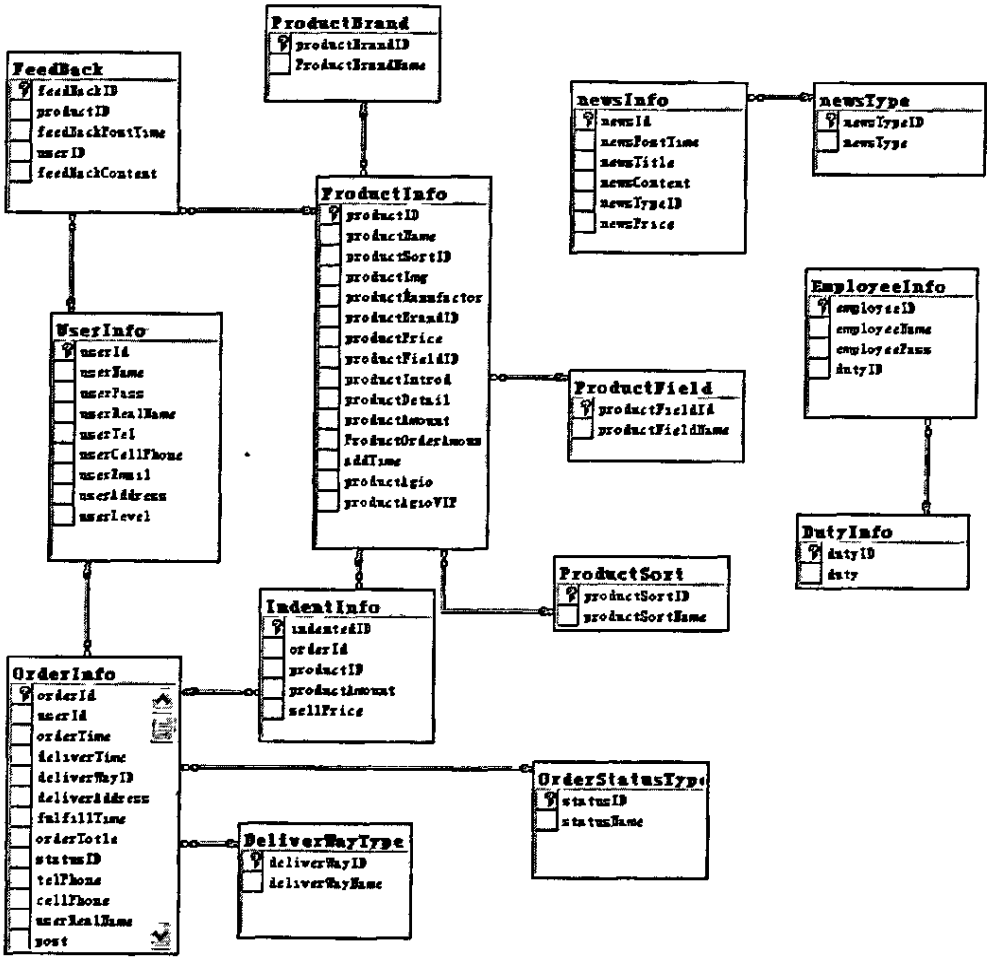


图 4.1 关系模型图

4.2 数据流程设计

4.2.1 新闻模块流程图

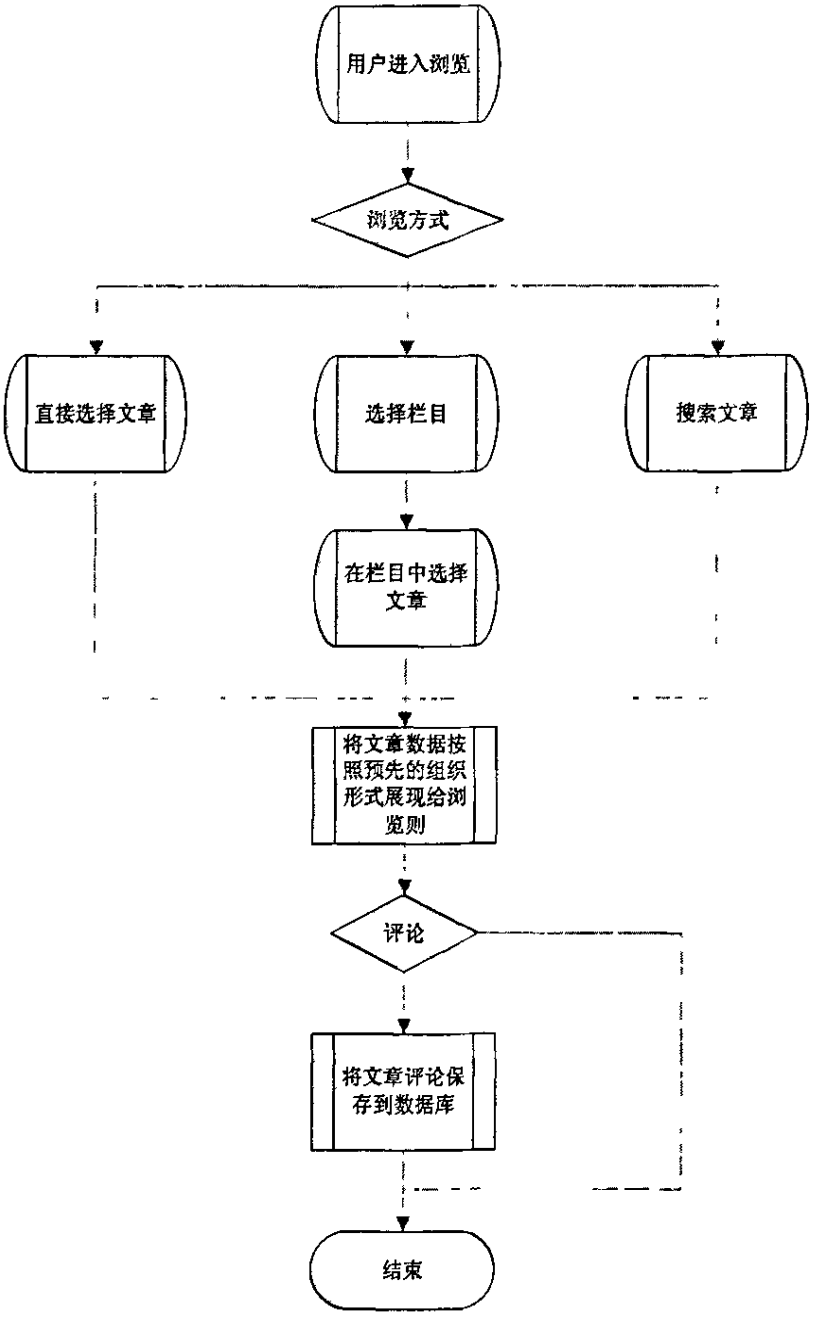


图 4.2 前台用户模块流程图

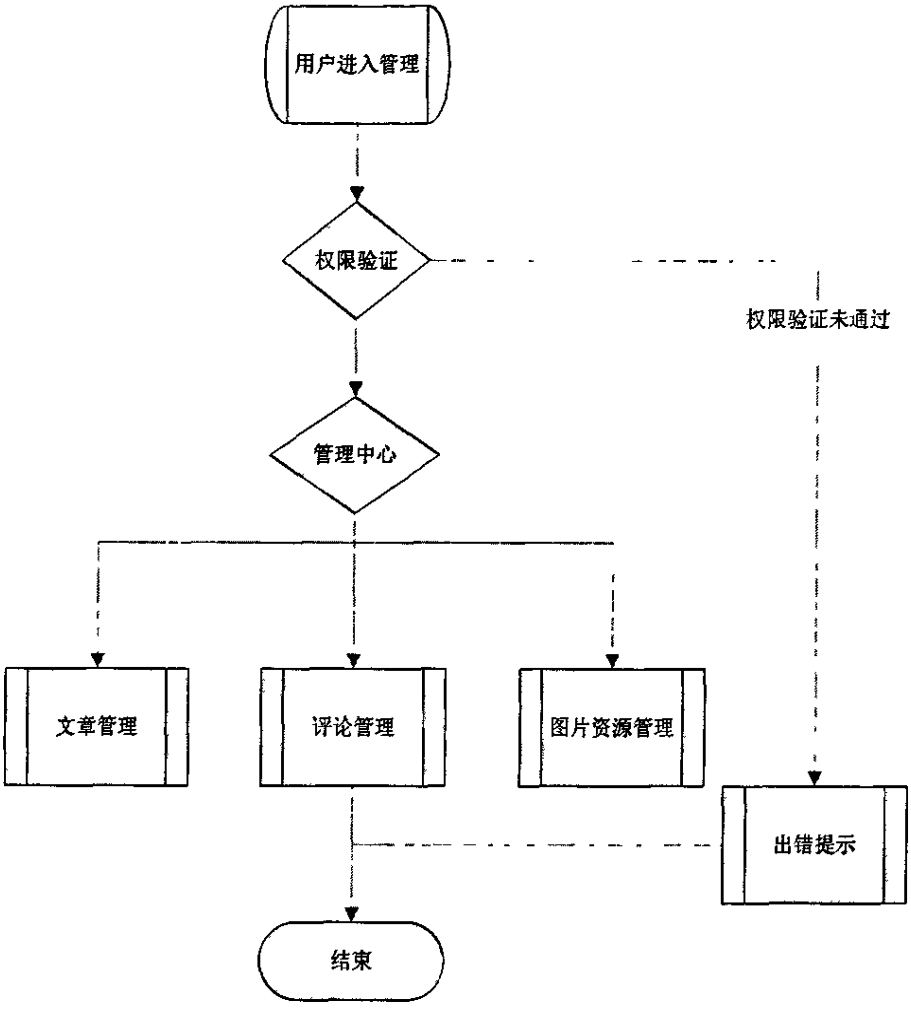


图 4.3 后台用户模块流程图

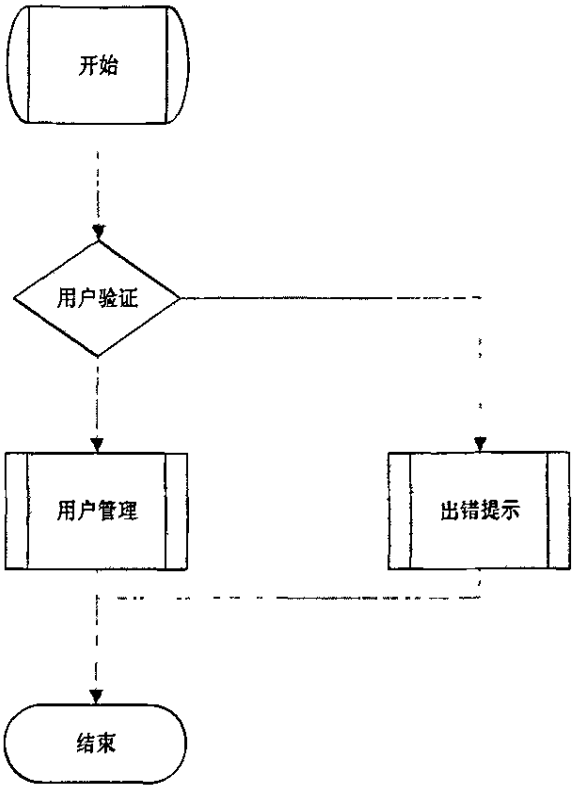


图 4.4　管理员模块流程图

4.2.2　用户模块流程图

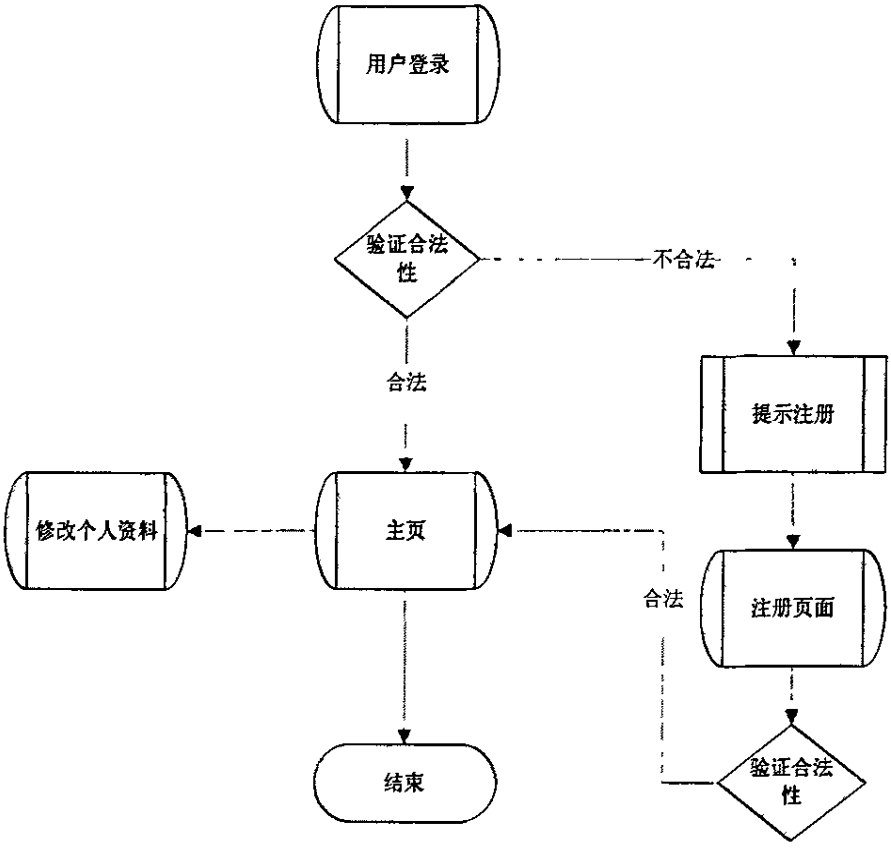


图 4.5 用户流程图

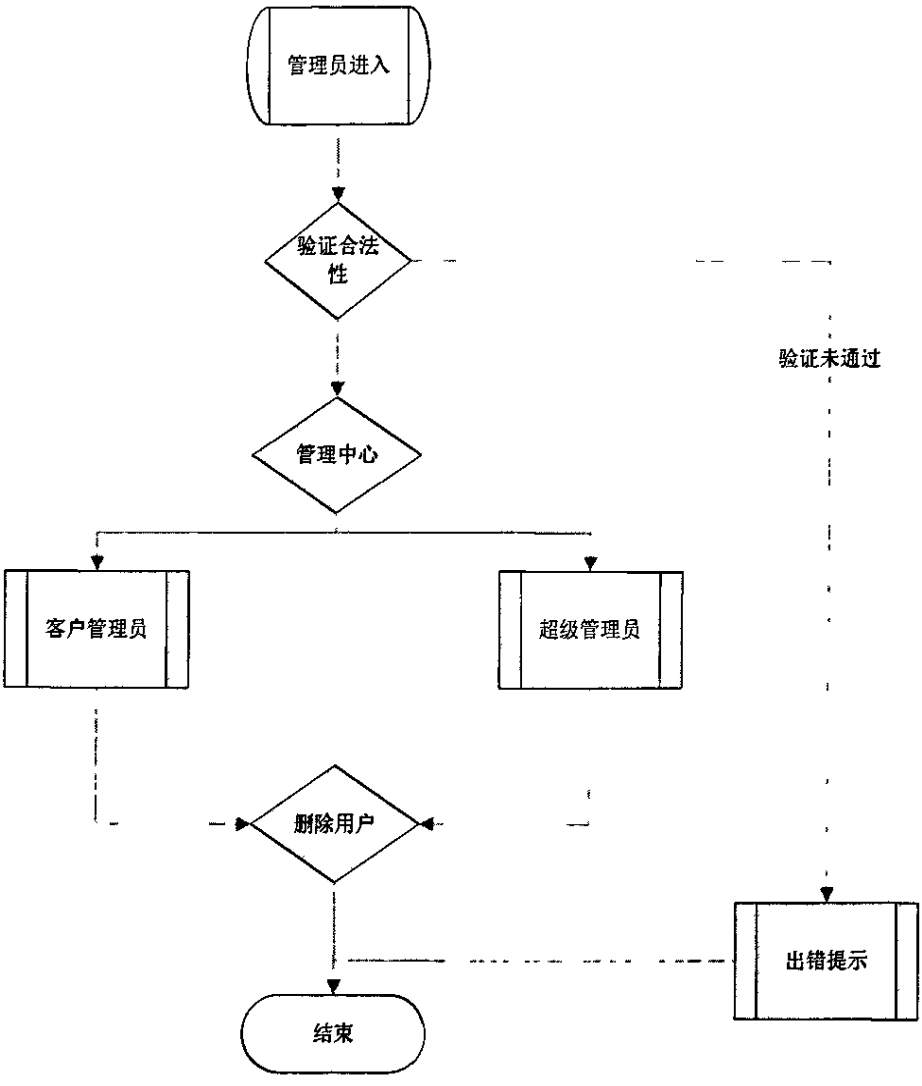


图 4.6 管理员流程图

4.3 购物系统新闻模块页面设计

页面包括：

新闻标题，新闻内容概要，发表时间和类型（体育、电脑、娱乐、汽车、生活等），并且页面还能根据新闻的标题及类型查询相关新闻。如下图：

新闻列表

按 新闻标题 查询 请输入关键字 查询

新闻标题	新闻内容概要	发表时间	类型
◇周末大优惠	IBMThinkPad T42p	2005-10-13	行业
◇神舟优雅12宽屏	神舟优雅12宽屏Bothao赛扬	2005-10-13	行业
◇IBM笔记本全线	IBM笔记本全线特价,T42p	2005-10-13	行业
◇言文秀 VT33	言文秀 VT3300最新上市仅售	2005-10-13	行业
◇汉王手写输入系统 .	汉王手写输入系统，让您摆脱键盘的	2005-10-13	行业
◇足球大赛	足球大赛 足球改成篮球大小啦，足	2005-10-13	体育
◇BTV-8生活出	BTV-8生活出新招,BTV-8	2005-10-13	生活
◇CPU大卖	CPU P43 0 大卖 500	2005-10-13	电脑
◇小虎队重出江湖	小虎队重出江湖小虎队重出江湖小虎	2005-10-13	娱乐
◇国产汽车大卖，进 .	国产汽车大卖，进口车不要钱,国产	2005-10-13	汽车

[ ] | [ ] | [下一页] | [末页]

图 4.7 购物系统新闻模块页面

第5章 实现方案

在整体设计完成后，就是具体的程序实现。本章将已用户模块为例，给出采用轻量级架构后，各层次的程序实现。

5.1 业务层的实现

1．声明 Customer 接口，描述了顾客这一实体

```
package com.bean;
 public interface CustomerInterface {
public String getId();
public void setId(long id);
public String getEmail();
 public void setEmail(String email);
public String getAddress ();
public void setAddress (String address);
public String getPassword();
public void setPassword(String password);
public String getTelphone () ;
public void setTelphone (String telphone);
public String getUserName();
public void setUserName(String userName);
public String getRealname ();
public void setRealname (String realname);
public String getCellphone ();
public void setCellphone (String cellphone);
public String getLevel ();
```

```
 public void setLevel (String level);
 }
```

2. 声明 CustomerManage 接口，描述了顾客所提供的有关服务

```
package com.bean;
import java.util.Collection;
import java.util.List;
public interface CustomerManage {
 public Customer newCustomer();                          //创建一个新的用户
 public Customer read(String id);                        //根据主键 id 读取一个用户
 public List query(String scope,Collection paras);       //用户查询
 public Customer readByName(String userName); //根据用户名读取一个用户
 public Customer login(String userName,String password);//用户登录
 public boolean save(Customer cus);                      //保存用户
 public boolean update(Customer cus);                    //修改用户
 public boolean del(Customer cus);                       //删除用户
}
```

3. 上面定义了接口，下面来实现这个接口

Spring 中提供了 hibernate DAO，可以让使用 hibernate 的朋友们更加简单使用 hibernate 访问数据库，该示例中把 Service 及 DAO 合并在一起。全部 UserDao 的代码如下：

```
package com.dao;
import java.util.Collection;
import java.util.List;
import
org.springframework.orm.hibernate3.support.HibernateDaoSupport;
```

```
import com.bean.*;
public class CustomerDao extends HibernateDaoSupport
    implements CustomerManage {
//查找用户的方法
public List query(String scope,Collection paras) {
  return this.getHibernateTemplate().find("from User where
"+scope,paras.toArray());
 }
 public Customer login(String userName, String password) {
  Customer cus=readByName(userName);
  if(cus==null || user.getPassword()==null
||(!cus.getPassword().equals(password)))
//用户不存在或者密码不正确
  {
  return null;
  }
  return cus;
 }
 public Customer new Customer () {
  return new Customer ();
 }
//添加用户的方法
public boolean save(Customer cus) {
  boolean ret=true;
  try{
  this.getHibernateTemplate().save(Customer);
  }
  catch(Exception e)
  {
   ret=false;
```

```
    }
    return ret;
  }
//修改用户的方法
} public boolean update(Customer cus) {
  boolean ret=true;
  try{
  this.getHibernateTemplate().update(cus);
  }
  catch(Exception e)
  {
   e.printStackTrace();
   ret=false;
  }
  return ret;
 }
//删除用户的方法
 public boolean del(Customer cus) {
    boolean ret=true;
    try{
    this.getHibernateTemplate().delete(cus);
    }
    catch(Exception e)
    {
     ret=false;
    }
    return ret;
 }
```

从上面程序中我们看到，CustomerDao 类继承于 Spring 的 HibernateDaoSupport 类，目的在于方便通过 hibernate 操作数据库，同时实现了 CustomerManage 接口中的相关功能。

4. 配置 spring

下面看看 Spring 的配置文件内容，本例中是 applicationContext.xml，通过这个配置文件把业务逻辑，持久化层和表示层关联、整合起来。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
<bean id="dataSource"
    class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
 <property name="driverClassName">
  <value>com.microsoft.jdbc.sqlserver.SQLServerDriver</value>
 </property>
 <property name="url">
<value>jdbc:microsoft:sqlserver://localhost:1433;DataBaseName=Cu
stomerManager</value>
 </property>
 <property name="username">
  <value>sa</value>
 </property>
 <property name="password">
  <value></value>
 </property>
</bean>
<bean id="mySessionFactory"
```

```
class="org.springframework.orm.hibernate3.LocalSessionFactoryBea
n" singleton="true">
 <property name="dataSource">
  <ref local="dataSource" />
 </property>
 <property name="mappingResources">
  <list>
   <value>com/hibernate/Customer.hbm.xml</value>
  </list>
 </property>
 <property name="hibernateProperties">
  <props>
   <prop
key="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</pr
op>
    <prop key="hibernate.show_sql">true</prop>
  </props>
 </property>
</bean>
<bean  id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionMa
nager">
 <property name="sessionFactory">
  <ref local="mySessionFactory" />
 </property>
</bean>
<bean id="customerManage" class="com.dao.CustomerDao">
 <property name="sessionFactory">
  <ref local="mySessionFactory" />
 </property>
```

```
</bean>
</beans>
```

其中 dataSource 部分是配置的数据源,如用户名及密码等，mySessionFactory 部分主要针对 hibernate 的配置文件，userService 部分针对 hibernateDao 的配置，事务的部分没用到。

当然，要让 Spring 跟 Java Web 应用集成，还需要配置一下 web.xml 文件，下面的 web.xml 会提到，这里先单独拿出来：

```
<context-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
<listener> <listener-class>org.springframework.web.context.Contex
tLoaderListener</listener-class>
</listener>
```

通过配置，就可以把刚才我们的几个部分集成到了一起。换句话说，一旦我们需要修改系统的某一部份(比如数据库中间件不能用 hibernate，而要用 iBatis、EasyDBO 或者别的)，也只需要直接在配置文件中修改即可，而系统的其它部分仍然保留不变。而且各部分的测试也相对独立，因此可以测试到更多的东西(测试工作量虽然变大，但是出错的机会就少，维护成本就低了)。这即是传说中配置编程，也是面向接口编程乃至面向对象编程的精华之所在。

5.2 持久化层的实现

若要使用 Hibernate 需要以下的工具包：

Hibernate3.jar

Antlr-2.7.4.jar

Cglib-full02.0.2.jar

Asm.jar

Asm-attrs.jars

Commons-colections-2.1.1.jar

Commons-logging-1.0.4.jar

Dom4j.jar

Ehcache-1.1.jar

Jta.jar

Log4j-1.2.9.jar

1. 创建 Hibernate 的配置文件

Hibernate从其配置文件中读取和数据库连接有关的信息，Hibernate 的配置文件有两种姓氏：一种是 XML 格式的文件；还有一种是 Java 属性文件，下面介绍如何以 XML 格式的文件来创建 Hibernate 的配置文件，这种配置文件的默认文件名为 hibernate.cfg.xml。如下图所示

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property name="show_sql">true</property>
                <property name="connection.driver_class">
            com.microsoft.jdbc.sqlserver.SQLServerDriver
        </property>
                <property name="connection.url">
    jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=CustomerManager
        </property>
                <property name="connection.username">
        sa
        </property>
                <property name="connection.password">
        </property>
                <property name="dialect">
            org.hibernate.dialect.SQLServerDialect
        </property>
                <mapping resource="Customer.hbm.xml"/>
        </session-factory>
</hibernate-configuration>
```

图 5.1 hibernate.cfg.xml

以上 hibernate.cfg.xml 文件包含了一系列属性及其属性值，Hibernate 将根据这些属性来连接数据库，上图为连接 SQL Server 数据库的配置代码。下表对 hibernate.cfg.xml 文件中的所有属性做了描述。

表 6.1 Hibernate 配置文件的属性

show_sql	如果为 true，表示在程序运行时，会在控制台输出 SQL 语句，这有利于跟踪 Hibernate 的运行状态，默认为 false。
connection.driver_class	指定数据库的驱动程序
connection.url	指定连接数据库的 URL
connection.username	指定连接数据库的用户名
connection.password	指定连接数据库的口令
dialect	指定数据库使用的 SQL 方言

2. 创建持久化类

持久化类是指其实例需要被 Hibernate 持久化到数据库中的类。持久化类通常都是域模型中的实体域类。持久化类符合 JavaBean 的规范，包含一些属性，以及与之对应的 getXXX() 和 setXXX()方法，下面定义了一个 Customer 的持久化类，是对顾客这一实体的描述：

```
package com.bean;
public class Customer   implements CustomerInterface {
    private Long id;
    private String username;          //用户名
    private String password;          //密码
    private String realname;        //真实姓名
    private String telphone;          //座机号码
    private String cellphone;       //手机号码
```

```
private String email;          //电子邮箱
private String address;          //地址
private String level;          //操作权限
public Customer(){}

public Long getId() {
    return id;
}
public String getUsername() {
    return username;
}
public String getPassword() {
    return password;
}
public String getRealname() {
    return realname;
}
public String getTelphone() {
    return telphone;
}
public String getCellphone() {
    return cellphone;
}
public String getEmail() {
    return email;
}
public String getAddress() {
    return address;
}
public String getLevel() {
```

```
        return level;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public void setRealname(String realname) {
        this.realname = realname;
    }
    public void setTelphone(String telphone) {
        this.telphone = telphone;
    }
    public void setCellphone(String cellphone) {
        this.cellphone = cellphone;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public void setLevel(String level) {
        this.level = level;
    }
}
```

Hibernate 要求持久化类必须提供一个不带参数的默认构造方法，程序运行时，Hibernate 运用 Java 反射机制，调用 Java.lang.reflect.Constructor.newInstance()方法来构造持久化类的实例。在 Customer 类中没有引入任何 Hibernate API，Customer 类不需要继承 Hibernate 的类或实现 Hibernate 的接口，这提高了持久化类的独立性。

3. 创建对象一关系映射文件

Hibernate 采用 XML 格式的文件来指定对象和关系数据之间的映射。在运行时，Hibernate 将根据这个映射文件来生成各种 SQL 语句。下面将与系统中的用户表为例，把 Customer 类映射到 UserInfo 表，创建一个名为 Customer.hbm.xml 的文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
  <class name="package.Customer" table="UserInfo">
    <id name="id" column="userID" type="java.lang.Long">
      <generator class="increment">      </generator>
    </id>
    <property name="username" type="java.lang.String" column="userName" not-null="true"/>
    <property name="password" type="java.lang.String" column="userPass" not-null="true"/>
    <property name="realname" type="java.lang.String" column="userRealName" not-null="true"/>
    <property name="telphone" type="java.lang.String" column="userTel"/>
    <property name="cellPhone" type="java.lang.String" column="userCellPhone"/>
    <property name="email" type="java.lang.String" column="userEmail"/>
    <property name="address" type="java.lang.String" column="userAddress"/>
    <property name="level" type="java.lang.Long" column="userLevel" not-null="true"/>
  </class>
</hibernate-mapping>
```

图 5.2 Customer.hbm.xml

上图中 Customer.hbm.xml 文件用于映射 Customer 类。如果需要映射多个持久化类，那么既可以在同一个映射文件中映射所有类，也可以为每个类创建单独的映射文件，映射文件和类同名，扩展名为“hbm.xml”，后一种做法更值得推荐，因为在团队开发中，这有利于管理和维护映射文件。

下表为 Customer 类的属性的 Java 类型、Hibernate 映射类型，以及 UserInfo 表的字段的 SQL 类型这三者之间的对应关系。

表 6.2 Java 类型、Hibernate 映射类型以及 SQL 类型之间的对应关系

Customer 类的属性	Java 类型	Hibernate 映射类型	UserInfo 表的字段	SQL 类型
id	long	long	userID	int
username	String	String	userName	Varchar(50)
password	String	String	userPass	Varchar(50)
realname	String	String	userRealName	Varchar(50)
telphone	String	String	userTel	Varchar(50)
cellphone	String	String	userCellPhone	Varchar(50)
email	String	String	userEmail	Varchar(50)
address	String	String	userAddress	Varchar(50)
level	long	long	userLevel	int

Hibernate 采用 XML 文件来配置对象－关系映射，有以下优点：

- Hibernate 既不会渗透到上层模型中，也不会渗透到下层数据模型中。
- 软件开发人员可以独立设计域模型，不必强迫遵守任何规范。
- 数据库设计人员可以独立设计数据模型，不必强迫遵守任何规范。
- 对象－关系映射不依赖于任何程序代码，如果需要修改对象－关系映射，只需修改 XML 文件，不需要修改任何程序，提高了软件的灵活性，并且使维护更加方便。

4. 通过 Hibernate API 操纵数据库

Hibernate 对 JDBC 进行了封装，提供了更加面向对象的 API。上面的 CustomerDao 类（具体详见业务层 Customer 接口的实现）即是通过 Hibernate API 对 Customer 对象进行持久化的操作。

上面这个例子演示了通过 Hibernate API 访问数据库的一般流程。首先应该在应用的启动阶段对 Hibernate 进行初始化，然后就可以通过 Hibernate 的 Session 接口来访问数据库。初始化过程结束后，就可以调用 SessionFactory 实例的 openSession()方法来过的 Session 实例，然后通过它执行访问数据库的操作。Session 接口提供了操纵数据库的各种方法：如上例中：

- Save()方法：把 Java 对象保存数据库中。
- Update()方法：更新数据库中的 Java 对象。
- Delete()方法：把 Java 对象从数据库中删除。
- Load()方法：从数据库中加载 Java 对象。

Session 是一个轻量级对象。通常将每一个 Session 实例和一个数据库事务绑定，也就是说，每执行一个数据库事务，都应该先创建一个新的 Session 实例。如果事务之行中出现异常，应该撤销事务。不论事务执行成功与否，最后都应该调用 Session 的 close()方法，从而释放 Session 实例占用的资源。

5.3 UI 层的实现

1. 创建一个 ActionForm，用来捕获用户输入的数据，并返回动态显示给用户的数据

```
package com.form;
import org.apache.struts.action.ActionForm;

public class CustomerForm extends ActionForm {
 private Long id;
```

```
private String username;          //用户名
private String password;          //密码
private String realname;          //真实姓名
private String telphone;          //座机号码
private String cellphone;         //手机号码
private String email;             //电子邮箱
private String address;           //地址
private String level;             //操作权限
public Customer(){}

public Long getId() {
    return id;
}
public String getUsername() {
    return username;
}
public String getPassword() {
    return password;
}
public String getRealname() {
    return realname;
}
public String getTelphone() {
    return telphone;
}
public String getCellphone() {
    return cellphone;
}
public String getEmail() {
    return email;
```

```
}
public String getAddress() {
    return address;
}
public String getLevel() {
    return level;
}
public void setId(Long id) {
    this.id = id;
}
public void setUsername(String username) {
    this.username = username;
}
public void setPassword(String password) {
    this.password = password;
}
public void setRealname(String realname) {
    this.realname = realname;
}
public void setTelphone(String telphone) {
    this.telphone = telphone;
}
public void setCellphone(String cellphone) {
    this.cellphone = cellphone;
}
public void setEmail(String email) {
    this.email = email;
}
public void setAddress(String address) {
    this.address = address;
```

```
        }
        public void setLevel(String level) {
            this.level = level;
        }
}
```

2. 创建 Action

```
package com.action;
import java.util.ArrayList;
import java.util.Collection;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.*;
import org.springframework.web.context.WebApplicationContext;
import
org.springframework.web.context.support.WebApplicationContextUt
ils;
import com.bean.*;
import com..form.CustomerForm;
public class CustomerAction extends Action {
 private CustomerManage customerManage;
 public CustomerManage getCustomerManage () {
  return customerManage;
 }
 public void set CustomerManage (CustomerManage
customerManage)
 {
  this.customerManage = customerManage;
 }
 public ActionForward execute(ActionMapping mapping,
```

```
  ActionForm form,
 HttpServletRequest request,
  HttpServletResponse response) throws Exception {
 WebApplicationContext  wac
=WebApplicationContextUtils.getRequiredWebApplicationContext(t
his.getServlet().getServletContext());
 this.        customerManage        =        (CustomerManage)
wac.getBean("customerManage ");
 String command=request.getParameter("easyJWebCommand");
if("add".equals(command))
 {
  return doAdd(mapping,form,request,response);
 }
public ActionForward doAdd(ActionMapping mapping,
  ActionForm form,
  HttpServletRequest request,
  HttpServletResponse response)
{
 Customer obj=form2Po(form);
 if(obj==null)
 {
 request.setAttribute("msg","无法创建要保存的对象，添加失败！
");
 return mapping.getInputForward();
 }
    if(userService.save(obj))
    {
     request.setAttribute("msg","用户添加成功！ ");
    }
    else
```

```
    {
     request.setAttribute("msg","用户添加失败");
    }
 }
 public Customer form2Po(ActionForm form) {
    UserForm vo=(UserForm)form;
  Customer cus=null;
  if(vo.getId()!=null && (!vo.getId().equals("")))
cus= customerManage.read(vo.getId());
  if(cus==null)
cus= customerManage.newCustomer();
  cus.setUserName(vo.getUserName());
   cus.setAddress (vo.getAddress());
   cus.setEmail(vo.getEmail());
   cus setTelephone (vo.getTelephone());
  cus.setPassword(vo.getPassword());
  cus setRealname (vo.getRealname());
  cus.setCelephone(vo.getCelephone());
  return user;
 }
}
```

Action 中使用的都是 CustomerInteface，CustomerManage 等接口，其中 execute 方法中有一句有点不一样：

```
WebApplicationContext     wac
=WebApplicationContextUtils.getRequiredWebApplicationContext(t
his.getServlet().getServletContext());
 this.          customerManage          =          (CustomerManage)
wac.getBean("customerManage");
```

这就是 Spring IOC 的使用。因为有了他,我们的 Web 开发人员

也不用知道这个 CustomerManage 究竟是什么东西了，有时候也无法知道。做这一步的时候，可能另外分工做后台商业逻辑层的伙伴还没有把 CustomerManage 的实现写出来。Web 开发人员需要用的 CustomerManage 的时候，直接通过类似的方法找 Spring 取就是了。Spring 一个角色是扮演 IOC 容器中间件。

3. 配置 Struts 配置文件 struts-config.xml

struts-config.xml 的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN" "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <data-sources />
  <form-beans >
                <form-bean name="CustomerForm" type="com.form.CustomerForm" />
  </form-beans>
  <action-mappings >
<action attribute="CustomerForm" input="/CustomerEdit.jsp" name="CustomerForm"
    path="/CustomerManage" scope="request"
    type="com. action.CustomerAction">
    <forward name="doAdd" path="/CustomerRegist.jsp" />
    <forward name="list" path="/CustomerList.jsp" />
   </action>
  </action-mappings>
                                                    <plug-in
className="org.springframework.web.struts.ContextLoaderPlugIn">
 <set-property property="contextConfigLocation"
```

```
                value="/WEB-INF/applicationContext.xml" />
 </plug-in>
</struts-config>
```

4. 最后还差一步，就是配置 web.xml 文件

Web.xml 文件内容如下

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
 <context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/applicationContext.xml</param-value>
 </context-param>
 <listener>
  <listener-class>org.springframework.web.context.ContextLoaderLi
stener</listener-class>
 </listener>
 <servlet>
  <servlet-name>action</servlet-name>
<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
   <param-name>config</param-name>
   <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
 </servlet>
 <servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
 </servlet-mapping>
</web-app>
```

第6章 总结与展望

至此，整个系统设计已经完成，纵观整个程序，基于 Struts 架构的项目开发，首先需要有一个很好的整体规划，整个系统中包括哪几个模块，每个模块各需要多少 FormBean 和 ActionBean 等，而且最好有专人负责 Struts-config.xml 的管理。开发基于 Struts 的项目的难点在于配置管理，尤其是对 Struts-config.xml 的管理。持久化层的难点在于 hibernate 太灵活，相同的问题，至少可以设计出十几种方案来解决。而哪种方案好呢？谁的效率更高呢？确实比较为难。作者的体会是：最好的办法就是查阅相关资料或者询问有经验的开发人员来取得最好的解决方案。

在解决 3 层的整合方面，本论文提出了其中一些难点问题解决方法，比如，

1. 如何将 request 的 action 请求转发到 spring

解决方案有两种：

1) 在 struts-config.xml 中将 actiontype 属性改为有 spring 控制即 "org.springframework.web.struts.DelegatingActionProxy",

2) 建立一个 baseaction 类文件将 struts 所有的类都继承它。即在调用 action 的时候转到 spring 中。

2. 怎样用 spring 控制 hibernate 方法：

在 applicationContext.xml 中设置 bean:"dataSource" 和 "sessionFactory"就行

总之，一个好的框架可以减轻开发者处理复杂的问题的负担。一个好的框架将使系统有良好的扩展性、健壮性和可维护性。Spring+Hibernate+Struts 就是这样一个典型的轻量级框架。它拥

有一个强大的用户团体，并且是开源的，作为下一代 J2EE 构架的基础有着无法比拟的优势。所以轻量级框架的应用前景是十分光明的。

网上购物系统则是轻量级框架应用需求的很好案例。

致 谢

首先感谢清华 IT 给了我这次实践机会，使得我能够把在学校学到的理论知识和具体项目中的实践联系起来，在此过程得到了充分的锻炼，提高了自己的工程实践能力。在论文的写作期间，清华 IT 的领导和工程师们，以及同组的同学们给了我极大的支持和帮助，从参考资料的收集到论文的写作，他们都不遗余力地提供帮助，在此表示衷心的感谢。

导师李红辉在我论文写作的过程中，从论文的整体构思到论文的具体写作都给予了精心的指导和耐心的帮助，他那严谨治学的态度给我留下了深刻的印象，在此我向他表示最真挚的敬意和深深的谢意。另外，我还要感谢学院院长和班主任以及学院的其他老师，他们在我完成学业和论文期间也给了很多的关心和帮助。

最后，我再次由衷地感谢所有支持我完成学位论文研究工作的老师和同学们。由于能力有限，论文写作中的不足之处在所难免，希望批评指正。

参考文献

[1] 孙卫琴.基于 MVC 的 JAVA WEB 设计与开发.电子工业出版社.2005.6

[2] 赵强.基于开源软件的 J2EE 企业级应用开发.电子工业出版社.2005.4

[3] WILLIAM CRAWFORD, JONATHAN KAPLAN.J2EE 设计模式.中国电力出版社.2005.4

[4] 罗时飞.精通 SPRING 电子工业出版社.2005.4

[5] Bruce Eckel(侯捷译).Java 编程思想(第二版).机械工业出版社.2004.6

[6] 孙卫琴. 精通 Hibernate:Java 对象持久化技术详解.电子工业出版社.2006.1

[7] 飞思科技产品研发中心编著.JSP 应用开发详解(第二版).电子工业出版社.2004.7

[8] 敬铮.SQL Server 高级开发与专业应用.国防工业工业出版社.2002

[9] Subrahmanyam Allamaraju、Cedric Buest、John Davies.J2EE 编程指南(1.3 版). 电子工业出版社.2002

[10] 廖义奎.JAVA WEB 开发之 STRUTS 编程基础与实例精讲.中国电力出版社.2006.1

[11] 丁鹏 刘方 邵志峰 何丙胜.STRUTS 技术揭密及 WEB 开发实例.清华大学出版社.2004.3

[12] 孙卫琴.精通 STRUTS:基于 MVC 的 JAVA WEB 设计与开发.电子工业出版社.2004.8

[13]夏昕 曹晓钢 唐勇.深入浅出 HIBERNATE.电子工业出版社.2005.5

[14]JIM KEOGH.J2EE 参考大全.电子工业出版社.2003.5

[15]蔡剑 景楠.JAVA WEB 应用开发:J2EE 和 TOMCAT（第 2 版）.清华大学出版社.2005.1

[16]张宏展 蔡宗琰 吴欣.实战 J2EE 与 WEBLOIC SERVER 应用开发.电子工业出版社.2004.4

[17]张桂元 贾燕枫.STRUTS 开发入门与项目实践.人民邮电出版社.2005.9

[18] ARCHITECT FOR J2EE(EXAM 310-051)（英文版）.人民邮电出版社.2003.8

[19] ALAN MONNOX .Rapid J2EE Development: An Adaptive Foundation for Enterprise Applications.机械工业出版社.2006.1

[20] JOHN HUNT,CHRIS LOFTUS.Guide to J2EE:Enterprise Java.清华大学出版社.2004.7

[21]JAMES ELLIOTT.Hibernate:A Developer's Notebook.东南大学出版社.2005.11

[22]Christian Bauer、Gavin King.2004.Hibernate In Action.Manning Publications

[23]GRAIG A.BERRY JOHN CARNELL MATJAZ B.JURIC .J2EE Design Patterns Applied.电子工业出版社.2003.1

[24]PAUL R.ALLEN,JOSEPH J.BAMBARA .J2EE 学习指南—SUN CERTIFIED ENTERPRISE. 2003.8

[25]James Goodwill,Mastering Jakarta,Struts Wiley Publishing,Inc.2002

[26]Chuck Cavaness.Jakarta Struts.O'Reilly.2002

[27]James Goodwill.Mastering Jakarta Struts.Wiley Publishing.2002

附录 部分源程序代码

1. 建表的 SQL 语句

客户信息表
表名：UserInfo
SQL 脚本：

```
create table UserInfo
(
userId              int             not null,
username            varchar(50)     not null,
userPass            varchar(50)     not null,
userRealName        varchar(50),
userTel             varchar(50),
userCellPhone       varchar(50),
userEmail           varchar(50),
userAddress         varchar(50),
userLevel           varchar(50),
constraint pk_ UserInfo primary key (userId)
)
```

管理员信息表
表名：EmployeeInfo
SQL 脚本：

```
create table EmployeeInfo
(
employeeID              int             not null,
employeeName            varchar(50)     not null,
```

```
employeePass        varchar(50)   not null,
dutyID                  int           not null,
constraint pk_ EmployeeInfo primary key (employeeID),
constraint fk_EmployeeInfo_1 foreign key(dutyID) references
DutyInfo(dutyID)
)
```

管理员权限表

表名： DutyInfo

SQL 脚本：

```
create table DutyInfo
(
dutyID          int           not null,
duty            varchar(50)   not null
constraint pk_ DutyInfo primary key (dutyID)
)
```

商品类别表

表名：ProductSort

SQL 脚本：

```
create table ProductSort
(
productSortID       int           not null,
productSortName     varchar(50)   not null
constraint pk_ ProductSort primary key (productSortID)
)
```

商品基本信息

表名：ProductInfo

SQL 脚本：

```
create table ProductInfo
(
productID              int              not null,
productName            varchar(50)    not null,
productSortID          int              not null,
productImg             varchar(50)    not null,
productManufactorer    varchar(50)    not null,
ProductBrandID         int              not null,
ProductFieldID         int              not null,
productPrice           float            not null,
productAgio            float            default '1',
ProductAgioVIP         float            not null,
productIntrod          varchar(50)      not null,
productDetail          varchar(200)     not null,
productAmount      int              default '0',
ProductOrderAmount     int              not null,
AddTime                datetime         default getdate()
constraint   pk_ ProductInfo   primary key (productID),
constraint   fk_ ProductInfo_1 foreign key(productSortID) references
ProductSort(productSortID),
constraint       fk_ProductInfo_2    foreign    key(ProductBrandID)
references
ProductField(ProductBrandID),
constraint fk_ProductInfo_3 foreign key(ProductFieldID) references
ProductField (ProductFieldID)
)
```

商品区块表

表名：D ProductField

SQL 脚本：

```
create table ProductField
(
productSortID          int             not null,
productSortName        varchar(50)     not null
constraint pk_ ProductField primary key (productSortID)
)
```

商品品牌表

表名：ProductBrand

SQL 脚本：

```
create table ProductBrand
(
productBrandID         int             not null,
ProductBrandName       varchar(50)     not null
constraint pk_ProductBrand primary key (productBrandID)
)
```

订单表

表名：OrderInfo

SQL 脚本：

```
create table OrderInfo
(
orderid                varchar(50)     not null,
userId                 int             not null,
orderTime              datetime        default getdate(),
deliverTime            datetime,
DeliverWayID           int             not null,
userRealName           varchar(50)     not null,
```

```
telPhone            varchar(50)    not null,
cellPhone           varchar(50)    not null,
deliverAddress      varchar(50)    not null,
statusID            int            not null,
fulfillTime         datetime       not null,
orderTotle          float          not null,
post                varchar(50)    not null,
constraint  pk_ OrderInfo  primary key (orderid),
constraint  fk_ OrderInfo _1 foreign key(userId) references
UserInfo(userId),
constraint  fk_OrderInfo _2 foreign key(DeliverWayID) references
DeliverWayType(DeliverWayID)
constraint fk_OrderInfo _3 foreign key(statusID) references
OrderStatusType (statusID)
)
```

订单配送方式表

表名：DeliverWayType

SQL 脚本：

```
create table DeliverWayType
(
deliverWayID      int            not null,
deliverWayName    varchar(50)    not null
constraint pk_ DeliverWayType primary key(deliverWayID)
)
```

订单状态表

表名：OrderStatusType

SQL 脚本：

```
create table OrderStatusType
```

```
(
statusID        int              not null,
statusName   varchar(50)   not null
constraint pk_ OrderStatusType primary key(statusID)
)
```

订货表详细信息

表名：IndentInfo

SQL 脚本：

```
create table IndentInfo
(
indentedID            int              not null,
orderId               varchar(50)      not null,
productID          int              not null,
productAmount   int              not null,
sellPrice             float            not null,
constraint   pk_ IndentInfo   primary key (indentedID),
constraint   fk_ IndentInfo _1 foreign key(orderId) references
OrderInfo (orderId),
constraint   fk_ IndentInfo _2 foreign key(productID) references
ProductInfo (productID)
)
```

新闻信息表

表名：newsInfo

SQL 脚本：

```
create table newsInfo
(
newsId            int              not null,
newsPostTime    datetime         not null,
```

```
newsTitle          varchar(50)       not null,
newsContent        varchar(500)      not null,
newsTypeID         int               not null,
newsPrice          varchar(50)
constraint   fk_ newsInfo _1 foreign key(newsTypeID) references
newsType (newsTypeID)
)
```

新闻类别表

表名：newsType

SQL 脚本：

```
create table newsType
(
newsTypeID         int               not null,
newsType   varchar(50)   not null
constraint pk_ newsType primary key(newsTypeID)
)
```

会员评论

表名：FeedBack

SQL 脚本：

```
create table FeedBack
(
newsTypeID             int              not null,
productID              int              not null,
feedBackPostTime       datetime         not null,
userID                 int              not null,
feedBackContent        varchar(200)     not null
constraint pk_ FeedBack   primary key(feedBackID),
constraint   fk_ FeedBack _1 foreign key(productID) references
ProductInfo (productID),
```

```
constraint    fk_ FeedBack _2 foreign key(userID) references
UserInfo (userID)
)
```

2. 系统购物车模块页面代码 buyCar.jsp

```
<%@page contentType="text/html; charset=GBK"%>
<%@page import="java.util.Vector"%>
<%@page import="com.eshop.javabean.guest.BuyCarBean"%>
<%@page import="com.eshop.javabean.ProductInfo"%>
<%@page import="com.eshop.operationbean.ProductInfoBean"%>
<%@page import="com.eshop.javabean.UserInfo"%>
<%@page import="java.math.BigDecimal"%>
<%
  String path = (String) request.getContextPath();
  int                userType                =                ((UserInfo)
session.getAttribute("userInfo")).getUserLevel();
%>
<html>
<head>
<title>buyCar</title>
<style type="text/css">
  <!--
    .style1 {color: #FF0000}
  -->
</style>
</head>
<script language="JavaScript" type="">
function delBuyCared(){
var count=0;
    if(document.form1.delPro.length>1){
```

```
        for(var i=0;i<document.form1.delPro.length;i++){
            if(document.form1.delPro[i].checked){
                count++;
            }
        }
    }else if(document.form1.delPro.value!=0){
        if(document.form1.delPro.checked){
                count++;
        }
    }
    if(count>=1){
        form1.action="/buyCarDelAction.do";
        form1.submit();
    }else{
        alert("请至少选择一项再删除!");
    }
}
function allChecked(){
    if(document.form1.delPro.length>1){
        for(var i=0;i<document.form1.delPro.length;i++){
            document.form1.delPro[i].checked =
document.form1.allCheck.checked;
        }
    }else{
        document.form1.delPro.checked =
document.form1.allCheck.checked;
    }
}
</script>
<form name="form1" method="post" action="">
```

```
<table width="700" border="1" align="center">
  <tr>
    <td colspan="5" align="center">购 物 车</td>
  </tr>
  <tr align="center">
    <td width="100">商品 ID</td>
    <td width="273">商品名称</td>
    <td width="83">已购数量</td>
    <td width="121">单价</td>
    <td width="89">删除</td>
  </tr>
<%
  boolean b = true;
  Vector vector = (Vector) session.getAttribute("buyCar");
  if (vector == null) {
    b = false;
%>
  您什么也没有购买呐~~~~~~~~
<%
  } else {
    float sum = 0;
    float price = 0;
    for (int j = 0; j < vector.size(); j++) {
      BuyCarBean bb = (BuyCarBean) vector.get(j);
      ProductInfo pdi = bb.getProInfo();
      if (userType == 0) {
        price = pdi.getProdAgio();
      }
      if (userType == 1) {
        price = pdi.getProdAgioVIP();
```

```
        }
%>
  <tr>
    <td align="center"><%=pdi.getProductID()%>      </td>
    <td>
      <a
href="<%=path%>/jsp/guest/productDetail.jsp?id=<%=pdi.getProduc
tID()%>"><%=pdi.getProductName()%>          </a>
    </td>
    <td align="center"><%=bb.getProNumber()%>      </td>
    <td>          &yen;
<%=price%>      </td>
    <td align="center">
      <input              type="checkbox"              name="delPro"
value="<%=pdi.getProductID()%>">
    </td>
  </tr>
<%
  sum += price*bb.getProNumber();
  }
       BigDecimal bd = new BigDecimal(sum);
  sum = (bd.setScale(2, 5)).floatValue();
%>
  <tr>
    <td> </td>
    <td> </td>
    <td align="center"> </td>
    <td> </td>
    <td align="center"> </td>
  </tr>
```

```
  <tr>
    <td> </td>
    <td> </td>
    <td colspan="2" align="left">          .. 总价 ..
      <span class="style1">&yen;<%=sum%></span>
    </td>
    <td align="center">
      <input type="checkbox" name="allCheck" value="checkbox"
onClick="allChecked()">
      全选
    </td>
  </tr>
<%}%>
  <tr>
    <td> </td>
    <td>          :::
      <a href="<%=path%>/jsp/guest/index.jsp">继续购买</a>
      :::
    <%if (b) {      %>
      <a href="<%=path%>/jsp/guest/toOrder.jsp">生成表单</a>
      :::
    <%}      %>
    </td>
    <td align="center"> </td>
    <td> </td>
    <td align="center">
      <input type="button" name="delBuyCar" value="删除"
onClick="delBuyCared()">
       
    </td>
```