摘　要

网络销售是指销售各方均是通过完全电子形式在计算机网络上来进行的多种销售形式的总称。它是商业和现代信息技术的结合，是利用现有的计算机硬件设备、软件和网络基础设施，在电子网络环境中进行销售活动的先进方式。随着计算机互联网的飞速发展，网络销售形式已经越来越普及，诸如电子产品、书籍、衣服以及越来越多的商品等，都可以在网上进行销售。足不出户就能购买到所需要的各种类型的商品已经成为一种基本的电子商务模式。而所有的网络销售系统几乎都大同小异，基本都采用了 Browser/Server（B/S）模式，大多是通过 ASP 程序代码连接 SQL Server 或 Access 数据库来实现网络销售系统。

本文针对 Java EE 多层体系结构、关键设计模式进行研究，为系统设计与实现提供方法与技术支持;采用 UML 建模技术对系统需求进行了分析,结合 Java EE 设计模式，设计了多层分布式应用框架，该框架实现了表现逻辑和业务逻辑的分离，使系统具有较好的扩展性和可维护性；从系统的功能性、非功能性双方面需求入手，设计并实现了页面静态化、两级缓存、动态可替换页面模板、可配置式定时任务等多项高效实用的技术，满足了网络销售系统多方面的需求；对于分布式系统从需求分析到系统实现进行了全面细致的分析和说明，这对于同类系统的开发也具有一定的参考意义。本系统的主要目的是应用 Java EE 企业开发平台实现的安全性高的网络销售平台，对比以原有技术开发的网络销售平台有很好的安全性和系统效率优势。

买家只要在销售网站注册，就可以浏览商品信息、收藏货物以及加入购物车等在线购物等操作，并最终通过网上银行或支付宝等网络支付方式来完成最终的交易。系统对卖家设有管理员，管理员可以通过网站的后台管理页面对前台显示的产品进行添加、修改、删除等操作，从而实现对网站产品的动态更新管理。此外，还可以随时发布网站公告，并以滚动形式显示在首页内，同时也实现了动态管理添加或删除友情链接及网上投票等功能。这样，既达到了做广告的目的，也可利用销售网络更好地为各种类型的企业进行服务。

近年，企业级 Java 开发平台 Java EE 的不断发展为计算机网络上的计算平台软件开发提供了诸多可行有效的实施解决方案。目前，Java EE 开发平台已经成为构建应用服务框架、软件架构结构的最为核心的应用开发技术，在很多大型的分布式应用系统中，特别是现行电子商务的诸多应用中体现了强大生命力。

关键词：网络商城；Java EE；UML；AOP 模式；非功能性需求

Abstract

Network marketing is the Floorboard of Various forms of marketing which means the business is totally on the network. It combines the business and the modern information technology, it is an advanced sales way in electronic network environment which take full advantage of the existing computer hardware equipment, software and basic network infrastructure. With the rapid development of the computer network, the form of network marketing has been become more and more popular. Nearly all goods can sales on the network, such as electronic products, books, clothes, etc. It is becoming a basic electronic business model that we can buy any types of goods without leaving home. What's more, Nearly all the system of network marketing are the same, basically in Browser/Server (B/S) model, and for achieving the system, it mostly using ASP code to connect the database of SQL Server or Access.

To research Java EE multi-layer structure and key design patterns, the paper provide methods and technical support to realize system design and implementation. For design the multilayer distributed application framework, it using UML modeling technology to analysis the system needs, combining Java EE design patterns, the framework realized the separation of presentation logic and business logic. The system has good expansibility and maintainability; The system started from double needs-functional and non-functional .For meeting the needs of various network marketing system, It resigned and realized many effective and practical technologies. Such as page static, double buffers, dynamic replaceable page template, configurable timing task, etc. The paper provided comprehensive and detailed analysis and explanation from requirement analysis to system realizing in the distributed system, so it has some referential meaning to the development of similar system. The purpose of the system is to realize high safety network marketing platform utilizing the Java EE enterprise development platform. Obviously, it is more safe and effective compared to the former system which developed by previous technology.

As long as we registered in the sales website, we can go online shopping, such as browse the commodity information, collect goods and add to cart. Finally, we complete the trade by network payments using online bank or payment. The system have administrator to provide serves, they achieve the dynamic update administration

through the background of website, they can add, modify, deleted the products which showed in the foreground. In addition, they can publish website announcement anytime that rolling showing on the home page, also they can achieve dynamic management, delete friendship linking and online voting, etc. So, not only they got the purpose of adverting, but also they can provide better services for several types of enterprises using the system.

In recent years, the enterprise Java development platform Java EE keep providing the implementation of effective solutions for the software development of computing platform. At present, the Java EE development platform has becoming the key application development technology in constructing application service framework and software architecture structure, it reflected strong vitalities in many large distributed application system, especially in the current e-commerce.

Keywords： Online shopping; Java EE; UML; AOP; Non-functional

目　录

插图索引

附表索引

第 1 章 绪论

1.1 研究背景与意义

随着中国市场经济的日趋成熟，一种企业要想生存，就必须有一种高效的、便于用户购物和支付的购物形式[1]。网络商城作为一种基于浏览器、服务器实现的购物方式类的网站，越来越多的运用到商家的竞争中，并得到了大多数客户的认可。现在社会上流行的三种网络商城分别是 BTB（企业与企业之间）、CTC（消费者与消费者之间）、BTC（企业与消费者之间）。本文阐述的网络销售系统属于 BTC 类型的网络商城[2]。

在过去传统企业的常规商务营销模式里面，销售商品的行为表现出来的是一种非常间接的流动体制，在这种体制里面，制造企业生产的大部分商品都需要经过商品批发运营商进行商品分销和商品拆销，有时候甚至会需要不止一个批发商来分销以后，才能转达到商品零售商的手里面，这种体制给商品的销售流通无谓增加了诸多中间环节，而且这些中间环节增加了流通环节产生的储存和运营费用，再加上各级别批发运营商都会最大化自身可以获取的毛利率，这样情况下，会使得企业商品的成本出厂价和运营零售价存在很大区别，这是社会诸多生产企业和广大消费者都不想也不愿意接受的相关运营现实。在这种情况下电子商务应运而生，网络商城实现了消费者网络购物和商户之间进行网络交易一种新型的商业运营模式，为企业的商品提供了最直接的销售渠道，也使消费者能更详细，快捷地了解商品信息，从而使生产厂商和消费者达到双赢的局面[3]。

网络商城是在互联网络里面完全开放的互联网网络环境中，基于浏览器/服务器（Browser/Server）的全新应用模式，以信息化手段来实现消费者在网络实现购物行为，各个网络商户间的网络交易行为和在线的电子支付可以说是一种全新的商业模式，给人们的生活带来了巨大的影响。对一个运营商业网站的企业来说，网络商城是其生存的理由和基础，同时也是企业对外展示信息、从事商务活动的窗口[4]。

当今互联网使用发达，BTC 这种标准的企业对消费者直接面对面的电子商务模式成为了网络进行销售的最好营销手段之一，它借助了在线销售活动为主的网络零售方式，为销售者，企业甚至顾客都达到了最好最方便的买卖环境。因为用户的消费习惯慢慢地的改变，再加上一批优秀的企业示范效应的正面推动和促进，网络进行购物的用户在迅速增长。2011 年，我国现存的网络购物规模迅速扩大，以北京、上海、广州和深圳（北上广深）为代表的国内最为中心的城市群体，网

络购物的用户数量在全国网民中已经达到了 52.5%，比如 2011.11.11 和 2011.12.12 两天，当天全国网络购物金额就超 40 亿人民币，说明网络运营模式得到了绝大多数网民的认可和承认。这种商业模式在我国已经基本成熟。利用这种网络商城的基础设施，支付和安全平台等共享到的资源，低成本有效的开展了自己的商业活动。纵观现在互联网的商业巨头，无论是阿里巴巴还是卓越的亚马逊书城，都利用了网络商城这种有效的武器，打造了自己的网络商标[5]。

紧随着手机 3G 技术行业在中国的兴起，这一信息预告了网络将更快更有效地贴近我们的生活，而作为网络上最方便的网络商城更是当红之秀，开发优秀的功能更完善的网络商城绝对有相当的经济效益，本次系统的设计正是伴随着网络商务潮流的发展而开展的[6]。

本网络商城系统分两种角色：客户和管理员。这两种角色各自都有不同的使用功能和权限，方便了现代的网络购物者和商家，减轻了他们的负担，提高了商务买卖的效率，突显现代电子商务的特点。随着现代网络的普及和应用，这种商业化的电子商务管理系统会越来越在潮流中担当不可缺少的角色。

1.2 国内外研究现状

随着中国市场经济的日趋成熟，中国企业面对的竞争压力也越来越大，企业要想生存，就必须充分利用信息化手段来提高管理效率及市场响应速度。电子商务是在互联网开放的网络环境下，基于浏览器/服务器应用方式，实现消费者的网上购物、商户之间的网上交易和在线电子支付的一种新型的商业运营模式。电子商务作为一种独立的经济形态，已初具规模，一些电子商务网站的成立，给人们的生活带来了巨大的影响[7]。

如何建立企业的电子商务，如何把企业业务建在 Internet 上，涉及到建立电子商务网站、开发符合 Internet 特点的有效的业务应用、管理网上的交易信息、保证网上数据安全、快速反映市场变化以及充分满足 Internet 业务进一步发展的要求等等。对一个运营商业企业来说，电子商务网站是其生存的理由和基础，同时也是企业对外展示信息、从事商务活动的窗口和界面。如何设计、建立一个经济、实用、安全、高效、稳定的网站是每个电子商务网站必须考虑的问题。而要解决好这些问题，就必须在提高企业内部管理效率、充分利用企业内部资源的基础上，从整体上降低成本，加快对市场的响应速度，提高服务质量，提高企业的竞争力。但是企业在利用信息化技术时，必须要考虑成本、技术难度、创造的价值等几个方面。越是强大的电子商务系统，那么实现它的技术难度也越大[8]。

结合实际情况，本系统是一个小型的电子商务系统，其设计主要用于实现消费者的网上购物，商家和客户之间的网上交易，以及商家对系统的维护。其开发主要包括前台和后台两大功能模块。前台主要用于为方便客户购买尽可能的提供

全面的服务。后台主要用于商家对商品基本信息的建立和维护，对客户基本操作信息的查询和执行。除了要功能完善外，一个好的电子商务系统还要求具有新颖友好的界面。

电子商务系统极大提高了传统商务活动的效益和效率。与传统商务活动相比它具有很多竞争优势[9]。

（1）降低交易成本。首先通过网络营销活动企业可以提高营销效率和降低促销费用，据统计在 Internet 上做广告可以提高销售数量 10 倍，同时它的成本是传统广告的 1/10；其次，电子商务可以降低采购成本，因为借助 Internet 企业可以在全球市场寻求最优惠价格的供应商，而且通过与供应商信息共享减少中间环节由于信息不准确带来的损失。

（2）它可以帮助企业减少库存。企业为应付变化莫测的市场需求，不得不保持一定库存产品和原材料库存。产生库存的根本原因是信息不畅，以信息技术为基础的电子商务则可以改变企业决策中信息不确切和不及时问题。通过 Internet 可以将市场需求信息传递给企业决策生产，同时企业的需求信息可以马上传递给供应商适时补充供给，从而实现零库存管理。

（3）缩短生产周期。一个产品的生产是许多企业相互协作的成果，因此产品的设计开发和生产销售可能涉及许多关联的企业，通过电子商务可以改变过去由于信息封闭导致的分阶段合作方式改为信息共享的协同并行工作方式，从而最大限度减少因信息封闭而无谓等待的时间。

（4）增加商机。传统的交易受到时间和空间限制，而基于 Internet 的电子商务则是 24 小时全球运作，网上的业务可以开展到传统营销人员销售和广告促销所达不到的市场范围，如我国湖南一养毒蛇农民通过 Internet 将其产品卖到美国一个它未曾谋面的公司。

（5）减轻物资的依赖。传统企业的经营活动必须有一定物资基础才可能开展业务活动，而通过 Internet 可以创办虚拟企业，如网上商店和网上银行开设和发展基本不需要很多的实物基础设施，同时企业还可以将节省费用转让给消费者，这正是著名的网上书店 Amazon 为什么能给消费者提供传统书店无法提供的优惠折扣原因所在。

（6）减少中间环节。电子商务重新定义了传统的流通模式，减少了中间环节，使得生产者和消费者的直接交易成为可能，从而在一定程度上改变了整个社会经济运行的方式。

自从互联网诞生以来，越来越多的企业“触网”，随着技术的进步和时代的发展，电子商务系统的发展经过了几个阶段[10]。

第一阶段，黄页型 yellow page，互联网提供企业或产品黄页，取代了传统的传播介质，与之相比，它的优势在于使用方便，内容新，多，传播范围广，获得

成本低，直到现在，这种服务依然受到市场的欢迎，生命力极强。

第二阶段，广告型 pamphlet，取代了传统的企业介绍画册，增加了多媒体内容，信息量更大，作用相当于一个广告，同时为企业和消费者建立了平等的沟通渠道，由于成本低廉，更多受到小企业的欢迎，拉近了小企业和消费者的距离，降低了小企业和大企业竞争的资本。

第三阶段，销售型 sale，取代传统的销售方式，一些适合在网上销售的产品开始向互联网转移，主要是出于减少流通环节和降低经营成本的考虑，同时因为互联网具有其他销售方式不可比拟的优势，集成了前两个阶段的功能，消费者和企业都更加乐意接受，最先采纳这种销售方式的是原有的邮购商品，大大降低了经营成本，使之成为最快获利的商业网站。

当代社会，人们已经深深的领略到信息革命第二次浪潮的冲击。现在信息技术突破了单位性和地域性的局限，实现了网络和全球化，以英特网为代表的现在信息网络正在以每月均 15%的速度急剧增长，其应用范围也开始从单纯的通讯。教育和信息查询向更具效益的商业领域扩张。据统计，2000 年英特网用户已经达到 3 亿用户，通过英特网实现的商业销售额突破 1000 亿美元[11]。这一趋势告诉人们，电子商务在信息技术的强有力的推动下，已经叩响了人类的大门。计算机的全球联网，形成了与地域、空间无关的时间一体化市场，一种新的，基于计算机网络的新型商业机制正在逐步形成，这是处于激烈竞争环境中的各国政府、制造商、销售商及有关研究部门所不能回避的现实，在已经迈入 21 世纪的今天，认识电子商务、了解电子商务、进而参与电子商务，是每一位政府官员、每一位企业家和每一位消费者都必须认真对待的一项新任务。

1.3 本文研究的主要内容

本文主要研究内容概括如下：

1．首先分析了网络销售系统目前主流的两种技术，通过比较两种主流分布式应用模型，选用 Java EE 作为系统实施模型，并对其相关技术进行了深入的研究。

2．其次分析了系统功能性、非功能性两大方面的需求，利用建模技术对系统的业务进行了解析。

3．再次基于对 Java EE 多层企业应用体系结构的研究和对系统业务需求的了解，结合 MVC、DAO、AOP 模式和系统功能性、非功能性两大需求，给出多层分布式应用框架和相关设计。

4．最后，本文在系统设计的指导下运用 Java EE 技术对网络销售系统给予实现，验证了系统设计，并且提出未来可以改进的地方。

1.4 论文组织结构

第一章 绪论。介绍了本文的研究背景、意义和本文研究内容，同时简要介绍国内外研究现状和本文的组织结构。

第二章 可行性研究。比较了主流的开发技术并阐述了为何选择 Java EE 的理由。介绍了 Java EE 平台及 Java EE 中常用的 Tomcat 中间件、MySQL 数据库，研究了对本系统设计起关键作用的 MVC、DAO、AOP 设计模式和 UML 建模技术，为系统的设计与实现奠定技术与方法基础。

第三章 系统需求分析。从功能性要求和非功能性要求两大方面分析了系统需求。

第四章 系统总体设计。利用 UML 建模技术从系统用例、业务流程、业务状态等方面对系统进行了总体分析。

第五章 系统详细设计。将 Java EE 多层分布式应用框架运用到系统中，从实用、高效的角度出发设计出系统的体系结构。按照分层设计的思想，结合系统功能性、非功能性两大方面的需求对系统进行了设计，并进一步设计了数据库和界面。

第六章 系统实现。基于系统设计进行实例实现，详细阐述了关键技术的实现细节，并展示了系统实现的结果。

最后是结论。概括了现有的研究进度和成果，提出了可持续的研究方向，并对下一步要进行的工作进行探讨。

第 2 章 可行性研究

2.1 开发平台选择

（1）.NET

.NET 框架平台是美国微软公司在 2003 年就提出的新型分布式编程模式，它是以 XML 语言作为基础，以网页服务为其框架核心，辅助其它各种编程技术来综合实现，目的是综合利用互联网网络上的众多运算能力资源和计算机网络带宽资源，最终目的是提高人们的工作效率水平。.NET 平台结构包括如下三个方面：.NET 的框架、Visual Studio.NET 和.NET 的相关服务[12]。

.NET 平台中主要部分是.NET 的框架结构，.NET 框架平台实现了编程语言的独立性和互操作性，.NET 框架拥有的是一种常规通用的框架运行环境，它与操作系统底层有着密切联系。与.NET 框架联系最为密切的编程者开发工具是 Visual Studio .NET 开发系列工具，Visual Studio.NET 是语言独立的，我们可以把它当成是开放程序设计的平台，各种不同的高级程序设计语言都可以被添加插入到此平台上来。

.NET 相关服务是基于 XML 语言的网页服务平台，主要目的是帮助程序开发人员利用“网页服务”来创建任何不同类型的相关设备进行协调访问应用。

.NET 是一种多层次的分布式技术，它得到了操作系统底层平台的全力支持；能够解决微软公司 Windows 操作系统的分布式和多用户通信相关问题，是 Windows 操作系统专业专用的分布式相关架构。

（2）Java EE

Java EE 是 Sun 公司归纳推出的完全开放、基于开放开发标准的开发平台（起初命名为 Java 2 企业版，简称 J2EE。后来因为 J2EE 词汇名称易受到误导，Sun 则将 J2EE 名次改名为 Java EE 版本），Java EE 应用于企业开发、方案部署和项目管理方面，尤其是以服务器为中心的面向网页的诸多相关的应用程序。Java EE 是标准的开发技术使用规范，它能够向广大程序开发人员提供搭建通用工作平台的手段，在 Java EE 平台上完整定义了整套应用标准开发体系和通用结构的部署环境。Java EE 的技术规范是以控件容器为核心内容，控件容器是服务器上运行的实际软件实体，作用是为具体类型的构件能够提供相关运行环境和相应服务。Java EE 标准体系结构中包括四种容器：应用程序客户容器、Applet 容器、EJB 容器、Web 容器四种标准容器[13]。

Java EE 平台中最为核心的技术包括 JSP、Servlet、EJB、JDBC、JQuery 等相

关技术。Java EE 标准体系结构中的四种容器和各种核心技术一起组成应用框架。通过 Sun 公司提供的统一开发应用平台，Java EE 平台大大降低了多层次开发应用的所需费用和执行复杂度，同时能够针对现有的各种应用程序进行集成，提供了强大支持，并且能够增强平台安全性，提高整体性能[14]。

（3）.NET 与 Java EE 的比较选择

为了迎合目前企业级别应用进行开发的需求，美国微软公司和美国 SUN 公司都各自给出了独立方案：微软公司的.Net 平台和 SUN 公司的 Java EE 平台，这两套平台都是为企业应用能够提供可靠性高、分布式的解决方案，它们之间的比较情况，如表 2-1 所示[15]。

微软公司的.Net 平台和 SUN 公司的 Java EE 平台在结构和框架上有许多类似地方。但是，两者之间还是存在差别[16]。

1．.NET 平台是一系列软件产品，但 Java EE 却是一组开放规范和开放开发标准；

2．.NET 平台需要 Windows 操作系统平台支撑，但 Java EE 平台却支持跨平台和平台独立；

3．Java EE 平台依赖于 Java 高级程序设计语言，而.NET 平台则支持多种高级程序设计语言，完全保持语言中立。

表 2.1　Java EE 和.NET 的特征区别

特征	Java EE	.NET
分布协议	RMI/IIOP	DCOM
编程语言	Java	C#,C++等多语言
运行时环境	JVM	CLR
胖客户端	Java Swing	Windows Forms
目录服务	JNDI	ADSI
数据访问	JDBC	ADO.NET
消息队列	JMS	MSMQ
表示层技术	Servlets，JSP	ASP.NET
中间层组件	EJB，JavaBean	COM+，COM
分布事务处理	JTS	MTS
开发工具	Eclipse，J Developer……	Visual Studio.NET

对企业应用来说，Java EE 平台和.NET 平台都支持多层分布式 B/S 应用，但 Java EE 平台在内部系统整合、系统的延展性、安全性等方面更具有优势。为了充分利用企业原有的软硬件资源，不能将系统捆绑于某个平台上，因而选择具有跨平台和平台独立性的 Java EE 平台。

目前，网络销售系统中针对其数据安全性、系统可扩展性、系统快速二次开

发能力等相关要求越来越高。本文拟实现的网络销售系统将采用在 Java EE 平台上的浏览器/服务器模式（俗称瘦客户端模式），原因是想充分利用 Java EE 平台开放规范的开发技术规范、安全性能高的开放开发体系，尤其是平台可移植性等方面，它的性能突出表现如下[17]：

1．安全性：Java 程序设计语言提供的验证授权服务能够为 Java EE 企业应用平台保障身份验证、权限管理、验证证书等诸多实施框架和标准应用接口 API。

2．可扩展性：企业级 Java 组件 EJB 是针对各种不同的业务逻辑进行封装，它对于各种应用是完全透明的。

3．快速开发：现代商业逻辑中强调开发速度，尤其是以组件形式进行二次部署和开发的过程。在企业级 Java 组件服务器之中，企业级 Java 组件服务器内部集成的各种组件命名服务，可以使得各种组件的位置达到透明化目标，同时简化了组件部署和组件维护的各种过程，从而使得开发过程中达到容易分工、最终达到加快二次开发速度的最终目标。

4．支持 XML 语言的数据交换：Java EE 平台全方位支持新一代 XML 网页语言，它能够利用 XML 语言的平台无关性、Java EE 平台跨平台独立性来迅速建立行之有效的数据交换执行平台。

本文网络销售系统采用 Java EE 平台来构建，可充分利用 EJB、Java Bean 来执行连接数据库、商业逻辑处理等等需要反复重复使用的相关功能，这些功能系统会将它们封装成为独立组件，以满足 JSP 网页页面、Servlet 容器调用。本文研发系统拥有如下所述各种优点：

1．利用面向对象高级程序设计语言和面向组件的程序设计思想，达到提高软件代码重用性，缩短系统二次开发时间等重要时间。

2．商业逻辑与表现逻辑、后台控制与流程控制进行分开，均可以增加程序可读性和程序鲁棒性。

3．系统的不同合作伙伴间可约定服务接口的规范，有利于实现各种系统间的相互信息之间的共享和基于交互的研究：研究 Java EE 平台各种实现技术与开发研发方法，将会合理利用 Java EE 框架来构建网络销售管理平台系统，具有非常广泛的市场价值和优良的实用价值。

2.2 Java EE 开发平台

2.2.1 体系结构

Java EE 平台是利用 Java 2 标准平台来实现简化企业实施解决方案的一种开放性体系结构，它能够解决软件开发、系统部署和系统管理等诸多复杂问题的开放性结构。Java EE 平台采用多层次分布式的应用程序的开发模型[18]。

Java EE 平台应用程序的相关逻辑会根据各种系统平台实现的不同功能进行封装。封装到组件中以后，将应用 Java EE 应用程序中的大量相关组件来根据其所位于的不同层次进行分别安装，直到安装到不同机器中为止[19]。这样，Java EE 平台体系结构就形成了，Java EE 体系结构，如图 2-1 所示。

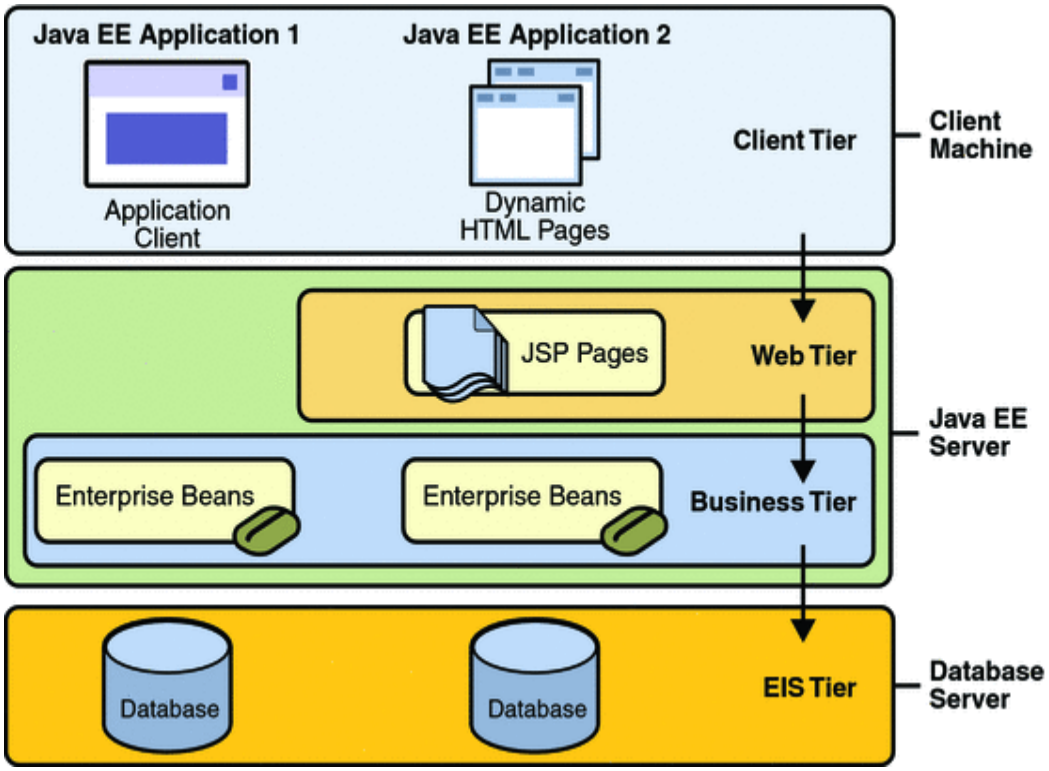

图 2.1　Java EE 体系结构

1．客户层（Client Tier）：Java EE 平台应用是可基于网页，也可不基于网页。在基于网页的 Java EE 平台之中，各种用户的普通浏览器会在客户层中执行，从网页服务器上进行下载网页层中相关静态的 HTML 页面，或者是由 JSP/Servlet 动态生成相关 HTML 页面；在非网页的 Java EE 平台的应用之上，会给独立客户程序或其它运行程序，不经过网页层就直接访问系统的业务层。

2．网页层（Web Tier）：Java EE 网页组件可由基于网页的 Applet、JSP 页面和显示 HTML 静态网页的 Servlet 容器一起构成的。

3．业务层（Business Tier）：业务层内拥有所需的各种业务逻辑代码，由在业务层内的 EJB 来进行执行，企业级 Java 组件马上从客户端进行数据接收，针对数据进行相关处理，再将相关数据发送到相关的企业信息层；并且将最终执行结果返回给相关客户程序。运行在业务事务层的企业级 Java 组件依赖各种容器提供的如：生命期、事务、状态管理等各种复杂的系统级的功能调用。

4．企业信息系统层（EIS Tier）：企业信息系统层中，运行相关企业信息系统功能软件，包括各种企业所需的基础设施系统，比如企业资源管理系统 ERP、企业后台数据库等等。

Java EE 平台结构可以通过把业务逻辑、表现逻辑、后端数据服务进行分隔开，主要的目的是为了向系统开发者提供基于组件的模块化方式，模块方式可以进行系统设计、系统开发、系统装配及应用部署等诸多企业级应用程序。Java EE 平台应用能够为不同种类的应用组件提供对应的相关容器。如图 2.1 所示的 Java EE 体系结构，就包括了 Applet 容器、应用客户端容器、EJB 容器、Web 容器四大容器，容器的作用是为其内部运行的组件提供运行时的服务程序[20]。

1．应用客户端容器：作用是负责运行所有网页客户端的相关组件。

2．Applet 容器：Applet 容器可以当作功能特别的应用客户端容器，它是负责在网页浏览器和 Java 程序小插件上执行运行 Java 程序的小应用程序 Applet，应用客户端容器与 Applet 容器对应多层次结构里面的客户层。

3．网页 Web 容器：它的功能是管理所有 Servlet、JSP 等 Web 网页组件运行机制，针对客户请求进行响应和返回对应的结果，同时为容器提供运行时的支持机制，网页 Web 组件及其相关容器运行在网页的 Web 服务器中。

4．企业级 Java 组件容器：此容器负责所有 EJB 运行机制，它支持组件管理事务和生命周期的管理工作，以及组件查找和其它相关服务，EJB 和容器均是运行在 Java EE 平台的应用服务器上。企业级 Java 组件容器是满足 Java EE 多层次结构的基础，它是控制业务实现的主要运行环境，提供了比如服务事务、安全性、持久性等重要系统等级服务的重要容器。从而，使得开发人员没必要去设计开发这些基础性服务，大大节约系统设计事件，同时把宝贵时间放在业务逻辑实现方面。

2.2.2 设计模式

1．MVC 模式

当前的应用系统正在向多层次网页结构方向进行发展。设计多层次网页应用的难点是怎样组建结构合理、结构整洁的瘦客户结构模型。这就需要能够开发出松耦合高内聚的有效的应用程序解决方案：模型/视图/控制器（MVC，Model/View/Controller）设计模式[21]。

（1）MVC 模式在传统设计中的常规应用

模型/视图/控制器模式来源于早期传统的面向对象程序设计语言 SmallTalk。它是首个区分表示实用逻辑和业务逻辑的新型设计模式。MVC 模式引入了视图（表示层）、模型（数据）和控制协调用的控制器（Controller）。在实际出现 MVC 模式以前，用户进行界面设计时经常会把这些层次进行合并。MVC 模式把它们进行主动分离，从而使得各个模块会相对独立，同时能够提高系统灵活性与系统复用性。MVC 将应用程序系统分为三个组成部分：View 层、Controller 层、Model 层，View 层、Controller 层、Model 层三者之间的动态协作关系，如图 2-2 所示。

从图 2.2 中可见，Controller 控制器的作用是用来起到接收相关使用者发出的不同种类的消息，同时要求 Model 层进行处理相关应用领域资料的操作；Model 层会通知 View 层，让 View 层次能够明白 Model 层次里面的相关内容已经有了更新，View 层次得到通知以后，并且开始进行相关的准备工作，等 View 层次准备就绪了以后，才会要求 Model 送来相关的更新内容，最后显示在 View 视窗中。图 2.2 是特别典型的 MVC 结构，除此之外还有很多应运而生衍生出来的变型结构。

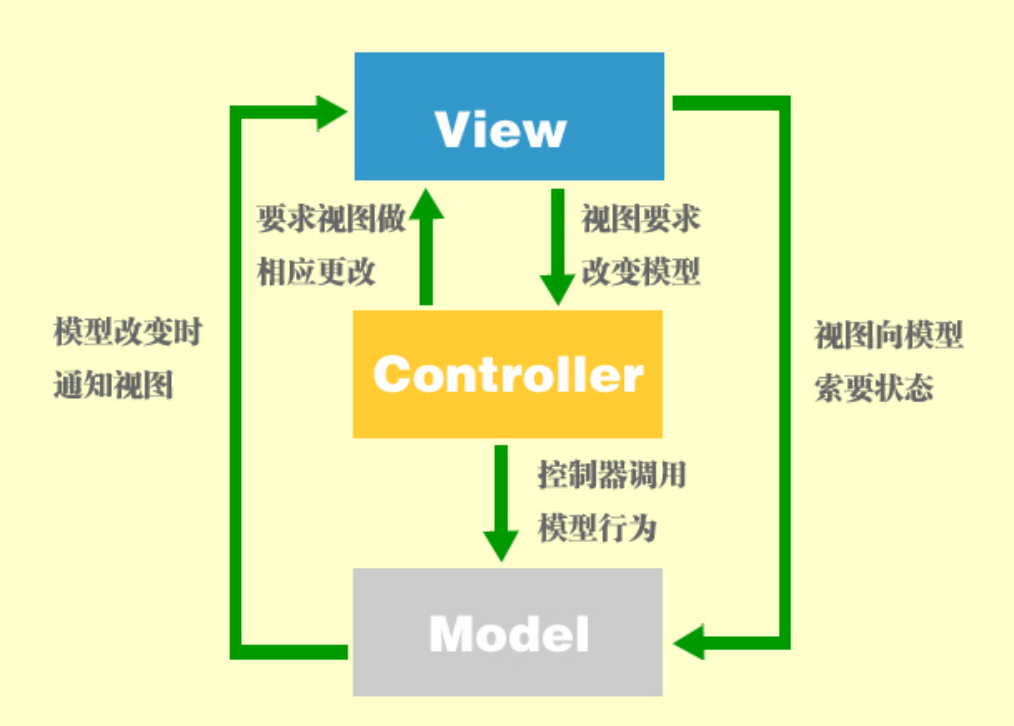

图 2.2　MVC Model

过去传统经典的 MVC 结构只是能够从不同程度理论上，将 View 视图能够从流程控制和业务逻辑中单独独立出来，并进行定义，尤其是对相互之间的各种作用机制进行定义，这样的结构设计可以使得各个模块之间的设计开发工作完全独立出来[22]。基于此种结构来针对不同具体的研究对象，我们在应用 MVC 模式时，需要针对此情况进行进一步研究工作。

（2）MVC 在网页中的应用

传统 MVC 结构模式主要是针对普通一般的各种应用程序，而网页开发设计中，传统 MVC 结构模式的缺点向软件开发设计人员提出了一些非常复杂的挑战，最典型的就是 Client 客户端和 Server 服务器之间的无状态进行连接[23]。这种无状态进行连接行为，会使得 MVC 模型不能将信息修改告知视图层。在网页应用中，为了检测对应用程序的状态进行修改的具体内容，浏览器会不断重新进行查询服务器的操作。因此，在网页中导入 MVC 设计模式，需要对 MVC 设计模式进行做进一步改进，从而迎合网页的具体环境。

MVC Model 2 是在网页此类特殊环境下进行的 MVC 设计模式的变种，它主要的变化是把 JSP 组件和 Servlet 组件进行结合[24]。在 MVC Model 2 设计模式中，View 层次是整个应用程序的最外围，具体是由 JSP 组件来进行实现的。Controller 控制器则是由 Servlet 组件来执行实现的，控制器负责处理相关导航数据流，可以调用原有的 Model，并且负责选择正确的 JSP 页面来执行显示网页的动态内容。MVC Model 2 的整体体系结构，如图 2.3 所示。

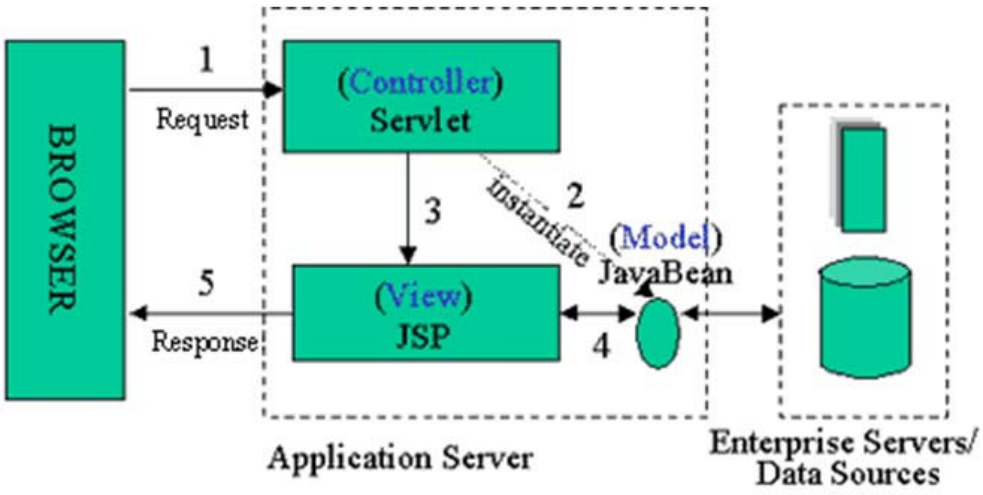

图 2.3　MVC Model 2

创建 MVC Mode1 2 的初衷和目的与创建 MVC 框架相同，都是想应用控制器作为中间处理单元来执行分离模型层与视图层的操作，从而达到层次间松耦合层次内高内聚的真实效果，最终达到提高系统复用性、灵活性、可维护性的最高目标。

2．DAO 模式

应用实体 Bean 组件虽然可以实现相关的数据操作，但有时系统也会需要提供一部分快速进行数据库访问或一些相对复杂的数据库操作手段。此时，我们需要通过 JDBC 或者 JQuery 这样的应用程序相关接口来执行访问数据库中的相关数据表。随着数据库品牌类型的区别，应用 SQL 语句的时候执行的词法语法也会稍有不同，访问数据的逻辑区别就很明显，这样就造成程序代码与数据访问代码之间存在着的依赖关系[25]。

DAO（Data Access Object）数据访问对象实现用来操作访问数据源，DAO 数据访问对象可以向客户端隐藏数据源实现的细节。当数据源底层如果发生了变化，DAO 会向客户端接口发出变化请求，所以该 Java EE 设计模式允许 DAO 调整到不同的存储模式，而不会影响客户端或业务组件。图 2.4 中显示了数据访问对象 DAO 的类图结构[26]。

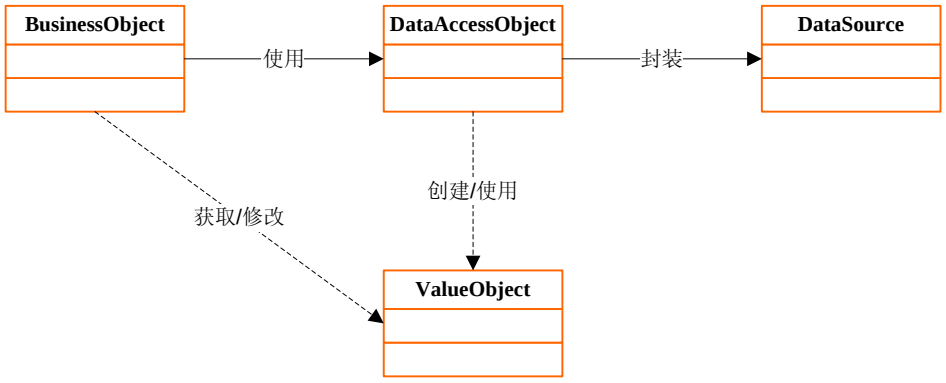

图 2.4　DAO 模式类结构图

它的主要参与者包括业务对象、数据访问对象、数据源和传输对象，主要内容如下：

（1）业务对象：业务对象是表示用户数据，它需要对数据进行访问，单一业务对象可用实体 bean、会话 bean、其余 Java 代码来具体实现。

（2）数据访问对象：数据访问对象是主题，它能够提供数据底层访问的基本对象，同时数据访问对象也将数据加载、存储操作动作都移交给了数据访问对象 DAO 来进行处理。

（3）数据源：数据源可以是一种某一个数据库，包括文件系统、关系数据库、面向对象数据库中的某一种。

（4）传输对象：此处的传输对象是一种数据载体。数据访问对象 DAO 利用传输对象向相关用户进行数据返回，而且，DAO 数据访问对象可以从用户那里得到传输对象来对数据源中的各种数据进行实时更新。

数据访问对象 DAO 是进行数据处理操作的一种典型方法，在分布式系统中起着非常重要的作用。它具有透明性好、可移植性好、减少代码复杂度、集中处理数据访问等特点。

（3）AOP 模式

AOP 是 OOP 的延续，是 Aspect Oriented Programming 的缩写，意思是面向方面编程。AOP 实际是 GoF 的设计模式的一种延续，设计模式孜孜不倦追求的是调用者和被调用者之间的解耦，AOP 可以说也是这种目标的一种实现。

AOP 主要用途是用在记录日志、性能统计，安全控制，处理事务等各个方面。我们可把这些行为看成是系统的各个不同的方面，就是面向方面的编程方法。通过针对这些行为进行分离，我们希望可以将它们单独配置进入商业方法，而需要改变这些行为，也不需要影响商业代码方法。

AOP 的主要术语与含义[27]：

（1）Cross-cutting concern

AOP 模式中，一些系统层面的服务如安全（Security）检查、事务（Transaction）、日志（Logging）等，横切（Cross-cutting）至各个对象的处理流程之中，这些动作在 AOP 的术语中被称之为 Cross-cutting concerns。

以图片说明可强调出 Cross-cutting concerns 的意涵，例如原来的业务流程是很单纯的：

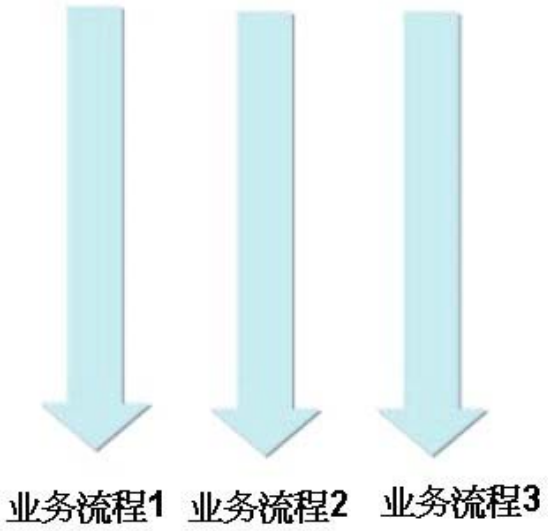

图 2.5　AOP 示意图--一般业务流程

现在为了要加入日志与安全检查等服务，对象的程序代码中若被硬生生的写入相关的 Logging、Security 程序片段，则可使用以下图解表示出 Cross-cutting 与 Cross-cutting concerns 的概念：

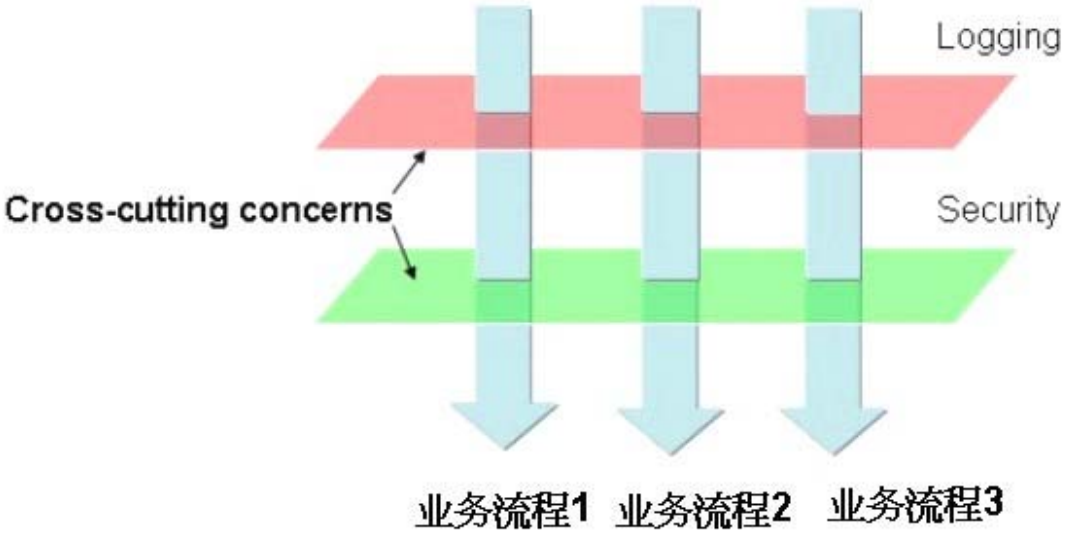

图 2.6　AOP 示意图--Cross-cutting concerns 与业务流程

Cross-cutting concerns 若直接撰写在负责某业务的对象之流程中，会使得维护程序的成本增高，例如若您今天要将对象中的日志功能修改或是移除该服务，则必须修改所有应用了日志功能的程序代码，然后重新编译，另一方面，

Cross-cutting concerns 混杂于业务逻辑之中，使得业务对象本身的逻辑或程序的撰写更为复杂。

（2）Aspect

将散落于各个业务对象之中的 Cross-cutting concerns 收集起来，设计各个独立可重用的对象，这些对象称之为 Aspect，例如把所有登录的动作设计为一个 LogHandler 类别，LogHandler 类别在 AOP 中的描述术语是 Aspect 的具体实例。在 AOP 中重点是 Aspect 辨认分别，将它从业务流程中完全独立出来。在该服务时，缝合（Weave）至各个应用程序上面，不需要服务时，也可以马上从应用程序中进行脱离，可重用组件不需要进行任意修改[28]。

另一方面，对于可重用组件来说，以 AOP 进行的设计方式，它不用知道处理提供服务的对象之存在，具体的说，与服务相关的 API 不会出现在可重用的各种应用程序的组件中，因而可通过提高这些组件的代码组件重用性，你可以将这些组件的应用到各种不同的应用程序中。

（3）Advice

Aspect 的具体实例称为 Advice。以记录动作来说而言，Advice 中包含了真正的程序记录代码，系统关注 Advice 是如何操作的，比如上节所说的 LogHandler 类别是 Advice 的一个具体实例，Advice 包括 Cross-cutting concerns 行为或需要提供的各种服务[29]。

（4）Joinpoint

Aspect 在应用程序执行时加入业务流程的点或时机称之为 Joinpoint，具体来说，就是 Advice 在应用程序中被呼叫执行的时机，这个时机可能是某个方法被呼叫之前或之后（或两者都有），或是某个例外发生的时候[30]。

（5）Pointcut

Pointcut 的定义是一种定义，藉由这个定义你可以指定某个 Aspect 在哪些 Joinpoint 时被应用至应用程序之上。具体的说，您可以在某个定义档中撰写 Pointcut，说明了哪些 Aspect 要应用至应用程序中的哪些 Joinpoint。

（6）Target

一个 Advice 被应用的对象或目标对象。

（7）Introduction

对于一个现存的类别，Introduction 可以为其增加行为，而不用修改该类别的程序，具体的说，您可以为某个已撰写、编译完成的类别，在执行时期动态加入一些方法或行为，而不用修改或新增任何一行程序代码。

（8）Proxy

在《Expert One-on-One J2EE Development Without EJB》一书中[4]，Rod Johnson、Juergen Hoeller 在第八章中有提到，AOP 的实作有五个主要的策略：

Dynamic Proxies、Dynamic Byte Code Generation、Java Code Generation、Use of a Custon Class Loader、Language Extensions。本系统的 AOP 主要是透过 Dynamic Proxies（动态代理）来实现的。

（9）Weave

Advice 被应用到对象之上的过程称之为缝合（Weave），在 AOP 中缝合的方式有几个时间点：编译时期（Compile time）、类别加载时期（Class load time）、执行时期（Runtime）。

结合日志（Logging）和安全（Security）检查的例子，图 2-7 将以上的 AOP 相关名词使用图片来表示：

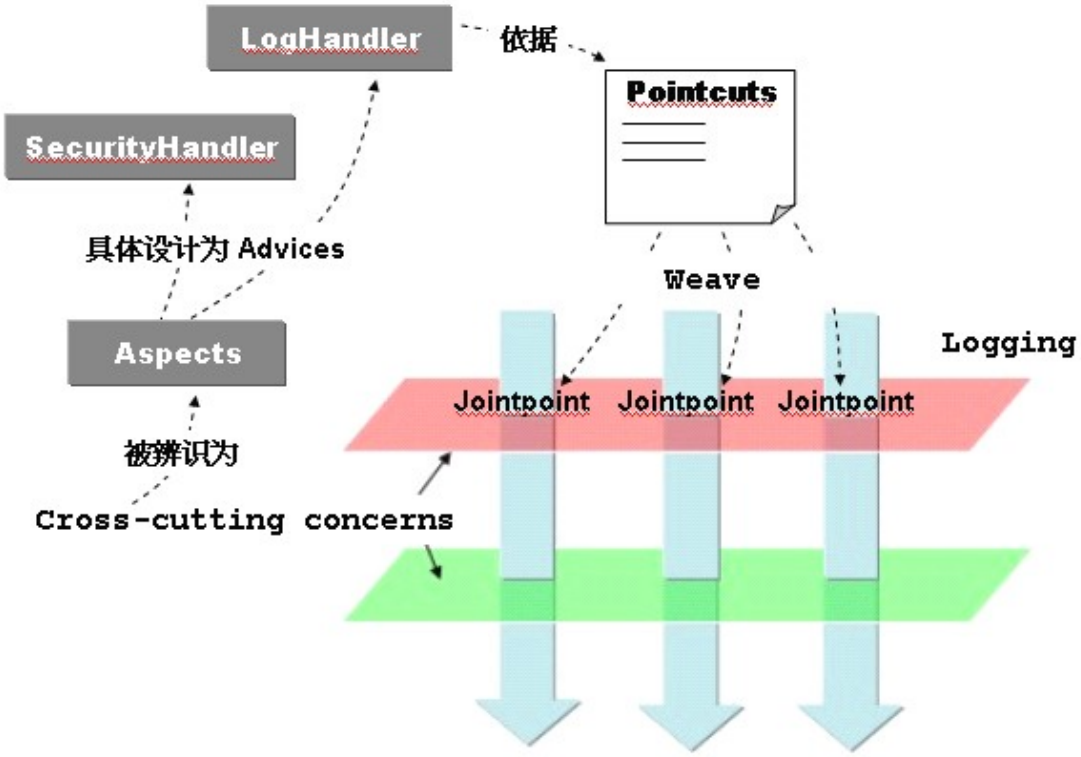

图 2.7　AOP 相关名词示意图

2.2.3 常用开源服务器

1．Tomcat 中间件

Tomcat 是免费开源的网页应用服务器软件中的一种典型代表，它是 Apache 软件基金会执行的 Jakarta 项目里面的一个非常核心的重要项目，当时是由 Apache 软件基金会、Sun 公司和其他一些小型公司以及若干个人进行共同开发完成的。由于 Sun 公司对此项目的积极参与和大力支持，最新版本的 Servlet 和 JSP 规范都能够在 Tomcat 中间件中体现出来[31]。因为 Tomcat 软件技术先进、性能足够稳定，而且代码开源免费，因而特别受到 Java 爱好者的钟爱，并得到了部分软件开发商的认可，成为目前比较流行的 Web 网页应用服务器[5]。

与传统的桌面应用程序不同，Tomcat 里面的应用程序是一个 WAR（Web Archive）文件。WAR 是 Sun 提出的一种 Web 应用程序格式，与 JAR 类似，也是许多文件的一个压缩包。这个包中的文件按一定目录结构来组织：通常其根目录下包含有 Html 和 JSP 文件或者包含这两种文件的目录，另外还会有一个 WEB-INF 目录，这个目录很重要。通常在 WEB-INF 目录下有一个 web.xml 文件和一个 classes 目录，web.xml 是这个应用的配置文件，而 classes 目录下则包含编译好的 Servlet 类和 JSP 或 Servlet 所依赖的其它类（如 JavaBean）。通常这些所依赖的类也可以打包成 JAR 放到 WEB-INF 下的 lib 目录下，当然也可以放到系统的 CLASSPATH 中，但那样移植和管理起来不方便。

在 Tomcat 中，应用程序的部署很简单，只需将 WAR 放到 Tomcat 的 webapp 目录下，Tomcat 会自动检测到这个文件，并将其解压。在浏览器中访问这个应用的 JSP 时，通常第一次会很慢，因为 Tomcat 要将 JSP 转化为 Servlet 文件，然后编译。编译以后，访问将会很快。另外 Tomcat 也提供了一个应用：manager，访问这个应用需要用户名和密码，用户名和密码存储在一个 xml 文件中。通过这个应用，辅助于 Ftp，用户可以在远程通过 Web 部署和撤销应用。当然本地也可以。

Tomcat 不仅仅是一个 Servlet 容器，它也具有传统的 Web 网页服务器的传统功能“专业处理静态网页页面”。与 Apache 软件进行比较，Tomcat 处理静态网页 Html 语言的能力就远远比不上 Apache 软件。那么，广大用户可以充分利用 Tomcat 与 Apache 各自的优点，将两个软件进行集成化，比如主要让 Apache 软件处理静态网页 Html 页面，而让 Tomcat 软件主要处理 JSP 与 Servlet。这种软件集成执行起来非常简单，只需简单修改 Apache 与 Tomcat 的相关配置文件[32]。

另外，Tomcat 支持 Realm。Realm 相当于 Unix 操作系统里面的组 group。在 Unix 操作系统中，一个组 group 会对应系统里面的一定数量的资源，某个 group 没有权限访问不属于该组的相关资源。Tomcat 软件应用 Realm 针对不同应用来赋给不同用户（和 group 相似）。没有相关权限的系统用户不允许访问这个相关系统应用。

Tomcat 软件目前提供三种 Realm：（1）JDBCRealm，此 Realm 会将用户信息储存在后台数据库中，通过 JDBC 连接函数来获得相关用户信息来实现验证过程；（2）JNDIRealm，LDAP 的服务器中存放用户信息，通过 JNDI 连接函数来获取相关用户信息；（3）MemoryRealm，应用 XML 文件来存储用户信息，manager 应用会执行验证用户，此时使用 Realm。通过应用 Realm 用户，可以快捷针对访问某个应用客户来进行验证[33]。

在 Tomcat 软件中，还可以利用 Servlet 组件所具有的事件监听器的功能，来针对应用或 Session 来实行相关监听工作。Tomcat 软件也提供其它的一些不同特征，比如与 SSL 集成在一起实现相应的安全传输。还有，Tomcat 软件也能够提供

JNDI 接口支持，这与那些 Java EE 应用服务器提供的接口是一致的。

首先介绍应用服务器 WebLogic 软件与 Tomcat 软件有哪些区别。应用服务器能够提供 Java EE 的很多特征，比如 EJB、JAAS、JMS 等相关特征，同时也支持 JSP 组件与 Servlet 组件。而 Tomcat 的功能没有 Weblogic 那么强大，它只是一个轻量级的服务器，不能提供企业级 Java 组件等方面的支持。但如果能够与 JBoss（一个开源社区提供的应用服务器）集成在一起，则可实现 Java EE 平台的全部所有功能。虽然应用服务器软件都兼顾具有 Tomcat 功能，那么 Tomcat 软件还有没有继续存在的价值和必要性呢？实际情况是诸多中小应用型程序是完全不需要采用企业级 Java 组件等高级别技术，JSP 组件与 Servlet 组件功能已经非常足够，这种情况下，如果使用应用服务器软件 Weblogic 就有些“大材小用”了。Tomcat 软件小巧，功能实用，配置简便，能够满足绝大多数中小型客户用户需求。在这种情况下，大多数用户自然而然都会选择 Tomcat 软件。

基于 Tomcat 软件进行的开发其实就是基于 JSP 组件和 Servlet 组件进行开发的过程，开发 JSP 语言和 Servlet 组件简单易行，广大程序设计人员能够使用最简单的记事本、普通集成开发环境等普通软件实现，再将写出来的文件打包保存成 WAR 后缀名。在 JSP 程序开发中，可利用内置标签库来实现 Java 程序设计代码与静态网页 Html 文件进行分离，这种方法可以使 JSP 语言维护起来快捷方便。

Tomcat 软件可与一些其他软件进行集成，集成后可以实现更多丰富功能。比如：如果与 JBoss 软件进行集成，一起来开发企业级 Java 组件；如果与 Cocoon 项目进行集成，一起来开发 XML 的相关应用；如果与 OpenJMS 进行集成，一起来开发 JMS 的相关应用等。Tomcat 软件功能强大、开放源代码、免费。

2．MySQL 数据库

（1）MySQL 概述

MySQL 是一个小型关系型数据库管理系统，开发者为瑞典 MySQL AB 公司。目前 MySQL 已经被广泛应用于互联网网络上的众多中小型级别的网站之中。由于该数据库速度快、体积小、成本付出低廉，特别是“开源”的特点，使得众多中小型级别网站出于降低网站生产成本和免除不必要的知识产权纠纷，选择 MySQL（图 2.8）作为中小型网站的后台标准数据库[6]。

（2）MySQL 的特性

1）MySQL 中应用 C 或 C++都可以进行编写，并且可以使用多种不同编译器进行各种测试，还能够保证相关源代码的平台可移植性

2）MySQL 可以在各种 Unix 操作系统、Windows 标准平台、苹果公司操作系统等各种操作系统之上运行。

3）MySQL 支持多种程序设计语言，并为多种程序设计语言提供了相应的程序设计接口 API 函数。

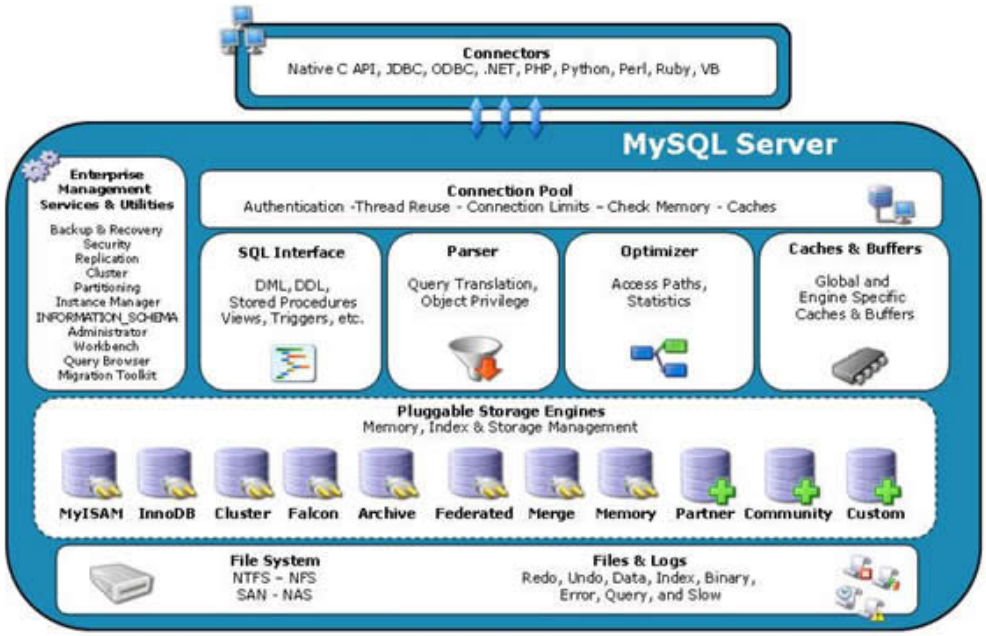

图 2.8　MySQL 数据库架构

4）MySQL 数据库具有多线程的特点，可以充分调度 CPU 来执行相关工作。

5）MySQL 数据库中拥有经过优化的各种查询算法，可以非常有效地提高相关查询速度。

6)MySQL 数据库能够提供支持 TCP/IP 协议、ODBC 接口、JDBC 接口、JQuery 新型接口等各种数据库的连接方法途径。

7）MySQL 能够提供管理、检查、优化相关数据库相关操作的各种管理工具。

8）MySQL 数据库能够支持大规模记录的各种大型数据库。

（3）MySQL 的各种相关应用

MySQL 有相关不足之处，比如：规模较小、功能受限、MySQL 数据库不支持视图和相关事件等不足，但是这些不足之处完全没有减低它的热度。针对一般的普通中小型规模企业和个人用户来说，MySQL 能够提供的相关功能已经足够满足各种要求。

目前互联网网络上最为流行的各种网站架构方式是 Linux 操作系统、Apache 服务器软件、MySQL 数据库平台、PHP 程序设计语言，在整个构架中，我们使用 Linux 操作系统作为系统操作系统，Apache 服务器软件作为网页服务器支撑软件，MySQL 数据库平台作为数据库构建平台，PHP 程序设计语言可以作为服务器端的脚本编译器和脚本解释器。因为 Linux、Apache、MySQL、PHP 这四个软件均遵循 GPL 开放源码规则，因此免费是这个构建方法最大的优点。

2.3 统一建模语言

UML（Unified Modeling Language，统一建模语言）是面向对象开发中一种

通用的图形化建模语言，它定义良好、易于表达、功能强大且普遍适用。面向对象的分析主要在加强对问题空间和系统任务的理解、改进各方交流、与需求保持一致和支持软件重用等 4 个方面表现出比其他系统分析方法更好的能力，成为主流的系统分析方法[7]。

UML 的出现既统一了 Booch、OMT、OOSE，以及其他方法，又统一了面向对象方法中使用的符号，并且在提出后不久就被 OMG 接纳为其标准之一。从而改变了数十种面向对象的建模语言相互独立且各有千秋的局面，使得面向对象的分析技术有了空前发展。它本身成为现代软件工程环境中对象分析和设计的重要工具，被视为面向对象技术的重要成果之一。

UML 建模是应用模型元素来构建整个系统的原型模型，原型模型的各个元素中包括系统中用到的各种类、各种类和各种类之间的关系、各种类的相关实例，还包括相互配合能够实现系统动态的各种行为等等。UML 标准建模语言中拥有多种类型的图形化、可视化描述元素来进行模型说明。UML 标准建模主要分为结构建模、动态建模、模型管理建模三个相关方面：结构建模方面是从系统的内部结构和静态角度来描述系统的，在静态视图、用例视图、实施视图和配置视图中适用，采用了类图、用例图、组件图和配置图等图形。例如类图用于描述系统中各类的内部结构（类的属性和操作）及相互间的关联、聚合和依赖等关系，包图用于描述系统的分层结构等；动态建模方面是从系统中对象的动态行为和组成对象间的相互作用、消息传递来描述系统的，在状态机视图、活动视图和交互视图中适用，采用了状态机图、活动图、顺序图和合作图等图形，例如状态机图用于一个系统或对象从产生到结束或从构造到清除所处的一系列不同的状态；模型管理建模方面描述如何将模型自身组织到高层单元，在模型管理视图中适用，采用的图形是类图。建模的工作集中在前两方面，而且并非所有图形元素都适用或需要采用。UML 建模工具非常多，常用的有 IBM Rational Rose 和 Microsoft Visio。

2.4 本章小结

.NET 框架平台是美国微软公司在 2003 年就提出的新型分布式编程模式，Java EE 平台是利用 Java 2 标准平台来实现简化企业实施解决方案的一种开放性体系结构，为了充分利用企业原有的软硬件资源，不能将系统捆绑于某个平台上，因而选择具有跨平台和平台独立性的 Java EE 平台。

本章主要介绍了.NET 体系和 Java EE 开发平台各自特点，并通过比较选择针对本文开发系统要求选择了 Java EE 开发平台，然后针对 Java EE 开发平台的体系结构、设计模式和常用开源服务器进行阐述，最后针对本文将使用的 UML 建模技术进行说明。

第 3 章　系统需求分析

3.1 系统功能性要求

本系统的全称是网络电子商城系统，简称 EShop。主要分为前台部分和后台部分，前台部分主要实现的功能针对顾客使用，有顾客的注册和登录功能、查看商品信息功能、搜索商品的功能、购物车功能、反馈意见或留言功能；后台部分主要由商城的管理员使用，分别为实现商品管理员管理商品信息及特价商品信息；订单管理员管理订单信息；会员管理员管理顾客的情况及查看顾客的反馈信息；系统管理员可以对管理员进行管理。此处的系统功能性模块主要依据使用者的区别进行相关分配。

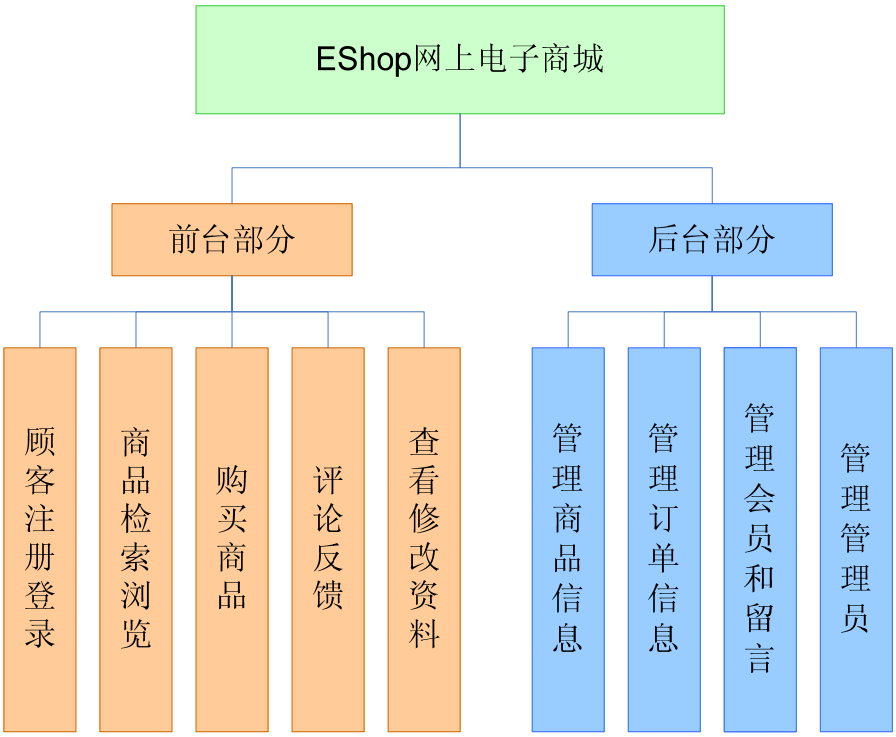

图 3.1 系统具体模块图

它们各自的功能如下所述：

（1）前台部分

前台部分主要包括以下几个方面，顾客的注册登录，顾客查看商品信息，搜索商品信息，查看购物车，删除购物车，修改购物车，确认下订单操作，查看留言反馈信息，增加留言信息，修改顾客自己的信息等。这里的使用对象主要是顾客[8]。

首先顾客可以方便地注册与登录。若还未注册的顾客只能进行商品的查看，

不能购物；未注册的顾客可进入注册页面进行注册，然后可以选择不同的会员级别，有不同的打折率。但是若顾客不符合要求或是有重大表现，后台的管理员可以对其会员级别进行调整。

顾客可以灵活地检索所需的商品信息及了解特价商品信息。可采用关键字搜索。

顾客可以方便地查看详细的商品资料。

顾客可管理自己的购物车（包括添加或删除选购商品、修改商品购买数量及在线下订单等操作）。

顾客还可以反馈自己的意见或留言。

（2）后台部分

后台部分，管理员必须登录才能进行管理，管理员根据不同的权限可以进行相应的操作。

商品管理员可以方便快捷地进行商品管理和特价商品信息管理（比如：新增、删除、修改商品信息及特价商品信息）。

订单管理员能够拥有权限进行查看相关订单、安排发货与缺货警示等相关处理。

会员管理员能够查看顾客的注册信息以及调整会员的级别，还可以查看顾客的反馈信息及具体的处理情况，进行回复。

系统管理员可以进行管理员的管理（新增、删除、修改管理员信息）。

3.2 系统非功能性要求

软件产品的需求可以分为功能性需求和非功能性需求。其中软件产品的非功能性需求是常常被轻视、甚至被忽视的一个重要方面。其实，软件产品非功能性定义不仅决定产品的质量，还在很大程度上影响产品的功能需求定义。如果事先缺乏很好的非功能性需求定义，结果往往是使产品在非功能性需求面前捉襟见肘，甚至淹没功能性需求给用户带来的价值。

所谓非功能性需求，是指软件产品为满足用户业务需求而必须具有的、除功能需求以外的特性。软件产品的非功能性需求包括系统的性能、可靠性、安全/保密性、运行限制、适应性、可扩充性、对技术和对业务的适应性，等等。

本论文除了分析系统的功能性要求之外，更重要的是关注系统的非功能性需求，以体现系统相比同类产品的优越性。本节主要从安全、性能、可维护性、用户体验、成本预算五方面提出了对本系统的非功能性要求。第 5、6 章有对这些非功能要求的设计与实现的详细描述。

（1）安全

1．多角色权限控制，每个角色都有自己的权限范围，并可以对每个角色的权

限进行配置

2．能够防止 SQL 数据库注入式攻击

3．能够防止脚本源代码泄露

（2）性能

1．网站的业务并发用户数在 500 以上

2．支持用户某些操作后台运行，减少用户等待时间

3．使用缓存等技术提高页面访问速度

（3）可维护性

1．实现某些全局参数的可配置化，如系统名称、网站 logo、copyright 信息等

2．提供用户通过管理页面远程查看、执行任务的功能，如数据库备份、访问日志查看等

3．对可以预计到的可变因素实现可配置化，避免系统发布频繁更改源代码

（4）用户体验

1．易操作，有详细的商品分类和导航

2．使用模板实现界面风格可替换

（5）成本预算

1．使用成熟的开源免费技术，减少开发、运营成本

2．优化系统后期运维方式，减少运维人力成本

3.3 本章小结

本章主要介绍了系统功能要求和系统非功能性要求，系统功能性要求朱宝包括前台部分和后台部分，系统非功能性要求主要包括安全、性能、可维护性、用户体验和成本预算。

第 4 章 系统总体设计

4.1 UML 建模

用例建模是 UML 标准建模的重要部分，用例建模也是 UML 中最为基础的相关部分。用例建模的重要功能是用来表达系统的功能性需求或行为。用例图由参与者（Actor）、用例（Use Case）、系统边界、箭头组成，用画图的方法来完成。本节利用 UML 用例图，将上一章的系统功能性要求用图形方式直观地展现出来。

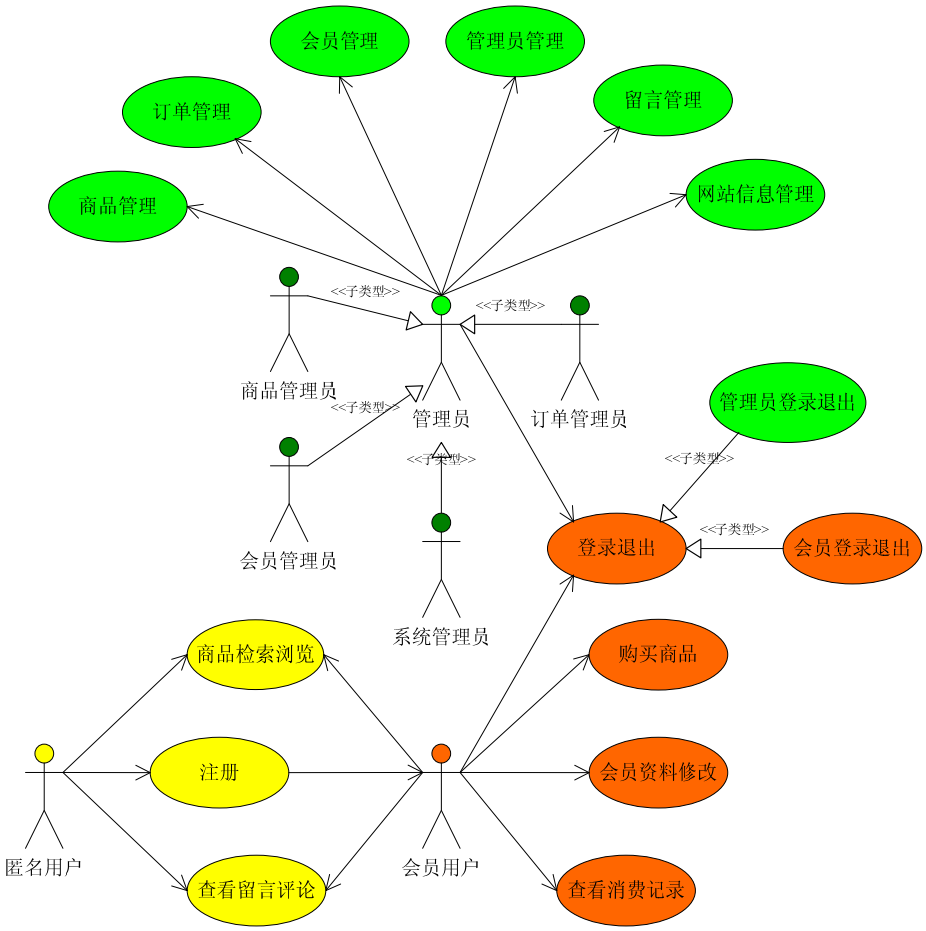

图 4.1 系统主要用例

4.2 体系结构

《Expert One-on-One J2EE Development Without EJB》一书提到，传统的 J2EE 架构方案得到的结果常常无法让人满意：过于复杂的应用程序、令人失望的性能、难以测试、开发和维护成本高昂。经验表明，EJB 往往招致更高的开发成本，并且提供的利益也并不像它最初所鼓吹的那么丰富。在使用 EJB 的过程中，开发者们遇到了很多棘手的问题。

EJB 的问题就在于它违背了“帕累托法则”：这个法则也常常被称为“80/20（或者 90/10）法则”，也就是说：花比较少（10%—20%）的力气就可以解决大部分（80%—90%）的问题，而要解决剩下的少部分问题则需要多得多的努力。在软件架构这里，这个法则告诉我们：架构的价值在于为常见的问题找到好的解决方案，而不是一心想要解决更复杂也更罕见的问题。EJB 为了满足少数情况下的特殊要求，它给大多数使用者强加了不必要的复杂性。比如说，也许只有 10%的应用需要分布式的业务对象，然而 EJB 的基础结构却与对象分布紧密相关。

对于本系统的需求来说，EJB 并不是必须的，完全可以用 POJO（Plain Old Java Object）和 DAO 来取代 EJB。这样一来甚至不需要重量级的应用服务器，而用轻量级的 Web 服务器。

本系统的体系架构是基于对第二章中阐述的 Java EE 体系结构和 MVC、DAO、AOP 模式的研究，以及结合系统需求得出的。

4.2.1 逻辑结构

1. UI 层：

（1）HTML Cache：在用户访问某链接时检查对应的 HTML 页面是否已经存在，如果存在的话取得 HTML 页面内容并显示给用户；如果对应的 HTML 尚未生成或已过期，则为该链接生成 HTML 页面并显示给用户。

（2）View Templates：在将 HTML 页面内容显示给用户之前，先将设定网页模板的应用于该页面。这一层同时实现了网页模板的动态切换，即系统管理人员只要在指定的目录更换了模板文件，则网站的页面风格即刻可以改变。

（3）Control：UI 层的控制中枢，接收用户的访问请求，调用业务逻辑层 Java Bean 得到相关数据，并将数据返回给前台页面。

2. 业务逻辑层：

（1）AOP Proxy：拦截所有业务调用请求，对配置设定的请求加载日志、事务、安全等各方面功能。采用 Java 的动态代理功能实现。

（2）App API：核心业务逻辑实现类，采用 POJO。

（3）Floor API：在业务逻辑 Java Bean 与 DAO 对象之间的衔接类，使得业务逻辑 Java Bean 只包含纯粹的商业逻辑实现代码，而不能掺杂数据层逻辑。

3. 数据访问层：

（1）DAO：数据访问对象，包含数据访问逻辑，同时也是数据承载器。将商业逻辑请求转化为数据请求，从数据库得到数据，并作为传输桥梁输送到 UI 层。

（2）Memory Cache：数据层的缓冲机制，减少了一些相同数据请求的调用，提高数据访问效率。

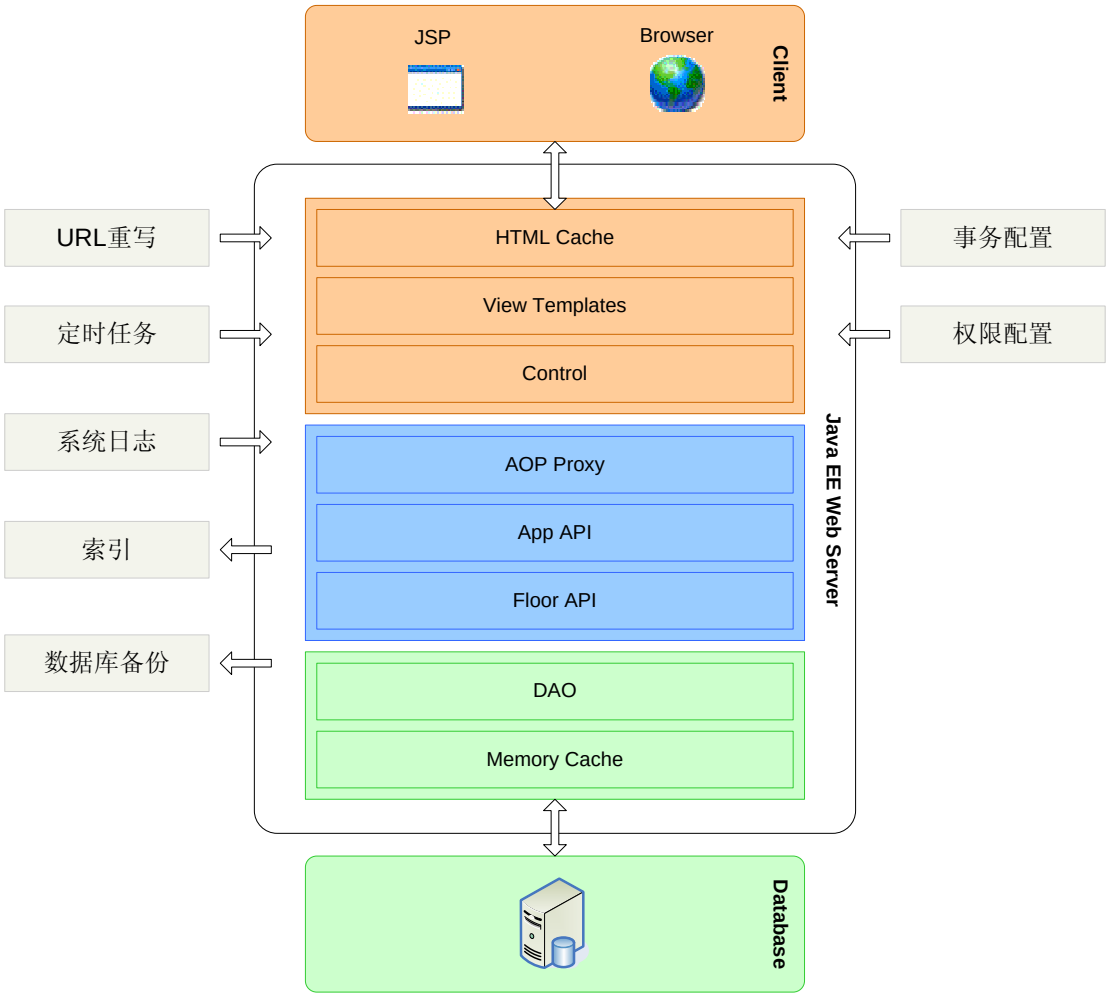

图 4.2 系统逻辑架构

4.2.2 数据流程

（1）页面请求数据流程

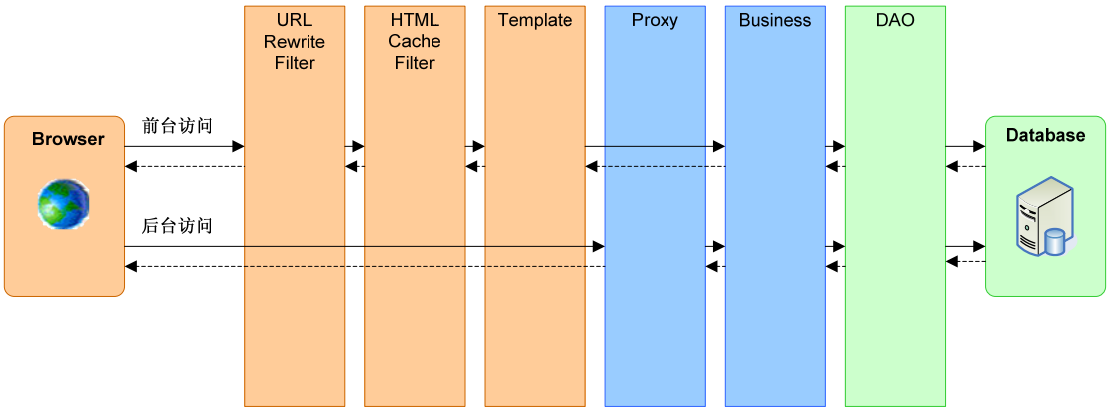

图 4.3 页面请求数据流程

浏览器访问请求，主要分两种情况：前台访问和后台访问。前后台访问一个页面，主要区别在于模板使用和鉴权。因为前台访问需要使用模板，并且不需要做权限验证，所以需要经过 URL 重写过滤器，页面调用的方法不需要通过代理

获取，也就是不必进行鉴权。后台获得数据页面为 JSP，所以不需要经过 URL 重写过滤器过滤器，页面内调用的所有方法，都要通过代理获取，以便通过数据库配置实现粗细粒度权限控制。

不使用代理获取一个类：

ProductMgr productMgr = （ ProductMgr ） factory.getInstanceNonProxy（"com.eshop.app.api.ProductMgr"）；

使用代理获取一个类：

ProductMgrIFace productMgr = （ .ProductMgrIFace ） factory.getInstanceViaProxy（"com.eshop.app.api.ProductMgr"）；

（2）业务提交数据流程

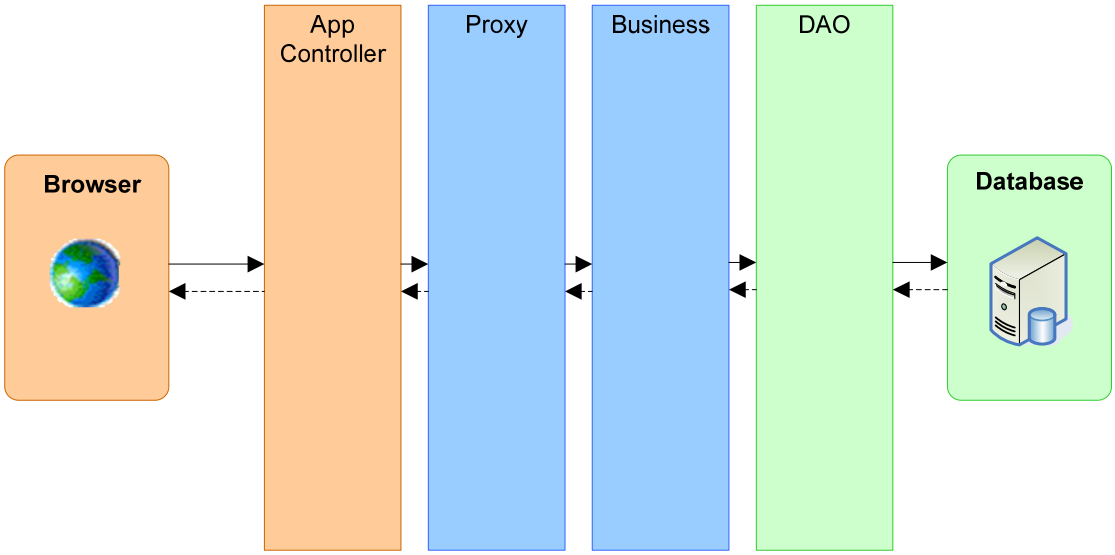

图 4.4 业务提交数据流程

提交表单等操作，都属于提交一个业务流程，所有提交请求都会提交到 appController（命令控制器），经过命令解析后调用相应函数处理，根据处理结果跳转到不同页面。任何提交请求都会经过代理，以便通过数据库配置实现粗细粒度权限控制。

提交请求处理成功后，可以调用相关的提示函数实现页面弹出框提示，或根据 backurl、xml 配置返回到相关提示页面。

4.3 业务流程

活动图是 UML 用于对系统的动态行为建模的另一种常用工具，它描述活动的顺序，展现从一个活动到另一个活动的控制流。活动图在本质上是一种流程图。本节用活动图描述了本商城系统中的一些主要业务流程及顺序。

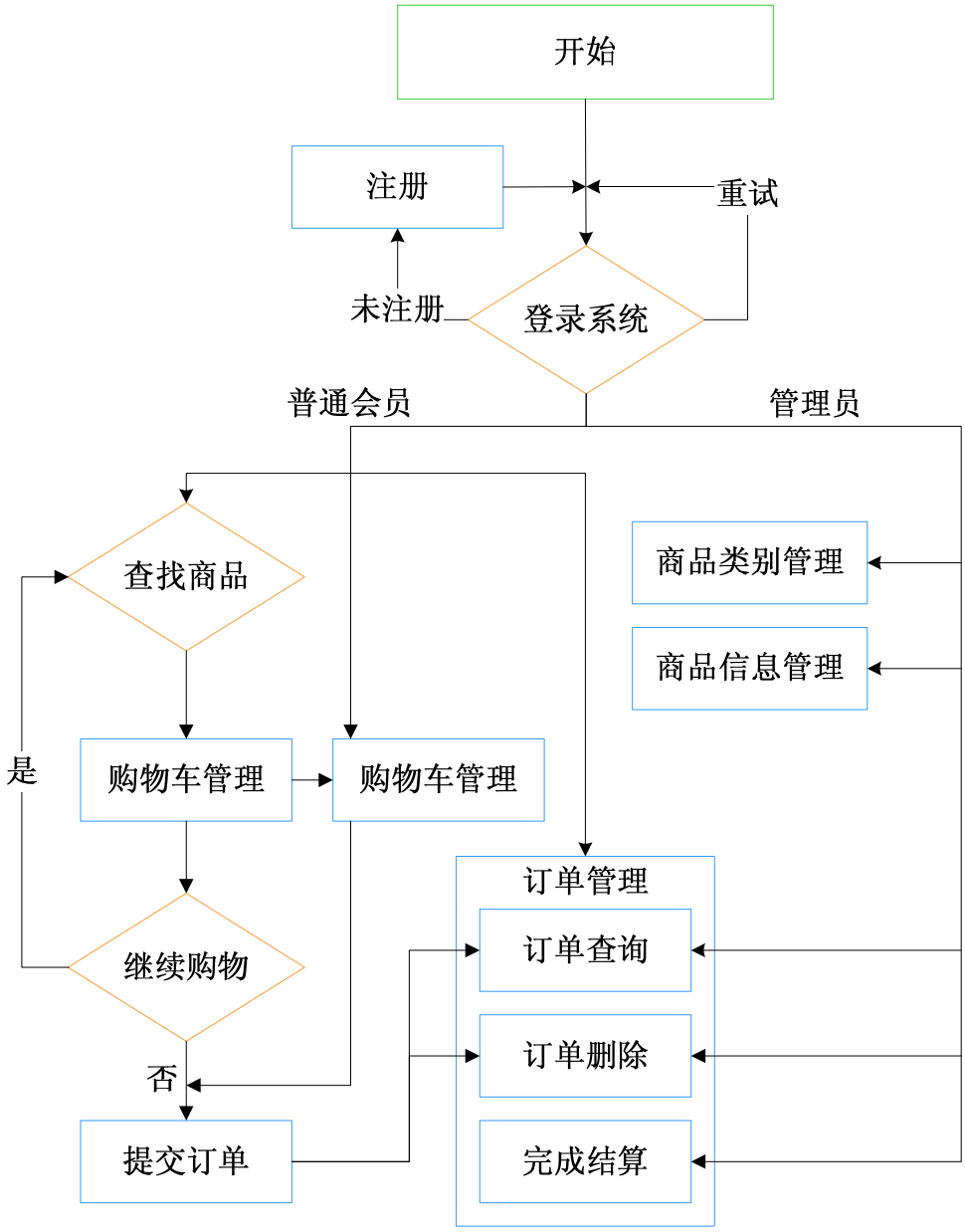

图 4.5 系统主要业务流程

4.4 状态模型

UML 标准建模语言中的状态图主要目的是用来描述一个相关对象在它的生存期间的各种动态行为，它可以表现为对象所经历的各种状态序列，从而会引起状态转移的类似事件（Event），以及因为状态转移而产生的相关操作行为动作（Action）。一般情况下，可以使用状态机来对对象生命周期进行建模，状态图是用于显示状态机的最优化工具，状态图的重点是与描述相关状态图的相关控制流。本商城系统中最主要的有多种状态转换的对象是购物订单，本节用状态图对订单的状态转换进行了分析。

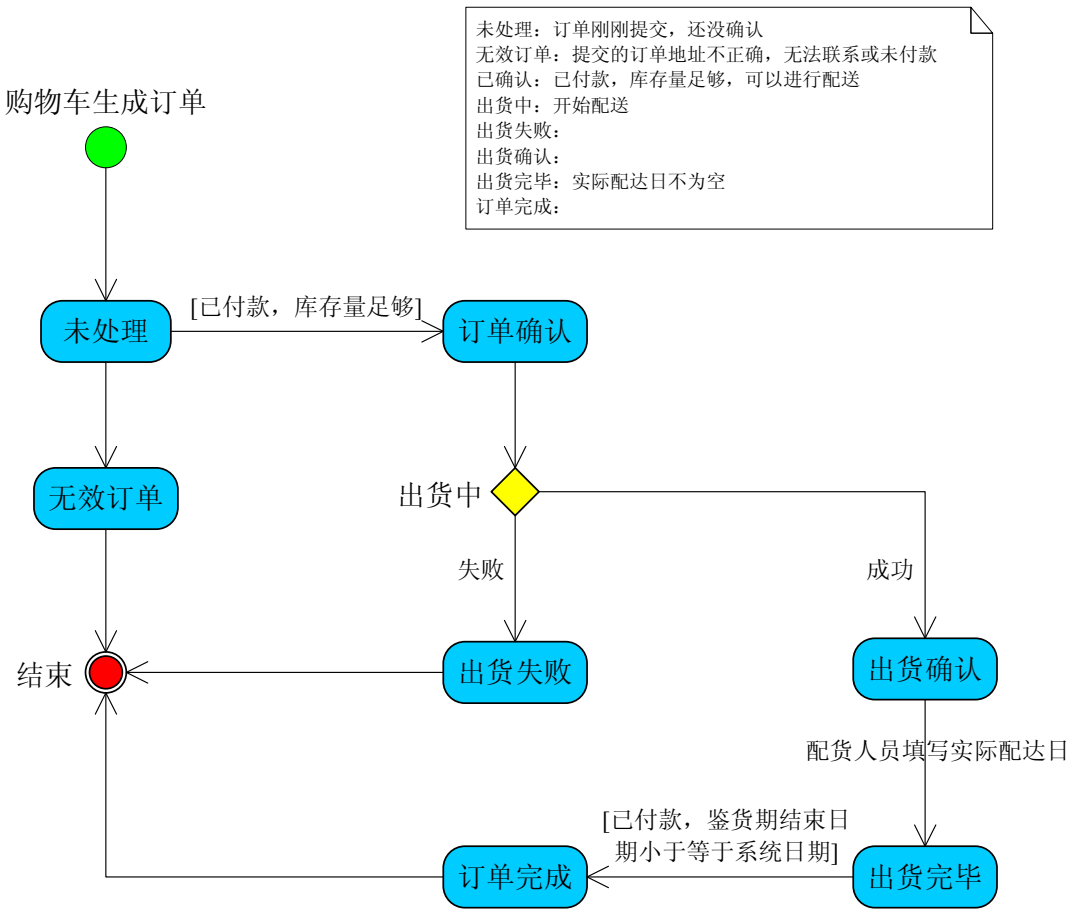

图 4.6 订单状态转换过程

4.5 本章小结

本章主要介绍了系统概要设计过程中涉及到的 UML 建模实例、业务流程和状态模型，还针对系统体系结构进行了具体说明，具体包括逻辑结构和数据流程。

在系统建模和系统分析阶段中，先使用 UML 标准建模语言技术绘制各个阶段和细节的相关分析图，这些分析图都是用来进行全面描述待开发的系统细节。主要包括状态图、用例图、活动图、类图、协作图、序列图、配置图、构建图等各种图形，具体要绘制哪些分析图都得根据实际的具体情况来决定。本章节应用 UML 标准建模语言，详细地从系统用例、主要业务流程、核心功能状态各方面对系统的需求进行了解析，为下一章的系统设计提供了重要的依据。

第 5 章 系统详细设计

5.1 功能性要求设计

（1）会员注册登录

顾客可以通过填写注册信息注册成为会员，会员登录后才能进行购物车的管理与意见反馈，未登录的会员或未注册的顾客只能检索与浏览商品信息。顾客注册时可以选择不同的会员级别，有不同的打折率。但是若顾客不符合要求或是有重大表现，后台的管理员可以对其会员级别进行调整。在注册信息时，系统会对注册信息进行有效性的验证，有效才会注册成功。

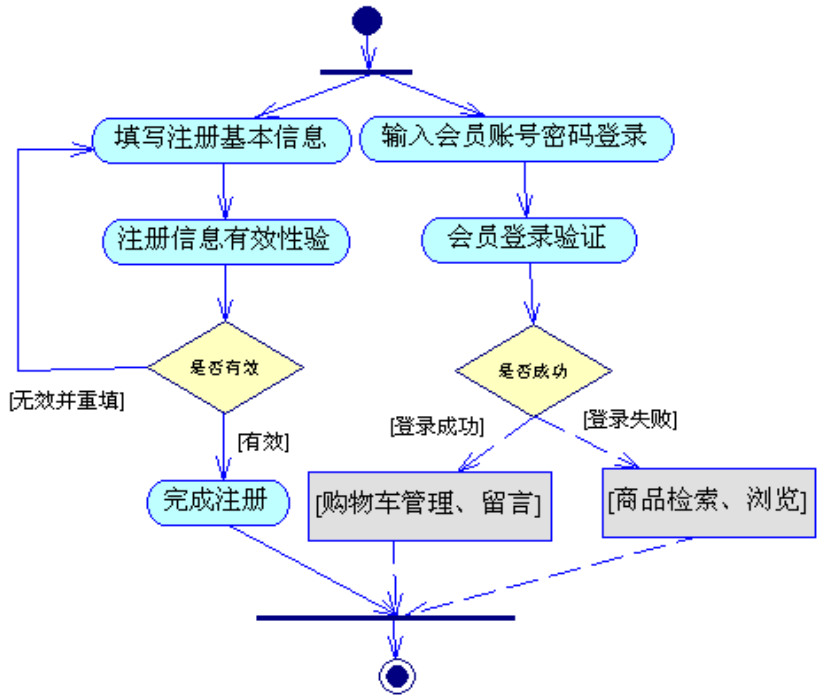

图 5.1 会员注册登录模块活动图

（2）商品检索浏览

顾客可以直接查看商品的全部信息（如商品名称、商品图片、商品型号、商品价格、生产厂商、生产日期等），也可以根据不同的商品种类浏览商品信息，也可以透过输入关键字对商品进行检索，查看其信息。

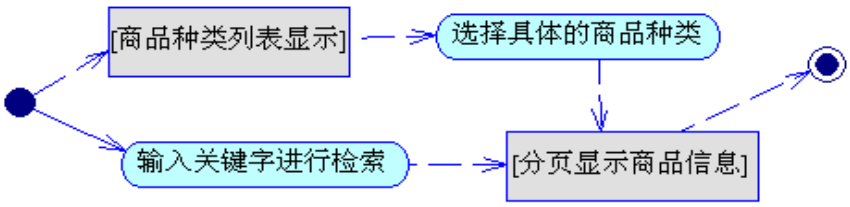

图 5.2 商品检索浏览模块活动图

（3）购买商品

顾客经过登录后便可以购买商品，将选中的商品增加到购物车中、修改选购商品的数量、删除购物车中的商品、查看购买商品的总价、在线下订单及订单查看等操作。

购物车是网络商城的核心部分，它记录了顾客在购买商品过程中的数据变化情况，最后结算和发送订单都要依赖于购物车中记录的商品信息。

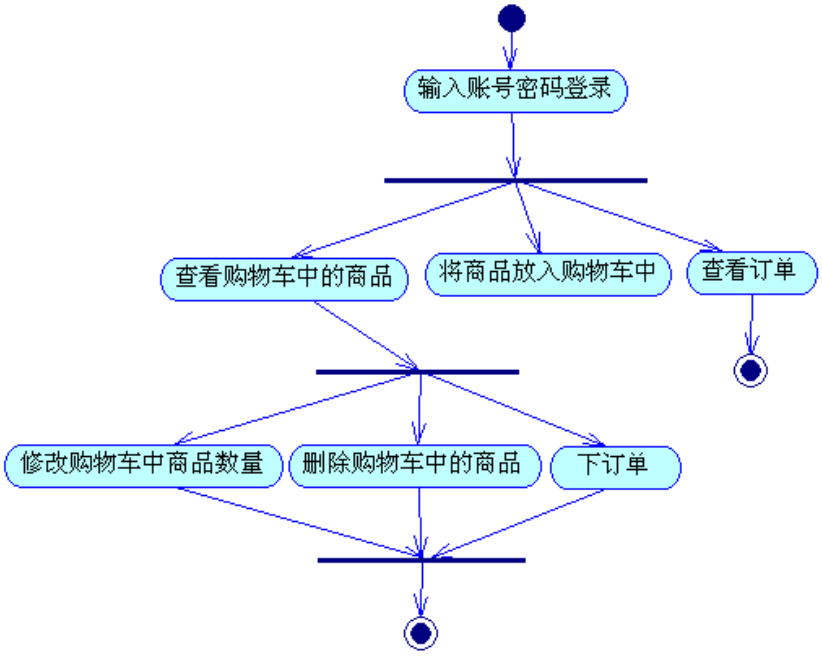

图 5.3 购买商品模块活动图

购物车可以顺理成章地理解为是网络顾客在网络购物过程中，用来临时存放自己心仪物品信息的临时数据的缓冲区。因此，购物车缓冲区必须在整个网络购物的生存周期中都必须存在，在网络顾客执行登录网站后就会自动创建（初始化状态为空状态），在网络购物过程中购物车会不断增加、删除发生变化，在每次结算后，发送订单完成后将购物车清空，从而准备等待下一次的网络购物过程，在网络顾客注销离开网站时，达到自行撤销的目的；并且要求，同时参与购物的顾客的购物车必须相互独立。为达到这些目的，可以采用 Cookie 的网络设置方法，这样每次用 Web 服务器打开一个会话，会得到一个 Cookie。这个 Cookie 严格用于将客户端的浏览器和为该会话预留的服务器内存链接起来，当会话结束时撤销；而且对同一网站，不同的访问用户，它们拥有各自的 Cookie，所以在此我们就可以利用 Cookie 实现购物车的功能，具体方法为：当用户注册并登录网站后，在客户端创建一个 Cookie，里面记录了有关用户的信息。

（4）会员评论留言

顾客登录成功后，进入留言信息填写页面，可发表自己的意见或留言，经验

证有效方可提交。也可进入查看别人的留言或管理员的回复。

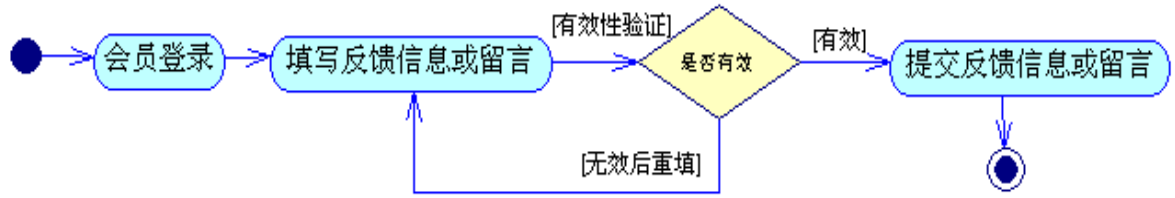

图 5.4 会员评论留言模块活动图

（5）管理员登录

管理员输入账号密码可以在后台登录，登录验证模块根据输入的账号和密码来进行识别拥有不同权限角色的系统管理员，然后再进入各种不同的相应后台的管理模块。账号和密码如果输入错误，系统会给出错误提示并且要求用户执行重新输入的过程。

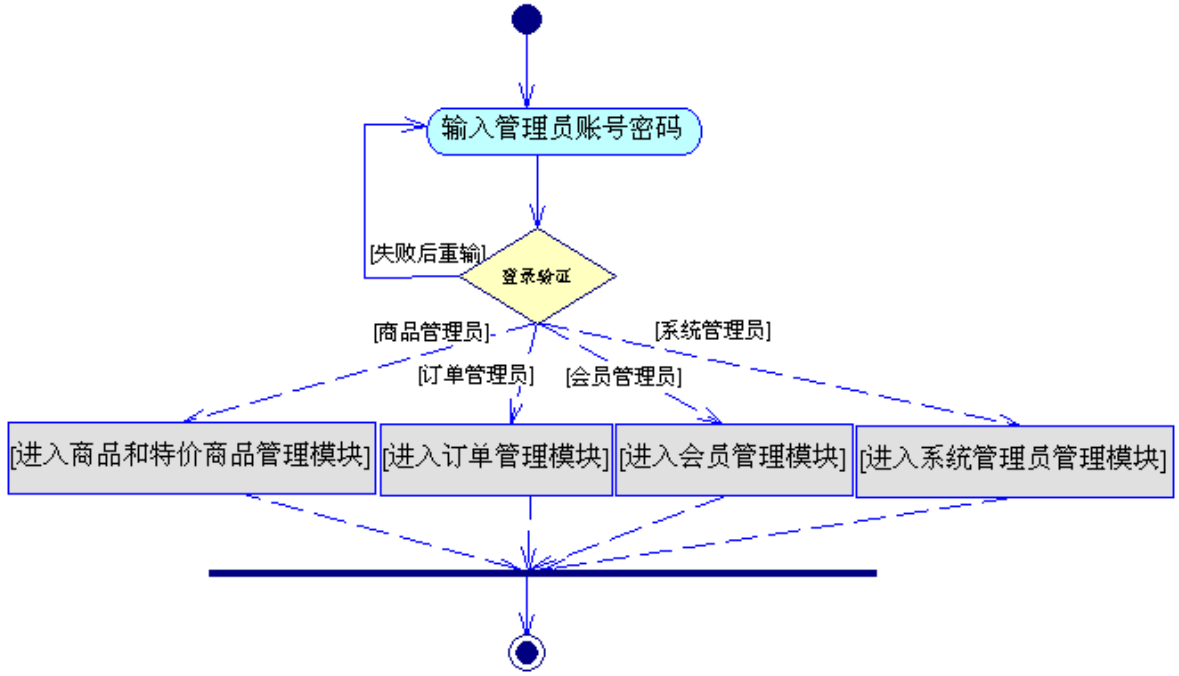

图 5.5 管理员登录模块活动图

（6）商品管理

商品管理员拥有可查看商品各种种类和各种信息的权限，商品管理员可执行新增、修改、删除各种商品种类，同样可以查看商品和特价商品，对其进行新增、修改、删除商品和特价商品的信息等操作。

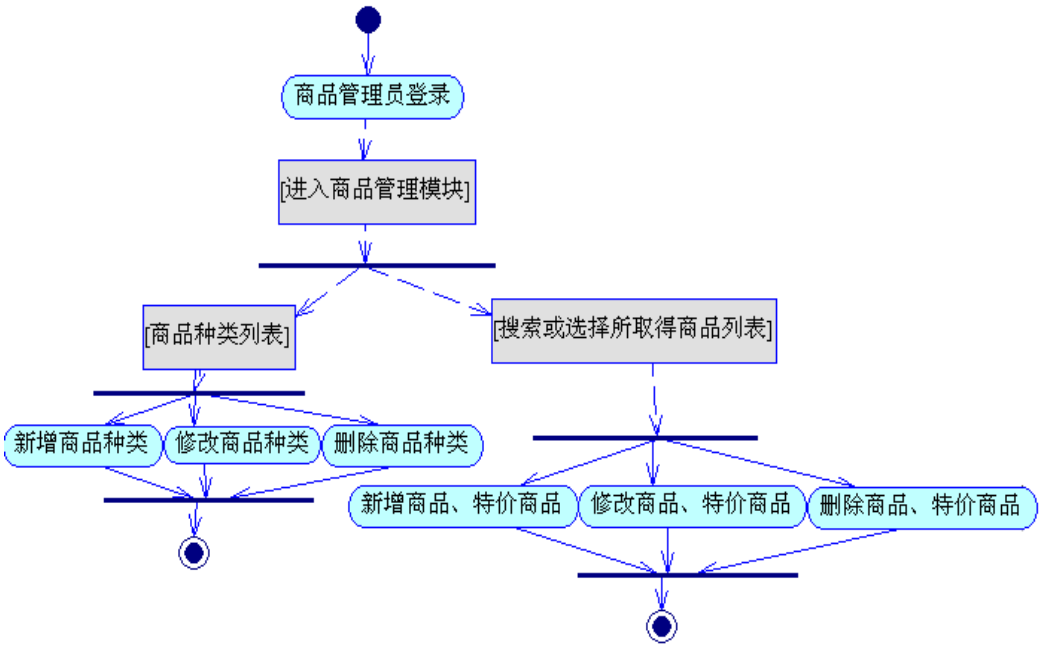

图 5.6 商品管理模块活动图

（7）订单管理

订单管理员执行登录过程以后，可以根据订单的日期范围、订单的完成状态、订单的编号等详细信息来获取订单列表，同时可以查看某张订单的具体详情。订单管理员也可以修改订单的相关完成状态以及订单中的具体商品或具体数量，还可以删除本订单及其订单中涉及的相关商品。

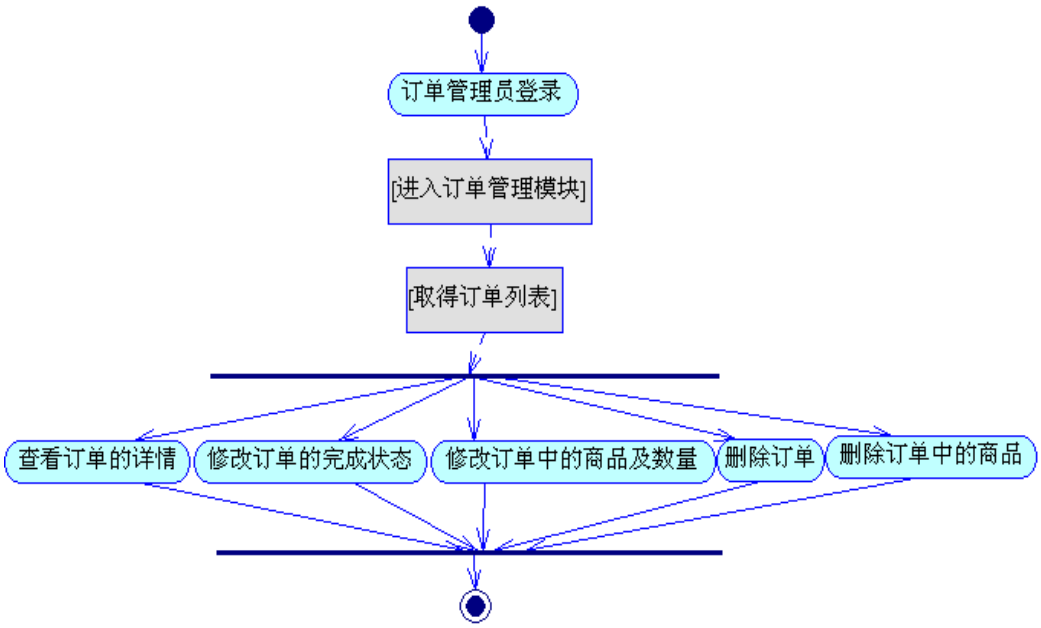

图 5.7 订单管理模块活动图

（8）会员管理

会员管理员执行登录以后，会员管理员可根据相关注册时间的范围、会员等级级别及会员相关 ID 来拉取相关会员的列表，然后针对某单个会员进行查看详细资料信息、调整会员的级别与执行删除会员操作；会议管理员也可根据留言的

时间区域范围、回复状态及会员相关 ID 拉取相关留言列表，然后针对某个不同留言进行内容查看、内容回复与内容删除等相关操作。

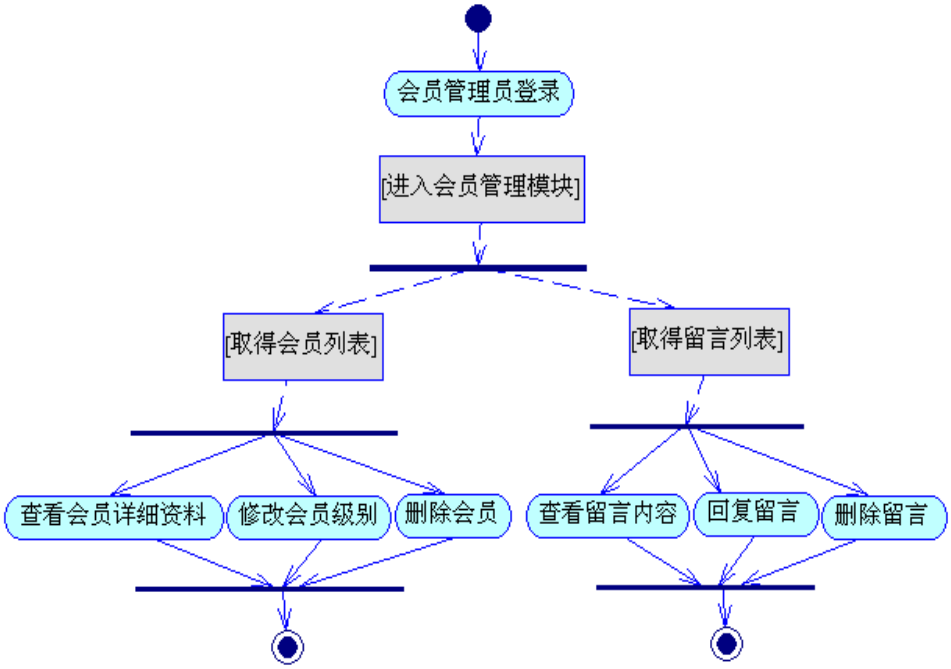

图 5.8 会员管理模块活动图

（9）管理员管理

系统管理员在执行登录操作以后，可以根据管理员列表进行管理员查看详细资料操作、新增管理员、修改管理员信息以及删除管理员操作。

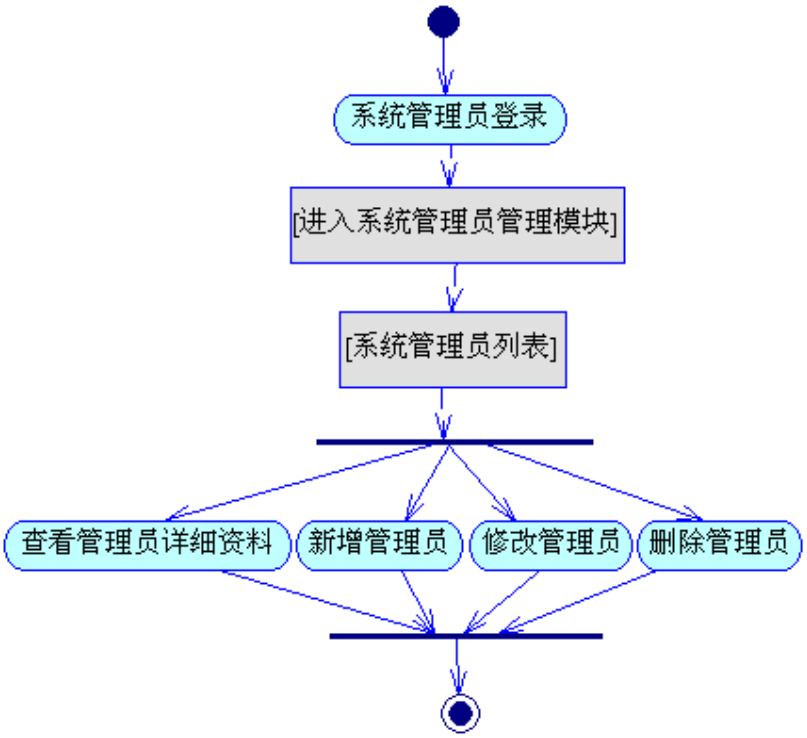

图 5.9 管理员管理模块活动图

（10）退出系统

在前台部分和后台部分都设置了退出系统模块。在前台部分，会员退出系统后，仍可以浏览、检索商品的操作。在后台部分，管理员退出系统，即回到管理员登录页面。

5.2 非功能性要求设计

本系统高度重视非功能性方面的需求，通过分析研究现阶段最新的电子商务网站开发技术，针对系统的各项非功能性需求提出了相应的解决方案。

（1）安全

A．双重权限验证

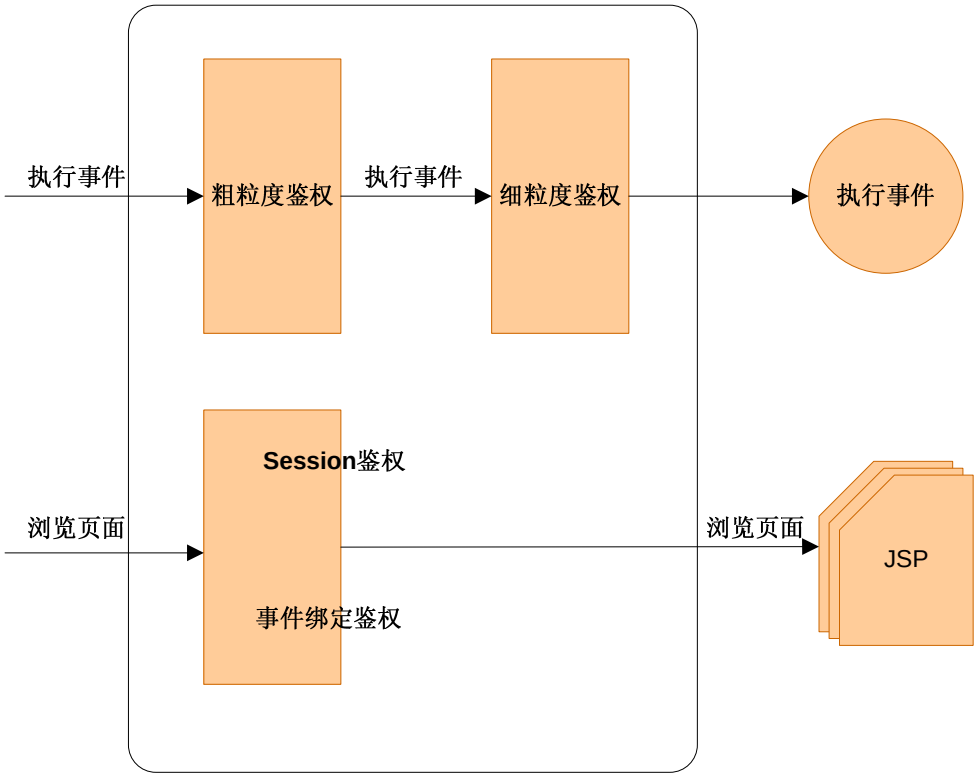

图 5.10 双重权限验证

对于电子商务系统，后台安全非常重要，本系统采用了基于 Session 的权限验证和 AOP 事件拦截双重鉴权机制，低耦合、可分离，通过数据库配置即可实现对指定事件进行鉴权。

把一个操作（读取数据、删除商品等），定义为一个事件。所有权限配置以事件为中心，把事件跟角色关联，就形成了最基本的粗粒度鉴权。

后台每个页面，都可以通过数据库配置和事件进行绑定，从而实现了浏览级别的角色鉴权。

对于页面内的局部数据鉴权也提供一个鉴权标签，可以对页面中的某块内容跟多个事件进行绑定，只有具备所有事件权限的角色才能看到相关内容[9]。

B．URL 重写

URL 重写就是可以把一般访问 lisy.jsp?a=1 改写成 list-1.html 这样的形式。本系统提供的 URL 重写是整合了静态文件的生成，通过参数设置，可以决定被重写的 URL 是否生成静态页面还是只是一个重写后的动态访问。本商城全站都是 xxx.html 这样的静态网址访问形式，但实质上包含了商品页面、货架列表等这些真正静态页面的访问和像购物车、下单流程这些伪静态页面的访问。

使用了 URL 重写主要有两大好处，一是增强了搜索引擎对数据页面的收录度。二是增强了网站的安全性，有效地防止了 SQL 注入攻击，因为重写的 URL 对参数输入格式都有严格的限制。

（2）性能

A．页面静态化

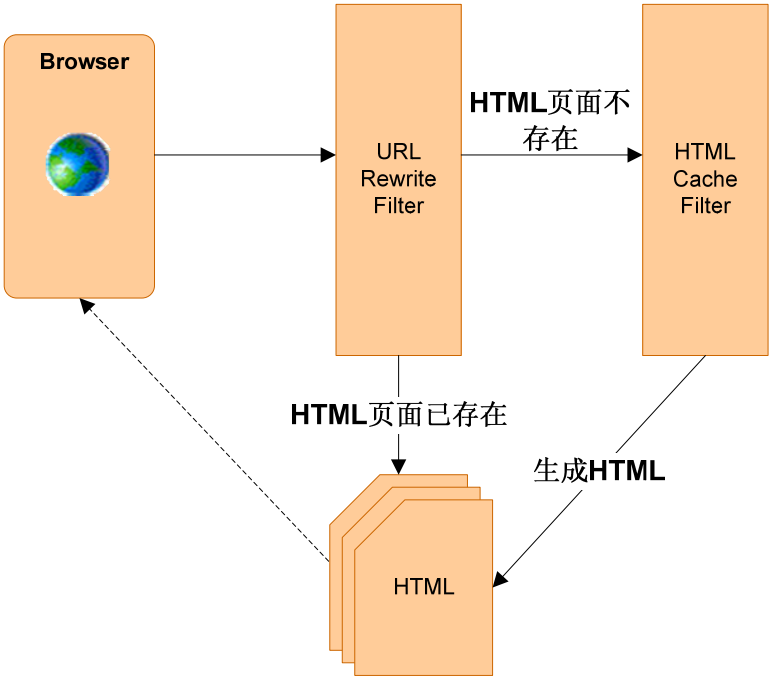

图 5.11 自动生成静态 HTML 机制

为了提高系统性能，减短系统响应时间，本系统采用页面静态化技术，把站点做成静态的，用后台的发布系统发布出来。静态页面在性能上具有不少优势。

另外很重要的一点是，静态页面可以引起搜索引擎的重视，而搜索引擎对动态页面的重视远远比不了静态页面。由于大量动态网页的存在，使得有些全动态网页的网站失去很多被用户发现的机会。为了网络营销的需要，如果网站无法全部用静态网页实现，应采用静动结合的基本方式，即能用静态网页解决的决不用动态网页，尤其是对一些重要网页要采用静态的方式来处理。静动结合反映的是一种网络营销基本思想。

本系统采用的是一种触发式自动 HTML 静态缓存技术，具有可分离、无入

侵业务、低耦合和高可配性特点，使页面静态化成为一个相对独立的模块。开发人员在开发业务逻辑功能的时候，不需要考虑生成静态页面的问题，也不需要单独做一个生成静态页面的功能模块，因为这些都可以通过后期简单配置实现，而且还不需要考虑页面数据发生变化时，数据同步问题，页面重新生成问题，这些都由页面静态化模块自动处理[9]。

B．两级缓存

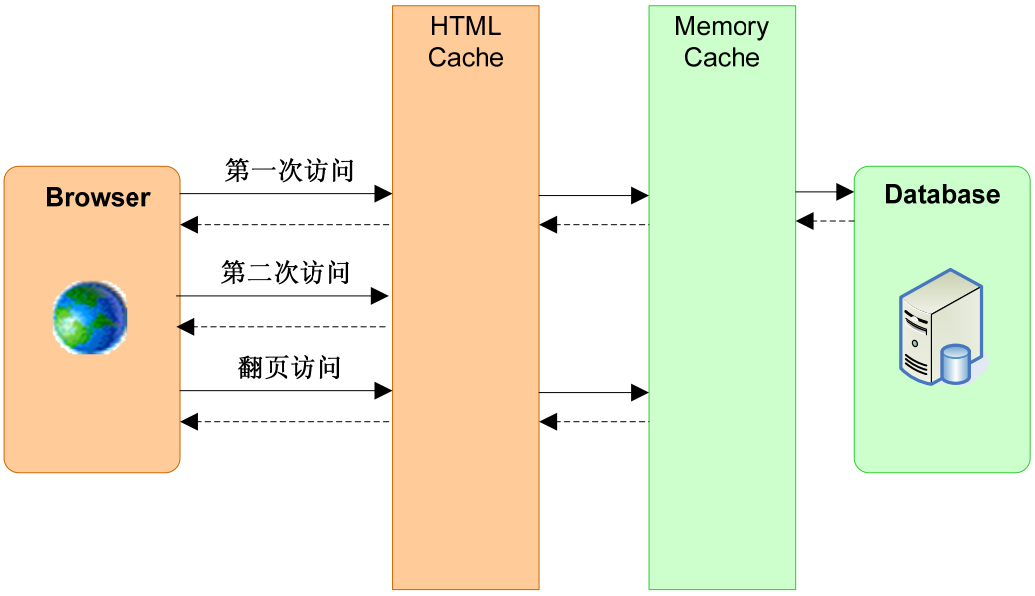

图 5.12 两级缓存实现

本系统底层 DAO 数据库操作集成了数据内存缓存和数据智能同步功能，在开发业务的时候使用数据库操作函数，即可透明实现数据内存缓存，同时开发业务流程时也不需要考虑如何把内存数据跟数据库最新数据进行同步问题，因为这些是独立封装的模块[9]。

把内存缓存和静态文件缓存综合起来，实质上提供了一种二级缓存的解决方案。次方案具有无入侵、低耦合、非常灵活的特点。

我们可以看一个列表商品数据的例子，看看二级缓存的好处：

（1）当第一次访问一个商品列表页面时，所有数据从数据库读取，并同时把页面中各个数据模块存放在内存中，同时把页面生成 HTML 静态文件。

（2）当第二次访问同一个商品列表页面的时候，系统直接把请求转到静态页面。

（3）当翻到第二页时，该页面的新商品数据会直接从数据库读取，但是页面中其他公共数据模块会从内存中读取。

C．多线程异步

商城里面的发送邮件和管理商品索引，都是比较耗时的操作。比如在下单的时候，连接邮件服务器时间过长，那么下单流程就卡在那里了，严重影响了用户

体验。次数多了，会直接影响到用户商城的业务。

对于类似这些特殊任务，本系统都会创建一个新的异步线程去执行，使得用户感觉不到这些耗时操作的存在，从整体来看，也是提升了系统的执行效率。

（3）可维护性

本系统提供了一套很灵活的粗细粒度权限架构，以角色-事件为配置核心，既可以对单独事件进行权限配置，也可以对页面进行事件绑定，对页面中局部内容进行事件绑定等粗粒度权限控制。同时，开发用户可以创建一个细粒度控制类，实现 MiniPrivilegeIFace 接口，最后在 mini_privilege.xml 进行装配，把粗粒度事件和细粒度控制类绑定，从而非常灵活的实现来粗细粒度的权限控制。如果以后需要更改细粒度控制策略，只需要修改细粒度控制类或者是重新装配一个策略，具有很高的可维护性和扩展性[9]。

（4）用户体验

A．Velocity 模板技术

本系统前台页面都采用模板技术制作，一个商城，可以拥有无数套模板，每套模板，独占一个文件夹，通过商城后台的“网站模板管理”功能，能够非常方便的切换不同模板。切换新模板后，前台静态页面会自动重新生成，但是商品页面，货架页面等链接，都保持不变。这样就不会破坏被搜索引擎收录的友好度。

本商城采用了 Velocity 模板技术，商城前台模板制作就是对静态的 HTML 模板文件进行编辑，数据展示、拆分，都是用易于理解和容易上手的模板语言，使得即使是美工人员，也很容易编辑、排版、制作商城模板。

本系统采用多套模板切换技术，每套模板独占一个文件夹，前台访问连接不改变，切换新模板后，HTML 自动重新生成[10]。

B．Lucence 全文索引技术

Lucence 是一个开源全文检索技术，支持海量数据高性能搜索，可以用于开发搜索引擎。本系统通过集成 Lucence 技术，提供相关操作函数，商城业务通过调用这些函数，实现对商品 ID，商品名称，商品简介，商品属性，商品编号，商品关键字进行全文索引，从而实现普通数据库不可能实现的海量商品多字段高效全文检索，让用户非常容易搜索到他们想了解的商品，并且大量搜索都不会对数据库造成压力，从另一个角度来看，也是提高了系统的性能[11]。

（5）成本预算

A．技术和软件的选择

选择何项开发技术，其实很大程度上取决于该技术开发、运行的成本。有的技术需要昂贵的配套软硬件的支持，一般企业很难承受。此时需要考虑是否可以采用替代的方案，或去除某些不是非常必要的部件，从而可以选择另一款性价比高的设备。

从 Java EE 的范围来看，有很多可选择的应用服务器，如 Oracle WebLogic、IBM Websphere、JBoss 等。有很多可选择的数据库服务器，如 MS SQL Server、Oracle Database、MySQL 等。大公司出品的产品固然功能强大、性能优越，但价格太高，固性价比不高。在系统需要设计范围内性价比高的服务器是较好的选择。

基于上述原则，结合本系统的设计需求，选择 Tomcat 和 MySQL 这两款业界赞誉声很高的免费的 Java EE Web 服务器和数据库服务器。

B．运维优化

本系统采用了一种开放式的定时任务实现，开发人员可以创建一个任务类，通过继承 TimerTask，并在 run（）函数实现任务内容，最后通过 asynchronized.xml 进行装配，根据需求，可把任务配置成隔多少秒执行一次，隔多少小时执行一次以及指定隔多少天几时几分执行一次。

通过配置成定时任务，本商城自动进行一些数据统计、清理过期静态文件、清理临时上传图片、清理多余日志文件、自动备份数据库等操作。这些任务放到夜间执行，以便不影响白天的系统访问，并大大减少了运维的人力成本。

5.3 数据库设计

5.3.1 概念数据模型

本系统概念数据模型包括 Admin、Message、ProductCategory、MemberGrade、Product、Member、SelectedProduct、Cart、Order 九个模型，其中 Admin 模型与 Message 模型是采用 Admin_Message 消息进行传递，Message 模型与 Member 模型是采用 Member_Message 消息进行传递，Member 模型与 MemberGrade 模型是采用 Grade_Member 消息进行传递，Member 模型与 Order 模型是采用 Member_Order 消息进行传递，Member 模型与 MemberGrade 模型是采用 Grade_Member 消息进行传递，Cart 模型与 Order 模型是采用 Cart_Order 消息进行传递，Cart 模型与 Member 模型是采用 Member_Cart 消息进行传递，Cart 模型与 SelectedProduct 模型是采用 Cart_SelectedProduct 消息进行传递，SelectedProduct 模型与 Product 模型是采用 Product_Selected 消息进行传递，Product 模型与 ProductCategory 模型是采用 Product_Category 消息进行传递。

Admin 模型中主键是 AdminID，Message 模型中主键是 MessageID，MemberGrade 模型中主键是 GradeID，Member 模型中主键是 MemberID，Product 模型中主键是 ProductID，ProductCategory 模型中主键是 ProductCategoryID，SelectedProduct 模型中主键是 SelectedProductID，Cart 模型中主键是 CartID，Order 模型中主键是 OrderID。

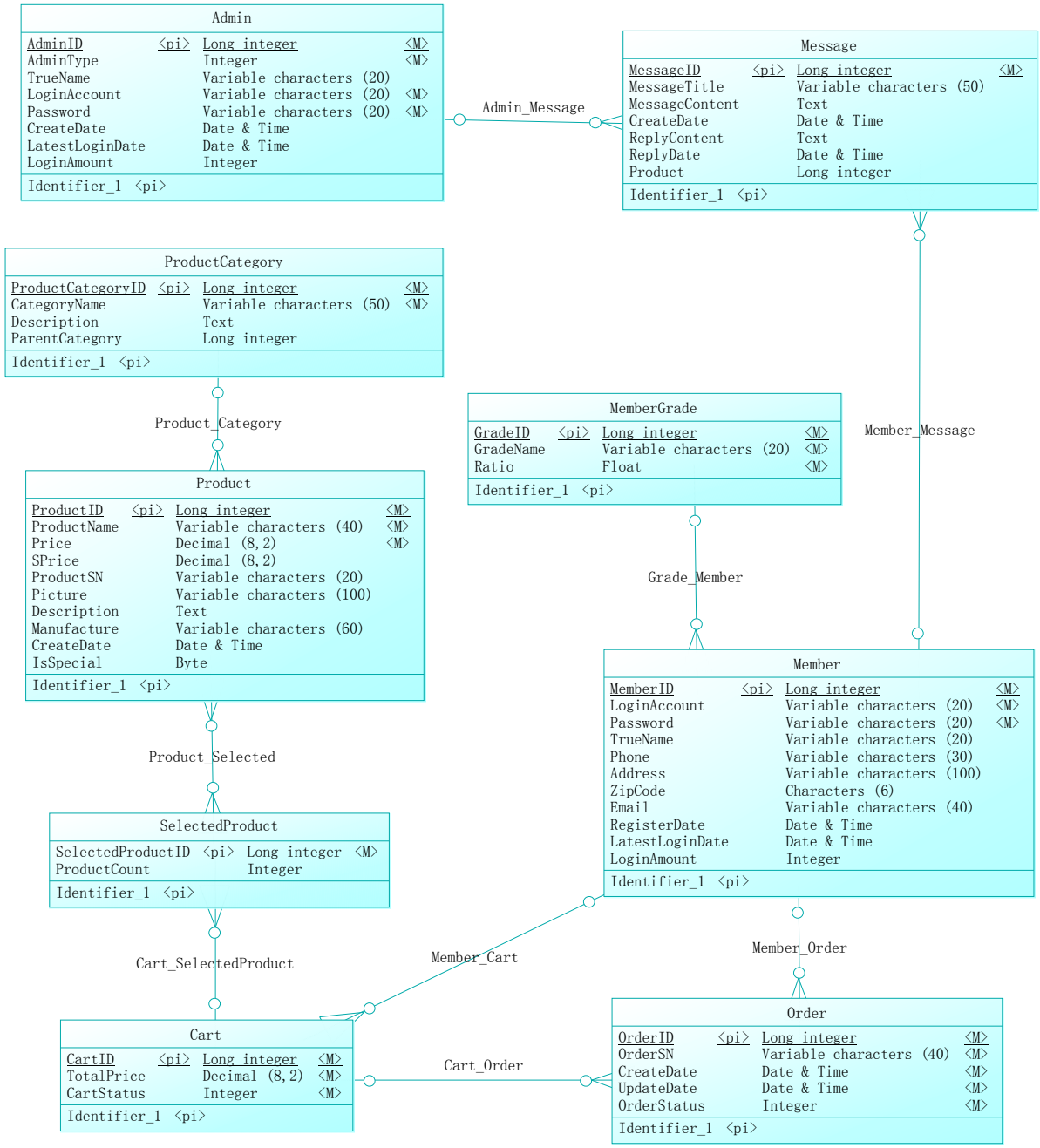

图 5.13 概念数据模型

5.3.2 数据表设计

表 5.1 商品类别表（ProductCategory）

字段名称	数据类型	主键	是否允许为空	描述
ProductCategoryID	Long Integer	√		商品类别的 ID 号
CategoryName	Var Char（50）			类别名称
Description	Text		√	类别描述
ParentCategory	Long Integer		√	父类别

表 5.2　商品表（Product）

字段名称	数据类型	主键	是否允许为空	描述
ProductID	Long Integer	√		商品的 ID 号
ProductCategoryID	Long Integer	外键		商品类别的 ID
ProductName	Var Char（40）			商品名称
Price	Decimal（8,2）			商品价格
SPrice	Decimal（8,2）		√	商品特价
ProductSN	Var Char（20）		√	商品型号
Picture	Var Char（100）		√	商品图片
Description	Text		√	商品描述
Manufacture	Var Char（60）		√	生产厂家
CreateDate	DateTime		√	生产日期
Special	Byte		√	有无特价（0：无，1：有）

表 5.3　商品选购表（SelectedProduct）

字段名称	数据类型	主键	是否允许为空	描述
SelectedProductID	Long Integer	√		选购 ID 号
CartID	Long Integer	外键		购物车 ID
ProductID	Long Integer	外键		商品 ID
ProductCount	Integer		√	商品数量

表 5.4　购物车表（Cart）

字段名称	数据类型	主键	是否允许为空	描述
CartID	Long Integer	√		购物车的 ID 号
MemberID	Long Integer	外键		会员 ID
TotalPrice	Decimal（8,2）			总金额
CartStatus	Integer			购物车状态

表 5.5　订单表（Order）

字段名称	数据类型	主键	是否允许为空	描述
OrderID	Long Integer	√		订单 ID 号
CartID	Long Integer	外键		购物车的 ID 号
MemberID	Long Integer	外键		会员 ID
OrderSN	Var Char（20）			订单编号
CreateDate	Date Time			订单生成日期
UpdateDate	Date Time			订单更新日期
OrderStatus	Integer			订单状态

表 5.6　会员级别表（MemberGrade）

字段名称	数据类型	主键	是否允许为空	描述
GradeID	Long Integer	√		会员级别的 ID 号
GradeName	Var Char（20）			级别名称
Ratio	Float			优惠百分比

表 5.7　会员表（Member）

字段名称	数据类型	主键	是否允许为空	描述
MemberID	Long Integer	√		会员的 ID 号
MemberGrade	Long Integer	外键		会员级别的 ID
LoginAccount	Var Char（20）			登录账号
Password	Var Char （20）			登录密码
TrueName	Var Char （20）		√	真实姓名
Phone	Var Char （30）		√	联系电话
Address	Var Char（100）		√	联系地址
ZipCode	Char （6）		√	邮政编码
Email	Var Char（100）		√	电子邮箱
RegisterDate	Date Time		√	注册日期
LastLoginDate	Date Time		√	最近登录日期
LoginAmount	Integer		√	登录次数

表 5.8　管理员表（Admin）

字段名称	数据类型	主键	是否允许为空	描述
AdminID	Long Integer	√		系统管理员 ID 号
AdminType	Integer			管理员类型
TrueName	Var Char（20）		√	管理员真实姓名
LoginAccount	Var Char（20）			管理员登录账号
Password	Var Char（12）			管理员登录密码
CreateDate	Date Time		√	创建日期
LastLoginDate	Date Time		√	最近登录日期
LoginAmount	Integer		√	登录次数

（管理员类型分为 4 种：1—商品管理员，2—订单管理员，3—会员管理员，4—系统管理员）

表 5.9　留言表（Message）

字段名称	数据类型	主键	是否允许为空	描述
MessageID	Long Integer	√		留言 ID 号
MemberID	Long Integer	外键		会员 ID
AdminID	Long Integer	外键		管理员 ID
Title	Var Char（50）		√	留言标题
MessageContent	Text		√	留言内容
CreateDate	Date Time		√	留言日期
ReplyContent	Text		√	回复内容
ReplyDate	Date Time		√	回复日期
Product	Long Integer		√	商品 ID

（会员留言分两种，一种是直接向管理员反馈信息；另一种是对商品的评论，需关联到商品 ID）

5.3.3 数据库优化

系统架构为商城提供了有力的性能保证，另外在商城的数据库设计上也做了很多的考虑。比如商品的点击数和货架的点击数，这个是每次访问都需要递增的一个字段，如果按照传统的方式每次访问都 update 一次，那么在访问量大时，这将会是一个瓶颈。所以本系统在数据库里面设计了一个临时表，专门用来存放商品和货架的点击记录。每一次访问商品或货架，都会往表里面插入一条数据，而不是直接更新商品表或货架表的点击数字段，也就是把 update 操作转换为 insert 操作（规避了频繁 update 造成的性能瓶颈），然后通过装配一个定时任务，每隔 12 小时汇总更新一次商品和货架的点击数。

总的来说，在设计数据库时尽可能的通过一些手段方法来优化表的设计，减少和错开一些不必要的更新，以便为系统提供更好的性能。

5.4 界面设计

5.4.1 主界面设计

主界面力图给顾客一个产品丰富、查找方便、功能强大的印象，但又不能带给顾客界面繁杂、到处都是广告或无用信息的感受。所以要借鉴成熟的、已经得到广大用户体验认可的网络商城界面模式。

在布局上，采用 Web 标准，用 CSS 来做整体的布局。将页面划分为几个主要的功能区域：

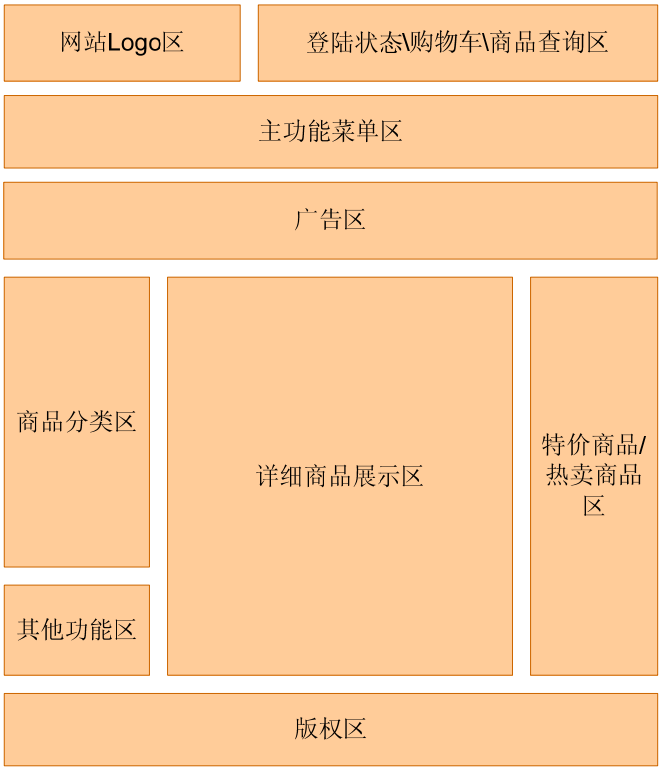

图 5.14 系统主界面布局

在色彩上，要有一目了然的主颜色，辅以其他几种对比度不大的色彩。避免色彩过多，给顾客造成光怪陆离的感觉。

在页面大小上，页面宽度要兼顾当今主流和上一代显示屏的分辨率；页面高度既不能仅仅一页，又不能列举太多商品以至于顾客要滚动好几屏幕才能全看完。合适的程度是两至三页大小，即用户只需滚动最多两屏就可见页脚的商城版权区。

5.4.2 数据编辑界面设计

数据编辑界面要尽可能减少用户敲键盘的次数，而尽量提供候选项，这样用户可以通过点击鼠标快速完成一些操作。

图 5.15 数据编辑选项功能设计

另外，提供功能强大的数据编辑专用插件用于商品的详细介绍，使得商品的详细介绍可以有不同的字体大小、字体样式、编排格式、图片等。

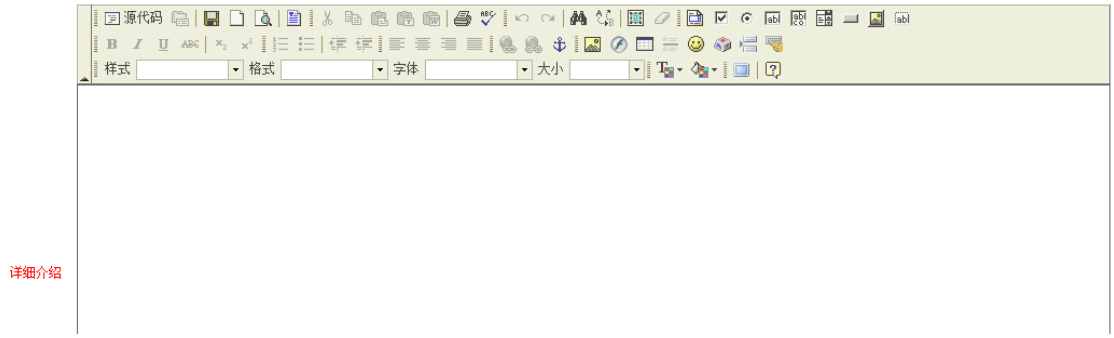

图 5.16 数据编辑专用插件

5.4.3 数据查询界面设计

数据查询界面设计本着简约、快捷的原则，提供模糊搜索和精确搜索两种模式。

顾客可以输入关键字进行查询，范围可以限制在某一大类或某一小类，也可以是从所有商品中查询：

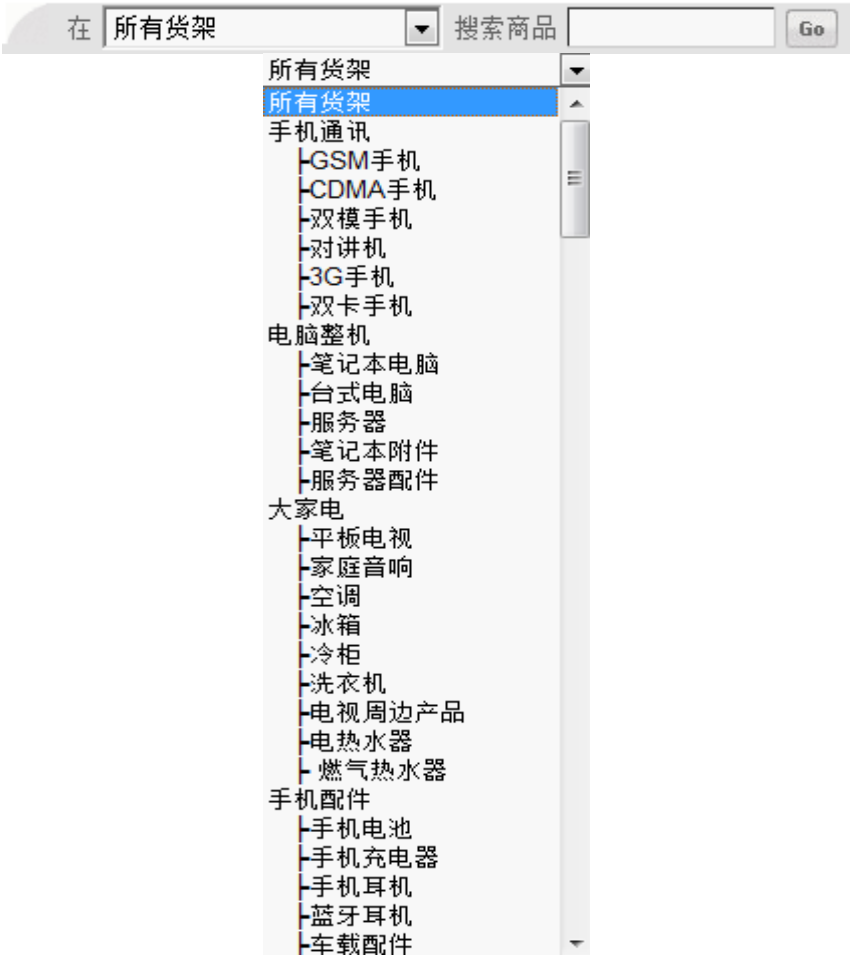

图 5.17 数据查询界面设计 1

顾客也可以在快速导航栏中选择相应的选项快速得到某一类商品列表：

图 5.18 数据查询界面设计 2

5.4.4 列表界面设计

顾客查看的商品列表除了要提供明确、重要的商品信息，如商品的价格、品牌等之外，还要注重美观、直观，所以要结合显示图片的方式，并归类显示。

图 5.19 首页商品列表设计

管理员查看的商品列表则要兼顾管理功能，使管理员可以快速地对某个或某些商品进行编辑修改乃至批量处理。所以在列表中除了要显示商品的名称、编号、所属类别之外，还要加上编辑功能，和一些用户的反馈信息如商品被点击次数等。

	商品ID	商品名称	商品货架/品牌	商品价格	所属群组	上移	上/下架	货架推荐	点击	操作
□	20344	100 | 飞利浦（PHILIPS）电饭煲HD4723	电饭煲 /飞利浦	189.00	最新到货 | 精品推荐 |		☑	□	2	
□	20343	100 | 苏泊尔（Supor） CFXB40YB4A-70陶晶电饭煲	电饭煲 /苏泊尔	188.00	精品推荐 |		☑	☑	1	
□	20342	100 | 格力（GREE） GD-501D电脑版圆形电饭煲	电饭煲 /格力	387.00	购物圈首页推荐 | 精品推荐 |		☑	☑	0	
□	20341	100 | 格兰仕（Glanz）B601T-30F5G电脑方形电饭煲	电饭煲 /格兰仕	198.00	精品推荐 |		☑	☑	0	
□	20340	100 | 美的（Midea）商用12升电饭煲MB-SYJ120	电饭煲 /美的	998.00	详细新闻推荐 | 精品推荐 |		☑	☑	0	
□	20339	100 | LG XM-110鼠标	鼠标 /LG	59.00	精品推荐 |		☑	□	0	
□	20338	0 | 雷柏（Rapoo）nano3100 2.4G笔记本型无线光学鼠标	鼠标 /雷柏	139.00	购物圈首页推荐 | 最新到货 | 新闻列表推荐 | 精品推荐 |		☑	☑	2	

图 5.20 管理后台商品列表设计

5.5 本章小结

本章主要介绍了功能性要求设计和非功能性要求设计，功能性要求具体包括会员注册登录、商品检索浏览、购买商品、会员评论留言、管理员登录、商品管理、订单管理、会员管理、管理员管理和退出系统，非功能性要求包括安全、性能、可维护性、用户体验和成本预算，然后介绍了数据库设计部分，具体包括概念数据模型、数据库设计和数据库优化，最后介绍了系统界面设计，包括主界面设计、数据编辑界面设计、数据查询界面设计和列表界面设计。

第 6 章 系统实现

6.1 网络环境

6.1.1 网络拓扑图

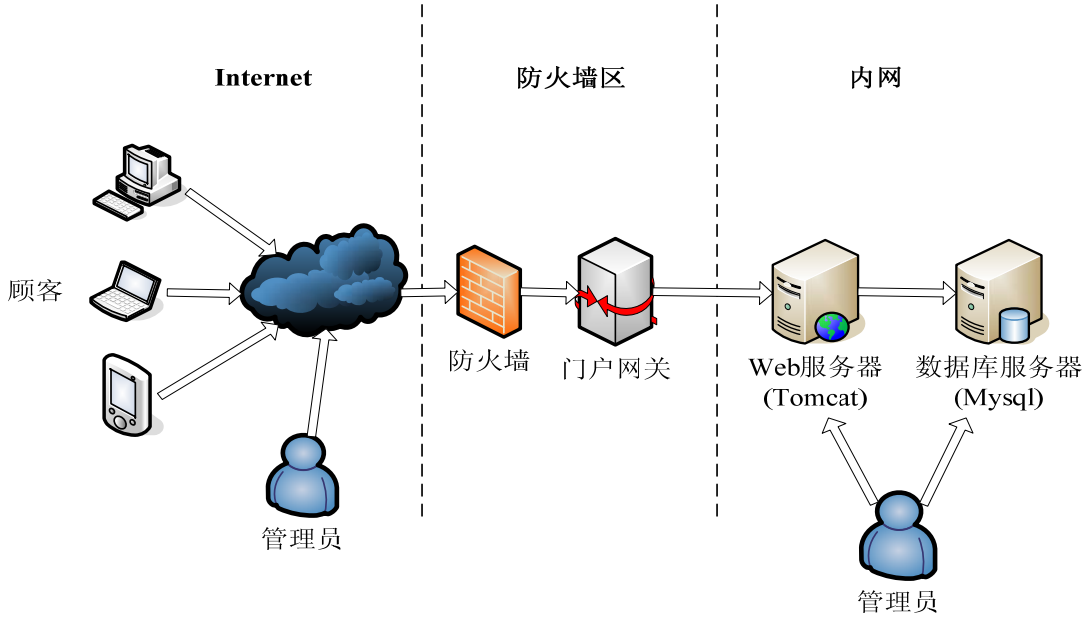

图 6.1 系统网络拓扑图

6.1.2 网络设备

（1）服务器端：

100M 以太网+网线连接 Internet

100M 宽带路由器

网关防火墙

（2）客户端：

1M 以上宽带网+网线连接 Internet

6.2 软硬件要求

6.2.1 硬件要求

（1）服务器端：

Web 服务器一台：CPU 类型：Xeon E5504；CPU 频率：2000MHz；内存：4G

数据库服务器一台：CPU 类型：Xeon E5504；CPU 频率：2000MHz；内存：4G

（2）客户端：

PC 机，CPU 频率 PIII 733 以上

6.2.2 软件要求

（1）服务器端：

Redhat Enterprise Linux/Windows Server 2003

JDK1.5 以上

Tomcat 5

MySQL 5

（2）客户端：

Windows XP/Windows Vista/Windows 7/Linux

Internet Explorer 6 以上/Opera/Firefox

6.3 关键技术实现

6.3.1 AOP 事件拦截机制

AOP 是一种先进的编程理念，可以用在日志，安全，事务处理方面，AOP 是对 OOP 的补充，是一种更通用和强大的编程思想，它可以使代码更加简洁，紧凑，减少重复代码。

本系统使用 JDK 动态代理机制来实现 AOP，JVM 动态生成代理类，代理类在运行期间生成一系列特定的接口，对某个类的方法调用，变为对隐藏的代理类的方法调用。AOP 的实现步骤如下。

（1）创建一个代理工厂

代理工厂是集中生成所有目标类的代理类的工厂。客户不需要知道具体的生成机制。

```
public interface DynamicProxyFactory{
    <T> T createProxy（Class<T> clazz,
        T target,
        AOPInterceptor interceptor）;
}
```

要实现一个动态代理类，需要目标类和一组接口，对接口有一些规定，这里使用一个简单的接口。

```
public <T> T createProxy（Class<T> clazz,
```

```
        T target, AOPInterceptor interceptor） {
        InvocationHandler handler =
            new DynamicProxyInvocationHandler（target,
                interceptor）;

        return （T）Proxy.newProxyInstance（
            Thread.currentThread（）.getContextClassLoader（）,
            new Class<?>[] {clazz},
            handler）;
    }
```

代理工厂的实现比较简单，首先实现一个 InvocationHandler 的实例，然后它调用 static method Proxy.newProxyInstance（），返回一个代理类，这个代理类实现了作为第二个参数传入的接口。注意第二个参数是 Class<?> 数组。

对生成的代理类的方法调用被转移到 InvocationHandler 的 invoke 方法。

```
    public Object invoke（Object proxy,
                    Method method,
                    Object[] args）
```

这个方法在 DynamicProxyInvocationHandler.java 中的实现如下，它使用了 Method 对象的反射机制。

```
    public class DynamicProxyInvocationHandler
            implements InvocationHandler {
        private Object target;
        private AOPInterceptor interceptor;

        public DynamicProxyInvocationHandler（Object target,
            AOPInterceptor interceptor） {
            this.target = target;
            this.interceptor = interceptor;
        }

        public Object invoke（Object proxy, Method method,
```

```
            Object[] args）  throws Throwable{
            try {
                interceptor.before（method, args）;
                Object returnValue = method.invoke（target, args）;
                interceptor.after（method, args）;
                return returnValue;
            } catch（Throwable t）  {
                interceptor.afterThrowing（method, args, t）;
                throw t;
            } finally {
                interceptor.afterFinally（method, args）;
            }
        }
    }
```

（2）添加 Interceptor

因为 invoke 方法代理了所有对我们的类的方法的调用，在它调用我们的类的方法之前或之后，添加相应的代码，就实现了所谓的 AOP 机制。我们定义一个 AOPInterceptor 接口：

```
public interface AOPInterceptor {
    void before（Method method, Object[] args）;
    void after（Method method, Object[] args）;
    void afterThrowing（Method method, Object[] args, Throwable throwable）;
    void afterFinally（Method method, Object[] args）;
}
```

InvocationHandler 的 invoke 方法，在调用我们的方法之前，调用 AOPInterceptor.before（）；之后调用 after（）；异常抛出时调用 afterThrowing（）；finally 处理之后调用 afterFinally（）。

（3）实现拦截

当然我们不希望对所用的方法都实现 Interceptor，只对我们需要的一部分类做 Intercept，本系统将这些类和方法定义在 annotations 中，在运行期间框架会根据 annotations 动态决定是否实现 Intercept。比如针对事务定义的 TransactionAnnotation：

```
@Retention（RetentionPolicy.RUNTIME）
@Target（ElementType.METHOD）
public @interface TransactionAnnotation {
    String value（）;
}
```

假设有一个服务，PersistenceService，在 save（） 中需要一个新的事务，在 load（）中不用事务，则 PersistenceService 中可以加入 TransactionAnnotation 如下：

```
public interface PersistenceService {
    @TransactionAnnotation（"REQUIRES_NEW"）
    void save（long id, String data）;

    @TransactionAnnotation（"NOT_SUPPORTED"）
    String load（long id）;
}
```

现在需要一个事务. 假设 transaction API 如下：

```
public interface Transaction{
    void open（）;
    void rollBack（）;
    void commit（）;
    void closeIfStillOpen（）;
}
```

这是一个用于事务处理的 Interceptor：

```
 public class TransactionInterceptor
        implements AOPInterceptor {
    private Transaction transaction;

    public void before（Method method, Object[] args） {
        if （isRequiresNew（method））{
            transaction = new TransactionAdapter（）;
```

```
            transaction.open（）；
        }
    }

    public void after（Method method, Object[] args） {
        if （transaction != null） {
            transaction.commit（）；
        }
    }

    public void afterThrowing（Method method,
        Object[] args, Throwable t） {
        if （transaction != null） {
            transaction.rollBack（）；
        }
    }

    public void afterFinally（Method method, Object[] args） {
        if （transaction != null） {
            transaction.closeIfStillOpen（）；
        transaction = null；
        }
    }   protected boolean isRequiresNew（Method method） {
        TransactionAnnotation transactionAnnotation =
            method.getAnnotation（TransactionAnnotation.class）；

        if （transactionAnnotation != null） {
            if （"REQUIRES_NEW".equals（
                transactionAnnotation.value（））） {
            return true；
          }
        }

        return false；
    }
```

```
}
```

当创建代理类时，就可以将 TransactionInterceptor intercept 到 PersistenceService 类中了：

```
DynamicProxyFactory proxyFactory = new DynamicProxyFactoryImpl（）；
AOPInterceptor interceptor = new TransactionInterceptor（）；
PersistenceService proxy =
      proxyFactory.createProxy（PersistenceService.class,
            new PersistenceServiceImpl（）,
            interceptor）；
proxy.save（1, "China"）；
String data = proxy.load（1）；
```

6.3.2 全站静态访问技术

本系统的页面静态化基于 URL 重写，如果希望哪个 URL 能生成静态文件，只需要在 conf/ urlrewrite.xml 配置一个相应的 MAPPING，例如顶层商品类别的地址的配置：

```
<rule>
<from>^/catalog/list-p-（[0-9]+）-（[0-9]+）\.html$</from>
<to>/controler/PageControler?pcid=$1&amp;amp；cid=$2&amp;amp；
groups=eshop_product|eshop_catalog|eshop_config&amp;amp；
page=com.eshop.app.servlet.page.catalog.ListP
</to>
</rule>
```

顾客点击某一顶层商品类别页面链接时，看到的是一个静态网页地址，比如 http：//www.eshop.com/catalog/lisp-p-0-10334.html。该地址通过<from>正则表达式实现 URL 匹配，传递到<to>的实际访问 URL 上。它的实际 URL 是 PageControlerServlet，并能提取参数，比如上面的<to>中有 pcid，cid，groups，page 四个参数。PageControlerServlet 会调用 HTML Cache 模块，检查该链接是否已经存在相应的静态页面，如果有，直接访问该静态页面，如果没有则生成静态页面。

有些页面数据经常变化，例如购物车，下单页面等，并不需要生成静态，可以增加 nocache=1 参数，实现纯 URL rewrite。这样同样能对提高网站安全性起很大作用。实现纯 URL rewrite，以上配置可改为：

<rule>

<from>^/catalog/list-p-（[0-9]+）-（[0-9]+）\.html$</from>

<to>/controler/PageControler?pcid=$1&amp；cid=$2&amp；nocache=1&amp；page=com.eshop.app.servlet.page.catalog.ListP

</to>

</rule>

6.4 实现效果

图 6.2　商城首页

图 6.3　下单成功页

图 6.4　后台管理首页

6.5 本章小结

本章首先介绍了系统实现的相关网络环境，包括网络拓扑图和相关网络设备，然后介绍了系统对软硬件的具体要求，最后针对关键技术实现部分进行阐述，包括 AOP 事件拦截机制和全站静态访问技术。

结　论

随着中国市场经济的日趋成熟，中国企业面对的竞争压力也越来越大，企业要想生存，就必须充分利用信息化手段来提高管理效率及市场响应速度。电子商务是在互联网开放的网络环境下，基于浏览器/服务器应用方式，实现消费者的网上购物、商户之间的网上交易和在线电子支付的一种新型的商业运营模式。电子商务作为一种独立的经济形态，已初具规模，一些电子商务网站的成立，给人们的生活带来了巨大的影响。

如何建立企业的电子商务，如何把企业业务建在 Internet 上，涉及到建立电子商务网站、开发符合 Internet 特点的有效的业务应用、管理网上的交易信息、保证网上数据安全、快速反映市场变化以及充分满足 Internet 业务进一步发展的要求等等。对一个运营商业企业来说，电子商务网站是其生存的理由和基础，同时也是企业对外展示信息、从事商务活动的窗口和界面。如何设计、建立一个经济、实用、安全、高效、稳定的网站是每个电子商务网站必须考虑的问题。而要解决好这些问题，就必须在提高企业内部管理效率、充分利用企业内部资源的基础上，从整体上降低成本，加快对市场的响应速度，提高服务质量，提高企业的竞争力。但是企业在利用信息化技术时，必须要考虑成本、技术难度、创造的价值等几个方面。越是强大的电子商务系统，那么实现它的技术难度也越大。

本论文首先在分析基于 Java EE 平台技术的基础上，借鉴已有的系统设计模式，并采用 UML 建模方法，对网络商城进行了具体的需求建模，最后构建了基于 Java EE 架构的分布式系统。该系统满足了企业不断变化的应用需求，它实现了跨平台性、分布性、安全性、事务性、可扩展的服务器端组件模型，具有封装程度高、开发运营成本低、运行高效等特点，为企业产品的销售带来极大的便利，提高了企业效率，节约了企业运作成本。

由于水平和时间的限制，以及 Java EE 技术规范还处于在不断发展变化之中，所以本论文作者所开展的研究工作还有许多可以进一步深入和完善之处。例如：研究并增加一些给用户带来更好体验的辅助功能，因为一些细小但重要的辅助功能往往带给用户深刻的感受，例如增加短信或邮件通知功能。另外，需要对该体系结构是否能承受超高并发访问量进行进一步的研究和测试。

参考文献

[1] 豆丁网. 阿里查查电子商城系统的设计[OL]. http://www.docin.com/p-33493349.html

[2] 程祥. 基于 J2EE 架构的销售管理信息系统设计与实现: [硕士论文]. 成都: 成都电子科技大学, 2007, 2-9

[3] 林信良(良葛格)的专栏. AOP 观念与术语[OL]. http://blog.csdn.net/caterpillar_here/archive/2006/05/24/753473.aspx, 2006-05-24

[4] Od Johnson, Juergen Hoeller. Expert One on one J2EE Development Without EJB. Wrox, 2004

[5] 开源中国社区. oschina.net. Java 应用服务器 Tomcat[OL]. http://www.oschina.net/p/tomcat

[6] 中文 Java 技术网. cn-java.com. MySQL 数据库的介绍与历史[OL]. http://www.cn-java.com/www1/?action-viewnews-itemid-56112

[7] 鲁博, 柴跃廷. 关于统一建模语言--UML[JL]. 计算机工程与科学, 2000, (4)

[8] 潘冲. 基于 Struts + Hibernate + Ajax 的网络电子商城系统: [硕士学位论文]. 浙江工业大学, 2008, 11-15

[9] Turboshop 架构白皮书[OL]. http://www.turboshop.cn/intl/zh-cn/turboshop_architecture_whitebook.pdf.html, 2008-07-16

[10] Java 学习室. 如何使用 velocity 模板引擎开发网站[OL]. http://www.java3z.com/cwbwebhome/article/article2a/240.html?id=222

[11] 周平. Lucene 全文检索引擎技术及应用[JL]. 重庆工学院学报(自然科学版), 2007, (4): 28-29

[12] 刘力宏. 基于 J2EE 的分布式销售管理系统的设计与实现: [硕士学位论文]. 中国农业机械化科学研究院, 2007

[13] 赵建娇, 王雪光. 基于 J2EE 虚拟商场的设计与分析. 商场现代化, 2008

[14] 赛奎春主编. JSP 工程应用与项目实践. 北京：机械工业出版社， 2005：25-33

[15] 黄明， 梁旭编著. JSP 信息系统设计与开发实例. 北京：机械工业出版社，2004：80

[16] 鲁晓东， 李育龙， 杨健 编著. JSP 软件工程案例精解. 北京：电子工业出版社， 2004.12：78-112

[17] 孙佳， 刘中兵， 李伯华编著. JSP+Oracle 动态网站开发案例精选. 北京：清华大学出版社， 2005：212-243

[18] 石志国，薛为民，董洁编著. JSP 应用教程. 北京：清华大学出版社， 2004：56-62

[19] 王国辉， 王易编著. JSP 数据库系统开发案例精选. 北京：人民邮电出版社，2006.5：72-75

[20] 赛奎春主编. JSP 信息系统开发案例精选. 北京：机械工业出版社， 2006.1：93-102

[21] [美 Wright Michael Freed]Brian man. JSP 在数据库中的应用与开发（英文版）. 北京：北京希望电子出版社， 2001.1：73-76

[22] Jeffrey A. Hoffer，University of Dayton，Joey F. George. Modern Systems Analysis and Design， Third Edition. Pearson Education， 2003：66-67

[23] 林上杰，林康司编著. JSP 2.0 技术手册. 北京：电子工业出版社， 2004.5：22-45

[24] 阎毓杰. JSP 数据库编程入门. 长春：吉林电子出版社， 2004：55-58

[25] 光军，胡波著. JSP 应用开发实例详解. 北京：航空航天大学出版社，2006.5：215-217

[26] 姜晓铭等著. JSP 程序设计精彩实例. 北京：清华大学出版社， 2003.5：43-65

[27] 天宏工作室译著. JSP 程序设计指南. 北京：清华大学出版社，2002.5：98-150

[28] Vivek Chopra， Jon Eaves， Rupert Jones 等著. JSP 程序设计（英文版）. 北京：人民邮电出版社， 2006.9：20-30

[29] 黄进华. J2EE 平台基于 Web Services 的企业应用集成技术研究[D]: [硕士学位论文]. 青岛大学, 2007.

[30] 万军民. 基于 Java 的代码生成器的设计与实现[J]. 计算机工程, 2004, 12(30): 122-124.

[31] 刘鹰. 代码生成技术及其在企业开发中的应用[J]. 西安文理学院学报, 2006, (9): 97-100.

[32] 金雪云, 徐卫东, 刘红卫. 基于 NET 的代码生成工具的设计与实现[J]. 计算机工程与应用, 2004, (21): 123-130.

[33] 余浩东. J2EE 应用框架设计与项目开发[M]. 北京: 清华大学出版社, 2008, (1): 1-18.

[34] [美]阿卢, 刘天北, 熊节译. J2EE 核心模式[M]. 北京: 机械工业出版社, 2005, (2): 3-12.

[35] 戴林, 陈菊明译. J2EE 平台高级开发(应用集成层模式). 北京: 清华大学出版社, 2007, (1): 1-5.

[36] 盛刚, 韩莉莉. J2EE 代码自动生成研究[J]. 计算机系统应用, 2006, (8): 31-33.

[37] 孙茂增, 李凤华, 都婧. 基于 Velocity 的 J2EE 应用代码生成系统[J]. 仪器仪

表用户, 2008, 105-106.
[38] [美]盖尔兹, 陈晓燕译. 实用 J2EE 应用程序体系结构[M]. 北京: 清华大学出版社, 2007, (1): 5-26.
[39] [美]贝姆. 软件工程经济学[M]. 李师贤等译. 北京: 机械工业出版社, 2004, (1): 33-40.
[40] [美]勃姆. 软件成本估算: COCOMO II 模型方法[M]. 北京: 机械工业出版社, 2005, (1): 22-42.

致　谢

在此硕士论文完成之际，特别想在此向我在研究生的学习期间，一直以来帮助过我的指导老师、相关老师、院系领导、同窗同学和至亲的亲人们致以衷心感谢！

首先在此我要感谢我的硕士研究生导师鲁佑文老师!在硕士研究生学习期间、科学研究和日常生活中，导师给予了我无数耐心的教导和细致入微的关怀。无论是在本文的课题选择、难题解决、系统设计、系统实现、系统测试的所有过程中，导师都依据他丰富的系统实践经验帮我严格把关，并通过一次一次的耐心讲解，一次一次与我的深入讨论和交流，才能使得本课题能够顺利完成。导师的学术造诣和治学严谨的为人处事作风、良好的师德风范以及对教育事业的不断追求都给我留下深刻印象。

在此，感谢所有帮助过我的同仁。

刘　　丹

2012 年 4 月 26 日