

东南大学
硕士学位论文
基于XML和JSP的移动电子商务技术及实现
姓名：王劲松
申请学位级别：硕士
专业：计算机应用
指导教师：徐宏炳;王健
20040304

## 摘 要

因为JSP有强大的Java技术做支持, 所以有相当的优势。对于开发大型的电子商务和移动电子商务站点J2EE及J2ME技术是首选。

基于 JSP 的 MVC (Model、View、Controller) 是 Web 应用开发中的一项重要技术, 可使系统结构清晰, 易于管理和扩展。

运用 J2ME 体系, 移动设备运行 MIDlet 访问 Internet 网站构成了移动用户端界面视图, 整个系统包含了 IE 浏览器和手机浏览器两极用户。

用 XML 标记来部署 JSP 三层模型, 能实现代码重用和大粒度的构件, 能统一数据格式。

针对具体情况, 文章以一个花卉订购站点为实例, 运用 MVC 进行合理布局, 设计并实现了一个能同时向手机用户和 PC 机用户提供服务的花卉销售站点。并从四个方面实际地论述了 XML 在 JSP 中的应用。

## 关 键 词

E-business (电子商务); JSP (Java Server Pages); MVC (Model、View、Controller); J2ME (Java2 Micro Edition); XML (可扩展标识语言); JDBC;

## **Abstract**

By reason that strong technology of Java to support JSP, so it has goodish advantage. To create a large-scale site of E-business and Moving E-business, The best choice is using J2EE and J2ME.

MVC base on JSP is an important technology of applying and developing on Web. It gets clear structure so that easy to manage and to expand for system.

To handle architecture with J2ME, Running program of MIDlet on moving device visiting Internet site that form interface in side by Moving client. In this way, overall system contain two kinds of client for IE browse and handset browse.

Using XML mark to plan three-layer model of JSP, enable reuse code and realize big size component, enable unify format of data.

This paper according to concrete situation, For instance a Web site of flower ordering, Proceed reasonable layout with MVC. Finally, Design and realize a flower web site, which provide service for user on handset and PC. At the same time, actual discuss application XML in JSP by four ways.

## **Key Words**

E-business; JSP; MVC; J2ME; XML; JDBC;

## 东南大学学位论文独创性声明

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。尽我所知，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，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：\_\_\_\_\_日 期：\_\_\_\_\_

## 东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

研究生签名：\_\_\_\_\_导师签名：\_\_\_\_\_日 期：\_\_\_\_\_

随着 Internet 的发展,电子商务和移动电子商务迅速发展起来。信息管理,安全验证等已成为热门话题。然而,它们的最终实现还得依靠 Web 网络编程。J2EE 和 XML 是主要技术。

## 第一章 引言

当今,网络、通信和信息技术快速发展,Internet 在全球迅速普及,使得现代商业具有不断增长的供货能力、不断增长的客户需求和不断增长的全球竞争三大特征,任何一个商业组织都必须改变自己的组织结构和运行方式来适应这种全球性的发展和变化。随着信息技术在国际贸易和商业领域的广泛应用,利用计算机技术、网络通信技术和 Internet 实现商务活动的国际化、信息化和无纸化,已成为各国商务发展的一大趋势。电子商务正是为了适应这种以全球为市场的变化而出现和发展起来的。作为近年来社会关注的焦点,电子商务从广义上讲,是指人们通过计算机网络进行的各种商务活动。它可以使企业与供应商更紧密地联系,更快地满足客户的需求,也可以让企业在全世界范围内选择最佳供应商,在全世界市场上销售产品。

电子商务之所以受到重视,是因为它具有区别于其他方式的特点,具有诱人的发展前景。它可以使企业从事在物理环境中所不能从事的业务,有助于降低企业的成本,提高企业的竞争力,尤其能使中小企业以更低的成本进入国际市场参与竞争。同时,它能为广大消费者增加更多的消费选择,使消费者得到更多的利益。

电子商务也是一场革命,它打破了时空的局限,改变了贸易形态,使 Internet 成为一种重要的业务传送载体。它汇聚信息,生成新的业务,产生新的收入,它能使企业进行相互连锁交易。其自适应导航技术,便于用户通过网上搜索交换信息,使业务交往个人化,具有动态特征,因而受到用户欢迎,更具成本效益。但是,由于电子商务的发展处于初级阶段,理论上尚需进一步探索,而真正在社会各行业推广应用电子商务也并非一蹴而就。

因此,实现电子商务的各项技术正经历着一场从产生、发展到成熟的变革,

其中服务器端主要技术有：CGI、PHP、ASP 到 JSP，XML、XSLT，SQL 等，客户端主要技术有：HTML、Java Applet、VB Script、Java Script、Java MIDlet 等。

电子商务的发展依赖于从服务器端到客户端各项技术的实现，其中必然贯穿了信息管理、安全验证这样一些热门话题。

## 1. 1 电子商务

电子商务是指利用 Internet、Intranet、Extranet 来解决商业交易问题，降低产供销成本，开拓新的市场，创造新的商机，通过采用新网络技术手段，从而增加企业利润的所有商业活动。简单地讲，电子商务是指两方或多方通过计算机和某种形式的计算机网络（直接连接的网络或 Internet 等等）进行商务活动的过程。

电子商务这个概念起源于七十年代。当时一些大公司通过建立自己的计算机网络实现各个机构之间、商业伙伴之间的信息共享，这个过程被称为 EDI（电子数据交换）。EDI 通过传递标准的数据流可以避免人为的失误、降低成本、提高效率，据估计在世界 1000 个最大的企业中，95%以上的在使用这一技术。它过去是、现在也是电子商务的基础。

今天，Internet 为电子商务带来了飞速的增长。例如 EDI 技术已经摆脱了以前旧式的昂贵的公司独立网络，而融于 Internet；而更多的企业和企业之间的商务活动则干脆直接采用 WEB 技术来进行。不过现在您天天从收音机、电视、报纸和网络上听到看到的电子商务概念实际上指“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”起源于 1995 年，它的先驱是那些不进行传统零售业的 Internet 公司，如 AMAZON，但今天象 WAL-MART 这样的超市也建立了自己的网上商店。

在发达国家，您只要拥有一个计算机、浏览器、Internet 连接和信用卡，今天就可以从网络上购买到书本、CD、鲜花、飞机票、电视甚至到汽车。而明天您能购买到什么就没有人能知道了，因为宇宙是无限的。

在美国，大多数企业已经建立自己的网站介绍和销售他们的产品。而更重要的是网上商店的销售额在剧烈膨胀，1996 年是 6 亿美元，而到 1998 年则达到 24 亿。1999 年参与网上购物的家庭已经超过 700 万，几乎是 1998 年的两倍。热点

包括计算机软硬件、书本、音乐、鲜花等等。

几乎所有的专家都预测在未来的几年内，电子商务会飞速发展。eMarket 认为公司和公司之间的电子商务贸易额将从 1997 年的 56 亿美元发展到 2002 年的 2680 亿；个人网上购物将从 1997 年的 18 亿发展到 2002 年的 260 亿。在发达国家，电子商务技术和电子商务市场已经发育成熟。

电子商务通过基于因特网事务自动化提高了企业业务处理效率和可靠性。目前主要有两种类型电子商务：企业对客户型（Business-to Consumer 简称 B2C）、企业对企业型（Business-to Business B2B）。B2C 参与的对象是企业和单个的用户，其实现相对容易；但是对 B2B，情况就不一样，B2B 涉及到多个不同规模企业的参与。

电子商务提供了一种崭新的商业贸易的途径，是未来推动经济增长的关键动力，它促进了全世界经济的发展。只有鼓励电子商务的所有参与者（包括政府、消费者、工商界、公众团体等）之间的合作和社会对话，才能推动电子商务的发展。

进入九十年代以来，随着计算机网络、通讯技术的迅速发展，特别是 Internet 的普及应用，使人们传统的行为方式和观念受到巨大的冲击和影响。电子商务是 Internet 发展的最新阶段，它代表着下个世纪网络应用的发展方向。可以预言，电子商务作为一种崭新的商务运作方式，将带来一次新的产业革命，这场革命的最终结果是将人类真正带入信息社会。

Internet 是电子商务发展的基石，可以说，没有 Internet，电子商务就无从谈起。随着技术进步和计算机技术的不断普及，越来越多的人认识到网络时代已经到来，电子商务将为我们的生活方式带来重大变革。在电子商务时代来临之际，每个人首先需要了解 Internet，知道它能够为自己做什么，如何进入网络王国……由此，打开通向 Internet 的大门，迎接个人的网上生活。电子商务是信息社会发展到一定阶段的产物。网络时代的电子商务将人类带入下一个世纪，全球经济将因此而受益，而人们的生活品质也将从中得到大幅度提高。

随着 Internet 应用的日益广泛，越来越多的人开始进入电子商务领域，世界上著名的厂商也纷纷抢占电子商务市场，对信息技术的商业应用提出了各自的解决方案，而解决方案也因服务客户的不同而异。多年来，很多商业机构一直在提



高自动化水平。企业的网络化经营是现代企业的明显特征，而在电子商务环境下具体展开业务活动则是企业适应市场经济的集中表现。

从早期的电子收款机（ECR, Electronic Cash Register）到电子数据交换（EDI, Electronic Data Interchange）技术的应用，直至今天 Internet 上的电子商务，商业自动化程度越来越高，技术手段更为先进和复杂。可以说，电子商务是商业自动化发展的高级阶段。

## 1. 2 移动电子商务

电子商务方兴未艾，移动电子商务又走进了人们的视线。不少 IT 业界人士也开始关注移动电子商务的发展之路。移动电子商务带给企业的效益是显而易见的。通过个人移动设备来进行可靠的电子交易的能力被视为移动因特网业务的最重要方面。移动通信提供了高度的安全性，而且其安全性还可通过各种方式得到进一步增强，如电子签名、认证和数据完整性。因特网与移动技术的结合为服务提供商创造了新的机会，使之能够根据客户的位置和个性提供服务，从而建立和加强其客户关系。有了移动电子商务技术，就可以实现流动办公，大大节约成本，提升企业或个人的营运作业效率。

从移动电子商务的特点来看，移动电子商务非常适合大众化的应用。移动电子商务具有灵活、简单、方便的特点。移动电子商务不仅仅能提供在因特网上的直接购物，还是一种全新的销售与促销渠道。它全面支持移动因特网业务，可实现电信、信息、媒体和娱乐服务的电子支付。不仅如此，移动电子商务不同于目前的销售方式，它能完全根据消费者的个性化需求和喜好定制，用户随时随地都可使用这些服务。设备的选择以及提供服务与信息的方式完全由用户自己控制。

通过移动电子商务，用户可随时随地获取所需的服务、应用、信息和娱乐。他们可以在自己方便的时候，使用移动电话或 PDA 查找、选择及购买商品和服务。服务付费可通过多种方式进行，以满足不同需求，可直接转入银行、用户电话账单或者实时在专用预付帐户上借记。

Wap 是开展移动电子商务的核心技术之一。通过 wap，手机可以随时随地、方便快捷地接入互联网，真正实现不受时间和地域约束的移动电子商务。Wap



是一种通信协议，它的提出和发展是基于在移动中接入 internet 的需要。Wap 提供了一套开放、统一的技术平台，用户使用移动设备很容易访问和获取以统一的内容格式表示的 internet 或企业内部网信息和各种服务。它定义了一套软硬件的接口，可以使人们像使用 pc 机一样使用移动电话收发电子邮件以及浏览 internet。同时，wap 提供了一种应用开发和运行环境，能够支持当前最流行的嵌入式操作系统。

由于便携式电脑、手机、PDA 等各种各样的移动终端的大量出现，为移动电子商务的发展打下了良好的基础。特别是我国使用手机的用户规模正在扩大。手机用户已达世界第一的规模，并且还在快速增长。据信息产业部数据，到 5 月末为止国内手机用户数达到 1.7 亿人，每月平均有 530 万人加入到手机用户中来。

以手机来做电子商务，还出现了一个专门名词，就叫“移动电子商务”，或者就叫“移动商务”(m-commerce)，以显示和互联网上通过 PC 进行的“电子商务”(e-commerce)的不同。

因特网、移动通信技术和其它技术的完美结合创造了移动电子商务，移动电子商务以其灵活、简单、方便的特点将受到消费者的欢迎。

### 1.3 服务器端主要技术

随着 Internet 的飞速发展，电子商务领域各种新技术不断涌现并逐渐走向成熟。这些技术中最主要的两大系列是 XML、J2EE。XML 统一数据格式，大大改善了电子数据交换的方式。J2EE 在分布式逻辑控制上，在电子商务系统的开发、管理、维护和扩展方面优势突出。它们的组合形成优势互补，将是开发电子商务平台之首选技术。

#### 一、XML

XML 即扩展标识语言 (eXtensible Markup Language) 是 (w3c) 创建的一组规范。与 HTML 不同，XML 用来描述结构化数据而 HTML 用来显示内容。XML 包括以下几个方面内容：DTD，XSL，XLL 等。DTD 规定了 XML 文件的逻辑结构，XSL 是用于规定 XML 文档样式的语言，XLL 将扩展目前 Web 已有的简单链接。

HTML 主要是用来数据的显示，它不利于数据的格式化，结构化。而 XML 只关心数据本身，是纯数据文件不包含数据的表现形式，客户端只要指定不同的要求，系统就可以根据用户传入参数的不同找到对应的 XSL 文件，然后结合同一个 XML 数据文件，即可生成满足客户需求的 HTML 显示文件。该 HTML 文件对数据的选择和显示方式都是以客户的配置为根据，动态地进行表现。这种数据内容和表现形式的分离，在代码级实现了个性化服务。

XML 所采用的标准技术最适合 Web 开发，应用于 Internet EDI，则可以得到真正 Web 风格的 EDI——XML / EDI。XML 支持结构化的数据，可以更详细地定义某个数据对象的数据结构。而且 XML / EDI 引进了模板概念，解决了 EDI 存在的主要问题——映射问题。模板描述不是消息数据，而是消息的结构以及如何解释消息，能做到无须编程就可实现消息的映射。

传统的 EDI 是通过使用 SMTP 和 FTP 来进行数据格式转换的，而 XML 具有一套统一的数据格式，它使数据管理和交换的成本更低。通过应用 SSL 等现代的加密技术，在以 XML 描述信息中方便地加上数字签名，可对一段甚至整个 XML 文档进行加密，使基于 Internet 和 XML 的 EDI 系统达到传统 EDI 同样的保密性和安全性。

XML 为 Web 数据带来了结构化、智能化和互操作性，将引发 Web 查询技术，Web 数据库技术以及 Web 数据交换技术的全面革新，B-to-B，和 B-to-C 模式的电子商务的数据将更加容易交换。XML 非常适于应用程序之间数据交换的格式，特别是松耦合的应用程序，如：分布式 Web 系统。作为一种通信协议，HTTP 具有跨平台性，对于应用程序数据来说，XML 具有同等的协议。XML 可以促进应用程序代码的重用，提高应用程序在面对需求程序变化时的适应能力。

XML 提供在应用程序和系统之间传输结构化数据的方法，可用来在 Web 服务器、浏览器之间，贸易伙伴之间传输数据。因此 XML 非常适合于电子商务。

在众多编程语言中，Java 是支持 XML 的优秀平台，而 XML 又是 Java 应用的优秀数据表示方法。XML 和 Java 都与 Internet 关系密切。XML 被设计成一个优化的、灵活的可读格式，可直接用于 Internet；而 Java 从一开始就支持 socket、HTTP、HTML 和服务端。它们都支持 Unicode 字符集，因而很容易实现本地化应用和个性化服务。正如 Java 向程序员提供了表达复杂数据和建立面向对象模型的能力一样，用 XML 表达复杂的层次化数据模型是很理想的。

## 二、J2EE

J2EE 包含许多技术,在构建电子商务网的应用中,主要用到的 J2EE 技术有 JSP, Servlets, EJB 和 JDBC 等。

JSP 是一个特别的 Java 语言。JSP 加入了一个特殊的引擎,它是一个 JSP 到 Java Servlet 的生成器。JSP 具有诸多优点,如一处编写随处运行,系统的多平台支持,强大的可伸缩性,多样化和功能强大的开发工具支持等。

Servlets 提供的功能大多与 JSP 类似,而实现的方式不同。JSP 通常是大多数 HTML 代码中嵌入少量的 Java 代码,而 Servlets 全部由 Java 写成并且生成 HTML。

EJB 是一种允许开发者快速建立企业应用的组件结构。使用 EJB,开发者可集中自己的精力去写真实世界中的现实问题的应用,而不是复杂的分布式服务器端系统,当编写管理特定业务功能的 J2EE 应用程序时,开发者可将完成这些任务的业务逻辑放置在 EJB 层的 Enterprise Beans 中。以使代码集中在解决手边的业务问题,而利用 Enterprise Bean 容器来支持低层服务,比如状态管理、事务管理、线程管理、远程数据访问和安全等。

JDBC API 以一种统一的方式来对各种各样的数据库进行存取,并提供了数据库存取的平台独立性。其中 JDBC-network Bridge。JDBC 网络桥驱动程序不再需要客户端数据库驱动程序,它使用网络上的中间服务器来存取数据库。这种应用使得负载均衡、连接缓冲池和数据缓存等技术的实现成为可能。由于只需要相对少的下载时间和平台独立性,而且不需要在客户端安装并取得控制权,很适合于 Internet 上的应用。

典型的 J2EE 三层模型,包括表示层、商业层、数据层。

表示层:表示层中的组件主要用来处理用户接口与用户交互。在 J2EE 中,表示层包括 CORBA 客户端,Java Applet,Java Application,Java Servlet,Java Server Page (JSP),静态 Web 网页。

商业层:商业层中的组件要协同工作,来解决诸如结帐、处理订单等商业逻辑。商业层也叫业务层或 EJB (Enterprise Java Bean) 层,作为解决或满足某个特定业务领域(比如银行、零售或金融业)的需求的逻辑的业务代码由运行在商业层的 Enterprise Beans 来执行。一个 Enterprise Beans 从客户程序处接收数据,对数据进行处理,再将数据进行发送到数据层存储。一个 Enterprise Beans 还从

存储中检索数据，并将数据送回客户程序。

数据层：数据层运行企业信息系统软件，这层包括企业基础设施系统，如企业资源计划（ERP）、大型机事务处理（Mainframe Transaction Processing）、数据库系统及其他遗留信息系统（Legacy Information Systems）。

利用 JSP / Servlet 的三层解决方案，实现了表示层和逻辑层的分离，使得系统具有良好的扩展性，由于 Java 的平台独立性，使得系统可在不同的操作系统不同的硬件环境下运行，而 XML 的数据格式与表现分离、传输标准化，便于个性化管理和系统的扩展、维护。

总的来说，一个基于Web的应用程序使用HTML来显示数据；用XML来定义数据以使其可被另一个程序读取并处理；使用JSP页面或Servlet来管理用户与业务层或存储层之间的数据流。

## 1.4 客户端相关技术

### 一、PC 浏览器用户

客户端采用瘦客户方式，只需安装 Web Browsers(网页浏览器)，通过 HTML 页面中嵌入 Applet 小应用程序来进行简单的逻辑控制。请求以 URL 地址定位，发送到应用程序服务器端的 JSP 页面，由 JSP 页面接收用户信息，并根据用户的不同、请求的不同 JSP 将生成不同的个性化页面来返回给用户。完成对客户端的显示控制。

HTML、Java Applet 和 Java Script 是主要技术，基于 Web 的产品开发过程管理系统中，在客户端静态显示方面，采用 HTML 技术。在客户端动态显示方面，采用 Java Applet 和 Java Script 技术。

JavaScript 是一种基于对象的编程语言。JavaScript 作为一种脚本语言，简单、易用。多用于用户端浏览器动态效果的显示，使页面更加生动、有吸引力。也常用于用户端的初级密码验证，从而提高安全性，达到减轻服务器工作量的目的。JavaScript 的特点：可把源码用普通文本形式写进 HTML 网页里。编写的 JavaScript 程序会在浏览器载入网页的过程中被执行。JavaScript 源码不能加密，它能被任何人看到或复制。JavaScript 不受特定的操作系统或浏览器的限制。有着很好的跨平台性。如图（1-1）是采用 JavaScript 在用户端进行密码验证的一

个界面。



图 (1-1)

Java 语言以其简单、独立于平台、面向对象、分布式、健壮、结构中立、可移植性、解释性、高效率、多线程等特点,成为一种在网络上广泛应用的网络编程语言. Java 语言程序的运行环境分为两类,一类是在普通处理器上实现的 Java 虚拟机 (JavaVirtualMachine, 简称 JVM), 另一类是专用于运行 Java 语言程序的处理器. JVM 是由软件实现的 Java 语言程序执行环境, Java 语言程序采用字节码作为中间代码, 程序代码短小精悍, Java 小程序 (applet) 与 Web 页完美结合, 为 Web 页提供多媒体和交互功能. 如图 (1-2) 是一个使用 JavaApplet 效果的页面。



图 (1-2)

## 二、手机浏览器用户

在信息社会中手机及其它无线通信设备越来越多地走进普通老百姓的工作和生活。随着信息网络化的不断进展,手机及其它无线设备上网的趋势越来越明显。但是传统手机存在以下弊端:第一,传统手机中的软件均是出厂时由硬件厂商固化在其硬件设备中,要更新手机功能必须更换一部手机;第二,传统手机访问互联网是通过 WAP (Wireless Application Protocol),所有网络资源必须在网络接通时才可使用,不仅耗时,费用亦很高。国内外手机软件的最新动态表明,Java 手机将是未来手机的发展方向,是业界的热点。

Java 手机软件的真实运行环境是 Java 手机,只有在支持 Java 的手机上软件才能正常运行。一般来说,Java 手机软件的开发过程先是在台式机上模拟,模拟成功后再将其硬化到手机中。所用的模拟开发环境是:系统环境:Windows 2000 Server;平台支持:J2ME CLDC、J2ME MIDP,

Java 手机的好处之一是无需实时在线,通过无线互联网将程序下载到电话内便可以离线运行。这样,Java 手机可以为用户减少很多在线时间,充分节约了费用。对运营商而言,也能避免由于过多用户同时在线而导致的网络过载,减轻了网络的负担。于是,对 Wap 产生致命影响的速度不会成为制约 Java 手机发展的障碍。Java 手机的另一个好处就在于它的交互性。现阶段手机上的文字式静态内容将被 J2ME 技术提供的交互式服务所取代。同时,由于标准的开放性,使应用程序只需开发一次就可以在任何无线设备上使用,内容的开发将更为方便。

手机端的主要技术有:WAP, J2ME。

WAP (Wireless Application Protocol) 即无线应用协议,它是一组向移动终端提供互联网应用和服务的开放式协议,是一个用于解决手机上网浏览问题,并适用于不同无线网络技术的全球性无线网络规范。

J2ME (Java 2 袖珍版), J2ME 提供了 HTTP 高级 Internet 协议,使移动电话能以 Client-server 方式直接访问 Internet 的全部信息,不同的 Client 访问不同的文件,此外还能访问本地存储区,提供最高效率的在线交易。



总之，从服务器到客户端整个体系都是基于 Java 的，由于 Java 是一种简单易用、完全面向对象、具有平台无关性且安全可靠的主要面向 Internet 的开发工具。自从 1995 年正式问世以来，Java 的快速发展已经让整个 Web 世界发生了翻天覆地的变化。随着 Java Servlet 的推出，Java 在电子商务方面开始崭露头角，最新的 Java Server Page 技术的推出，更是让 Java 成为基于 Web 的应用程序的首选开发工具。

用 JSP 构建电子商务网络服务器。采用 MVC 三层模型。M (Model) 模型: 封装对象及其属性和方法, V (View) 视图: 完成显示逻辑和界面接口, C (Controller) 控制器: 体现商务逻辑、实现系统控制。数据库连接池 为三层模型提供了稳定的后台基础。

## 第二章 电子商务 JSP 三层模型

目前电子商务网站服务器的实现主要有 ASP 和 JSP 技术。

ASP(Active Server Pages)是 Microsoft 开发的运行在 IIS(Internet Information Server)下的一个服务器端的(Server side)脚本执行环境,用户可用它产生和执行动态的、交互的、高性能的 Web 服务器应用程序。ASP 只在服务器端运行,并将执行的结果以 HTML(Hypertext Markup Language)文件形式传给 Web 浏览器,对客户端要求低,属于胖服务器瘦客户机的运行模式。

JSP(Java Sever Pages)是 Sun 公司 1996 年 6 月推出的新技术,是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术,是 Servlet2.1API 的扩展。2000 年 9 月 Sun 公司发布了 JSP1.2 规范和 Java ServletAPI2.3。JSP 是基于 Java 开发 Web 应用的集成部分,它通过使用脚本文件和面向对象的组件,组合成标准标示文本,把表示从实现逻辑中分离,利用这一技术可以建立先进、安全和跨平台的动态网站。

(1) ASP 只能运行在 Windows 平台下, JSP 和 Java 是跨平台的。

(2) ASP 以 Java script 或 VB script 作为其脚本语言,而 JSP 中使用 Java 代码, JSP 的执行速度比 ASP 的速度快。

(3) ASP 与 COM/DCOM 组件接合,而 JSP 与 JavaBean 整合。开发 COM 组件要比开发 Java Bean 组件难度大:其次,使用时 COM 组件需要在服务器上注册后才能使用,这样如果修改一个组件,就必须注册并重新启动服务器才能使用,而 Java Bean 在修改后不要重新注册。

(4) ASP 使用 ODBC 通过 ADO(ActiveX Data Object, ActiveX 数据对象)连接数据库。JSP 使用 JDBC 连接数据库, JDBC 除了具有 ODBC 的特点外,更具有对

硬件平台、操作系统异构性的支持的特点。

JSP是一种基于Java Servlet的Web开发技术,在JSP下,嵌入HTML页面的程序代码是Java代码;页面中嵌入的程序代码被编译成Servlet(这种编译操作仅在对JSP页面的第一次请求时发生)并由Java虚拟机执行。在实际的JSP开发过程中,和传统的ASP或PHP页面相比,JSP页面将会是非常简洁的,由于JavaBean开发起来简单,又可以利用Java语言的强大功能,许多动态页面处理过程实际上被封装到了JavaBeans中。

这里Java代码可以通过JDBC访问多个异构的数据库,其平台无关性特别好。JDBC是用于执行SQL语句的Java应用程序接口,由一组用Java语言编写的类与接口组成,在JSP中将使用JDBC来访问数据库。JDBC是一种规范,它让各数据库厂商为Java程序员提供标准的数据库访问类和接口,这样就使得独立于DBMS的Java应用程序的开发工具和产品成为可能。一般的Java开发工具都带有JDBC—ODBC桥驱动程序,这样,只要是能够使用ODBC访问的数据库系统,也就能够使用JDBC访问了。当前,Internet上的数据库应用已越来越多,可见JSP比ASP更适合开发Web应用程序。

ASP和JSP对组件技术的支持已经很完善了,而PHP直到前不久才开始支持COM和JavaBean。但支持也不是很完善,如果PHP不能在将来完善对组件技术的支持,在大型Web应用程序方面将很难与JSP和ASP竞争。但由于PHP技术本身的易学易用,加上众多的函数支持和开放源代码的特性,在中小型Web站点的开发上,PHP还是会占有一席之地的。其实,JSP本身对于ASP和PHP并没有明显的优势,JSP的强大是因为其后面有强大的Java技术做支持。包括JavaBean和J2EE技术在内的Java技术是JSP强大生命力的所在。Microsoft最新推出的ASP.NET技术和ASP技术相比有了许多激动人心的进步,但是从企业级应用的角度看,JSP技术仍然有相当的优势。有理由认为,在将来的Web开发中,中小型站点将出现JSP、ASP.NET和PHP三分天下的局面,但是对于大型的电子商务站点,JSP及J2EE技术将成为首选。

在此将采用JSP来开发一个简单花卉网站Web应用程序。JSP担负动态页面生成的任务,Servlet担负起决定整个网站逻辑流程的任务,将常用的数据库连接及数据模型写入Java Beans中。

## 2.1 MVC（模型、视图、控制器）模式

### 一、MVC 三层结构

在基于组件的J2EE 平台充分内置了灵活性的情况下，剩下的问题可能是如何组织应用程序以实现简单高效的程序升级和维护，以及如何让不懂程序代码的人员避开程序数据。答案就在模型、视图和控制器架构（ MVC ）的使用之中。MVC 这样的架构是一个描述重现的问题及其解决方案的设计范式，但每次问题重现时，解决方案都不会完全相同。

随着动态网页技术的发展，JSP 与其多方面优势正逐渐成为 Internet 网上的主流开发工具。JSP 文件能动态地生成 HTML 文件和 WML 文件，来实现这双重应用。但是由于以下几点：第一，各种用户不同的显示要求、不同的文件格式。第二，电子商务的交易和安全等技术涉及到一系列复杂的商务逻辑。第三，电子商务网站的信息刷新率要求高，访问的人数多。这些原因，使得页面间的导向凌乱，显示逻辑和商务逻辑混合，系统效率低，难以管理和维护。因此，合理采用 MVC 三层结构来进行开发已成为必要。

MVC (Model、View、Controller) 是 Web 应用开发中的一项重要技术，可使系统结构清晰，易于管理和扩展。基于 Web 的 MVC 模型实现了视图和控制器的分离，简化了软件开发过程中所有相关人员的工作，明确了网站开发工作中的分工，同时开发出的软件维护方便。正在越来越广泛的得到网站开发人员的认同，并被很好的应用到实际中去。

Model 是整个商务应用的模型，实际应用中所涉及到的有关类、事务、数据结构以及商务逻辑等。定义 Model 要对软件流程中的对象进行合理抽象，封装对象的属性和对象的方法。

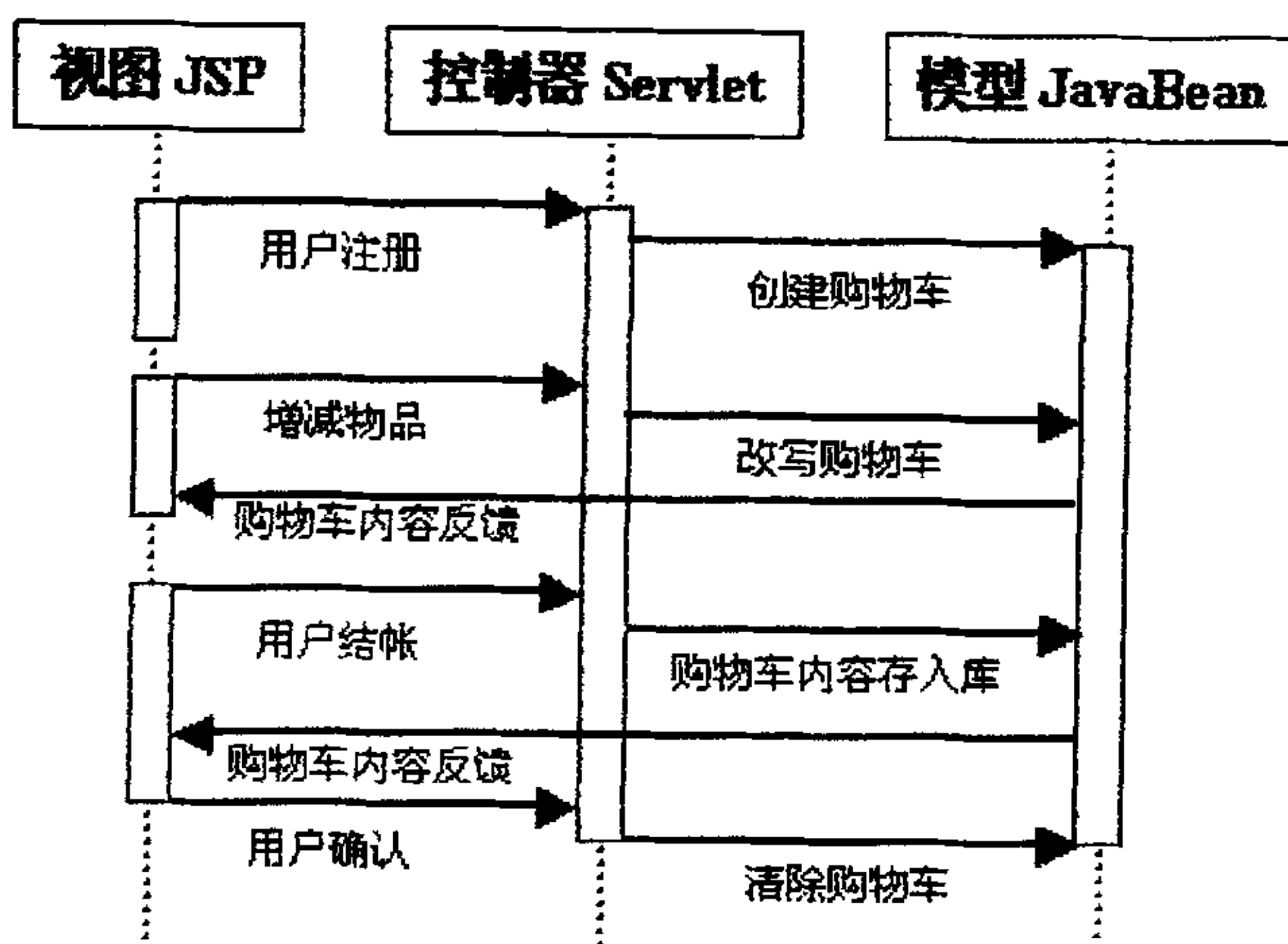
View 是用户视图，它访问 Model，将相应 Model 中的信息、商务数据等显示给用户，同时，接收用户的请求和信息的输入，然后触发 Controller 运行。

Controller 是流程控制器，从 View 获得用户的请求信息，转换为相应商务逻辑的动作并执行。根据用户交互信息调用 Model 的方法，改写 Model 的属性，使程序流程定向到新的 View 并展现给用户。

采用 MVC 设计模式，将应用的数据、数据表示和用户动作分离开来，形成数据层、表示层、事务逻辑层，各层之间是松耦合的关系，你可以修改每一层的实

现而不会影响其他的层，各层的开发工作可以同步进行，使开发人员分工明确，提高开发效率。

对于模型、视图、控制器三层结构，以一次购买活动的用况图来说明，如图（2-1）所示。



图（2-1）：一次购买活动的用况图

系统是这样实现的，用户在登录 JSP 页面注册，注册信息由 JSP 传向 Servlet，经 Servlet 核实后创建购物车模型 Java Bean，然后把购买 JSP 页面展现给用户，用户每次增减商品的操作都由 JSP 传给 Servlet，Servlet 再改写购物车 Java Bean 中的内容，并把改写后的内容反馈给用户。最后，用户结账，购物车内容写入数据库永久保存，清除购物车。

## 二、一个面向两种浏览器的 MVC 实现

下面就以一个花卉电子商务网站来实现可用于 PC 机和手机两种浏览器的 MVC，根据 MVC 的设计原理，各层之间是松耦合的关系，每一层的改变不会影响其他层设计。现在要设计出两种不同的界面（视图）来针对不同的浏览器，所以，需要有对应于手机浏览器的基于 WML 的 JSP 页面和对应于 PC 机浏览器的基于 HTML 的 JSP 页面。而商务逻辑（控制器）和数据模型（模型）是共同的，如图（2-2）所示。两种 View 使用相同的 Controller 和 Model，当相同的帐号同时从两种浏览器登录时系统也能够识别。这将会保持事务处理的唯一性，从而维护

整个系统的安全。

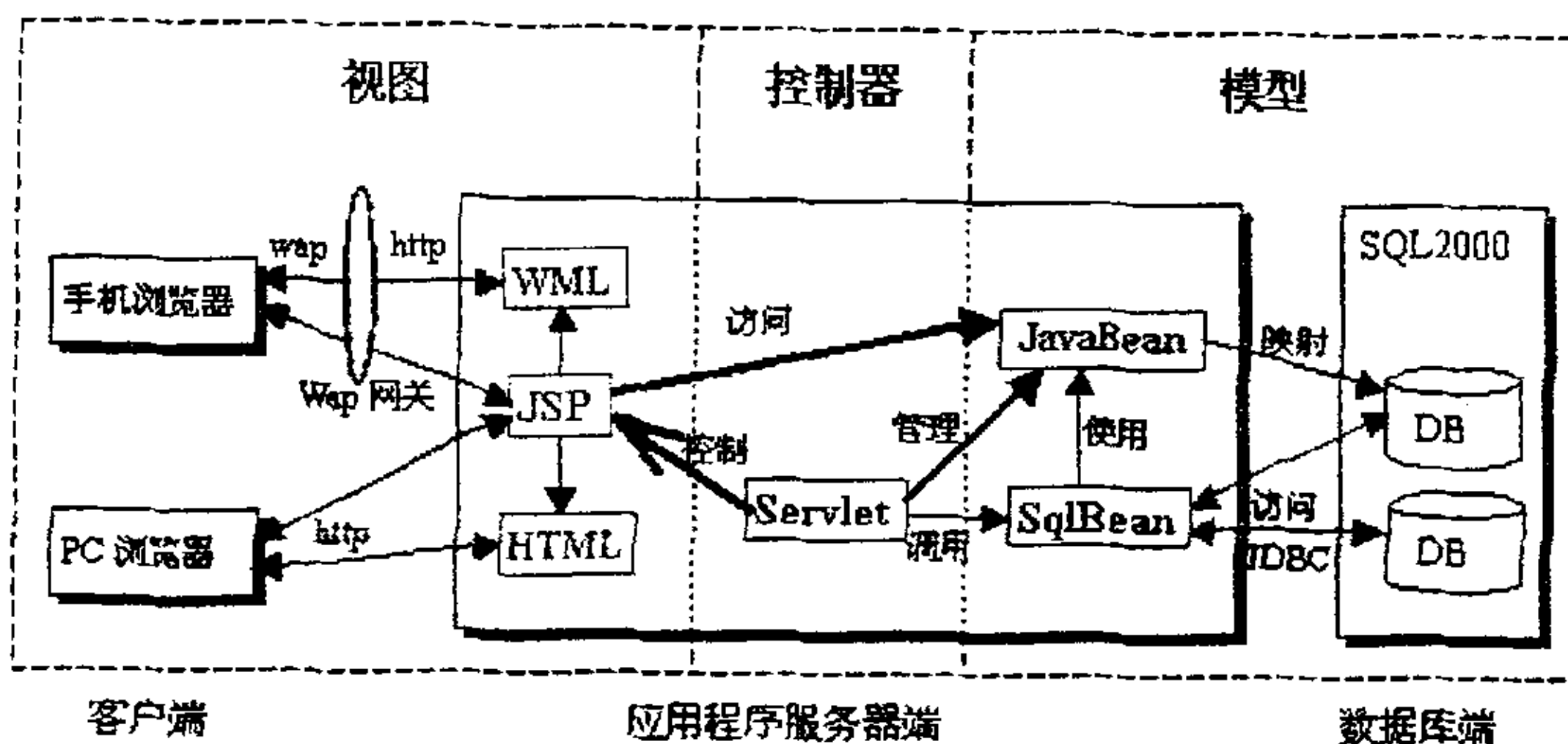


图 (2-2): MVC 三层结构图

采用 MVC 技术，结构清晰，各功能模块划分明确。擅长美工、外观布局的 Web 开发人员可专门从事 JSP 显示页面的制作工作，精通 Java 网络原理的软件程序员可以集中精力进行商务逻辑的设计工作，而各种模型类则由熟悉数据库和数据结构的人员来完成。三个部分的开发工作能同步进行，系统便于维护和扩展。实践证明，MVC 技术适用于商务、事务逻辑比较复杂的大中型 Web 系统的开发。能充分满足目前不断发展变化中的电子商务的需要，具有广阔的前景。

## 2.2 模型 Model 映射数据结构和事物

模型应能合理的映射出数据库中的表、视图及商务逻辑中所涉及到的事物。模型一般由一些 JavaBean 构成，例如，对应于用户帐户表 userid (userid 序号, username 用户名, password 密码, useraccount 帐号, money 资金数) 的模型为 userid 类，其代码如下：

```
import java.io.*;
import java.util.*;
import java.sql.*;
public class userid{    //下面定义属性
    public int userno;
```



```

        public String username, useraccount;
        public float money;
        public String password;
//构造函数
    public userid()
    {
        usernum=0;          //属性赋初值
        username="";
        .....
    }
//使用结果集参数的构造函数
    public userid(ResultSet re) throws SQLException
    {
        usernum=re.getInt("usernum");
        username=re.getString("username");
        password=re.getString("password");
        useraccount=re.getString("useraccount");
        money=re.getFloat("money");
    }

    //下面定义操作属性的方法
    public float getMoney() {return money;}
    //获得用户资金数
    public void setMoney(float pri) {money=pri;}
    //设置用户资金数
    public String getUsername() {return username;}
    //获得用户名
    .....
}

```

其中每一属性映射数据库表中的一个字段，后面我们将会看到带参数的构造函数能方便地利用结果集进行初始化。同理，对应于花卉信息表的模型为 Score 类。而对应于用户的购物车模型，则应考虑到每个购物车都有唯一的用户帐号，

同时它又要含有多个花卉信息记录。所以，购物车模型类 `buygood` 应继承于 `userid` 类，又包含多个 `Score` 类，代码如下：

```
.....

public class buygood extends userid{

    public Vector vs;    //增加一个矢量属性

    .....

    public buygood(){    //构造函数
        vs=new Vector();    //增加操作矢量属性的方法
    }

    public void delinVs(int index){vs.removeElementAt(index);}

        //用于向购物车添加物品

    public void addinVs(Score score1){vs.addElement(score1);}

        //用于删除购物车中的物品

    public Vector getVs(){return vs;}

        //获得购物车中的所有物品

    ..... }
```

可以看出，类 `buygood` 除了拥有类 `userid` 的所有属性和方法外还增加了一个矢量属性 `vs` 用于装载多个 `Score` 类的实例。同时添加了操作购物车的一系列方法，实现对购物车这一事物的抽象。这里通过类的继承体现了软件构件代码重用的思想。

## 2.3 视图 View 针对各种浏览器界面

可使用 `FrontPage`、`WAPtor` 等工具先设计出 PC 机的 HTML 文件和手机的 WML 文件，然后分别在这两种文件中都同样的加入 Java 代码以获取和显示 `Model` 中的内容，若需要也可以加入一些控制分页的代码，以查询花卉信息为例代码如下：

```
.....

Vector v=(Vector)session.getAttribute("myv");

    //通过 session 获得矢量

Score score1=(Score)v.elementAt(0);

    //从矢量中取出 Score 类
```

```

myname=score1.getName();

//用 Score 类的方法获取属性值

price=score1.getPrice();

.....

...<%=myname%>...

//用<%=...%>形式将属性值内容插入页面

...单价: <%=price%>元...

.....

```

再把文件的后缀名改为 .jsp 如: ShowHTML.jsp、ShowWML.jsp 等。下面是在两种浏览器中的显示效果图。见图 (2-3)、图 (2-4)。

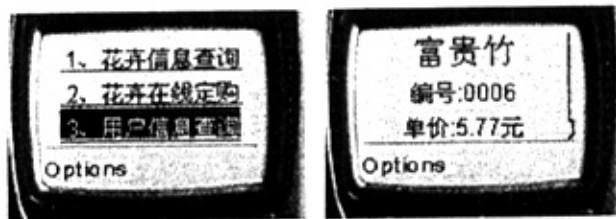


图 (2-3): 手机浏览器界面

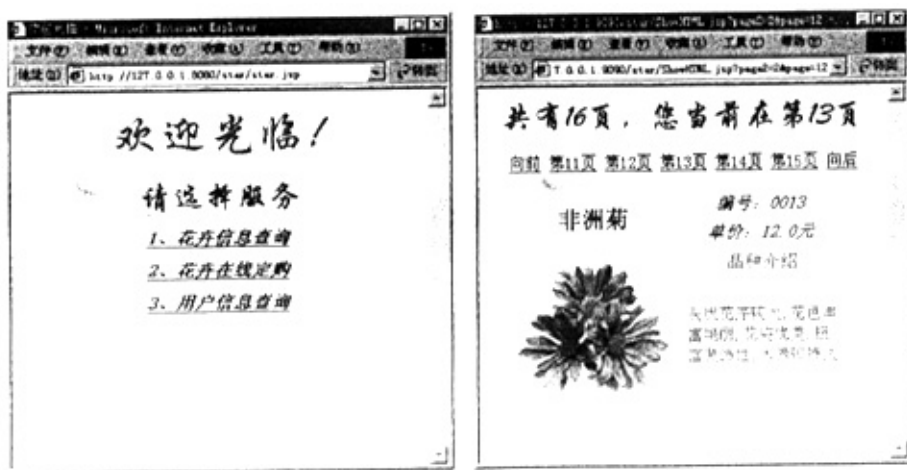


图 (2-4): PC 浏览器界面

## 2.4 控制器 实现商务逻辑、控制程序流程

在本系统中控制器主要解决两个问题。

首先，要对不同浏览器发出的请求进行判断然后决定用 ShowWML.jsp（手机）还是用 ShowHTML.jsp（PC 机）展现给用户。判断和定向的语句如下：

```
.....

String destPage="/ShowHTML.jsp";
//默认目标页面是 PC 机的 ShowHTML.jsp
String accept=request.getHeader("Accept");
//获取 accept
if(accept.indexOf("text/vnd.wap.wml")>=0) {
//判断 accept 中的 mime 类型是否为 text/vnd.wap.wml
destPage="/ShowWML.jsp"; }
//若是，则目标页面改为手机的 ShowWML.jsp
getServletContext().getRequestDispatcher(destPage).forward(request, response);
//跳转到目标页面
.....
```

其次，要根据用户的请求调用 SqlBean（数据库操作类）完成对数据库的操作，并把结果发给相应用户。控制器 Servlet 主要操作代码如下：

```
.....

Vector v=SqlBean.executeQuery("SELECT * FROM goodsinfo");
//用矢量 v 来装 SqlBean 类的查询结果
HttpSession session=request.getSession();
//通过 request 获得 session
session.setAttribute("myv", v);
//把矢量 v 放进 session，准备转向 JSP 页面
.....
```

要强调一点这里的结果不是通常所用的结果集，在 SqlBean 类中已经将查询

的结果集读出，并封装到相应 Model 类的矢量中，这样做的好处是 SqlBean 在执行完查询后可以及时关闭与数据库的连接，释放资源，不必等到 JSP 页面取出数据后才关闭，在 SqlBean 中执行查询并把结果集读出封装在矢量中的有关代码如下：

```
.....  
  
conn = connMgr.getConnection("idb");  
//通过管理类获得连接池 idb 中的一个连接  
  
Statement stmt = conn.createStatement();  
//由连接创建语句对象  
  
ResultSet rs = stmt.executeQuery(sql);  
//由语句执行 sql 定义的数据库查询操作  
//并把结果给 Resultset 对象 rs  
userid userid1=new userid(rs);  
//使用带结果集参数的构造函数  
//来构造用户信息类 userid 的一个实例  
v.addElement(userid1);  
//再把这个实例放入矢量中  
  
stmt.close();  
//查询语句关闭  
  
connMgr.freeConnection("idb", conn);  
// 连接池管理类及时收回连接池 idb 中的这个连接  
  
return v;  
//最后将装有查询结果的矢量返回  
  
.....
```

所以 Servlet 和 JSP 中操作的不是结果集而是矢量。在 3.2 视图部分可以看到 JSP 是通过 session 来获得这个矢量的，然后使用模型中的 Score 类来解析它。

## 2.5 数据库连接池技术

在使用 Java 语言进行和数据库有关的应用开发中，一般都使用 JDBC 来进行和数据库的交互，其中有一个关键的概念就是 Connection（连接），它在 Java 中是一个类，代表了一个通道。通过它，使用数据的应用就可以从数据库访问数据了。对于一个简单的数据库应用，由于对于数据库的访问不是很频繁。这时可以简单地在需要访问数据库时，就新创建一个连接，用完后就关闭它，这样做也不会带来什么明显的性能上的开销。但是对于一个复杂的数据库应用，情况就完全不同了。频繁的建立、关闭连接，会极大的减低系统的性能，因为对于连接的使用成了系统性能的瓶颈。

本节给出的方法可以有效的解决这个问题。在方法中提出了一个合理、有效的连接管理策略，避免了对于连接的随意、无规则的使用。该策略的核心思想是：连接复用。通过建立一个数据库连接池以及一套连接使用管理策略，使得一个数据库连接可以得到高效、安全的复用，避免了数据库连接频繁建立、关闭的开销。另外，由于对 JDBC 中的原始连接进行了封装，从而方便了数据库应用对于连接的使用（特别是对于事务处理），提高了开发效率，也正是因为这个封装层的存在，隔离了应用的本身的处理逻辑和具体数据库访问逻辑，使应用本身的复用成为可能。

首先，要建立一个静态的连接池，所谓静态是指，池中的连接是在系统初始化时就分配好的，并且不能够随意关闭的。Java 中给我们提供很多容器类可以方便的用来构建连接池，如：Vector、Stack 等。在系统初始化时，根据配置创建连接并放置在连接池中，以后所使用的连接都是从该连接池中获取的，这样就可以避免连接随意建立、关闭造成的开销。

其次，提供一套自定义的分配、释放策略。当客户请求数据库连接时，首先看连接池中是否有空闲连接，这里的空闲是指，目前没有分配出去的连接。如果存在空闲连接则把连接分配给客户，并作相应处理，主要的处理策略就是标记该连接为已分配。若连接池中沒有空闲连接，就在已经分配出去的连接中，寻找一个合适的连接给客户，此时该连接在多个客户间复用。当客户释放数据库连接时，可以根据该连接是否被复用，进行不同的处理。如果连接没有使用者，就放入到连接池中，而不是被关闭。可以看出正是这套策略保证了数据库



连接的有效复用。

系统数据库采用 SQL Server 2000。为使数据库运行稳定, 采用了连接池技术, 增加 DBConnectionManager.class (连接池管理者) 来配合 SqlBean.class 工作。部分代码的说明如下: DBConnectionManager.java

```

.....

// 管理类 DBConnectionManager 支持对一个或多个由属性文件定义的数据库连接池的访问。客户程序可以调用 getInstance() 方法访问本类的唯一实例。

public class DBConnectionManager {
    static private DBConnectionManager instance; // 唯一实例
    .....

// 返回唯一实例. 如果是第一次调用此方法, 则创建实例
static synchronized public DBConnectionManager getInstance() {
    if (instance == null) {instance = new DBConnectionManager();}
    clients++;
    return instance; }

// 建构私有函数以防止其它对象创建本类实例
private DBConnectionManager() {init();}

// 获得一个可用的(空闲的)连接. 如果没有可用连接, 且已有连接数小于最大连接数限制, 则创建并返回新连接

public Connection getConnection(String name) {
    DBConnectionPool pool = (DBConnectionPool) pools.get(name);
    if (pool != null) {return pool.getConnection(); }
    return null; }

// 关闭所有连接, 撤销驱动程序的注册

public synchronized void release() {
    // 等待直到最后一个客户程序调用
    if (--clients != 0) { return; }
    .....

// 此内部类定义了一个连接池。它能够根据要求创建新连接, 直到预定的

```

最大连接数为止。在返回连接给客户程序之前,它能够验证连接的有效性。

```

class DBConnectionPool {
    private int maxConn; //此连接池允许建立的最大连接数
    private String name; //连接池名字
    private String password; //密码
    private String myurl; //数据库的 JDBC URL
    private String user; // user 数据库帐号
    .....

    // 创建新的连接池
    public DBConnectionPool(String name, String myurl, String user,
String password, int maxConn) {
        this.name = name;
        this.myurl = myurl;
        this.user = user;
        this.password = password;
        this.maxConn = maxConn; }

    // 将不再使用的连接返回给连接池
    public synchronized void freeConnection(Connection con) {
        // 将指定连接加入到向量末尾
        freeConnections.addElement(con); checkedOut--;
        notifyAll(); }

    // 关闭所有连接
    public synchronized void release() {
        Enumeration allConnections = freeConnections.elements();
        while (allConnections.hasMoreElements()) {
            Connection con = (Connection) allConnections.nextElement();
            try { con.close();
                log("关闭连接池" + name+"中的一个连接");}
            catch (SQLException e) {

```

```

log(e, "无法关闭连接池" + name + "中的连接"); } }
freeConnections.removeAllElements(); }
.....

```

此数据库连接池 大大地提高了数据库的性能和稳定性, 是 JSP 三层模型顺利实现的后台基础, JSP 服务器通过数据库连接池访问 JDBC-ODBC 桥 从而达到访问数据库的目的。同时, 数据库连接池使用方便, 可灵活部署在不同的服务器上, 而只需修改一个属性文件 db.properties 。其内容如下:

```

drivers=sun.jdbc.odbc.JdbcOdbcDriver //数据库驱动程序
logfile=D:\\bao\\log.txt //日志文件位置
idb.url=jdbc:odbc:guestbook //数据库名
idb.user=sa //用户帐号
idb.password=ww //密码
idb.maxconn=2 //最大连接数

```

连接池的工作情况可通过查看 log.txt 日志文件来获得, 下面是某次使用后的日志文件记录情况:

```

Thu Nov 07 23:08:14 CST 2002: 成功注册 JDBC 驱动程序 JdbcOdbcDriver
Thu Nov 07 23:08:14 CST 2002: 成功创建连接池 idb
Thu Nov 07 23:08:14 CST 2002: 连接池 idb 创建一个新的连接
Thu Nov 07 23:08:15 CST 2002: 连接池 idb 收回一个连接
Thu Nov 07 23:08:15 CST 2002: 释放连接池 idb 中的一个连接
Thu Nov 07 23:08:15 CST 2002: 撤销 JDBC 驱动程序 JdbcOdbcDriver 的注册

```

作为移动设备程序开发和运用的平台 J2ME 基于 Java 体系,有着良好的软硬件兼容性,代表了 Internet 移动电子商务实现技术的发展方向。通过 MIDP 编程中的 Display 类可设计移动终端界面。由 MIDlet 与 Servlet 通讯来实现手机用户和服务器的交互。

### 第三章 移动电子商务 Java 实现

通过 Internet 网进行电子商务,通过移动电话网进行移动电子商务。PC 机上运行的 HTML 文件能传递每种商品的大量信息和彩色图片,具有很强的表现力,使消费者能详细了解和比较,从而,决定去购买。手机的特点是:操作简单、携带方便、可信度和安全度高,所以真正的购买行为大多是通过手机来完成的。一个成功的电子商务网站应该同时提供这两种服务,使它们有机地配合,协调地运行。

实现移动通讯设备上网的两种方法是:WAP、J2ME。本章通过对 Java 手机软件开发及相关技术的阐述,以简单的例子,介绍了基于 J2ME 的 Java 手机软件开发的一般步骤。Java 手机即将普及,对其软件的开发也应该是业界的一个热门,开发出实用的基于 J2ME 的 Java 手机软件是未来发展的方向。

J2ME: Java 的最大目标和特点,就是“一次编写,到处运行”的平台无关性。但是,很自然的,正如 Sun 认识到的,“One size does'nt fit all”,一套标准无法适应各种不同的需求。因此,Java 技术目前共有三套,分别针对不同的平台和应用。

- **Standard Edition** (J2SE, 标准版): 针对桌面端 PC 和工作站的个人和低端商务应用。
- **Enterprise Edition** (J2EE, 企业版): 针对服务器端企业级应用,支持 Servlets, JSP 和 XML 等等。
- **Micro Edition** (J2ME, 袖珍版, 也有翻译为小型版或者移动版的。)针对有限内存,显示和处理能力的设备,主要是消费电子和嵌入式设备领域(这实际正是 Java 语言设计最初的目标领域)。

J2ME 技术有两个设计中心：手持的设备，和可以插到墙上插座的设备。第一类设备的 Configuration 称为 CLDC (Connected, Limited Device Configuration)，第二类称为 CDC (Connected Device Configuration)。Configuration 就是支持一组通用设备的最小 Java 平台，这里的 Java 平台主要是指 Java 虚拟机 (JVM) 和核心库。但是仅仅有 Configuration 显然是不够的，特殊的具体设备其独有的功能和硬件条件都没有得到支持。建筑于 Configuration 之上，作为 Configuration 的扩展和补充。如 MIDP、PDAP。关于 J2EE、J2SE、J2ME 体系见图 (3-1)。

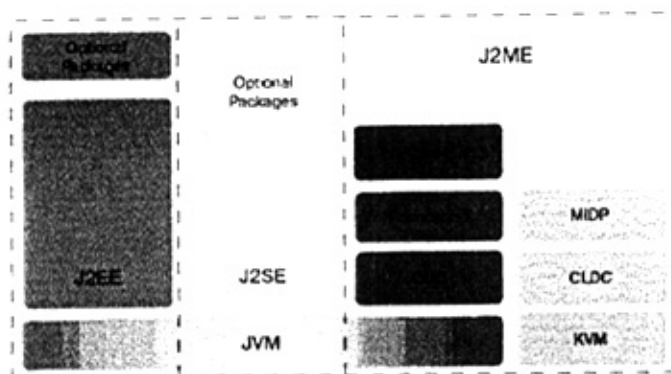


图 (3-1) J2EE、J2SE、J2ME 体系结构

### 3.1 WAP 与 J2ME

WAP (Wireless Application Protocol, 无线应用协议) 是 XML 的一个应用，其目的是在无线设备如手机上显示 Internet 内容。由于无线设备的显示区域有限，需要将标准的 HTML 重新调整以适应硬件条件。

WAP 实现上网简述如下：手机发出请求，移动网络 (GSM, CDMA 或其它移动网络) 接通本地 ISP；本地 ISP 接通 WAP 网关服务器。WAP 网关服务器向目标内容服务器发出请求，目标内容服务器将信息传给 WAP 网关服务器；WAP 网关服务器将处理后的信息发到本地 ISP；本地 ISP 将传回移动网络。手机接受到 Internet 信息，其中的关键部分为 WAP 网关。

许多听说 J2ME 的人都会想到 WAP，其实 WAP 和 J2ME 并不冲突，而且是很好的互补。WAP 对于文本为主的内容是很合适的，需要持续的网络连接。但是对于图形较多的应用 WAP 就不太合适，只能提供轻量级的脚本执行能力。WAP 的网关

也有安全性问题。而 Java 技术可以用于间断的网络连接, 可以将应用和服务逻辑在设备和服务器间分离, 很适合于图形应用, 有很稳定而可靠的安全模型。

结合 WAP 和 Java 技术的一个方法是在设备上安装一个 WAP 浏览器, 并且实现 CLDC 和 MIDP 及其与 WAP 浏览器之间的通讯。也有一些 WAP 技术是用 Java 实现的, 比如 K Browser (<http://www.4thpass.com>) 就是一个用 Java 实现的浏览器, 运行于 J2ME 之上。

无线 Java 技术: 无线这个概念在 J2ME 的相关文章中经常出现, 但是, 无线 Java 不等于 J2ME。J2ME 中, 无线设备只是其中的一小部分。而无线 Java 技术也可能包括这种情况: 在笔记本上运行 J2SE 应用, 通过 802.11 LAN 连接网络。MIDP 不是全部的 J2ME。MIDP 发布最早, 因此也得到了最广泛的支持, 但是 J2ME 当然不仅仅是 MIDP。MIDP 不是全部的无线 Java 技术。还有许多其他 Java 技术属于无线技术, 比如 Personal Java, PDA Profile, 甚至无线设备上的 J2SE。

在消费电子和嵌入式设备的广阔领域中, 目前最受关注的是移动信息设备, 移动信息设备主要包括 PDA 和智能手机, 现在和将来都还会有一部分设备处于 PDA 和智能手机交界的位置。其中手机的市场远比 PDA 要大得多。Symbian 实现了 CLDC 和 MIDP, 作为智能手机的操作系统, 是理想的 MIDP 应用平台。基本可以相信, 主要的移动信息设备操作系统都将一直提供 J2ME 的支持。

J2ME 虽然和 J2SE 有着许多不同, 但仍然属于 Java 技术, 具有 Java 方便开发的优点, 也使得 Java 程序员学习移动信息设备开发没有太大困难(事实上, 开发 J2ME 应用的主要困难不在于具体编码, 而在于标准的复杂性)。

用于 J2ME 应用程序虚拟机的被称作 Kilobyte virtual machine 或简称 KVM。就像它名称的含义, KVM 比较小, 通常只有 128K 或更少。这比起我们通常了解和使用的 Java 2 标准版 Java 虚拟机 (JVM) 的 32 MB 来说就小得多了。用于连接虚拟机的是一系列配置和简表, 它们提供了用于特定 J2ME 环境的类应用程序接口 见图 (3-2)。



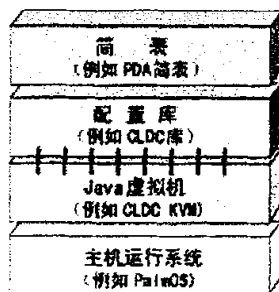


图 (3-2) J2ME 层次

Java 虚拟机是 J2ME 技术的核心，但是配置和简表提供特殊环境的类应用程序接口。配置是用于一组通用设备的最小的 Java 平台，而简表则为具体的设备家族或特别的应用程序提供更具体的能力。

Java 规范定义的这两种配置是 Connected Limited Device Configuration (有限连接设备配置，CLDC) 和 Connected Device Configuration (连接设备配置 CDC)。CLDC 是为使用较小的存储容量的设备设计的，而 CDC 用于比 PC 机小但是具有比 512K 内存多的设备。

简表为相同消费电子设备的不同的生产商提供了标准化的 Java 类库，五个已知简表已经有了规范，记住，每个简表的责任都是为了完善配置的不足，下表列出了这五个简表：

简表	完善配置
<b>Mobile information devices profile (MIDP)</b>	移动电话和呼叫器 <b>CLDC</b>
<b>Personal digital assistant profile</b>	<b>Palm 和 Handspring 的 PDA 设备 CLDC</b>
<b>Foundation profile</b>	用于所有不需要 GUI 的 <b>CDC</b> 设备的标准简表 <b>CDC</b>
<b>Personal profile</b>	替代 <b>PersonalJava</b> 的 <b>Foundation</b> 完善的简表 <b>CDC</b>
<b>RMI profile</b>	提供 <b>RMI</b> 的 <b>Foundation</b> 完善的简表 <b>CDC</b>

如果你想为小型信息家电编写 Java 应用程序的话，你就需要两个前提：一个配置和至少一个简表。现在，一般是配置捆绑了虚拟机和一套针对你的平台能够用的横向分组设备的 Java 类库。其次，你至少还需要一个简表来为你的平台提供附加的 Java 类，这个简表通常会为你的设备提供用户界面、输入和数据库类。有了这两个前提，你就使用了 Java 为你的设备编写应用程序的基本的 J2ME 环境。

### 3. 2 移动设备与 MIDlet

MIDP 应用程序称为 MIDlet, 为了创建一个 MIDlet, 必须写一个扩展基本 MIDlet 类的类。这有点类似常见的 applet 或 servlet。MIDlets 独有的东西是把多个 MIDlet 组成一个 MIDlet 套件的能力。这就允许 MIDlet 在一个单独的 JVM 环境中共享资源, 比如一个数据库等等。当 MIDlet 被请求时, MIDlet 通过构造程序实例化, 然后调用实例的 startApp() 方法。MIDlet 的用户界面或显示是由 Display 类的一个实例管理的。对于每个 MIDlet, 只有一个显示管理器实例。所有可以显示的项目, 像屏幕或画布 (canvas), 通过这个管理器都能够成为可见的。因为移动电话和呼叫器能力的多样化, 又因为用于这些设备的应用程序类型的差异, MIDP 规范提供了两种类型的用户界面。一个可移植性稍差、明确设备、低水平的应用程序接口, 允许图形元素精确的控制和放置。这个接口类型是用于应用程序特性比较典型的设备特别设计的, 比如电子游戏。一个可移植性稍好的、抽象的、高级的 GUI 应用程序接口, 提供来用于商业应用程序。

在本设计中使用的是高级的应用程序接口和典型的用户界面组件 (文本框, 列表等等), 是这类界面通用的。

由于是电子商务网站, 手机端用户也要通过 HTTP 协议来访问服务器, 与服务器进行对话来完成商务活动。所以必须实现网络功能。而 J2ME 一个很重要的特性就是使用 J2ME 连接结构打开网络连接并传送数据的能力。javax.microedition.io 包内的这个结构包括 Connection 类和好几个很有用的接口 (包括 StreamConnection、ContentConnection 和 HTTPConnection)。

MIDP 程序称为 MIDlet, 这可能因为所有的 MIDlet 都是扩展 javax.microedition.midlet.MIDlet 类 (正象 Java applet 扩展 Applet 类一样) 除了从键盘或点击设备上接受输入的信息以外, MIDlet 类还提供用于激活、暂停和终结 MIDlet 的接口, 即分别是 startApp()、pauseApp()和 destroyApp() 方法。startApp()方法在概念上与 Java applet 的 start()方法类似, 当 MIDlet 启动时它被调用, 而且在一个 MIDlet 暂停之后恢复时也被调用。

### 3.3 用 Java 体系设计移动终端界面

移动用户终端界面以手机为例，用户首先是进入欢迎界面，有三项服务（1、花卉信息查询 2、花卉在线订购 3、用户信息查询）可供选择。欢迎界面 WelcomeScreen.java 扩展了 List 列表类，程序如下：

```
import javax.microedition.lcdui.*;
import java.util.*;

public class WelcomeScreen extends List implements CommandListener
{
    private Controller _controller = null;

    public WelcomeScreen(Controller controller)
    {
        super(" 请选择服务:", List.IMPLICIT);
        _controller = controller;
        Ticker ticker = new Ticker("欢迎您光临在线花卉站点");
        //显示滚动标题
        this.setTicker( ticker);
        initialize();    }

    private void initialize()    //提供三项服务
    {
        append(" 花卉信息查询", null);
        append(" 花卉在线订购", null);
        append(" 用户信息查询", null);
        this.setCommandListener(this);    }

    public void commandAction (Command c, Displayable d)
    {
        .....

        switch (shown.getSelectedIndex())
        { case 0:  _controller.nextScreen( new QueryScreen( _controller) );    break;
          case 1:  _controller.nextScreen( new BuyScreen( _controller) );    break;
          case 2:  _controller.nextScreen( new UserScreen ( _controller) );
          .....    }
    }
}
```

可以看出 WelcomeScreen 类还实现了 CommandListener 接口。程序可根据

用户的选择来进入 QueryScreen 查询界面, BuyScreen 花卉购买界面, UserScreen 用户信息界面。查询界面、花卉购买界面和用户信息界面等都是用来向用户显示有关花卉内容文字信息的, 有着共同之处, 把这些共用部分抽象出来作为一个基类即界面的父类 Flower Screen.java。有助于代码重用。界面父类如下:

```
import javax.microedition.lcdui.*;

public class FlowerScreen extends Form    //扩展 Form 框架类
{    //定义 5 个标签私有变量
    private StringItem item1 = null;
    .....
    //定义 2 个文本输入框私有变量
    protected TextField _innumber = null;
    protected TextField _mima = null;
    //定义 2 个命令按钮
    protected Command nextCommand = null;
    protected Command backCommand = null;
    protected Controller _controller = null;
    public FlowerScreen(String title, Controller controller)
    {    super(title);    //先执行父类构造函数
        _controller = controller;
    }
    //1 个参数的 displayScreen()函数
    public void displayScreen(String s1)
    {    item1 = new StringItem(" ",s1);
        _mima = new TextField(" 至少 7 位:", "", 9, TextField.ANY);
        append(item1);
        append(_mima);
        generateButtons();
    }
    //2 个参数的 displayScreen()函数
    public void displayScreen(String s1,String s2)
```

```

        { ..... }

//3 个参数的 displayScreen()函数
        public void displayScreen(String s1, String s2,String s3)
        { ..... }

//4 个参数的 displayScreen()函数
        public void displayScreen(String s1, String s2, String s3, String s4)
        { ..... }

//5 个参数的 displayScreen()函数
public void displayScreen(String s1, String s2, String s3, String s4,String s5)
        { ..... }

//添加按钮并实现命令监听的函数
public void generateButtons()
{   this.nextCommand = new Command("确定", Command.SCREEN, 1);
    this.backCommand = new Command("返回", Command.BACK, 1);
    this.addCommand(backCommand);
    this.addCommand(nextCommand);
}
}

```

这个类继承了 Form 框架类，类中主要的三种属性：StringItem、TextField、Command，分别用来表示标签、文本框和命令按钮，同时实现了对命令按钮的监听。标签、文本框可显示信息和获取用户输入。具体的界面显示工作由 displayScreen()函数完成，此函数有一个参数到五个参数的不同形式，可根据具体情况来调用，实现了方法的重载。在本设计中共有：QueryScreen.java、DisplayScreen.java、MimaScreen.java、BuyScreen.java、BuyplayScreen.java、FinishScreen.java、UserScreen.java、UseplayScreen.java 八个类是界面父类 FlowerScreen.java 的扩展类。它们分别是：花卉查询界面、查询结果显示界面、密码验证界面、花卉购买界面、购买显示界面、购买完成信息反馈界面、查询用户信息界面、显示用户信息界面。

由于 FlowerScreen.java 界面父类中封装了用于显示和控制的大部分属性

和方法，所以它的子类就只须实现一些具体的内容（如不同界面的标题，不同界面要显示的标签及不同的文本输入框等），相对来说编程代码就简单的多。QueryScreen.java 类继承 FlowerScreen.java 类，展现查询界面并完成与用户的交互。代码如下：

```
import javax.microedition.lcdui.*;

public class QueryScreen extends FlowerScreen implements CommandListener
{
    public QueryScreen(Controller controller)
    {
        super(" 花卉查询", controller); //界面标题
        _controller.putin("c", "", "", "star"); //设置状态字
        _controller.mystart('c'); //设置控制字符
        String sl="10";
        Flower bbhh=_controller.myget();
        sl=bbhh.getf0();
        super.displayScreen("花卉"+sl+"种", "花卉编号输三位数(XXX)");
        //设置标签内容
        this.setCommandListener(this); //命令监听
    }

    public void commandAction(Command c, Displayable d)
    {
        if ( c == backCommand)
        {
            _controller.delall(); //重置状态字
            _controller.lastScreen(); //返回上一界面
        }
        else
        {
            _controller.delall(); //重置状态字
            String innumber = _innumber.getString(); //获得编号
            String inn="h"+innumber;
            _controller.putin(inn, "", "", "star");
            //放入要查询的号码
            _controller.mystart('h'); //执行查询
        }
    }
}
```

```

_controller.nextScreen( new DisplayScreen( _controller) );
        //转到查询结果界面
    }
}

```

同理, BuyScreen. java 类也继承 FlowerScreen. java 类, 展现了在线购买界面并完成与用户的交互。其代码和 QueryScreen. java 类相比只有很少的不同之处: 现把它们列在下面以便比较。

```

super(" 花卉购买", controller); //界面标题不同
_controller.putin("c", "", "", "star"); //设置状态字不同
_controller.mystart('c'); //设置控制字符不同
super.displayScreen("花卉"+s1+"种", "要购买的花卉编号三位(XXX)");
        //设置标签内容不同
_controller.lastScreen(); //返回上一界面
_controller.nextScreen( new BuyplayScreen( _controller) );
        //转到购买完成界面

```

在此只介绍了 QueryScreen. java 类和 BuyScreen. java 类其它的几个类与它们相似, 不再一一介绍。部分界面的效果如图 (3-3) 所示:



图 (3-3) 部分界面显示效果图

这里需说明的一点是，不同界面之间的转换问题，是通过 `public void nextScreen(Displayable display)` 和 `public void lastScreen()` 等方法来完成。这些方法的定义在主类 `FlowerManager.java` 中，而方法的说明在接口 `Controller.java` 中。主类（即由 `MIDlet` 扩展而得）通过实现 `Controller` 接口来定义 `nextScreen()` 和 `lastScreen()` 等方法。

//进入下一屏幕界面的方法

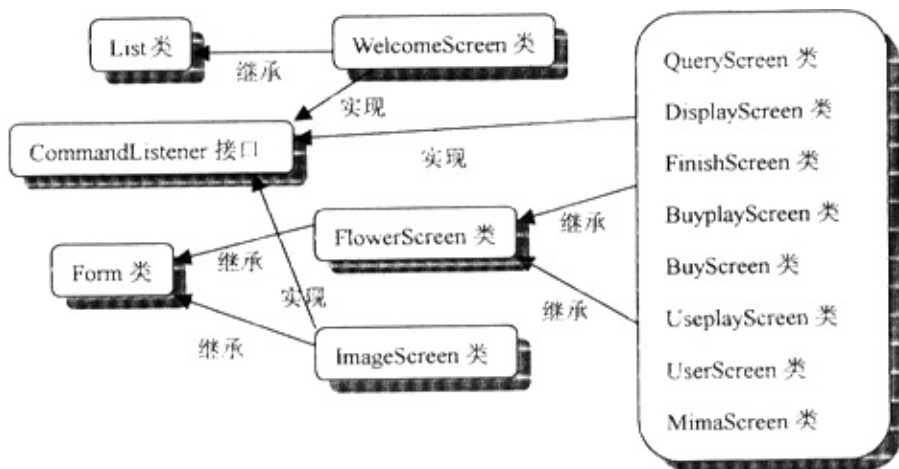
```
public void nextScreen(Displayable display) //形参为屏幕类 display
{
    Displayable currentScreen = getDisplay().getCurrent();
    //获取当前屏幕类
    if ( currentScreen != null) //如果存在
    {
        _screenStack.push( currentScreen );
    } //则把它压入到堆栈_screenStack 中
    getDisplay().setCurrent( display); //把当前屏幕设置为 display
}
```

//返回上一屏幕界面的方法

```
public void lastScreen()
{
    Displayable display = null; //定义一个屏幕类 display
    if ( !_screenStack.empty() ) //如果_screenStack 堆栈非空
    {
        display = (Displayable) _screenStack.pop();
    } //则从_screenStack 堆栈中弹出一个屏幕类并赋值给 display
    else
    {
        display = new WelcomeScreen( (Controller) this );
    } //否则, 新建一个欢迎界面
    getDisplay().setCurrent( display); //转向欢迎界面 }
}
```



在不同界面的转换中,要依赖一个界面堆栈属性, private Stack \_screenStack 这个私有属性 栈 用来将所经历过的所有界面依次推入其中,当要返回时又从栈中将最近的一个界面弹出,展现给用户。移动终端界面的类和接口的继承关系如图(3-4)所示:



图(3-4) 移动终端界面的类和接口的继承关系

另外,为显示花卉图片,使用 ImageScreen 类直接继承 Form 类。其中有语句:

```
private Image imm = null; //定义一个 Image 类来存放图片
```

(这里采用 J2ME 中常用图片格式, \*.Png 格式)

并通过方法 displayScreen()来获得和显示图片,

```
public void displayScreen()
```

```
{ imm=_controller.getImage();
```

```
append(imm);
```

```
generateButtons(); }
```

图片彩色显示效果如图(3-5)



图(3-5) 移动用户端彩色图片显示

### 3. 4 J2ME 实现移动终端与网络服务器的通讯

整个移动终端由运行一个 MIDlet 来控制，这个 MIDlet 是 FlowerManager.java, 它扩展了 MIDlet 并实现了 Controller 接口, 而 Controller 接口声明了:

nextScreen() 下一屏幕, lastScreen() 上一屏幕,  
 putin(String x1, String x2, String x3, String x4) 设置状态字符串,  
 mystart(char hy) 与服务器交互, delall() 清除状态字符串,  
 mygetmima() 获得密码, mysetmima(String mm) 设置密码,  
 mystarttp(String tp) 向服务器获取图片, getImage(), 返回图片等一系列控制 MIDlet 运行以及与服务器通讯的方法, 而具体这些方法又在 FlowerManager.java 中得到实现。

```

.....

public class FlowerManager extends MIDlet implements Controller
{
    private Stack _screenStack = null; //定义屏幕堆栈初值为空
    private String usermima=null;    //定义用户密码初值为空
    private static String url =
        "http://127.0.0.1:8080/star/servlet/SampleServer";
        //url 与服务器的 SampleServer.class 相连
    String text[]={"", "", "", ""}; //定义程序状态字符串数组初值为空
    .....

    public FlowerManager() //构造函数
    {
        _screenStack = new Stack(); //建立屏幕堆栈
        nextScreen( new WelcomeScreen( (Controller) this) ); //显示欢迎界面
    }
    .....

    private Display getDisplay() //获取屏幕显示类 Display;
    {
        if ( _display == null)
        {
            _display = Display.getDisplay(this);
        }
        return _display;
    }

    public void putin(String x1, String x2, String x3, String x4)

```

```

    { text[0]=x1; text[1]=x2; text[2]=x3; text[3]=x4; }
//向服务器传送字符串 text0, 并接收服务器的应答放在字符串 output 中

    public void mystart(char hy)
    { //将字符串 text0 转换为字节数组放在 data 中
    try { ByteArrayOutputStream bout = new ByteArrayOutputStream();
        DataOutputStream dout = new DataOutputStream( bout );

        if(hy=='h' || hy=='c' || hy=='m') //执行查询、密码等操作
        {dout.writeInt(1); //向服务器发送一个参数 text[0]
            dout.writeUTF( text[0] );
            if(hy=='u') //执行花卉品种查询操作
            {dout.writeInt(2); //向服务器发送两个参数 text[0]、text[1]
                ..... }

            if(hy=='y') //执行用户信息查询操作
            {dout.writeInt(3); //向服务器发送三个参数 text[0] 、text[1] 、text[2]
                ..... }

            if(hy=='g') //执行花卉购买操作
            {dout.writeInt(4); //向服务器发送四个参数
                ..... }

            data = bout.toByteArray();
            dout.close();
        }

        catch( IOException e ){ // should handle this.... }

//建立和服务器的链接并传输协议头, 把字节数组 data 中的数据发到服务器

        HttpURLConnection conn = null;
        try { conn = (HttpURLConnection)Connector.open( url );
            conn.setRequestMethod(HttpURLConnection.POST );
            conn.setRequestProperty( "User-Agent",
                "Profile/MIDP-1.0 Configuration/CLDC-1.0" );
            conn.setRequestProperty("Content-Language", "en-US");
            conn.setRequestProperty("Accept", "application/octet-stream");

```

```

conn.setRequestProperty("Connection", "close" );
conn.setRequestProperty("Content-Length",
Integer.toString( data.length ) );

    OutputStream os = conn.openOutputStream();
    os.write( data );
    os.close();      }
    catch( IOException e ){ }

//接收服务器传送来的数据放在字符串缓存类 text 中,
//最后存放到字符串 output 中;

    try{    int rc = conn.getResponseCode();
        if( rc == HttpURLConnection.HTTP_OK )
        { StringBuffer text = new StringBuffer();
DataInputStream din=new DataInputStream(conn.openInputStream() );

        .....

        din.close();

        conn.close();    }    .....    }

```

按照地址: <http://127.0.0.1:8080/star/servlet/SampleServer>, FlowerManager 与服务  
器 8080 端口下 star/servlet/下的 SampleServer 通讯, 来完成移动客户与服务  
器的交互操作。SampleServer 扩展了 HttpServlet, 用方法 doPost  
(HttpServletRequest request, HttpServletResponse response)来实现通讯的  
链接, 其中根据从用户端发来的操作字不同而有不同的动作并将结果发回到用户  
端。部分代码说明如下:

```

.....

public class SampleServer extends HttpServlet    //扩展了 HttpServlet 类
{ public void doPost (HttpServletRequest request,
    HttpServletResponse response)
throws ServletException, IOException {
    ServletInputStream in=request.getInputStream(); //建立链接
    DataInputStream din=new DataInputStream(in);

```

```

String text[]={ "", "", "", "" }; //用来装用户字段初值为零
int n=din.readInt();           //读取用户字段数
String sstt=null;
if(n>0)    //如果有
{int j=0;
    while( n-- > 0 ) //循环
    {
        sstt=din.readUTF(); //读取用户请求字段
        text[j]= sstt;    j++;
    }
    din.close(); //关闭输入流
    .....
    ByteArrayOutputStream bout=new ByteArrayOutputStream();//建立字节数组输出流
    DataOutputStream dout=new DataOutputStream(bout);//套接字符数组输出流
    //然后根据不同请求来选择操作
    if(text[0].startsWith("h"))//如果操作字符为"h"则查询花卉信息
    {
        .....
        v=qq.executeQuery("SELECT * FROM goodsinfo where number='"+text[0]+"");
        //执行查询把结果赋值给矢量 v
        if(v.size()>0) //如果有则取出数据
        {
            Score hscore=new Score();
            hscore=(Score)v.firstElement();
            hname=hscore.getName(); //获得名称
            hnumber=hscore.getNumber(); //获得编号
            float hprice=hscore.getPrice(); //获得单价
            shprice=tomyString(hprice);
        }
        dout.writeInt(3); //反馈三个数据给用户
        dout.writeUTF(hname); //名称
        dout.writeUTF(hnumber); //编号
        dout.writeUTF(shprice); //单价
    }
}

```

```

if(text[0].startsWith("g"))    //如果操作字符为"g"则执行用户购买插入记录
    {.....}

if(text[0].startsWith("y"))    //查询用户信息
    .....

//用如下方式将图片发送给客户端
.....

sstt=din.readUTF();    读取所需图片名称
File f=new File("c:\\tomcat\\webapps\\star\\myimage\\"+sstt+".png");
    //找到图片文件

    int filelength=(int)f.length();    // 取图片文件长度
    byte buffer[]=new byte[filelength];

    FileInputStream fin=new FileInputStream(f); // 设置文件的输出流
fin.read(buffer,0,filelength);
response.setContentType("application/octet-stream");
    response.setContentLength(buffer.length);
    response.setStatus(response.SC_OK);

    OutputStream out=response.getOutputStream(); //打开输出流
    out.write(buffer); //输出图片文件给客户端
    out.close(); //关闭输出流
.....

```

至此，可以看到通过移动用户端的 MIDlet 与服务器端的 Servlet 进行对话，就能实现 J2ME 移动终端与网络服务器的通讯。而且 MIDlet 和 Servlet 都是基于 Java 语言的，无论是相互之间的兼容性，还是整个运用的跨平台性，都有很好的表现。

XML 与 Java 结合是实现大型电子商务网站的最佳技术组合, 在一个 JSP 的 MVC 三层模型中使用 XML 来布局。能: 映射 Servlet 动作、封装 Java 代码、装配 JSP 文件、统一 Data 格式。达到 JSP 页面构件重用, 产生 XML 形式的规范数据格式, 为站点运行维护以及功能扩展奠定了坚实的技术基础。

## 第四章 模板布局 XML 结构

随着 Internet 网的迅速发展, 电子商务、电子政务、远程教育等, 基于服务器和客户端的技术逐渐得到普及。Web 服务器的编程, 正经历一场从产生、发展到成熟的变革, CGI、PHP、ASP 到 JSP, 一浪高过一浪。特别是 JSP, 以 Java 语言作为基础, 一处编程多处运行, 有着良好的跨平台性, 以线程作为运行的单位, 大大地提高了网络的性能。计算机、手机所有可联网的家用电器都能通过网络与 JSP 服务器交换信息, J2EE、J2SE、J2ME 分别为不同可编程设备规范了平台, 更是进一步拓广了 JSP 服务器的应用范围。

比较成熟的 JSP 技术采用 MVC 三层结构 (即模型 Model、视图 View、控制器 Controller)。模型对流程中的数据对象进行抽象和封装, 视图将各种数据以界面的形式展现给终端用户, 控制器体现商务逻辑控制整个程序的流程。采用 MVC 技术, 结构清晰, 各功能模块划分明确。擅长美工、外观布局的 Web 开发人员可专门从事 JSP 显示页面的制作工作, 精通 Java 网络原理的软件程序员可以集中精力进行商务逻辑的设计工作, 而各种模型类则由熟悉数据库和数据结构的人员来完成。三个部分的开发工作能同步进行, 系统便于维护和扩展。

纯 Java 的 Java bean、JSP 和 Servlet 虽然实现了 MVC 三层结构。但是, 一个 Servlet 的改变将直接影响到调用它的所有 JSP 页面, 使它们都要作相应的改动。同时, 为了完成显示逻辑, 在 JSP 页面中仍然包含大量的 Java 代码, 降低了页面的可读性。而且, 难以实现代码重用和大粒度的软件构件, 缺乏数据交换的通用性、兼容性。

解决这些问题的最好方法是引入 XML 技术, XML 即扩展标识语言 (eXtensible

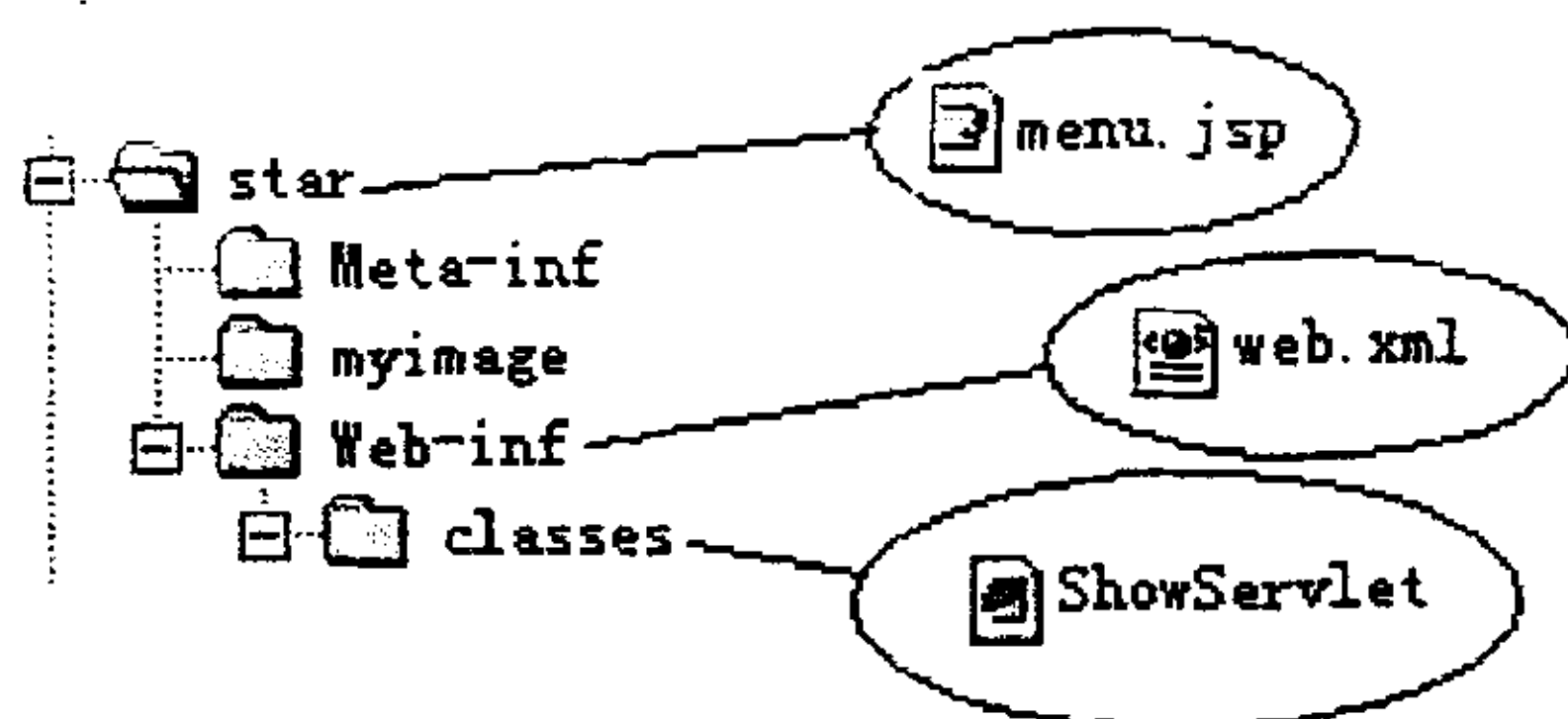


Markup Language), 具有一套统一的数据格式, 非常适于应用程序之间数据的交换, 特别是松耦合的应用程序, 如: 分布式Web系统。XML可以促进应用程序代码的重用, 提高应用程序对需求变化的适应能力。所以, 把XML处理数据方面的跨平台性与JSP处理商务逻辑的跨平台性进行组合将是一种较为理想的结合。下面就以一个具体的JSP三层模型花卉站点来应用XML技术。

用XML技术规划JSP服务器, 这是一个电子商务系统(花卉在线订购网站)采用JSP三层模型结构, 系统平台为Windows 2000, 以Apache作为服务器, 以Tomcat作为JSP引擎, 数据库采用SQL Server 2000。本系统在下列四个方面应用了XML技术。

#### 4.1 XML部署文件映射Servlet动作

在JSP默认的运行目录下的WEB-INF目录下有个web.xml文件, JSP可以使用这个XML文件来映射Servlet动作。如图(4-1)所示:



图(4-1)

默认的运行目录为star, 视图JSP文件都在star目录中, 控制器Servlet类都在classes目录中。如果menu.jsp要调用ShowServlet, 那么在menu.jsp中应有类似语句:

```
<a href="servlet/ShowServlet">1、花卉信息查询</a>
```

这种用名称直接调用的方法使页面代码不便于理解和记忆。而映射的方法是在Web-inf目录下的web.xml文件中加入内容:

```
<servlet>
    <servlet-name>
```



```

        to-show
    </servlet-name>
    <servlet-class>
        ShowServlet    //执行花卉信息显示动作
    </servlet-class>
</servlet>

```

能实现 to-show 到 ShowServlet 的映射,以后要调用 ShowServlet 只需用语句:

```
<a href="servlet/to-show">1、花卉信息查询</a>
```

这样做的好处在于若 ShowServlet 发生变动(改换或更名)时只需在 web.xml 文件中找到上述内容将<servlet-class>元素的内容改为新的名称即可,例如:

(将 ShowServlet 改为 PlayServlet)。而所有调用它的 JSP 页面都不用作任何改动,就能把 to-show 动作改为映射到 PlayServlet。也就是说控制器 Servlet 的更新不会直接影响到视图 JSP 页面,这样做使两部分的设计可以自成体系。

在 web.xml 文件中还映射了几个实现商务逻辑的 Servlet 动作类:

```

to-buy      映射  ExeServlet    //执行花卉购买动作
to-form     映射  FormServlet   //执行购买表报产生动作
to-return   映射  ReServlet     //执行返回主页动作
to-browse   映射  BrowServlet   //执行用户相信浏览动作
to-mima     映射  MimaServlet    //执行用户密码验证动作
to-logout   映射  DelServlet     //执行用户密码注销动作

```

## 4.2 XML 定制标记 封装 Java 代码

三层模型虽然把控制逻辑封装在 Servlet 中,在一定程度上减轻了 JSP 页面的份量,也简化了 JSP 的设计,但是为了实现显示逻辑,JSP 页面中仍然含有大量的 Java 代码。解决这个问题的一個方法是使用 XML 定制标记,在 JSP 中使用 XML 定制标记表示的是特殊域的功能。按照 JSP1.1 规范:执行时,JSP 页面的实现过程将使用可用的 Tag 实例……接着停止使用 Tag 实例……然后释放 Tag 实例。javax.servlet.jsp.tagext 包提供了实现定制标记需要的接口和类,几乎

所有的标记处理程序均通过扩展 TagSupport 或 BodyTagSupport 类来实现 Tag 接口。下面具体说明这一过程。

若 JSP 页面通过 session 从 Javabean 模型中获得单价:pricebuy 购买数量:sum 然后计算出总价 (float 型)  $sumprice = pricebuy * sum$ , 由于是货币在显示时需保留两位小数且应转化为字符串。并且所有包含货币显示的页面都要这么去做, 因此可把实现这种功能的 Java 代码封装在 toStringTag.java 标记类中, 部分语句如下:

```

.....

public class toStringTag extends TagSupport
{
    private float value; //用来存放待加工的 float 型变量

    public void setNum(float num)
    {
        this.value = num; //获得标记属性值传给 value
    }

    public int doStartTag() throws JspException
    {
        ..... //这里是把 float 型的 value 保留两位小数且转化为 String 型
        并存储在 sback 中的 Java 代码。 .....

        try { pageContext.getOut().print(sback); }

        //将结果输出到 JSP 文件中

        catch(java.io.IOException ex) {
            throw new JspException(ex.getMessage());
        } //捕获输出异常

        return SKIP_BODY;

        //继续执行本标记后面的 JSP 页面内容
    }
}

```

注意: 标记类首先执行属性设置函数 setNum(float num), 获得要转化的 float 型变量, 然后执行标记开始函数 doStartTag(), 产生需要的 String 型变量并把它输出到页面。

为使用这一定制标记还要对图 (4-2) 所示位置的 web.xml 和 mylib.tld 两个文件进行配置。

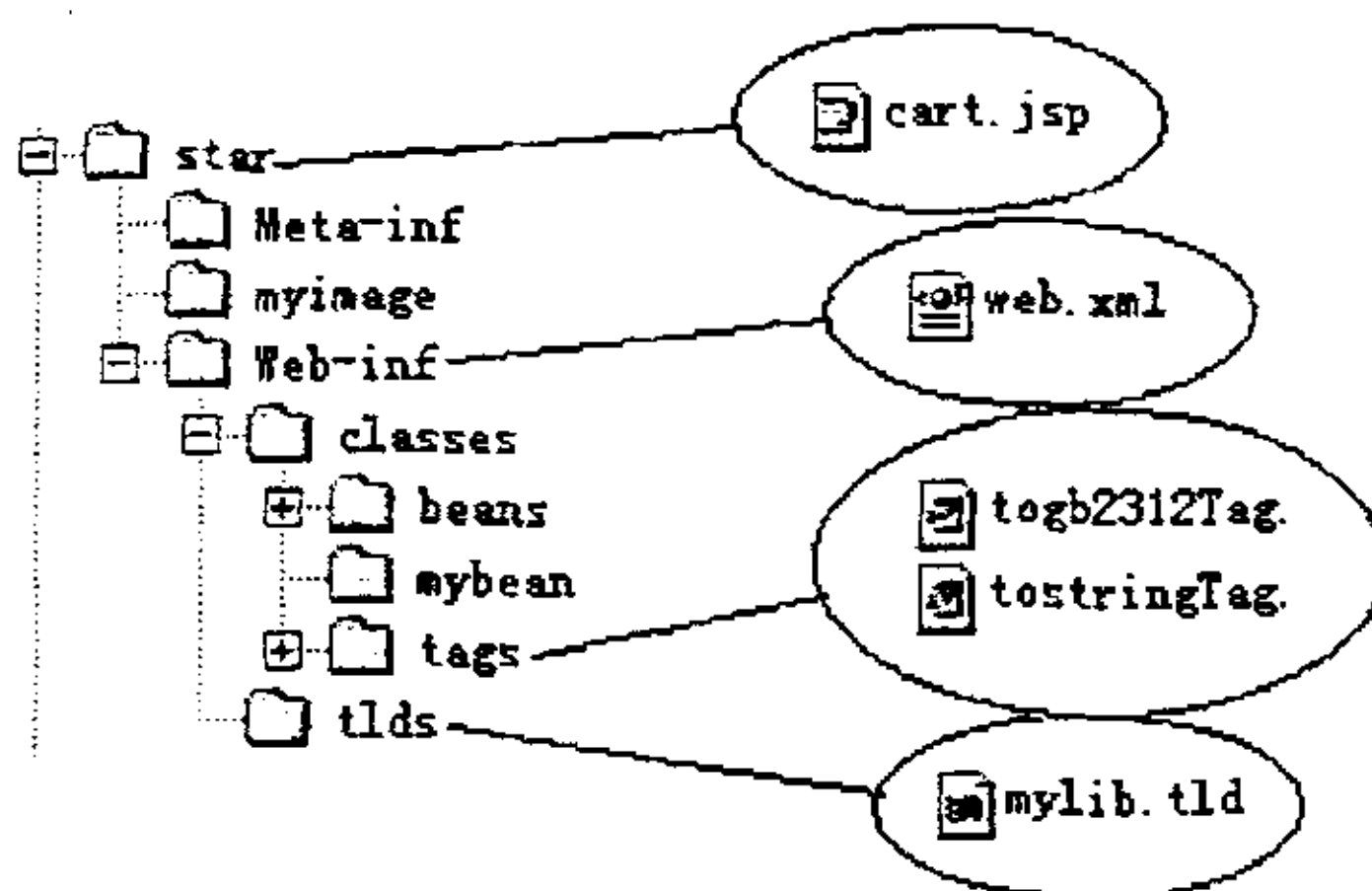


图 (4-2)

在 web.xml 文件中加入内容:

.....//把 mylib 映射为 /WEB-INF/tlds/mylib.tld

<taglib>

    <taglib-uri>mylib</taglib-uri>

<taglib-location>/WEB-INF/tlds/mylib.tld</taglib-location>

</taglib>

.....

在 mylib.tld 文件中加入内容:

.....//把标记 toString 映射到 tags 包下 toStringTag 类

<tag>

<name>toString</name>

<tagclass>tags.toStringTag</tagclass>

<bodycontent>empty</bodycontent>

<attribute>

    <name>num</name>

    <required>true</required>

    <rtexprvalue>true</rtexprvalue>

</attribute>

</tag>

.....

在需要使用标记的 JSP 文件中导入标记库:

```
<%@ taglib uri='mylib' prefix='mylib' %>
```

然后就可以使用标记了。同理，也可以构造一个 `togb2312Tag.java` 标记类来实现 ISO8859\_1 码到中文 GB2312 码的转换。

.....

```
public class togb2312Tag extends TagSupport {
    private String str;
    public void setZhnum(String zhnum)
    {
        this.str = zhnum;
    }
    public int doStartTag() throws JspException {
        String temp_p="", temp="";
        try{
            temp_p=str;
            byte[] temp_t=temp_p.getBytes("GBK");
            temp=new String(temp_t, "ISO8859_1");
            .....
            return SKIP_BODY;
        }
    }
}
```

比如购物车 `cart.jsp` 页面使用标记：

.....//以中文 GB2312 码显示花卉名称

```
<mylib:togb2312 zhnum='<%=mynamebuy %>' />
```

.....//以两位小数且字符串形式显示购买金额

```
<mylib:tostring num='<%=pricebuy%>' />
```

效果如图 (4-3) 所示：上未使用标记，下使用了标记

1. ??? 9份 117.450005 元	上
1. 康乃馨 9份 117.45 元	下

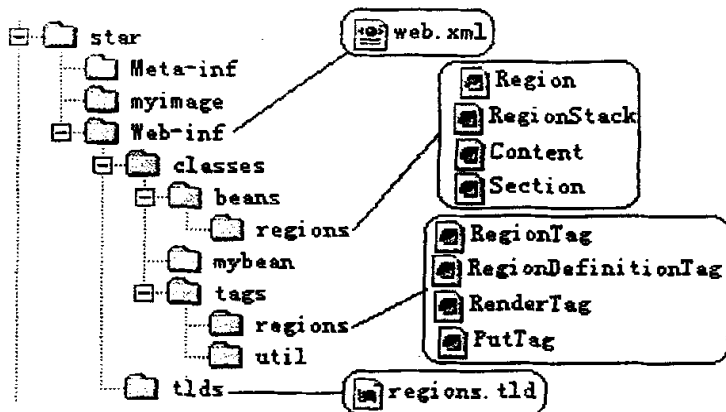
图 (4-3)

当然若不用标记也能达到图 (4-3) 下面的效果，而使用标记的优势在于：更多的 Java 代码从 JSP 页面中分离出来，使 JSP 文件条理更加清晰，同时，把一些通用的功能封装为标记类，所有页面都能使用，大大增强了软件代码的重用性。

### 4.3 XML 模板布局 装配 JSP 文件

XML 在 JSP 中的一个重要应用就是模板布局，其思想是：按不同的显示功能及要求来创建和划分 JSP 文件，使得一个 JSP 文件实现一个特定的显示功能，如：greet.jsp 显示欢迎信息，bigmenu.jsp 显示功能菜单，log.jsp 显示注册界面，foot.jsp 显示返回菜单等。这些功能单一的 JSP 文件可以看成是一个个的构件，再由这些构件来组成功能完整的页面，如：主页面 Star.jsp 应包含欢迎信息 greet.jsp、功能菜单 bigmenu.jsp 等，购物页面 BuyHTML.jsp 应包含物品列表 viewHTML.jsp、购物车 cart.jsp、返回菜单 foot.jsp 等。

这样每个小页面就是一个大粒度的构件，可以装配到任何需要它的组合页面中，若某一显示功能需要修改，则只需修改实现这一功能的 JSP 小页面，其它页面都不用改动。同样，若一组合页面需要增删功能，则只需增加或减少相应的 JSP 小页面即可。如此设计就要求组合页面能合理的布局它所包含的小页面，仅仅用 include 语句显然是很不够的，这里采用 XML 模板布局。如图（4-4）所示：



图（4-4）

首先，在 tags/regions 目录下创建实现模板的标记类：RegionTag、RegionDefinitionTag、RenderTag、PutTag，（其中 RegionTag 是基类），在 beans/regions 目录下创建这些标记类所使用的 Java bean：Region、RegionStack、Content 和 Section。并建立标记类的引导文件 regions.tld，在 web.xml 文件中添加相应的映射内容，使得 JSP 中的调用能映射到标记类中。这些内容与上一节（4.2 节）相似，这里不再描述。

其次, 创建组合页面布局使用的模板 module.jsp 文件, 主要代码如下:

```
<body>
<table align="center">
    <tr> <td> <region:render section='sidebar' /> </td>
    <td> <table><tr><td>
<region:render section='header' />
</td></tr> <tr><td>
<region:render section='content' />
</td></tr> <tr><td>
<region:render section='footer' />
</td></tr> </table>
    </td> </tr> </table>
</body>
```

从文件中可以看出这个模板包含四个部分: **侧面** sidebar、**头部** header、**内容** content 和 **脚部** footer, 很多组合页面都能使用这一模板。比如: 主页面 Star.jsp 代码:

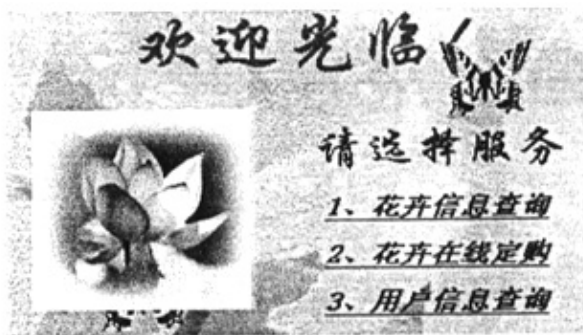
```
<%@ taglib uri='regions' prefix='region' %>
<region:render template='module.jsp'>
    <region:put section='header' content='greet.jsp' />
    <region:put section='content' content='bigmenu.jsp' />
</region:render>
```

第一行导入标记库 regions, 第二行使用布局模板 module.jsp, 第三行欢迎 greet.jsp 放在**头部**, 第四行菜单 bigmenu.jsp 放在**内容**, 第五行标记结束。又如: 购物页面 BuyHTML.jsp 代码:

```
<%@ taglib uri='regions' prefix='region' %>
<region:render template='module.jsp'>
    <region:put section='sidebar' content='Bigcart.jsp' />
    <region:put section='content' content='viewHTML.jsp' />
    <region:put section='footer' content='foot.jsp' />
</region:render>
```

所不同的地方是，购物车 cart.jsp 放在**侧面**，物品列表 viewHTML.jsp 放在**内容**，返回菜单 foot.jsp 放在**脚部**。从这两个组合页面可以看到一个共同的特点，那就是，实现了如：购物车 cart.jsp、菜单 bigmenu.jsp 等 JSP 形式的大粒度构件，可以在任何需要它们的组合页面中灵活装配。整个页面完全符合 XML 格式，结构清晰，一目了然，容易装配和维护。

第三，还可以创建多块模板来实现不同的布局风格（不同的摆放位置、不同的背景、不同的广告甚至不同的 JavaScript 动画特效等）。例如刚才的主页面 Star.jsp，第二行使用布局模板 module.jsp 时显示效果如图（4-5）。



图（4-5）

还是主页面 Star.jsp，第二行使用布局模板改为 module2.jsp 时显示效果如图（4-6）。



图（4-6）

在布局方面能实现大粒度的构件，这一点是很有意义的，由于构件的重用和方便的组合使得基于网络的程序开发和维护变得相对容易得多。比如用户初次进入登录验证密码输入的页面 logHTML.jsp 由三个小的页面组成它们是：欢迎信



息、密码输入框、返回主页按钮。

```

.....

<region:put section='header' content='greet.jsp' /> //欢迎信息
<region:put section='content' content='log.jsp' /> //密码输入框
<region:put section='footer' content='foot.jsp' /> //返回主页按钮
.....

```

而当密码输入错误后转到提示密码错误同时让用户再次输入密码的页面 log2HTML.jsp 也由三个小的页面组成它们是：密码错误提示、密码输入框、返回主页按钮。

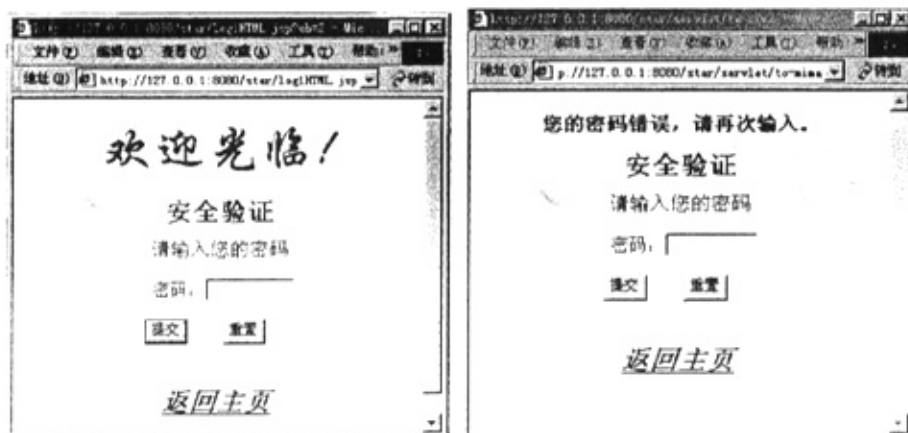
```

.....

<region:put section='header' content='merror.jsp' /> //密码错误提示
<region:put section='content' content='log.jsp' /> //密码输入框
<region:put section='footer' content='foot.jsp' /> //返回主页按钮
.....

```

这样在两个不同的大页面中却使用了相同的构件：密码输入框 log.jsp，返回主页按钮 foot.jsp。从而实现了页面构件的重用。其显示效果如图（4-7）所示：



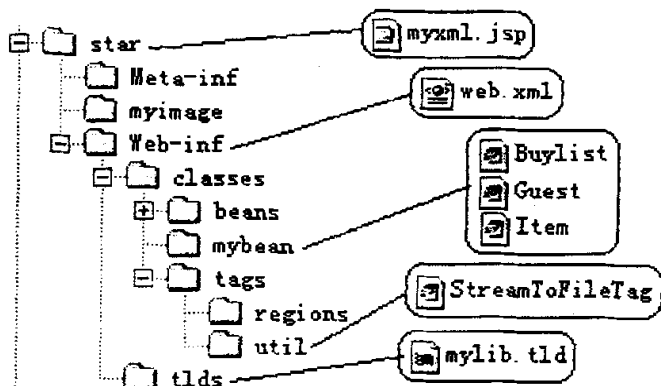
图（4-7）

可见，使用 XML 模板布局来装配 JSP 文件，使网站灵活布局以满足不断变化的需要。

## 4.4 XML 报表输出 统一 Data 格式

一个电子商务网站，总要产生一些数据报表，以便管理部门作为分析资料或送货公司作为订单依据来处理。数据信息在网上最通用的格式是 XML，因此，本系统中将产生 XML 形式的客户订货报表文件。

下面就用 XML 定制标记来实现 XML 形式的报表文件，如图（4-8）所示：



图（4-8）

首先，在目录 `star\WEB-INF\classes\tags\util\` 下创建报表的数据模型三个 Java bean，它们是 `Buylist`（表示报表单）、`Guest`（表示客户）和 `Item`（表示购买项目）。一个报表文件包含一个报表单，一个报表单可以包含多个客户，而每个客户又可以有多个购买项目。所以，`Buylist` 中有一个矢量 `guests` 用来装多个 `Guest`，`Guest` 中有一个矢量 `items` 用来装多个 `Item`。而 `Item` 封装了花卉名称、单价、数量、定购日期等信息。

然后，在目录 `star\WEB-INF\classes\tags\util\` 下创建能产生 XML 数据报表的主体标记类 `StreamToFileTag` 其主要代码如下：

```

public class StreamToFileTag extends BodyTagSupport {
    private String filename; // filename 属性用来保存文件名
    public void setFile(String filename) { //设置文件名
        this.filename = filename; }
    public int doAfterBody() throws JspException {
        try { FileWriter writer=
            new FileWriter(new File(filename)); //建立写文件流

```

```

writer.write(bodyContent.getString().trim()); //执行写入
writer.close(); } //关闭写文件流
catch(java.io.IOException e) { //捕获异常
    throw new JspException(e.getMessage()); }
return SKIP_BODY; } } //继续执行后面的 JSP 内容

```

执行过程是：从标记获得属性 `setFile()`（要生成的文件名），然后对标记主体求值（产生文件的内容），然后是主体后操作 `doAfterBody()`（把内容写入文件）。

要使用这一标记同前两节一样，需对 `web.xml` 和 `mylib.tld` 文件实施相应的布置。最后使用这一标记的 `myxml.jsp` 文件代码如下：

```

<mylib:streamToFile file='C:/tomcat/webapps/star/list.xml'>
// streamToFile 标记开始，其中 file 是标记的属性
<?xml version="1.0" encoding="gb2312"?> //产生 XML 文件头
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<list> //根元素 list 表单开始
    <% while(it.hasNext()) { %> //对表单循环产生客户
        <% guest = (mybean.Guest)it.next(); %>
        <guest>
            .....取出 guest 中的数据.....
            <% itt=guest.getItems();%>
            <% while(itt.hasNext()) { %> //对客户循环项目
                <% item = (mybean.Item)itt.next(); %>
                <goods>
                    .....取出 item 中的数据.....
                </goods>
                <% } %>
            </guest>
        <% } %>
    </list> //根元素结束
</mylib:streamToFile> //标记结束

```

此 JSP 的运行将产生一个名为 `list.xml` 的报表文件，下面是某一次运行时

产生的 list.xml 文件内容:

```
<?xml version="1.0" encoding="gb2312"?>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<list>
  <guest>
    <name>王劲松</name>
    <sumcost>42.20</sumcost>
    <goods>
      <hname>红玫瑰</hname>
      <Fcost>15.80</Fcost>
      <Fnum>1</Fnum>
      <Fdate>2003 年 4 月 15 日</Fdate>
    </goods>
    <goods>
      <hname>白玫瑰</hname>
      <Fcost>26.40</Fcost>
      <Fnum>3</Fnum>
      <Fdate>2003 年 4 月 15 日</Fdate>
    </goods>
  </guest>
  <guest>
    <name>令狐冲</name>
    <sumcost>65.25</sumcost>
    <goods>
      <hname>康乃馨</hname>
      <Fcost>65.25</Fcost>
      <Fnum>5</Fnum>
      <Fdate>2003 年 4 月 15 日</Fdate>
    </goods>
  </guest>
```

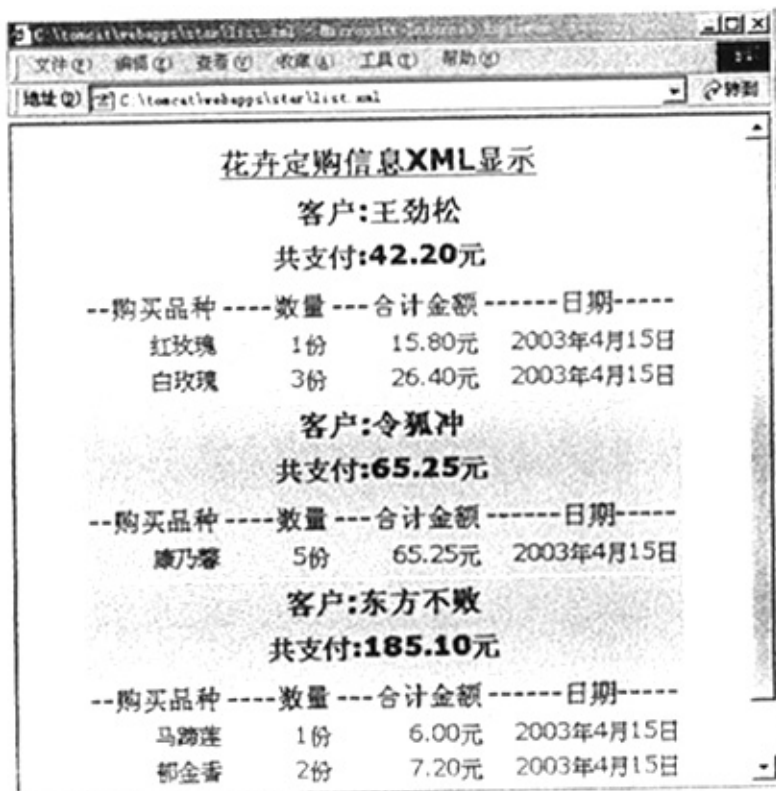
```

<guest>
    <name>东方不败</name>
    .....
</guest>
.....

</list>

```

若将此文件配上相应的 xsl 文件, 就会有良好的报表效果, 图 (4-9) 是使用 style.xsl 式样时的效果:



花卉定购信息XML显示			
客户:王劲松			
共支付:42.20元			
--购买品种----	数量	合计金额	-----日期-----
红玫瑰	1份	15.80元	2003年4月15日
白玫瑰	3份	26.40元	2003年4月15日
客户:令狐冲			
共支付:65.25元			
--购买品种----	数量	合计金额	-----日期-----
康乃馨	5份	65.25元	2003年4月15日
客户:东方不败			
共支付:185.10元			
--购买品种----	数量	合计金额	-----日期-----
马蹄莲	1份	6.00元	2003年4月15日
郁金香	2份	7.20元	2003年4月15日

图 (4-9)

在 JSP 三层模型中使用 XML 定制标记进行布局, 能使网站页面结构清晰, 形成良好的文件格式, 易于理解, 易于维护。能大大提高程序的代码重用, 创建大粒度的构件, 灵活装配, 灵活使用。整个系统的组装就像一台机器上组装零件一样直观和方便。

JSP 的语言基础是 Java, 由于其良好的平台兼容性, 使得它能实现 Internet

网，无线电讯网，以及将来的家用电器设备网的编程。而 XML 来源于 SGML，其数据的自我描述性，在表示数据方面有着突出优势，已逐渐成为 Internet 网上的通用格式。JSP 的程序跨平台性与 XML 的数据跨平台性相结合是开发跨平台网络系统的最佳组合。能充分满足当前不断发展中的网络编程的需要，具有广阔的前景。

三层模型，两级用户，一块模板，Java 实现。

## 第五章 总 结

本系统是以一个简单的花卉电子商务网站为示范，说明电子商务信息的技术及其实现。

第一：系统采用了最能体现跨平台的 Java 体系，它们是 J2EE、J2SE 和 J2ME。由 JSP 编程起步，构建真正意义上的 MVC 三层模型。视图 V：针对用户而开发，用 HTML、JavaScript、JavaApplet（甚至 Flash 等）与 JSP 相结合，界面精美、表现生动、控制灵活的页面来与用户交互。模型 M：以 JavaBean 的形式存在，用于完成对后台数据库的操作和封装程序中的数据结构，使得各项操作完整而规范，系统便于管理、维护和扩展。控制器 C：由几个实现特定功能的 Java Servlet 组成，实现商务逻辑和计算，调用模型 JavaBean 来完成操作，控制着程序和页面的转向。

第二：运用 J2ME 中的 MIDP 编程，移动用户端运行 MIDlet 程序，来达到移动设备访问 Internet 网站的目的，MIDlet 直接通过 HTTP 协议与 Servlet 对话，这样 MIDlet 就构成了移动用户的界面即视图 V，而模型 M 和控制器 C 仍然与 IE 浏览器访问的相同。所以 IE 浏览器和手机浏览器不同的用户的需求仍由相同的数据库操作 sql JavaBean 来完成，从而保证数据库事务处理的一致性。而且 J2ME 属于 Java 语言体系，与服务器的 Servlet 兼容，采用 Unicode 字符集，对中文支持良好，对彩色图片显示支持良好。以此，实现了针对两级用户的移动电子商务系统。

第三：为更加便于网站的维护和扩展，为使最终信息的格式达到统一，采用了 XML 技术。XML 部署文件 映射 Servlet 动作、XML 定制标记 封装 Java 代码、XML 模板布局 装配 JSP 文件、XML 报表输出 统一 Data 格式。优点是：代码重用，构件的粒度增大，数据与格式分离，显示逻辑与商务控制逻辑分离，各功能模块之间的内聚度降低，耦合度提高。JSP 的程序跨平台性与 XML 的数据跨平台性形成开发跨平台网络系统的最佳组合。



## 致 谢

此毕业设计及论文是在导师王健副教授的精心指导和帮助下完成，在导师的关怀与培养下我受益良多，首先向王老师表示衷心的感谢。王老师渊博的学识、严谨的学风、敏捷的思路、无私的育人、都将是我永远学习的榜样。王老师办事认真果断，工作作风扎实，为人随和宽厚，使我终生难忘。

同窗三年，匆匆而过，回首往事，感慨无限。在此要感谢和我并肩学习、携手奋斗的黎才茂、关丽霞、陈万华等同学以及所有帮助过我的人，是他们在生活上、精神上和学业上的帮助，使我克服了重重困难。

最后要感谢我的妻子张倩，一个贤惠的女人，离开了她的支持我可不能完成学业。

## 参考文献

- 1、 王劲松, 王 健 《基于两种浏览器的 MVC》 微机发展 (中国计算机学会会刊) 2003 年第五期录用。
- 2、 王劲松, 王 健 《XML 部署 JSP 应用》 计算机系统应用 (中国计算机学会会刊) 2003 年第十期录用。
- 3、 (美) Aaron Skonnard、Martin Gudgin 编著 牛韬 英宇 译《XML 精要》人民邮电出版社 2002、10
- 4、 (美) David M. Geary 编著 贺民 译,《advanced JavaServerPages》科学出版社 2002、9
- 5、 (美) Mark Wutka 编著 程显华 译《JSP 和 Servlet 程序设计使用专辑》机械工业出版社 2002、3
- 6、 邵维忠、杨芙清 编著《面向对象的系统分析》清华大学出版社 1998、12
- 7、 (美) P. J. Louis 编著《移动电子商务速成教程》人民邮电出版社 2002、2
- 8、 王 森 编著《Java 手机程序设计 入门与运用》中国铁道出版社 2003、1
- 9、 卢 军 编著《J2ME 应用程序开发 手机、PDA 程序开发捷径》中国铁道出版社 2002、9

基于XML和JSP的移动电子商务技术及实现

作者：[王劲松](#)  
学位授予单位：[东南大学](#)

本文读者也读过(1条)

1. [吴淑雷](#) [基于XML题库系统的设计与研究](#)[学位论文]2005

引用本文格式：[王劲松](#) [基于XML和JSP的移动电子商务技术及实现](#)[学位论文]硕士 2004