分 类 号：TP393.09　　单位代码：10019
密　　级：　　学　　号：S02734

中国农业大学

学位论文

# 基于 Struts 框架的企业办公系统的设计与实现

# DESIGN AND IMPLEMENTATION OF ENTERPRISE OFFICE AUTOMATION SYSTEM BASED ON THE STRUTS FRAMEWORK

研　　究　　生：王　欢
指　导　教　师：孙 龙 清　副教授
合 作 指 导 教 师：
申请学位门类级别：工 学 硕 士
专　业　名　称：计算机应用技术
研　究　方　向：计算机网络技术应用
所　在　学　院：信息与电气工程学院

2005 年 5 月

# 独 创 性 声 明

本人声明所呈交的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，论文中不包含其它人已经发表或撰写过的研究成果，也不包含为获得中国农业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：　　时间：　　年　　月　　日

# 关于论文使用授权的说明

本人完全了解中国农业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件和磁盘，允许论文被查阅和借阅，可以用影印、缩印或扫描等复制手段保存、汇编学位论文。同意中国农业大学可以用不同方式在不同媒体上发表、传播学位论文的全部或部分内容。

**（保密的学位论文在解密后应遵守此协议）**

研究生签名：　　时间：　　年　　月　　日

导师签名：　　时间：　　年　　月　　日

# 摘要

框架是特定应用领域中的面向对象软件重用技术，是由特定应用领域的软件体系结构所决定的软件构架。应用框架可以降低软件开发的成本、提高软件质量。随着面向对象技术、Web 技术和 J2EE 技术的发展，软件框架技术在 Web 开发中得到了广泛应用。Struts 作为一个开源的 Web 层框架，是 MVC 及 J2EE 若干核心模式的标准实现，方便了代码的编写和管理，适用于大型、复杂的 Web 应用开发。本文即以 Office Mate 企业办公系统的开发作为背景展开讨论。

论文分为四部分。第一部分介绍了课题的研究背景、企业办公系统的发展历程以及国内、外的研发现状，从而引出 Office Mate 企业办公系统的需求。第二部分分析了 Web 层应用程序的体系结构、MVC 设计模式及实现；着重对基于 Struts 框架的 Web 应用系统的工作原理和主要开发技术进行了探讨。第三部分介绍了 Office Mate 企业办公系统的设计和实现，详细探讨了 Struts 框架在系统实现中的运用，以及系统实现过程中的关键问题和解决方案。第四部分总结了课题研究和系统建设过程中主要工作和未来工作展望。

**关键词**：J2EE，MVC，Struts 框架，企业办公系统

# ABSTRACT

Framework is application of Object-Oriented Software Reuse technology in specific domains and a software architecture decided by specific domains. Frameworks contribute to save cost and enhance software quality during developing process. With sophisticated technologies like Object-Oriented, Web and J2EE being applied, software framework technology is used extensively in Web application development. As a best popular open-source framework, Struts Framework is a standard implementation of MVC and some core J2EE design patterns, which greatly facilitates writing and managing codes. Struts Framework is especially suitable to develop large-size and complex Web applications. The thesis discusses application and research of Struts Framework with Office Mate Enterprise Office Automation System.

The thesis consists of four parts. The first part introduces research background, developing history of Office Automation System and research status in quo in order to present general requirements of Office Mate Enterprise Office Automation System. The second part focuses on holistic knowledge body of web applications, like web application architecture, MVC design pattern and its J2EE implementation as well as web tier frameworks, emphasizes architecture, working principles and crucial technology of Struts Framework. The third part describes system level analysis and design of Office Mate System as well as implementation procedure of the system based on the Struts Framework, discusses solutions to some key issues during system building. Last but not least, the thesis synopsizes assignments, issues, and original ideas in research and system construction as well as designs future research roadmaps.

**Keywords:** J2EE，MVC，Struts Framework，Enterprise Office Automation System

# 目录

# 插图和附表目录

# 第一章 绪论

办公自动化[1][7][8]（Office Automation，简称OA）是将现代化办公和计算机网络功能结合起来的一种新型办公方式，利用计算机、网络、通信等技术，收集、处理、存储和传输信息以提高办公效率和辅助决策，形成高效和智能的办公环境，使办公自动化、网络化、无纸化，实现协同工作。OA是当前新技术革命中一个非常活跃、具有很强生命力的技术应用领域，是信息化社会的产物。

随着软件规模和复杂度的不断升级，尤其是面向对象技术、计算机网络技术、通信技术和多媒体技术的发展，软件设计方法和开发环境产生了巨大变化，人们对软件的特性和基础结构有了更深刻的认识。在设计理论和方法方面，系统总体结构设计和规格说明的重要性已远远超过算法和数据结构。最近几年，我国办公自动化的研究和实践呈现出迅猛的发展势头，已经成为政府机关、工矿企业、商业、高等院校和研究机构等单位的必备系统。

为了快速反应瞬息万变的市场行情，现代企业一般具备如下的特点[3]：随着市场的变化及时修正计划执行中各种偏差；迅速有效地解决好本企业内部组织间的冲突；信息流畅、数据共享，以真实的信息准确地支持上层决策。计算机网络和 Web 技术的迅速发展为现代企业提供了功能强大的信息处理平台，使得 OA 系统的体系结构、计算模式等必须与 Internet 技术相融合。

## 1.1 课题研究背景

国内很多企业的日常办公还停留在手工操作上，大量信息得不到共享，为信息的管理、分类、索引和使用带来相当大的困难。信息的大量冗余使得难以挖掘出有效数据以支持决策，增加了管理的困难。尽管已有不少政府机关单位、大型企业开发了办公自动化系统，但大多采用以关系型数据库为基础的 MIS 系统或是在老版本 Lotus Domino/Notes 平台上开发的办公自动化系统。由于当时网络技术和信息管理软件的局限，已经不能满足现代企业的需要，如集成化程度低、资源共享不充分、缺乏韧性、适应性差等。并且这些办公系统大多仅仅能提供文档的存储、管理、查询，远远不能适应网络化信息时代的要求。现代企业需要一个高度自动化的办公系统，该系统应该能够显著提高办公效率、降低成本、充分利用资源、加快工作流程以及适应环境变化等。

第一代《Office Mate 企业办公系统》采用 Windows NT、ASP、SQL Server 7.0 作为开发平台。随着现代企业需求的不断变化，客户迫切需要一套信息采集、处理、传递与共享高度自动化的办公系统，提高系统的功能性、可伸缩性、可用性、可扩展性以及可靠性。为此，要求开发一套具备跨平台、开放性、安全性高等特点的现代企业办公系统，并将企业的传统业务整合成面向 Web 的应用。

## 1.2 办公自动化系统的现状及分析

### 1.2.1 办公自动化系统的发展

自 20 世纪 80 年代中期至今，国内办公自动化系统经历了三个阶段。80 年代中期的第一代办公系统以个人电脑、办公套件为主要标志，实现了数据统计和文档电子化，完成了信息载体从纸

质方式向电子方式的飞跃。从 20 世纪 90 年代中期开始，以 Lotus Domino/Notes 为代表的工作流群件技术面世，并伴随网络通信技术的发展，第二代办公自动化系统以网络技术和协同工作技术为主要特征，实现了收、发文从传统的手工方式向工作流自动化方式飞跃。21 世纪是知识经济的时代，知识已经成为经济增长、社会发展及企业成长的关键性资源，最大限度地掌握和利用知识日益成为各单位信息化建设的核心。在这种背景下，以知识管理为核心的第三代办公自动化系统应运而生。

### 1.2.2 办公自动化系统的研究现状

目前国内的大部分研究还处于如何利用第二代办公自动化系统、开发以工作流自动化为核心的办公自动化系统上，大多数采用 Lotus Domino/Notes 开发平台。也有少数公司、大学和个人进行第三代办公自动化系统的研究。目前，基于 Lotus Domino/Notes 平台的第二代办公自动化系统应用在国内的一些单位、企业和学校中，如农业银行四川省分行办公自动化系统、宝钢办公自动化系统等[5]。基于 J2EE 体系结构的网上办公自动化系统也有很多，如北京市对外经济贸易委员会的网上审批系统、山西省工商办公自动化系统和北京市公安部网上办公自动化系统等。

国外的办公自动化研究一直处于领先地位，第三代办公自动化系统得到了广泛支持。IBM 公司的 Lotus Domino/Notes 产品是用于开发办公自动化系统的主流平台[6]，其它软件生产厂商也纷纷开发自己的知识管理平台。

### 1.2.3 办公自动化系统实现技术的选择

办公自动化系统采用的实现技术总结起来有三大类：一是基于 Lotus Domino/Notes 平台二次开发的系统。企业必须购买正版的许可证书，成本和系统维护代价昂贵，并且需要专业 Notes 程序员来进行各种设置和日常维护；第二种是由 MIS 系统转变而来的，把日程表、会议、考勤、人事等本应独立的模块组合在一起，再加上统一的用户认证和菜单界面调用，这种系统的缺点是各模块之间无法实现互动；第三种是采用 J2EE 等分布式技术实现的系统，是近几年迅速增长的开发平台。

Lotus Domino/Notes 具有独特的体系结构，是一个集数据库、邮件系统、动态 Web 信息发布、可视化集成开发环境于一体的基础平台，适合处理办公协作流程中产生的非结构化文档信息，并可以利用灵活的邮件机制在用户、部门之间传递文档。但是 Lotus Domino/Notes 开发的 OA 系统需要安装客户端软件，存在维修、病毒等各种隐患。采用 B/S 结构，程序主要安装在服务器端，员工只需要通过浏览器就可以使用各种功能。服务器的损坏率是极低的，这样，企业的维护成本也会降低。C/S 结构中的数据传送通过邮件服务中转，例如，传递一个通知，实际上是将信息复制了 n 份，产生了大量的数据传送，给网络带来巨大压力。同时可能造成病毒在局域网内的快速传播，安全方面存在潜在的隐患。B/S 结构[2][4]是虚拟数据传送，传送的只是一个标识，具体内容储存在服务器上。另外，尽管在 Lotus Domino/Notes 平台上开发速度快、代码量少，但是数据库被封装在开发平台之内，灵活性差，难以满足系统的自适应需求。

J2EE 规范定义的 B/S 结构具有独特的优势。采用 J2EE 技术实现的办公自动化系统，业务逻辑和业务数据分离，应用服务器负责事务管理、生命周期管理，开发人员只关注需要实现的业务

逻辑，而无需了解底层的通信和管理细节。提高了开发效率和系统的稳定性，降低了系统出错率，易于与现有系统无缝集成。

## 1.3 本文的主要研究内容

在设计开发 Office Mate 企业办公系统时，综合考虑了代码重用、系统管理、操作简便以及应用安全和后期维护等方面的要求，采用了 J2EE 平台开发系统，全面贯彻 MVC 设计模式。并将系统的 Web 层构建于 Struts 框架之上，不再采用传统 JSP 文件简单堆砌的方式。Struts 框架是基于 Servlet、JSP 以及标签库等技术的一个实现良好的 Web 层框架，利用它可以开发大型、复杂的 Web 应用系统，并简化代码开发和管理，提高系统的重用性。

本文以“基于 Struts 框架的企业办公系统设计与实现”为题，主要包括以下几个方面：

- 分析了 Web 应用程序的体系结构。介绍了 Web 应用程序的多层结构、J2EE 平台的基本结构、MVC 设计模式及其基于 J2EE 平台的实现方式。
- 对几种 Web 层框架进行了比较分析。着重阐述了 Struts 框架的概念、体系结构和工作原理。
- 详细阐述了 Office Mate 企业办公系统的分析和设计。介绍了系统的设计思想和设计目标、系统网络拓扑结构和软件运行环境，系统软件总体设计和日常办公、日常业务、综合信息、系统管理四大模块的功能描述以及一些与课题研究相关子模块的类设计和数据库设计。
- 详细描述了 Struts 框架在 Office Mate 企业办公系统实现中的运用。以用户管理模块为例，从 Struts 框架的体系结构出发，详细阐述了配置、视图、模型和控制器的实现过程，以及系统实现过程中的一些关键问题和解决方案。

## 1.4 本章小结

本章主要介绍了课题的背景，国内外办公自动化系统的发展和研究现状，以及现阶段采用的主要开发技术，最后对本文的主要研究内容做了总结。

# 第二章 Web 应用程序的体系结构

## 2.1 软件体系结构与模式

### 2.1.1 软件体系结构

软件体系结构（software architecture）[13]为软件系统提供了一个结构、行为和属性的高级抽象，由系统元素描述、元素间的相互作用、指导元素集成的模式以及模式的约束组成。软件体系结构指定了系统的组织结构和拓扑结构，显示了系统需求和构成系统的元素之间的对应关系，提供了一些设计的基本原理。软件体系结构是设计抽象的进一步发展，满足了更好地理解软件系统，更方便地开发更大、更复杂软件系统的需要。

### 2.1.2 模式

模式[9]（pattern）描述了一个不断重复发生的问题和解决方案的核心。不同模式处于不同的抽象层次。在软件工程领域，模式进一步细化为三类[12]：体系结构模式、设计模式、惯用法。模式源于实践，有助于利用软件工程师的集体经验构建软件，是构造高质量软件体系结构的重要方法。使用模式构造的软件系统，支持使用已定义的软件元素构造软件，有助于建立一个复杂、异构的软件体系结构以及管理软件的复杂度。

### 2.1.3 软件体系结构模式

体系结构模式（architecture pattern）是最高等级的模式，表示软件系统的基本结构化组织图式和组织方案。每个体系结构模式处理一个软件系统的设计或实现中一种特殊的重复出现的问题，适用于粗粒度设计的开始阶段。体系结构模式提供了一套预定义的子系统，规定子系统的职责和子系统之间的规则。设计模式（design pattern）[42]提供一个用于细化软件系统的子系统或组件。设计模式是中等规模的模式，比体系结构模式小，独立于特定编程语言，适用于粗粒度设计的结束阶段以及需要细化或扩展软件系统的基本体系结构时。惯用法（idiom）是底层模式，具体针对一种编程语言的模式，描述如何使用给定语言的特征来实现组件的特殊方面或它们之间的关系，适用于实现阶段。

软件体系结构设计的核心问题在于能否使用重复的体系结构模式，达到体系结构级的软件重用。体系结构模式主要有 8 种[11]：层（layers）、管道和过滤器（pipes and filters）、黑板（blackboard）、代理者（broker）、模型－视图－控制器（Model-View-Controller，简称 MVC）、表示－抽象－控制（Presentation-Abstraction-Control，简称 PAC）、微核（microkernel）、映像（reflection）。面向模式的软件体系结构把软件系统看成是各种模式的结合。

分层次处理是软件系统问题分析和设计实施的基本方法[10]。在层次方法中，每层由一组相关的类或组件构成，完成特定的功能；层与层之间存在自上而下的依赖关系，即上层组件访问下层组件的 API，而下层组件不依赖上层组件。每层对上层公开 API，实现细节对外透明。恰当地使用层体系结构模式对应用系统分层，可以提高系统的伸缩性、可维护性、可扩展性、可重用性和可管理性。

## 2.2 Web 应用程序的体系结构

Web 应用程序使用 HTTP 作为核心的通信协议，也被称为基于 Web 的应用程序。随着 Web 系统复杂度的提高，应用服务器程序采用多层结构（N-Tier Architecture），进一步对原有三层结构的中间层进行细分。从上而下依次为用户界面层、表示逻辑层、业务层、数据访问层和数据层，如图 2-1 所示[21]。

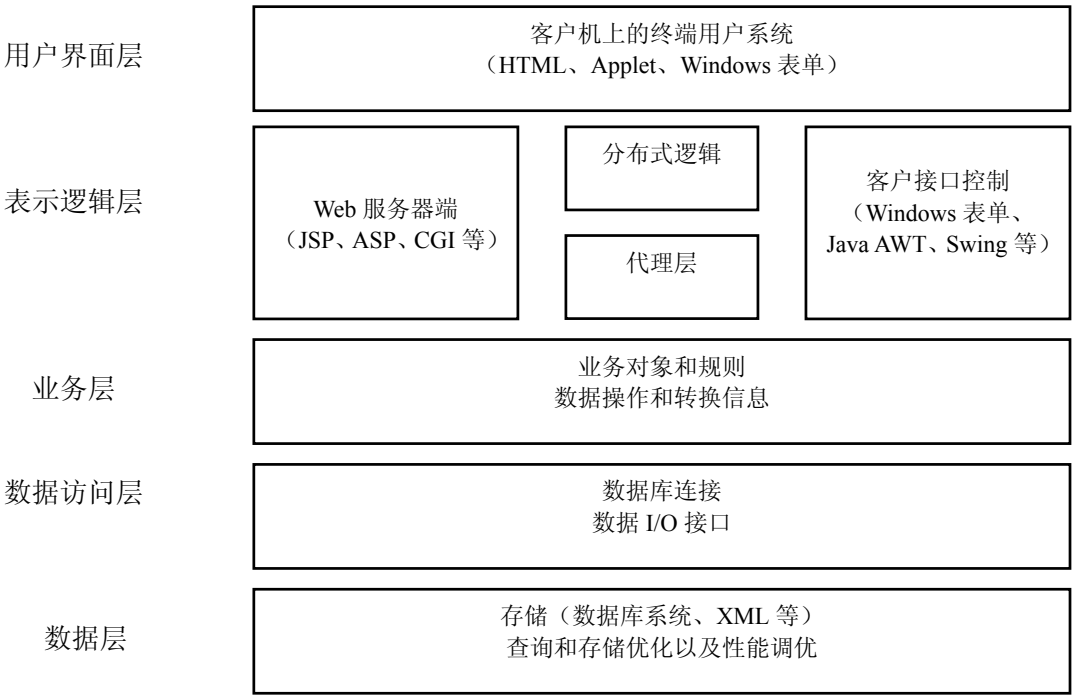



**图 2-1 多层结构的 Web 应用程序**

### 2.2.1 J2EE 体系结构

#### 2.2.1.1 J2EE 技术简介

J2EE（Java 2 Platform Enterprise Edition，简称 J2EE）是 Java 2 平台的企业版，适用于企业级体系结构的开发、部署和管理等，为开发多层次分布式、针对服务器的应用系统提供了统一的技术平台[20]。J2EE 平台已经成为使用最广泛的 Web 应用程序设计技术，主要支持两类软件的开发和应用[21]：一类是做高级信息系统的 Web 应用服务器，一类是在 Web 应用服务器上运行的 Web 应用程序。

J2EE 体系结构扩展了标准的两层结构，提供中间层来满足经济、高可用性、高可靠性、可扩展性的需求。这种结构简化了客户端，主要程序都运行在能保证正常运行的服务器端，系统更加安全可靠，也更容易扩充和移植。在 J2EE 的应用中，软件体系架构模式[60]（如层模式、MVC 模式、多层分布模式等）和设计模式（如截取过滤器、视图帮助器、前端控制器、值对象等）得到广泛使用。

#### 2.2.1.2 J2EE 的四层模型

J2EE 中一个重要的设计原则是应用程序可以低耦合地使用组件组装，在组装或者部署应用

程序时，而不是组件开发时定义或者改变这些组件的相互连接。一个典型的多层 J2EE 应用结构如图所示，包括以下四个层次[19]：客户层、Web 应用层、业务层和企业信息系统层（Enterprise Information System，简称 EIS），如图 2-2 所示。

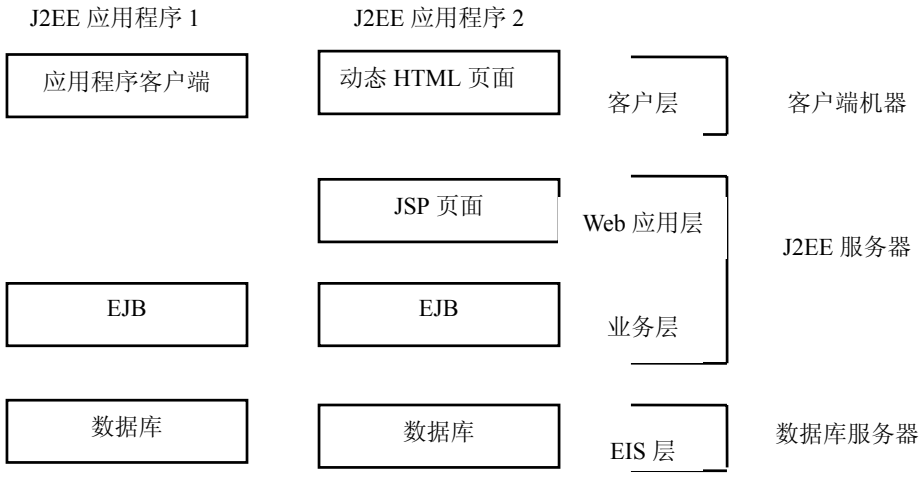



**图 2-2 J2EE 多层应用结构**

- 客户层组件：客户层是用户的使用界面。J2EE 客户端既可以是 Web 客户端，也可以是应用程序客户端。
- Web 应用层组件：可以是 JSP 页面或者 Servlet。Web 层处理用户输入，并把输入发送到运行在业务层上的 EJB 处理。
- 业务层组件：业务层满足业务需求，由运行在业务层上的 EJB 处理。
- 企业信息系统层：企业信息系统层处理企业信息系统软件，包括企业基础建设系统、企业资源计划（Enterprise Resources Planning，简称 ERP）、大型机事务处理、数据库系统和其它的遗留信息系统。

“容器”指应用服务器提供的特定功能的软件模块，是能提供基本功能的底层平台。J2EE 为多层 Web 应用系统提供了容器平台。开发者开发的程序组件在相应的容器内部署运行。

**表 2-1 J2EE 1.4 的容器**

| 容器名称 | 描述 |
|---|---|
| 应用客户端容器 | 负责 Web 应用程序在客户端组件的执行。 |
| Applet 容器 | 特殊的应用客户端容器，负责在 Web 浏览器和 Java 插件上运行 Java Applet 程序。应用客户端容器和 Applet 容器基本对应 Web 应用程序多层结构中的用户接口层。 |
| Web 容器 | 管理 JSP、JSTL 和 Servlet 等 Web 组件的执行，这些组件主要负责程序和 Web 层通信。对应 Web 应用程序多层结构的表示层。 |
| EJB 容器 | 负责 EJB 的运行。主要负责数据处理以及和数据库或其它 Java 程序的通信，对应 Web 应用程序多层结构的业务层和数据访问层。 |

### 2.2.2 J2EE 的核心组件

J2EE 平台由一整套服务、应用程序接口（API）和协议构成，对基于 Web 的多层应用的系统开发提供支持，主要包括以下技术规范[20]：

- JDBC（Java Database Connection，Java 数据库连接技术）
- JNDI（Java Naming and Directory Interface，Java 命名和目录接口）
- EJB（Enterprise JavaBean，企业级 JavaBean）
- RMI（Remote Method Invoke，远程方法调用）
- Java IDL/CORBA（Java Interface Definition Language，Java 接口定义语言）
- JSP（Java Server Pages，Java 服务器页面）
- Java Servlet
- XML（Extensible Markup Language，可扩展标记语言）
- JMS（Java Message Service，Java 消息服务）
- JTA（Java Transaction Architecture，Java 事务框架）
- JTS（Java Transaction Service，Java 事务服务）
- JavaMail（Java 邮件）
- JAF（JavaBeans Activation Framework，JavaBeans 激活框架）

### 2.2.3 MVC 设计模式

#### 2.2.3.1 MVC 设计模式概述

MVC 设计模式（以下简称 MVC）[13]是 Trygve Reenskaug 于 20 世纪 80 年代为编程语言 Smalltalk-80 发明的一种软件设计模式，主要用于处理用户界面开发面临的问题。近几年被推荐为 J2EE 平台的设计模式。

Web 应用系统多采用 B/S 模型的三层或多层结构，如图 2-1 所示，主要包括用户界面设计、业务逻辑设计、数据库设计三个主要方面。Web 应用的传统开发方式将业务逻辑和表现逻辑集成在一起，混合了 HTML 代码与应用程序逻辑，使得界面设计的更改和业务逻辑的更新困难，进而导致系统容易出错、调试困难、开发进展缓慢。理想的 Web 程序要求无论客户是何种类型（如浏览器、Applet 或手机等），数据的查询和处理都使用相同的数据资源，改变客户类型和程序界面不会影响到数据处理的部分。这就要求将程序中“变”和“不变”的部分分开，尽量减少组件之间不必要的联系，使其相对独立并保持松散连接。

针对 Web 程序中的用户界面、业务逻辑、数据库三种系统设计需求，MVC 将系统划分为模型层、视图层和控制器层。模型层封装了内核数据和功能，是应用程序的核心。模型表示业务数据和业务逻辑，一个模型为多个视图提供数据，提高了应用的可重用性。视图层与用户交互，从模型层获取相关数据，每个视图都有一个相关的控制器组件。控制器接受用户输入并调用模型和视图，完成用户的请求。MVC 的处理过程是：控制器接受用户的请求，决定调用哪个模型进行处理，模型再根据用户请求进行相应的业务逻辑处理，并返回数据，控制器调用相应的视图装载模型返回的数据，最后通过视图返回给用户。

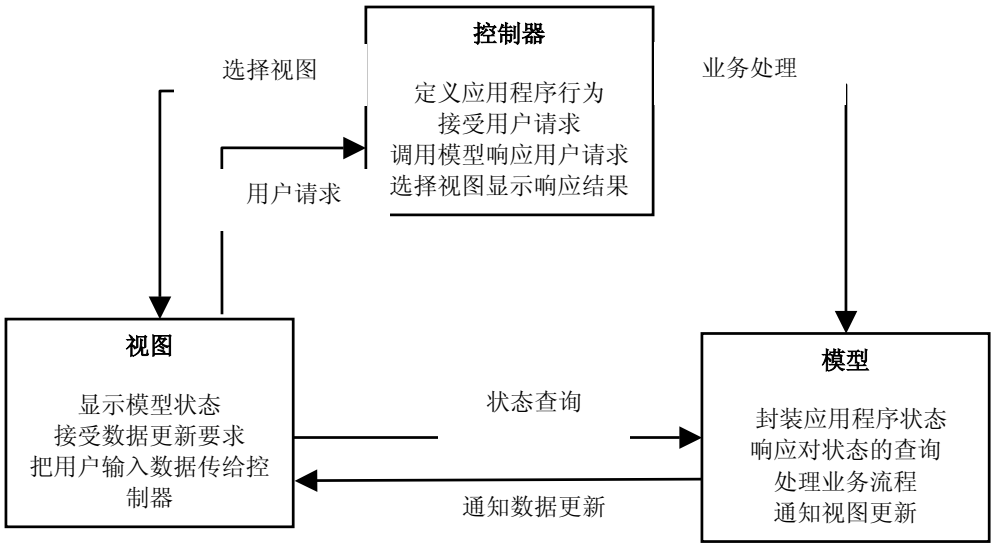



**图 2-3 MVC 设计模式**

#### 2.2.3.2 MVC 设计描述的优点和不足

MVC 的优点表现在以下几方面：

- 一个模型在运行时可以同时建立和使用多个视图，提高模型代码的重用性。
- 相关的视图可以及时得到模型数据的变化，从而使所有关联的视图和控制器同步。
- 允许更换视图和控制器对象，而且可以根据需求动态地打开或关闭，甚至在运行期进行对象替换。
- 模型的可移植性。模型独立于视图和控制器。把一个模型独立地移植到新平台工作，而只需在新平台上对视图和控制器进行修改，以此构造良好的低耦合组件。
- 潜在的框架结构，可以基于此模式开发应用程序的框架。

MVC 的不足表现在以下几方面：

- 增加了系统结构和实现的复杂性。简单的界面严格遵守 MVC，会增加系统的复杂性，减少灵活性，并可能产生过多的更新操作，降低了运行效率。
- 视图与控制器间连接过于紧密。视图与控制器是相互分离、却又联系紧密的部件。没有控制器，视图应用很有限，反之亦然，这样就妨碍了它们的独立应用。
- 视图对模型数据的低效率访问。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未发生变化数据不必要的频繁访问，也将弱化性能。

由此可见，MVC 并不适合所有的软件开发。对于存在大量用户界面、业务逻辑复杂的大型应用程序，MVC 有利于提高健壮性和重用性。MVC 框架在构建初期进展缓慢，但从长远角度看，提高了后期软件开发的效率。

### 2.2.4 MVC 设计模式的实现

早期应用 MVC 设计模式的 Web 系统中，程序语言和 HTML 的分离一直难以实现。通常在

JSP 页面中执行业务逻辑的程序代码，和 HTML 表示层数据混杂在一起，难以分离出单独的业务模型，造成 HTML 和 Java 代码强耦合、调试困难等诸多问题。为了解决这些问题，Sun 公司先后制定了两种规范[56]——JSP 模型 1（以下简称模型 1）和 JSP 模型 2（以下简称模型 2）。

#### 2.2.4.1 模型 1

模型 1 被称为“以 JSP 为中心”的 MVC 设计模式，是 JSP 和 JavaBean 技术的结合。JSP 页面独自响应请求，处理后把结果返回给客户端。所有的数据都通过 JavaBean 处理，JSP 页面同时实现显示、业务逻辑和流程控制功能，从而快速完成应用开发。其结构如图 2-4 所示：

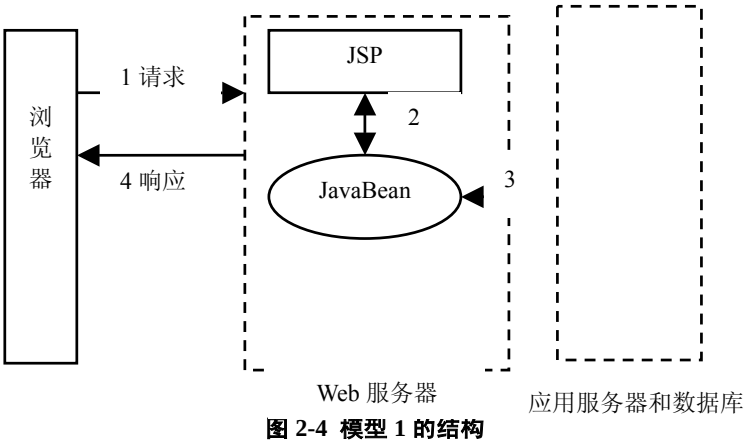



**图 2-4 模型 1 的结构**

使用模型 1 开发的 Web 应用是一组 JSP 页面，对于快速小规模应用开发比较有优势。但是从工程角度考虑，不足之处有两点：一是应用的实现依赖过程，一组 JSP 页面实现一个业务流程，如需改动，必须进行多处修改，不利于扩展和更新；二是应用不是建立在模块上，业务逻辑和表示逻辑混合在 JSP 页面中，没有进行抽象和分离，不利于应用系统业务的重用和改动。因此，模型 1 只是在一定程度上实现了 MVC 设计模式。

#### 2.2.4.2 模型 2

模型 2 是模型 1 的改进，完全基于 J2EE 体系结构。模型 2 又被称为“以 Servlet 为中心”的 MVC 设计模式，是 JSP、JavaBean 和 Servlet 技术的结合，充分利用了 JSP 和 Servlet 两种技术原有的优点，其结构如图 2-5 所示：

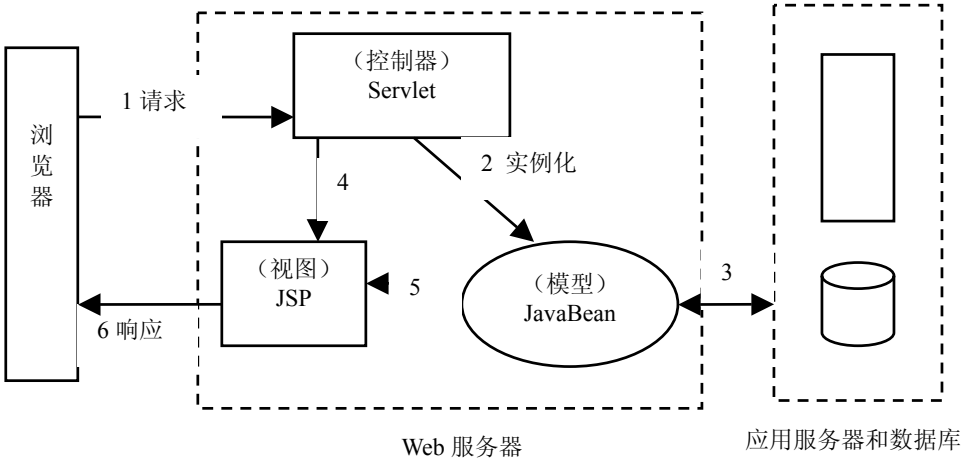


**图 2-5 模型 2 的结构**

模型2实现了MVC设计模式，使用一个或多个Servlet作为控制器。在模型层中，通过JavaBean（或 EJB 组件）实现应用的业务逻辑；在视图层中，由 JSP 页面产生应用的表示；在控制层中，Servlet 作为控制器处理来自 Web 浏览器的所有请求，然后返回 HTTP 响应信息。从而实现了 MVC 的分离机制，弥补了模型 1 的不足。

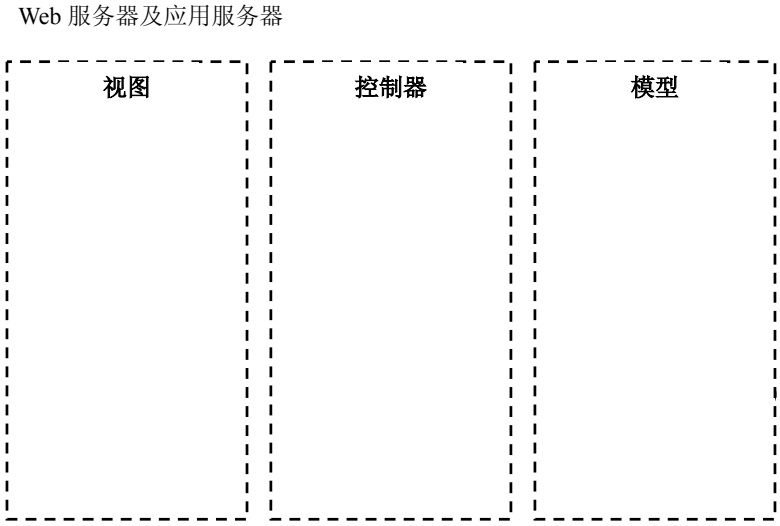



**图 2-6 Java Web 应用的结构**

基于对模型 1 及模型 2 的描述，总结两种模式的异同如下：

- 模型 1 开发速度比模型 2 快。尽管在模型 1 中，JSP 会在第一次被调用时编译成 Servlet，系统却无法将这个动态生成的 Servlet 作为静态类型使用。
- 模型 1 的业务逻辑和显示数据逻辑混合在 JSP 中，使得二者无法独立演化。而模型 2 中的 Servlet 不参与显示数据的工作，只负责产生中间数据，并将这些数据以 JavaBean 对象的形式存储在 Session 对象中。模型 2 有组件化的优点，更易于开发、管理大规模系统。
- 使用模型 1 比使用模型 2 更简单，模型 2 需要更多的时间学习和掌握。模型 2 中的 JSP 不含业务逻辑，Servlet 和 JavaBean 中含有所有的业务逻辑。因此在项目开发中可以根据团队的专业技能分配开发工作。

## 2.3 本章小结

本章从软件体系结构的一些基本概念等入手，介绍了 Web 应用程序的多层体系结构、J2EE 体系结构和 MVC 设计模式等相关知识。剖析了实现 MVC 设计模式的两种方法——JSP 模型 1 和 JSP 模型 2，着重分析了采用模型 2 实现的 Java Web 应用结构，最后总结了两种模型的异同。

# 第三章 Web 层应用框架研究

## 3.1 框架

### 3.1.1 框架的概念

框架[16]（framework）是软件系统的整体或部分的可重用设计，阐明了整个设计、协作组件之间的依赖关系、责任分配和控制流程，表现为一组抽象组件及组件实例之间交互的方法。另一种定义认为，框架[12]关注应用领域中已建立的系统结构，是可以被开发人员定制的应用系统的架构，是大粒度的可重用部件。前者是从体系结构的角度给出的定义，而后者是从设计模式的角度给出的。在面向对象的开发方法中，框架由多个抽象类或具体类组成，是一个具有特定关系的类集合。框架是组件技术、软件体系结构研究和应用软件开发结合的产物。

框架实现了某个应用领域通用功能的底层服务，使用框架可以在通用功能已经实现的基础上进行具体开发。应用框架强调软件的设计重用性和系统的可扩充性，以缩短大型应用软件系统的开发周期，提高开发质量。与传统的基于类库的面向对象重用技术相比，应用框架更注重于面向专业领域的软件重用，组件根据框架进行复合生成可运行的系统。框架的粒度越大，包含的领域知识就越完整。

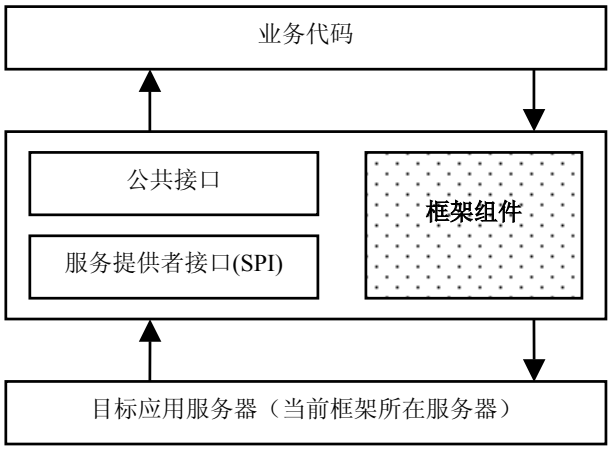



**图 3-1 框架在高层系统结构中的位置**

### 3.1.2 框架、组件和设计模式

软件开发中有三种级别的重用：内部重用，即在同一应用中公共使用的抽象块；代码重用，即将通用模块组合成库或工具集，以便在多个应用和领域都可以使用；应用框架的重用，即为专用领域提供通用的或现成的基础结构，以获得最高级别的重用性。

框架、组件（component）和设计模式是一个成熟平台中的三个重要组成部分。框架是为实现一个或多个设计模式的可重用、可扩展的设计，是软件系统开发的半成品，使用设计模式的框架能够达到更高水平的设计和代码重用。组件是框架中可重用的工具包、类库或 API，组件通常是代码级重用。设计模式则重用设计。框架则介于二者之间，部分重用代码，部分重用设计，有时重用分析。

随着软件技术的发展，软件重用已经从模块、对象等代码级重用发展到了组件、设计模式和

框架的重用。框架与设计模式虽然相似，但有着根本的不同。设计模式是对某种环境中反复出现的问题以及解决方案的描述；框架可以用代码表示，也可以直接执行或重用，对于模式而言只有实例才能用代码表示；设计模式是比框架更小的元素，一个框架往往含有一个或多个设计模式，框架总是针对某一特定应用领域，但同一模式却可适用于各种应用。可以说，框架是软件，而设计模式是软件的知识。

### 3.1.3 框架对于软件开发的意义[15]

目前，企业应用开发竞争日益激烈，需求变更频繁，系统集成商面临巨大的生存压力。大部分企业仍然没有摆脱手工作坊式的做法，每个项目或者产品由于管理人员或者团队的不同需求，重新设计系统框架，在结构验证和调整上浪费大量时间[25]。针对这种情况，应用框架在软件开发中的意义就突显出来。

- **知识积累**

框架的核心价值是知识的积累。软件开发是知识性活动，但是知识存在于人的大脑中，是最难进行积累的。在软件开发活动中，代码是确定的知识，所以从代码出发进行知识积累是最佳方式。框架包含了大量代码，这些代码是对某个特定问题域中抽象概念及抽象概念之间关系的描述。

- **资产保护**

知识积累本身是一项对资产的保护工作。另一项很重要的保护工作是软件组织（尤其是企业）需要保证对知识的学习和改进是经过合法授权的。框架可以以源代码或库形式发布。为不同的用户选择不同的发布形式，可以起到权限控制的作用。

- **鼓励重用**

框架最大的特点是重用。以框架为核心的开发方式是在开发过程中使用框架，在开发完成后改进框架。如此迭代，将通用的行为抽取出来，实现重用。

- **优化架构**

框架代表优秀的软件架构，使软件保持整体架构的稳定性和一致性。使用框架可以节省大量代码，结构更加清晰。

- **大规模软件设计**

大规模软件设计的关键在于对应用进行合理划分，并提供一种一致的方式建立架构。在大规模软件设计中，框架可以实现核心设计人员工作在同一个抽象层次上。

## 3.2 影响 Web 层应用框架选择的因素[17][18][22][23]

目前国内中小企业的 Web 应用系统中，大多数以单纯 JSP 代码构建。随着业务逻辑的日益复杂，系统构建所带来的复杂度和所耗费的成本进一步提高，由于结构不清晰导致系统可读性降低，后期的维护和扩展难度增加。在 Web 应用开发中，如何提高 Web 系统的可重用性、可扩展性、可维护性，降低构建和维护成本，成为业界普遍关注的焦点。近年来，涌现出许多 Java Web 层开源框架，据 2004 年 3 月的初步统计[26]，共有 54 种，如 Struts、WebWork、Spring MVC 等。面对诸多框架，如何选取合适的框架用于实际开发成为一个棘手的问题。通过运用软件体系结构和设计模式等基本原理进行分析和研究，并查阅大量资料，本文采用以下一些方面作为选择 Web 层框

架的主要因素：

- **适用性方面**

任何框架都只是适用于特定范围的应用开发。在选择框架时，需要考虑分布式应用的复杂性、生命周期、可伸缩性、可扩展性等方面的要求。

- **文档方面**

开发人员仅仅从代码理解框架非常困难，尤其对于开源项目而言，必须辅以层次高于代码的文档。框架的文档分为两种：参考文档（Tutorial）和使用文档（JavaDocs）。

- **源代码方面**

开源项目的最大优点就是可以获得框架的源代码。研究源代码可以使开发者更好地学习优秀的编码风格、理解框架以及提高自身的业务水平。框架本身具有的设计架构将直接提高应用系统的质量。框架的设计应该简单，并使开发者易于扩展。

- **工具支持方面**

集成开发环境（Integrated Development Environment，简称 IDE）对框架的支持将进一步降低框架使用的难度。

- **用户界面方面**

是否易于创建用户界面容易是评价框架的因素之一。大多数 Web 层框架都提供了客户自定义标签或者模板来创建用户界面。强大的用户界面组件可以使开发人员方便地设计出复杂的界面。

- **创新性方面**

框架自身具备一些特性增加了易用性。但在利用框架创新性的同时，还需权衡考虑这些便利所带来的限制。

## 3.3 Web 层框架的比较

通过运用以上因素研究 Web 层框架之后，本文重点选取了三个常用的轻量级框架——Struts、WebWork、Spring 进行深入比较分析。

### 3.3.1 WebWork、Spring MVC 和 Struts 框架的比较

- **适用性方面**

Spring 框架[46]是一个基于 POJO（Plain Ordinary Java Object，简单原始的 Java 对象，简称 POJO）的轻量级 J2EE 框架实现。在 Spring 框架中，可以实现多个子框架的组合。Spring MVC 提供了面向 Web 应用的 MVC 实现，为了与 WebWork 和 Struts 框架在 Web 层上比较，只选择 Spring 框架中的 Spring MVC 部分。

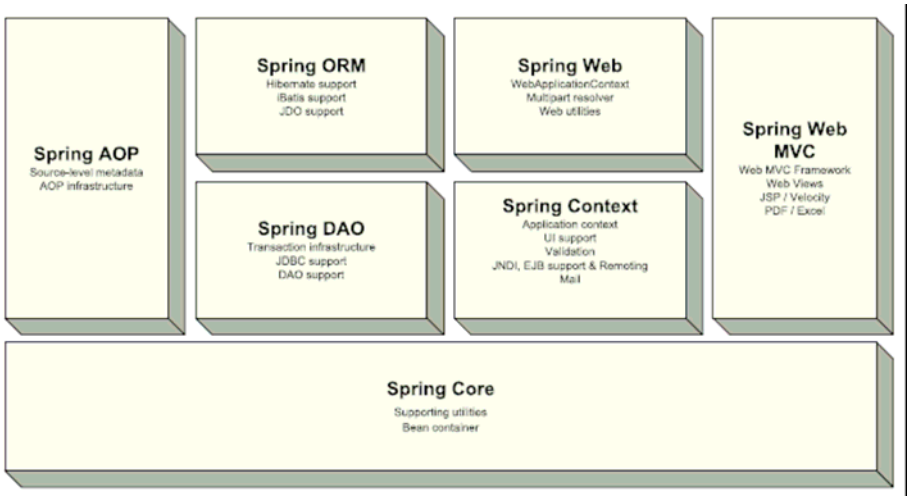



**图 3-2 Spring 框架的结构**

Struts、WebWork 和 Spring MVC 都是开源的轻量级框架。重视开发速度的大中型 Web 应用系统宜选用轻量级框架。但是这三个框架都没有对于系统可伸缩性的测试文档，开发者仍需要自行研究其结构，编写验证程序测试。

- **文档方面**

随着开源项目复杂程度的增加，文档的质量对于框架越来越重要。JavaDocs 是项目的重要参考文档，有些开源框架有丰富的类级描述，在方法级则有所欠缺，如 WebWork 框架包含 TBD 部分（To Be Done，简称 TBD）。文档应该更注重质量而非数量，Spring 的文档过于复杂。Struts 具有组织良好的文档，在类级和方法级都很详细，可以无需参考源代码。并且相关的技术文章和书籍也很多。

- **源代码方面**

Struts 在其自身的架构设计中采用了很多经典设计模式，这使得基于 Struts 框架实现的应用具有良好的设计模式。优秀的框架能够指导代码如何分布，基于 Struts 框架开发的 Web 应用程序比标准开发方法节省 30%－40%的编码量[22]。

- **工具支持方面**

WebWork 和 Spring MVC 没有 IDE 支持，开发依赖于 Eclipse 的相关插件等。Struts 有很多 IDE 工具和插件的支持。事实上，Struts 已经与一些产品集成，如获得 2002 年 JAVA IDE 大奖的 JBuilder 8 等。很多专门的开源项目也为 Struts 提供支持，例如 StrutsConsole（配置文件 GUI）、EasyStruts（代码生成器）、StrutsTestCase（单元测试工具）等。

- **用户界面方面**

WebWork 的视图部分可以使用 JSP、Velocity 和 JasperReports 等。Struts 的视图部分可以使用 JSP、Velocity 等组件。Spring 的视图层没有 Struts 灵活。

- **创新性方面**

WebWork 2.0 中，WebWork 被拆分成了 Xwork 和 WebWork2。WebWork2 建立在 Xwork 上，处理 HTTP 的响应和请求。WebWork[45]采取分级拉出式的 MVC 模式（Pull Hierarchical Model-View-Controller，简称 Pull HMVC）。“拉出式”是指视图组件根据要求，从控制器中将模型信息拉出作为响应。“分级”指视图数据的存放，“值堆栈”为视图提供信息。视图掌握需求信

息，并可访问该信息而无需等待控制器的响应。图 3-3 是 WebWork 的整体结构和工作流程：

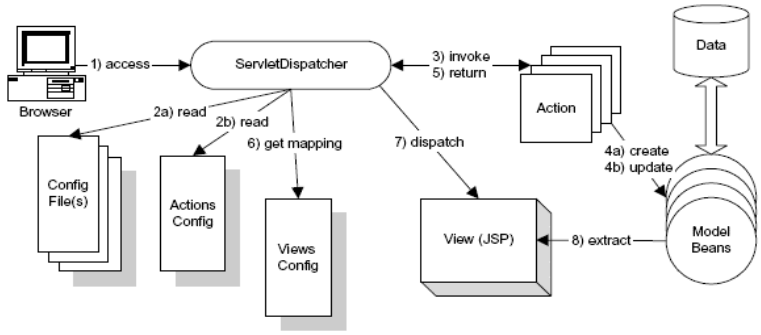



**图 3-3 WebWork 框架的结构和工作流程**

WebWork 启动时，调用主控制器 ServletDispatcher 读取配置文件，并为请求创建一个 Action 对象。模型 Bean 由 Action 创建，用来访问、改变数据，实现业务逻辑。Action 返回响应代码，ServletDispatcher 用它寻找对应的视图。ServletDispatcher 向视图组件分发控制，视图从值堆栈抽取信息。从表面上看，WebWork 的体系结构与 Struts 类似，但二者的处理细节不同。和 Struts 一样，WebWork 使用一个中心控制器，用于创建 Action 对象。WebWork2 使用 ServletDispatcher 将 HTTP 请求的变成 action，session 和 application 范围的映射，request 请求参数映射。Pull HMVC 需要一个所有视图都可得的数据存放（Data Repository），视图基于实时（Just-In-Time，简称 JIT）访问数据。因此，WebWork 的实现机制比 Struts 复杂。

Spring 的 Web 框架是围绕分发器（DispatcherServlet）设计的，DispatcherServlet 将请求分发到不同的处理器。缺省的处理器是一个简单的控制器接口。Spring 是一个以 IoC（Inversion of Control，控制倒置，简称 IoC）原则为基础的轻量级框架。控制倒置是用于“基于组件的体系结构”的设计模式，它将“判断依赖关系”的职责移交给容器，而不是由组件本身来判断彼此之间的依赖关系。Spring 的 MVC 类似于 Struts。Spring MVC 具有不同的对象角色：它支持一个控制器，一个可选的 command 或 form 对象，以及一个能够被传递给视图的模型的概念。模型通常会包含 command 或 form 对象，而且还可以是任意的引用数据。Spring 的控制器类似于 Struts 的 Action，只有一个实例处理所有客户请求，但是易用性没有 Struts 好。

Struts 框架改进了模型 2，使用标准的 Web API 和标准的 HTTP 请求响应模式，其核心是基于 Servlet、JavaBean、ResourceBundles 和 XML 技术的控制层。Struts 对内部实现机制透明，如果理解 Web 开发和设计模式，可以很快使用 Struts，开发人员的学习曲线更平滑，而 WebWork 和 Spring MVC 还需要对其实现 MVC 的原理进行学习。Struts 包括一系列通用标记扩展，核心组件可以被重写和子类化，开发人员可以定制关键类，如 ActionForm 和 Action 等。

### 3.3.2 Struts 框架的优点和缺点

- **适用性方面**

Struts 框架是开源的轻量级框架，对于大中型项目的开发优势明显。利用 Struts 框架实现的应用具有高内聚、低耦合的特点，并且符合模型 2 应用开发的规范。

- **文档方面**

Struts 的文档结构清晰，在类级和方法级都很详细，可以无需参考源代码。并且相关的技术文章和书籍也很多。

- **源代码方面**

Struts 在其自身的架构设计中采用了很多经典设计模式，使得基于 Struts 框架实现的应用具有良好的设计模式。

- **工具支持方面**

Struts 得到很多 IDE 及插件的支持，使得 Struts 的使用更为简单。

- **用户界面方面**

Struts 的视图部分可以使用 JSP、Velocity 等组件，实现复杂用户界面的设计。

- **创新性方面**

Struts 框架改进了模型 2，使用标准的 Web API 和标准的 HTTP 请求响应模式。

Struts 框架本身也有一些缺陷[59][61]。使用 Struts 之前，开发者要理解 Struts 组件和一些特殊类以及如何交互。Struts 并不倾向于某个特定的持久层，由开发者自行选择。JSP 实现的视图组件，只能借助一些基本的包含和转发功能，限制了视图的灵活性。Struts 的消息资源对建立国际化的资源和错误信息非常好，但不适合于处理大文本块。Struts 中一些名称容易混淆。例如，web.xml 中的 validate 选项和 Action 对象的 validate 方法无关，而和如何解析配置文件有关。

综合以上对 Web 层常用框架的比较和分析，Struts 框架提供了一种创建 Java Web 应用程序的高度可配置、可扩展的通用 MVC 框架结构，对 Web 应用程序的显示、表示和数据的代码进行了抽象，更好地适应了需求，提高了开发速度。Office Mate 企业办公系统选取 Struts 作为 Web 层的开发框架。

## 3.4 Struts 框架

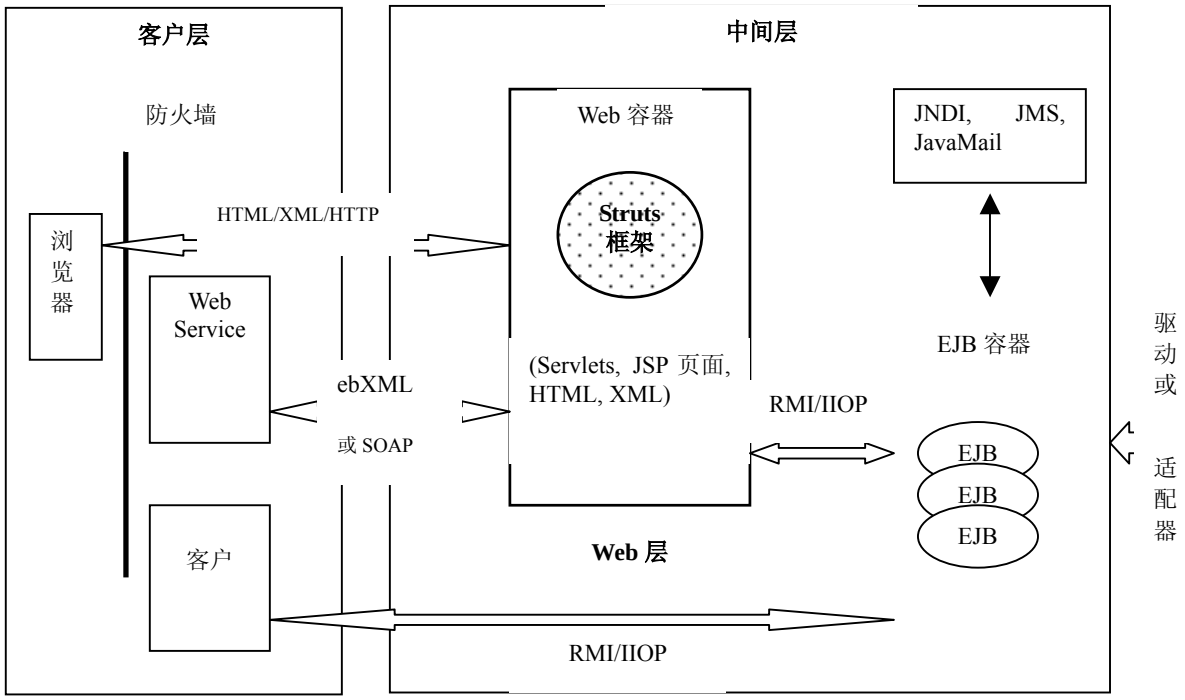


**图 3-4 Struts 框架与 Web 应用的关系[48]**

Jakarta Struts[47]是 Apache 软件组织提供的一项开放源码项目，Craig R. McClanahan 于 2000 年 5 月提出 Struts 框架的雏形。根据 J2EE 体系结构的规范，Struts 框架处于 Web 层，位于 Web 容器中，如图中阴影部分所示。

### 3.4.1 Struts 框架的体系结构[49][57][58]

Struts 框架由一组相互协作的类/组件、Servlet 和 JSP 标签库组成。基于 Struts 框架的 Web 应用程序符合模型 2 的设计标准。根据 2.2.4 节中对模型 2 的分析，采用模型 2 开发的应用程序必须基于 MVC 模式设计应用结构。MVC 是一个非常复杂的设计模式，选用框架实现 Web 应用可以取得事半功倍的效果。

Struts 框架改进了模型 2，对比图 3-5 与图 3-6 可以得出：Struts 框架更加灵活地运用了 MVC 设计模式，设计了自己的控制器——ActionServlet，增强了视图和控制器之间的联系，并且将业务逻辑从模型层分离出来，更好地实现了重用。

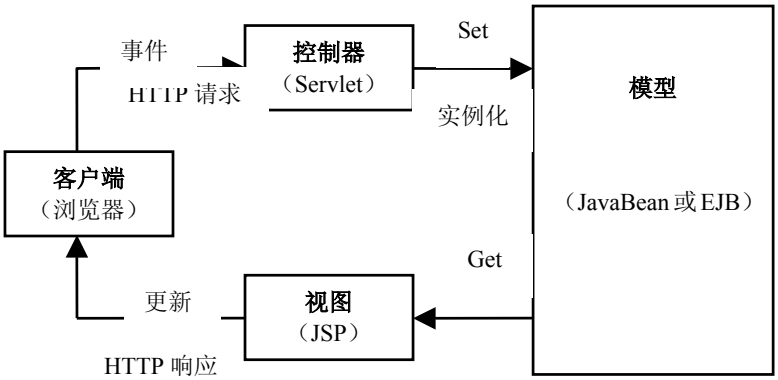



**图 3-5 模型 2 的体系结构**

Struts 框架的体系结构如图 3-6 所示。

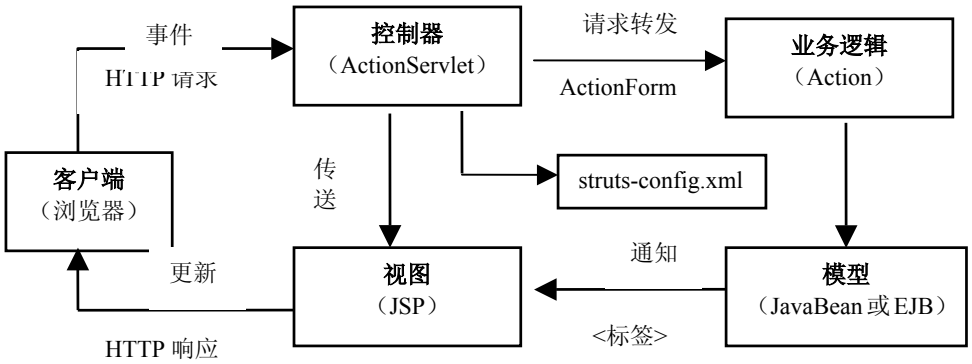



**图 3-6 Struts 框架的体系结构**

在 Web 层应用程序中使用 Struts 前后的对比如图 3-7 所示。

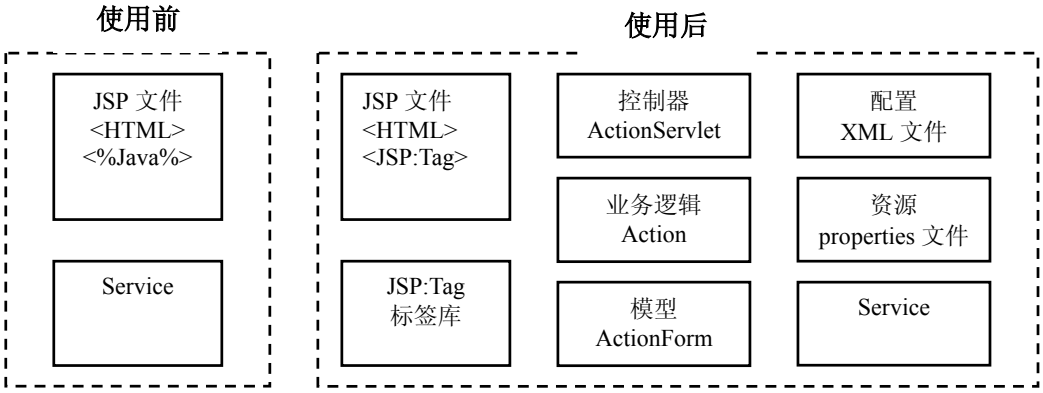



**图 3-7 Struts 框架使用前后的比较**

- **从视图角度**

Struts 框架的视图部分是一组包含了 Struts 自定义标签的 JSP 文件，可以实现和模型部分中的 ActionForm 映射，完成对用户数据的封装，还提供了国际化、表单验证、模板定制等功能。

- **从控制器角度**

Struts 框架的控制器组件主要由 ActionServlet 实现，接收并转发用户请求，调用合适的模型组件执行相应的业务逻辑，获取业务逻辑结果和选择合适的视图组件返回给用户。业务逻辑的操作由 Action、ActionForward 和 ActionMapping 几个组件协调完成。ActionServlet 类的实例接受客户端的请求。通过定义一组相应的 ActionMapping 来配置映射。Action 实现控制逻辑，与模型交互，执行改变或查询状态，通知 ActionServlet 下一个选择的视图，其功能主要由 execute()方法实现，该方法返回一个 ActionForward 对象，控制器通过该对象进行转发工作。

- **从模型角度**

模型部分从概念上可以分为两类：系统的内部状态和改变系统状态的动作。Struts提供了Action和ActionForm对象。所有的Action处理器对象都是从Action类派生的子类。Action处理器对象封装了具体的处理逻辑，调用业务逻辑模块，并且把响应提交到合适的视图组件以产生响应。ActionForm对象通过定义属性描述客户端表单数据。可以从ActionForm派生子类对象，和Struts自定义标记库结合，实现对客户端的表单数据的良好封装和支持，Action处理器对象可以直接对它进行读写，无需和request、response对象进行数据交互。通过ActionForm对象实现了对视图和模型之间交互的支持。Struts通常使用JavaBean表示系统的内部状态，根据系统的复杂度也可以使用EJB。

- **Struts 的配置文件**

Struts 主要采用两个 XML 文件来配置应用：web.xml 和 struts-config.xml。web.xml 是 Java Servlet 规范规定的 Web 应用配置文件，Servlet/JSP 容器使用该文件载入和配置应用。它指定了 ActionServlet 类、访问映射和其他 Struts 相关的选项，并且包含标签库的完整路径信息。struts-config.xml 是 Struts 框架的配置文件，用来创建和配置各种 Struts 组件。Struts 框架启动时，该文件中的配置信息读入到内存，存放到 config 包中相关 JavaBean 类的实例中。

- **应用资源文件 application.properties**

该文件为 Struts 应用提供资源，存储本地化信息和标签，使应用程序更易于国际化。应用资源文件是一个文本文件，每一行是一个关键字/值对。在 Java 和 JSP 代码中，给定一个关键字，

消息文本在运行时从应用资源文件中检索。应用资源文件通过 struts-config.xml 中的<message-resources>元素进行部署，在 web.xml 中由初始化参数 application 决定。

### 3.4.2 Struts 框架的核心组件和类[50][51][52]

Struts 大约由 15 个包，200 多个类组成[47]。Struts 框架中包含主要包的描述参见表 3-1。

**表 3-1 Struts 框架中的主要包**

| 包名称 | 描述 |
|---|---|
| org.apache.struts.action | Struts 框架的核心类和组件基本上都在这个包中。如 ActionServlet、Action、ActionForm、ActionMapping 等。 |
| org.apache.struts.actions | 提供客户的 HTTP 请求和业务逻辑之间的特定适配器转换功能。 |
| org.apache.struts.config | 提供对配置文件 struts-config.xml 元素的映射。 |
| org.apache.struts.util | 包含对一些常用服务功能的支持，如 Connection Pool 和 Message Source。 |
| org.apache.struts.validator | 用于动态配置对客户提交的表单验证。 |

Struts提供了可扩展的自定义标签库，简化了创建用户界面的过程。Struts标签库由四组标签组成，如表3-2所示。

**表 3-2 Struts 框架中的标签库**

| 标记库描述符 | 描述 |
|---|---|
| struts-html.tld | 包含用来创建 Struts 输入表单和其它基于 HTML 用户界面的标签 |
| struts-bean.tld | 包含用于定义新的 Bean 以及访问已经存在的 Javabean 及其属性的标记 |
| struts-logic.tld | 能够有条件地产生输出文本，在对象集合中循环并重复地产生输出文本，以及应用程序的流程控制 |
| struts-template.tld | 包含的标记用来定义模板机制，为页面创建动态的 JSP 模板 |
| struts-tiles | Tiles 框架，用于简化 JSP 开发 |
| struts-nested | 使得以上的基本标记库嵌套使用以及表达 JavaBean 之间的嵌套关系 |

Struts 框架包含的核心组件如表 3-3 所示。

**表 3-3 Struts 框架中的组件**

| 组件名称 | 描述 |
|---|---|
| ActionServlet | 控制器 |
| Action | 包含业务逻辑 |
| ActionForm | 显示模块数据 |
| ActionMapping | 帮助控制器将请求映射到操作 |
| ActionForward | 指向操作转移的对象 |
| ActionError | 存储和回收错误 |

- **ActionServlet**

ActionServlet 是控制器的重要组成部分，继承自 javax.servlet.http.HttpServlet。ActionServlet

负责接受 HTTP 请求信息，根据 struts-config.xml 中的配置信息，将请求转发给相应的 Action 对象。ActionServlet 在 web.xml 中的配置信息如下：

```
<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
</servlet>
```

全部请求 URI 以*.do 的模式存在并映射到 servlet：

```
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

- **Action**

ActionServlet 把全部提交的请求委托给 RequestProcessor 对象。RequestProcessor 使用 struts-config.xml 检查请求 URI 找到 Action 标识符。Action 类是执行业务逻辑的请求处理器（RequestHandler）。所有的 Action 对象都从 org.apache.struts.action.Action 派生。Action 对象封装具体的处理逻辑，将请求和业务逻辑分开，调用业务逻辑模块，并提交到相应视图组件产生响应。Action 的子类必须覆盖 execute()方法。当 ActionForm Bean 创建，表单验证顺利通过后，Struts 调用 Action 的 execute()方法，该方法返回 ActionForward 对象，包含请求转发路径信息。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                             HttpServletRequest request, HttpServletResponse response)
  throws IOException, ServletException
```

- **ActionMapping**

ActionMapping 类负责 Action 和请求信息相对应的映射配置说明。Struts 运行时，每个<action>元素都与 ActionMapping 的一个实例对应。特定请求映射到特定 Action 的相关信息储存在 ActionMapping 中，ActionServlet 将 ActionMapping 传送到 Action 类的 execute()方法，Action 使用 ActionMapping 的 findForward()方法，返回一个指定名称的 ActionForward，Action 完成了本地转发。若没有找到具体的 ActionForward，就返回一个 null。下面是一个典型的登录配置：

```
<action-mappings>
    <action    path="/LoginAction"
               type="com.omate.LoginAction"
               name="LoginForm"
               scope="request"
               input="/login.jsp"
               validate="false">
        <forward name="login"          path="/dispatch.jsp"/>
        <forward name="fail"           path="/login_error.jsp"/>
    </action>
</action-mappings>
```

- **ActionForm**

ActionForm Bean 是 Struts 提供的数据传输对象（Data Transfer Object，简称 DTO），用于在视图层和控制层之间传递 HTML 表单数据。ActionForm 符合 JavaBean 规范，除了具有 JavaBean 的常规方法，还有 validate()和 reset()两种特殊方法。控制层从 ActionForm Bean 中读取用户输入的表单数据，也可以把来自模型层的数据存放到 ActionForm Bean 中，再将它返回给视图。ActionForm Bean 使用 validate()方法进行表单验证，为模型层过滤不合法数据，但该方法不涉及

对数据的业务逻辑验证，只是完成简单的数据格式和语法检查。ActionForm Bean 和逻辑 Bean 的关系是：逻辑 Bean 表示模型的状态，ActionForm Bean 表示状态的改变。ActionForm Bean 收集和校验输入，业务逻辑 Bean 处理捕获的数据并将新数据合并到模型中。

上面的 ActionMapping 配置表示，当通过/LoginAction.do 提交请求消息时，控制器将信息委托 com.omate.LoginAction 处理，调用 LoginAction 实例的 execute()方法，同时将 Mapping 实例和所对应的 LoginForm Bean 消息传入。其中 name=LoginForm，使用的<form-bean>元素所声明的 ActionForm Bean，如下所示：

```
<form-beans>
    <form-bean name="LoginForm"      type="com.omate.LoginAction"/>
    </form-bean>
</form-beans>
```

### 3.4.3 Struts 框架的工作原理

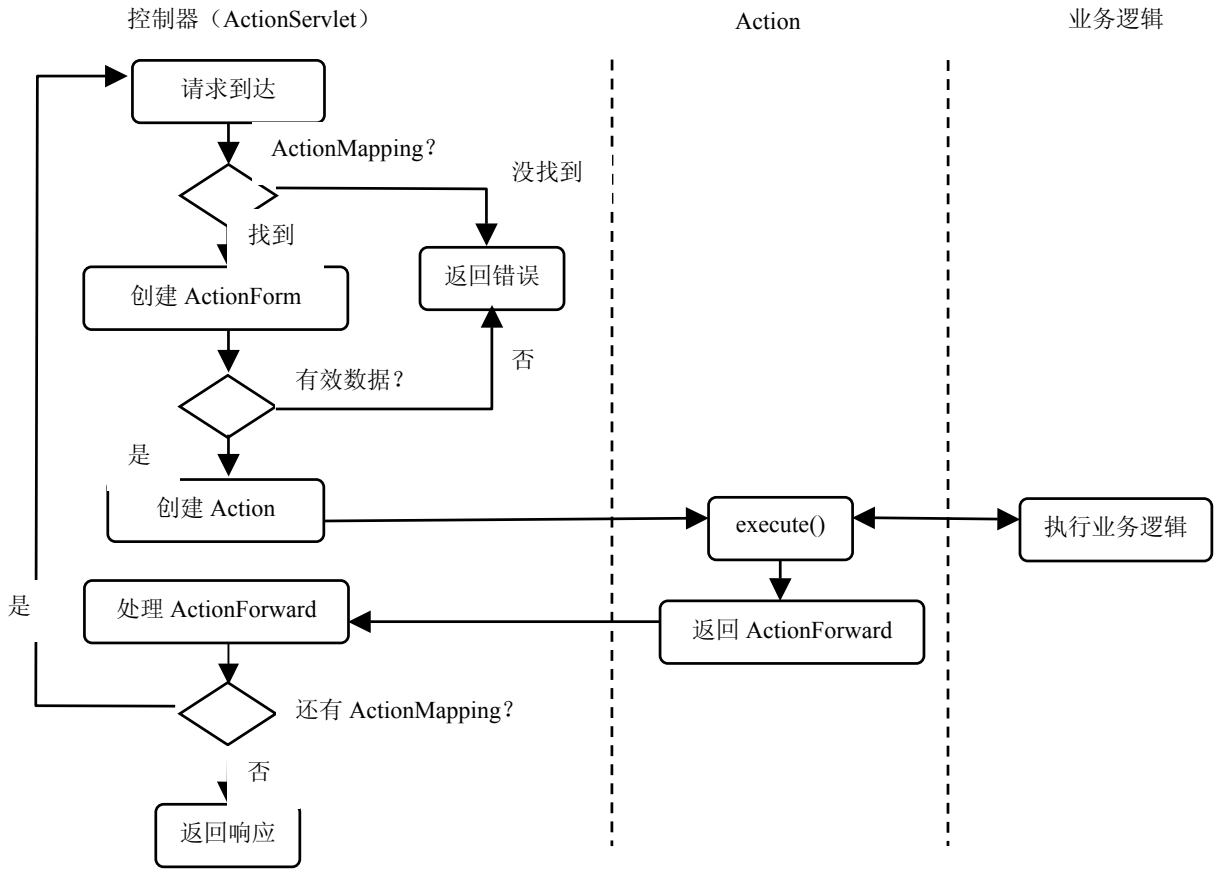



**图 3-8 Struts 框架的工作原理**

Web 应用启动时，Struts 框架加载并初始化 ActionServlet，ActionServlet 从 struts-config.xml 中读取配置信息，存放到各种配置对象中。当 ActionServlet 接受到用户请求时，执行以下流程：

1. 检索和用户请求匹配的 ActionMapping 对象，若不存在，返回用户请求路径无效的信息。
2. 若 ActionForm 对象不存在，则新建一个，并把客户提交的表单数据保存到该对象中。

3. 根据配置信息决定是否需要表单验证。若需要验证，则调用 ActionForm 的 validate()方法。
4. ActionForm 的 validate()方法返回 null 或返回一个不包含 ActionError 的 ActionErrors 对象，则表单验证成功；返回包含一个或多个 ActionError 的 ActionErrors 对象，则表单验证失败，此时，ActinServlet 将直接把请求转发给客户提交表单的 JSP 组件，不再创建 Action 对象并调用 Action 的 execute()方法。
5. ActionServlet 根据配置信息决定将请求转发给哪个 Action。ActionServlet 创建一个 ActionMapping 对象存放这个 Action 的配置信息，若对应的 Action 实例不存在，则创建该实例，然后调用 Action 的 execute()方法。
6. Action 的 execute()方法返回一个 ActionForward 对象，ActionServlet 再把客户请求转发给 ActionForward 对象指向的 JSP 组件。
7. ActionForward 对象指向的 JSP 组件生成动态网页，返回给客户。

## 3.5 本章小结

本章介绍了框架的概念及其在软件开发中的意义。根据影响 Web 层应用框架选择的因素，对 Struts、WebWork 和 Spring MVC 三个框架进行了深入分析，总结了 Struts 框架的优缺点。阐述了 Struts 框架的体系结构、核心组件和工作原理。

# 第四章 Office Mate 企业办公系统的系统分析与设计

## 4.1 系统需求分析

Office Mate 企业办公系统（以下简称 Omate）基于网络、力图使企业各部门都成为其信息流中的一个环节，而不再是一个个信息孤岛。应用该办公系统，企业组织结构得到简化，各部门在信息共享的基础上协作，部门和员工责任明确，决策层迅速综合各方面信息，制定战略决策。

### 4.1.1 系统的设计思想

- **功能丰富**

  具有完善的公文处理、电子邮件、信息发布、资料共享传输、即时通讯、在线业务处理等功能。满足员工和部门协同办公的需要，以进一步支持 ERP、电子商务等应用。

- **使用便捷**

  作为企业信息化诸多应用系统中服务面最大、用户最多的系统，系统必须高度集成。每个系统用户只须拥有一套用户账号和口令。操作界面简单易用，适用性强。

- **灵活性和可维护性**

  开发所选用的体系结构、开发工具灵活、轻量，系统具有开放性、模块化的特点。

- **可靠性和安全性**

  作为全天候、大范围使用的系统，软件系统和硬件设备稳定可靠，确保在故障情况下数据不丢失，应用不受影响。

### 4.1.2 系统的实现目标

基于以上设计思想，对 Office Mate 企业办公系统设定实现目标如下：

- **信息发布平台**

  将企业日常管理有价值的信息分类存储、发布，实现有权限的信息查询与共享。

- **资料传输平台**

  实现各种大容量电子文件资料在用户之间便捷传输。

- **电子邮件平台**

  建立性能稳定、统一处理内外部邮件的企业电子邮局，每个注册用户拥有一个电子邮箱。

- **公文处理平台**

  实现各种公文从起草到归档全过程的规范化、高效率，并建立公文上报、下发和流转的机制。

- **业务处理平台**

  使员工方便地在线处理业务。

## 4.2 系统运行环境

### 4.2.1 系统网络拓扑

Office Mate 企业办公系统基于网络环境。企业内部设置 OA 服务器、数据库服务器、应用服务器，邮件服务器、电子商务网站服务器等，配置远程访问服务器作为网络线路备份，同时支持移动办公和家庭办公。

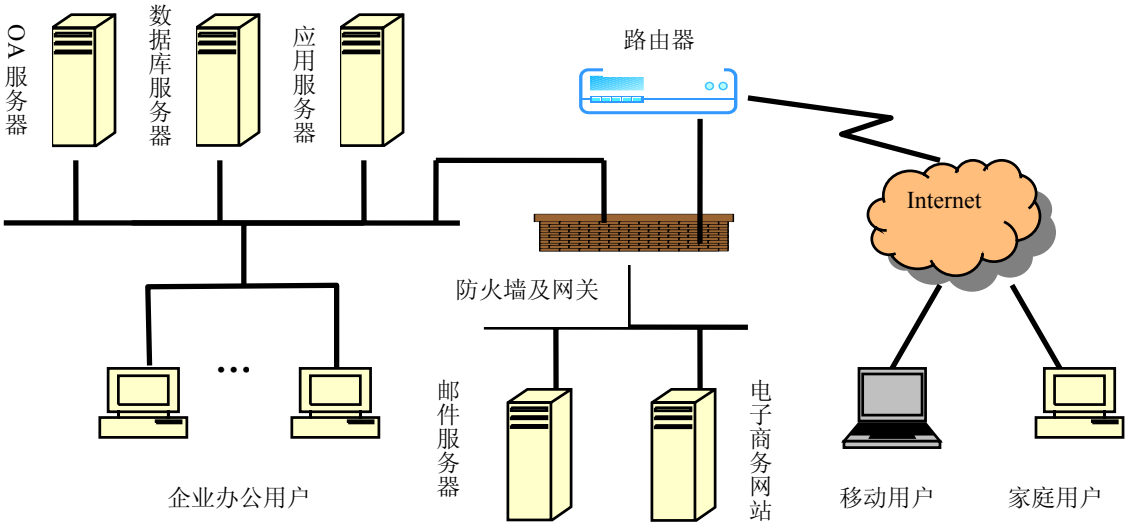



**图 4-1 Office Mate 企业办公系统的网络拓扑图**

### 4.2.2 系统平台及技术

#### 4.2.2.1 技术选型说明

根据 4.1 节的系统需求分析，该系统应具备良好的交互性、可维护性、可扩展性和跨平台性。选用 B/S 三层架构进行应用系统的开发，J2EE 作为开发平台，采用统一的用户管理与认证，为不同的平台和应用系统提供统一的前台用户环境。

- **IDE**

支持Java的IDE有Borland JBuilder、Eclipse等。JBuilder集成了开发与测试环境，减少了系统的配置和调试时间。本系统选择的集成开发工具是Borland JBuilder 9企业版，为Struts 1.1提供了良好的IDE支持。

- **JSP/Servlet容器**

Tomcat 是 Jakarta 项目中的一个重要子项目[55]，在 SUN 公司的 JSWDK（JavaServer Web Development Kit，简称 JSWDK）基础上发展而来。Tomcat 是一个开放源代码、运行 Servlet 和 JSP 应用程序的 Web 容器，是 JSP/Servlet 规范的官方参考实现。考虑到本系统并非是即将投入到实际运行的系统，Web 服务器选择 Tomcat，虽然它在稳定性、安全性等方面不甚完善，但用于 Omate 系统的开发研究足以胜任。

- **设计工具**

系统采用UML（Unified Modeling Language，统一建摸语言）建模。UML是一种支持面向对象开发过程的建模语言。Rational Rose是目前流行的CASE工具（Computer Aided Software Engineering，简称CASE）。在系统需求阶段、OOP、OOD、软件的实现和测试阶段，提供了清晰的表达方法和完善的工具，以建立相应的软件模型[44]。本系统采用Rational Rose 2003完成基于UML的设计建模工作。

- **数据库**

采用免费的 MySQL 4.1 作为后台的关系型数据库。MySQL 是一个功能强大，具有灵活、丰富的 API 和系统结构的数据库管理系统[63]。

- **Web 层框架**

根据 3.3 节中对 Web 层框架的比较，Web 层框架采用 Struts 1.1，以达到层间松散耦合，提高系统灵活性、重用性和可维护性，同时可以简化分析、设计、编码、测试和发布等阶段的工作。

#### 4.2.2.2 技术路线小结

- ✓ 操作系统： Windows Server 2003 中文版
- ✓ 数据库： MySQL 4.1
- ✓ 开发语言： Java 1.4
- ✓ 集成开发工具： JBuilder 9.0
- ✓ Web 层框架： Struts 1.1
- ✓ Servlet/JSP 服务器： Tomcat 5.0
- ✓ Web 浏览器： IE 6.0
- ✓ 系统分析建模工具： Rational Rose 2003

## 4.3 系统详细设计

整个系统的软件架构采用 J2EE 模型，采用分层设计模式和低耦合、高内聚的设计思想进行设计。基于 UML 的系统建模过程[27][29][30]，分为系统需求分析、系统静态分析、系统动态分析、系统设计几个阶段，历经多次迭代，最终建立一个完整详细的系统。建模之前，通过对问题域进行详细的分析确定用例，根据用例创建概念模型，模拟问题域中的真实实体。在软件设计阶段[43][62]，在概念模型的基础上创建设计模型，用 UML 类框图、活动图以及状态图来描述设计模型。

在对 Omate 建模之前，通过对原有系统功能模块的分析、研究，并对企业的组织结构、管理需求、日常办公需求、业务需求的了解和分析，为了给企业内部提供一个统一的信息通信平台，实现业务工作流自动化、企业内部信息的共享和管理，系统划分成四大模块：日常办公、日常业务、综合信息和系统管理，如图 4-2 所示。

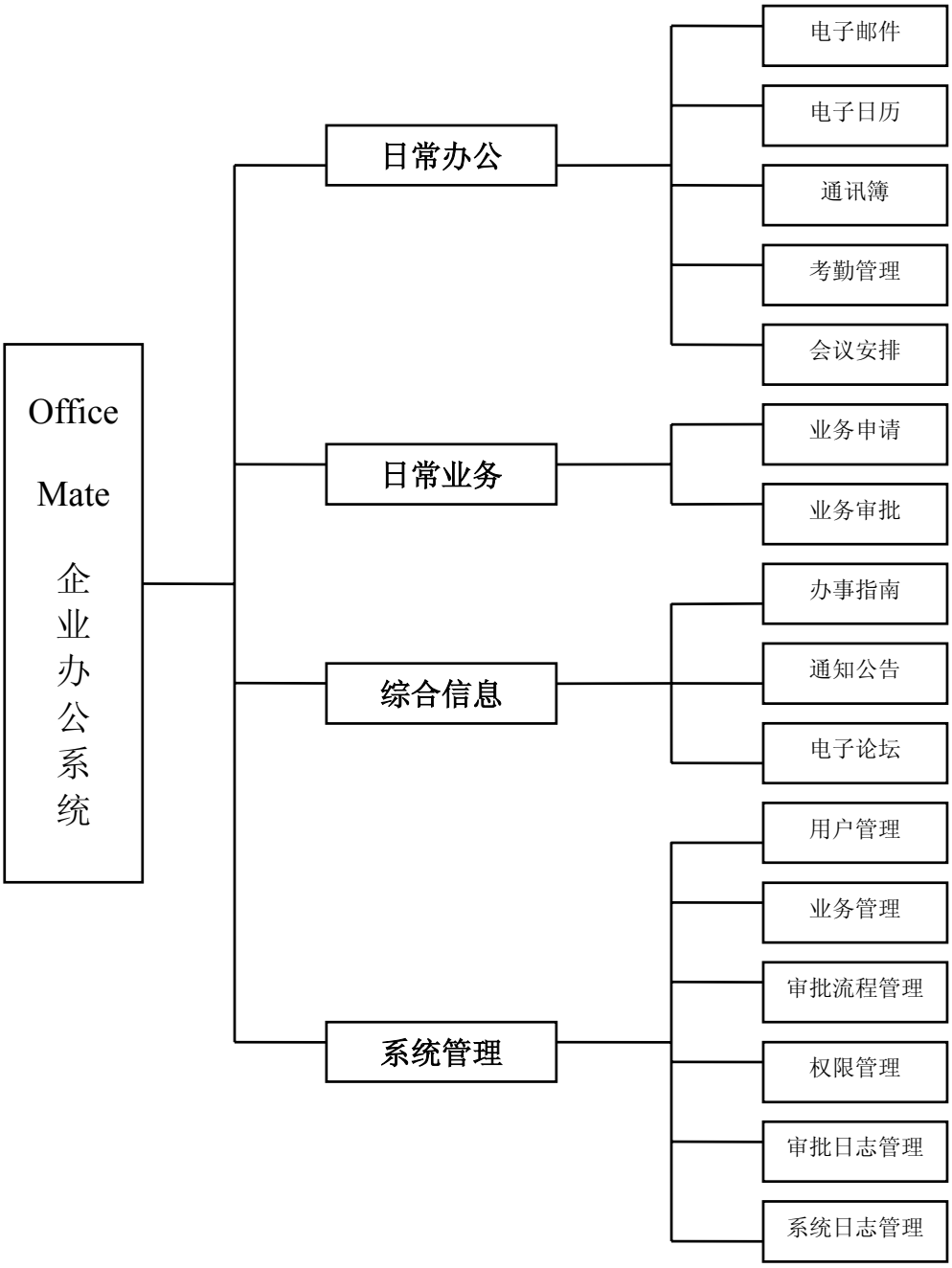



**图 4-2 Office Mate 企业办公系统的模块结构图**

### 4.3.1 各模块功能简介

- **日常办公**

该模块实现了系统用户各种日常办公事务的处理，分为电子邮件、电子日历、通讯簿和考勤管理四个子模块。

- 电子邮件

为了满足系统用户的日常邮件联系，提供电子邮件功能。系统用户可以接收、发送、转发、撰写和管理邮件。

- 电子日历

系统用户可以安排自己的日程，包括日期、时间、预计的完成时间、工作内容、事件的重要程度、要通知的个人、要做的准备工作等。

- 通讯簿

系统用户可以管理自己的通讯簿，添加新联系人，编辑现有联系人，删除和整理通讯簿，对联系人进行分组管理。

- 考勤管理

为贯彻企业时间制度建立的模块。记录员工上下班、加班、出/缺勤时间，处理考勤数据，并提供与薪资系统的接口，根据此模块可查询企业内各时段的员工出、缺勤状况及加班状况，作管理决策方面的分析。

**➢ 日常业务**

该模块体现了工作流自动化，主要由两个部分组成：业务申请和业务审批。

- 业务申请

系统用户可以申请不同类型的业务。根据业务类型，下载业务类型的申请表格。系统用户填写表格，然后上传，正式提交之前，可以修改申请内容和表格，或者删除该项业务申请，提交后就不能再做修改，只能查看业务的详细信息，以及申请业务的审批状态。

- 业务审批

具有审批权限的用户可以审批相关的业务。一项业务的每个审批步骤都由不同的审批者审批。当审批者在某个审批步骤同意该业务申请时，该项业务会根据相应的业务类型的审批流程自动流向下一个审批者。

**➢ 综合信息**

系统内部和外部的信息发布于该模块中，以供查询。主要分为电子论坛（BBS）、通知公告和办事指南三个部分。

- 电子论坛

允许系统用户相互讨论问题，最大限度地发挥企业办公系统的协作功能。

- 通知公告

实现企业内部通知、布告、海报、快讯等内容的起草、审核、批示、发布、查阅等功能。具有发布权限的系统用户有权发布通知。

- 办事指南

实现对企业内部的某些事务办理流程的管理。将办理办法和过程限定等录入系统。系统用户可以随时了解企业的规章制度、办事流程等信息。允许有权限的系统用户发布信息。

**➢ 系统管理**

该模块实现对整个系统的管理，细分为以下几个子模块：系统日志管理、审批日志管理、业务管理、用户管理、权限管理和审批流程管理。

- 系统日志管理

主要负责系统监控和日常维护等工作。用户从登录到退出系统的每个动作都记录在系统日志中。系统管理员可以查看系统日志，查询错误原因和时间，恢复系统。

- 审批日志管理

针对业务审批模块的日志。记录某个审批者在某个时间审批了某项业务的某个审批步骤。具

有相应权限的系统用户可以查看某项业务的审批过程。

- 业务管理

主要针对日常业务的业务申请和业务审批。实现业务的增加、删除、修改和业务审批步骤的添加、修改，确定审批步骤和每个审批步骤需要的审批人数。实现业务申请表格的制定和上传。

- 用户管理

该模块主要负责系统用户的注册、注销、修改等，增强对系统用户的管理。

- 权限管理

主要负责对企业资源的访问控制，设定不同权限的用户使用的资源级别。

- 审批流程管理

管理、定制和更改审批流程。设定业务审批模块中的业务审批自动随审批者的审批动作转向下一位审批者。具体负责一项业务中多个审批步骤的权限分配，根据审批步骤的要求决定最少需要分配的审批者个数。

结合 Struts 框架在系统实现中运用研究的需要，主要设计和实现了电子日历、通讯簿、会议安排、考勤管理、用户管理、权限管理、系统日志管理等几个模块。

### 4.3.2 电子日历模块的设计

该模块为系统用户提供安排个人日程的功能。系统用户可以安排个人日程，包括日期、时间、预计的完成时间、工作内容、事件的重要程度、要通知的个人、要做的准备工作。允许系统用户查看、修改日程表、管理日常活动。

#### 4.3.2.1 电子日历模块的类设计

电子日历模块的主要功能是系统用户可以安排自己的工作日程，部门领导可以安排本部门的日常活动。该模块中的模型部分由个人日程安排和群组日程安排的业务逻辑组成：个人日程安排和群组日程安排，相应设计 PersonalCalendar 和 GroupCalendar 两个类。如图 4-3 所示。

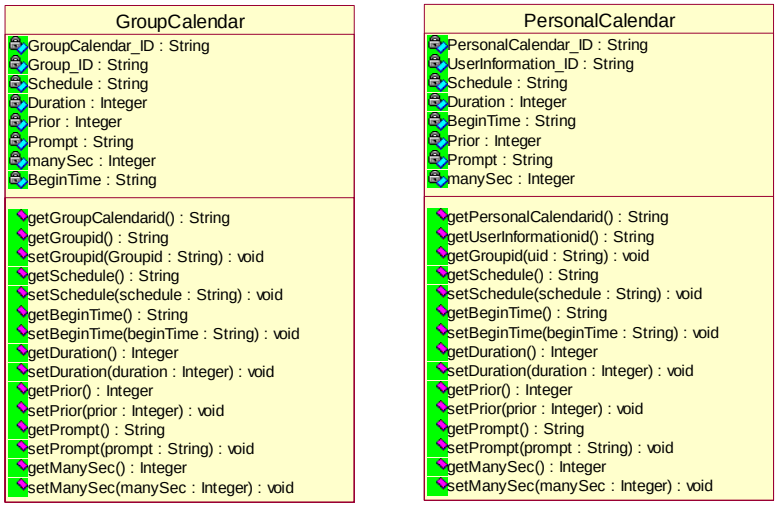


**图 4-3 电子日历的类图**

#### 4.3.2.2 电子日历模块的表设计

电子日历包括个人日历表和群组日历表两个表结构，分别映射 PersonalCalendar 和 GroupCalendar 类。

**表 4-1 PersonalCalendar 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| PersonalCalendar_ID | varchar(16) | 否 | 关键字，日程安排的唯一标识号 |
| UserInformation_ID | varchar(16) | 否 | 引用自 UserManag，系统用户 ID |
| Schedule | varchar(500) | 否 | 日程安排的具体内容 |
| BeginTime | varchar(20) | 否 | 表示日程安排的开始时间 |
| Duration | varchar(20) | 否 | 表示日程安排的持续时间 |
| Prior | varchar(2) | 否 | 日程安排的优先级 |
| Prompt | varchar(20) | | 给系统用户的提醒方式 |
| manySec | varchar(20) | | 表示提前多长时间提醒系统用户，相对于 BeginTime 而言。 |

**表 4-2 GroupCalendar 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| GroupCalendar_ID | varchar(16) | 否 | 关键字，日程安排的唯一标识号 |
| Group_ID | varchar(16) | 否 | 部门 ID，即系统用户组 |
| Schedule | varchar(500) | 否 | 日程安排的具体内容 |
| BeginTime | varchar(20) | 否 | 表示日程安排的开始时间 |
| Duration | varchar(20) | 否 | 表示日程安排的持续时间 |
| Prior | varchar(2) | 否 | 日程安排的优先级 |
| Prompt | varchar(20) | | 给系统用户的提醒方式 |
| manySec | varchar(20) | | 表示提前多长时间提醒系统用户。相对于 BeginTime 而言的。 |

### 4.3.3 通讯簿模块的设计

系统用户可以管理自己的通讯簿，如添加新联系人，编辑现有联系人，删除和整理通讯簿，对联系人分组管理等。

#### 4.3.3.1 通讯簿模块的类设计

在该模块中设计了两个类，分别是系统用户的联系人（PersonalContact）和联系人的群组管理（GroupContact）。ContactAdmin 类对这两个类进行管理，提供所需的 API。这三个类之间的关系如图 4-4 所示：

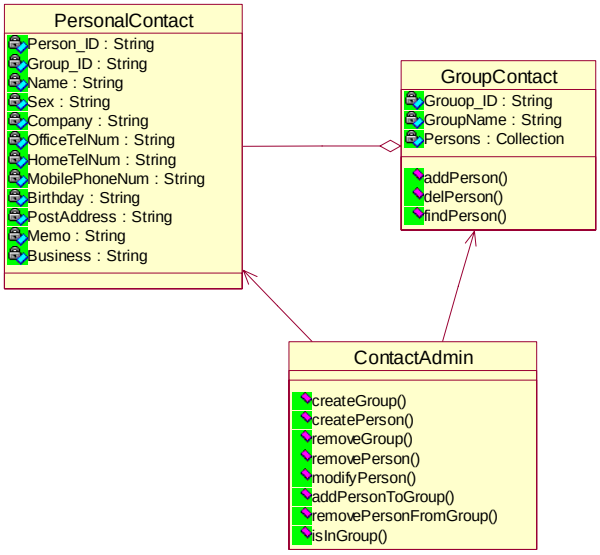



**图 4-4 通讯簿的类图**

#### 4.3.3.2 通讯簿模块的表设计

该模块对联系人和群组两类业务数据进行管理。表结构如表 4-3 所示：

**表 4-3 PersonalContact 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Person_ID | varchar(16) | 否 | 关键字，联系人的唯一标示号 |
| Group_ID | varchar(16) | | 表示联系人群组，引用 GroupContact 表的 Group_ID |
| Name | varchar(30) | 否 | 联系人的姓名 |
| Sex | varchar(2) | | 联系人的性别 |
| PostAddress | varchar(300) | | 联系人的地址 |
| Company | varchar(300) | | 联系人的公司名称 |
| Business | varchar(20) | | 联系人的职务 |
| Birthday | varchar(20) | | 联系人的生日 |
| OfficeTelNum | varchar(30) | 否 | 联系人的工作电话 |
| HomeTelNum | varchar(30) | | 联系人的住宅电话 |
| MobilePhoneNum | varchar(30) | 否 | 联系人的移动电话号码 |
| Memo | varchar(500) | | 备注，如联系人的爱好等等 |

**表 4-4 GroupContact 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Group_ID | varchar(16) | 否 | 关键字，群组的唯一标示号 |
| GroupName | varchar(16) | 否 | 群组名称 |

### 4.3.4 会议安排模块的设计

系统用户能够接受各级会议通知，查询会议安排。允许办公人员主持会议、发送与会人员通知，对会议进行纪要和归档。

#### 4.3.4.1 会议安排模块的类设计

该模块实现对会议和会议室的管理。设计管理类（MeetingAdmin）对会议和会议室进行管理。会议室集合类（MeetingRooms）负责会议室的管理，可以增加、删除、修改、查询会议室等。

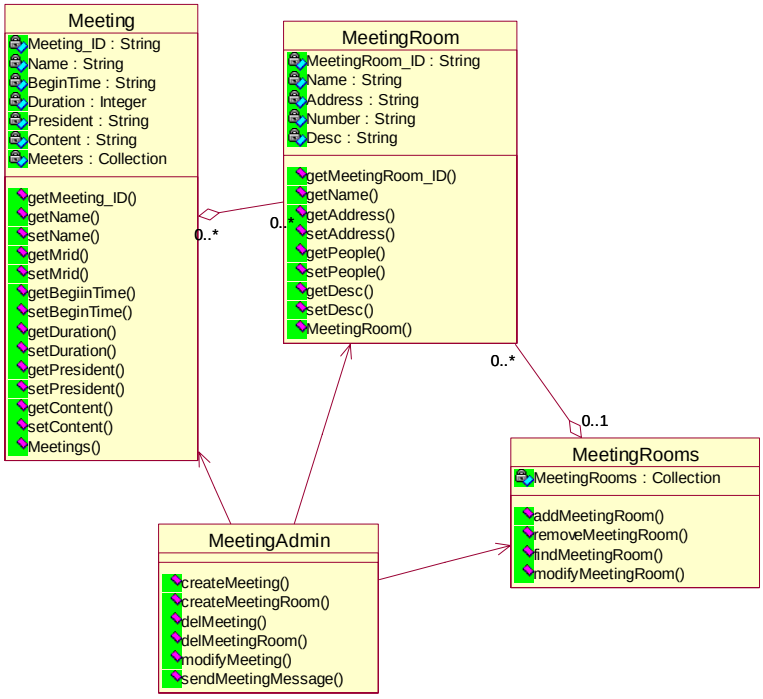



**图 4-5 会议安排设计的类图**

#### 4.3.4.2 会议安排模块的表设计

该模块对会议和会议室进行管理，后台数据主要包括会议和会议室。Meeters 表表示参加某次会议的系统用户。表结构如下列表所示。

**表 4-5 Meeting 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Meeting_ID | varchar(16) | 否 | 关键字，会议的唯一标示号 |
| MeetingName | varchar(16) | 否 | 会议名称 |
| Address | varchar(16) | 否 | 会议召开的会议室，引用表 MeetingRoom 中的 MeetingRoom_ID |
| BeginTime | varchar(20) | 否 | 表示日程安排的开始时间 |
| Duration | varchar(20) | | 会议持续时间 |
| Content | varchar(500) | 否 | 会议内容（纪要） |
| President | varchar(16) | 否 | 引用自 UserManag 表，系统用户 UserInformation_ID |

**表 4-6 Meeters 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Meeting_ID | varchar(16) | 否 | 会议 ID，引用 Meeting 表中的 Meeting_ID |
| UserInformation_ID | varchar(16) | 否 | 引用自 UserManag 表，系统用户 ID |

**表 4-7 MeetingRoom 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| MeetingRoom_ID | varchar(16) | 否 | 关键字，会议室的唯一标示号 |
| Name | varchar(16) | 否 | 会议室名称 |
| Address | varchar(500) | 否 | 会议室地址 |
| Number | varchar(20) | 否 | 会议室能够容纳的人数 |
| Desc | varchar(400) | | 会议室描述，如配套资源、是否具有多媒体设备等 |

### 4.3.5 考勤管理模块的设计

考勤管理模块是企业时间制度管理的模块。企业员工出勤率的高低直接影响到工作质量。考勤管理模块记录员工上下班、加班、出/缺勤时间，可将考勤数据输出处理，提供薪资系统的接口，以便作薪资结算。企业还可根据此考勤系统查询企业内各时段的员工出、缺勤状况及加班状况，作管理方面的分析。

日常考勤数据以登录时间即为签到时间，第二次登录时间为离开时间。办公系统用户成功登录考勤管理模块后，接收到用户 ID 时，调用系统时间，对于因病、因公出、或请假的人，提前将其信息录入特勤情况表。对于特勤调特勤情况表，根据相应的特勤原因调出相应的符号填入月考勤表，既没有登录又没有请假的视为缺勤。信息的统计可分别按月，按年进行。根据日考勤表和特勤情况表的信息，在月考勤表中填入用户出勤情况所对应的符号。根据用户的签到时间和离开时间将相应的符号填入月考勤表，月末对当前的信息进行备份，之后将对应数据表清空，以便下个月使用，年终时，将相应统计表信息备份清空，便于下一年使用。

#### 4.3.5.1 考勤管理模块的用例分析

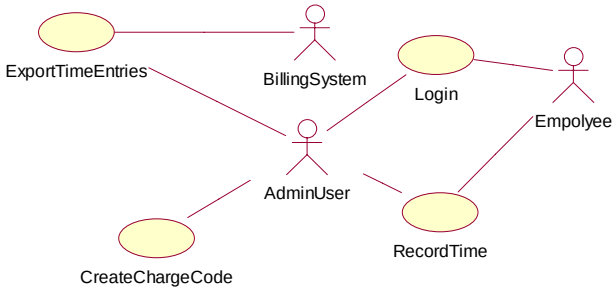



**图 4-6 考勤管理系统的用例分析**

角色：薪资系统（BillingSystem）、系统用户（Employee）、管理员（AdminUser）

用例：包括 Login、CreateChargeCode、ExportTimeEntries、RecordTime

CreateChargeCode：系统管理员 AdminUser 用 CreateChargeCode 用例来为考勤管理模块增加项目代码。

RecordTime：系统用户使用 RecordTime 用例登记工作时间

ExportTimeEntries：系统管理员 AdminUser 使用系统管理员 ExportTimeEntries 将特定的考勤数据保存到格式文件上。

#### 4.3.5.2 考勤管理模块的表设计

**表 4-8 Department 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Department_ID | varchar(20) | 否 | 关键字，部门的 ID |
| Department_Name | varchar(20) | 否 | 部门名称 |
| Found_Date | varchar(20) | | 成立时间日期 |

**表 4-9 Daily_TimeTracking 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(20) | 否 | 引用自 UserManag 用户 ID |
| Department_ID | varchar(20) | 否 | 引用自 Department 表的 ID |
| Signin_Time | varchar(20) | 否 | 签到日期时间 |
| Signout_Time | varchar(20) | 否 | 离开日期时间 |

**表 4-10 Month_TimeTracking 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(20) | 否 | 引用自 UserManag 用户 ID |
| Department_ID | varchar(20) | 否 | 引用自 Department 表的部门 ID |
| Department_Name | varchar(20) | 否 | 部门名称 |
| D1 | varchar(2) | | 一个月第一天的出勤情况 |
| D2 | varchar(2) | | 一个月第二天的出勤情况 |
| .. | … | … | … |
| D31 | varchar(2) | | 一个月第 31 天的出勤情况 |

**表 4-11 Special_TimeTracking 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(20) | 否 | 引用自 UserManag 用户 ID |
| STT_Reason | varchar(20) | 否 | 特勤的原因 |
| STT_Date | varchar(20) | 否 | 特勤的时间 |

**表 4-12 Year_TimeTracking 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(20) | 否 | 引用自 UserManag 用户 ID |
| Department_ID | varchar(20) | 否 | 引用自 Department 表的部门 ID |
| Department_Name | varchar(20) | 否 | 部门名称 |
| M1 | varchar(2) | | 第一个月的出勤情况 |
| M2 | varchar(2) | | 第二个月的出勤情况 |
| .. | … | … | … |
| M12 | varchar(2) | | 第十二个月的出勤情况 |

**表 4-13 Symbol 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Status | varchar(8) | 否 | 出勤情况 |
| Symbol | varchar(8) | 否 | 符号 |

### 4.3.6 用户管理模块的设计

用户管理模块实现对用户的分类管理，分为两部分：一是系统管理员对用户信息的管理，包括用户的查询、添加、修改以及删除等操作；二是用户对个人信息的更改，包括注册申请，密码修改，个人设置修改以及注销申请的操作。该模块主要包括用户信息管理子模块、用户注册子模块、用户登录子模块、用户信息变更子模块等。

#### 4.3.6.1 用户信息管理子模块

用户信息管理子模块为系统管理员显示配置完毕的用户信息，同时完成用户个人信息的查询、添加、修改、删除操作。该功能接收系统管理员输入的用户名或其它关键字信息检索条件作用于数据库，并从数据库中查找到所需要的用户信息，显示给管理员，并进行修改，同时可以添加新的用户信息；该模块具有设定用户级别，划分用户权限的功能，作为用户登录系统、使用系统特定功能的依据。

用户信息管理子模块采集用户个人信息，设定相应的用户级别、用户状态等信息。模块被激活后，首先向服务器申请显示所有配置完毕的用户个人信息，可以修改用户信息。可以通过检索的形式，从服务器获得符合检索条件的用户信息，并允许系统管理员修改。可以添加新的用户信息，配置用户级别、用户状态，删除已配置完毕的用户信息。所有对信息的编辑操作保存之前需经过确认，确认后返回成功信息。对于操作成功的编辑操作，数据库记录修改日志信息，内容包括系统管理员名称，编辑时间，编辑操作等日志信息。处理过程如下：

1. 通过检索的形式，选择并显示满足检索条件的用户个人信息；
2. 选择修改功能，对选定的用户个人信息进行修改，包括用户级别、用户状态等基本信息的修改；
3. 系统管理员对用户状态进行修改时，对于新用户的激活操作，将用户的注册信息等以电子邮件的方式发送给用户；
4. 选择添加用户信息功能，录入用户个人信息，设定用户级别、用户状态；
5. 对修改和添加的各项用户个人信息进行合法性校验；
6. 选择删除功能，对配置完毕的用户信息执行删除操作，在删除前请求用户确认；
7. 选择保存功能，显示编辑信息，确认正确后，申请保存用户个人信息；
8. 保存成功，返回用户信息管理子模块的初始页面，并向数据库记录修改日志信息。同时自动给用户发送电子邮件，通知其个人信息修改内容。

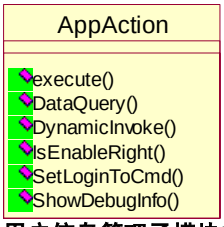



**图 4-7 用户信息管理子模块的类设计**

#### 4.3.6.2 用户注册子模块

用户注册子模块为新用户提供申请系统登录身份的界面。该功能采集用户的个人信息，并进行必要的合法性校验，符合条件后，将用户信息提交给服务器，等待系统管理员审批。用户获得批准以前，处于待审批状态，为最低等级。

用户注册子模块为新用户提供合法身份的申请。用户输入自定义的用户名，提交给服务器。服务器对用户名进行合法性以及唯一性校验成功后，返回用户名注册成功标志，并要求用户继续输入个人基本信息，以作为系统管理员审批的依据。得到用户的确认后，将信息提交给服务器。合法性校验成功后，将该用户信息以“待审批”状态录入数据库，并返回成功注册标志，同时写入日志，将注册信息以电子邮件方式发送给用户。处理过程如下：

1. 用户输入注册用户名；
2. 服务器采集用户名，校验合法性以及唯一性，成功后返回用户名注册成功标志，同时提供用户个人基本信息的输入界面；校验失败的操作，重新回到 1；
3. 用户输入个人基本信息，根据申请等级的高低，对个人信息详细程度的要求是不同的，设定必填项目与选填项目，得到用户的确认后提交给服务器；
4. 服务器采集用户输入的基本信息，进行合法性校验，返回必要的提示信息，同时强调电子邮件地址正确的输入；
5. 成功后返回用户注册成功等待审批标志，同时将用户数据写入数据库，用户状态置为“待审批”状态，并写入日志，将注册信息以电子邮件方式发送给用户；
6. 失败的操作，重新回到 3；
7. 对于没有完整执行用户注册子模块的操作，不受理用户的申请。

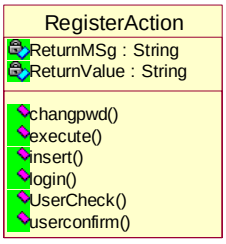



**图 4-8 用户注册子模块的类设计**

#### 4.3.6.3 用户登录子模块

用户登录子模块为用户提供系统登录的界面，并为登录成功的用户提供各种服务的链接按钮，同时将用户的登录信息写入日志。设定监视键盘鼠标动作的计数器，鼠标和键盘无动作开始倒计时，超过 30 分钟无动作自动返回用户登录界面，用户需要重新登录。用户登录子模块是用

户和系统管理员登录系统的主界面，显示系统的概要信息，同时根据用户级别的不同，给予用户不同的权限。

用户登录子模块为用户和系统管理员提供身份认证功能。用户输入用户名和口令，提交服务器。身份认证成功后，服务器返回成功标志、用户等级、用户状态等用户信息。对于待审批的用户，设定最低用户权限。子模块同时提供用户密码找回界面。处理过程如下：

1. 采集用户输入的用户名和口令，提交给服务器；
2. 对于待审批的用户，赋予最低用户权限，提交给服务器；
3. 服务器对于用户名和口令不一致的用户登录，提示输入新的用户名和密码；
4. 服务器对于合法用户返回相应的用户等级、用户状态等基本信息，允许进行符合其特征的动作；
5. 为用户提供密码找回功能，对于遗忘密码的合法用户，提供此功能。

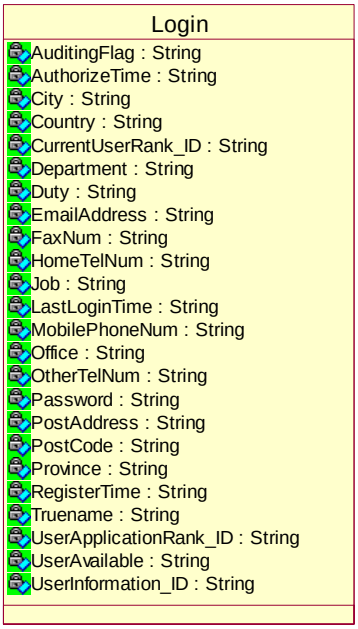



**图 4-9 用户登录子模块的类设计**

#### 4.3.6.4 用户信息变更子模块

用户信息变更子模块提供用户修改个人信息的界面。该功能从客户端采集用户修改的个人信息，完成合法校验后，提交给服务器，确认后提交给数据库。该功能同时提供密码修改、用户等级变更以及用户注销功能。

用户信息变更子模块从客户端采集用户修改的个人信息，包括用户密码、电话、地址等基本信息，但是不包括用户等级信息。对这些信息进行合法性校验，符合条件后，将用户信息提交给服务器，得到用户确认后，提交数据库，同时返回修改成功标记，并发送电子邮件告知用户已变更的信息。对于用户口令以及用户等级的更改进行特殊处理。对于用户等级的变更，需提交申请，在得到确认后，由系统管理员实施用户等级的更改。用户的注销申请，在系统管理员确认后，将该用户执行销户操作。处理过程如下：

1. 显示数据库中密码以外的用户基本信息；

2. 用户对基本信息进行修改，不包括密码修改以及用户等级修改；
3. 采集修改信息，提交服务器进行合法性校验；
4. 成功后，将修改信息存入数据库，并写入日志信息，同时给用户发送邮件；
5. 失败后，提示失败信息，返回用户基本信息编辑界面；
6. 提供密码修改按钮，提示用户输入旧密码、新密码以及确认新密码，提交服务器，进行合法性校验，并确认用户身份，成功后，将新密码存入数据库，并写入日志信息。失败后返回用户基本信息编辑页面；
7. 提供更改用户等级按钮，用户提交申请后，系统管理员在三个工作日内对其进行确认，并及时更改用户等级信息。
8. 提供用户注销按钮，系统管理员在用户提交注销申请后三个工作日内，确认用户的申请，成功后，从数据库中删除该用户。

#### 4.3.6.5 用户管理模块的表设计

该模块主要是对系统用户进行管理，需要创建一个表来存储系统的用户。创建表 UserManag，由它详细反映系统用户的各种特性和实体属性，来支持该模块对系统用户的管理。如下表所示。

**表 4-14 UserManag 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(16) | 否 | 关键字，用户的唯一标示号 |
| Password | varchar(16) | 否 | 用户密码 |
| Sex | varchar(2) | | 用户性别 |
| Email | varchar(40) | 否 | 用户的 Email 地址 |
| TrueName | varchar(40) | 否 | 用户真实姓名 |
| Country | varchar(40) | | 国家 |
| Province | varchar(40) | | 省份 |
| City | varchar(40) | | 城市 |
| PostAddress | varchar(100) | | 邮政地址 |
| PostCode | varchar(20) | | 邮政编码 |
| Department | varchar(20) | 否 | 用户所属部门 |
| Office | varchar(20) | | 用户办公室 |
| Duty | varchar(20) | 否 | 用户职务 |
| Job | varchar(20) | 否 | 用户职称 |
| AuditingFlag | varchar(8) | 否 | 审批标志 |
| RegisterTime | varchar(20) | 否 | 注册时间 |
| AuthorizeTime | varchar(20) | 否 | 授权时间 |
| UserAvailableFlag | varchar(8) | 否 | 用户是否可用标志 |
| LastLoginTime | varchar(20) | | 最后登录时间 |
| HomeTelNum | varchar(20) | | 用户家庭电话号码 |
| OfficeTelNum | varchar(20) | 否 | 用户办公电话号码 |
| MobilePhoneNum | varchar(20) | | 用户移动电话号码 |

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| FaxNum | varchar(20) | | 用户传真号码 |
| OtherTelNum | varchar(20) | | 其他电话号码 |
| CurrentUserRank_ID | varchar(8) | 否 | 用户级别代码 |
| UserApplicationRank_ID | varchar(8) | 否 | 用户申请级别代码 |

### 4.3.7 权限管理模块的设计

主要负责不同权限的系统用户对系统资源和企业资源访问控制的定义。在 Omate 中，每个用户拥有一个角色，通过定义角色限定了可访问的资源。用户必须先登录系统，才能使用与这种身份对应的资源。

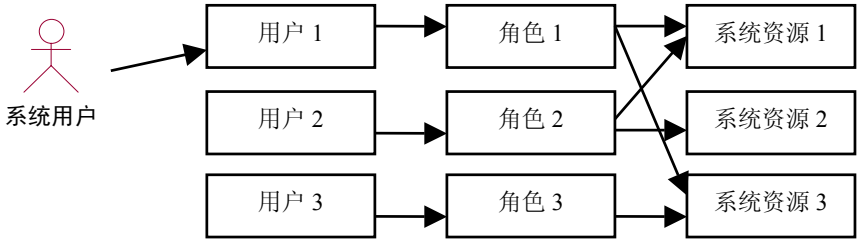



**图 4-10 系统用户、角色和系统资源的关系**

- **资源**

可以使用的系统资源。如注册用户、修改用户信息等都是资源。

- **角色**

绑定可操作资源的集合。比如系统管理员，可以使用全部资源，而一般用户可以注册和修改查看自己的信息。角色可以有依赖角色，即系统中存在 RoleA、RoleB、RoleC 三种角色，如果 RoleC 依赖 RoleA，则 RoleC 默认拥有 RoleA 可使用的资源。

- **用户**

系统中的使用者。角色之间可以依赖，因此，一个用户只有一种角色。

上述三个概念之间的绑定关系为用户绑定一个角色，角色绑定若干资源。设计四个表：资源表、动作表、动作资源关联表和用户权限表，表结构如下所示：

**表 4-15 Resource 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Resource_ID | varchar(16) | 否 | 关键字，资源的唯一标示号 |
| Name | varchar(64) | 否 | 资源名称 |
| Is_Need_Auth | varchar(16) | 否 | 是否需要验证 |
| Is_Admin_Access | varchar(16) | 否 | 是否属于管理员权限 |

**表 4-16 Actions 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Sign | varchar(16) | 否 | 关键字，动作类型 |
| Name | varchar(16) | 否 | 动作名称 |

**表 4-17 ResAct 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| Resource_ID | varchar(16) | 否 | 关键字，部门的唯一标示号 |
| Sign | varchar(16) | 否 | 关键字，动作类型 |

**表 4-18 RightSetting 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| UserInformation_ID | varchar(16) | 否 | 关键字，用户的唯一标示号 |
| Account_ID | varchar(16) | 否 | |
| Resource_ID | varchar(16) | 否 | 资源名称 |
| Action | varchar(16) | 否 | |

### 4.3.8 系统日志管理模块的设计

主要负责系统监控和日常维护等工作。用户从登录到退出系统的每个动作都记录在系统日志中，如登录、业务申请、业务审批、回复帖子等。系统管理员可以查看系统日志，查找出错原因和出错时间，从出错时间恢复系统。

系统日志主要记录系统运行信息，包括运行错误，以文件方式存储。系统日志底层封装了 Log4j，可以根据系统需求改写 Log4j 的记录方式。业务日志主要记录用户的业务操作，包括用户对系统数据操作的成功信息和失败信息。日志数据统一记录在数据库中，日志维护工具提供对日志数据的维护机制，如日志数据的删除和迁移等。系统中发生的事件都在系统日志中有记录，记录系统日志集中在对数据的操作上。

**表 4-19 SysLog 的表结构**

| 字段名称 | 字段类型 | 是否为空 | 说明 |
|---|---|---|---|
| SysLog_ID | varchar(16) | 否 | 关键字，系统日志的唯一标示号 |
| UserInformation_ID | varchar(40) | 否 | 该事件的发起用户 ID，引用自 UserManag 表 |
| Name | Varchar(100) | 否 | 事件名称 |
| Content | Varchar(500) | 否 | 事件的具体内容 |
| Date | Varchar(20) | 否 | 事件的发生时间，精确到以秒为单位 |

## 4.4 本章小结

本章介绍了 Office Mate 企业办公系统的设计。该系统包含四个模块：日常办公、日常业务、综合信息和系统管理。结合 Struts 框架在系统实现中运用研究的需要，主要设计和实现了电子日历、通讯簿、会议安排、考勤管理、用户管理、权限管理、系统日志管理等几个模块。

# 第五章 Struts 框架在 Office Mate 企业办公系统实现中的运用研究

## 5.1 系统的模块配置

在实际的开发过程中，开发人员可能同时需要修改配置文件。为了解决并行开发的问题，Struts 1.1 提供了两种解决方案：一是多配置文件，一是多应用模块。第一种方式为 ActionServlet 指定多个配置文件。

```
<servlet>
   <servlet-name>action</servlet-name>
   <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
   <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml,
                   /WEB-INF/struts-config-usermanag.xml
      </param-value>
   </init-param>
   …
</servlet>
```

这样，开发人员可以在不同的配置文件中设置 Action、ActionForm 等，但是仍然潜在冲突。ActionServlet 初始化时，struts-config.xml、struts-config-usermanag.xml 等配置文件最终还是要合并到一起。如果在 struts-config.xml 和 struts-config-usermanag.xml 中配置了同名的<forward>元素，执行时就会发生冲突。

Struts 1.1 引进了多应用模块的方法，彻底解决了这类冲突问题。一个模块中包含多个子应用，每个子应用可以处理相关的一组功能，拥有自己的 Struts 配置文件、JSP 页面、Action 类等等。ActionServlet 将不同的模块信息保存在不同的 ModuleConfig 对象中。在各自的模块中进行所需配置，不会和其它配置文件发生冲突。

Omate 的实现采用多应用模块方式的方式。在缺省模块的基础上，添加 Calendar（电子日历）、Contact（通讯簿）、TimeTracking（考勤管理）、UserManag（用户管理）等模块。对各个新添加的模块建立对应的目录，每个模块分布在各自的目录下，有助于定位、开发和部署资源。UserManag 模块的资源（如 ActionForm、Action、JSP 等）放在%UserManag%目录下，TimeTracking 模块的资源放到%TimeTracking%目录下，缺省目录的资源放在 ROOT 下和其他模块类路径的上级目录。对于多个应用模块的开发方式，进行以下配置工作：

- **为每个子应用创建单独的 Struts 配置文件**

每个应用模块都有自己的 struts 配置文件。struts-config.xml 是缺省应用模块的配置文件。Omate 系统的模块各自创建了配置文件，如 struts-config-usermanag.xml 、struts-config-timetracking.xml 、 struts-config-calendar.xml 、 struts-config-contact.xml 等。其中 struts-config-usermanag.xml 是用户管理模块的配置，依此类推。

- **配置 web.xml 文件**

Omate 在 web.xml 文件进行如下配置，将这些信息通知 ActionServlet。

```
<servlet>
   <servlet-name>action</servlet-name>
   <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
   <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
   </init-param>
   <init-param>
      <param-name>config/UserManag</param-name>
      <param-value>/WEB-INF/struts-config-usermanag.xml</param-value>
   </init-param>
   <init-param>
      <param-name>config/TimeTracking</param-name>
      <param-value>/WEB-INF/struts-config-timetracking.xml</param-value>
   </init-param>
   …
</servlet>
```

第 1 个<init-param>的<param-name>属性为 config，表示默认的应用模块，第 n（n>1）个<init-param>表示增加的应用模块，<param-name>表示为“config/模块名”的形式，<param-value>指定模块所对应的 Struts 配置文件。

- **为各个应用准备所需的资源**

模块的资源（ActionForm、Action 和 JSP 等）放在各自的目录下。有两种方法可以实现不同模块间的转发，一种是在<forward>（全局或者局部）中定义，另一种是利用 SwitchAction 类。在 Omate 系统实现中，使用<forward>转发的方法。

系统用户成功登录后，进入 Omate 系统的主界面（mainmenu.jsp），在此可以链接到系统的各功能模块。mainmenu.jsp 通过<html:link forward=”UserInformationpage”>链接转到用户信息管理子模块，在 mainmenu 所属的缺省模块的 struts-config.xml 文件中配置如下：

```
<global-forwards>
      <forward name="UserInformationpage" contextRelative="true"
                path="/UserManag/UserInformationpage.do" redirect=”true”/>
      ...
</global-forwards>
```

即在原有的 path 属性前加上模块名，同时将 contextRelative 属性设置为 true，表示 path 属性的值是整个应用范围（false 表示模块范围）。在 struts-config-usermanag.xml 中接受跳转的配置为：

```
<action-mappings>
    <action path="/UserManag" forward="/UserInformation.jsp"/>
    …
</action-mappings>
```

也可以在<action>中定义一个类似的局部<forward>。如果从其它模块转回到缺省模块，使用类似如下定义。

```
<action-mapping>
   <action>
   …
      <forward name="success" contextRelative="true"
                path="/UserManag/UserInformationpage.do" redirect="true"/>
   </action>
…
</action-mapping>
```

从默认模块切换到 UserManag，采用类似如下 URL：

http://localhost:9999/toModule.do?prefix=/UserManag&page=/index.do

从 UserManag 切换到默认模块，采用类似如下 URL：

http://localhost:9999/toModule.do?prefix=&page=/index.do

## 5.2 用户管理模块的实现

用户注册子模块的结构如图 5-1 所示。处理过程是：用户通过页面链接进入用户注册模块。用户填写相关注册信息，然后由 HTTP 请求提交给 ActionServlet，ActionServlet 会寻找相应的 RegisterForm 和 RegisterAction。首先将提交的 request 对象映射到 form 中，进行请求数据的填充处理；然后将 form 传递给 action 进行处理。Action 得到 form 后，对 mapping、form、request、response 等对象调用 execute()方法。execute()调用业务层接口方法，实现与模型对象 RegisterBean 的通信。RegisterBean 执行业务逻辑，通过 JDBC 与数据库通信，将用户信息写入数据库中，然后由 execute()方法返回一个 forward URL 给 ActionServlet，最终由 display.jsp 显示用户注册的结果。

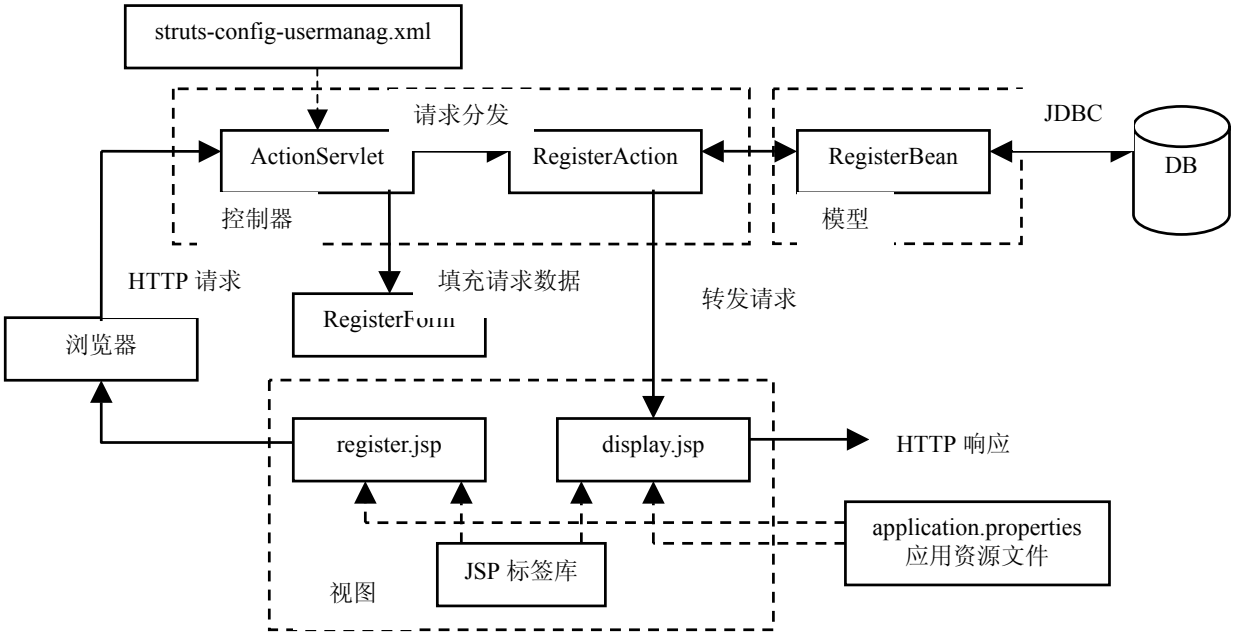



**图 5-1 用户注册子模块的结构图**

### 5.2.1 构建视图

register.jsp、display.jsp 和 JSP 标记库构成了用户注册部分的视图。register.jsp 是与用户的交互页面，采集用户注册信息（包括用户名、电子邮件、申请用户级别等等）。display.jsp 显示注册完成后的用户信息。利用 Struts 标签库，通过 struts-config-usermanag.xml 配置的应用资源文件 application.properties 内的一些关键字/值对来构建全部 JSP 页面表单，页面内不含 Java 代码。以 register.jsp 为例，说明 Struts 中 JSP 页面的设计。

首先，引入需要用到的 JSP 标签库。

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
```

<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>中的 URI 是逻辑引用，标签库描述符（TagLib Definition，简称 TLD）的位置在 web.xml 文件中给出，prefix 是前缀，JSP 中的每个 Struts 标记通过前缀引用相关的标记库描述符。JSP 页面中使用的 Struts 的标记库分为 6 类，register.jsp 引入了 html、bean 和 logic 标签。

用户填入注册消息时，有些输入域不能为空，否则 RegisterForm 无法通过验证，将在当前输入页面显示错误信息。<html:errors/>标记加工此错误消息。如果没有错误消息，则标记不输出。<html:errors/>标记与 ActionErrors 结合显示错误信息。这个标记首先从当前模块的资源文件中读取消息关键字 errors.header，然后显示消息的文本。接着在 ActionErrors 对象（通常作为请求参数存储在 Action.ERROR_KEY 关键字下）中循环，读取单个 ActionError 对象的消息关键字，从当前模块的资源文件中读取并格式化相应的信息，并且显示它们。然后读取它与 errors.footer 关键字相对应的消息并且显示出来。

<html:form>标记产生一个 HTML 表单供数据输入。

```
<html: form action="/register.do" focus="UserInformation_ID">
```

action 属性是对 struts-config-usermanag.xml 配置文件中 ActionMapping 的引用。它通知由哪个 JavaBean 助手类来组装 HTML 组件。JavaBean 助手基于 ActionForm 类。

在 registerjsp 中，与表单相关的操作路径是%UserManag%，对该表单而言，ActionForm Bean 的其他信息，包括 bean 的名称、类型、作用域，都是从 struts-config-usermanag.xml 配置文件的<action-mapping>和<form-beans>定义部分检索到的。

用户进行输入时的提示消息和接受输入的文本输入域的代码为：

```
<th align="right"><bean:message key="UserManag.UserInformation_ID"/></th>
<td align="left"><html:text property="UserInformation_ID" size="16"/></td>
```

<bean:message>标记检索并输出一个国际化的消息字符串。消息的关键字/值对在资源文件 application.properties 中定义。消息关键字 key="usermanag.UserInformation_ID"，在 application.properties 中的关键字/值为“usermanag.UserInformation_ID＝用户名”，运行时将输出消息文本“用户名”。

<html:text>标签创建一个 HTML 输入组件供文本域输入，用 ActionForm 的 UserInformation_ID 属性填充该输入域。

为防止用户不登录而直接输入地址访问，每次都回到 login.jsp。

以下是通过浏览器看到的效果图：

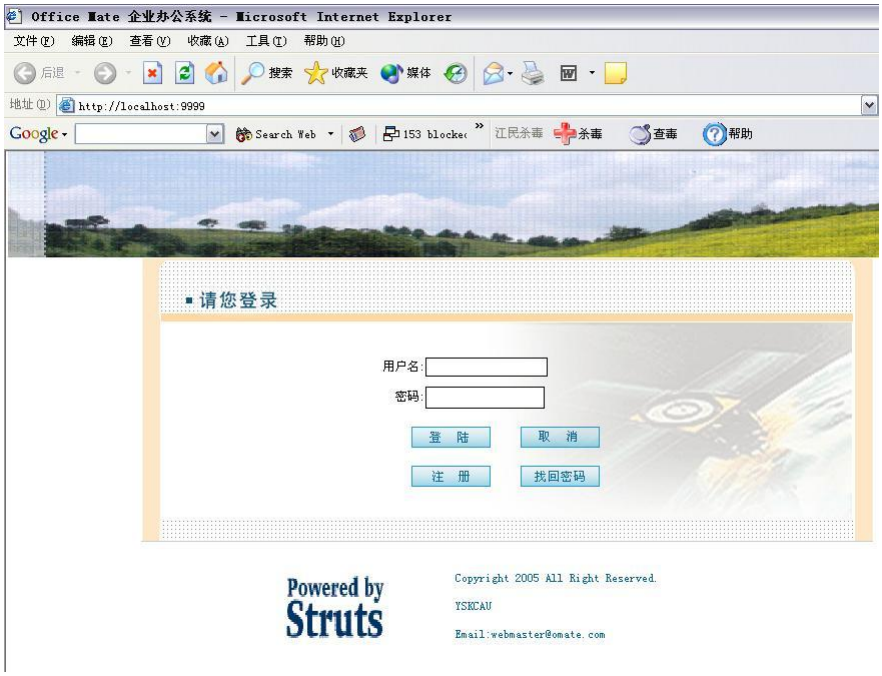



**图 5-2 用户登录界面**

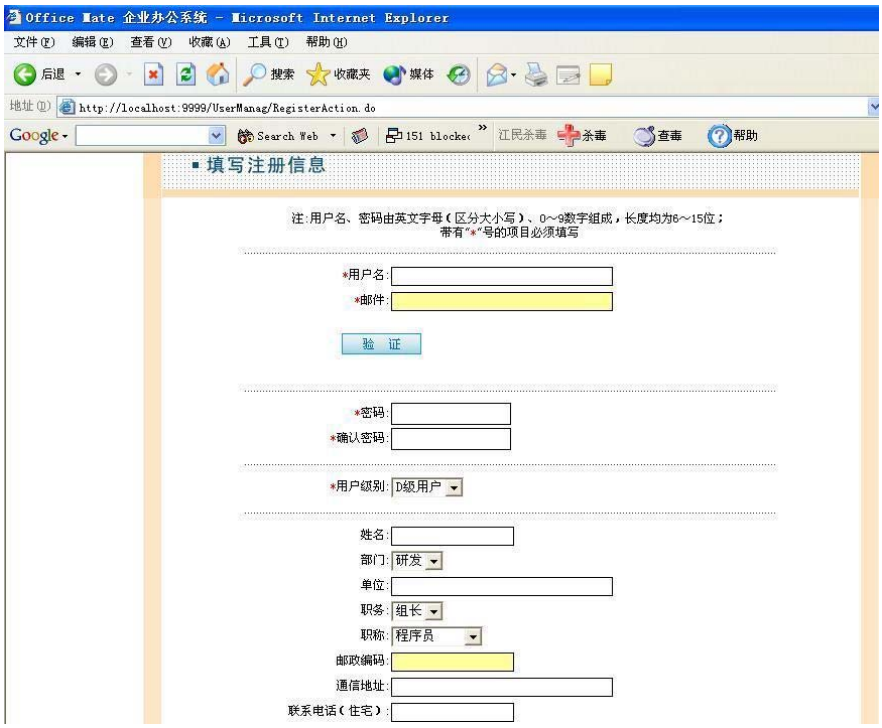



**图 5-3 用户注册界面**

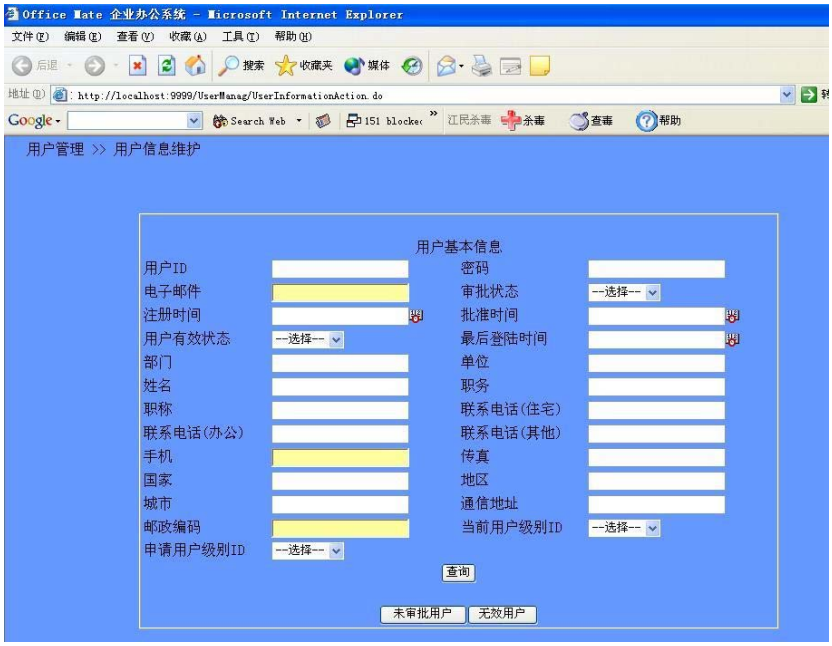



**图 5-4 用户管理界面**

表单提交时，涉及两个框架对象：ActionForm 和 Action，实现细节需要开发人员编写。ActionServlet 使用 Struts 配置来决定使用哪个 ActionForm 和 Action，属于控制器范围。

### 5.2.2 构建控制器

ActionServlet 接收所有来自客户端的请求，并根据 struts-config.xml 将 HTTP 请求传送给 Action 对象。ActionServlet 由框架本身提供，一般无需对其进行再次开发。Action 类是应用开发的重点，负责事件的流程控制。ActionForm 位于视图和控制器之间，不访问模型组件，在构建控制器部分讨论。

Register 的视图在配置文件中引用了 RegisterForm 和 RegisterAction 两个对象。

创建视图时，<html:form>标签与 struts 配置紧密配合，由 action 参数通知<html:form>标签采用哪个 ActionMapping。

```
<html: form action="/register.do" focus="UserInformation_ID">
```

Struts 配置文件中的<action-mappings>元素的<action>元素配置如下：

```
<action    path="/register" type="com.omate.usermanag.RegisterAction"
           name="RegisterForm" scope="request" input="/register.jsp" validate="true">
      <forward name="displayRegResult" path="/display.jsp"/>
</action>
```

path 属性 mapping 的唯一标识，是 Action 处理的 URL，如 http://localhost:9999/omate/register.do。type=”com.omate.usermanag.RegisterAction”表明/register 路径被请求时调用的 Action 对象是 com.omate.usermanag.RegisterAction。name＝”RegisterForm”表明和 HTML 表单一起使用的 ActionForm 类的名称是 RegisterForm，与<form-bean>元素的 name 属性匹配。scope=”request”指定 RegisterForm 的生命周期。validate=”true”表明在调用 RegisterForm 对象的 execute()方法前，

ActionServlet 将调用 validate()方法验证输入数据的有效性。input=”/register.jsp”表明验证失败后，将控制发到/register.jsp。<forward>元素表示 RegisterAction 的 execute()方法返回名为 displayRegResult 的 ActoinForward 对象，控制转到/display.jsp。form-bean 的配置如下，该元素使 RegisterForm 的逻辑名字和 com.omate.usermanag.RegisterForm 关联。

```
<form-beans>
        <form-bean name="RegisterForm" type="com.omate.usermanag.RegisterForm">
</form-beans>
```

- **RegisterForm**

当 HTML 表单提交时，关键字/值对被控制器获取，并应用到 ActionForm 中。HTML 表单中的字段和 ActionForm 中的属性一一对应。Struts 比较 ActionForm 属性的名称和输入关键字/值对的名称。如果匹配，控制器设置属性值为相关的输入域的值，其他属性会被忽略。忽略的属性保持其缺省值。RegisterForm 中定义的公共属性如下，分别和 register.jsp 产生的 HTML 表单中的输入文本域相对应。

```
private String UserInformation_ID=null;//用户名
private String PassWord=null;//用户密码
private String EmailAddress=null;//用户电子邮件地址
private String AuditingFlag=null;//审批标志
private String RegisterTime=null;//用户注册时间
…
```

RegisterForm 提供了访问和设置这些属性的 getter 和 setter 方法。如 EmailAddress 属性的 getter 和 setter 方法为：

```
//获取用户电子邮件地址
public String getEmailAddress(){
      return EmailAddress;
}
public void setEmailAddress (String EmailAddress){
this.EmailAddress =DBUtilities.getUnicodeStr(EmailAddress);
}
```

getEmailAddress()方法通过 DBUtilities.getUnicodeStr(String str)对参数 EmailAddress 进行了处理，以显示中文信息。

```
//使 EmailAddress 显示汉字信息。
public static String getUnicodeStr(String str)
{
    byte b[]=str.getBytes("UTF-8");
   str=new String(b);
   return str;
}
```

该方法中，首先将获取的字符串用 UTF-8 进行编码，并将编码存放到一个字节数组中，然后将这个数组转化为字符串对象。这样提交的任何信息（无论是汉字字符或西欧字符）都能正确地显示。

RegisterForm 调用对应的 setter 方法，对各个状态属性赋值后，调用 validate()方法检查请求参数的有效性。

```
//校验用户名、用户密码等输入域的有效性
public ActionErrors validate(ActionMapping mapping,HttpServletRequest request){
   ActionErrors errors=new ActionErrors();
   if(UserInformation_ID==null||username.equals("") ){
    errors.add("UserInformation_ID",
              new ActionError("error.usermanag.UserInformation_ID.required"));
}
…
   return errors;
}
```

validate 方法对请求参数进行简单验证，当输入域没有输入数据时，系统将产生一个 ActionError 对象，视图页面通过<html:error/>显示。

error.usermanag.UserInformation_ID 等都是消息关键字，用于从 application.properties 中查找实际的消息。如果用户名不填，通过消息关键字 error.usermanag.UserInformation_ID.required 在输入页面产生“用户名不能为空”的消息。

- **RegisterAction**

RegisterForm 收集数据并执行一些初始化校验后，控制器将 RegisterForm 传递给 Mapping 标明的 RegisterAction 类，RegisterAction 将 RegisterForm 收集的数据传递给业务层方法，然后由业务层完成用户注册数据写入数据库的操作。

在 RegisterAction 中，使用一个辅助方法 isRegisterSuccess 调用业务层方法。虽然可以将代码放在 Action 的 execute()方法中，但是考虑到结构清晰，决定将通用业务方法和控制器代码分开。

```
//调用业务层方法，与 RegisterForm 交互
public boolean isRegisterSuccess(String UserInformation_ID, String PassWord, String EmailAddress,
                                 String AuditingFlag, String RegisterTime…){
    return(new RegisterBean().isRegisterSuccess(
                    UserInformation_ID, PassWord, EmailAddress,AuditingFlag, RegisterTime…));
}
```

该方法调用 RegisterBean 的 isRegisterSuccess 方法，最终完成数据库写入操作。Action 将输入从视图层带到业务层，这里从 ActionForm 中获取 UserInformation_ID、PassWord、EmailAddress、AuditFlag、RegisterTime 等等，将它们存为平面的 String：

```
//从视图层获取用户名、用户密码等
RegisterForm theForm=(RegisterForm)form;
String UserInformation_ID=theForm.getUserInformation_ID.trim();
String PassWord=theForm.getPassWord.trim();
String EmailAddress=theForm.getEmailAddress.trim();
String AuditingFlag=theForm.getAuditingFlag.trim();
String RegisterTime=theForm.getRegisterTime.trim();
…
```

用户成功登录后，将 UserInformation_ID、AuditFlag 等等设置为 session 属性，在以后的操作中就可以使用该属性。在 RegisterAction 中，获取 UserInformation_ID 的代码如下：

```
HttpSession sessioin = request.getSession(true);
String user=(String)session.getUserInformation_ID(UserInformation_ID);
```

然后将 UserInformation_ID 等传递给业务层接口方法 isRegisterSuccess，并将处理结果保存在

request 对象中，返回 ActionForward 对象，由控制器将请转发到 display.jsp。

```
String result="处理失败"
if(isRegisterSuccess(UserInformation_ID,PassWord, EmailAddress, AuditFlag, RegisterTime…))
result=("处理成功");
request.setAttribute("result",result);
return mapping.findForward("displayRegResult");
```

### 5.2.3 构建模型

根据 MVC 的分层原则，应将业务逻辑执行分离到单独的 JavaBean 中。而 Action 只负责错误处理和流程控制，应作为控制器的一部分。考虑到其重用性，在执行业务逻辑的 JavaBean 中不引用任何与 Web 应用相关的对象，如 HttpServletRequest、HttpServletResponse 等，而是将其转化为普通的 Java 对象。

用户注册模块的模型由 RegisterBean 实现，负责业务逻辑的执行，通过 JDBC 与数据库通信，将注册信息写入 OmateDB 数据表中。RegisterBean（逻辑 Bean）中有一套和 RegisterForm（ActionForm Bean）完全相同的属性——输入的注册数据信息，其数据的获取是通过 Action 的业务接口方法 isRegisterSuccess 传递的。虽然 ActionForm Bean 和逻辑 Bean 最终都是表达相同的数据，但是它们在各自的生命周期内表达数据的不同方面。逻辑 Bean 表达的是模型的状态，ActionForm Bean 表达状态的改变。ActionForm Bean 收集和校验输入，业务逻辑 Bean 处理捕获的数据并将新数据合并到模型中。在 RegisterBean 中，被 Action 调用的方法 isRegisterSuccess 代码如下：

```
//与控制器 RegisterAction 交互
public boolean isRegisterSuccess(UserInformation_ID, PassWord, EmailAddress,
                                 AuditFlag, RegisterTime…){
   setUserInformation_ID(UserInformation_ID);
   setEmailAddress(EmailAddress);
   setAuditFlagAuditFlag);
   setRegisterTime(RegisterTime)'
   …
   boolean result=dbMySQL(UserInformation_ID);
   return result;
}
```

isRegisterSuccess()接受传入的参数，用 setter 方法设置对应的属性，然后调用该 Bean 中的 dbMySQL 方法将数据插入到数据库中。

dbMySQL 通过 JDBC 连接池实现数据库操作，它利用从 Action 传入的 UserInformation_ID，首先在 OmateDB 中查找该用户，然后连同获取的其他属性作为 OmateDB 中的一条记录插入。

### 5.2.4 构建持久层

对象只能存在于内存中，如需永久保存对象的状态，需要进行对象的持久化，即把对象存储到数据库中。如 4.2.2 节所述，Omate 采用 MySQL 关系型数据库存储数据。关系型数据库中存放的是关系数据，将业务对象映射到关系数据库中，会出现阻抗不匹配（impedance mismatch）[36]。因为对象由状态和行为组成，而关系型数据库由表组成，无法直接表示对象间多对多的关联和继

承关系。

传统的三层结构中，业务逻辑层负责业务逻辑，而且直接访问数据库。为了把数据细节和业务逻辑分开，将数据访问作为单独的持久层，负责存储从应用到数据库的数据以及数据的检索、更新和删除。持久层封装了数据访问细节，为业务逻辑提供了面向对象的 API[64]。

在实际应用中，除了需要把内存中的对象持久化到数据库外，还要把数据库的关系数据库再加载到内存中，以满足用户查询业务数据的需求。频繁地访问数据库，会对应用的性能造成巨大影响。为了降低访问数据库的频率，可以把需要经常被访问的业务数据存放在缓存中，并且通过特定的机制来保证缓存中的数据和数据库中的数据同步。

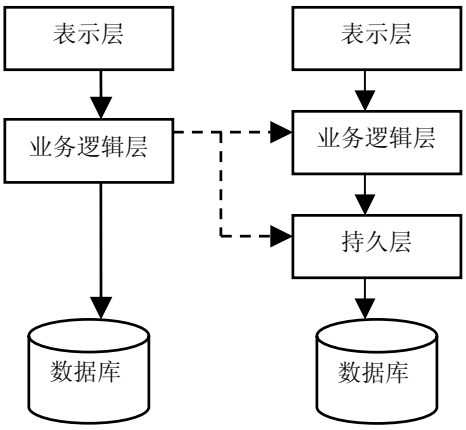



**图 5-5 持久层的位置**

Struts 框架没有指定实现持久层的框架。一般地，有 4 种方式[65]：JDBC、EJB、JDO（Java Data Object）、对象/关系映射工具（Object/Relation Mapping，简称 ORM）。总的来说，JDBC 面向关系型数据库，适合关系数据库模式驱动的应用，例如统计表格数据，生成报表之类的应用。EJB 技术以 J2EE 应用服务器为中心，需要灵活、可声明的事务边界，支持大容量的访问和不间断的服务，适合于应用服务器的集群。对于以域对象为中心的应用（如包含图，树模型），宜选用 JDO。ORM 使用于数据库表比较多，业务逻辑比较复杂，性能要求比较高的应用系统。

JDBC 是访问关系数据库最原始、最直接的方法，运行效率高，并且获得了几乎所有数据库厂商的支持。JDBC 是统一的标准接口，只要掌握标准的 SQL 语言就可以访问各种不同的数据库，增强了数据库间的可移植性。开发时选择一个最合适于应用的技术尤为重要。鉴于实际开发项目中的数据模型的具体情况，Omate 系统采用 JDBC 开发持久化层，把数据库访问操作封装起来，提供简洁的 API，供业务层统一调用。

数据库连接是一种有限的资源，在多用户的 Web 应用程序中尤为突出。对数据库连接的管理会显著影响整个应用程序的伸缩性和健壮性。在传统的数据库连接方式中，一个数据库对象均对应一个物理数据库连接，每次操作都打开一个物理连接，使用完关闭，造成系统性能低下。数据库连接池的解决方案是在应用程序启动时建立足够的数据库连接，并将这些连接组成一个连接池，应用程序动态地对池中的连接进行申请、使用和释放。对应多于连接池中连接数的并发请求，在请求队列中等待。应用程序可根据池中连接的使用率，动态增加或减少池中的连接数。

JDBC 3.0 规范支持数据库连接池，仅仅规定了如何支持连接池的实现，没有规定连接池的具体实现。开发人员可以通过这个框架使不同角色的开发人员共同实现数据库连接池。

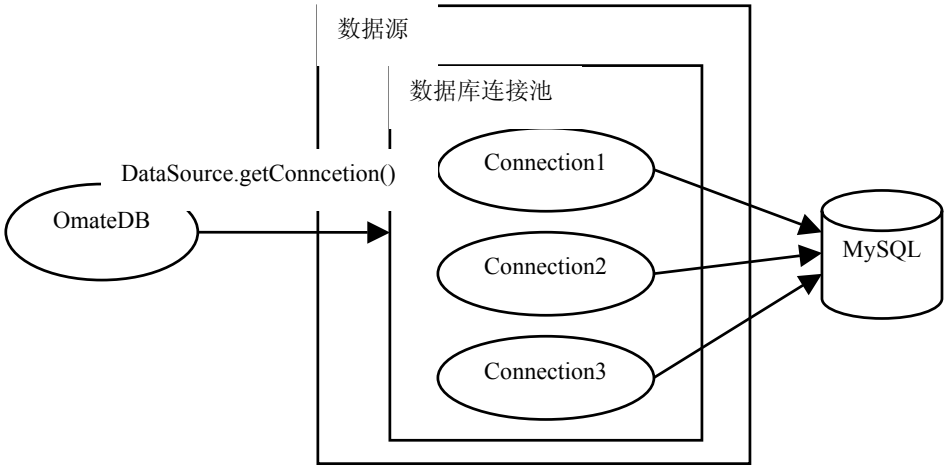



**图 5-6 OmateDB 通过数据源访问数据库**

在 Omate 中，采用 Tomcat 容器提供的数据库连接池技术实现应用系统与 MySQL 数据库系统的连接使用。

#### 5.2.4.1 配置连接池

Tomcat 控制台配置详细信息如下：

```
JNDI Name: jdbc/mysql          //采用 Java 的 JNDI 技术获得对 DataSource 对象的引用
Data Source URL: jdbc:mysql://localhost:3306/OmateDB?autoReconnect=true
JDBC Driver Class: com.mysql.jdbc.Driver
User Name:root
Password: ********
Max. Active Connections: 100
Max. Idle Connections: 30
Max. Wait for Connection: 10000
Validation Query:
```

server.xml 中配置 Tomcat 标准数据源如下：

```
<Context path="/Omate" docBase="Omate" debug="0" reloadable="true">
〈!--声明连接池--〉
<Resource name="jdbc/OmateDB" auth="Container" type="javax.sql.DataSource"/>
〈!--对连接池的参数进行设置--〉
<ResourceParams name="jdbc/OmateDB">
    <parameter>
      <name>factory</name>
      <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
…
</Context>
```

修改 web.xml，声明对 JNDI Resource 的引用。

```
<resource-ref>
        <description>DB Connection</description>
        <res-ref-name>jdbc/OmateDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
</resource-ref>
```

#### 5.2.4.2 使用连接池

在 Omate 中通过数据源访问数据库。将查找数据源和访问连接的操作封装起来，在 Acion 类或 JavaBean 类中，采用以下方式获得数据库连接池的一个连接和释放这些连接。

```
Context initCtx = new InitialContext();
Context ctx = (Context) initCtx.lookup("java:comp/env");//获取连接池对象
DataSource ds = DataSource ctx.lookup("jdbc/OmateDB");
Connection conn=ds.getConnection();
Statement stmt=
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,  resultSet.CONCUR_UPDATABLE);
//使用此连接存取数据库表，将连接返回给连接池
conn.close();
```

## 5.3 系统实现中关键问题的解决方案

### 5.3.1 表单重复提交的解决方案

如果用户多次提交同一个 HTML 表单，则 Web 应用能够判断用户重复提交的行为，并做出相应处理。例如，对于注册表单，如果用户已经提交表单并且服务器端成功地注册了用户信息，此时用户又通过浏览器的后退功能，退回到原来的页面，重复提交表单，服务器端应该能够识别用户的误操作，避免为用户重复注册。

Struts 提供的 Token（同步令牌）机制能够很好地解决表单重复提交的问题，其基本原理是：服务器端在处理到达的请求之前，会将请求中包含的令牌值与保存在当前用户会话中的令牌值进行比较，是否匹配。处理完该请求，且在答复发送给客户端之前，将会产生一个新的令牌，该令牌除了传给客户端以外，也会将用户会话中保存的旧令牌进行替换。这样如果用户回退到刚才的提交页面并再次提交的话，客户端传过来的令牌和服务器端的令牌不一致，有效地防止了重复提交的发生。

Action 类中提供了几个和 Token 相关的方法：

- protected boolean isTokenValid(javax.servlet.http.HttpServletRequest request)
  判断存储在当前用户会话中的令牌值和请求参数中的令牌值是否匹配。如果匹配，就返回 true，否则返回 false。
- protected void resetToken(javax.servlet.http.HttpServletRequest request)
  从当前 session 范围内删除令牌属性
- protected void saveToken(javax.servlet.http.HttpServletRequest request)
  创建一个新的令牌，并把它保存在当前 session 范围内。如果 HttpSession 对象不存在，就先创建该对象。

具体的 Token 处理逻辑由 org.apache.struts.util.TokenProcessor 类来完成，其中的 generateToken(request)方法根据用户会话 ID 和当前系统时间来生成一个唯一的令牌。

下面以通讯簿模块为例，介绍在 Omate 系统中如何利用 Token 机制避免重复提交表单的过程。用户请求 insert.jsp 之前，首先把请求转发给 PrepareInsertAction，它调用 saveToken(request)方法，创建一个新的令牌，并把它保存在当前的 session 范围内。PrepareInsertAction 接着再把请求转发

给 insert.jsp。insert.jsp 中<html:form>标签的处理类判断在 session 范围内是否存在 Token，如果存在，则在表单中生成一个包含 Token 信息的隐藏字段，由 org.apache.struts.taglib.html.FormTag 类的 renderToken()方法完成：

```
protected String renderToken(){       //在表单中生成一个包含 Token 信息的隐藏字段
    StringBuffer results = new StringBuffer();
    HttpSession session = pageContext.getSession();
    if(session!=null){
        String token = (String)session.getAttribute(Globals.TRANSACTION_TOKEN_KEY);
    if(token!=null){
        results.append("<input type=\"hidden\" name=\"");
        results.append(Constants.TOKEN_KEY);
        results.append("\" value=\"")
        results.append(token);
        if(this.isXhtml()){
           results.append("\"/>");
           }else{
                   results.append("\">");
               }
        }
    }
}
```

用户收到 insert.jsp 后，在源文件中会看到表单定义了一个包含 Token 消息的隐藏字段：

```
<form name="insertForm" method="post" action="/contact/insert.do">
   <input type="hidden" name="org.apache.struts.taglib.html.TOKEN"
     value="6aa35341f25184fd996c4c918255c3ae">
…
</form>
```

Action 类的 isTokenValid(request)方法在判断 Token 是否有效时，把这个隐藏字段的值和当前用户会话中的令牌值比较。

用户提交表单后，由 InsertAction 处理请求。在 InsertAction 中，首先调用 isTokenValid(request)方法，判断当前用户会话中的令牌值和请求参数中的令牌值是否匹配。如果不匹配，生成错误消息，并调用 saveToken(request)方法，创建一个新令牌，然后返回；如果匹配，调用 resetToken(request)方法，从当前会话中删除 Token，然后执行插入数据的操作。

这样，用户提交表单之后，如果又通过浏览器的后退功能，退回到刚才的 insert.jsp，再次提交表单，该请求将由 InsertAction 来处理。InsertAction 先调用 isTokenValid(request)方法判断当前用户会话中的令牌值和请求参数中的令牌值是否匹配。此时，由于当前会话中已经不存在 Token，isTokenValid(request)方法返回 false。

PrepareInsertAction 的 execute()方法代码如下：

```
public ActionForward execute ( ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception{
             saveToken(request);
             return mapping.findForward(Constants.FORWARD_SUCCESS);
}
```

修改 InsertAction，在 execute()方法的开头加入判断 Token 是否有效的逻辑：

```
public ActionForward execute(ActionMapping mapping,
                             ActionForm form,
                             HttpServletRequest request,
                             HttpServletResponse response)
        throws Exception{
                ActionMessages errors = new ActionMessages();
                if (!isTokenValid(request) {
                    errors.add(ActionMessage.GLOBAL_MESSAGE,
                               new ActionMessage("error.invalid.token"));
                    saveErrors(request,errors);
                    saveToken(request);
                    return(new ActionForward(mapping.getInput()));
                }
                else {
                resetToken(request);
                }
                // insert the record
}
```

根据以上代码，当用户首次提交表单时，isTokenValid()方法返回 true。如果又通过浏览器的后退功能，返回到刚才的 insert.jsp，再次提交表单，isTokenValid()方法返回 false，并生成错误消息，调用 saveToken()方法创建一个新的 Token，然后返回到原来的 insert.jsp。如果此时用户再提交表单，则 isTokenValid()方法又会返回 true。修改 struts-config.xml，配置转发关系：PrepareInsertAction→insert.jsp→InsertAction。对于<global-forwards>元素中的<forward>子元素修改如下：

```
<forward name="insert" path="/prepareinsert.do">
```

配置 PrepareInsertAction：

```
<action path="/prepareinsert"
        type="contact.actions.PrepareInsertAction">
  <forward name="success" path="/index.jsp"/>
</action>
```

修改资源文件，在 ApplicationResources.properties 文件中加入验证 Token 无效时的错误消息：

```
Error.invalid.token=<li>Not Allowed to Submit the Same Form Twice</li>
```

## 5.3.2 权限管理的解决方案

### 5.3.2.1 Servlet 过滤器

Servlet 过滤器在 Java Servlet 规范 2.3 中定义，对 Servlet 容器的请求和响应对象进行检查和修改。Servlet 过滤器负责过滤的 Web 组件可以是 Servlet、JSP 或 HTML 文件。Servlet 过滤器中结合了许多元素，使得过滤器成为独特、强大和模块化的 Web 组件。Servlet 过滤器通过一个 XML 配置文件来灵活声明模块化的可重用组件。过滤器动态地处理传入的请求和传出的响应，无需修改应用程序代码就可以添加或删除，并且独立于平台，可以轻易地部署到 J2EE 应用中。

过滤器本身并不生成请求和响应对象，只提供过滤作用，是自包含、模块化的组件，可以将其添加到请求/响应链中。过滤器只是改动请求和响应的运行时处理，可以通过 Servlet API 中良好定义的标准接口来实现。

Web 资源可以配置为无过滤器关联（默认情况）、与单个过滤器关联（典型情况），或者与一个过滤器链相关联。和 servlet 一样，过滤器接受请求并响应对象，检查请求对象，并决定将该请求转发给链中的下一个组件，或者中止该请求并直接向客户端发回一个响应。如果请求被转发了，它将被传递给链中的下一个资源（另一个过滤器、servlet 或 JSP 页面）。在这个请求设法通过过滤器链并被服务器处理之后，响应将以相反的顺序通过该链发送回去。这样就给每个过滤器都提供了根据需要处理响应对象的机会。

Servlet 2.4 规范中，过滤器可以应用于 J2EE Web 应用中存在请求和响应对象的任何地方。实现一个 Servlet 过滤器需要经历三个步骤。首先要编写过滤器实现类的程序，然后把该过滤器添加到 Web 应用程序中（通过在 web.xml 中声明），最后把过滤器与应用程序一起打包并部署。

**1. 编写实现类的程序**

过滤器 API 包含 3 个接口，位于 javax.servlet 包中，分别是 Filter、FilterChain 和 FilterConfig。过滤器类实现 Filter 接口，并使用 FilterChain 和 FilterConfig 接口。该过滤器类的一个引用将传递给 FilterChain 对象，把控制权传递给链中的下一个资源。FilterConfig 对象将由容器提供给过滤器，允许访问该过滤器的初始化数据。为了与三步模式保持一致，过滤器使用 init()、doFilter()、destroy()三个方法，以实现 Filter 接口。

在 Omate 中，使用以下过滤器实现了跟踪满足来自用户的 Web 请求所花的大致时间。

```
public class TimeTrackFilter implements Filter{
        private FilterConfig filterConfig = null;
        public void init(FilterConfig filterConfig)
            throws ServletException{
            this.filterConfig = filterConfig;
        }
        public void destroy(){
            this.filterConfig = null;
        }
        public void doFilter(ServletRequest request,
                    ServletResponse response, FilterChain chain)
                    throws IOException, ServletException {
                    Date startTime, endTime;
                    double totalTime;
                    startTime = new Date();
                    //将处理任务传递给链中的下一个资源
                    chain.doFilter(request, wrapper);
                    //处理响应，计算开始时间和结束时间的差值
                    endTime = new Date();
                    totalTime = endTime.getTime()-startTime.getTime();
                    totalTime = totalTime/1000; //将毫秒转换为妙
                    StringWriter sw = new StringWriter();
                    PrintWriter writer = new PrintWriter(sw);
                    writer.println();
                    writer.println("===============");
                    writer.println("Total elapsed time is: " + totalTime + " seconds." );
                    writer.println("===============");
                    //记录结果字符串
                    writer.flush();
                    filterConfig.getServletContext().
                        log(sw.getBuffer().toString());
                }
}
```

init()在容器实例化过滤器时被调用，主要设计用于使过滤器为处理做准备。该方法接受一个 FilterConfig 类型的对象作为输入。doFilter()与 servlet 的 service()方法（这个方法又调用 doPost()

或 doGet()）处理请求一样，过滤器拥有单个用于处理请求和响应的方法——doFilter()。该方法接受三个输入参数：一个 ServletRequest、response 和一个 FilterChain 对象。destroy()方法执行任何清理操作，在自动垃圾回收之前进行。

- 初始化

当容器第一次加载该过滤器时，init()方法将被调用。该类在这个方法中包含了一个指向 FilterConfig 对象的引用。

- 过滤

执行过滤器的功能。doFilter()方法被容器调用，同时传入分别指向这个请求/响应链中的 ServletRequest、ServletResponse 和 FilterChain 对象的引用。过滤器处理请求，将处理任务传递给链中的下一个资源（通过调用 FilterChain 对象引用上的 doFilter()方法），之后在处理控制权返回该过滤器时处理响应。

- 析构

在垃圾收集之前调用 destroy()方法，以便能够执行任何必需的清理代码。

**2. 配置 Servlet 过滤器**

过滤器通过 web.xml 文件中的两个 XML 标签声明。标签定义过滤器的名称，并且声明实现类和 init()参数。标签将过滤器与 servlet 或 URL 相关联。

web.xml 文件声明一个过滤器，它展示了如何声明过滤器的包含关系：

```
<filter>
    <filter-name>Page Request Timer</filter-name>
    <filter-class>TimeTrackFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>Page Request Timer</filter-name>
    <servlet-name>MainServlet</servlet-name>
</filter-mapping>
<servlet>
    <servlet-name>Main Servlet</servlet-name>
    <servlet-class>MainServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Main Servlet</servlet-name>
    <url-pattern>/*</url-pattern>
</servlet-mapping>
```

上面的代码声明了一个过滤器（Page Request Timer），并把它映射到一个 servlet（Main Servlet）。然后为该 servlet 定义了一个映射，以便把每个请求（由通配符指定）都发送到该 servlet。这是控制器组件的典型映射声明。

**3. 部署 Servlet 过滤器**

过滤器可以和 Web 应用程序一起部署。只需把过滤器类和其他 Web 组件类包括在一起，把 web.xml 文件（连同过滤器定义和过滤器映射声明）放在应用的目录结构中，其他交由 servlet 容器处理。

#### 5.3.2.2 权限管理的实现

如 4.3.7 节中对权限管理模块的描述，Omate 系统中的用户通过角色来访问系统资源，将资源和动作绑定放在一个 resactmapping.xml 文件中完成。

```
<urlmodules>
  <urlmodule path="" parametername="action">
    <resource id="1" name="用户管理">
      <actions>
        <action name="search" value="X">
                <parameter name="UserInformation_ID" type="VARIABLE"/>
        </action>
        <action name="login" value="X"/
        <action name="saveregister" value="X"/>
      </actions>
    </resource>
  </urlmodule>
…
</urlmodules>
```

每个<urlmodule>中，完成了对某个用户请求的资源动作绑定。比如，用户进入用户管理模块首页是激发一个 system_usermanag_select.do 请求。用户管理是一个资源，对应的动作有增删改查等，资源要和动作绑定在一起才对权限的赋予和验证有意义。

权限验证在过滤器中得到实现。设计了一个 RightFilter 过滤器，主要实现步骤在过滤器的 doFilter()方法中实施，步骤如下：

1. 解析 XML 文件，即使用 resactmapping.xml 文件。过滤器截取请求，通过解析这个文件，然后根据请求中的 URL 获得对应的资源操作对象 ResourceOperation。如果出现异常，就通过 RequestDispatcher 转到 error.jsp 页面。
2. 通过 session 取得用户的 UserInformationForm 对象。包含用户的一些基本信息，如用户 ID、所在单位、所在部门等。
3. 判断是否需要资源验证。如果相应的资源个数为零，表示无需验证，反之，则需要验证。
4. 验证用户权限。根据用户 ID 到数据库中查找相应的资源动作授权，如果该资源绑定权限包含于 ResourceOperation 对象中，则表示该用户有权限，继续执行过滤器的下一链，否则，跳转到 noAuth.jsp 页面。

在 web.xml 文件中，配置过滤器如下。

```
<filter>
    <filter-name>RightController</filter-name>
    <filter-class>com.filter.RightFilter</filter-class>
</filter>
```

### 5.3.3 国际化的解决方案

#### 5.3.3.1 国际化的概念

国际化（internationalization，简称 I18N）[47]，指在软件设计阶段，软件即具有支持多种语言和地区的功能。当需要在应用中添加对一种新的语言和国家的支持时，不用对已有的软件返工，无需修改应用的程序代码，而是将局部元素的依赖性从应用程序中的源代码分离出来。

本地（Locale）指一个具有相同风俗、文化和语言的区域。以前的应用程序开发中，软件开发人员集中于实现具体的业务逻辑，面向的客户群固定，由于软件的可移植性差，大多数软件只支持一种语言。如果一个应用事先没有把国际化作为内嵌的功能，当应用需要支持新的 Locale 时，开发人员必须对嵌入在源代码中的文本、图片和消息进行修改，然后重新编译源代码。每当

应用需要支持新的 Locale 时，就必须重复这些繁琐的步骤，降低了软件开发的效率。如今，随着国际交流的日益频繁，需要一款软件能同时支持多种语言和国家。软件的本地化意味着针对不同语言的客户，开发出不同的软件版本，而国际化意味着同一个软件可以面向使用各种不同语言的客户，如图 5-6 所示。

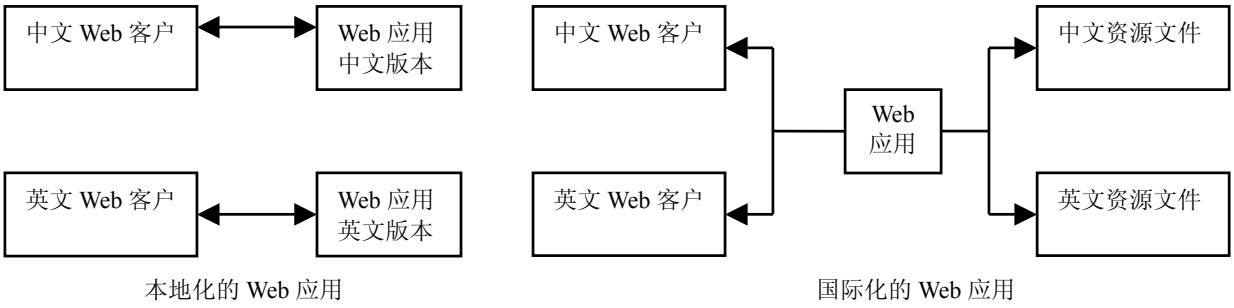



**图 5-6 软件的本地化和国际化的区别**

一个支持国际化的应用，具备以下特征：

- 应用需要支持一种新的语言时，无需修改应用程序的代码。
- 文本、消息和图片从源程序代码中抽取出来，存储在外部。
- 根据用户的语言和地理位置，对相关数据，如日期、时间和货币，进行正确的格式化。
- 支持非标准的字符集。
- 可以方便快捷地对应用做出调整，使它适应新的语言和时区。

由于 Web 服务器不和客户端保持长期的连接，因此每个发送到 HTTP 请求中包含了 Locale 信息。Struts 配置文件的<controller>元素的 locale 属性指定是否把 Locale 对象保存在 session 中，默认值为 true，表示会把 Locale 对象保存在 session 范围中。如图 5-7 所示，当用户的 Locale 为英文时，Struts 框架会向用户返回来自 application_en.properties 文件的文本内容，当用户的 Locale 为中文时，Struts 框架会向用户返回来自 application_ch.properties 文件的文本内容。

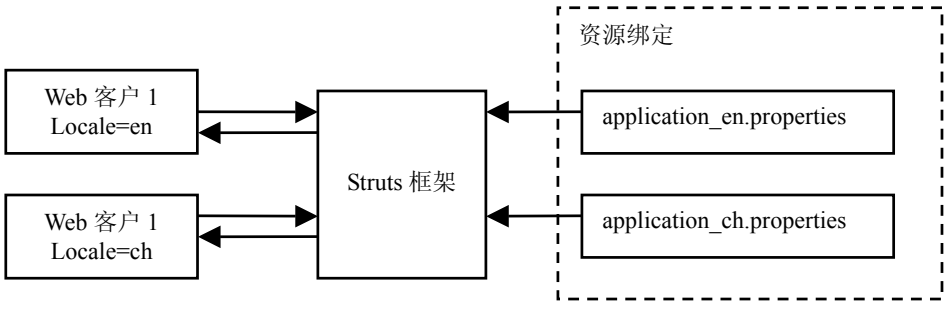



**图 5-7 Struts 框架实现的国际化**

对于多应用模块的 Struts 应用，可以为每个子应用配置一个或多个资源绑定，资源绑定的持久化消息文本存储在*.properties 资源文件中。应用模块中的 Action、ActionForm Bean、JSP 页和客户化标签都可以访问这些资源。struts-config.xml 中的<message-resources>元素定义了一个资源绑定。当应用中包含多个资源绑定时，通过<message-resources>元素的 key 属性来区别。

```
<message-resources parameter="application"/>
<message-resources key="IMAGE_RESOURCES_KEY" parameter="imageresources">
```

Struts 中的每个资源绑定和 MessageResources 类的一个实例对应。MessageResources 对象中存放了资源文件的文本，通过编写代码或者使用与资源绑定的 Struts 组件（如<bean:message>标签）来访问。

#### 5.3.3.2 国际化的实现

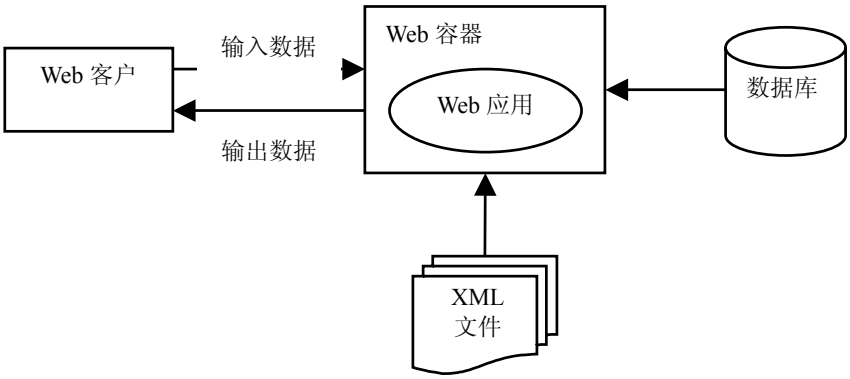



**图 5-8 Web 应用程序的输入/输出流**

从图 5-8 可以看出，Omate 作为一个 Web 应用程序，需要对 HTTP 请求数据、数据库数据、XML 配置文件和响应结果的字符编码进行转换。通过以下几个步骤实现对 Omate 的国际化：

**1. 对 JSP 文件、图片和按钮进行国际化**

设置 JSP 文件的字符编码，将所有 JSP 页面的字符编码统一设为“UTF-8”。

<%@ page contentType=”text/html; charset=UTF-8” language=”java” %>

JSP 文件中不应该直接包含本地化的消息文本，需通过<bean:message>标签从 Resource Bundle 中获得文本。如<bean:message key=”omate.jsp.page.title”>

在 login.jsp 的 HTML 表单中定义了如下按钮：

```
<html:submit property="submit" value="Submit"/>
<html:reset/>
```

为使按钮的 Label 支持多种语言，对以上代码修改如下：

```
<html:submit property="submit">
      <bean:message key="omate.jsp.page.submit"/>
</html:submit>
<html:reset property="reset">
       <bean:message key="omate.jsp.page.reset"/>
</html:reset>
```

**2. 创建临时资源的中文资源文件**

由于在 login.jsp 中增加了一些消息 key，需在默认的 application.properties 文件中添加相应的消息文本。

```
omate.jsp.page.submit=Submit
omate.jsp.page.reset=Reset
```

根据默认的 application.properties 文件，创建一个临时的中文资源文件

application_temp.properties，将上述内容与中文对应。

```
omate.jsp.page.submit=提交
omate.jsp.page.reset=复位
```

**3. 对临时资源文件进行编码转换**

JDK 中的 native2ascii 命令可以实现字符编码转换：

native2ascii -encoding gb2312 application_temp.properties application_ch.properties

该命令将 ISO 编码的消息资源文件转换成 GB2312 编码格式，同时保存到 application_ch.properties。转换后的中文消息资源文件中，显示格式如下：

ISO-8859-1 格式：error.passwd.required=<li>口令错误</li>

GB2312 格式：error.passwd.required=<li>\u53e3\u4ee4\u9519\u8bef</li>

当 Web 客户的 Locale 为中文时，Struts 框架将自动选择来自 application_ch.properties 文件的消息文本。

**4. 创建资源文件**

默认的资源文件是 application.properties。在 Omate 中，创建 application_ch.properties 的中文资源文件和英文资源文件 application_en.properties。

**5. 采用 Servlet 过滤器设置请求数据的字符编码**

调用 HttpServletRequest 的 SetCharacterEncoding("UTF-8")方法，把用户的请求数据的字符编码也设为 UTF-8。图 5-7 中的 Web 应用输入输出采用同一种编码，无需在程序中进行编码转换。Servlet 过滤器可以预处理所有 HTTP 请求，使用过滤器实现可以避免在每个 Web 组件中设置请求数据编码。

```
public class SetCharacterEncodingFilter implements Filter {
      public void destroy() {
      }
      public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
         throws IOException, ServletException {
            request.setCharacterEncoding("UTF-8");
                        chain.doFilter(request, response); //传递到下一个过滤器
      }
      public void init(FilterConfig filterConfig) throws ServletException {
      }
}
```

在 web.xml 中配置 SetCharacterEncodingFilter 过滤器，预处理所有的 URL：

```
<filter>
    <filter-name>Set Character Encoding</filter-name>
    <filter-class>com.omate.util.SetCharacterEncodingFilter</filter-class>
    <init-param>
       <param-name>encoding</param-name>
       <param-value>GBK</param-value>
    </init-param>
</filter>
<filter-mapping>
      <filter-name>Set Character Encoding</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

如果 Omate 需要支持日本客户，则不用修改应用程序的代码或配置，创建一个

application_jp.properties 资源文件即可。

## 5.4 本章小结

本章重点介绍了 Struts 框架在系统实现中的运用。以用户管理模块为例，详细描述了在 Web 应用程序的开发过程中如何应用 Struts 框架构建视图、模型和控制器以及配置文件，总结了系统开发过程中遇到的表单重复提交、国际化和权限管理等关键问题及解决方案。

# 第六章 总结与展望

## 6.1 总结

论文以Office Mate企业办公系统的研究与开发为背景，详细介绍了Struts框架的体系结构、核心组件和工作原理，并基于Struts框架实现了第二代Office Mate企业办公系统的部分模块。课题研究的主要工作集中在以下几个方面：

- 在阅读和参考大量国内外有关理论和应用的资料基础上，分析了企业办公系统的现状，并且结合现代企业的特点和实际需求，详细分析、设计并部分实现了Office Mate企业办公系统。
- 对影响Web层框架选择的因素进行总结，并采用这些因素对几种常见框架进行比较分析，总结了Struts框架的优缺点。
- Office Mate企业办公系统基于Struts框架技术开发，实现了业务逻辑和业务数据的分离，提高了系统的稳定性和可扩展性，应用程序具有层次结构清晰，易于分工协作，代码重用率高，维护扩展性好等优点。Struts框架是目前先进的Web层应用框架的代表，采用Struts框架使得Office Mate企业办公系统的实现具有技术先进性。
- 利用Struts框架，解决了系统开发中的表单重复提交、国际化和权限管理等关键问题。

## 6.2 展望

目前系统框架已经搭建完毕，一期目标已经实现。由于时间和水平有限，基于 Struts 框架的 Office Mate 企业办公系统的研究还有许多不足和尚未完成的工作，可以在以下几个方面进一步细化和完善。

- 在框架研究方面，有专家分析，几种框架集成的开发模式是将来应用框架的发展趋势[24]。为了进一步研究框架以及框架间的集成，计划将 Office Mate 系统构建于三个框架上，即表示层采用 Struts 框架，业务层采用 Spring 框架，持久层采用 Hibernate 框架。
- 在系统架构方面，提取办公业务流程以形成一个工作流平台将是系统二期建设中的主要研究课题。同时，将对 Office Mate 企业办公系统进行体系结构的发现、演化和评价。
- 办公自动化系统的理论和相关的技术也在不断发展，如何从第二代办公系统向第三代知识管理型的办公自动化系统过渡也将是今后研究的重点。

# 参考文献

[1] 陈江东．办公自动化系统的系统分析．计算机系统应用，1998.10

[2] 杨杰，刘丹．基于工作流和 B/S 结构的 OA 系统设计．武汉理工大学学报，2005.2

[3] 张金雄．计算机技术与企业扁平化管理．轻金属，2000.2

[4] 吴应良．基于 Web 的公文管理信息系统的设计与实现．计算机工程与应用，2002.12

[5] 高充．基于 Lotus Domino/Notes 的宝钢办公自动化系统．软件世界，2000.5

[6] Lotus 第三代基于知识管理的办公自动化系统．软件世界，2000.6

[7] Arn, Joseph. *Office Automation: an information system approach* [M]. Boston Boyd & Fraser，1998

[8] 姜旭平．信息系统开发方法[M]．北京：清华大学出版社，1999

[9] C.Alexander. *The Timeless Way of Building* [M]. Oxford University Press，1979

[10] 武伟 陆建德．层模式在 J2EE 中的应用开发研究．微机发展，2005.1:125-127

[11] [德]Frank Buschmann 等著．贲可荣等译．面向模式的软件体系结构卷 1：模式系统[M]．北京：机械工业出版社，2003

[12] 万建成，卢雷．软件体系结构的原理、组成与应用[M]．北京：科学出版社，2002

[13] 张友生．软件体系结构[M]．清华大学出版社，2004

[14] 郑人杰等．实用软件工程（第二版）[M]．北京：清华大学出版社，2001

[15] 21CMM．框架不是框框——应用框架的基本思想．
http://www.digitalspirit.info/phf/artichecture/applyFrame.htm

[16] Don Roberts, Ralph Johnson. *Evolving Frameworks: A Pattern Language for Development Object-Oriented Frameworks*. http://st-www.cs.uiuc.edu/users/droberts

[17] Humphrey Sheil. *Frameworks save the day*.
http://www.javaworld.com/javaworld/jw-09-2000/jw-0929-ejbframe.html

[18] 林星．软件质量之路．IBM DeveloperWorks，2004、3

[19] SUN 公司．SUN 技术蓝皮书．http://java.sun.com/blueprints/code/index.html

[20] Eric Armstrong et al. *The J2EE™ 1.4 Tutorial*. SUN 公司，2005

[21] 蔡剑，景楠．Java 网络程序设计：J2EE 1.4 [M]．北京：清华大学出版社，2003

[22] Neal Ford. *Art of Java Web Development* [M]. Manning Publications，2004

[23] 龚永生．当前流行的 J2EE Web 应用框架分析．IBM DeveloperWorks，2002

[24] Mark Eagle. *Wiring Your Web Application with Open Source Java* .
http://www.onjava.com/pub/a/onjava/2004/04/07/wiringwebapps.html?page=3，2004、4

[25] 曲俊生．JAVA 开放源码项目与工具在企业应用开发中的运用．IBM DeveloperWorks，2003.5

[26] *How Many Java Web Frameworks Can You Name?*.
http://www.manageability.org/blog/stuff/how-many-java-web-frameworks，2004、3

[27] Bruce E.Wampler. Java 与 UML 面向对象程序设计[M]．北京：人民邮电出版社，2002

[28] 阎洪. Java 与模式[M]. 北京：电子工业出版社，2003

[29] Tom Pender 著. 耿国桐等译. UML 宝典[M]. 北京：电子工业出版社，2004

[30] Timothy C.Lethbridge 等著. 张红光等译. 面向对象软件工程[M]. 北京：机械工业出版社，2003

[31] Bruce Eckel. *Thinking in Java, 3rd Edition* [M]. 北京：机械工业出版社，2004

[32] Bruce Eckel. *Thinking in Enterprise Java* [M]. www.bruceeckel.com

[33] Bruce Eckel. *Thinking in Patterns* [M]. www.bruceeckel.com

[34] Ian F. Darwin. Java 经典实例[M]. 北京：中国电力出版社，2002

[35] 王克宏主编. 第六届全国 Java 技术与应用大会文集. 2003

[36] Abraham Silberschatz 等. 数据库系统概念（原书第 4 版）[M]. 北京：机械工业出版社，2003

[37] [美]Aadam Drozdek. 数据结构与算法（Java 语言版）[M]. 北京：机械工业出版社，2003

[38] 邵维忠，杨芙清. 面向对象的系统设计[M]. 北京：清华大学出版社，2003

[39] 邵维忠，杨芙清. 面向对象的系统分析[M]. 北京：清华大学出版社，1998

[40] [美]Paul J. Perrone 著. 刘文红等译. J2EE 开发使用手册[M] 北京：电子工业出版社，2004

[41] [美] Justin Couch 著. J2EE 宝典[M]. 北京：电子工业出版社，2002

[42] [美]Erich Gammay 等著. 设计模式：可复用面向对象软件的基础[M]. 北京：机械工业出版社，2000

[43] Wendy Boggs，Michael Boggs 著. 邱仲潘等译. UML 与 Rational Rose 2002 从入门到精通[M]. 北京：电子工业出版社，2002

[44] 王智学. ROSE 对象建模方法与技术[M]. 北京：机械工业出版社，2003

[45] WebWork 文档 www.opensymphony.com/webwork，2004、12

[46] Spring 中文开发手册 http://spring.jactiongroup.net/，2005

[47] http://struts.apache.org Struts 官方网站

[48] Chuck Cavaness. *Programming Jakarta Struts, 2nd Edition* [M]. O'Reilly Press，2004

[49] Sue Spielman. *The Struts Framework Practical Guide for Java Programmers* [M]. Morgan Kaufmann Press，2003

[50] Bill Siggelkow. *Jakarta Struts Cookbook* [M]. O'Reilly Press，2005

[51] James Turner，Kevin Bedell. *Struts Kick Start* [M]. Sams Publishing，2002

[52] Ted Husted et al. *Struts in Action: Building web applications with the leading Java framework*[M]. Manning Publications，2003

[53] 飞思科技产品研发中心编著. JSP 应用开发详解（第二版）[M]. 北京：电子工业出版社，2005

[54] [美]Marty Hall 著. 邓英材等译. Servlet 与 JSP 核心技术[M]. 北京：人民邮电出版社，2001

[55] 孙卫琴，李洪成. Tomcat 与 Java Web 开发技术详解[M]. 北京：电子工业出版社，2004

[56] 孙卫琴. 精通 Struts：基于 MVC 的 Java Web 设计与开发[M]. 北京：电子工业出版社，

2004

[57] Malcolm Davis. Struts，MVC 的一种开放源码实现. IBM DeveloperWorks，2001.2

[58] 赵晨希. Struts 建立 MVC 应用的介绍. IBM DeveloperWorks，2002.12

[59] James Goodwill. *Mastering Jakarta Struts* [M]. Wiley Publishing，2002

[60] Deepak Alur et al. *Core J2EE Patterns: Best Practices and Design Strategies, 2nd Edition*[M]. Pearson Education，2003

[61] Rod Johnson. *Expert One-on-One™ J2EE™ Development without EJB™* [M]. Wiley Publishing，2004

[62] CT Arrington. *Enterprise Java with UML* [M]. Johns Wiley & Sons，2002

[63] 晏子译. MySQL 中文参考手册. http://manual.phpv.net/mysql/

[64] 陈隽伟. 选择数据访问模式，合理规划数据访问层. IBM DeveloperWorks，2005.2

[65] 孙卫琴. 精通 Hibernate：Java 对象持久化技术详解[M]. 北京：电子工业出版社，2005

# 致谢

研究生生活即将结束，这三年中，承蒙各位师长和亲友们的无私帮助，使我得以成长和进步。在此，向所有给予我帮助和关怀的老师、同学和朋友表示衷心的感谢。

首先，谨向我的导师孙龙清副教授致以诚挚的谢意。孙老师在论文的选题、研究以及撰写过程中，都给予了精心的指导。在攻读硕士学位期间，孙老师在学术研究上对我要求严格，在生活上对我关怀备至，并且为我竭力创造了宽松舒适的科研氛围。孙老师严谨的治学态度、在科研事业上勤奋、精益求精和以身作则、为人师表的风范对我产生了非常深刻的影响，是我学习的楷模。

在课题研究和论文撰写的过程中，我还得到了计算机应用研究所、计算机网络中心和计算机系诸位老师和同学们的大力帮助，在此表示感谢。

向在百忙之中抽出宝贵时间对本文进行评阅的专家和学者表示诚挚的谢意。

感谢就读七年的母校——中国农业大学。

感谢北京邮电大学 2003 级研究生冯静在 Struts 框架以及系统设计上给予的宝贵建议。

感谢现任职于美国 Applied Technology & Management（ATM）公司的刘海青博士给予的精神动力。

感谢室友们的呵护，她们是葛晓棠、庆兆珅、杨晓辉、朱静。

感谢朋友们一如既往的支持，他（她）们是蔡铭洁、冯莉花、耿颖、焦培、康丽、王大鹏、向前、徐建华。

最后，感谢我的父母，这个世界因为您们而美丽。

**王欢**

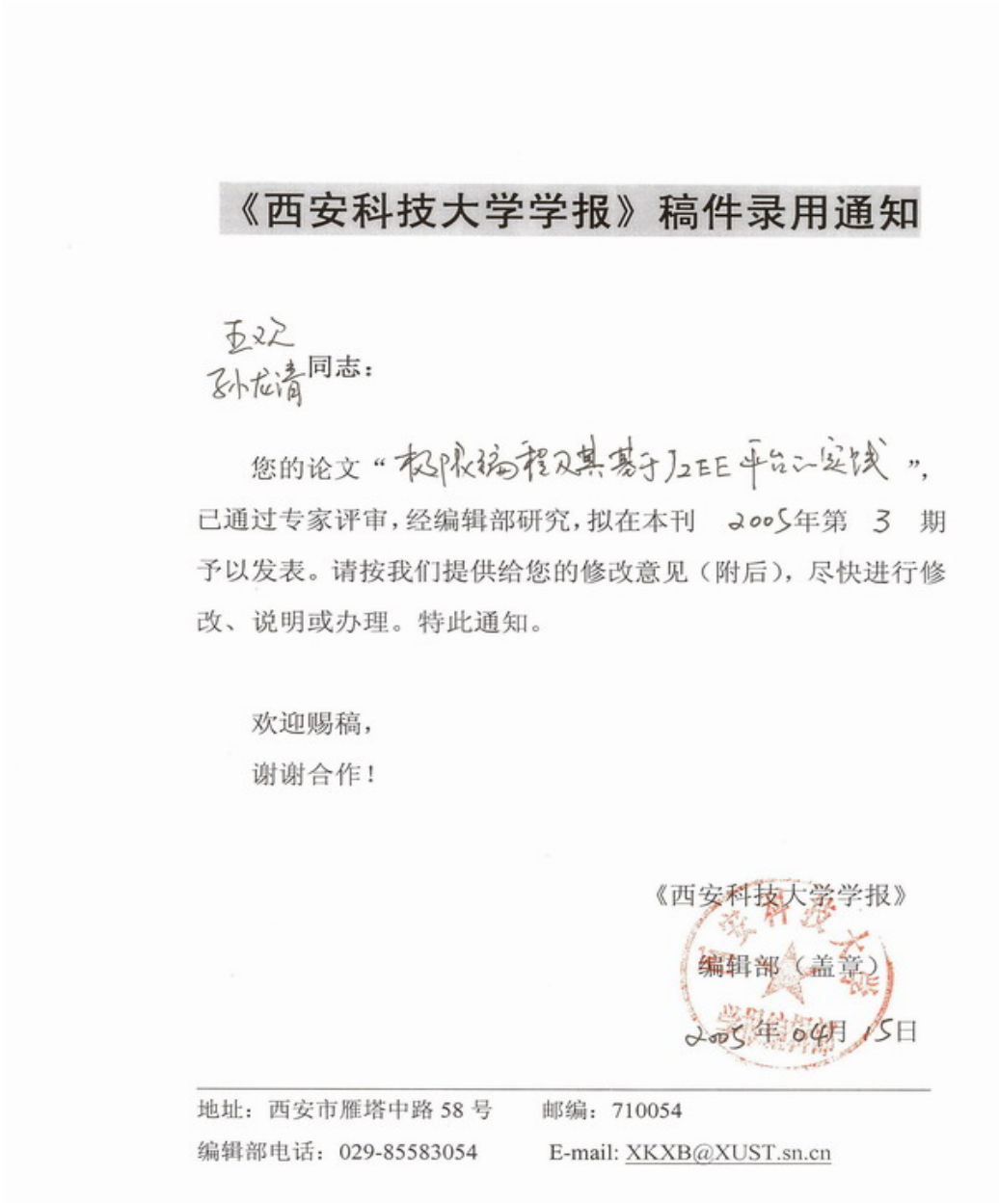
2005-5-20

# 附录 I：索引

| | | | |
|---|---|---|---|
| 1. | Office Automation | OA | 办公自动化 |
| 2. | Model-View-Controller | MVC | 模型－视图－控制器 |
| 3. | Presentation-Abstraction-Control | PAC | 表示－抽象－控制 |
| 4. | Java 2 Platform Enterprise Edition | J2EE | Java 2 平台企业版 |
| 5. | Enterprise Information System | EIS | 企业信息系统 |
| 6. | Enterprise Resources Planning | ERP | 企业资源计划 |
| 7. | Application Programming Interface | API | 应用编程接口 |
| 8. | Java Database Connection | JDBC | Java 数据库连接技术 |
| 9. | Java Naming and Directory Interface | JNDI | Java 命名和目录接口 |
| 10. | Enterprise JavaBean | EJB | 企业级 JavaBean |
| 11. | Remote Method Invoke | RMI | 远程方法调用 |
| 12. | Java Interface Definition Language | Java IDL | Java 接口定义语言 |
| 13. | Java Server Pages | JSP | Java 服务器页面 |
| 14. | Extensible Markup Language | XML | 可扩展标记语言 |
| 15. | Java Message Service | JMS | Java 消息服务 |
| 16. | Java Transaction Architecture | JTA | Java 事务框架 |
| 17. | Java Transaction Service | JTS | Java 事务服务 |
| 18. | JavaBeans Activation Framework | JAF | JavaBeans 激活框架 |
| 19. | Service Provider Interface | SPI | 服务提供者接口 |
| 20. | Integrated Development Environment | IDE | 集成开发环境 |
| 21. | Plain Ordinary Java Object | POJO | 简单原始的 Java 对象 |
| 22. | To Be Done | TBD | 未完成 |
| 23. | Pull Hierarchical Model-View-Controller | Pull HMVC | 分级拉出式的 MVC 设计模式 |
| 24. | Inversion of Control | IoC | 控制倒置 |
| 25. | Just-in-time | JIT | 实时 |
| 26. | Data Transfer Object | DTO | 数据传输对象 |
| 27. | Java Server Web Development Kit | JSWDK | Java 服务器 Web 开发包 |
| 28. | Unified Modeling Language | UML | 统一建模语言 |
| 29. | Computer Aided Software Engineering | CASE | 计算机辅助软件工程 |
| 30. | TagLib Definition | TLD | 标签库描述符 |
| 31. | Java Data Object | JDO | Java 数据对象 |
| 32. | Object/Relation Mapping | ORM | 对象/关系映射 |
| 33. | internationalization | I18N | 国际化 |

# 附录 II：攻读硕士学位期间发表的论文

## 文章录用通知



《西安科技大学学报》稿件录用通知

王欢
孙龙清同志：

您的论文“权限框架及其基于J2EE平台实现”，已通过专家评审，经编辑部研究，拟在本刊 2005年第 3 期予以发表。请按我们提供给您的修改意见（附后），尽快进行修改、说明或办理。特此通知。

欢迎赐稿，

谢谢合作！

《西安科技大学学报》

编辑部（盖章）

2005年04月15日

地址：西安市雁塔中路 58 号 邮编：710054

编辑部电话：029-85583054 E-mail: XKXB@XUST.sn.cn

# 个人简历

姓名：王欢

性别：女

出生日期：1979 年 8 月

籍贯：天津

最后学历（学位）：硕士

毕业院校：中国农业大学

- 在学期间参加的研究项目：

2004、3——2004、9　敏捷建模和极限编程方法学研究

2003、10——2005、6　基于 Struts 框架的企业办公系统的设计与实现

- 在学期间发表论文：

王欢，孙龙清. 极限编程及其基于 J2EE 平台的实践. 西安科技大学学报, 2005. 3