摘　要

随着信息技术和Intemet技术的发展以及市场竞争的加剧，电信管理行业中计算机应用也得到了飞速的发展，建立一个反应迅速、智能灵活、安全可靠的电信管理信息系统对当前的电信管理行业具有非常重要的意义。为此本文设计出一种基于MVC模式的Web应用体系结构为企业高效地构建管理信息系统。

本文阐述了MVC设计模式的体系结构和工作原理，并将其引入到基于J2EE的企业应用开发中，构建出适合中小型企业管理信息系统架构，本论文的研究是以淄博市电信卡管理信息系统为设计背景而展开的。

论文利用现有的Web技术，整合mvc1、mvc2 和struts三个框架技术来设计实现企业级的Web应用，详细分析了淄博市电信卡管理信息系统的业务流程及系统需求，根据系统的设计原则和功能目标进行总体架构，给出了系统的总体设计和功能模块的划分，研究并实现了一个电信卡管理信息系统，能够完成系统管理功能的同时还能为管理者和决策者提供管理决策功能。该系统主要由九大模块组成：制作管理模块，入库管理模块，财务管理模块等，每一模块根据角色的不同，享有不同的权限。

根据系统的设计原则，在具体实现电信卡管理信息系统时，结合三个开源框架技术设计了一个基于J2EE的MVC模式。该模式是分层的、低耦合的框架结构，实现了系统的表示层、业务逻辑层的分离，为软件的可维护性、健壮性提供了保障。

实践表明，采用该MVC模式的设计方案使得整个系统的结构清晰，容易理解，并提高了系统的开发效率和可维护性。

关键词：MVC，J2EE，EJB，Struts，设计模式，体系结构

ABSTRACT

With the development of the Information Technology and the computer technology ,the market's competition intensfies.the application of computer in the management of telecom industry also got the development at full speed. It has a very important significance to establish a rapid response, intelligence, flexibility, security information system. So a web application architecture based on MVC pattern is designed for enterprise to construct information system.

The thesis is based on research of information management of temcom cards.Integrated the Web framework of mvc1、mvc2 and struts to design and realize enterprise application .It's main tasks include:

First, the thesis introduced the inadequacy of the traditional information system put-forward to structure information system of a muti-tier B/S model after research deeply of J2EE, MVC pattern and the opening source framework of mvc1、mvc2 and struts.

Second, analyse deeply the business process and system requirement, acording to design principles and function goals, the general framework of the Managemnet information System of telecom card be given, and the system and functional model be designed.

Third, according to the system's design, realized the Managemnet information System of telecom card based on MVC pattern with the integrated framework technology which made up of mvc1, mvc2 and struts.

Focus of the thesis is to construct a general framework for realizing he Managemnet information System of telecom card by integrating mvc1, mvc2, struts framework technologies. It has been proved that the general framework has characters of cleaner structure, looser coupling, legible, simplicity of coding, furthers the developing efficiency of the Web application system.

Key Words: MVC, J2EE, EJB, Struts, Design Pattern, Architecture

原创性声明和关于论文使用授权的说明

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名：李艳 日 期：09.4.5

关于学位论文使用授权的声明

本人完全了解山东大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权山东大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

(保密论文在解密后应遵守此规定)

论文作者签名：李艳 导师签名：[illegible] 日 期：09.4.5

引　言

随着 WebLogic，Webspheer 等 ApplicationServer．对 J2EE(Java2 Enterprise Edition)的全面支掉，J2EE 正逐渐成为 Web 应用软件开发的标准.

虽然 J2EE 的确是一个非常优秀的企业应用开发平台，它能使开发人员开发出非常稳定、健壮的系统。但事实上，在 J2EE 所提供的能力、服务与开发人员所要建立的企业应用之间仍然存在很大的差距。人们通常认为，掌握了一门新的技术本身就可以开发出非常优秀的系统，但事实和经验却不是如此。“除了掌握新的技术外，还需要其他方面的东西才能建立一个成功的系统。模式可以帮助实现知识积累和传递的过程，它帮助我们记录和交流已被证实的解决方案，这些解决方案可以解决在不同环境里重现的问题。有效的运用模式，可以使我们远离重复投资的怪圈。”开发人员在学习技术的同时必须充分地认识到学习设计，学习模式的应用同样具有非常重要的意义，构建一个成功的企业应用，良好的模式运用是系统成功的所在。在运用 J2EE 平台技术进行应用开发时，J2EE 模式正如上述所言具有非凡的意义。J2EE 模式正是 J2EE 关键技术的最佳实践、设计策略和经过验证的解决方案。在 J2EE 平台技术的应用中，J2EE 模式覆盖了分布于 J2EE 所有层的不同需求，将 J2EE 层次模型化，为开发人员提供了基于各层的最佳实践经验的总结。

为了使用 Java 和 HTML 开发出更具柔软性的应用软件，必须解决好以下两个间距：(1)用户界面和事物逻辑(Business Logic)的分离；(2)动态生成网员的逻辑部分和 HTML 的分离。

为了解决问题(1)，一股采用面向对象设计中常用的 MVC 模型。MVC 模型中的 3 要素 Model，View 和 Controller 分别由 javaBean，JSP 和 Servlet 担当。

为了解决问题 (2)，一般采用 JSp 提供的标记库功能，它是通过在 Html 中插入特殊标记，来达到逻辑部分和 HTML 的分离的目的.

本文所研究的MVC (Model/View/Controller)模式是软件工程学中一个非常重要的设计模式，为交互式系统提供了一个非常优秀的开发模型，MVC模式对类

对象的强制分离使系统开发变得非常灵活，同时又提高了系统设计的可重用性。正是MVC模式在交互式系统开发中所体现的强大优势，MVC模式被开发人员逐步引入J2EE体系之中，成为J2EE进行交互式应用开发时，特别是web应用开发时一个非常重要的设计模式。

论文的结构大体如下：

第一章绪论阐述了课题的背景、来源，给出了课题的主要内容，并指出论文中将要使用到的一些技术。第二章综述了J2EE体系结构规范，分析其Web应用的优势；阐明了MVC的概念，它的工作原理，以及对MVC设计模式的理解。第三章详细分析了该企业的业务流程及系统需求，根据系统的设计原则和功能目标进行总体架构，给出了系统的总体设计和功能模块的划分。第四章针对淄博市通讯公司的电信卡管理信息系统，采取基于J2EE的MVC模式的总体设计，并对其进行详细设计、编码实现。第五章对论文进行总结。

第一章 绪论

1.1 课题背景

随着网络和通信技术的飞速发展，信息化已经成为当今社会不可阻挡的发展趋势，人类社会正加速向信息社会迈进，信息与物质、能源成为了现代化社会的三大基础，信息化程度的高低是衡量一个国家、地区现代化水平的重要标志，信息处理和利用效率的高低反映了一个国家的经济发展水平和科学技术水平。信息管理和信息科学已经渗入到了社会的各行各业，成为人们生活中不可或缺的一部分。作为信息系统的一部分电信卡管理信息系统得到了广泛地应用，对电信部门有着重要的意义。[1]

目前在淄博的市场上卡的应用系统有许多，如IP电话卡、201卡、300卡、会员卡、校园“一卡通”卡等系统，部分系统的功能是由人工来实现的，还有部分系统功能主要围绕着局部的应用进行开发的，这些系统虽然可以满足基本使用要求，但它们对决策者、管理者方面的支持有很大欠缺。它一般只是采取定时上传数据，没有办法真正实现实时交换数据，不能及时反映信息，因此存在信息延迟等问题。

如何快速且高质量地开发出满足不同需求的软件？传统的软件开发方法在这时已显得无能为力了，如何进行企业的Web应用开发，如何方便、快捷地构造出企业Web应用系统已成为一个目前急需解决的课题。

Web应用也已经从过去的发布相对静态的内容发展到如电子商务、信息管理系统等动态交互信息的处理。Web应用先后出现了CGI, PHP, JSP技术等，这些技术的产生缓解了Web编程的难度。但是它们有一个共同点，就是未能将业务逻辑和界面显示分离开来，也就是说，Web编程往往由一个或少量的开发人员来完成且开发难度大，在应用开发上依然存在着较大的困难。

正是基于以上的分析，本课题进行了Web应用开发方面的研究和探讨，将面向对象的MVC设计模式与J2EE多层体系结构结合起来形成一种快速高效的开发模式，来组建企业信息应用系统、电子商务系统等Web应用，并在淄博市通讯公司的电信卡综合管理系统中应用和实现。

1.2 研究现状

MVC是一种目前广泛流行的软件模式，国内外对MVC框架的研究与应用早已有之。早在20世纪70年代，IBM就推出了Sanfrons。isieo项目计划，其实就是MVC模式的研究。SUN公司针对MVC模式先后制定了两种规范，称为JSP Modell和JSP Model2。

MVC设计模式应用于Web应用程序时，JSP对应于视图，Servlet对应于控制，JavaBean对应于模块。当web客户端的HTML或JsP网页向服务器提交时，服务器端的控制器Servlet统一处理这些提交请求。这个控制器Servlet根据提交的业务不同，将请求传递给相应的业务Bean操作处理，然后将业务Bean的处理结果再传递给视图JSP。视图JSP在服务器上处理之后以HTML的方式回显给客户端。

在Web开发领域，也有很多基于MVC的框架，目前最主流的请求驱动MVC框架是Struts WebWork2和Spring MVC a Struts将Java Servlet和JSP技术结合在一起，从而实现了一个Web的MVC框架。现在Struts框架在Web中得到了广泛的应用，通过Web可以构造大型、易变的Web月及务[4]。

Struts本身就是一个可重用的MVC框架，同时Struts本身是一个开放源代码的MVC框架，因此开发人员可以有针对性的在其基础上添加与本身相关的内容，这是Struts的优点。Struts也有其不足之处，首先是学习困难，虽然对于复杂的、大型的Web, Struts很有用处，但Struts很复杂，不利于学习：其二Struts它创建一个ActionForm JavaBean，在系统中就会产生大量的JavaBean类，同时加大了处理的难度。

Spring MVC则是其中最灵活的一个，它给人的感觉是Spring MVC就像一个高度可扩展的插件体系，可以根据需要随意的替换其中的组件[}s}。但是灵活的代价就是增加了复杂性，众多功能类似、但是实现机制不同的组件也增加了不一致性。现在Struts和Webwork2的开发团队已经合并到Struts Action Framework下，新发布的Webwork2.2又提供了对现在炙手可热的AJAX技术支持，相比而言Spring MVC的发展似乎有些慢了。

另外，基于MVC的Web框架还有Hibemate等，这些框架各具有其特点和优点。

针对以上所描述的不足之处，作者提出了一个基于J2EE平台下对MVC模式的

扩展。该模型的最终目标是最大限度解除模型、视图、控制器、数据库四者之间的耦合，从而提高Web应用程序的可复用性、易扩展性、结构清晰性。

1.3 本文工作

在本课题中，主要有以下几个方面的内容：

（1） 对现有信息系统开发技术JEE, MVC设计模式、mvc1框架、mvc2框架和Struts框架分析和研究, 提出一个基于J2EE平台下对MVC模式的扩展。

（2） 对淄博市通讯公司的卡管理过程做需求分析，把握住客户需求，能够实现电信卡的综合管理，即从生产、销售、库存、调拨出库以及统计报表生成的整个流程，并能够为决策提供一定的支持。体系结构上要求B/S结构，取代传统的C/S结构。该系统主要的功能模块有：制作管理模块，入库管理模块，财务管理模块，综合查询模块，调拨管理模块，销售管理模块等。

（3） 根据需求，采用J2EE多层体系构架，引入MVC设计模式，进行系统的总体设计。主要包括，模型(Model)设计，以JavaBean实现；视图(View)设计，以Jsp/Html实现；控制器设计(Controller)，以Servlet实现。

（4） 按照面向对象的关系数据库设计原则，一个对象基本对应一个表中的一条记录或一个记录集，再按照遵循数据库设计第三范式的原则，进行数据库设计。

（5） 系统采用基于J2EE的MVC模式架构，以制卡管理模块为例，实现详细设计和编码实现。

（6） 测试运行。

1.4 小结

这一章主要阐述了课题的来源和背景，并给出了基于J2EE体系结构的MVC模式研究的现状，引出在课题中将要研究和实现的内容。

第二章 J2EE 与 MVC 模式

本章给出了模式的概念并分析综述了 J2EE 规范，介绍了 MVC 设计模式，并引出了基于 J2EE 的经典 MVC 设计模式，为下一章引出论文的主题提供背景。

2.1 J2EE

J2EE 是一个涉及多个层面的复杂的概念，是 SUN 基于 java 的体系结构。J2EE 规范的主要技术包括:EJB(服务器端分布式组件技术)、Servlet/JSP (主要用于 Web 服务器端来完成请求/响应等 Web 功能及简单商业逻辑的技术)、JDNI(名称与目录服务 API)、JDBC(对关系型数据库进行操作的连接桥)、RMI/RMI IIOP(进程间相互通讯的重要机制)、JMS(提供异步消息处理机制)、JTA/TTS(组件的事物处理支持)、JavaIDL(应用 Java 语言实现 COBOR 标准的模型)、JavaMail/JAF(提供与平台无关的电子邮件服务功能)、JCA(用于与其它系统进行集成)以及 X M L (一些 J2EE 技术的所依靠的技术)。[2]

J2EE 架构是一个多层的结构，包括以下层:

（1）用户层:用来与用户交互，并把来自系统的信息显示给用户。J2EE 平台支持不同类型的用户。包括 HTML 用户.Java Applets 和 Java 应用等;

（2）Web 层:Web 层产生表示逻辑，并接受来自客户端的用户反馈。在所接收的客户端请求的基础上，表示层对用户的请求产生相应的回应。在 J2EE 平台中，是由 Web 容器内的 Servlet 和 JSP 来实现这一层;

（3）业务层:业务层处理应用的核心业务逻辑。业务层为低层业务提供必要的接口。业务组件通常被实现为 EJB 容器内的 EJB 组件。其中，EJB 容器提供组件生命周期，管理持久性、事务和资源分配等;

（4）EIS 层:该层为企业的信息服务系统服务，包括数据库系统，事物处理系统，企业资源计划系统等。EIS 是 J2EE 应用与非 J2EE 应用的连接点。

层次结构如图 2.1 所示：

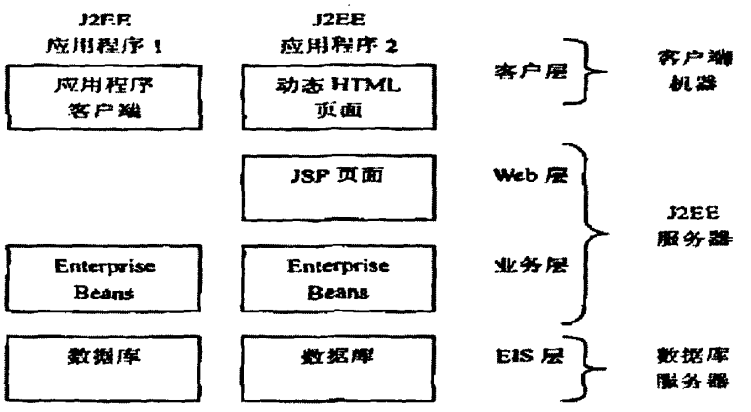

图 2.1 J2EE 层次体系结构

2.2 MVC 模式

2.2.1 MVC 模式概念

MVC（Model-View-Controller）即模型-视图-控制设计模式，是一种面向对象的设计模式，旨在实现表示和逻辑的分离，使软件体系结构层次清晰，便于开发升级和维护，也为软件的健壮性提供了保障。

2.2.2 MVC 工作模型原理

MVC 模式将所面对的系统分为三个部分,分别是数据模型、视图和控制器,并定义了这三个部分之间实现通信的一种模式,使每个部分不必卷入到其他部分的状态表示和方法实现的细节中去.每个部分有自己的数据管理规则,各个部分对象之间的通信只能使用已定义的一个受限连接集合进行.保持这种分离性通常是好的面向对象编程和设计要追求的目标,让各个对象只需专注于自己的事务.[3]

Model 也叫模型,本质上封装了数据及行为,其中包含对数据控制及修改的规则,提供了一套查询、改变 Model 状态的方法。

View 也叫视图,是 Model 所表示出来的图形界面,它主要用于提交 Model 的信息展示给用户。

Controller 也叫控制器,是 Model 和 View 之间的协调者,它的主要作用有以下几点：(1)定义用户界面对用户输入的响应方式。

(2)解释用户的输入,并命令 Model 进行相应的操作创造相应模型。

(3)负责将模型信息传递给 View,必要时还要负责创建新的 View 和 Controller。

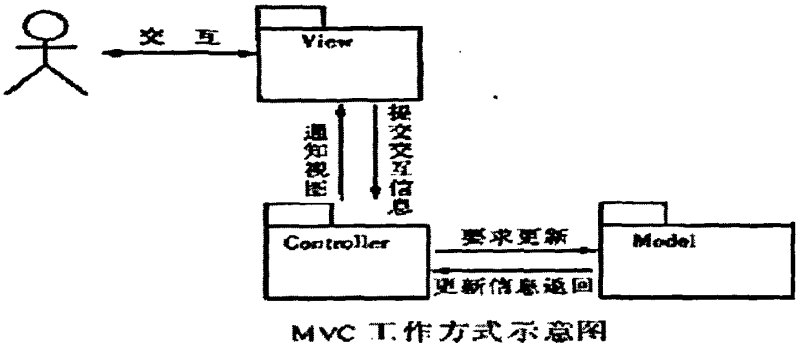

图 2.2 MVC 工作方式

这三者的工作方式可用图 2.2 来表示。MVC 模式将所面对的系统分为三个部分,分别是数据模型、视图和控制器,并定义了这三个部分之间实现通信的一种模式,使每个部分不必卷入到其他部分的状态表示和方法实现的细节中去.每个部分有自己的数据管理规则,各个部分对象之间的通信只能使用已定义的一个受限连接集合进行.保持这种分离性通常是好的面向对象编程和设计要追求的目标,让各个对象只需专注于自己的事务.

2.3 基于 J2EE 的 MVC 设计模式

2.3.1 MVC 模型 1 简介

MVC 模型 1 是面向 Web 应用软件开发的 MVC 模型，它有 2 个主要组成元素：JSP，JavaBean。其中 JSP 对应于 MVC 模型的 View,也对应 MVC 模 Controller；JavaBean 对应 MVC 模型中的 Model。其工作模型如图 2.3 所示：

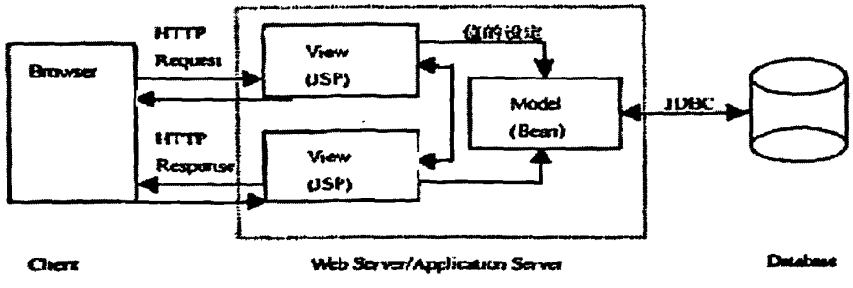

图 2.3 模型 1 体系结构

其工作流程为：

（Ⅰ） (1)Jsp 接受客户端请求；

(2)Jsp 把接收到的 HTMLFORM 数据保存到数据 Bean 中；

(3)逻辑 Bean 进行数据处理；

(4) 并返回逻辑 Bean 的处理结果，返回给客户端.

（Ⅱ）Jsp 作为控制器导航到其他 Jsp 页面。重复过程（Ⅰ）。

由此我们可以看到，模式 1 中，没有抽象分离出专门的 Controller 控制器,Jsp 即是 View 又充当控制器；但它实验了逻辑与表现的分离，即抽象出了 Model，使 Jsp 设计脱离逻辑，只注重表现和导航，因而它适合与小型的应用系统，快速灵活。而对大型复杂化的系统来说，因为它没有专门的控制器，结构层次不够清晰，对于维护、管理和升级都很困难。下面的模式 2 就是对模式 1 的缺陷的修缮和改进。

2.3.2 MVC 模型 2 简介

MVC 模型 2 是模型 1 的改进，也是面向 Web 应用软件开发的 MVC 模型，它有 3 个主要组成元素：JSP，Servlet 和 JavaBean。其中 JSP 对应于 MVC 模型的 View，Servlet 对应于 Controller，Bean 对应于 Model。Model 的 Bean 又分为逻辑 Bean 和数据 Bean：逻辑 Bean 用于事务处理，数据 Bean 用于保存 HTML FORM 数据。

MVC 模型 2 的体系构如图 2.4 所示。

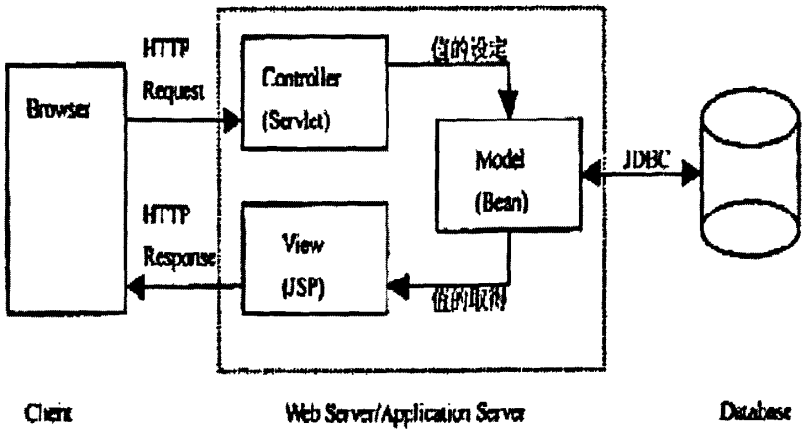

图 2.4 模型 2 体系结构

其工作流程为：

(1)Servlet 接受客户端请求；

(2)Servlet 把接收到的 HTMLFORM 数据保存到数据 Bean 中；

(3)逻辑 Bean 进行数据处理；

(4) Servlet 报据逻辑 Bean 的处理结果，调用响应的 Jsp 生成 HTML 页面，并返回给客户端.

MVC 模型 2 和 MVC 的主要区别是：在 MVC 模型中，View 和 Model 间是登录和通知的关系，当 Model 对象的数据发生变化时，通知已登录的 View 对象，显示新的数据，这是称为 Observer 的设计模型。而在 MVC 模型 2 中，View 和 Model 间没有采用 Observer 模型，这是由 Web 应用软件的特点所决定的。因为在 HTTP 协议是无连接的，客户端发出清求，收到服务器的应答数据后，客户端和服务之间的链接就断开了。

2.3.3 Struts 设计模式

Jakarta 是 Apache Software 的一个研究开发 java 产品的工程，主要为 Java 开发者提供各种开发工具及软件框架，Struts 是 Jakarta 工程提供的一个用于开发 Web 应用软件的框架，它采用了 MVC 模型 2。[4]

Struts 由 Servlet、标记库、实用类库等构成。其中，Servlet 用于 HTTP 请求的分配及 JSP 的调用；标记库用于页面的动态生成；实用类库用于 XML 的解析及 Bean 的属性设定等。Struts 有 3 个主要的类：ActiongServlet，Action，AcitionForm，它们的调用关系如图 2.5 所示。

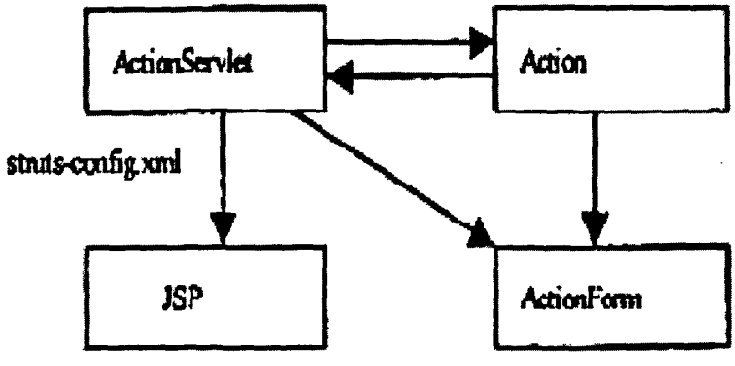

图 2.5 Strusts 工作方式

（1）ActionServlet：完成 Controller 的功能。它根据 HTTP 请求信息，调用 Action 并根据其执行结果，选择调用 JSP 文件．选择 JSP 文件时的配置信息保存在 XML 配置文件中。

（2）Action：完成 Model 中逻辑 Bean 的功能，它是事务处理的实装，完成数据的处理．

（3）ActionForm：完成 Model 中数据 Bean 的功能。它保存从 HTTP 请求中获取的数据，并包含数据的设定及取得等操作．

另外，Struts 中有个重要的配置文件 Strusts-Config.xml，主要用来存放 Action 信息。实际是 Strust 也是模式 2 的一种具体实现，并不是一种崭新的概念和框架。

struts 框架确实为我们进行网站的开发提供了一种新的思路，并在一定程度上实现了框架。但它作为一种研究性的项目，针对实际的应用开发，它仍有许多方面没有进行充分的设计。项目的开发需要的不是实现的可能性，而是要能够为整个开发过程(包括每一个细节)提供一套完备的解决方案。

2.4 MVC 模式应用的关键技术

2.4.1 Java Servlet

1. Servlet 概述

Servlet技术是J2EE的一个重要组成部分。在建立交互式Web应用程序方面，Servlet是非常流行的选择。

Java Servlet是运行于服务器上的程序模块，其目的在于扩展服务器应用的能力，其工作在Web服务器环境中，而且它们是服务器Java开发的关键组件。Servlet由容器管理，实现请求— 响应模型，动态生成内容，同Web客户实现交互。

在谈及Servlet时，不可不提CGI(通用网关接口)，ISAPI/NSAPI(因特网/网络服务器应用编程接口)等相关概念，在Web应用开发的早期，CGI,ISAPI/NSAPI发挥了十分重要的作用，当然现在Servlet是这些技术的高效替代品，目前基本上所有的服务器都支持Servlet API.

2.Java Servlet 在 MVC 中的应用

在MVC模式中，Servlet充当控制器的角色，用来处理HTTP请求，管理应用的

工作流程。

Web浏览器使用HTTP协议向Web应用服务器发送请求，这些请求组成Web应用，在MVC模式中，组件处理Web客户端和Web应用服务器之间的交互。Servlet通过用户所发送的HTTP请求，接收用户全部的输入事件，并把这些信号翻译成为消息传递给封装了请求业务逻辑的JavaBeans或EJB进行交互，最后激活JSP,反馈用户。

2.4.2 JSP

1. JSP 概述

JSP (Java Serve Pages)是在Sun公司的倡导下，并在很多公司的参与下共同建立的一种动态网页技术标准。这个标准扩展了Java Servlet API，能够为Web开发人员提供一种框架结构，从而让人们可以使用HTML和XML模板及Java代码在服务器上建立动态内容，同时，它还是一种安全、快速并且与服务器平台无关的设计方法。[5]

JSP技术是Java Servlet技术的发展和自然扩展，JSP页面由安装在Web服务器上的JSP引擎执行，JSP引擎接收客户端发送的对JSP页面的请求，然后从JSP页面生成响应，并传回客户端。通常JSP页面可以编译成为Java Servlet,这是标准Java扩展。如果调用时，JSP所对应的Servlet不存在，JSP页面将会被编译成为JavaServlet类，并存储在服务器的缓冲中，用户在下次调用时，将直接访问该Servlet，从而增加JSP页面调用的响应速度。

2.JSP 在 MVC 中的应用

在MVC中，JSP充当视图的角色，用来形成用户界面。MVC的处理过程分为视图/表示部分和处理控制组件，视图组件是在浏览器绘制页面时，根据用户界面生成的HTML/XML响应的JSP页面。

控制组件Servlet处理HTTP请求，它们负责创建表示组件使用的bean或对象，还根据用户的动作，决定把请求传送给哪个视图组件。前端组件可以实现为Servlet或JSP页面。视图从业务逻辑的分离有效的解决了开发人员和页面设计人员的角色和职责的分离，为系统的开发维护打下了良好的基础。

2.4.3 JavaBean 与 EJB

1 .JavaBean 、EJB 概述

JavaBean 是为 java 而设计的组件模型，它描述了怎么创建和复用称为 Bean 的组件模型。

EJB （EnterPrise JavaBean）即 企业级 JavaBean,它是 Sun,IBM 和 Oracle 等大公司共同制定的服务器端的组件对象模型，它综合了 RMI(RemoteMethodInvocation,RMI) 、 JavaBean 和 JNDI(JavaNamingandDirectoryInterface)等 Java 平台技术,借鉴了 CORBA 的许多优点,已成为服务器端组件对象模型最主要的标准之一。

2 .JavaBean、EJB 在 MVC 中的应用

在 MVC 中，JavaBean 或 EJB 充当模型（Model）的角色，用来完成事物逻辑即核心业务。当用户象控制器（Controller）发送请求时，控制器根据用户的请求调用适当的 JavaBean 或 EJB 即（Model），JavaBean 或 EJB 将操作结果返回给控制器，控制器调用 JSP(View) 以网页的形式响应用户请求。

2.5 小结

本章首先分析 Web 应用程序所面临的问题主要是技术多样化，各种技术优缺点明显，各有所长，因此需要多种技术的结合使用。从而，面对 J2EE 平台引入面向对象 MVC 设计模式。然后来分析、归纳了当前经典的基于 J2EE 的 MVC 应用模式，期望对我的设计提供参考和借鉴。最后给出了基于 J2EE 的 MVC 模式设计.

第三章 电信卡管理信息系统的需求和总体设计

本系统采用国际先进、成熟 J2EE 体系作为整个系统的体系结构，此技术已经在国内的电信、金融、移动等大型企业中被广泛应用。

3.1 系统设计原则及功能目标

本系统主要功能是以信息技术作为支撑为多种类型卡的生产、销售、库存、调拨出库、统计进行统一管理。同时为了减少人工劳动强度，提高工作效率，系统会和卡平台有很好的交互接口，负责数据的完整传输和存贮，以及各类统计报表。并为决策提供一定支持。根据用户级别的不同，角色的不同，享有不同的权限，所以此系统也包括对用户使用权限的管理。

3.2 系统需求分析

此管理系统应为企业各部门提供准确的数据和为企业发展服务，同时它也是一个相对较大的信息系统，涉及到该公司生产的卡的生产、销售、库存、调拨出库等方面。公司原有的系统为 C/S 架构，采用传统的运作方式，即将过去由手工完成的作业交由计算机来完成，存在很多的弊端，诸如只是采取定时上传数据，并没有真正实现与数据中心实时交换数据，不能及时反映信息，因此存在信息延迟等问题。为了适应公司发展，提高业务运作水平，确保工作质量，根据该公司现有条件，利用 Web 技术，为该公司开发基于 j2EE 的 MVC 模式的电信卡管理信息系统。

3.2.1 系统业务逻辑图

系统采用国际先进和成熟的基于 Java 的 J2EE 软件架构结构实现，这种架构在国内大型应用中被广泛的使用（电信、银行、证券等行业）。客户端采用浏览器方式为用户提供应用服务，采用浏览器方式优点是客户端无需安装任何程序，只需使用操作系统内置的浏览器就可以访问系统。当系统需要升级时只需升级后台应用核心就可以到达整个系统升级目的，从而大大减少升级和维护费用。[7]

本系统是一个通用系统它可以管理多种类型和用途的卡。支持的卡类型包括 IC 卡、磁条卡、条码卡等。支持的卡用途包括电话卡、储值卡、会员卡、银行卡、电子钱包等。

系统功能包括了卡从生产、销售、库存、调拨出库以及统计报表生成的整个过程，用户可以根据自己需要增删功能。系统不但能够完成系统管理上的功能，而且

还能为使用者提供更好的服务功能，以及为管理者和决策者提供强大的管理决策功能。系统业务逻辑图如图 3.1 所示：

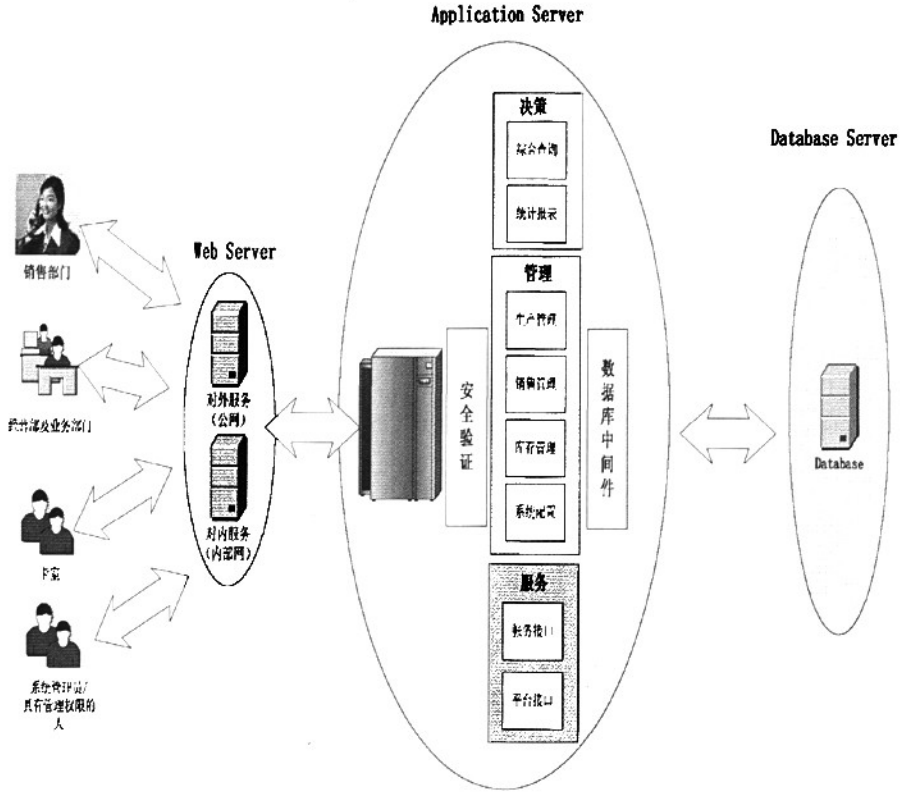

图 3.1 电信卡管理系统逻辑图

3.2.2 系统用例图

用例图是用例的可视化表示，它给出系统的外部行为视图或者说它提供计算机系统高层次的用户视图，表示从外部活动者的角度来看系统是如何使用。在确定了参与者和用例的基础上利用用例图可表示出参与者和用例之间的联系。[14]下面给出系统参与者相关的用例图如图 3.2 所示。

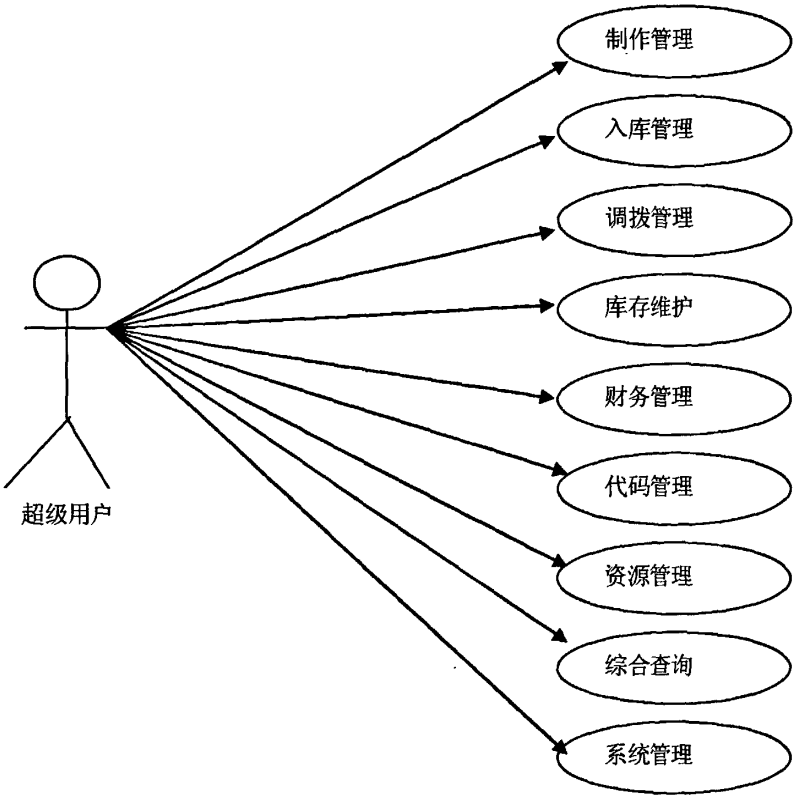

图 3.2 电信卡管理信息系统用例图

此用例图是粗粒度的业务用例图，主要从整体的角度体现系统中以超级用户身份参与业务活动的关系，但是根据需要大多数情况下各用例可根据实际业务状况按使用者划分为更细粒度的子用例，只是普通用户是无权使用系统管理模块。

3.2.3 系统功能结构

通过对业务流程的调查和研究，按照该系统的业务需求和工作层的具体需求，结合软件设计的需要，从功能结构来划分系统。为保证目标系统的可重用性，同时考虑到目标系统要逻辑层次分明、使用方便快捷，得出系统功能模块结构图如下图 3.3 所示，

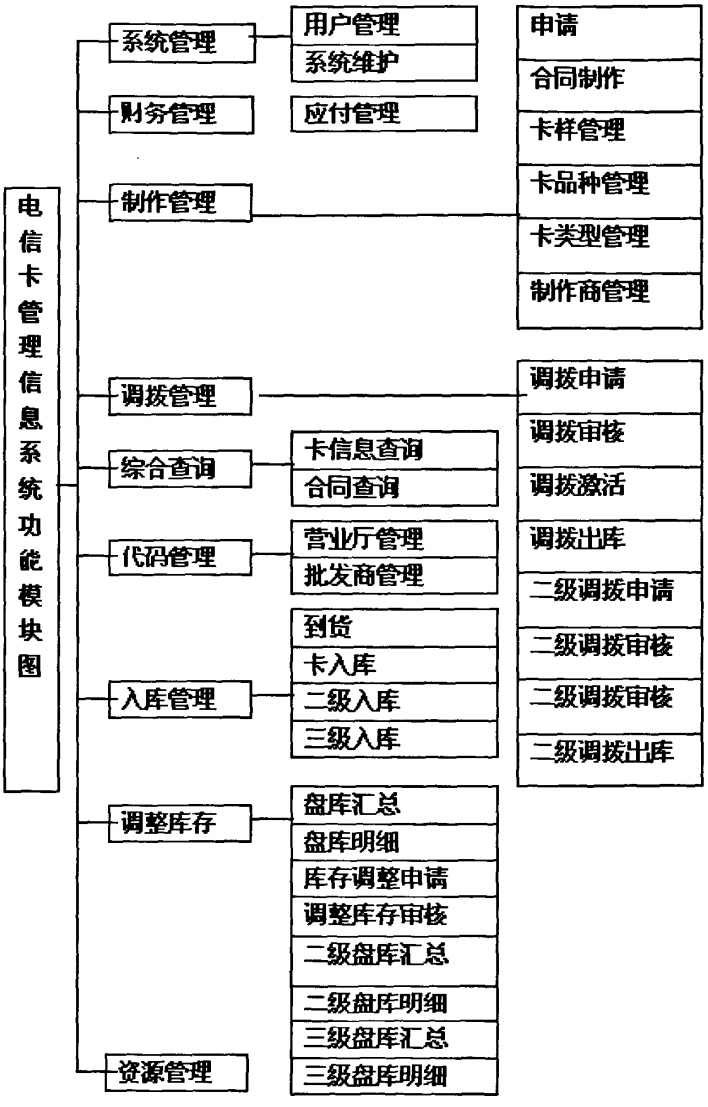

图 3.3 系统功能模块结构图

系统管理员(SA)根据员工岗位和工作职责的不同，来进行角色管理和相关权限的管理，赋予相关工作人员相应的功能模块访问权限。以不同的用户登陆本系统将看到不同的界面，享有不同的功能。比如销售人员只能享有零售和批发功能，而不能有对其他模块操作。

下面讲解一下主要模块需要实现的功能:

(1)制卡管理模块

此模块主要是完成卡的生产和制作过程，这在卡的管理中是很重要的一个环节，对卡制作的管理可使管理者了解卡的制作过程中生产商、制作成本、数量、生产周期等重要信息。以便能够最大限度的节约成本和提高效率。在制作管理中主要是对生产商、卡信息、生产任务进行统一管理。

用例图如图 3.4 所示

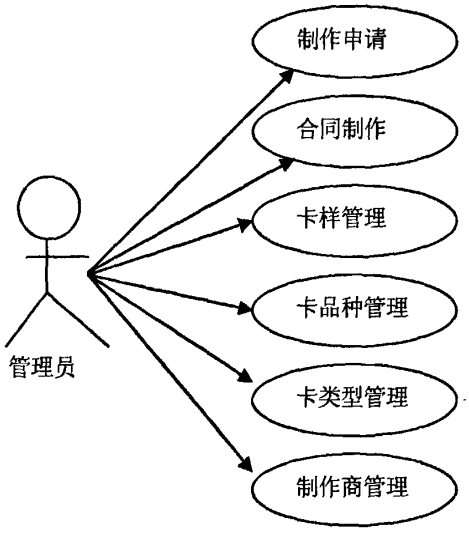

图 3.4 制卡管理模块用例图

对于功能模块中的子功能模块，其内部实现进一步细化，此处将每个子功能划分为三个层次：

(A) Sbean：参数解析，负责响应控制器的调用。是 Model 和 Control 的接口。

(B) Ebean：事物逻辑的实现。

(C) Dbean：数据库操作。

每个功能模块拆分成三个类来实现，结构上更加清晰，层次分明，便于管理和维护。为了统一接口，定义三个虚基类：

HiEnd_SBean：提供 Sbean 的标准接口.

HiEnd_Ebean：提供 Ebean 的标准接口.

HiEnd_Dbean：提供 Dbean 的标准接口.

下面以制作申请功能模块为例说明：

表 3—1 制作申请功能类实现

类名称(Class)	功能描述(Function)
SB_ApplyCard	解析用户参数，控制器接口
EB_ApplyCard	完成事务逻辑操作
DB_Applycard	数据库操作，数据库接口

表 3—2 制作申请功能类继承关系

类名称(Class)	基类(Based Class)
SB_ApplyCard	HiEnd_SBean
EB_ApplyCard	HiEnd_EBean
DB_Applycard	HiEnd_DBean

表 3—3 SB_ApplyCard 类功能实现

方法(Method)	功能(Function)
DP_NewApply	处理新建制作申请参数， 调用 EB_ApplyCard 对应方法 NewApply
DP_DelApply	处理删除制作申请参数， 调用 EB_ApplyCard 对应方法 DelApply
DP_ModiApply	处理修改制作申请参数， 调用 EB_ApplyCard 对应方法 ModiApply
DP_QueryApply	处理查询制作申请参数， 调用 EB_ApplyCard 对应方法 QueryApply
DP_ConfirmApply	处理确认制作申请参数， 调用 EB_ApplyCard 对应方法 ConfirmApply

我们可以看 SB_ApplyCard 是控制其的接口，负责处理用户的参数，并将其传送给 EB_ApplyCard，完成逻辑操作。

由 EB_ApplyCard 调用 DB_Applycard 实现数据库的操作，并将操作结果反馈给控制器(Controller)。

表 3—4 EB_ApplyCard 类功能实现

方法(Method)	功能(Function)
NewApply	完成新建制作申请逻辑操作,调用 DB_CardApply 完成数据库操作
DelApply	完成删除制作申请逻辑操作,调用 DB_CardApply 完成数据库操作
ModiApply	完成修改制作申请逻辑操作,调用 DB_CardApply 完成数据库操作
QueryApply	完成查询制作申请逻辑操作,调用 DB_CardApply 完成数据库操作
ConfirmApply	完成确认制作申请逻辑操作,调用 DB_CardApply 完成数据库操作

由 DB_ApplyCard 完成数据库操作，DB_ApplyCard 继承、封装了数据库的 JDBC 连接池(HiEnd_DBPool)，实现数据库的增加、修改、删除、和查询操作。

表 3—5DB_ApplyCard 类功能实现

方法(Method)	功能(Function)
AddData	对数据库的插入操作
ModiData	对数据库的修改操作
DelData	对数据库的删除操作
QueryData	对数据库的查询操作

我们这样三层设计的好处是接口干净、层次清楚便于管理和维护。我们将模块的基本功能尽量的集中到 Ebean 中，而 Ebean 只负责逻辑处理，不必考虑参数和数据库操作的问题。否则某个模块的功能实现分割成若干个类，不易于管理。其他功能模块的设计也类似，这里不在赘述。

(2)入库管理模块

入库管理将对制作完毕，准备入库的卡信息进行管理。同时，根据操作者的权限不同，系统的二级库及三级库入库操作均在此功能下完成。该部分功能介绍如下：

a) 到货：到货信息的登记及管理。
b) 入库：处理完毕到货信息之后，此批到货将进行入库处理
c) 入库查询：查询入库信息
d) 二级库入库：处理二级库入库信息
e) 二级库入库查询：二级库入库信息查询
f) 三级库入库：处理三级库入库信息
g) 三级库入库查询：三级库入库信息查询

入库管理模块用例图如图 3.5

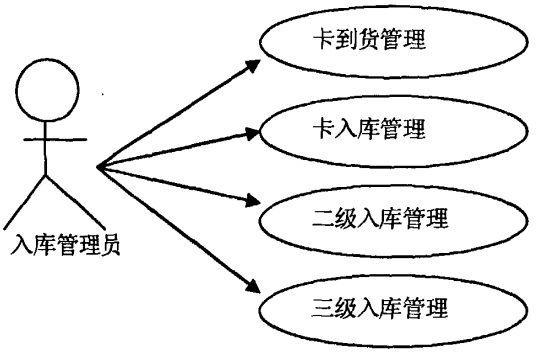

图 3.5 入库管理模块用例图

(3).调拨管理模块

调拨管理中包括经营部或部分部门向卡室管理员申请调拨，调拨审批，激活，卡出库等，以及营业厅向经营部申请调拨的过程，也包括审批，确认，出库等过程。

a) 调拨申请：经营部等向卡室申请
b) 调拨审核：卡室审核经营部申请
c) 调拨激活：卡室确认激活信息
d) 调拨出库：出库单打印
e) 调拨出库查询：查询出库信息

f) 二级库出库申请：营业厅向经营部申请
g) 二级库出库审核：经营部审核营业厅申请
h) 二级库出库确认：经营部确认出库信息
i) 二级库出库：出库单打印
j) 二级库出库浏览：查询二级库出库信息

调拨管理模块用例图如图 3.6

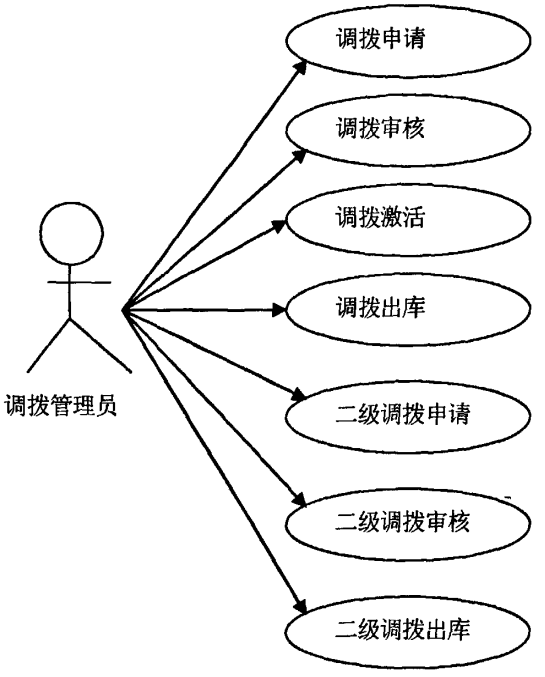

图 3.6 调拨管理模块用例图

(4).报废管理模块

由于卡会由于某些原因（损坏、退货、过期等）不能被出售，报废功能就是为用户提供对不能出售的卡进行报废处理。报废时用户需要填写报废原因，数量、卡号范围，系统自动记录报废信息以便查询和检查。

a) 报废申请：由卡室或卡库管理员填写报废申请
b) 报废审核：由管理员填写报废审核，审核之后，卡即报废。

(5).调整库存模块

仓库管理员需要定期对仓库中的卡进行数量进行核对，系统提供盘库功能为仓库管理员生成所有卡库存盘点单，以便管理员核对。

a) 盘库汇总：库存信息汇总

b) 盘库明细：明细库存信息
c) 调整库存申请：如盘库结果和计算机内存储的信息不同，需要做盘库申请
d) 调整库存审核：调整库存的审核过程，审核通过，库存调整完成
e) 调整库存浏览：察看调整库存的内容
f) 二级库盘库汇总：二级库信息汇总
g) 二级库盘库明细：二级库库存明细
h) 三级库盘库汇总：三级库信息汇总

调整库存模块用例图如图 3.7

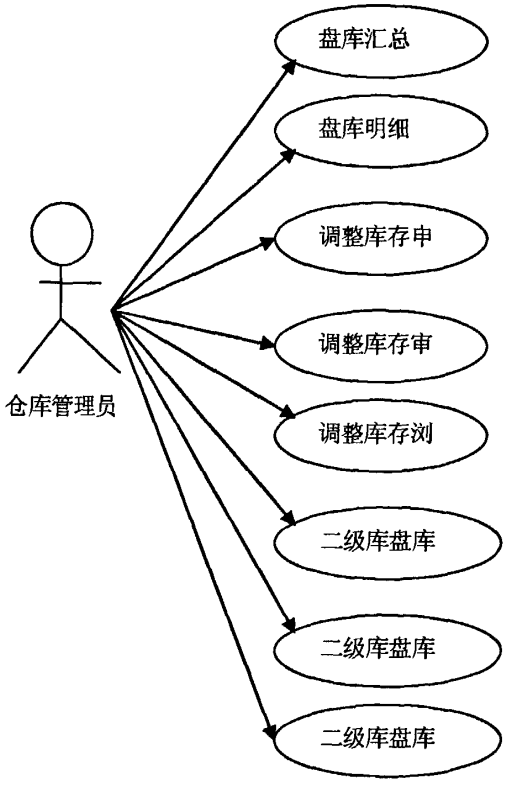

图 3.7 调整库存模块用例图

(6).财务管理模块

财务管理方面主要是针对与合同制作方面的结款情况完成的。

a) 应付管理：记录付款方式及时间等信息
b) 应付管理查询：查询应付信息

当卡制作完毕，完成验收入库后，系统会有一笔应付款信息进入财务管理系统。当财务完成付款之后，系统会记录下付款情况。

(7).综合查询模块

a) 卡信息查询：在查询框中输入卡号的后5位数，系统自动查询出尾号为这几位数的卡信息。选择需要查找的卡，回车即可看见此卡的全部信息。

b) 合同信息查询：在查询框中输入合同号的后5位数，系统自动查询出尾号为这几位数的合同信息。选择需要查找的合同，回车即可看见此合同的全部信息。

(8).销售管理模块

营业厅的营业员需要通过此系统登记每一笔销售业务，系统能够监控其销售金额以及折扣限度等信息。

如果销售金额及销售折扣以及销售数量不相符合时，系统会禁止继续操作。并提醒其此操作有误。营业员需要输入卡的后几位，系统自动调出卡信息。点击加入，卡号即可加入购买列表当中。

(9)系统管理模块：

a) 用户管理：允许用户增加、删除、修改系统用户。并且支持部门用户管理。

b) 权限管理：为了保证系统保密性、安全性，系统权限细化到每个小功能，即管理者可以为一个用户详细设置使用权限，用户只有被赋予权限才可以执行相应的功能。权限管理支持分级，对于不同的部门以及不同的职务，系统均能够单独为其设置权限。每一个部门均有相应自己部门的权限，和数据查询权利。

3.3 系统总体设计

3.3.1 系统的总体架构

本系统采用 MVC(Model—View—Control)的模式来设计系统结构并利用 J2EE 的技术来实现。本着业务逻辑和表现逻辑分离，对系统资源进行合理有效管理，优化系统性能的指导思想，设计总体框架如图3.8所示。

系统框架主要分为以下几部分：

(1)总控模块：用户请求由总控模块进行统一管理，总控模块作为用户请求总的入口，接收用户请求，调用业务逻辑处理。

(2)业务逻辑处理：业务处理模块是用户请求的具体功能实现。

(3)表示逻辑处理：业务逻辑处理结束后，表示逻辑(JSP、XML、HTML 等)从业务逻辑处理中提取处理结果进行显示。

(4)数据访问：数据库服务模块必须对数据库资源的访问进行有效的管理。考虑到系统的可扩展性，数据库服务定义统一的数据库访问接口，可自行扩展数据源和数据库的访问方式。

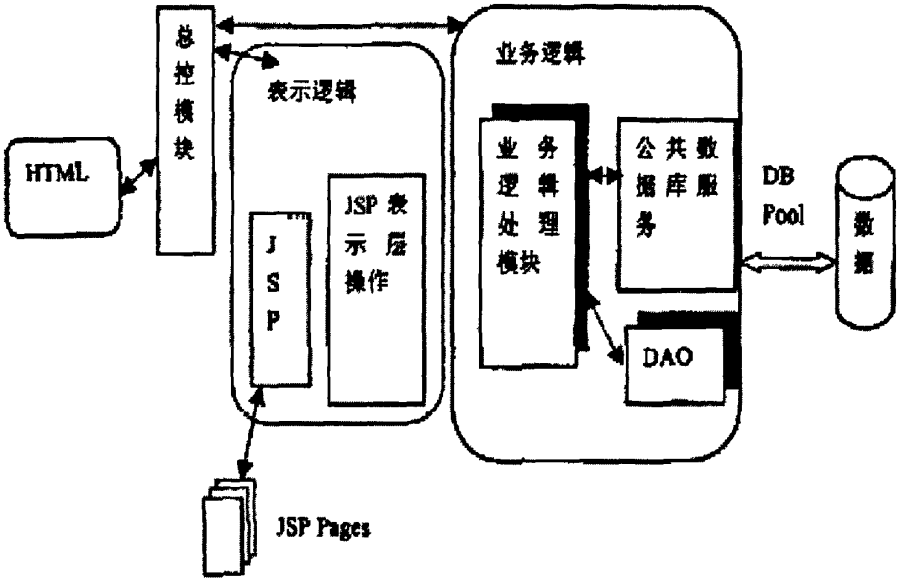

图 3.8 系统总体框架

该系统采用浏览器 / 服务器模型(Browser / Server)。一般的，客户通过浏览器发送 HTTP 请求给服务器端 Web 服务器，Web 服务器接收该请求并且进行相应处理，然后将处理后的结果返回到客户的浏览器中。在客户端，浏览器中呈现的正是该系统的视图部分。

该系统将很好地结合使用 JSP 与 Servlet，由 JSP 把用户提出的 httpRequest 送达 Servlet，Servlet 根据请求的类型不同分别进行不同的处理，选择不同的操作模块，具体地说比如说对数据库的读写更新查询操作等，Servlet 处理完这些比较复杂的请求之后，选定 JSP 页面把最后的结果回应给使用者，此时的了 SP 就只是起到了 view 的作用，并不做逻辑上运算只是将 Controller 方面传过来的资料展示给用户而已。[18]

其中在本系统中主要起到控制作用的是由 HiEnd_ControlCenter 完成的，在下面我们可以详细地看到这一点。在系统中对应于各个模块都有各自的三个层次的类(Sbean,Ebean,Dbean)，负责连接数据库完成各类操作，称为 Model，这部分应主要包括以下几种操作：

(1)对应于 httpRequest 的各种操作。

(2)逻辑运算操作：比较排序等。

(3)对数据库地更新操作。

(4)与数据库相对应的 object 数据主实体类，对数据实体封装成 HiEnd_ResultSet 类。作为公有类来存储从数据库读出的数据，这样设计使得数据的接口统一，便于控制器将数据转发给视图，完成数据的显示。

(5)各类查询操作。这样系统就把数据的逻辑运算、页面的流程转向的控制、数据的展现部分切分得非常干净，提高书写程序的方便性，代码层次结构更加清晰合理，易读易用。图 3.9 是电信卡管理信息系统的详细框架图，在设计上采用了 MVC 设计模式的思想，较好地做到了层次清晰一目了然。

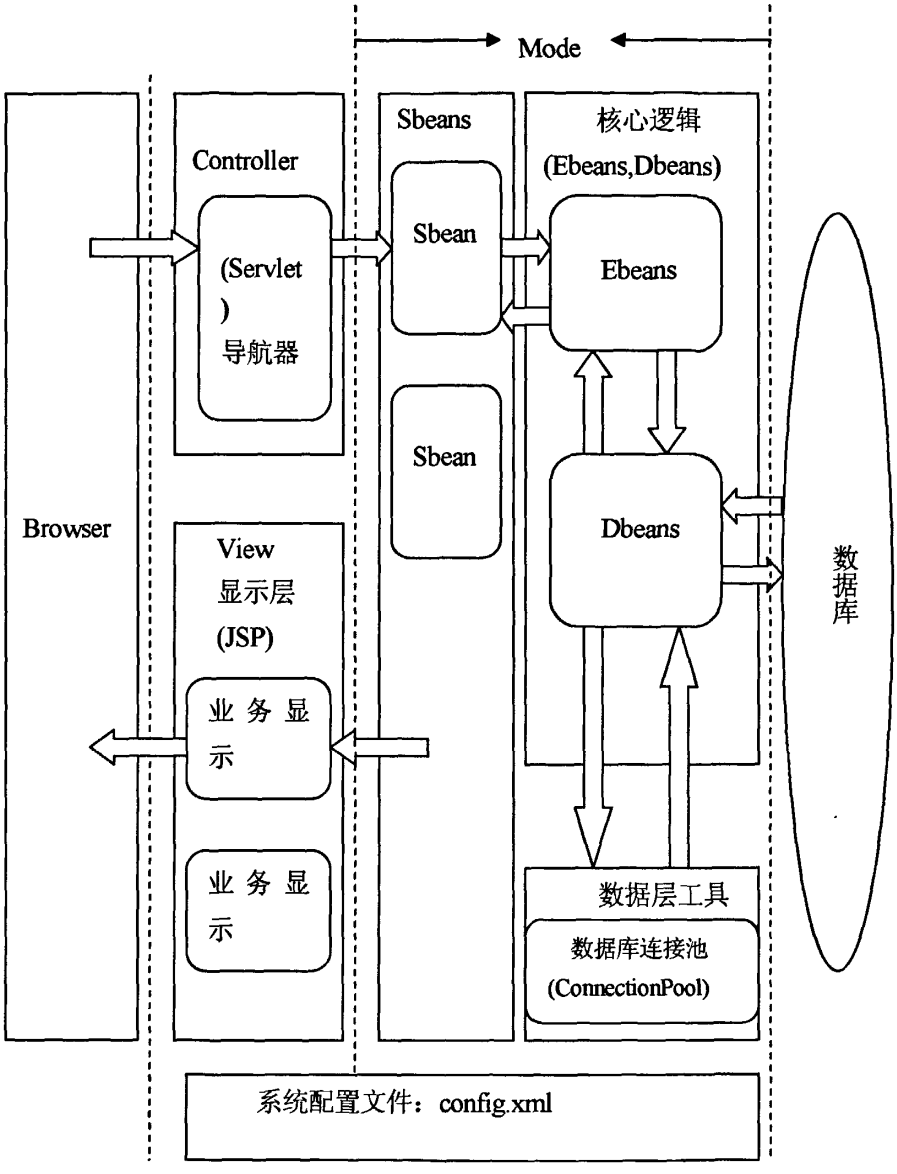

图 3.9 电信卡管理系统详细构架图

3.3.2 系统处理流程图

系统的使用对象是该企业的员工、客户、供应商，系统为每个初始用户名和密码，针对不同人员的使用权限，系统还对每个用户配，用户登陆只能使用自己权限范围内的业务，其它业务不能使用，处理流程图如图 3.10 所示。

用户以不同的身份登录，所能看到的系统页面有可能不同，对面所能进行的操作也有可能不同。

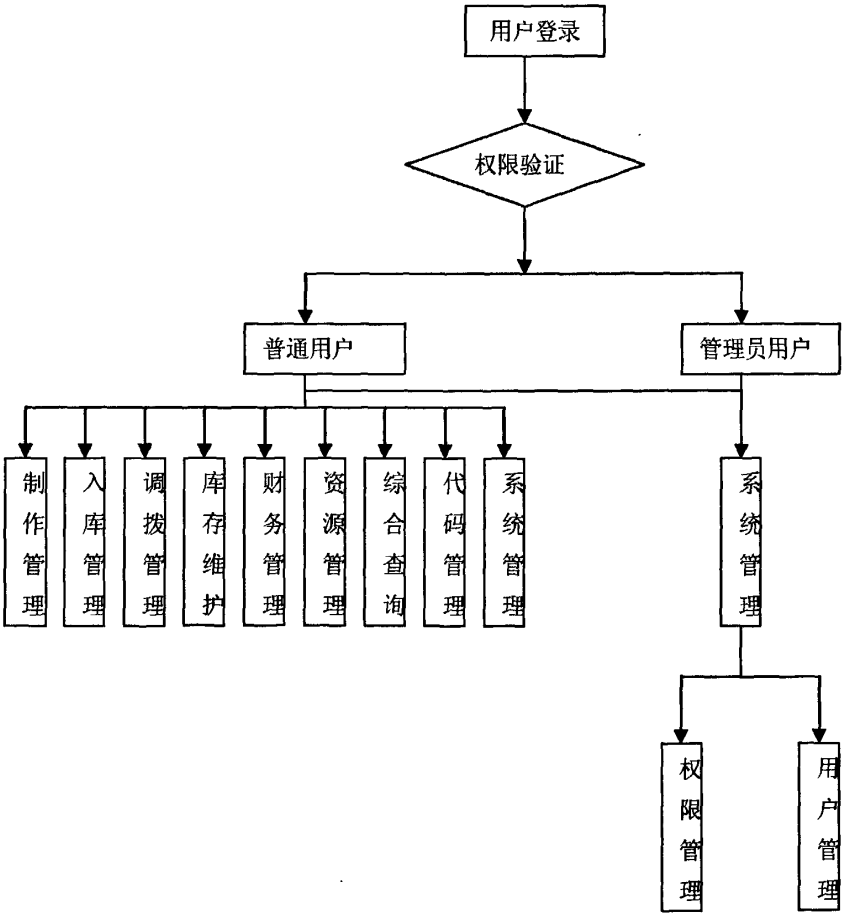

图 3.10 电信卡管理系统流程图

3.3.3 系统包图

随着系统的逐渐变大，理解和修改这个系统也就变得更加困难，为此总是希望能将复杂问题进行分解。但是如何将大系统拆分成小系统并不是一个简单的问题。

随着面向对象技术的成熟，解决这个问题的一个基本思路是将许多类集合成一个更高层次的单位，形成一个高内聚/低耦合的类的集合。这个思路被松散的应用到许多对象技术中。UML 把这种分组机制称作包。[19]

而分组需要遵循一定原则，在 UML 中，最有用的和强调最多的就是依赖性，通常，包图所显示的是类的包以及这些包之间的依赖关系。在类的关系中，导致依赖性的原因有多种。比如，一个类向另一个类发消息；一个类是另一个类的数据成员：一个类用另一个类作为它的某个操作的参数等。

如果两个包中的任意两个类之间存在依赖关系，则这两个包之间存在依赖关系。这种依赖关系是不传递的，层次结构的这种非传递特性正是人们经常采用层次结构的原因之一。在 Java 中，这是“import”行为的语义。

根据以上原则，结合本系统的结构，分包策略如下：

首先控制器(controller)封装在 servlet 包中，功能模块都封装在包 telcomcard 包中，而公用类封装在 util 包中，Jsp 的标签封装在 taglib 包中。然后再分别细分 telecomcard 包和 util 包。

在 telcomcard 包中，将与某一功能模块相关的类封装在一个子包中(如将制卡申请功能中的相关类都封装在包 telecomcard.cardApply 子包中)，在此包内，根据系统结构中模块的 Model 部分划分，再主要细分出三个包：

（1） Sbean 包：对应于 httpRequest 的各种操作类；

（2） Ebean 包：逻辑运算操作类：比较排序类等；

（3） Dbean 包：对数据库操作

在 util 包中，再分别将某些相同作用的公用类封装在一个子包中，如将数据库连接池 DB_ConnectionPool 封装在 util．db 子包中。

系统包的逻辑结构图如图 3.11 所示：

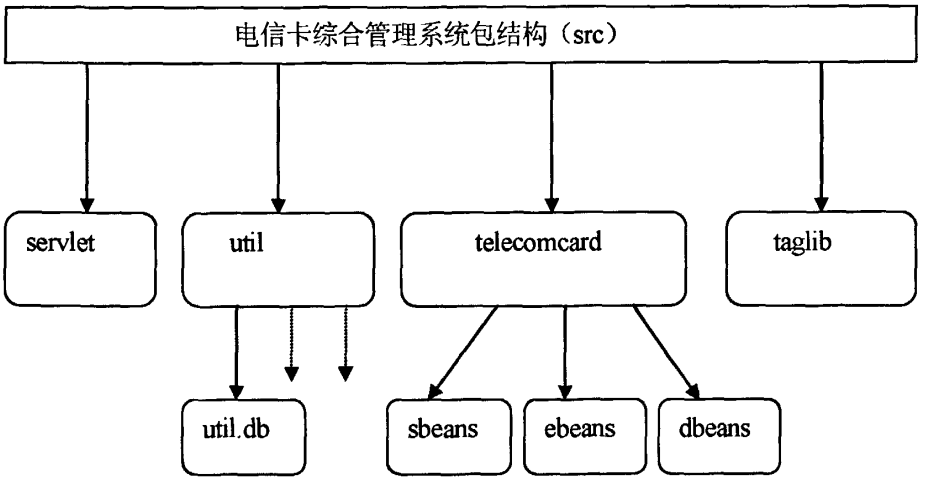

图 3.11 系统包逻辑结构图

3.4 数据库设计

数据库是信息系统的核心和基础，它把信息系统中大量的数据按一定的模型组织起来，提供存储、维护和检索数据的功能，使信息系统可以方便、及和准确地从数据库中获得所需的信息。一个信息系统的各个部分能否紧密地合在一起以及如何结合，关键在于数据库设计，因此数据库设计是信息系统开发和建设的重要组成部分，只有对数据库进行合理的逻辑设计和有效的物理设计才能开发出完善、高效的信息系统。[6][15]

3.4.1 数据库表结构设计

本系统已经分析所涉及的基本对象，它们的内容就是本系统需要存储的最主要信息。按照面向对象的关系数据库设计原则，一个对象基本对应一个表中的一条记录或一个记录集，再按照遵循数据库设计第三范式的原则，进行数据库设计。

本系统中的核心的数据对象是卡信息，包括卡样、卡类型、卡介质、卡面值、激活方式、激活平台、卡号等基础信息。根据第三范式的原则，我们将卡基础信息设计为主从表的关系，如图 3.12：

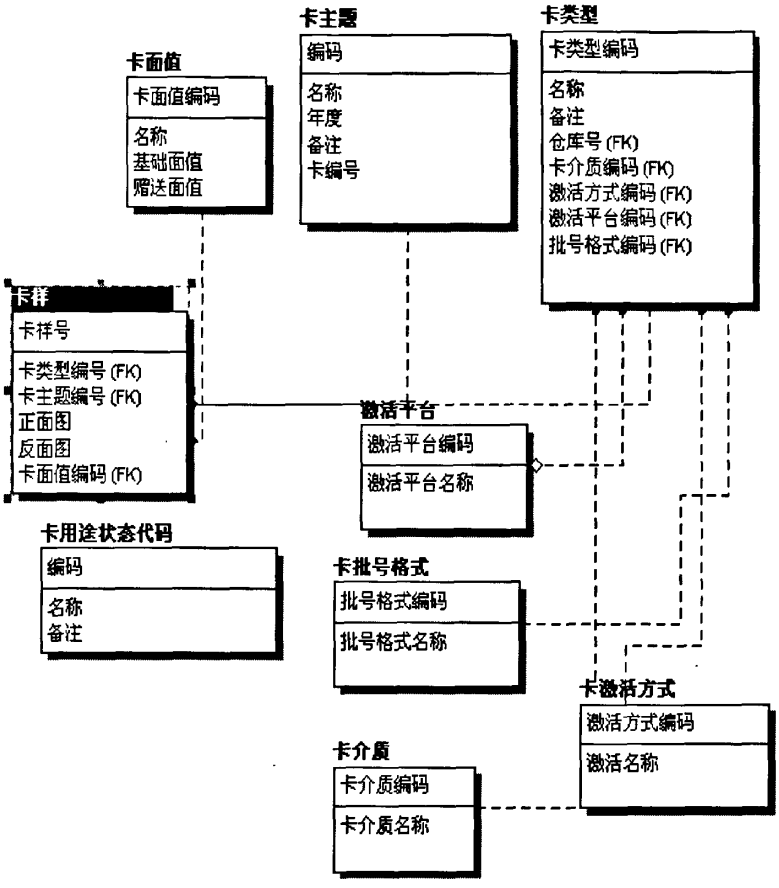

图 3.12 卡信息表结构

作为本系统的另一基础信息为用户的管理，包括角色和权限，同样遵循三范式原则。如图 3.13：

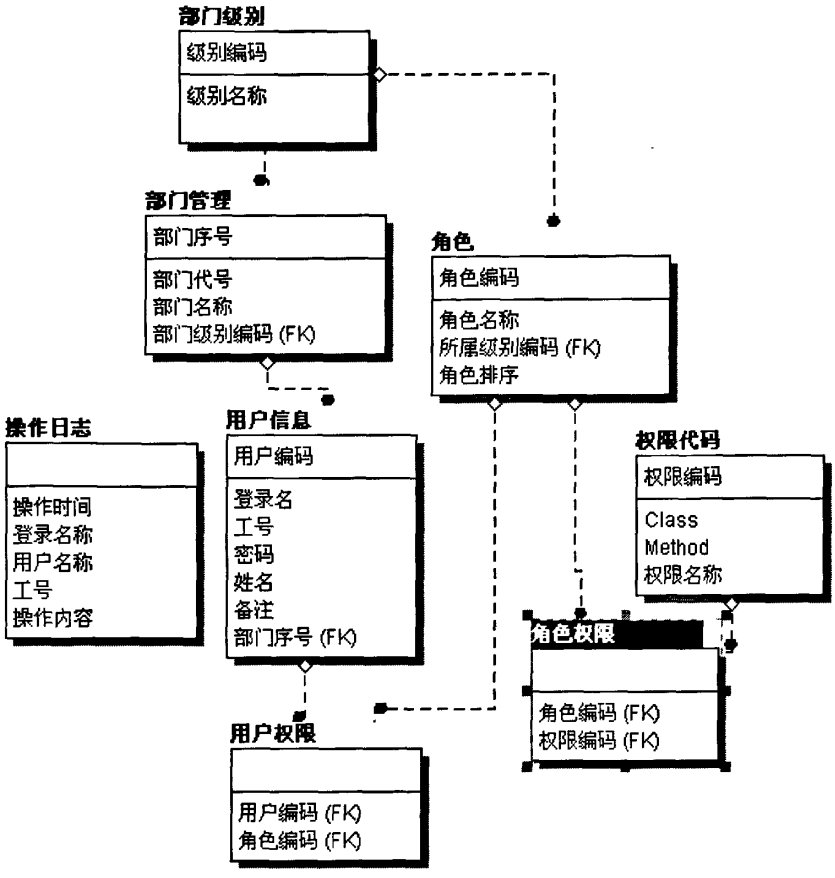

图 3.13 用户信息表结构

以上给出了卡信息和用户信息的设计 ER 图。对于卡的从申请到制作到存储销售所有涉及到的表都采取以上原则设计，最大限度的减小数据冗余，保障数据的一致性。

3.4.2 数据层设计

数据库有关的操作流程一般为：建立数据源，加载驱动程序，建立连接，建立语句对象，获取结果集合，若有必要，将处理结果写回数据源或添加数据到数据源，关闭对象。如果处理出错会抛出异常 SQLException 和 BatchUpdateException。[7]

在本系统中，为了提高扩展性和性能，采用了连接池(connection pool)技术，封装为(HiEnd_DBConnectionPool 类)。通过在服务器启动的时候建立一个数据库的连接池，连接池可减少建立和中断数据库连接时的系统开销。如果有数据库连接的需求，从池中选择一个，而不是创建一个新的连接，当连接不需要时，要释放该连

接到连接池，以便其它的数据库请求使用。这样数据库访问的效率就得到了显著的提高，尤其电信卡系统中数据量是相当庞大的。

建立数据库的连接池方法如下所示：

(1) 获取数据库连接：HiEnd_DataBase theDataBase = m_DBManager.GetConnection();

(2)设置 sql 语句：String sql="insert into pt_CardType(CardTypeID,CardName,Remark) values('"+cardtypeId+"','"+Name+"','"+Remark+"')";

(3)配置 Statement 语句：Statement stmt = theDataBase.CreateStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,java.sql.ResultSet.CONCUR_READ_ONLY);

(4)执行数据库操作：int i=stmt.executeUpdate(sql);

(5)释放数据库连接：theDataBase.Release();

HiEnd_DataBase 是对 HiEnd_DBConnectionPool 类的封装，HiEnd_DataBase 除了封装了 HiEnd_DBConnectionPool，同时由于这是一个多用户的系统，我们引入了事务的概念，保证了数据的一致性。

任何数据源的修改只要更新配置文件就可以解决了，数据引擎可以采用 JDBC-ODBC 连接桥，也可以使用 JDBC,当然考虑到后者效率高，一般直接用后者，但有些数据库对 JDBC 不直接支持，可以考虑使用 JDBC-ODBC 连接桥。这种方法实现的应用，可以很容易的改变它的数据源类型,这个模式分离了业务逻辑和数据访问逻辑，增强了应用的可伸缩性，因为数据源改变变得很容易，对数据访问没有任何限制。连接池的设计结构如表所显示：

表 3-6 数据库连接池(HiEnd_DBConnectionPool)类实现

方法(Method)	功能(Function)
Init()	初始化，读取数据库配置文件 db.conf
CreateConnectPool()	创建数据库连接池
GetDBManagerInstance()	创建数据库连接实例
GetConnection()	获取数据库连接
ReleaseConnection()	释放数据库连接

表 3-7 HiEnd_DataBase 类实现

方法(Method)	功能(Function)
CreateConnection()	创建新的连接
GetConnection()	获取连接池已有连接
CreateStatement()	创建 statement
CreateCallableStatement	调用存储过程
StartTransaction()	开始事务
StopTransaction()	事务结束
Commit()	提交数据操作
Rollback()	回滚

对综合查询或者其他复杂查询，涉及到多表查询并且查询频率较高，为了提高查询效率，我们采用了视图来实现。比如说要获得用户信息需要从 ptuserinfo，pt_Code_Department, pt_Code_DepartmentLevel 等表中分别提取姓名、部门，部门级别等信息，构造视图 VI_USERINFO 如下：

```
CREATE OR REPLACE VIEW VI_USERINFO ( USERID,
USERNAME, WORKERNO, PASSWORD, FULLNAME,
REMARK, DEPTSN, DEPTID, DEPTNAME,
DEPTLEVELID, DEPTLEVELNAME ) AS
select ptuserinfo.*,pt_Code_Department.DeptId,pt_Code_Department.DeptName,
pt_Code_DepartmentLevel.*
from ptuserinfo,pt_Code_Department,pt_Code_DepartmentLevel
where pt_Code_DepartmentLevel.DeptLevelId = pt_Code_Department.DeptLevelId
and pt_Code_Department.DeptSn = ptuserinfo.DeptSn;
```

这样对于用户信息就抽象到视图 VI_USERINFO 中，对它的查询就象是一个表一样快速方便。

上面提到了数据操作的事务性，若干个子任务可以组成一个事务来完成，当子任务较多，操作负责时，我们采用存储过程来实现，保证其速度和稳定性。举例说

明销售过程中卡的存储是起始号到终止号来存储，每卖一批卡就要将该卡记录到销售表中，而将原来的卡库中的记录修改。

使用了视图和存储过程，极大大提高了数据操作的效率和稳定性，由于视图和存储过程都在数据库服务器，很方便可以实现视图和存储过程的共享。但同时它也消耗很多系统资源，因而对于视图和存储过程的使用要适度，不要过分使用，本系统前面也提到了对于频繁使用的多表查询我们使用了视图，对于复杂的事务操作我们采用存储过程。并没有强调全部数据操作都使用视图或存储过程，对于视图和存储过程的使用尽量达到一个合理的平衡点。[21]

3.5 小结

本章对管理信息系统进行了总体设计。规范和分析了系统的设计原则和功能目标，结合系统需求及业务流程，设计出相应的系统框架和数据库设计，为下一章电信卡管理信息系统的具体实现提供准备。

第四章 电信卡管理信息系统的实现

4.1 环境的选取与配置

经过以上分析，我们已明确采用 MVC 架构和 J2EE 的技术，经过对相关的当前流行开发工具的分析比较，确定开发环境如下：

（1）开发工具：

Servlet 和相关类的开发工具：Jbuilder

JSP 页面开发工具：DreamWeaver2004

后台数据库：Oracle9i，SQL Server 2000

Oracle9i 数据库开发工具：toad

（2）软件环境：

系统平台：流行平台均可，本文采用了 Windows 2000 Server

Web 服务器：本文采用了 Tomcat

在我们的电信卡管理信息系统中，选择 Apache 组织的 Tomcat 作为 Java Web Server，它支持 JSPl．1 和 SerVlet2．2 规范，它是免费的。配置文件 Webconfig.xml,它主要包含字符集配置、控制器(Servlet)配置、Jsp 配置和标签库的配置。[8]

4.2 控制器(Controller)实现

控制器流程图 4.1 如下：

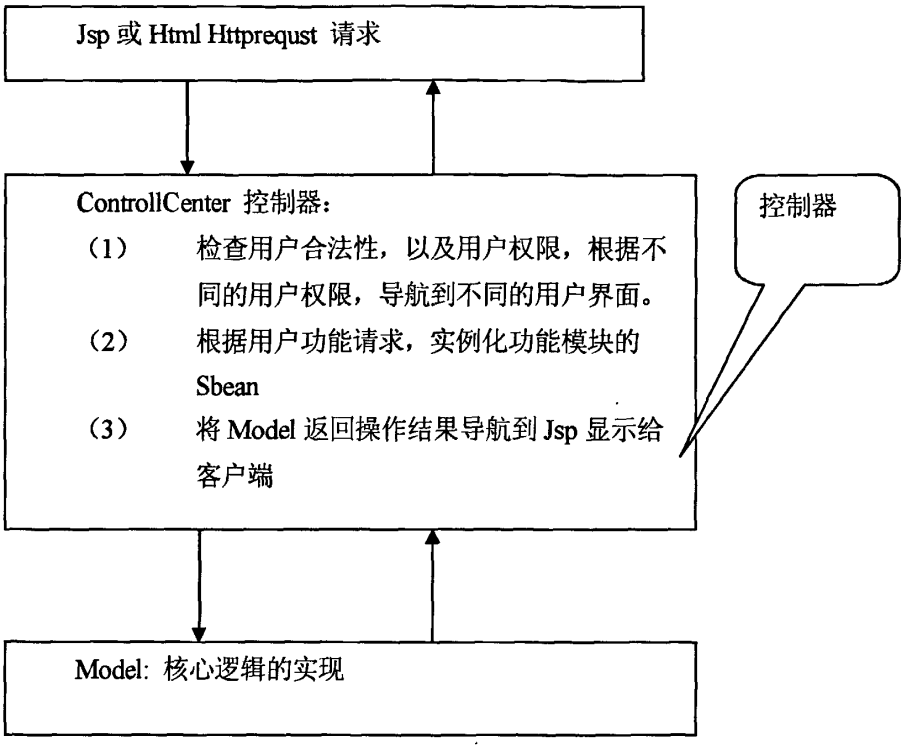

图 4.1 控制器工作流程图

使用控制器这一概念是为应用程序的控制提供一个集中点，在此处所有的客户端请求都可以先进行处理。由于控制器在客户端的输入与模型间扮演着一个中间人的角色，控制器可以为每一个客户端请求提供一些公用的功能，如安全、日志以及其它重要的功能。

MVC 框架的控制器由 HiEnd ControlCenter 这个类来担当。这个控制组件提供了所有发送到的 MVC 控制器的请求的入口点，它截取和分发这些请求到相应的 SBean。具体分发到哪个 SBean 根据 sbean dispatch.xml 的配置文件来决定。

用户登陆界面，如图 4.2，可以按照不同的角色进入系统。用户可以选择进入管理系统、一级库管理、二级库管理、三级库管理以及营业厅销售等不同的系统。在进入这些系统之后的界面是不一样的。

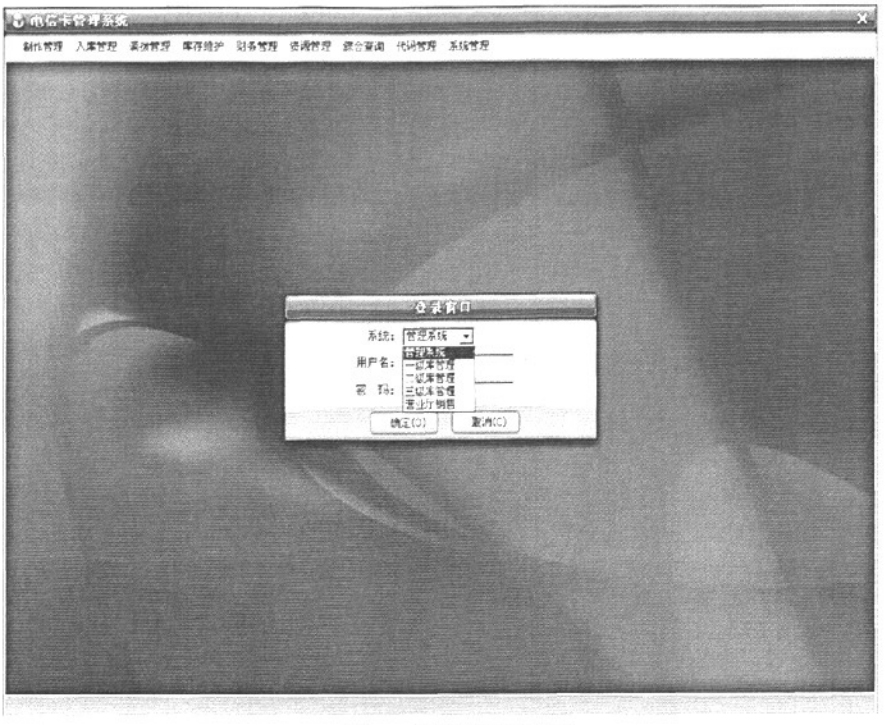

图 4.2 系统登陆界面

HiEnd_ControllCenter 将根据不同的用户，不同的权限，导航到不同的主界面，假设以管理员的身份登陆。HiEnd_ControllCenter 的比对权限并导航的核心代码：

```
if ( !ignoreUser )
                    {
                        HttpSession theSession = m_Request.getSession(false);
                        if(null != theSession)
                        {   String szUserID = "";
                            if (null != theSession.getAttribute("userID"))
                                szUserID = theSession.getAttribute("userID").
                                    toString();
String szRight = XMLAnalyze.GetExcuteRight(m_szClassName ,m_szMethod,
m_RightsXMLFile);
                            if (szRight == null || szRight.equals("") ||
                                ComparaUserRight(szUserID, szRight))
                            {
```

```
                                    bDoMethod = true;
                                }

if(ComparaUserRight(szUserID,m_szClassName,m_szMethod)){
                    bDoMethod = true;
                                }
```

根据不同用户的导航核心代码：

```
                    if (info.DeptLevelId.equals("1"))
                        url = "main_1.jsp";//普通用户 1
                    else
                    if (info.DeptLevelId.equals("2") )
                        url = "main_2.jsp";//普通用户 2
                    if (info.DeptLevelId.equals("0"))
                        url = "main_sys.jsp";
```

功能：根据登陆用户的级别不同导航到不同的主页，例如普通用户 1、普通用户 2、系统管理员等，系统管理员用户航到 main_sys.jsp。

```
                    if ( !url.equals("") )
                    {
                        HttpSession session = m_ControlCenter.m_Request.
                            getSession(true);
                        session.setAttribute("userID", String.valueOf(UserID));
                        session.setAttribute("userName",

String.valueOf(info.sFullName));
                        session.setAttribute("userInfo", info);
                    }
                    else
                        url="logonFailure.htm";
```

功能： 将用户信息保存在 session，以便能跟踪用户，如果用户信息错误，导航到错误提示 logonFailure.htm。

```
                rd = m_ControlCenter.m_Request.getRequestDispatcher(url);
               m_ControlCenter.m_Response.sendRedirect(url);
               rd.forward(m_ControlCenter.m_Request,m_ControlCenter.m_Response);
```

导航到与用户权限匹配的主页，这里我们以系统管理员身份登陆，点击系统的各个菜单进入不同的系统功能。主页如图 4.3：

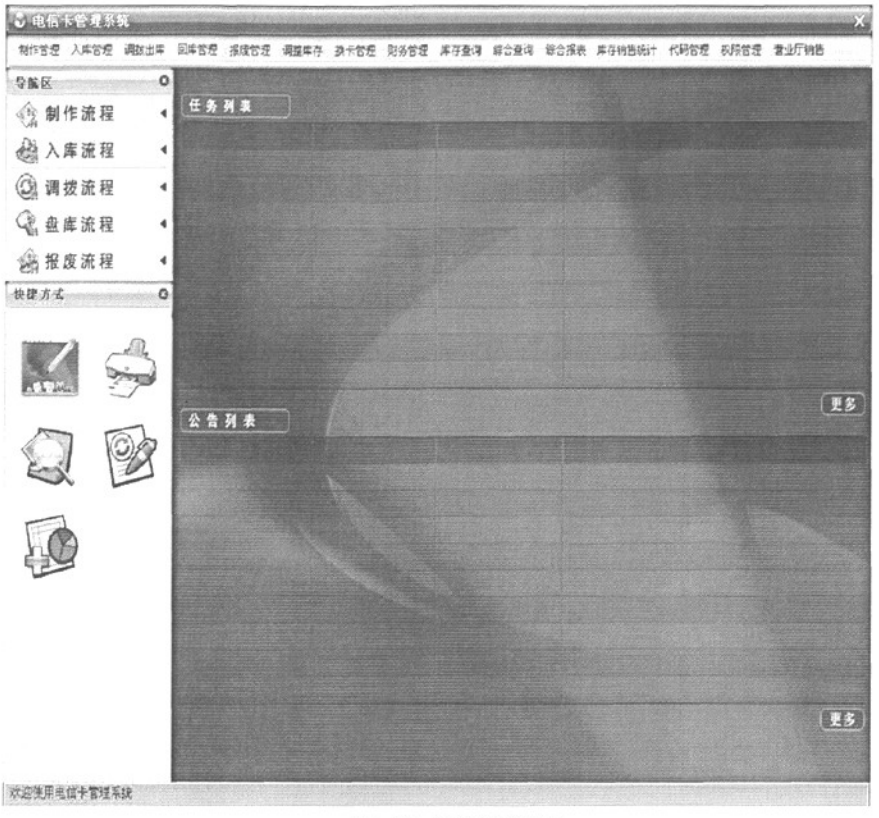

图 4.3 系统主界面

点击不同的菜单，即由控制器 HiEnd_ControllCenter，实例化对应的 Sbean，进而调用 Ebean,Dbean 来实现用户操作。其核心代码如下：

```
public boolean Do_Dispatch()
{
    try
    {   //解析用户请求参数
        Stringsbean ClassName = XMLAnalyze.GetSBeanClassName(m_szClas
            sName, m_XMLFile);
        if (sbeanClassName != null && !sbeanClassName.equals(""))
        {
            //根据类名创建 sbean 对象
            Class theSBeanClass = Class.forName(sbeanClassName);

            HiEnd_SBaseBean theSBean = (HiEnd_SBaseBean) theSBeanClass.
              newInstance();
```

```
                Method method = theSBeanClass.getMethod("DP_"+m_szMethod,
null);
                if (theSBean != null && method != null)
                {
                    //设置参数
                    theSBean._SetBaseInfo_(this);
                    //执行方法
                    method.invoke( (Object) theSBean, null);
                    return true;
                }
                return false;
            }

            return false;
        }
        catch (Exception e)
        {
            System.err.println("Do_Dispatch Exception: " +
                                e.toString());//错误提示信息
        }
        return false;
    }
```

由 Do_Dispatch()方法来实现根据用户请求实例化对应功能模块的 Sbean 。当功能执行完毕后的将操作结构以 JSP 反馈给用户的核心代码如下：

```
RequestDispatcher rd = m_ControlCenter.m_Request.getRequestDispatcher("cardt
ype/cardTypeFrm.htm");//获取用户请求
            rd.include(m_ControlCenter.m_Request, m_ControlCenter.m_Respons
e);//生成用户界面
```

4.3 模型(Model)实现

在设计部分我们提出了模型(Model)的三层的设计，即 Sbean 处理用户参数，Ebean 完成逻辑运算，Dbean 完成数据库操作。而在数据层，考虑到数据操作的效率我们封装了数据库连接池。

如图 4.4 所示：

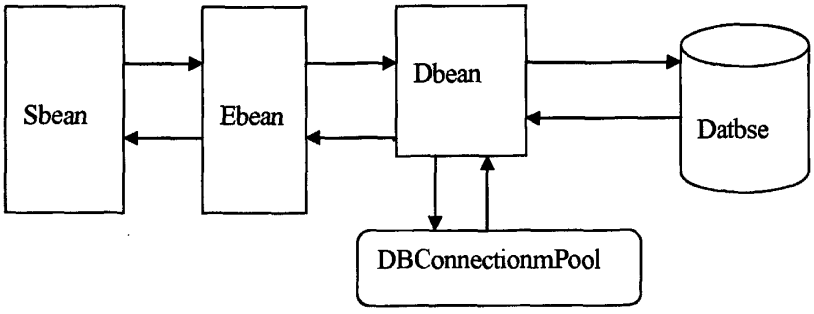

图 4.4 Model 三层结构

下面以制卡申请为例来说明其三层的实现。

(1) SBean 层是表现层和内部业务层的桥梁。它接收用户提交的参数，并对数据进行初步的转换和验证，为业务层提供合法的输入数据。SBean 层的存在更有效地分开了业务逻辑和表现层，减少了业务逻辑层对外部的依赖。制卡申请 SBean 类实现如下：

Sbean：telecomcard.applycard.sbeans 包

```
public class SB_ApplyCard
    extends HiEnd_SBaseBean
{
    public SB_ApplyCard()
    {
    }
    ……
}
```

制卡申请 SBean 类实例化，接收用户提交的参数，并对数据进行初步的转换和验证，为业务层提供合法的输入数据。

做卡申请管理主界面实现核心代码如下：

```
    public void DP_ApplyCardMain()
{
    try
    {
        //servlet 导航到制作卡申请主界面
            RequestDispatcher rd = m_ControlCenter.m_Request.
```

```
                    getRequestDispatcher(
                    "applycard/index.htm");
                rd.include(m_ControlCenter.m_Request,
m_ControlCenter.m_Response);
            }
            ……
        }
```

制作卡申请主界面如图 4.5：

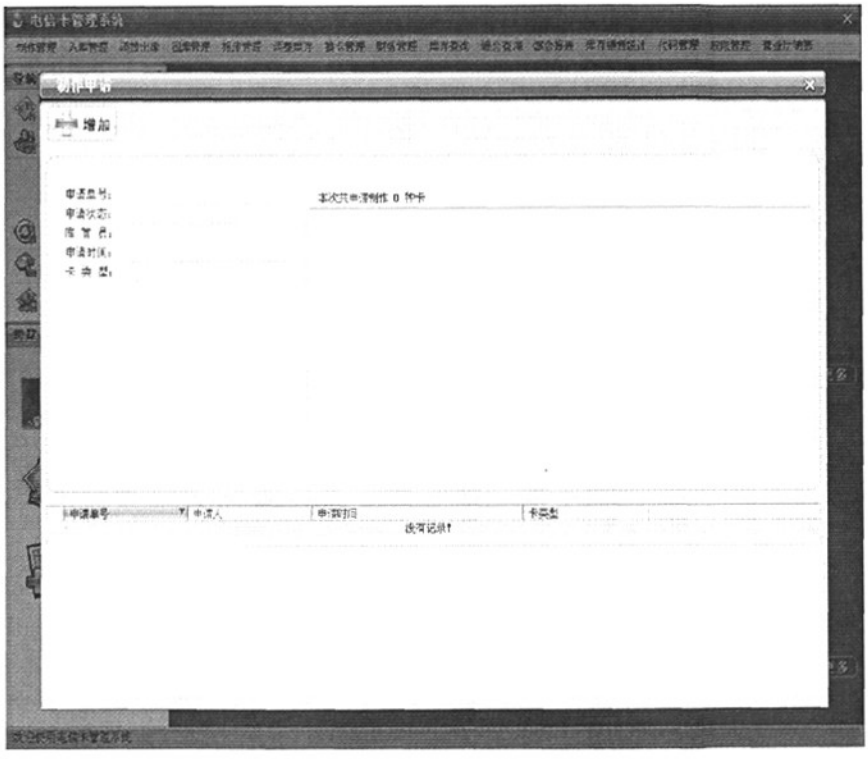

图 4.5 制作卡申请主界面

添加制作申请单的参数解析实现核心代码如下：

```
    public void DP_SubmitApply()
{
        HttpSession theSession =
m_ControlCenter.m_Request.getSession(false);
        try
        {
```

```
                    //获取参数
                    if (m_AnalyzeHttpParams.IsExist("cardType"))
                    {
                       //获取申请单
                m_ControlCenter.m_Response.setContentType(CONTENT_TYPE_H
                TML);
                        ApplyCardData m_ApplyCardData = (ApplyCardData)
        theSession.
                            getAttribute("sess_ApplyBill");
                      //获取卡类型
                        m_ApplyCardData.m_strCardType = m_AnalyzeHttpParams.
                            GetString(
                            "cardType");
                       //获取卡信息
                        ArrayList m_BillDetail = (ArrayList) theSession.
                            getAttribute(
                            "sess_ApplyBillDetail");
                        boolean bOk = false;
                         //调用 Ebeans
                }
                catch (Exception e)
                {
                  //异常处理
                 }
                }
           /*其他相关的功能比如说（修改、删除、查询申请单等）结构同上类似
*/
            .....
        }
```

制卡申请 SBean 获取制作卡信息参数确认后提交，调用 Ebeans 业务逻辑处理。提交界面如图 4.6：

图 4.6 提交界面

SBean 的框架部分在 HiEndSoft.webapp.sbeans 包下面。HiEnd_SBaseBean 所有 SBean 的基类。HiEnd_SBaseBean 封装了请求中会话信息，可以籍此判断用户是否合法用户，并根据此信息进行回应。如果是一个已登录的用户发出的请求，HiEnd SBaseBean 还包含用户的如 Id、登录名、用户名等信息。HiEndSoft.webapp.sbeans 包的其它类都是 HiEnd SBaseBean 的工具，为 HiEnd SBaseBean，最终为应用的 SBean 提供必要的支持。

HiEnd AnalyzeHttpParams 主要是提供参数的转换和检验。在 Web 环境下，服务器接收到的请求是基于 http 协议的，并没有针对特定的服务器端，服务器端可以是 Java 环境也可以是.net 环境。因此，基于 Java 的平台接收请求后必须将参数转换成应用程序能处理的 Java 类型。HiEndse SBaseBean 里包含一个

HiEnd AnalvzeHttnParams 的实例。

(2)业务对象的设计和实现对 EBean 乃至整个项目都很重要，EBean 的行为大部分都是对数据对象的操作。EBean 层所有的类部署在子模块下的 ebeans 包内，如电信卡系统的卡制作管理模块的 EBean 放在 telecomcard.makecard.ebeans 包里。EBean 不仅要与 SBean 和 DBean 通讯，EBean 之间也可以相互通讯以完成一个业务处理，但 SBean 之间不允许相互调用。

一个典型的 ebeans 包应该包括业务对象、EBean 两种类。与 SBean 相同，创建一个具体的 EBean 首先要继承一个基类。例如，卡制作模块有一个添加制作单明细的业务处理 EB_ApplyCard，其代码为:

Ebean：telecomcard.applycard.ebeans 包

```
public class EB_ApplyCard
    extends HiEnd_EBaseBean
{
    public EB_ApplyCard()
    {
    }
}
```

EB_ApplyCard 实例化，根据 Sbean 参数处理业务逻辑。

```
public boolean NewApply(int A_ApplyCardNo, ArrayList A_BillDetail)
{
    try
    {
        //调用 Dbean 执行数据库添加操作
        DB_ApplyCard applyCard = new DB_ApplyCard();
        bOk = applyCard.NewApply(A_ApplyCardNo, A_BillDetail);
    }
    catch (Exception e)
    {
        //显示错误的字符串
    }
}
```

EB_ApplyCard 从 SBean 处接收 A_ApplyCardN(用户申请单 Id), A_BillDetail (制作单明细)二个参数。EB_ApplyCard 首先调用 DBean 添加制作单，如果成功，

更新业务对象并将其存入数据库。如果中间任何一步操作出现异常，将回滚事务，恢复原来的状态。

（3） DBean 层是为 EBean 层服务的，为其提供持久性服务。DBean 层使 EBean 处理业务时不必关心数据如何从数据库获得以及如何存储到数据库。不同的项目使用的数据库可能不一样，用 DBean 分离数据与业务处理可以让应用方便地选择数据库，甚至项目进行到中途时更换数据库，只需修改相应的 DBean 层就能达到目的。

实例化 DB_ApplyCard 核心代码如下：

Dbean：telecomcard.applycard.dbeans 包

```
  public class DB_ApplyCard   extends HiEnd_DBaseBean
{
    public DB_ApplyCard()
    {
    }
    ……
}
```

以下核心代码实现新建做卡申请单的数据库操作：首先获取数据库连接，创建事务，执行插入入心得申请单，如果出错回滚。

```
    public boolean NewApply(ApplyCardData A_ApplyCard, ArrayList
A_BillDetail)
    {
        HiEnd_DataBase theDataBase = null;
        try
        {
            //获取数据库链接
            theDataBase = m_DBManager.GetConnection();
            theDataBase.StartTransaction();
            //创建 Statement
            Statement stmt=null;
            String sql="insert into ……";
            //执行插入操作
            int i= stmt.executeUpdate(sql);
        }
```

```
        catch (Exception e)
        {
            //显示错误的字符串
         }
        finally
        {
            //释放连接
            if (theDataBase != null)
            {
                theDataBase.Release();
                theDataBase = null;
            }
        }
    }
    /*其他操作*/
}
```

申请单插入成功后确认界面如图 4.7：

图 4.7 插入成功后确认界面

HiEnd_DBaseBean:是所有 DBean 的基类，它有一个成员变量 m_DBManager 是全局唯一的实例，DBean 通过这个全局数据库管理器来获取 HiEnd_DataBase 的实例，并通过这个实例来完成相应的操作。

4.4 Jsp 界面(View)实现

采用 MVC 设计模式，其中视图(View)是由 Jsp 来实现的。作为用户界面要求友好美观，因而在界面的外观设计请美工完成。

作为程序设计人员实现以下两方面的工作：

(1) 将模型(Model)的返回结果，嵌入到页面中，反馈给用户。嵌入实现：

```
<%
   String realTotal=(String)request.getAttribute("realTotal");
   String   ok=((String)request.getAttribute("ok")).trim();
   //处理、显示
  ..............
 %>
```

(2) 在 Jsp 页面中编写 javaScript 脚本，完成用户的输入合法性检查，在客户端而不是在服务器端完成用户输入的检验，提高系统的效率。JavaScript 嵌入实现：

```
<script language="JavaScript" type="text/JavaScript">
function Validate()
{
   if(用户输入合法)
   {
      //提交服务器
      submit()
    }
    else //否则提示用户，重新输入
    {
      ..........
    }
    //其他 javaScript 方法
    ..........
</script>
```

4.5 系统工作流程

在本节中，我们按照系统工作流程来描述系统结构的实现。

(1)发起请求

用户用浏览器载入初始登陆 HTML 页面，该页面是由 Web 服务器提供的。WebServer 以某种方式来进行 Web 文件服务。这个页面是整个 Web 应用的入口点。

(2)控制权传给 Controller

Controller 的作用主要是由 HiEnd_ControllCenter 完成的。所有的 url 链接都将转给 HiEnd_ControllCenter，控制器分析用户的登陆权限，导航到相应的主页面。再根据 url 中的模块名和功能名来负责调用处理用户请求的 Sbean 类，然后负责创建对用户的响应，在我们的应用结构中，响应将被传递到 JSP。

(3)Sbean 类执行请求

Servlet 执行类实例化的 Sbean 方法，例如：

```
String sbeanClassName = XMLAnalyze.GetSBeanClassName(m_szClassName,
m_XMLFile);
        if (sbeanClassName != null && !sbeanClassName.equals(""))
        {
            //根据类名创建 sbean 对象
            Class theSBeanClass = Class.forName(sbeanClassName);

            HiEnd_SBaseBean theSBean = (HiEnd_SBaseBean)
theSBeanClass.newInstance();
            Method method = theSBeanClass.getMethod("DP_"+m_szMethod,
null);

            if (theSBean != null && method != null)
            {
                //设置参数
                theSBean._SetBaseInfo_(this);
                //执行方法
                method.invoke( (Object) theSBean, null);
            ..............
            ..............
        }
```

(4)Ebeans 对用户的输入信息逻辑处理：

```
package telecomcard.cardtype.ebeans;
import HiEndSoft.webapp.ebeans.*;
```

```
import telecomcard.cardtype.dbeans.*;
import java.util.*;
//实例化 Dbean
telecomcard.cardtype.dbeans.DB_CardType db=new
telecomcard.cardtype.dbeans. DB_CardType();
//调用 Dbean 完成数据库的插入操作。
return db.addCardType(reasonName,remark,id);
```

(5)Dbeans 完成数据库操作：

```
package telecomcard.cardtype.dbeans;
import HiEndSoft.webapp.dbeans.*;
import HiEndSoft.webapp.ebeans.*;
import java.sql.*;
//获取数据库链接
HiEnd_DataBase theDataBase = m_DBManager.GetConnection();
String sql="insert into pt_CardType(CardTypeID,CardName,Remark)
values('"+cardtypeId+"','"+Name+"','"+Remark+"')";
Statement stmt =theDataBase.CreateStatement(java.sql.ResultSet.TYPE_SC
ROLL_INSENSITIVE,java.sql.ResultSet.CONCUR_READ_ONLY);
//执行插入操作
int i=stmt.executeUpdate(sql);
//释放连接
theDataBase.Release();
```

(6)控制权传给 JSP

我们已经提到，MVC 模式的重点就在于它隔离了表现层与业务逻辑层。Servlet 负责处理请求，它会去调用事务处理类，从 Sbeans 处理参数到 Ebean 业务逻辑处理，到 Dbean 数据库操作，即后端系统(Mode)包含了业务规则。而对用户的响应(View)是由 JSP 创建出来的。

Servlet 将 request 和 response 对象传递给 JSP 时遵循如下的语法：

```
RequestDispatcher rd = m_ControlCenter.m_Request.getRequestDispatcher(
                    "cardtype/cardTypeEdit.jsp");
                rd.include(m_ControlCenter.m_Request,
```

m_ControlCenter.m_Response); Inlude()方法允许 Servlet 将响应的处理传给第三方。它的参数 request 和 response 必须就是调用 Servlet 的 service 方法时传递的对

象，它使用 getRequestDispatcher 对象来得到 requestDispatcher，从而确定了 JSP 路径名被解释成相对于当前 Web 应用程序上下文环境的根路径。

(7)访问 Request 封装的数据

JSP 负责创建对用户的响应，它可以访问 action 类传递的数据，这些在 HTML 中用如下语法来完成：

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="HiEndSoft.webapp.ebeans.*"%>
<%
  HiEnd_ResultSet rs = (HiEnd_ResultSet)request.getAttribute("ResultSet");
%>
```

在 JSP 文件中，<%=和%>之间所有的内容被放入 JSP 引擎中处理，结果被作为输出送给 JSP 文件。同任何一种脚本语言一样，JSP 代码可以在 HTML 页面中任意嵌入。JSP 的语法也非常简单，这意味着 JSP 可以由 Web 页面设计人员来维护而不是应用开发人员。开发人员负责的是 Servlet 和数据访问类(Model)。在 JSP 中的任何改变不会影响到 Servlet，反之亦然。

(8)响应

JSP 在接到请求时被动态翻译成 JavaServlet，并在 Web 服务器中缓存。后续 JSP 请求的响应将明显加速。用户收到的最终响应都是 HTML 页面(以.jsp 为扩展名)，这些页面包含了动态产生的内容。

4.6 小结

在本文示例的应用结构中，客户端浏览器发起的请求直接到了 Servlet，然后用三层的数据处理类(Model)来处理请求，从后端数据库系统获取数据到 HiEnd_ResultSet 类把它放到 Response 中，然后调用了 JSP 来处理这个响应。Servlet 是初始请求到产生响应的总体控制者。

Model 决定产生给用户响应的内容，JSP 只应当包含如何格式化表现层的逻辑。这种隔离的优点在于它创建了在应用中可重用的、可移植的、平台独立的组件，这些组件可以作为将来更大的应用的一部分,实现组件的复用，提高开发效率，隔离 Servlet 开发与 JSP 显示为应用开发人员和 Web 页面设计人员彼此独立的工作带来了极大的方便。同样，这种方法也完美的符合了本文所介绍的 MVC 设计模型。

第五章 总结

通过对MVC模式以及其实现框架Struts的研究应用，本文作者深切的感受到模式应用对于系统开发的重要意义所在：一个成功的软件不仅需要好的技术支持，更重要的是必须有一个成功的架构、模式的指引。系统开发开发人员在学习基础技术的同时，必须重视与技术相关的设计模式学习，以及软件工程和计算机科学各方面的知识，高屋建瓴，从而开发出成熟、健壮的系统。

本系统是基于可视化面向对象建模技术研发，遵循网络化发展需要，采用先进的MVC设计模式，运用J2EE技术进行设计和开发的。

目前国内的许多 Web 应用，如果需求发生了变化，则网站的结构、页面结构、页面流程、功能和服务都发生了变化，造成了整个应用软件，从表现逻辑部分到应用逻辑部分都必须重新设计开发。

而本系统采用 MVC 设计模式进行设计、开发，这使得整个系统的结构清晰，容易理解：当新的功能增加时，很容易找到要修改或扩展代码的入口；代码之间的耦合度小，编码容易分工，容易进行模块化划分，重用性好；表现逻辑和业务逻辑之间的交互通过控制组件集中完成，大大提高了可维护性；采用事件机制，使得表现逻辑和业务逻辑能够很容易地挂接，大大提高了可扩展性；应用框架的重用性大大提高；具有不同技能的人员，开发分工容易。这样，在系统的运行维护阶段，就很容易找到入口点；增加新的功能时，对新的功能进行编码，经过配置后，就能够很容易挂接到应用框架中。

通过近一年的论文研究工作，课题研究已经完成以下几个方面内容：

（1）采用了可视化面向对象建模技术

（2）结合 J2EE 技术，运用了先进的 MVC 设计模式

（3）详细设计和部分编码

（4）测试、并试运行

参 考 文 献

[1] Walker. An Initial Assessment of Aspect-oriented Programming. Software Engineering, 2008.5(16):120-130

[2] Alexander. The Real Costs of Aspect-oriented Programming. Software, 2008.20(6):92-93.

[3] Arthur John, Azadegan Shiva. Spring Framework for Rapid Open Source J2EE Web Application Development: A Case Study. Proceedings — Sixth Int. Con# on Softw Eng., Artificial Intelligence, Netw. and Parallel/Distributed Computing and First ACIS Int. Piscataway: Institute of Electrical and Electronics Engineers Computer Society, 2007. 90-95

[4] Han Minmin, Hofmeister Christine, Nord Robert L. Reconstructing Software Architecture for J2EE Web Applications. Reverse Engineering-Working Conference Proceedings, 2007. 67-77

[5] Rod Johnson, Juergen Hoeller etc. Spring-J2EE Application Framework Reference Documentation.Spring:http://static.springframework.org/spring/docs/2.O.x/spring-eference.pdf, 2008.

[6] Rod Johnson, Juergen Hoeller, Alef Arendsen, Thomas Risberg, Colin Sampaleanu. Professional Java Development with the Spring Framework. John Wiley&Sons, Inc. Publishin}.2008: 4--30

[7] Greiner, Saso; Tutek, Simon; Brest, Janez et al. Quick Adaptation of Web-Based Information Systems With Aspect-Oriented Features. Proceedings of the International Conference on Information Technology Interfaces, TTI,2004. Croatia: University of Zagreb, 2006:145-150.

[8] Anil Hemrajani. Agile Java Development with Spring Hibernate and Eclipse. United States: Sams,2006

[9] Boehm, B. Engineering context (for software architecture), invited talk, In: Garlan D., ed. Proceedings of the 1st International Workshop on Architecture for Software Systems Seattle. New York: ACM Press, 2000. 1~8.

[10] Perry, D.E., Wolf, A.L. Foundations for the study of software architecture. ACM SIGSOFT Software Engineer Notes, 1992,17(4): 40~50.

[11] Kruchten, P.B. The 4+1 view model of architecture. IEEE Software, 1995,12(6):42~50.

[12] Clements, P.C., Weiderman, N. Report on the 2nd international workshop on development and evolution of software architectures for product families. Technique Report, CMU/SEI-98-SR-003, Carnegie Mellon University, 1998.

[13] Yang Fu-qing. Software reuse and related technology. Computer Science, 1999,26(5):1~4 (in Chinese).

[14] Garlan, D., Shaw, M. An introduction to software architecture. Technique Report, CMU/SEI-94-TR-21, Carnegie Mellon University, 1994.

[15] The Boeing Company-Defense and Space Group. STARS conceptual framework for

[16] reuse processes, lockheed martin tactical defense system. STARS Program Technical Report, 1994.

[17] 王晶.大连邮政国际物流信息系统的分析与设计:[硕士学位论文 J.大连:东北财经大学，2008.(12):12-35

[18] 王海强.论电子商务与现代物流信息化的关系:〔硕士学位论文 1.北京:对外经济贸易大学，2007(4):5~6

[19] 刘仙泽，邹辉霞.物流信息化实施模式探讨.物流技术，2007.6:12-14

[20] 梁宝兰.分布式信息系统的持久对象共享研究:[硕士学位论文 1.重庆:重庆大学 2006.4:4-14.

[21] 李刚.struts 在校园管理系统中的设计和实现:[硕士学位论文 1.济南:山东大学，2006.4:1-15

[22] [vJ 保铁汉.基于 B/S 结构的企业网络培训系统的设计与实现:【硕士学位论文}.苏州:苏州大学，2006.11:8-13

[23] 田禾.轻量级 JZEEWeb 应用的设计与开发研究:[硕士学位论文].成都:电子科技大学，2006.2:8-9

[24] 骆华俊,唐稚松,郑建丹.可视化体系结构描述语言 XYZ/ADL.软件学报,2000,11(8):1024~1029.

[25] 张家晨,冯铁,陈伟,等.基于主动连接的软件体系结构及其描述方法.软件学报,2000,11(8):1047~1052.

[26] 于卫,杨卫海,蔡希尧.软件体系结构的描述方法研究.计算机研究与发展,2000,37(10):1185~1191.

[27] 云晓春,方滨兴.基于构件设计的正确性验证.小型微型计算机系统,1999,20(5):330~334.

[28] 周莹新,艾波.软件体系结构建模研究.软件学报,1998,9(11):866~872.

[29] 陶伟.以体系结构为中心软件产品线开发[博士学位论文].北京:北京航空航天大学,1999.

[30] 周莹新.电信软件体系结构的研究[博士学位论文].北京:北京邮电大学,1997.

[31] 张广泉.关于软件形式化方法[J].重庆师范学院学报(自然科学版),2002,19(2):1 4.

[32] 张广泉.软件体系结构与 XYZ E [D].中国科学院博士后研究工作报告[R]北京:中科院软件研究所,2000 05.

[33] 赵会群,王国仁,高远.软件体系结构抽象模型[J].计算机学报,2002,25(7):730 736.

[34] 何新贵.软件能力成熟度模型[M].北京:清华大学出版社,2000.[9]刘超,张莉.可视化面向对象建模技术———标准建模语言 UML 教程[M]. 北京: 北京航空航天大学出版社,2000.

[35] 邓勇,丁峰,沈钧毅.基于 UML 的软件体系结构建模方法的研究[J].小型微型计算机系统,2001,22(10):1206—1209.

[36] Li Gong 著 . 王运凯，石磊等译 . Java2 平台安全技术——结构、API 设计和实现 . 机械工业出版社 . 2000 .

[37] Bruce Eckel 著 . 京京工作室译 . Java 编程思想 . 机械工业出版社 . 1999 .

[38] Deepak Alur, John Crupi, Dan Malks 著 . 牛志奇等译 . J2EE 核心模式 . 机械工业出版 . 2002.

[39] 戴翔宇 . 基于 MVC 模式的 Struts 框架的研究与应用 . 武汉理工大学硕士学位论文 . 2003 .

[40] 王仕超 . 基于 JAVA 的 MVC 模型框架研究 . 南京理工大学硕士学位论文 . 2002 .

[41] 熊丽 . 基于 J2EE 和 MVC 的 Web 应用开发方法的探讨 . 武汉理工大学硕士学位论文 . 2002.

致　谢

我的论文自始至终都是在我的导师沈磊老师的关心指导下完成。在上研究生期间，沈老师给了我许多的关心和指导，让我终生受益。从课题的选定、资料的收集、论文的撰写和修改等各个环节，他都倾注了大量的精力，使我最终顺利地完成了学位论文的撰写。沈老师的广博学识、严谨学风、认真态度、创新精神和对科学事业的不懈追求，是我永远学习的楷模。在此再一次向沈老师表示衷心的感谢。

最后，再次向关心和帮助我的各位老师、同学和朋友们致谢!

攻读学位期间发表主要论文

[1] 基于机器学习的智能防火墙设计．山东理工大学学报，2008，22(3):33-36 第一作者