摘要

人力资源管理系统是现代企业管理的核心内容。随着计算机信息技术的高速发展，电子商务模式的空前盛行，企业之间的竞争也从有形的经济市场转向了网络。开发以计算机技术、网络技术、信息技术支持的现代人力资源管理系统，既能提高企业人力资源管理的技术含量和企业的办事效率，也能使人力资源管理能够进入现代化、决策科学化的进程。现代人力资源管理系统采用了B/S模式，可以避免C/S模式的重用性差、维护难度高的缺点和不足。结合实际项目的功能需求，从系统分析、系统总体设计、系统详细设计、系统实现等方面进行了详细的论述。

系统开发主要是MVC设计思想的应用，主要采用Jsp+Servlet+JavaBean的开发方式。Jsp对应MVC设计思想的视图（View），主要是负责接收/响应客服端请求，以及动态生成Web现实页面。Servlet是对应MVC设计思想的控制（Controller），主要负责整个系统的核心业务控制。JavaBean对应的是MVC设计思想的模型（Model），主要负责整个系统的数据和业务规则。系统的实现过程中运用了一些开源项目，如Ajax技术、JfreeChart报表、fileupload技术。

本文运用了现代人力资源管理理论，结合企业的实际情况，比较国内外人力资源管理系统的现状，设计了一个基于Internet/ Intranet的人力资源管理系统。本文研究人力资源管理系统的主要内容有：招聘管理、人事管理、考勤管理、培训管理、薪资管理、系统设置。系统的开发目的是人力资源管理的业务逻辑实现高效化、智能化，从而能帮助企业的人力资源管理人员进行人力资源管理和数据分析。

关键词：人力资源管理；B/S模式；MVC模型：开源项目.

Abstract

Human resource management system is the core content of modern enterprise management. With the rapid development of the computer information technology and unprecedented prevalence of electronic commerce mode，the competition between enterprises is turning from visible economic markets to the network. Developing the human resource management system supported by computer technology，network technology and information technology can not only improve the skill of human resource management and the efficiency of the enterprises but also make human resource management modern and decision sciencefic,Modern human resource management uses B/S mode to avoid C/S modes short coming of difficult in maintdning and reusing.According to the functional requirements of the actual project,this article specificly state the analysis of system,the general desigin of the system,the detail design of system and the practice of the system.

The development of the system is the practice of MVC design ideas, maing using the Jsp+Servlet+JavaBean form of development.Jsp is the practice of MVC design ideas'view,in charge of receiving/responding the request of the customer.Servlet mainly responsible for the core business control of the whole system is the practice of the vontroller of MVC design idea to take charge of the statistics and rules of the whole system. In the practice of the system, somr open-source projrcts,such as the Ajax technique,JfreChart statements,fileupload technology,has been used.

Using the modern human resource management theropy and analysising the actual situation, comparing the current situation of human resource management system, a huaman resource management system basied on the Internet/Intranet has been designed. The main contents of the huaman resource management system includes recruitment management, personnel management,attendance management training management, salary management and system configuation.The development of the system aims at making the management business logic more efficient and intelligent to help people manage the human resource and analysis the statistics.

Keywords: human resource management; B/S mode; Open-source projects; MVC mode.

目录

第 1 章绪论

1.1 课题研究的背景

21世纪是一个日新月异的信息时代，随着电脑与网络技术的日益发达，电子商务空前的发展，企业之间的竞争已经从有形的市场经济转向了无形的网络领域。因此企业管理也进入了高效的信息化的时代，即人力资源管理系统也就应运而生，所谓人力资源管理系统，指人力资源管理电子信息化，是企业基于高速度、大容量的硬件和先进的IT软件的人力资源管理模式。通俗地说，就是人力资源管理信息化或自动化。

在一个现代化的公司中的企业管理主要涉及到招聘，人事、薪资、考勤、培训几大部分，本次系统开发主要是针对以上几大模块的工作逻辑来设计和实现人力资源管理系统。

1.2 国内外发展现状

现在，中国国内的人力资源管理系统的现况是[1]：中国的软件系统大多是源自信息系统，从部门的业务需求方面出发设计。管理信息系统的设计是为了服务于企业内部大多数业务操作员，将业务操作人员的重复性劳动进行初步自动化，即从管理理论抽象出理想化的业务管理模式，在基于该业务模式的基础上实现低层次的数据处理或业务流程电子化。管理信息系统的设计，是根据中小型企业业务单元的需求来编写的，一般无法满足多体制、多元化、多重组织结构的大型企业数据处理需求。

在与国外同类应用系统及解决方案相比较，目前中国的人力资源管理软件还有一些不足：第一，大部分是由管理信息系统演变而来，从单一的人力资源管理或人事行政管理的业务需求角度出发设计，如人事管理、考勤管理，或薪资计算与发放管理等，服务对象是某一具体业务的自动化操作需求；第二，目前国内的人力资源管理软件虽然已将模块功能扩展至企业人力资源管理或人才资本管理相关的整个业务领域，但系统在完整性、前沿性和集成性方面仍有欠缺。国外人力资源管理系统相对于国内人力资源管理系统来说，优势主要体现为：具有雄厚的经济实力，在技术力量的培养、研发、市场推广等方面大力投入；具有一定实力的包括硬件厂商、数据库公司、咨询公司在内的合作伙伴，形成很强的实力联手格局。国外人力资源管理系统伴随着管理理论的发展，其设计思路蕴涵了先进的管理理念和先进的开发技术；国外人力资源管理系统起步较早，完整性和成熟度高，能开发出了适用于不同行业的解决方案。

1.3 课题研究的目的和意义

人力资源管理系统（HRMS），包括人事日常事务、薪资、招聘、培训、考核，同时人力资源的管理也指组织或社会团体运用系统学理论方法，对企业的人力资源管理各个方面进行分析、规划、实施、调整，提高企业人力资源管理水平，使人力资源更有效的服务于组织或团体目标。人力资源管理系统就不仅可以完成日常业务需求，而且可以准确及时地搜索各种人力资源信息以方便管理者进行决策。

本系统是一个建立在成熟的Internet / Intranet[2]之上的人力资源管理系统。在系统需求分析的设计过程中，我们通过网络了解人力资源管理系统的功能组成部分，并通过对公司的咨询进行分析，最后通过小组会议讨论的方法获得需求分析，根据用户需求设计开发思路，采用图形来建立业务逻辑，最终确定系统功能模块。根据需求分析过程获取具体实体对象，从而设计系统类图，确定类之间的关系，对系统进行详细设计并实现。

该系统可以对企业员工各种信息和企业的各种部门信息进行统一管理，公司相应权限的管理人员可以登录本系统，进行相应的企业人力管理。使人力资源管理人员从繁杂、重复的劳动中脱离出来，集中时间、精力进行人力资源的整体规划与决策，提高企业的市场竞争力。系统应用了成熟的Internet / Intranet技术到人力资源管理系统中，使企业员工在全国各地都可以随时了解企业相关信息。在人员招聘方面，企业通过Internet对外发布招聘信息，应聘人员可以根据自己的特点填报相关空缺职位；在内部管理方面，也能方便员工交流；在业绩考勤管理方面，能对公司员工的日常考勤、加班、出差、请假考勤进行相关的记录统计，得到相关的负责人的批准方能生效。

第 2 章相关知识的介绍

在系统的开发过程中，运用面向对象[3]的开发语言，系统采取B／S结构，使用J2EE开发框架,主要是MVC框架，Ajax技术（jQuery），JFreeChart报表.权限控制时采用的是角色对应权限，配置文件配置URL进行action过滤来实现。数据库采用SQL Server 2005.

2.1 开发工具简介

本系统的开发环境是 jdk6.0+Eclipse3.5+Tomcat6.0，使用的语言是 java 语言。

JDK(Java Development Kit)是 Sun Microsystems 公司为 Java 开发人员设计的的产品。从 Java 诞生以来，JDK 已经成为使用最广泛 Java SDK。JDK 是整个 Java 的核心内容，包括了 Java 运行环境，Java 工具和 Java 基础的类库三部分。JDK 是学好 Java 的前提。而专门运行在 x86 平台的 Jrocket 在服务端运行效率也要比 Sun JDK 好很多。从 SUN 的 JDK5.0 开始,提供了泛型等非常实用的功能，其版本也不断更新，运行效率得到了非常大的提高。现在最新版本是 jdk6.0。

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。Eclipse 它只是一个框架和一组服务，用于通过插件组件构建开发环境，关键的是 Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。 Eclipse 最先是由 IBM 公司开发的替代商业软件 Visual Age for Java 的下一代 IDE 开发环境，2001 年 11 月贡献给开源社区，现在由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。 2003 年，Eclipse 3.0 选择 OSGi 服务平台规范为运行时架构。 2007 年 6 月，稳定版 3.3 发布。2008 年 6 月发布代号为 Ganymede 的 3.4 版。2009 年 7 月发布代号为 GALILEO 的 3.5 版。

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发研究而成。因为有了 Sun 的参与和支持的原因，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现，Tomcat 5 支持最新的 Servlet 2.4 和 JSP 2.0 规范。因为 Tomcat 技术先进、性能稳定，而且免费，因而深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的 Web 应用服务器。目前最新版本是 6.0。

2.2 MVC 框架简介

MVC 架构是"Model-View-Controller"的缩写中文翻译为"模型-视图-控制"[4].MVC 应

用程序是由这三个部分组成。Event(事件)的变化导致 Controller 改变 Model 或 View，或者同时改变两者。当 Controller 改变了 Models 的数据或者属性，所有依赖的 View 都会自动更新。对应地，当 Controller 改变了 View，View 会从潜在的 Model 中获取数据来刷新自己的变化。MVC 架构最早研发者是 smalltalk 语言研究团，主要应用于用户交互应用程序中。smalltalk 语言和 java 语言有很多相似性,都是面向对象语言，很自然的 SUN 在 petstore(宠物店)事例应用程序中就推荐 MVC 架构作为开发 Web 应用的架构模式。MVC 架构是一种架构，其实需要其他模式协作完成。在 J2EE 模式中，通常采用 service to worker 模式实现，而 service to worker 模式可由集中控制器模式，派遣器模式和 Page Helper 模式组成。

MVC 架构是一个复杂的架构，其实现也显得非常复杂。由于我们已经总结出了很多可靠的设计模式，多种设计模式结合在一起，使 MVC 架构的实现变得相对简单易行。Views 相当于一棵树，可以用 Composite Pattern 来实现。Views 和 Models 之间的关系可以用 Observer Pattern 体现。Controller 控制 Views 的显示，可以用 Strategy Pattern 实现。Model 通常是一个调停者，可采用 Mediator Pattern 来实现。

MVC 与 J2EE 架构的对应关系可以分析如下:View 处于 Web Tier 或者说是 Client Tier，通常是 JSP/Servlet，即页面显示部分。Controller 也处于 Web Tier，通常用 Servlet 来实现，即页面显示的逻辑部分实现。Model 处于 Middle Tier，通常用服务端的 javaBean 或者 EJB 实现，即业务逻辑部分的实现。

2.3 系统运行配置

服务器操作系统：可运行 Tomcat6.0 的 Windows 或 Linux 操作系统

客户操作系统： 支持 Fire fox 浏览器的操作系统

测试浏览器：Fire fox 浏览器

第 3 章 系统分析

系统分析主要是介绍在系统设计前的可行性分析和需求分析，为系统设计作必要的准备。可行性分析简单的对本系统的技术可行性、操作可行性、软硬件的选择的分析说明。需求分析，主要是对本系统大体要实现内容的总结，以便以后测试本系统是否达到设计标准。

3.1 可行性分析

3.1.1 技术可行性

本系统技术要求如下：

功能：对人事资料、人力资源、工资管理、考勤管理等进行综合管理。

输入/输出：输入查询条件，输出查询内容。

基本的数据流程和处理流程：先对人员信息进行录入，然后再对它们分类。可以对数据进行插入、删除、修改、查询。

用户与权限：此系统可以分为用户和管理员，用户可以设置自己的个人信息，管理员主要管理系统的各种信息。

以上系统技术要求使用 Eclipse 可以满足，它使用的是面向对象、高效率且能够实现 b/s 模式编程的 java 语言[5]开发，使用 MVC 思想把数据、视图、业务逻辑进行分开，使用 SVN 协同开发工具进行版本控制，因此，本系统的开发在技术上是可行的。

3.1.2 操作可行性

该系统在操作上很简单的，使用者完全可以没有专业的计算机知识。启动系统后进入登陆用户界面，用户用自己登陆名和密码进入系统操作页面。不同的用户拥有不同的权限，也只能浏览和操作相应的模块。用户能对自己拥有权限的功能模块进行相应的操作。例如人事主管进入系统：他就可以进入人事信息模块，对人事信息进行浏览，删除，修改，添加等操作。

3.2 需求分析

该系统分为六个模块：人事管理、考勤管理、工资管理、招聘管理、培训管理、权限管理。

招聘管理：针对公司的人才需求发布招聘职位信息,对应聘职位的简历进行录入并管理；符合要求的简历通知其进行面试，对面试人员的面试结果进行档案维护。另外该模块

还涉及到对应聘人员资料的统计分析。

人事管理：该模块是关于公司的人事信息维护，首先是对入职员工的基本信息的录入、修改、浏览等操作；其次是员工人事档案、合同档案等基本信息的管理和公司的部门管理；最后，并对公司的人事信息从工种，专业，学历等属性进行统计分析。

考勤管理：该模块主要涉及到公司员工的考勤统计。考勤主要是日常考勤，请假考勤、出差考勤、加班考勤几大内容；另外也通过各个部门的考勤进行报表分析。

工资管理：该模块主要是关于公司的薪资信息维护，相关人员能进入该模块给员工进行工资管理，员工的工资主要是工资方案计算得出，工资方案是由工资项目进行组合而成。另外还能对员工的工资信息能进行调整和数据统计。

培训管理：该模块主要是进行员工培训信息的维护，主要是对培训类型管理、培训档案管理和培训效果统计。

系统设置：该模块主要是用户的权限设计，不同的用户对应着不同的权限，也只能操作相应权限的模块，另外还有用户的创建和用户密码的修改等功能。

第 4 章 系统总体设计

系统总体设计是对系统的模块规划、系统功能结构及系统数据库的总体设计。

4.1 系统模块规划

本系统是一个典型的数据库开发应用程序，主要由招聘管理、人事管理、考勤管理、薪资管理、培训管理和系统设置六大模块组成，规划系统功能模块如下：

招聘管理模块:该模块主要是由招聘信息管理、简历管理、面试档案管理、招聘职位、统计分析组成。

人事管理模块：该模块主要是由人事基本信息管理、部门设置、人事档案、人事变动、合同管理、工种类型、职位类型、员工状态、学历资料、专业资料、统计分析组成。

考勤管理模块：该模块主要由请假管理、出差管理、加班管理、出勤管理、请假报表、出差报表、加班报表、出勤报表组成。

工资管理模块：该模块主要是由工资项目设置、工资方案、工资调整、工资发放、统计分析组成。

培训管理模块：该模块主要由培训信息管理、培训档案管理、培训类别、统计分析组成。

系统设置模块：该模块主要由用户管理、角色管理、资源权限管理、密码修改组成。

4.2 系统功能结构图

1. 主要模块结构图如 4-1：

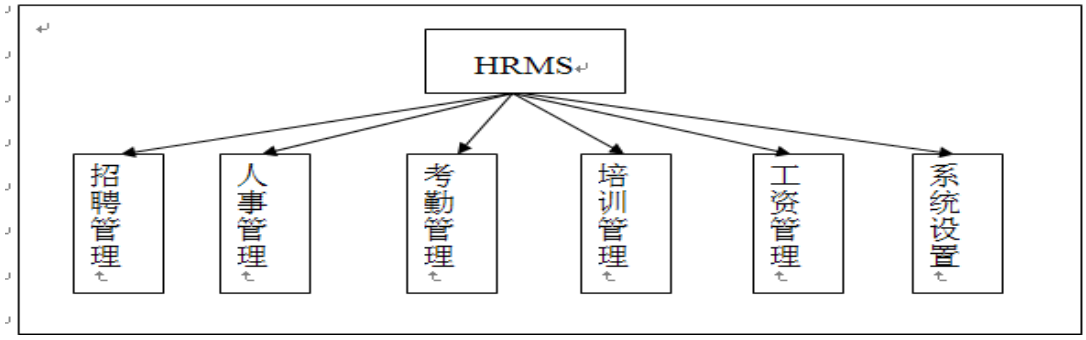

图 4-1 系统结构功能图

2. 模块结构图

1. 招聘管理子模块结构图如 4-2：

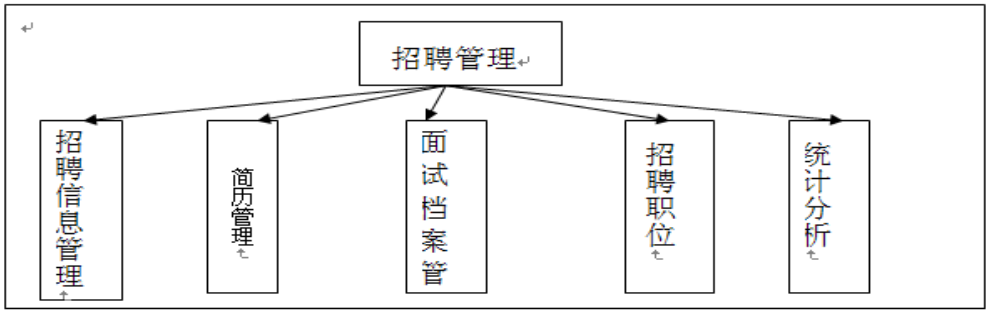

图 4-2 招聘管理子模块结构图

2. 人事管理子模块结构图 4-3：

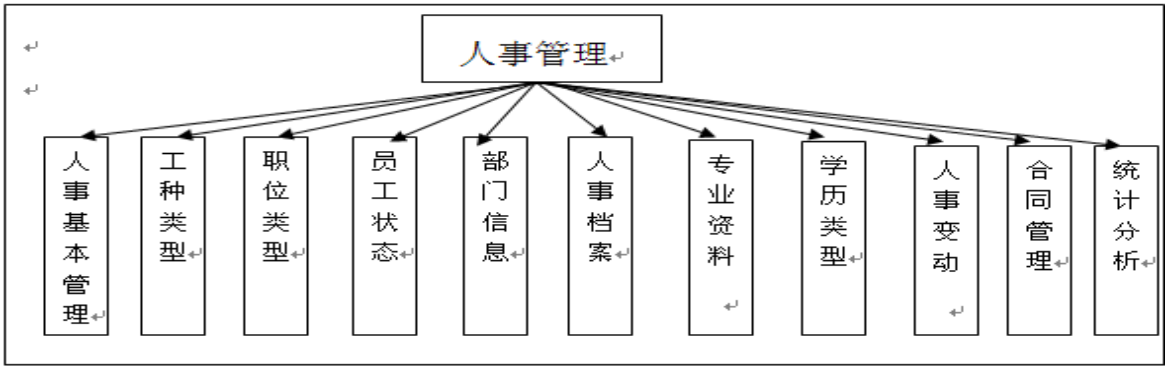

图 4-3 人事管理子模块结构图

3. 考勤管理子模块结构图 4-4：

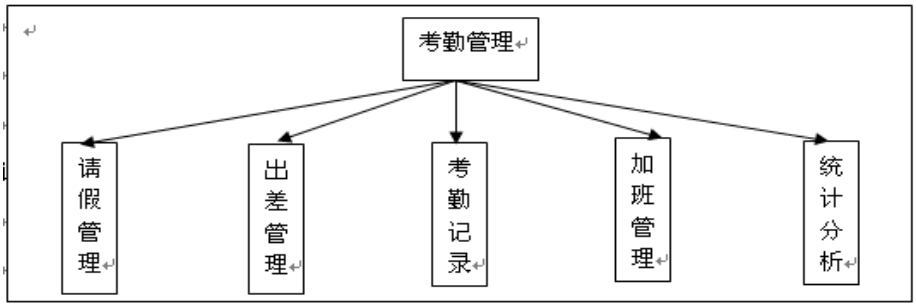

图 4-4 考勤管理

4. 培训管理子模块结构设计图 4-5：

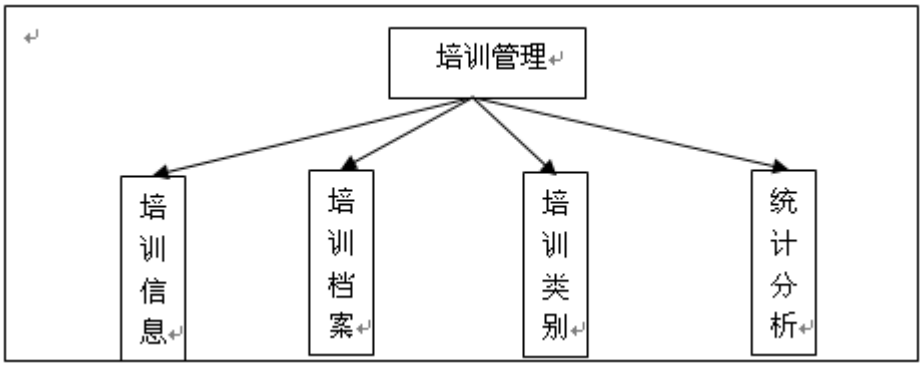

图 4-5 培训管理模块设计

5.工资管理子模块设计图 4-6:

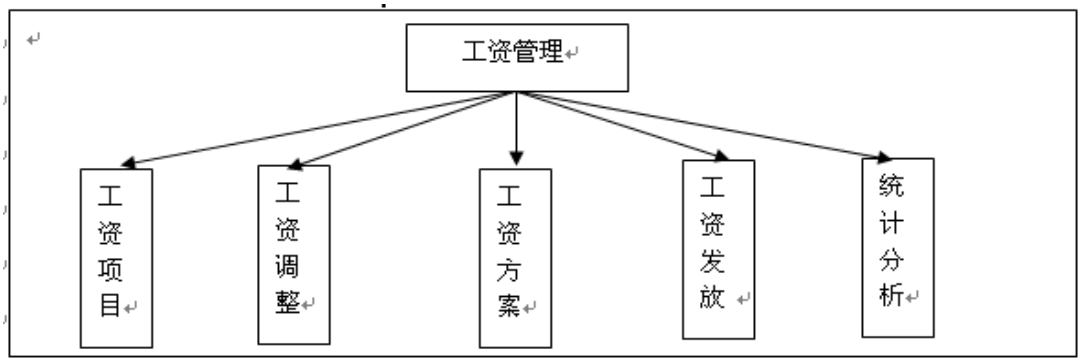

图 4-6 工资管理模块设计

6 系统设置子模块设计图 4-7：

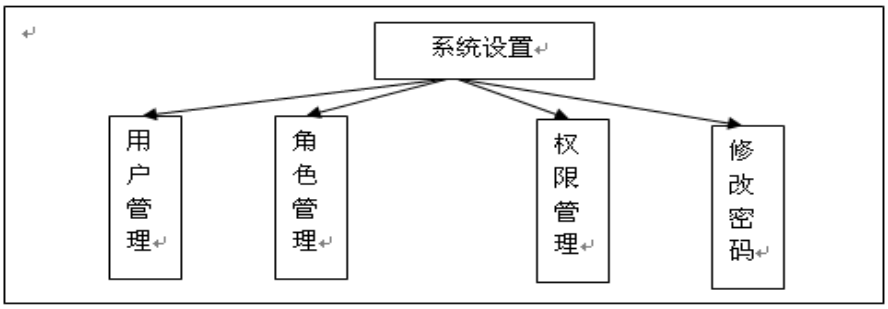

图 4-7 系统设置子模块设计

4.3 系统数据库设计

数据库设计的好坏在一个信息管理系统中地位十分重要，数据库结构设计将直接关系到对应用系统的效率，实现的效果产生影响。数据库结构设计合理可以提高数据存储的效率，保证数据的完整性[6]。

4.3.1 数据库需求分析

数据库系统应充分熟悉用户各方面的需求，对于本系统用户的需求具体体现在各种信息的提供、保存、更新和查询，这要求数据库的结构能充分满足各种信息的输入和输出，收集基本数据、数据结构及数据处理的流程，为后面的具体设计打下数据基础。

1．数据录入和处理的准确性和实时性：数据的准确输入是数据处理的前提，错误的输入会导致系统输出的异常和不正确，从而使系统的工作失去意义。数据的输入来源是手工输入。手工输入数据要考虑数据的长度和数据的属性。在系统中，数据的输入往往是大量的，因此系统要有一定的处理能力，以保证迅速的处理数据。

2．数据的一致性与完整性：由于系统的数据是共享的，在不同的部门中，信息是共享数据，所以如何保证这些数据的一致性，是系统必须解决的问题。要解决这一问题，要

有一定的人员维护数据的一致性，在数据录入处控制数据的去向，并且要求对数据库的数据完整性进行严格的约束。对于输入的数据，要为其定义完整性规则，如果不能符合完整性约束，系统应该拒绝该数据。

根据系统功能分析和需求总结，考虑到将来功能上的扩展，设计出员工基本信息表、工种类型表、职位表、部门表等十八个表。

4.3.2 数据库概念结构设计

得到上面的数据项和数据结构以后，就可以设计出能够满足用户需求的各种实体，以及它们之间的关系，为后面的逻辑结构设计打下基础。这些实体包含各种具体信息，通过相互之间的作用形成数据的流动。

本系统根据上面的设计规划出的实体有：员工基本信息表、工种类型表、职位表、部门表等十八个表。

4.3.3 数据库逻辑结构设计

数据的概念结构设计完之后，可以将上面的数据库概念结构转化为某种数据库系统所支持的实际数据模型，也就是数据库的逻辑结构。比如系统数据库中各表的详细设计如表所示。

表 4-1 员工基本信息表(hr_jbxxb)

字段名	数据类型`	描述	是否主键	外键	是否为空
yggh	Varchar（30）	员工工号	√		
ygxm	Varchar（30）	员工姓名			
ygxb	Varchar(10)	员工性别			√
csrq	Varchar(20)	出生日期			√
xl	varchar(20)	学历			√
zy	varchar(20)	专业			√
lxfs	Varchar(30)	联系方式			√
jzdz	Varchar(100)	居住地址			√
zw	Varchar(30)	职务			√
xz	Float(8)	薪资			√
rzsj	Varchar(20)	入职时间			√
ssbm	varchar（30）	所属部门			√
zzzt	Varchar(10)	在职状态			√

表 4-2 工种类型 (hr_gzlxb)

字段名	数据类型`	描述	是否主键	外键	是否为空
gzbh	Varchar（30）	工种编号	√		
gzmc	Varchar（30）	工种名称			√

表 4-3 职位类型 (hr_zwlxb)

字段名	数据类型	描述	是否主键	外键	是否为空
zwbh	Varchar（30）	职位编号	√		
zwmc	Varchar（30）	职位名称			√

表 4-4 员工状态（ hr_ygzt）

字段名	数据类型	描述	是否主键	外键	是否为空
zzztbh	Varchar（30）	在职状态编号	√		
zzztmc	Varchar（30）	在职状态名称			√

表 4-5 部门分类（hr_bmflb）

字段名	数据类型	描述	是否主键	外键	是否为空
bmbh	Varchar(30)	部门编号	√		
bmmc	Varvhar(30)	部门名称			√

表 4-6 人事变动（hr_rsbdb）

字段名	数据类型	描述	是否主键	外键	是否为空
bmbh	Varchar(30)	人事变动编号	√		
yggh	Varvhar(30)	员工工号			
ygxm	Varvhar(30)	员工性名			
ygxb	Varvhar(10)	员工性别			√
bdqzw	Varvhar(30)	变动前职务			√
bdqbm	Varvhar(30)	变动前部门			√
bdqxz	Flaot(8)	变动前薪资			√
bdhzw	Varvhar(30)	变动后职务			√
bdhbm	Varvhar(30)	变动后部门			√
bdhxz	Float(8)	变动后薪资			√
sxrq	Varvhar(20)	生效日期			√
pzr	Varvhar(30)	批准人			√
pzrq	Varvhar(20)	批准日期			√

表 4-7 人事档案（hr_rsdab）

字段名	数据类型`	描述	是否主键	外键	是否为空
yggh	Varchar（30）	员工工号	√		
ygxm	Varchar（30）	员工姓名			
ygxb	Varchar(10)	员工性别			√
sfzh	Varchar（30）	身份证号			√
hyzk	varchar(10)	婚姻状况			√
csrq	Varchar(20)	出生日期			√
mz	Varchar(30)	民族			√
sg	int	身高			√
tz	Int	体重			√
lxfs	Varchar(30)	联系方式			√
jzdz	Varchar(100)	居住地址			√
jg	Varchar(100)	籍贯			√
xl	varchar(20)	学历			√
zy	Varchar(20)	专业			√

wy	Varchar(20)	外语			√
byyx	Varchar(100)	毕业院校			√
bysj	Varchar(20)	毕业时间			√
sm	varchar（2000）	说明			√

表 4-8 专业资料（hr_zyzlb）

字段名	数据类型	描述	是否主键	外键	是否为空
zybh	Varchar(30)	专业编号	√		
zymc	Varchar(30)	专业名称			√

表 4-9 学历类型（hr_xllxb）

字段名	数据类型	描述	是否主键	外键	是否为空
xlbh	Varchar(30)	学历编号	√		
xlmc	Varchar(30)	学历名称			√

表 4-10 合同管理（hr_htglb）

字段名	数据类型	描述	是否主键	外键	是否为空
htbh	Varchar(30)	合同编号	√		
yggh	Varvhar(30)	员工工号			
ygxm	Varvhar(30)	员工性名			
htlx	Varvhar(30)	合同类型			√
htsx	Varvhar(30)	合同属性			√
sxrq	Varvhar(20)	生效日期			√
zzrq	Varvhar(20)	终止日期			√
htxz	Flaot(8)	合同薪资			√
qsrq	Varvhar(20)	签署日期			√
htnr	Varvhar(3000)	合同内容			√

表 4-11 请假信息表（hr_qjxxb）

列名	数据类型	描述	是否主键	外键	是否为空
qjdh	vchar（30）	假条单号	√		
ygbh	vchar（30）	员工编号		√	
qjsy	vchar（400）	请假事由			√
swjj	vchar（100）	事务交接			√
qjts	int	请假天数			√
spr	vchar（30）	审批人			√
rq	vchar（30）	日期			√

表 4-12 出差信息表（hr_ccxxb）

列名	数据类型	描述	是否主键	外键	是否为空
ccdh	vchar（30）	出差单号	√		
rq	vchar（30）	日期			
ygbh	vchar（30）	员工编号		√	
ccsy	vchar（400）	出差事由			√
swjj	vchar（100）	事务交接			√
ccts	int	出差天数			√
spr	vchar（30）	审批人			√

表 4-13 考勤日历表（hr_kqrlb）

列名	数据类型	描述	是否主键	外键	是否为空
kqh	vchar（30）	考勤号	√		
rq	vchar（30）	日期			
ygbh	vchar（30）	员工编号		√	
swcd	int	上午迟到			√
swzt	int	上午早退			√
swkg	int	上午旷工			√
xwcd	int	下午迟到			√
xwzt	int	下午早退			√
xwkg	int	下午旷工			√

表 4-14 用户表(hr_yhb)

列名	数据类型	描述	是否主键	外键	是否为空
yhm	varchar(30)	用户名	√		
mm	varchar(30)	密码			
yhms	varchar(100)	用户描述			√

表 4-15 角色表(hr_jsb)

列名	数据类型	描述	是否主键	外键	是否为空
jsbh	varchar(20)	角色编号	√		
jsm	varchar(30)	角色名			
Jsmm	Varchar(100)	角色描述			√

表 4-16 角色分配表(hr_jsfpb)

列名	数据类型	描述	是否主键	外键	是否为空
fpbh	uniqueidentifier(16)	分配编号	√		
yhm	varchar(30)	用户名		√	
jsbh	varchar(20)	角色编号		√	

表 4-17 权限表(hr_qxb)

列名	数据类型	描述	是否主键	外键	是否为空
qxbh	varchar(20)	权限编号	√		
qxm	varchar(30)	权限名			
zydz	varchar(100)	资源地址			

表 4-18 权限分配表(hr_qxfpb)

列名	数据类型	描述	是否主键	外键	是否为空
fpbh	uniqueidentifier(16)	分配编号	√		
jsbh	varchar(20)	角色编号		√	
qxbh	varchar(20)	权限编号		√	

第5章 系统详细设计

系统详细设计是对每个模块功能的具体设计，包括界面、功能模块和设计要点等内容。本次系统开发是分小组三人协同开发，我在本次开发中主要负责人事管理模块、考勤管理模块和系统设置三部分。

5.1 人事管理模块详细设计

人事管理模块由人事基本信息、工种类型、职位类型、员工状态、部门信息、人事档案、人事变动、专业资料、学历类型和统计分析等子模块组成。

5.1.1 人事基本信息

1.界面

员工基本信息设计界面如图 5-1：

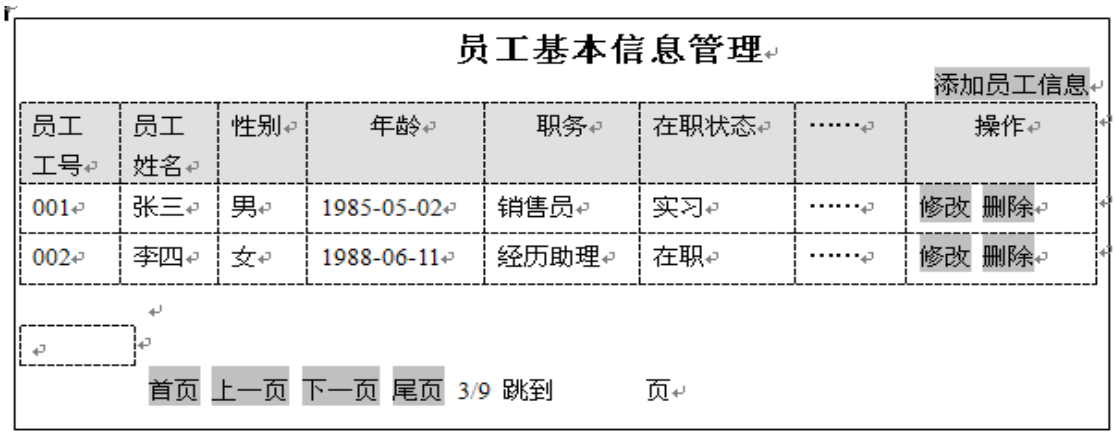

图 5-1　员工基本信息设计

2.功能模块

从表 hr_jbxx 中查询出公司所有员工信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改人员的数据，删除则从数据库中删除当前记录的数据。

3.设计要点

员工信息查询可查出公司所有员工的信息，多条记录需要分页功能，数据与工种类型表，职位类型表，在职状态表相关联。在删除数据时需要弹出确认窗口，避免误删。

5.1.2 工种类型

1.界面

职工工种类型设计界面为图 5-2：

2.模块功能

从表 hr_gzlx 中查询出公司所有工种信息，在页面中动态生成表格记录。如果是修改，

则从数据库中检索出被修改工种的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

工种类型可以任意添加，修改，当删除时如果此工种类型下有员工所属此工种类型，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

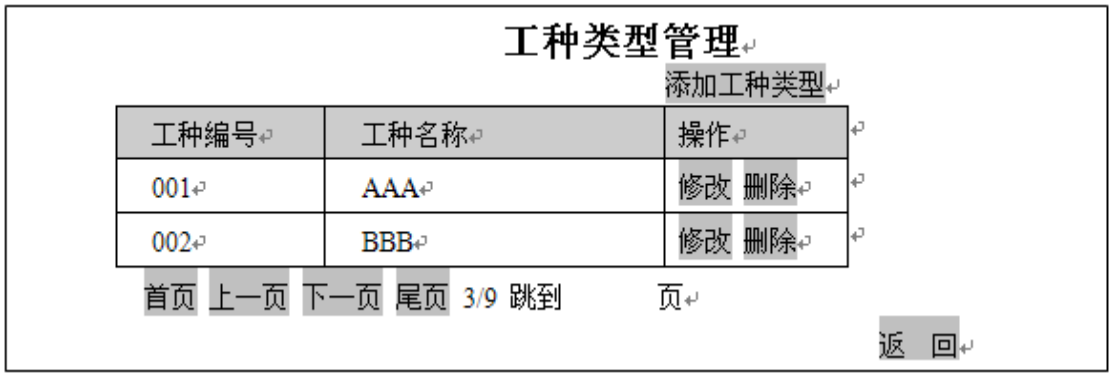

图 5-2 职工工种类型设计界面

5.1.3 职位类型

1. 界面

职位类型页面设计如图 5-3：

图 5-3 职位类型页面

2.模块功能

从表 hr_zwlx 中查询出公司所有职位类型信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改工种的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

职位类型可以任意添加，修改，当删除时如果有员工所属此职位类型，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

5.1.4 员工状态

1.界面设计：

员工状态界面设计如图 5-4：

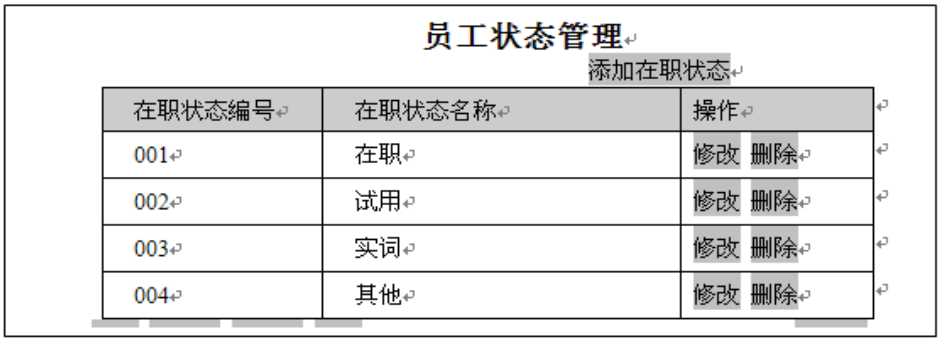

图 5-4　员工状态界面设计

2.模块功能

从表 hr_zzzt 中查询出公司所有在职状态信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改工种的数据，删除则从数据库中删除当前记录的数据。

3.设计要点

在职状态可以任意添加，修改，当删除时如果有员工所属此在职状态，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

5.1.5 部门信息

1.界面设计：

部门信息界面设计如图 5-5：

图 5-5 部门信息界面设计

2.模块功能

从表 hr_bmfl 中查询出公司所有部门信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改部门的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

部门分类可以添加，修改，当删除时如果有员工所属此部门分类，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

5.1.6 人事档案

1. 界面设计：

人事档案管理界面设计如图 5-6：

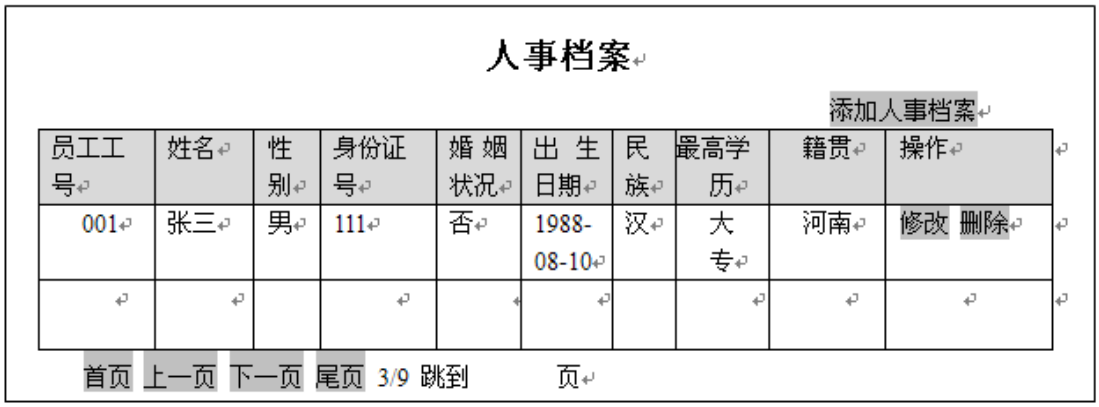

图 5-6 人事档案管理界面设计

2.功能模块

从表 hr_dagl 中查询出所有员工的档案信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改员工档案的数据信息，删除则从数据库中删除当前记录的数据。

3. 设计要点

档案信息可以添加，修改，如果要删除，需要弹出确认窗口，避免误删。

5.1.7 专业资料

1. 界面设计

专业资料管理界面设计如图 5-7：

2.模块功能

从表 hr_zyzl 中查询出公司所有工种信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改工种的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

专业资料可以任意添加，修改，当删除时如果有员工所属此专业资料，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

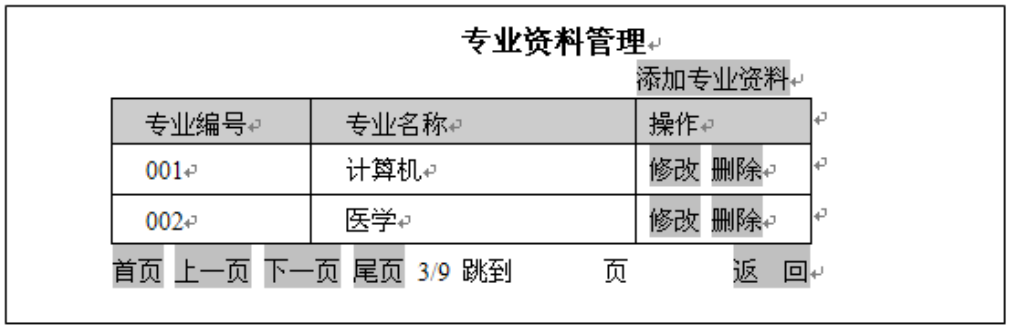

图 5-7 专业资料管理界面设计

5.1.8 学历类型

1. 界面设计

学历类型的界面设计图 5-8：

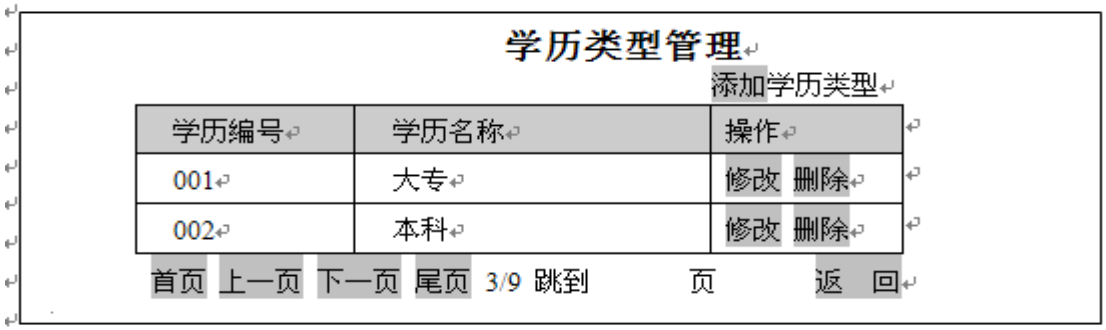

图 5-8 学历类型的界面设计

2. 模块功能

从表 hr_xllx 中查询出所有学历类型信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改学历类型的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

学历类型可以任意添加，修改，当删除时如果有员工所属此学历类型，则无法删除。如果可以删除，需要弹出确认窗口，避免误删。

5.1.9 人事变动

1. 界面设计

人事变动界面设计为图 5-9：

2. 模块功能

从表 hr_rsbd 中查询出公司所有人事变动信息，在页面中动态生成表格记录。如果是修改，则从数据库中检索出被修改部门的数据，删除则从数据库中删除当前记录的数据。

3. 设计要点

如果执行添加人事变动信息功能，保存成功则在表 hr_rsbd 中添加一条记录，同时更新员工基本信息表 hr_jbxx 中数据，保存按钮同时操作两个表的数据。

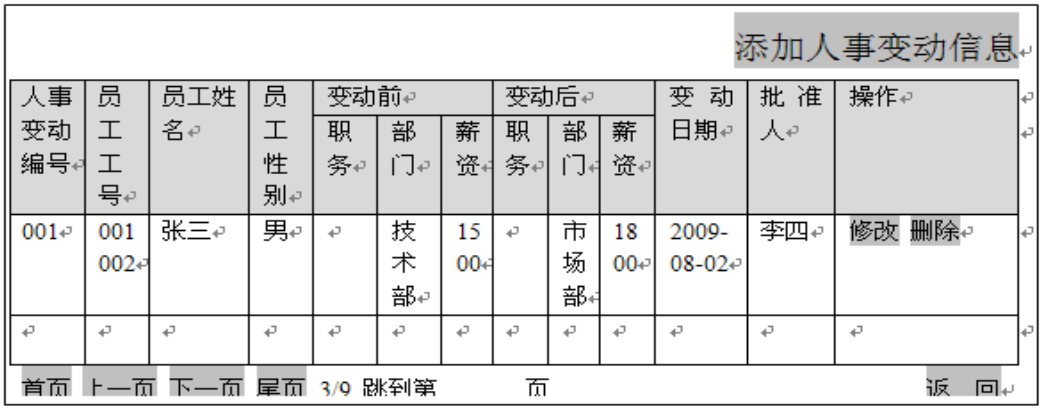

图 5-9 人事变动界面设计

5.1.10 合同管理

1. 界面

合同管理界面设计图 5-10：

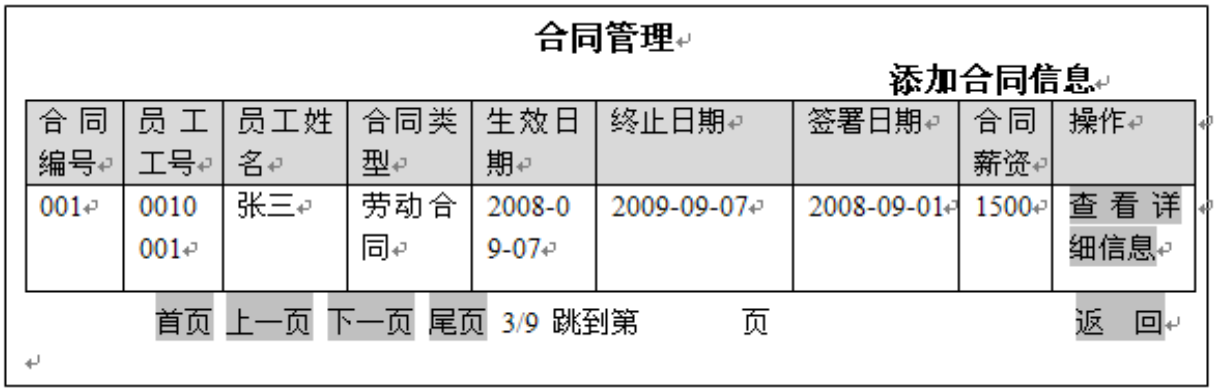

图 5-10 合同管理界面设计

2. 模块功能

从表 hr_htgl 中查询员工合同信息，查看合同概要，并提供合同详细信息查询。

3. 设计要点

合同管理只提供添加，删除功能，不提供修改功能

5.1.11 统计分析

1. 人事管理统计分析界面设计如图 5-11：

如图 5-11 人事管理统计分析界面设计

2. 模块功能

分析人事管理的员工职位比例报表、员工的学历分布报表、员工状态比例报表和员工专业分布报表。

3. 设计要点

统计分析要做到反映公司部门职位、专业、学历和员工职位状态结构。

5.2 考勤管理模块详细设计

该模块主要涉及到请假管理、出差管理、加班管理、出勤管理及相应的统计分析。

5.2.1 请假管理模块

1. 用户界面

请假管理用户界面设计如图 5-12：

图 5-12 请假管理用户界面

2.模块功能

实现请假信息的填加, 修改, 查询等.

用户点击写入请假信息按钮将请假信息写入数据库

用户可以按员工编号或者日期,或者部门查询请假信息,

也可以将查询出来的请假信息删除之.

3.设计要点

1. 用户输入日期,输入员工编号后,当焦点离开输入框后,使用 dwr 技术[7]到员工信息表用员工编号查询取得员工的相应信息,自动填充员工姓名,部门,职位等信息.

2. 请假天数用户输入的是字符型数据,写入数据库要转换成数字整形.

3. 请假事由,事务交接两项内容包含的字符串比较长,应该注意数据库中相应的字段数据类型的长度.

5.2.2 出差管理模块

1.用户界面:

出差管理界面设计如图 5-13:

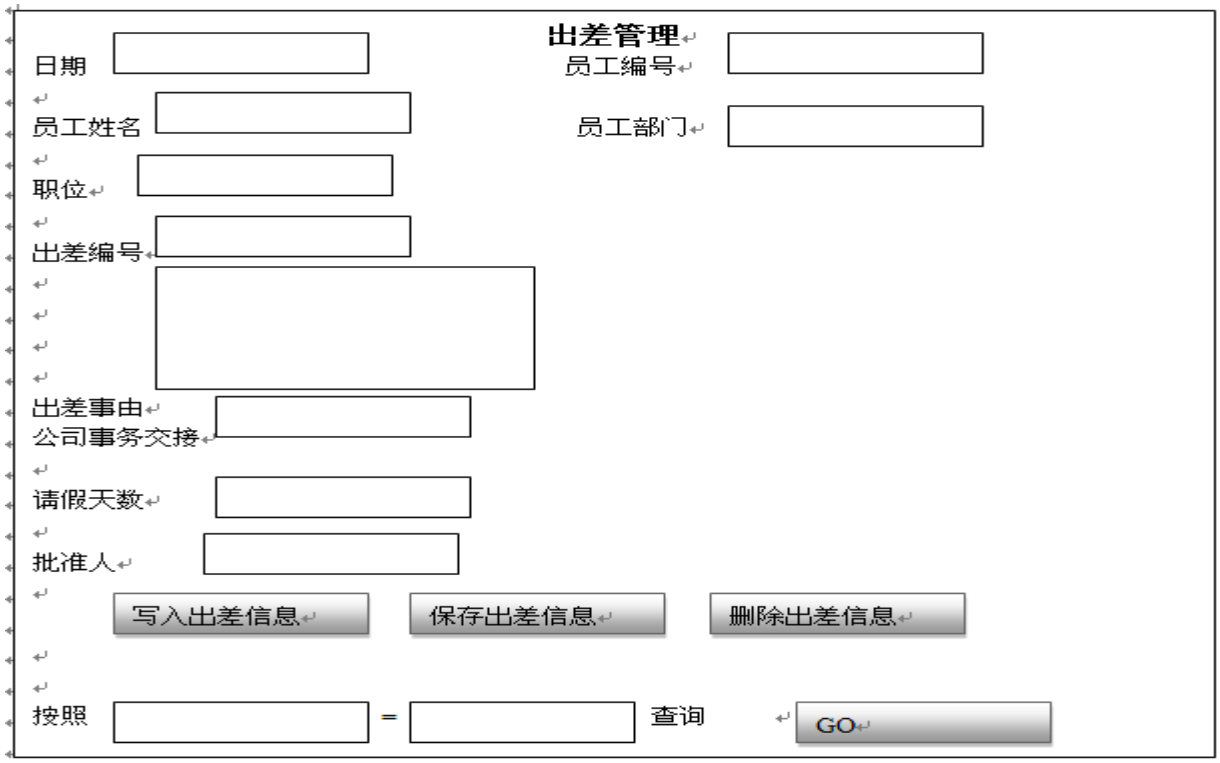

图 5-13 出差管理界面设计

2.模块功能:

实现出差信息的填加,修改,查询等。

用户点击写入请假信息按钮将出差信息写入数据库。

用户可以按员工编号或者日期,或者部门查询出差信息。

也可以将查询出来的出差信息删除之。

3.设计要点：

1. 用户输入日期,输入员工编号后,当焦点离开输入框后,使用 dwr 技术到员工信息表用员工编号查询取得员工的相应信息,自动填充员工姓名,部门,职位等信息.

2. 请假天数用户输入的是字符型数据,写入数据库要转换成数字整形.

3. 请假事由,事务交接两项内容包含的字符串比较长,应该注意数据库中相应的字段数据类型的长度.

5.2.3 日常考勤管理模块

1.界面设计

考勤管理界面设计如图 5-14：

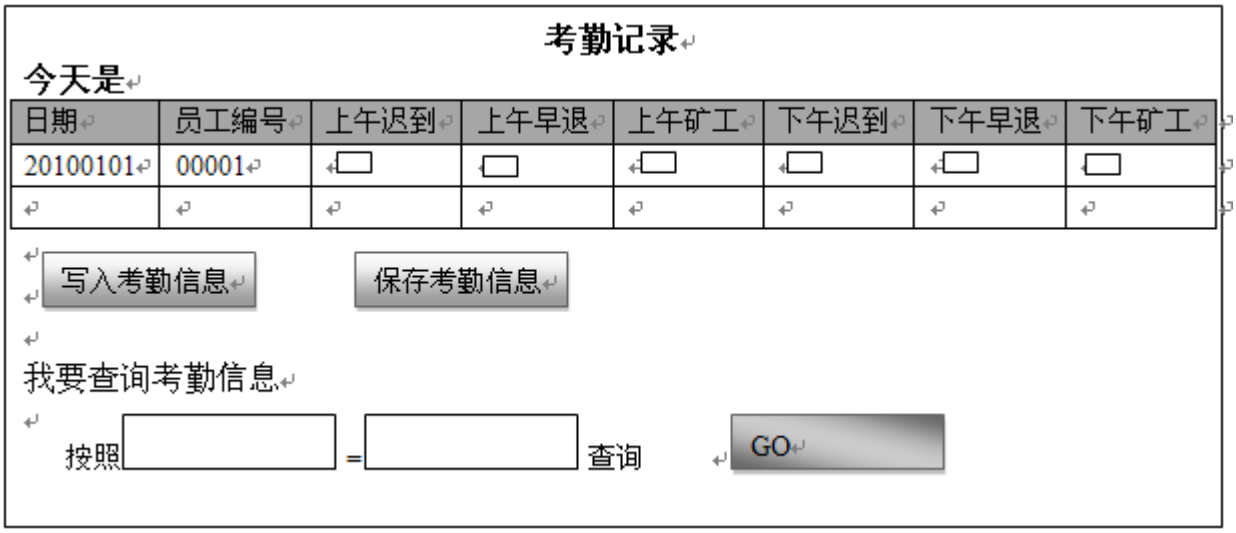

图 5-14 考勤管理界面设计

2.模块功能

实现考勤功能.并能将考勤信息保存到数据库中.并具备查询考勤情况的功能.

3.设计要点

考勤信息和工资的计算有一定的关系,所以考勤管理这里着一块的信息保存和查询比较重要，当每月发工资的时候,就需要通过循环语句从一个月的 30 天请假信息中查询该员工编号有无请假信息,有的话,每天扣多少工资,总共扣多少工资. 然后在查询该员工有无迟到早退或者矿工信息,用循环语句可以从考勤信息表中查出该员工一个月的考勤信息.根据公司的规定,指定相应的处理。

5.2.4 加班管理模块

1.界面设计

加班管理模块界面设计如图 5-15：

图 5-15 加班管理界面设计

2.模块功能

实现员工的加班考勤功能，对员工的加班考勤记录进行保存、删除、修改，并能按一定条件进行查询。

3.设计要点

1. 用户输入日期,输入员工编号后,当焦点离开输入框后,使用 dwr 技术到员工信息表用员工编号查询取得员工的相应信息,自动填充员工姓名,部门,职位等信息.

2. 加班天数用户输入的是字符型数据,写入数据库要转换成数字整形.

3.加班事由,事务交接两项内容包含的字符串比较长,应该注意数据库中相应的字段数据类型的长度.

5.2.5 考勤数据分析

考勤数据分析包括四部分：请假考勤报表、出差考勤报表、日常考勤报表、加班考勤

报表。

四部分的内容大体相同，所有界面进行统一设计。

1.界面设计

报表设计界面如图 5-16：

图 5-16 报表界面设计

3.模块功能

报表设计主要是对记录的考勤数据进行具体分析并以直观的视图给我们显示出来，方便进行统计分析。例如：出勤报表:可以统计某段时间里各个部门的迟到、早退和矿工次数比例。

4.设计要点

开始时间结束时间都是用 JavaScript 实现，通过单击时间触发可以选择自己要统计的时间段。报表实现技术是 jfreechar 报表技术实现。

5.3 权限管理

5.3.1 用户管理

1.界面设计

用户管理的界面设计如图 5-17：

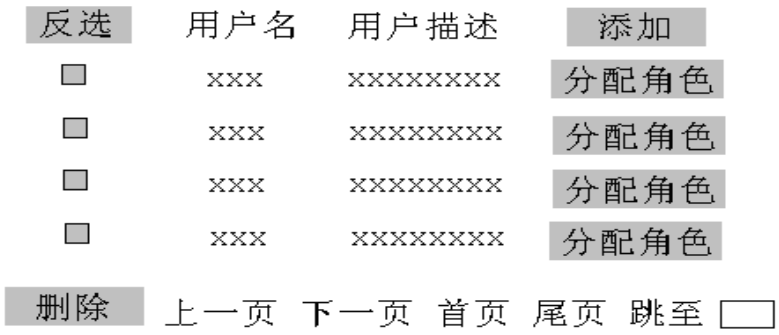

图 5-17 用户管理的界面设计

2. 模块功能

本模块是用户管理的主界面,一般只有管理员才有权使用,可以对用户进行添加、选择，浏览，删除与修改操作。

3. 设计要点

用户描述应为用户的员工号(如果有的话)、真实姓名、所在部门、职务为系统管理员分配角色和权限提供参考。

删除用户时需有出现提示框,防止管理员误删用户。

5.3.2 密码修改

1..界面设计

密码修改界面设计如图 5-18

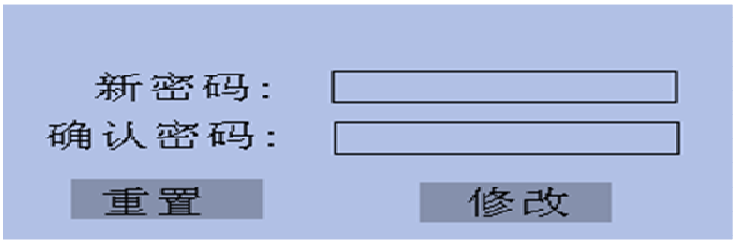

图 5-18 密码修改界面设计

2. 模块功能

本模块用以修改单前用户自己的密码，本功能只有当前用户能使用，即使是系统管理员也不能修改其它用户的密码。

3.设计要点

用户权限管理与用户名和密码有关的模块都需要用到“字符合法性检测”和防止 Sql 注入漏洞。

5.3.3 角色管理

1. .界面设计

角色管理界面设计如图 5-19：

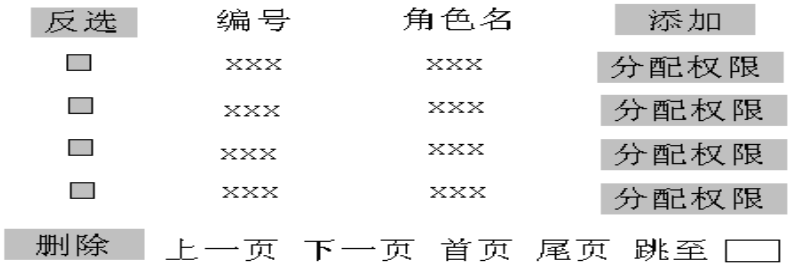

图 5-19 角色管理界面设计

2. 模块功能

本模块用于对角色的综合管理包括：添加新角色、删除已有角色、为角色分配权限，浏览系统所有角色。

3. 设计要点

成功添加角色后角色并没有权限，需要调用“权限分配”模块,为角色分配权限。

5.3.4 资源权限管理

1. 界面设计

资源权限管理界面设计如图 5-20：

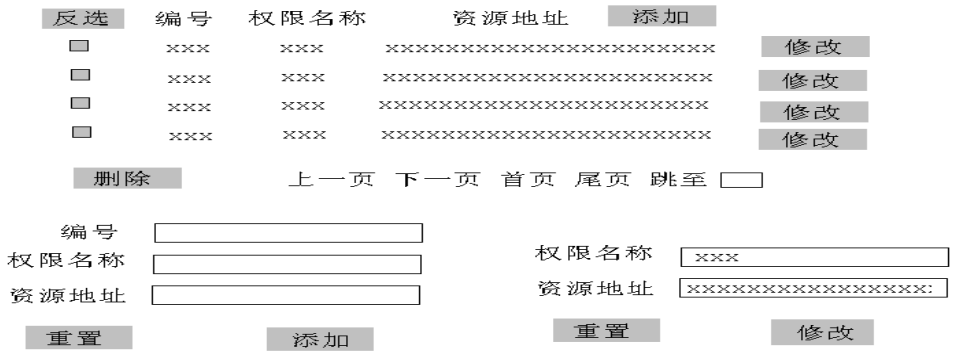

图 5-20 资源权限管理界面设计

2. 模块功能

对“权限”进行增、删、改、查操作。

3.设计要点

权限编号为主键，但权限名称也不允许重复。

第6章 系统实现

本系统是一个典型的MVC框架的扩展应用，首先表示层用jsp+servlet来实现，包括视图和控制器。模型层包括业务逻辑层和数据库持久层。业务逻辑层主要是应用了Fileter技术[8]进行过滤Action事务，数据库持久层主要是用了数据访问对象（DAO）和Factory设计模式来实现。

6.1 系统开发环境

本系统采用java开发环境JDKI.6,Web服务器使用Apache的tomcat6.0，数据库使用SqlServer2005,开发工具为Eclipse5.5.

向开发的Java Web项目加入数据库驱动包、dwr包、jstl包、xml解析的jdom包；项目用到的库文件如图6-1所示。

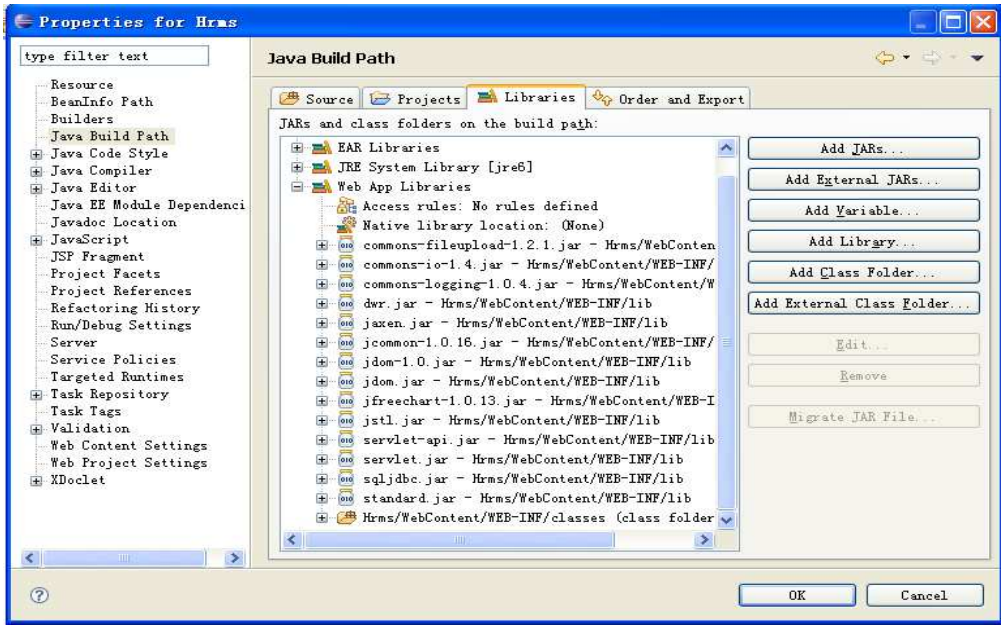

图6-1项目用到的库文件

6.2 创建配置文件

(1)web.Xml

通常所有的MVC框架都需要Web应用加载一个核心控制器，对于本本系统而言，需要加载ActionServlet，它就是本系统的核心控制器，只要Web应用加载了ActionServlet，就可以获得整个MVC框架所提供的服务。Web. xml_里<filter>元素用来指定要加载MVC框架的核心控制器ActionServlet，<Servlet—mapping>元素用来指定让MVC框架来处理用户的哪些请求(URL)，当它的子元素<url—pattern>的值为“*”时表示用户的所有请示都使用此框架来处理。在系统的实现过程中使用到了dwr技术、上传下载（Upload）技术和报表

处理等都要求在web.xml文件中配置。web.xml文件部分如下：

```
<servlet>
    <servlet-name>action</servlet-name>
        <servlet-class>
            com.softstone.controller.action.ActionServlet
        </servlet-class>
        <init-param>
            <param-name>action-config</param-name>
        <param-value>/WEB-INF/action-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>db-config</param-name>
            <param-value>/WEB-INF/db-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>sys-config</param-name>
            <param-value>/WEB-INF/sys-config.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>dwr-invoker</servlet-name>
        <servlet-class>
            org.directwebremoting.servlet.DwrServlet
        </servlet-class>
        <init-param>
            <param-name>debug</param-name>
            <param-value>true</param-value>
        </init-param>
    </servlet>
```

（2）action-config.xml

通过web.xml文件把action-config.xml文件进行加载，在MVC框架中对action-config.xml文件进行jdom解析并根据解析到的参数解析跳转。例如系统登陆Action如下：

```
<action path="/login"
        actionClass="hrms.qxgl.action.yhgl.LoginAction">
        <forward name="failure" path="login.jsp" />
        <forward name="success" path="" redirect="true"/>
    </action>
```

以上配置信息表示如果在系统登陆的时候能够正确的得到Login对应的Action，返回为

success则进入系统，返回为failure则跳转到login.jsp登陆页面。

6.3 实现数据持久层

(1)创建 db-config.xml 的配置文件

db-config.xml配置文件用于配置数据库连接属性以及Hibernate运行时所需的各种属性，包括指定数据库的驱动程序、连接数据库的URL、连接数据库的户名和口令、数据库方言、数据库名等。要将底层数据库内的变动映射到整个应用，只需要修改该文件内的属性值。配置文件有XML和Java属性文件两种形式。URL指定连接数据库的URL。driverClassName指定数据库的驱动程序。Username为数据库用户名，password是数据库登陆密码。db-config.xml文件如下：

```
<DataPool>
    <DataSource name="mssql_HRMS">
        <url>
<value>jdbc:sqlserver://localhost:1433;databaseName=HRMS</value>
        </url>
        <driverClassName>
    <value>com.microsoft.sqlserver.jdbc.SQLServerDriver</value>
        </driverClassName>
        <username>
            <value>sa</value>
        </username>
        <password>
            <value>123</value>
        </password>
    </DataSource>
</DataPool>
```

（2）创建持久化类

本系统实现一个DTO接口把数据库中的表映射到一个持久化类，通过这种映射，系统把所有对对数据库表的操作都转移到为对java类的操作。

持久化类是指其实例需要持久化到数据库中的类。持久化类通常都是域模型中的实体域类。持久化类符合JavaBean的规范，包含一些属性，以及与之对应的getter和setter方法。数据库中有几个表就有几个持久化类。

以下是角色分配表对应的一个持久化类：

```
import com.softstone.model.DTO;
public class Hr_jsfpbDTO implements DTO {
    private String fpbh;
    private String yhm;
    private String jsbh;
```

```
    public String getFpbh() {
        return fpbh;
    }
    public void setFpbh(String fpbh) {
        this.fpbh = fpbh;
    }
    public String getYhm() {
        return yhm;
    }
    public void setYhm(String yhm) {
        this.yhm = yhm;
    }
    public String getJsbh() {
        return jsbh;
    }
    public void setJsbh(String jsbh) {
        this.jsbh = jsbh;
    }
}
```

(3)通过 DAO 操作数据库。

DAO 是一个对 DTO 持久化类操作的接口，其中定义了 query（String pk）方法、update(DTO dto)方法、add(DTO dto)方法、delete(String pk)方法、List<DTO> queryAll()方法。其中 DAO 中有数据库链接的 Connection 、分页（Pagination）属性。

query（String pk）方法：查询数据库中 PK 用户信息；

update(DTO dto)方法：更新数据库中域对象的状态；

add(DTO dto)方法：增加数据库中域对象的状态；

delete(String pk)方法：删除数据库域对象的状态；

List<DTO> queryAll()方法：查询数据库中所有的域对象状态；

系统中都是运用 DAO 实现的数据库操作，在这以权限管理模块为例：

query（String pk）方法代码如下：

```
public   DTO query(String pk) {
        Hr_jsbDTO jsDTO = null;
        try {
            PreparedStatement pstmt = conn.prepareStatement(SQL_SELECT);
            pstmt.setString(1, pk);
            ResultSet rs = pstmt.executeQuery();
            if (rs != null && rs.next()) {
                jsDTO = new Hr_jsbDTO();
                jsDTO.setJsbh(rs.getString("jsbh"));
```

```
                jsDTO.setJsm(rs.getString("jsm"));
                jsDTO.setJsms(rs.getString("jsms"));
            }
            rs.close();
            pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return jsDTO;
    }
```

update(DTO dto)方法代码如下：

```
    public int update(DTO dto) {
        Hr_jsbDTO jsDTO = (Hr_jsbDTO) dto;
        int flag = 0;
        try {
            PreparedStatement pstmt = conn.prepareStatement(SQL_UPDATE);
            pstmt.setString(1, jsDTO.getJsm());
            pstmt.setString(2, jsDTO.getJsms());
            pstmt.setString(3, jsDTO.getJsbh());
            flag = pstmt.executeUpdate();
            pstmt.close();
        } catch (SQLException e) {
            flag = 0;
            e.printStackTrace();
        }
        return flag;
    }
}
```

add(DTO dto)方法代码如下：

```
    public int add(DTO dto) {
    Hr_jsbDTO jsDTO = (Hr_jsbDTO) dto;
        int flag;
        try {
      PreparedStatement pstmt = conn.prepareStatement(SQL_ADD);
            pstmt.setString(1, jsDTO.getJsbh());
            pstmt.setString(2, jsDTO.getJsm());
            pstmt.setString(3, jsDTO.getJsms());
            flag = pstmt.executeUpdate();
            pstmt.close();
        } catch (SQLException e) {
            flag = 0;
            e.printStackTrace();
        }
        return flag;
```

```
        }
```
delete(String pk)方法代码如下：
```
    public int delete(String pk) {
            int flag = 0;
            try {
    PreparedStatement pstmt = conn.prepareStatement(SQL_DELETE);
                pstmt.setString(1, pk);
                flag = pstmt.executeUpdate();
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return flag;
    }
```
List<DTO> queryAll()方法部分代码如下：
```
    public List<DTO> queryAll() {
            List<DTO> yhList = new ArrayList<DTO>();
  PreparedStatement pstmt =
conn.prepareStatement(SQL_SELECT_ALL,ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSe
t.CONCUR_READ_ONLY);
                ResultSet rs = pstmt.executeQuery();
                this.setStartRow(rs);
                while (rs != null && rs.next()) {
        if(this.isPagination&&rs.getRow()>pagination.getEndRow())break;
                    Hr_jsbDTO jsDTO = new Hr_jsbDTO();
                    jsDTO.setJsbh(rs.getString("jsbh"));
                    jsDTO.setJsm(rs.getString("jsm"));
                    jsDTO.setJsms(rs.getString("jsms"));
                    yhList.add(jsDTO);
                }
                rs.close();
                pstmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return yhList;
        }
```
6.4 控制层的实现

控制层负责接收用户请求、更新模型，以及选择合适的视图组件返回给用户，在本系统的实现中控制层是实现对一个抽象类Action的继承来实现，通过继承Action中的execute(HttpServletRequest request, HttpServletResponse response)的方法，另外还有一个

ActionServlet类继承了HttpServlet并对action-config.xml文件进行加载并解析。Action负责处理一项具体的业务，它是用户请求和业务逻辑之间的桥梁，每个Action充当客户的一项业务代理。Action的主要功能是接收页面中的一些数据，然后根据action-config.xml实例包含的映射信息决定将当前的请求转发给哪个Action。Action的工作就是通过访问HTTP会话、HTTP请求和表单参数等调用业务逻辑，最后把响应映射到以持久化类形式存在的模型上，来完成特定的功能。最后，Action返回的结果会通过配置文件映射到JsP页面上，JSP会渲染视图并显示给用户，以下是Action类部分代码：

```
public abstract class Action {
    protected final String SUCCESS="success";
    protected final String FAILURE="failure";
    protected ActionMessage message=new ActionMessage();
    protected Connection con;
public String process(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            request.setAttribute("message", message);
            return execute(request, response);
        }
 public void setConn(Connection conn){
            this.conn=conn;
        }
```

6.5 系统 WEB 层实现

本系统的WEB层的实现主要是jsp技术结合JSTL标签来实现。Jsp技术主要是用于显示用户界面，发送Action请求和接收响应完成交互，做到了视图与业务逻辑层的解耦。JSTL标签库为表示层提供了大量丰富的标签库，有利于JSP项目迁移到别的项目中、扩展页面功能和易于维护的优势。本系统每个显示页面都是由两部分组成：头页面，主页面。头页面做成了一个jsp文件，在每一个显示页面中只要将这两个文件include进来就可以。这样使得系统页面的整体风格一致，同时也避免了代码重复编写，程序员只需为每个页面实现其中间的主体部分。本系统除index．jsp页面外所有的显示页面都由action转发过来，在action中将页面所需要的数据封装成对象，jsp页面直接从容器中获取数据，通过这种数据传递方式避免了在页面写入java代码，从而使得显示层和逻辑层耦合性大大降低。

6.6 系统部署

本系统的系统部署如 6-12 图：

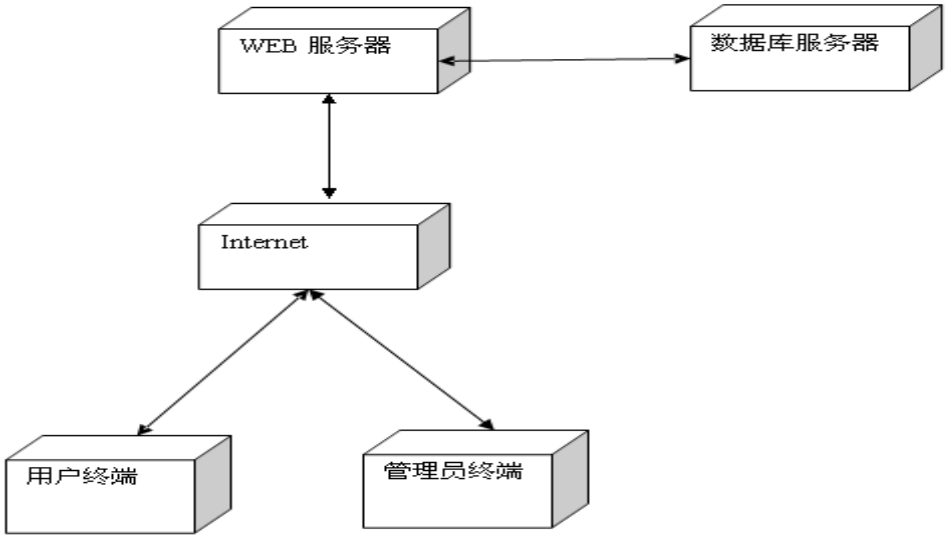

图 6-1 系统部署图

5.7 系统界面介绍

登陆页面如图 6-2，为本系统的入口，用户输入正确的用户名和密码才能进入系统主页：

图 6-2 登陆页面

系统主页页面如图 6-3：系统主页主要是系统的功能结构，相应的权限用户可以双击进入相应的模块进行操作，中间左部分系统默认显示的是招聘管理内容。

图 6-3 系统主页

人事管理页面如图 6-4：

图 6-4 人事管理页面

考勤管理页面如图 6-5：

图 6-5 考勤管理页面

系统设置管理页面图 6-6：

图 6-6 系统设置页面

第7章 总结

本系统的设计实现基本上满足了用户需求，性能需求，比较容易维护；开发框架松散耦合度高、可扩展且高效；代码重用性较高，利于调试和测试。

对本系统的开发大致总结如下：

1．人力资源管理系统用 MVC 框架扩展开发，使得层次之间达到一定的松散耦合。在开发过程中各个层与层之间的工作是相对独立的，不同的业务逻辑可以基于模块开发，有利于团队协同开发。业务规则、输入校验存在于中间控制层，因此当业务规则发生改变或增加时，只需要更改或增加相应业务处理逻辑，在很短的时间内就能完成系统开发和部署，而客户端应用程序不需要做任何逻辑处理，能很好的满足了各个部门快速多变的业务需求，从而拥有了很好的可扩展性和可维护性。

2．系统利用 DTO 的对象持久化服务，更有效地进行数据库数据到业务对象的 O / R 映射，简化了客户端对数据库数据的操作过程。开发人员可以专心地实现业务逻辑而不用分心于烦琐的数据库方面的逻辑，减小了出错机会，同时能为开发小组提供更合理的模块划分方法，节约开发成本和时间，提高业务应用的性能，提供灵活的业务逻辑。

3．系统中存在业务逻辑过于烦琐、页面跳转结构复杂问题。控制层中的 Action 映射复杂，对象关系之间的映射操作工作量大，配置文件过多，当页面表单频繁变化时灵活性较差，jsp 标签库 jstl 维护的成本较高，维护难度大。所以如果采用 Struts2、Hibernate、Spring 三层框架对系统进行有效整合，可以对本系统的层次和业务逻辑分析更加的清晰完善。

4.系统开发中的需求分析主要是通过从网络资料中提取，没有太多的与实际的公司人力资源管理的相关人员进行沟通，因此需求分析有些地方可能不太合理和健全。在系统实现的时候就会出现系统功能实现不是很优化、高效。

致 谢

本文是在导师张玉华老师的悉心指导下完成的。从开题、分析到撰稿和修改等整个论文撰写过程中，导师都给予了细心的建议和指导。导师对本人的学习等各个方面都给予了无微不至的关怀与帮助。在此，我向导师表达崇高的敬意和衷心的感激!

感谢大学四年求学期间所有老师在学习上给我的帮助和指导。我每一点知识的积累都与他们辛勤的帮助密不可分，他们的人格魅力和治学态度令我十分钦佩。

感谢所有同学的关心和帮助。最后，再次衷心感谢所有关心帮助我的老师、同学和领导!

参考文献

[1].俞彩云、李士雨企业人力资源管理系统的分析与设计计算机应用与研究，2004，21(2)：186—188

[2]. 王锋，张景，何文娟等基于Internet的人力资源管理系统微机发展，2003,13(9)：95—97

[3].张海藩编著<<软件工程导论>>(第四版) 清华大学出版社

[4].孙卫琴著<<精通Struts:基于MVC的Java Web设计与开发>>2007年03月电子工业出版社

[5].李刚编著<<疯狂Java讲义>>电子工业出版社

[6].程云志、张帆、崔翔编著<<数据库原理与SQL Server 2005应用教程>>机械工业出版社

[7].单东林、张晓菲、魏然著<<锋利的jQuery(前端开发系列) >>2009年06月人民邮电出版社

[8].李宁等编著<<Java Web 开发技术大全>>清华大学出版社

附录 I

英文原文

Human resource management system

Operators have especially exigent demands of persons with ability. Under the condition that qualified persons with ability cannot completely be obtained outside enterprise, the problem of upgrading enterprise internal staff's working capacity and the ultimate improvement of enterprise's rapid responding capacity and well execution need to be solved as soon as possible. Therefore, operators need to establish staff certification system so as to implement comprehensive post capacity certification of each post staff within the enterprise and sufficiently coordinate the system with training management system to confirm training effects. Simultaneously, training management system pertinent to the existing staff within the enterprise is required to be established and organization, duty, process and system which training management involves should be made clear to guarantee that the internal resources can be sufficiently and reasonably utilized to upgrade the internal staff's working capacity. So,the Human Resource Management System comes out.

A Human Resource Management System (HRMS, EHRMS), Human Resource Information System (HRIS), HR Technology or also called HR modules, or simply "Payroll", refers to the systems and processes at the intersection between human resource management (HRM) and information technology. It merges HRM as a discipline and in particular its basic HR activities and processes with the information technology field, whereas the programming of data processing systems evolved into standardized routines and packages of enterprise resource planning (ERP) software. On the whole, these ERP systems have their origin on software that integrates information from different applications into one universal database. The linkage of its financial and human resource modules through one database is the most important distinction to the individually and proprietary developed predecessors, which makes this software application both rigid and flexible.

The function of Human Resources departments is generally administrative and not common to all organizations. Organizations may have formalized selection, evaluation, and payroll

processes. Efficient and effective management of "Human Capital" has progressed to an increasingly imperative and complex process. The HR function consists of tracking existing employee data which traditionally includes personal histories, skills, capabilities, accomplishments and salary. To reduce the manual workload of these administrative activities, organizations began to electronically automate many of these processes by introducing specialized Human Resource Management Systems. HR executives rely on internal or external IT professionals to develop and maintain an integrated HRMS. Before the *client–server* architecture evolved in the late 1980s, many HR automation processes were relegated to mainframe computers that could handle large amounts of data transactions. In consequence of the low capital investment necessary to buy or program proprietary software, these internally-developed HRMS were unlimited to organizations that possessed a large amount of capital. The advent of client–server, Application Service Provider, and Software as a Service or SaaS Human Resource Management Systems enabled increasingly higher administrative control of such systems. Currently Human Resource Management Systems encompass:

1.Payroll

2.Work Time

3.Benefits Administration

4.HR management Information system

5.Recruiting

6.Training/Learning Management System

7.Performance Record

8.Employee Self-Service

The payroll module automates the pay process by gathering data on employee time and attendance, calculating various deductions and taxes, and generating periodic pay cheques and employee tax reports. Data is generally fed from the human resources and time keeping modules to calculate automatic deposit and manual cheque writing capabilities. This module can encompass all employee-related transactions as well as integrate with existing financial management systems.

The work time gathers standardized time and work related efforts. The most advanced modules provide broad flexibility in data collection methods, labor distribution capabilities and data analysis features was outdated. Cost analysis and efficiency metrics are the primary functions.

The benefits administration module provides a system for organizations to administer and track employee participation in benefits programs. These typically encompass insurance, compensation, profit sharing and retirement.

The HR management module is a component covering many other HR aspects from application to retirement. The system records basic demographic and address data, selection, training and development, capabilities and skills management, compensation planning records and other related activities. Leading edge systems provide the ability to "read" applications and enter relevant data to applicable database fields, notify employers and provide position management and position control not in use. Human resource management function involves the recruitment, placement, evaluation, compensation and development of the employees of an organization. Initially, businesses used computer based information systems to:

☆ produce pay checks and payroll reports;

☆ maintain personnel records;

☆ pursue Talent Management.

Online recruiting has become one of the primary methods employed by HR departments to garner potential candidates for available positions within an organization. Talent Management systems typically encompass:

☆ analyzing personnel usage within an organization;

☆ identifying potential applicants;

☆ recruiting through company-facing listings;

☆ recruiting through online recruiting sites or publications that market to both recruiters and applicants.

The significant cost incurred in maintaining an organized recruitment effort, cross-posting within and across general or industry-specific job boards and maintaining a competitive

exposure of availabilities has given rise to the development of a dedicated Applicant Tracking System, or 'ATS', module.

The training module provides a system for organizations to administer and track employee training and development efforts. The system, normally called a Learning Management System if a stand alone product, allows HR to track education, qualifications and skills of the employees, as well as outlining what training courses, books, CDs, web based learning or materials are available to develop which skills. Courses can then be offered in date specific sessions, with delegates and training resources being mapped and managed within the same system. Sophisticated LMS allow managers to approve training, budgets and calendars alongside performance management and appraisal metrics.

The Employee Self-Service module allows employees to query HR related data and perform some HR transactions over the system. Employees may query their attendance record from the system without asking the information from HR personnel. The module also lets supervisors approve O.T. requests from their subordinates through the system without overloading the task on HR department.

Many organizations have gone beyond the traditional functions and developed human resource management information systems, which support recruitment, selection, hiring, job placement, performance appraisals, employee benefit analysis, health, safety and security, while others integrate an outsourced Applicant Tracking System that encompasses a subset of the above.

译文

人力资源管理系统

操作人员随着能力的要求来增加容量。在这种情况下，拥有专门用途的容量将不能从企业外部获得，企业升级内部员工的工作量和企业的快速反应能力的提升，以及很好的执行能力，这些问题需要尽可能快的解决。因此，操作人员需要建立员工文凭证书系统，以便完成每一个企业内部的在职员工的职位接受能力和充分协调培训管理系统来证明培训作用。与此同时，对于企业内部现有员工来说，培训管理系统要求被建立，而且，培训管理涉及的组织，职责，进程和系统应该被清楚，从而来确保内部资源可以被充分合理地使用来升级内部员工的工作能力。因而，人力资源管理系统应运而生。

一个人力资源管理系统，人力资源信息系统，人力资源技术或者所谓的人力资源模块，或者像一个简单的“工资表”，就是指那些在人力资源管理和信息技术之间的系统和程序的交集。它整合了人力资源管理作为一门学科，尤其作为它的基本的人力资源活动和信息技术领域的程序，然而，这些数据处理系统的执行逐渐演变成了标准的计算机程序和企业资源规划软件。整体而言，这个企业资源规划系统在软件上也有它的起源，就是它可以把来自不同应用程序的信息集成到一个通用数据库中。财务和人力资源模块通过一个数据库连接是它与那些之前独立发展的先辈们最大的区别，它使这种软件应用程序变得既固定又灵活易变。

人力资源部门的功能通常就是具有管理员性质的，并且对所有的组织来说不常见。组织可能有正式的选拔，评估，和发薪活动。高效的和有效的管理“人力资本”已经发展到一个日益紧迫和复杂的过程。人力资源功能由那些跟踪现有雇员数据组成，包括传统意义上的个人历史，技能，能力，业绩和薪水。为了降低这些管理活动的人工工作量，很多组织开始通过引进专门的人力资源管理系统来使很多程序电子自动化。人力资源的执行依赖于内部或外部的 IT 专门功能，进而可以发展和维护一个集成的综合性人力资源管理系统。20 世纪 80 年代后期，在客户—服务器发生演变之前，许多人力资源自动化程序被降级到大型计算机，它们可以处理巨大的数据交易。由于较低的资本投入，去买或者执行一个专门的软件是有必要的，对那些拥有大量资本的组织者来说，这些被发展成为内部的人力资源管理系统是无限制的。客户端---服务器，应用服务提供者和软件的出现作为一种服务或者称为软件服务化的人力资源管理系统促使这些系统日益升高的管理控制能力成为现实。当前，人力资源管理系统包括：

1， 薪金总数

2， 工作时间

3， 福利管理

4， 人力资源管理信息系统

5， 招聘

6， 培训/学习管理系统

7， 性能记录

8， 员工自动服务

工资模块通过集合员工时间和出勤情况，计算各种扣除和税收，生成定期支付票据和员工税收报告来使支付程序自动化。数据通常来自人力资源，用来计算自动存储的时间保持模块以及那些手工书写支票的能力。这些模块可以包含所有与员工有关的交易数据，也集成了现存的财务管理系统。

工作时间集合了标准的时间和与付出相关的工作量。最高级的模块在数据收集方法，劳动分配能力和数据分析特色提供广泛的灵活性，成本分析和效益指标是两个基础的功能。

福利管理模块为组织管理和跟踪员工是否参与到效益活动中提供了一个系统。它通常包括保险，赔偿，利润分享和退休。

人力资源管理模块是一个包括其它许多人力资源方面的组件，从申请到退休。该系统记录和处理基本的人口统计和地址数据，筛选，培训和发展，能力和技能管理，赔偿计划记录和其他相关的活动。交叉系统提供了“读”应用程序的能力，并且可以进入一个与申请数据库相关的数据，通告员工，提供管理职位和没有被使用的控制职位。人力资源管理涉及到招聘，评估，赔偿和一个组织的员工的发展。最初，基于信息系统使用的商业计算机功能有：

生成支付票据和工资总额报告；

维护人事记录；

追求智能管理局。

在线招聘已经成为一个受雇于人力资源管理部门的基础方法，被用来吸纳那些在组织内部适合于某些职位的有潜力的员工。人才管理系统通常包括：

分析组织内部员工工作效率；

确定有潜力的申请者；

通过公司现在面临的情况来招聘员工；

通过在线招聘网址或者市场上用于应聘者和申请者的出版物来招聘员工。

随之而来的在维护一个有组织的招聘努力中，在一个跨领域的人员分配活动中或者行业定制的工作海报中，在维护一个已经被用来作为提升精确追踪者系统，或者称为自动试验系统的发展的具有竞争性的展示活动的模板中。这个模板为很多组织者的管理员和跟踪员工培训与发展努力的活动提供了一个系统。

这个系统，通常被称为学习管理系统，如果一个独立的商品，允许人力资源跟踪员工的教育，文凭和技能，就像列举出来那些培训的课程，课本，光盘，学习的网站或者那些用来发展技能的可用材料。在日期确定的会话课程会被提供，，它具有代表性和那些在同一个系统中被映射和管理的培训资源。先进的学习管理系统允许管理者提升培训，预算和绩效指标的日历的同时管理和考核。

员工自我评价模块允许员工查询与人力资源相关的数据，执行一些运行于该系统上的人力资源交易数据。员工也可以从系统上查询他们的出勤记录，而无需从人力资源部问这些信息。这个模块也可以使上级通过该系统批准那些系统不超载的任务，就由人事部门下属完成的要求。

许多组织已经超越了传统的职能，并且发展了人力资源开发管理信息系统，它支持招聘，选拔，聘用，工作安置，业绩考核，员工福利分析，卫生，安全和保障，而其他功能整合了包含以上功能的子集的外包申请人跟踪系统。

附录 II

核心代码

model 层代码：
AbstractDAO 类

```
public abstract class AbstractDAO implements DAO{
    protected Connection conn;
    protected Pagination pagination;
    protected boolean isPagination=false;

    public Pagination openPagination(){
        isPagination=true;
        pagination=new Pagination();
        return pagination;
    }
    public void closePagination(){
        isPagination=false;
        pagination=null;
    }

    public void initPagination(int pageNum,int rowOfPage) throws PaginationException{
        if(!isPagination)throw new PaginationException("分页功能没有找开，请先执行
openPagination()方法");
        pagination.setCurPage(pageNum);
        pagination.setRowOfPage(rowOfPage);
    }
    protected void setStartRow(ResultSet rs){
        if(!isPagination)return;
        try {
            rs.last();
            pagination.setRowCount(rs.getRow());
            int row=pagination.getStartRow();
            if(row==0){
                rs.beforeFirst();
            }else{
                rs.absolute(row);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void setConn(Connection conn) {
```

```
            this.conn = conn;
        }
}
```

DAOFactory 类：

```
public class DAOFactory {
    public static DAO createDAO(Class<?> daoCalss,Connection conn){
        try {
            Object obj = daoCalss.newInstance();
            AbstractDAO dao=(AbstractDAO)obj;
            dao.setConn(conn);
            return dao;
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

Pagination类：

```
public class public class Pagination {
    private int rowCount;
    private int rowOfPage=15;
    private int curPage = 1;
    private int pageCount;

    public Pagination() {

    }
    public Pagination(int rowCount) {
        setRowCount(rowCount);
    }
    public Pagination(int rowCount, int rowOfPage) {
        setRowCount(rowCount);
        setRowOfPage(rowOfPage);
    }
    public void setRowCount(int rowCount){
        this.rowCount=rowCount;
        init();
    }

    public int getRowCount() {
        return rowCount;
    }
    public void setCurPage(int pageNum) {
```

```
        this.curPage = pageNum;
    }
    public void setRowOfPage(int rowOfPage) {
        this.rowOfPage = rowOfPage;
        init();
    }

    private void init(){
        pageCount=(int) java.lang.Math.ceil((double)rowCount / (double)rowOfPage);
    }
    public int getCurPage(){
        return curPage;
    }
    public int getPageCount(){
        return pageCount;
    }
    public int getFirst() {
        return 1;
    }

    public int getNext() {
        if (pageCount> curPage) {
            return curPage+1;
        } else {
            return pageCount;
        }
    }

    public int getPrevious() {
        if (curPage > 1){
            return curPage - 1;
        }
        return 1;
    }

    public int getLast() {
        return pageCount;
    }

    public int getStartRow() {
        return (this.curPage - 1) * this.rowOfPage;
    }

    public int getEndRow() {
        return curPage * this.rowOfPage;
```

```
    }
}
{
    private int rowCount;
    private int rowOfPage=15;
    private int curPage = 1;
    private int pageCount;

    public Pagination() {

    }
    public Pagination(int rowCount) {
        setRowCount(rowCount);
    }
    public Pagination(int rowCount, int rowOfPage) {
        setRowCount(rowCount);
        setRowOfPage(rowOfPage);
    }
    public void setRowCount(int rowCount){
        this.rowCount=rowCount;
        init();
    }

    public int getRowCount() {
        return rowCount;
    }
    public void setCurPage(int pageNum) {
        this.curPage = pageNum;
    }
    public void setRowOfPage(int rowOfPage) {
        this.rowOfPage = rowOfPage;
        init();
    }

    private void init(){
        pageCount=(int) java.lang.Math.ceil((double)rowCount / (double)rowOfPage);
    }
    public int getCurPage(){
        return curPage;
    }
    public int getPageCount(){
        return pageCount;
    }
    public int getFirst() {
        return 1;
```

```
    }

    public int getNext() {
        if (pageCount> curPage) {
            return curPage+1;
        } else {
            return pageCount;
        }
    }

    public int getPrevious() {
        if (curPage > 1){
            return curPage - 1;
        }
        return 1;
    }

    public int getLast() {
        return pageCount;
    }

    public int getStartRow() {
        return (this.curPage - 1) * this.rowOfPage;
    }

    public int getEndRow() {
        return curPage * this.rowOfPage;
    }
}
```

Controller层代码：

Action类：

```
public abstract class Action {
    protected final String SUCCESS="success";
    protected final String FAILURE="failure";
    protected ActionMessage message=new ActionMessage();
    protected Connection conn;
   protected abstract String execute(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException ;
    public String process(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        request.setAttribute("message", message);
        return execute(request, response);
    }
    public void setConn(Connection conn){
        this.conn=conn;
```

```
    }
    protected abstract DTO initDTO(HttpServletRequest request);
    protected abstract boolean validate(DTO dto);
}
```

ActionConfig类：

```
public class ActionConfig {


    private static ActionConfig actionConfig;
    private Map<String,ActionInfo> actionInfos=new HashMap<String, ActionInfo>();
    private ActionConfig(URL url) throws JDOMException, IOException{
        SAXBuilder builder=new SAXBuilder();
        Document doc = builder.build(url);
        List<Element> list = XPath.selectNodes(doc,"//action-config/action");
        Iterator<Element> iter=list.iterator();

        while(iter.hasNext()){
            Element actionElt=iter.next();
            ActionInfo actionInfo=getActionInfo(actionElt);

            actionInfos.put(actionInfo.getPath(), actionInfo);
        }


    }
    private ActionInfo getActionInfo(Element actionElt) {
        ActionInfo action=new ActionInfo();
        action.setPath(actionElt.getAttributeValue("path"));
        action.setActionClass(actionElt.getAttributeValue("actionClass"));

        List<Element> list = actionElt.getChildren();
        Iterator<Element> iter=list.iterator();

        while(iter.hasNext()){
            Element forwardElt=iter.next();
            ActionForward forward=getForward(forwardElt);
            action.addActionForward(forward);
        }
        return action;
    }
    private ActionForward getForward(Element forwardElt) {
        ActionForward actionForward=new ActionForward();
        actionForward.setName(forwardElt.getAttributeValue("name"));
        actionForward.setPath(forwardElt.getAttributeValue("path"));
```

```
        actionForward.setRedirect(Boolean.parseBoolean(forwardElt.getAttributeValue("redirect"))
);
            return actionForward;
        }
        public static void init(URL url) throws JDOMException, IOException{
            if(actionConfig==null){
                actionConfig=new ActionConfig(url);
            }
        }
        public static ActionInfo getActionInfo(String path){
            if(actionConfig==null){
                System.out.println("没有初始化");
            }
            return actionConfig.actionInfos.get(path);
        }
}
```

ActionServlet类：

```
public class ActionServlet extends HttpServlet {

    public void init() throws ServletException {
        // 根据参数去解析xml
        try {
            // 初始化action
            String config = this.getInitParameter("action-config");
            URL url = getServletContext().getResource(config);
            ActionConfig.init(url);
            // 初始化DataSource
            config = this.getInitParameter("db-config");
            url = getServletContext().getResource(config);
            DbConfig.init(url);
            // 初始化系统配置SysConfig
            config = this.getInitParameter("sys-config");
            url = getServletContext().getResource(config);
            SysConfig.init(url);

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException, ServletException {
        process(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws IOException, ServletException {
        process(request, response);
    }

    protected void process(HttpServletRequest request,
            HttpServletResponse response) throws IOException, ServletException {
        String path = parseURI(request);
        ActionInfo actionInfo = ActionConfig.getActionInfo(path);
        try {
            if (actionInfo == null) {
                throw new ActionException("不能找到“" + path + "”的Action映射关系");
            }
            Action action = ActionFactory.createAction(actionInfo);
            // 为Action创建数据连接对象Connection,以提供处理请求时使用，请求处理完
毕，关才连接
            Connection conn=null;
            String forward = null;
            try {
                conn = ConnFactory.getConn(SysConfig.getDbName());
                action.setConn(conn);
                forward = action.process(request, response);
            } catch (Exception e) {
                throw e;
            } finally {
                conn.close();
            }
            ActionForward actionForward = actionInfo.findActionForward(forward);
            if (actionForward == null) {
                throw new ActionException("在action:“" + path + "”中不能找到“"
                        + forward + "”的映射关系");
            }
            if (actionForward.getRedirect()) {
                response.sendRedirect(actionForward.getPath());
            } else {
                request.getRequestDispatcher(actionForward.getPath()).forward(
                        request, response);
            }

        } catch (ActionException e) {
```

```
                e.printStackTrace();
            } catch (Exception e) {
                request.getRequestDispatcher("error.jsp")
                        .forward(request, response);
                e.printStackTrace();
            }
        }
        private String parseURI(HttpServletRequest request) {
            String path = request.getRequestURI();
            String contextPath = request.getContextPath();
            if (path.startsWith(contextPath)) {
                path = path.substring(contextPath.length());
            }
            int end = path.lastIndexOf(".");
            path = path.substring(0, end);
            return path;
        }
        public void destroy() {
        }
}
```