|  |  |
| --- | --- |
| **密 级** |  |
| **学 号** | **2018322048** |



**毕 业 设 计（论 文）**

|  |
| --- |
| **（黑体一号字，加粗）** |
| **基于JAVA的中小企业人事管理系统设计与实现** |

|  |  |
| --- | --- |
| **院（系、部）：** | **（宋体三号、加粗）** |
| **姓 名：** | **袁健** |
| **年 级：** | **计G181** |
| **专 业：** | **计算机科学与技术** |
| **指导教师：** | **徐华** |
| **教师职称：** | **教授** |

年 月 日·北京

# 摘 要

由于历史和java虚拟机跨平台及其自身语法的相对安全性等诸多技术原因，java语言技术对于企业级别的应用开发具有低成本高稳定性的优点，至此，一些企业常用的后台管理系统类也都以java技术独占一档。人事管理系统对公司为实现各个部门之间资源共享，增强企业即时通讯，志在提高员工工作效率，简化繁琐的手工统计、信息汇总和工资业务等大量手工工作，让人事组织和工资管理工作在人事组织相关的各部门之间活跃起来，进而增强企业凝固力。本论文主体用SSM框架实现了对系统后台的设计制作，并追随当下中小企业流行的前后端分离技术，所以后端看不到任何前端的设计过程，前端界面的展示结合了Vue和Element技术，从而避免了传统的JSP、Thymeleaf、FreeMarker等诸多前端界面技术混合在后端导致项目整体庞大冗余不便于管理，这将为后续的相关人员对项目维护、责任划分和功能扩展提供了极大的便利性，有利于项目解耦。

**关键词**：即时通讯，SSM框架，Vue，Element，分离，解耦

# Abstract

java language has the advantages of low cost and high stability for enterprise-level application development. So far, some background management background management system classes commonly used in enterprises are also monopolized by java technology. Personnel management system for the company to achieve the sharing of resources between various departments, enhance enterprise instant messaging, in order to improve the efficiency of staff work, simplify cumbersome manual statistics, information collection and wage business, such as a large number of manual work, so that personnel organizations and wage management work in personnel organizations between the relevant departments, and then enhance the solidification of enterprises. The main body of this paper uses the SSM framework to design and manufacture the back-end of the system, and follows the popular front-end separation technology of small and medium-sized enterprises, so the back-end can not see any front-end design process. The display of the front-end interface combines Vue and Element technology.

**Keywords:** instant messaging, SSM framework, Vue,Element, separation, decoupling

# 目 录

[第一章 绪论 1](#_Toc42479490)

[1.1 人事管理系统背后的意义 1](#_Toc42479491)

[1.2 国内外研究现状及发展 1](#_Toc42479492)

[1.3 研究的内容 3](#_Toc42479492)

[1.4 本论文的立项背景 6](#_Toc42479492)

[第二章 实验材料与方法 8](#_Toc42479493)

[2.1 概述 8](#_Toc42479494)

[2.2 实验材料与方法 9](#_Toc42479495)

[第三章 实验结果与讨论 12](#_Toc42479497)

[3.1 持久层操作数据库 14](#_Toc42479498)

[3.1.1 接口类说明 14](#_Toc42479498)

[3.1.2 xml文件常见sql标签对 15](#_Toc42479498)

[3.1.3 数据库文件及模型设计 19](#_Toc42479498)

[3.2 服务层获取持久层返回值 2](#_Toc42479499)6

[3.3 控制层提供RESTful服务 27](#_Toc42479499)

[3.4 axios请求接口的研究 28](#_Toc42479499)

[第四章 结论与展望 3](#_Toc42479500)1

[4.1 结论 3](#_Toc42479501)1

[4.2 对进一步研究的展望 3](#_Toc42479502)1

[参考文献 3](#_Toc42479508)2

[致 谢 3](#_Toc42479509)3

[附 录 3](#_Toc42479509)4

# 第一章 绪 论

## 1.1 人事管理系统背后的意义

人事管理系统在企业后台管理系统中甚是常见，它的主要用途在于加强各个部门之间的联系和提高工作效益。通过人事管理系统，人事组织部门能做到一人为中心，各个部门实现资源共享，并实现即时通讯。所以通过人事管理系统，就实现各个部门之间资源共享，增强企业即时通讯，志在提高员工工作效率。简化繁琐的手工统计、信息汇总和工资业务等大量手工工作，让人事组织和工资管理工作在人事组织相关的各部门之间活跃起来,通过更加专业化的管理系统来调动企业运用,为企事业单位提高生产力和解放传统员工的手工统计。

1.2 国内外研究现状及发展

生产力背后都伴随信息的交换，一直都伴随着人们社会发展不可或缺，信息作为企业经营管理系统中发挥生产效益的是在计算机发明之后的将近60年。

现代企业中的管理信息系统通常是将实现编好的程序存入计算机运行，通过计算机对信息进行采集、存储、索引、修改和传递，使相应信息为企业机构的生产提供帮助，该系统是上世纪70年代初工业经济时代的产物，在企业信息管理系统起步时期，只是简单的人机交互系统，不涉及发达的网络系统，流动性差，系统功能简略，对生产产生效益很有限。

步入上世纪80代末期，伴随着网络和通信技术的飞进，企业信息管理系统渐渐进入网络的发展时期，随后以美国为首的发达国家已经初步建立了完善的管理信息系统体系，对其社会经济发展展现了不可或缺的作用。

中国的管理信息系统大发展是在上世纪八九十年代，国家先后建立起了“金卡”、“金关”、“金桥”工程，因此多数企业单位建立起了局域网和广域网管理信息系统。伴随着时代的发展，无论是企业内部的管理信息系统，还是指向少数专家学者和专业管理人员的专家管理系统或决策支持系统，其解决的目标和服务的目标，自身的系统架构和解决问题的能力，都有了极大的跟进。

人事信息管理系统的发展史可以推前至上世纪60年代末期。在计算机技术已经有了生活实用性的时代，同期的大型企业单位也有幸抛弃了传统手工计算发放薪水这一既费时费力和易出错的弊端，进而转向了计算机人事管理系统。当然，由于当时网络技术和计算机编程语言技术的限制，用户也非常少，再者，那时的人事管理系统只不过是一种计算薪水的工具罢了，很少包含一些非财务的信息，更不包含历史账单信息，也没有公司报表生成功能和薪水数据分析能力。但是，即便如此，它的诞生和演进也为后期人事管理系统的美好发展提供了坚定的基础，为后期通过计算机的高运算力高准确度来代替企业巨大的手工信息统计提供了至关重要的贡献，使集中处理大规模业务能力变为可能。

二代人事管理系统的出现是在10年后。伴随计算机技术的高速发展和计算机的普及性，也依托计算机的系统工具和数据库技术的发展，都为现代人事管理系统的的完善提供了阶段性的成长。第二代人事管理系统基本解决了上一代的主要弊端，二代考虑了非财务功能和薪水历史账单记录，其生成报表和薪水数据分析能力都有了极大的改善。但这一代系统大多以大企业为主且由计算机专业人员研制和管理的，离现代的人事系统的需求和理念还相差甚远，主要体现在非财务的人事管理信息不够全面和完善。

人事管理系统的革命性发展出现在上世纪90年代末期。伴随市场竞争需求，该如何吸引留住人才，激发企业员工创造性、责任感和工作激情已然成为企业长久发展之计。“公正、公平、合理”的企业内部管理方式和经营水平的提高，使社会对人事管理系统有了更高的需求和依赖，同时个人计算机得到普及，数据库技术、web技术的发展，第三代人事管理系统渐渐出现。三代人事管理系统的特点是从人事管理角度出发，通过数据库技术将企业内部的各类信息（比如招聘人员信息、培训、职称、职位、奖惩等等）统一管理起来，进而形成了信息源汇总。其次，友好的用户界面，强大的报表生成和报表分析工具、信息共享是的企业hr等人事管理人员得以从繁重的日常工作中解脱，使他们更多的精力放在考虑人事管理规划和企业内部政策优化。

到了现代，企业内部的人事管理系统功能更完备丰富。有些内部人事管理系统不再仅局限在内部人员使用，也开放了外部人员信息注册查看修改功能。就拿医院的挂号系统来讲，不再需要患者亲自去医院找医生挂号这种一问一答式填写患者信息，这是以前想都不敢想的。背后的关键还是得益于互联网技术和计算机编程语言java的强大安全。由于国内人口众多，所以涉及的数据库技术要求更高，举个很现实的技术案例，目前市场上用到的数据库框架有hibernate和mybatis，国内偏爱mybatis，而国外更偏好hibernate，为何如此，原因就在于市场需求，国内人口众多，对数据库技术优化技术要求高而国外人口相对少很多，所以老外更偏好不需要手动敲代码优化数据库，故他们选择集成度更高的hibernate。由上可见，人事管理系统中，人员信息的多少也决定背后使用的技术的差异，这是国内外现状的一个重要体现。

1.3 研究的内容

人事管理系统中，如何对用户信息进行存储提取修改删除，如何搭建项目架构，业务的运行流程如何，如何减轻对数据的查询负担，前端用户界面如何展示，前端如何访问后端？接下来一一叙述解答。

首先，不论你研发什么系统，用户的信息永远都是系统存在的根基，系统的一切功能和特性都是围绕着用户的信息展开而来的。在本次毕设项目作业中，我对用户信息的操作用的相关技术是mybatis框架技术，故，谈到对用户信息的存储提取修改删除自然离不开mybatis数据库技术。传统的jdbc在操作数据库时候都是跟Java代码混合在一起写的，导致后起很难阅读和维护，所以一些数据库框架应用而生，其中就包括mybatis。

在springboot整合mybatis技术未诞生前，传统的在用mybatis操作数据库之前都先配置一下mybatis的配置文件，配置文件是mybatis和数据库建立连接的核心文件，除此之外，配置文件还包括了mapper映射文件的有关声明及其别名定义等功能，这个配置文件是整个mybatis的全局配置文件。有了配置文件，接下来要准备的XML配置文件就与后面要执行的SQL语句有关系了。在mybatis框架中，SQL语句与java类并无直接联系，几乎所有的的SQL语句都配置在mapper映射的XML配置文件中，在整个mybatis的运行机制中，会在适当的时候加载相应的配置文件，读取其中的SQL信息及输入输出相关参数信息。有关该XML配置文件的SQL语句配置有insert、delete、update、select的标签对，其中这些标签对都有parameterType、parameterMap、resultType、resultClass、resultMap等相关属性，这些属性分别代表了输入参数类型、输入参数集合、结果类型、结果类、结果集合。有了以上两个配置文件，接下来就可以进行相关Java类的开发，其中涉及到mybatis的一个至关重要的类SqlSession，都过此类的对象方法，比如selectOne,我们可以查询相应数据中表的信息。而现在有了springboot整合了mybatis框架之后，上述这些传统的配置得到了进一步简化，在springboot整合mybatis框架中，我们只需要提供一个java接口类和以这个Java接口类同样名称的XML文件就可以完成对数据库的操作，从而避免了大量与业务本身不相关的参数配置。

那么问题来了，为什么一个java接口类和一个XML文件就能够完成对数据库的完美操作呢？这也是本次我在项目实战中需要研究和学习的，有关这方面的理解如下：mybatis之所以通过接口和XML文件就能够操作数据库，底层实现原理是通过java动态代理模式，所以理解了动态代理模式也就对mybatis底层实现机制有了了解。那么什么又是动态代理模式呢？代理模式是常用的java设计模式，他的特征是代理类与委托类有同样的接口，代理类主要负责为委托类预处理消息、过滤消息、把消息转发给委托类，以及事后处理消息等。代理类与委托类之间通常会存在关联关系，一个代理类的对象与一个委托类的对象关联，代理类的对象本身并不真正实现服务，而是通过调用委托类的对象的相关方法，来提供特定的服务。简单的说就是，我们在访问实际对象时，是通过代理对象来访问的，代理模式就是在访问实际对象时引入一定程度的间接性，因为这种间接性，可以附加多种用途。举一个更简单的生活场景，客户需要一款软件，所以客户就找到软件公司项目负责人，告诉他的软件功能需求，软件公司的项目负责人在得知顾客的需求后在告知各个部门的软件开发人员，有软件开发人员负责生产软件，所以，软件公司项目负责人就起到了代理作用，由他去代替软件客户向开发人员传递客户的需求。再回到mybatis中的场景，在java的java.lang.reflect包下提供了一个Proxy类和一个InvocationHandler接口，通过这个类和这个接口可以生成JDK动态代理类和动态代理对象，这就是mybatis中提供接口类的主要目的，有了这个接口类的引用可以动态的指向它的子类，而通过XML解析器可以解析出mybatis中XML文件的信息，其中就包含重要的SQL语句，都过配置信息中的标签对和标签对中的属性，就能够实现将SQL脚本与相应的JAVA实体类相关联，进而达到操作数据库的目的。

关于研究的另一个问题是项目的架构该如何搭建，由于本项目是完全前后端分离项目，所以就分开来谈各自的项目框架。首先后端方面，通过springboot整合SSM框架，项目用到的各种类注入到spring中，由spring容器来帮助我们统一管理，只要项目用到这个类就可以随时随地的注入进来即可，其次，对于数据库的操作用到了mybatis框架技术，也很方便，最后通过类似springmvc建构来搭建后台接口，供前端调用。前端方面，通过当下中小型公司比较青睐的vue.js技术架构实现页面跳转和后端接口访问，具体的页面展示和渲染技术就交由Element标签库统一生成[4,6]。这些主要就是项目生成的架构。

有了上面描述的项目的整体架构，项目的运行流程也就好理解了。首先，后端开发好接口，接口是RESTful风格的，前端方面通过vue.js的axios发送类似Ajax的各类请求，后端控制层收到不同的请求后就分别调用服务层的逻辑，由服务层的判断逻辑来决定找到mybayis框架的那一条操作数据库的语句，最后将json数据返回给控制层，再交由控制层直接返回给前端，前端获取到来自后端控制层的json数据，再交由vue和element共同完成对数据的展示和渲染，这就是大致的业务运行流程。

另一个研究的内容是关于如何减轻项目对数据库频繁访问的负担，一个最简单的方式就是搭建redis环境，可以通过redis缓存技术来避免相同的操作多次访问数据库，进而影响数据库的性能。具体可以这么做，首先我们可以将redis安装在linux操作系统上，并完成相应的配置。然后在Windows开发环境的项目的配置文件中配置相关参数，比如redis服务器的IP地址，连接redis的用户名和密码等等。当这些配置我们都设置好以后，我们在项目的逻辑服务层的需要减轻对数据库操作的方法之上加上@Cacheable注解，这样，下次同样的操作不会直接从数据中的进行查取，而是直接从redis缓存中读取[1,8]。

关于前端页面展示的研究内容，首先要考虑使用的就是目前前端开发人员使用最多的Element元素标签库，这样以后出现问题也方便查找。Element官网为我们提供了很多常用的界面组件和布局方式，我们只需要阅读官网学习相关组建的学习使用还是很简单的。下面通过几个简单的例子展示界面的效果图：

图标按钮

带图标的按钮可增强辨识度（有文字）或节省空间（无文字）。



图1.1 按钮--效果（来源Element官网）

设置icon属性即可，icon 的列表可以参考 Element 的 icon 组件，也可以设置在文字右边的 icon ，只要使用i标签即可，可以使用自定义图标。

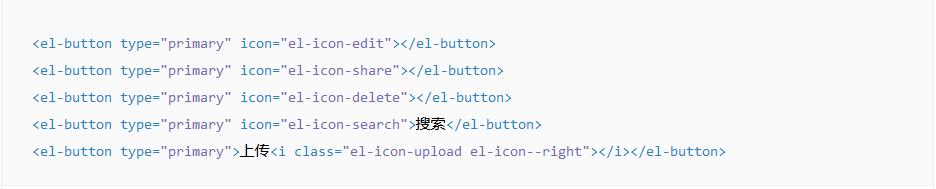


图1.2 按钮--代码（来源Element官网）

行内表单

当垂直方向空间受限且表单较简单时，可以在一行内放置表单。



图1.3 表单--效果（来源Element官网）

设置 inline 属性可以让表单域变为行内的表单域

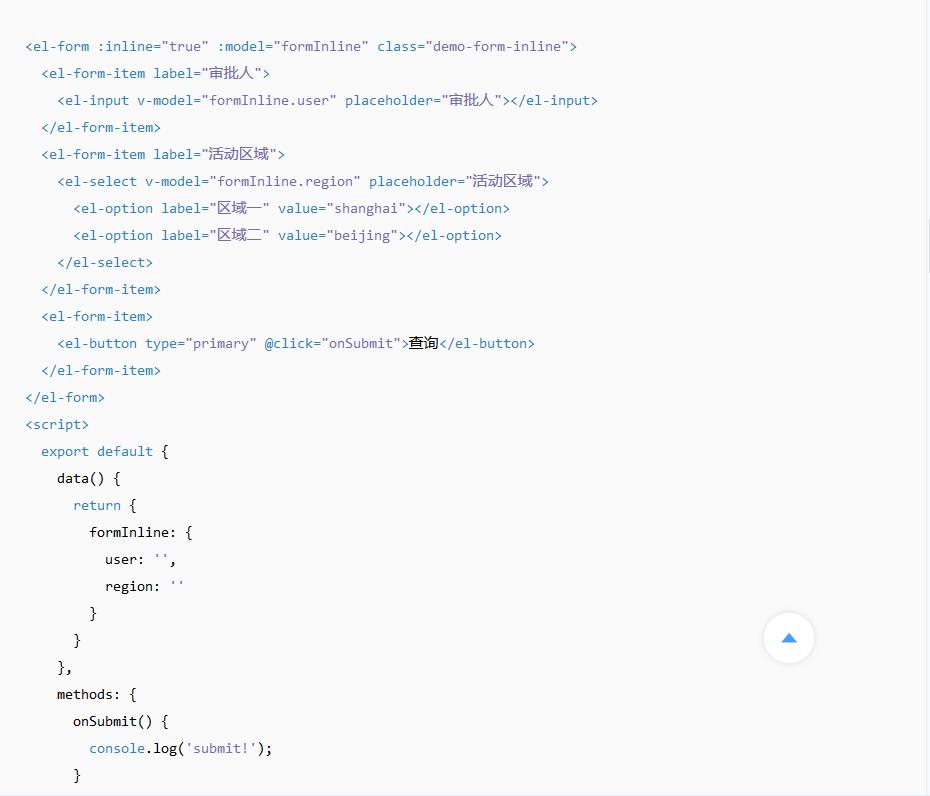


图1.4 表单--代码（来源Element官网）

通过以上这两个例子，前端姐买你展示也就一幕了然了。

最后一个问题是前端该如何访问后端，由于vue不支持直接发送AJAX请求，需要使用vue-resource、axios等插件实现，在node.js上实现请求转发,vue前端通过axios请求到node.js上,node.js将请求转发到后端,反之.响应也是,先到node.js上,然后转发vue-cil项目上。需要在根目录下新建文件vue.config.js。vue.config.js 是一个可选的配置文件，如果项目的 (和 package.json 同级的) 根目录中存在这个文件，那么它会被 @vue/cli-service 自动加载。你也可以使用 package.json 中的 vue 字段，但是注意这种写法需要你严格遵照 JSON 的格式来写。

1.4 本论文的立项背景

人事系统在企业后台管理系统中甚是常见，它的主要用途在于加强各个部门之间的联系和提高工作效益，通过人事管理系统，人事组织部门能做到一人为中心，各个部门实现资源共享人事管理，并实现即时通讯。所以通过人事管理系统，就能够把分散的企业单位的职工信息进行统一收集管理,通过更加专业化的管理系统来调动企业运用,为企事业单位提高生产力和解放传统员工的手工统计。

综上所述，开发出一款功能完备的人事系统，体现出了自己从事编程行业以来能符合企业实际利益，能为企业带来实际效益的，是对自身能力的体现。

# 第二章 实验材料与方法

## 2.1 概述

本论文主要从事人事管理系统的开发，使用当前备受中小企业欢迎的前后端分离技术来开发出满足企业人事管理实际需求的业务功能。

现代JAVA EE的企业级应用更加偏向于简化配置，一方面，后端框架Spring Boot作为后起之秀，带来了全新的自动化配置解决方案，使用Spring Boot可以快速创建基于Spring生产级的独立应用程序，简化了程序员对于SSM框架的整合配置，做到快速部署运行，降低了开发工作量[1,5]；另一方面，前端Vue技术的兴起，与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动[2,,11,14]。

内容主要是后端开发增删改查接口，用来供前端通过axios发送各类请求来调用后端开发的接口。

按照前后端分离可分为以下几个方面：

后端开发如下：

（1）、实体类：当数据库的各张表全部设计完毕后，根据每张数据表分别设计一个与之对应表的各个字段的成员变量，这里需要注意一点是，Java类中所有成员变量的取名最好跟数据表中的字段名称完全一致，或者也可以通过成员变量的驼峰命名法来匹配数据表的字段名称，这里举个例子，比如有数据表字段名user\_ID，那么成员变量命名必须为驼峰命名的userID[2]；

（2）、持久层：更具业务需求为数据库中经常用到的几张表设计对应的SQL脚本语句，结合MyBatis持久层框架编写SQL脚本，一般情况下，通过MyBatis框架编写持久层，为了迎合MyBatis框架的底层实现原理动态代理，都会为提供相应的接口类，SQL查询脚本的编写都会依据接口类中方法[2,7,14]；

（3）、接口层：根据业务需求编写接口方法调用持久层中设计好的增删改查方法，并将返回值保存至相应实体类中，接口层通常包含接口类和它的实现类，之所以这么分层，为了以后项目的扩展和解耦[1,2,12]；

（4）、控制层：根据业务需求开发相关接口，接口中的路径用来提供给前端请求，并将请求的JSON数据返回给前端，这里的JSON数据是前后端分离项目中数据交换最常用的数据格式[1,2]；

（5）、配置层：为Spring Boot和Spring框架提供基本的配置，比如用户页面访问权限的配置Spring Security，用于消息点对点发送的WebSocket的配置，登陆系统时验证码的相关配置等等[1,5]。

前端开发如下[4,6,10,11]：

（1）、template组件：此模块供页面基本的组建的搭建，这里的组件可以是Element组件，Element官方为开发者提供了丰富的交互式界面控件，开发者直接在Vue中映入Element库即可使用各种控件；

（2）、script脚本：此模块用于页面控件的逻辑实现，由于在Vue中引入了Node.js库，用Node.js提供的丰富的js脚本库，可以通过axios向后端发送各种请求，比如get、post请求等等，在向后端发送各种请求后会对返回的数据分析清洗存储，由于Vue的数据是绑定式的，当此模块中数据更新了，template模块中组件用到的数据也会自动更新，相比传统jsp而言，少了很多dom节点的操作，所以，前端开发，这个模块是重点难点；

（3）、style样式：此模块为template模块的各种组件提供在页面中的完美布局和组件自身本身的系列属性设置，样式的好坏直接关系的用的用户体验，所以一般由专业人事提供页面布局个管控。

2.2 实验材料与方法

由于是本项目是前后端分离项目，故开发环境也是前后端分离的。在开发之前，电脑预装的系统环境如下：

前端

下载安装WebStorm、nodeJS并配置本地仓储，通过vue-cli脚手架构建前端项目，其中Windows的CMD关键命令如下：

npm install -g vue-cli

vue init webpack veuhr

然后再webstorm中打开并开发前端项目

后端

下载安装VMware，MySQL，intlliJ IDEA，配置好系统的Java运行环境，并在VMware中安装Linux操作系统centos7，之后在centos7中安装redis和rabbitMQ，有关部分rabbitMQ安装的linux命令行命令如下：

安装依赖

yum -y install gcc glibc-devel make ncurses-devel openssl-devel xmlto perl wget gtk2-devel binutils-devel

下载erlang

wget http://erlang.org/download/otp\_src\_22.0.tar.gz

解压

tar -zxvf otp\_src\_22.0.tar.gz

移走

mv otp\_src\_22.0 /usr/local/

切换目录

cd /usr/local/otp\_src\_22.0/

创建即将安装的目录

mkdir ../erlang

配置安装路径

./configure --prefix=/usr/local/erlang

Rabbitmq下载

wget <https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.7.15/>

rabbitmq-server-generic-unix-3.7.15.tar.xz

由于是tar.xz格式的所以需要用到xz，没有的话就先安装

yum install -y xz

第一次解压

/bin/xz -d rabbitmq-server-generic-unix-3.7.15.tar.xz

第二次解压

tar -xvf rabbitmq-server-generic-unix-3.7.15.tar

移走

mv rabbitmq\_server-3.7.15/ /usr/local/

改名

mv /usr/local/rabbitmq\_server-3.7.15 rabbitmq

配置环境变量

echo 'export PATH=$PATH:/usr/local/rabbitmq/sbin' >> /etc/profile

刷新环境变量

source /etc/profile

创建配置目录

mkdir /etc/rabbitmq

之后就可以启动、停止

启动：

rabbitmq-server -detached

停止：

rabbitmqctl stop

至此，人事管理系统的项目整体所需实验材料和环境就算搭建完毕。

第三章 实验结果与讨论

## 3.1 持久层操作数据库

由于系统是是基于SSM框架开发的，所以在操作数据库时主要技术用到了mybatis技术，通过它编码来操作MySQL数据库，有关mybatis的使用，我们只需要提供数据库中各章表的Java实体类的相应接口和一个与接口同名的XML格式文件就可以完成基本的mybatis操作数据库的功能架构了。下面分别对接口和XML文件的细节进行解读。

3.1.1 接口类说明

在有关mybayis环境搭建的时候我们知道了SqlSessionFactory，既然有了 SqlSessionFactory ，我们就可以通过该类的静态方法从中获得 SqlSession 的对象，SqlSession 包含了面向数据库执行 SQL 命令所需的所有方法，环境搭建中的示例我们是通过 SqlSession 实例来直接执行已映射的 SQL 语句，尽管这种方式能够正常操作数据库，给对使用旧版本的 MyBatis 老用户来说也比较熟悉，可是随着mtbatis版本的更新和优化，现在有了一种更直白简单的操作数据库的方式，使用对于给定语句能够合理描述参数和返回值的接口（比如说EmployeeMapper.class），你现在不但可以执行更清晰和类型安全的代码，而且还不用担心易错的字符串字面值以及强制类型转换[2,14]。

以本项目人事管理环境搭建中的EmployeeMapper.xml映射文件为例我们来创建一下它所对应的接口。下面给出Employee实体类，映射文件EmployeeMapper.xml，即将要绑定的接口EmployeeMapper.java以及调用方法，部分源码如下给出：

public class Employee implements Serializable *{* private Integer id;  
  
 private String name;  
  
 private String gender;  
  
 @Override  
 public String toString*() {* return "Employee{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", gender='" + gender + '\'' +  
 ", birthday=" + birthday +  
 ", idCard='" + idCard + '\'' +  
 ", wedlock='" + wedlock + '\'' +  
 ", nationId=" + nationId +  
 ", nativePlace='" + nativePlace + '\'' +  
 ", politicId=" + politicId +  
 ", email='" + email + '\'' +  
 ", phone='" + phone + '\'' +  
 ", address='" + address + '\'' +  
 ", departmentId=" + departmentId +  
 ", jobLevelId=" + jobLevelId +  
 ", posId=" + posId +  
 ", engageForm='" + engageForm + '\'' +  
 ", tiptopDegree='" + tiptopDegree + '\'' +  
 ", specialty='" + specialty + '\'' +  
 ", school='" + school + '\'' +  
 ", beginDate=" + beginDate +  
 ", workState='" + workState + '\'' +  
 ", workID='" + workID + '\'' +  
 ", contractTerm=" + contractTerm +  
 ", conversionTime=" + conversionTime +  
 ", notWorkDate=" + notWorkDate +  
 ", beginContract=" + beginContract +  
 ", endContract=" + endContract +  
 ", workAge=" + workAge +  
 ", nation=" + nation +  
 ", politicsstatus=" + politicsstatus +  
 ", department=" + department +  
 ", jobLevel=" + jobLevel +  
 ", position=" + position +  
 '}';  
 *}*

public interface EmployeeMapper *{* int deleteByPrimaryKey*(*Integer id*)*;  
  
 int insert*(*Employee record*)*;  
  
 int insertSelective*(*Employee record*)*;  
  
 Employee selectByPrimaryKey*(*Integer id*)*;  
  
 int updateByPrimaryKeySelective*(*Employee record*)*;  
  
 int updateByPrimaryKey*(*Employee record*)*;  
  
 List*<*Employee*>* getEmployeeByPage*(*@Param*(*"page"*)* Integer page, @Param*(*"size"*)* Integer size, @Param*(*"emp"*)* Employee employee,@Param*(* "beginDateScope"*)* Date*[]* beginDateScope/\*,@Param("name") String name\*/*)*;  
  
 Long getTotal*(*@Param*(*"emp"*)* Employee employee,@Param*(*"beginDateScope"*)* Date*[]* beginDateScope*)*;  
  
 Integer maxWorkID*()*;  
  
 Integer addEmps*(*@Param*(*"list"*)* List*<*Employee*>* list*)*;  
  
 Employee getEmployeeById*(*Integer id*)*;  
  
 List*<*Employee*>* getEmployeeByPageWithSalary*(*@Param*(*"page"*)* Integer page, @Param*(*"size"*)* Integer size*)*;  
  
 Integer updateEmployeeSalaryById*(*@Param*(*"eid"*)* Integer eid, @Param*(*"sid"*)* Integer sid*)*;

*<?*xml version="1.0" encoding="UTF-8" *?>  
<!DOCTYPE* mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" *>  
<*mapper namespace="org.javaboy.vhr.mapper.EmployeeMapper"*>  
 <*resultMap id="BaseResultMap" type="org.javaboy.vhr.model.Employee"*>  
 <*id column="id" property="id" jdbcType="INTEGER"*/>  
 <*result column="name" property="name" jdbcType="VARCHAR"*/>  
 <*result column="gender" property="gender" jdbcType="CHAR"*/>  
 <*result column="birthday" property="birthday" jdbcType="DATE"*/>  
 <*result column="idCard" property="idCard" jdbcType="CHAR"*/>  
 <*result column="wedlock" property="wedlock" jdbcType="CHAR"*/>  
 <*result column="nationId" property="nationId" jdbcType="INTEGER"*/>  
 <*result column="nativePlace" property="nativePlace" jdbcType="VARCHAR"*/>  
 <*result column="politicId" property="politicId" jdbcType="INTEGER"*/>  
 <*result column="email" property="email" jdbcType="VARCHAR"*/>  
 <*result column="phone" property="phone" jdbcType="VARCHAR"*/>  
 <*result column="address" property="address" jdbcType="VARCHAR"*/>  
 <*result column="departmentId" property="departmentId" jdbcType="INTEGER"*/>  
 <*result column="jobLevelId" property="jobLevelId" jdbcType="INTEGER"*/>  
 <*result column="posId" property="posId" jdbcType="INTEGER"*/>  
 <*result column="engageForm" property="engageForm" jdbcType="VARCHAR"*/>  
 <*result column="tiptopDegree" property="tiptopDegree" jdbcType="CHAR"*/>  
 <*result column="specialty" property="specialty" jdbcType="VARCHAR"*/>  
 <*result column="school" property="school" jdbcType="VARCHAR"*/>  
 <*result column="beginDate" property="beginDate" jdbcType="DATE"*/>  
 <*result column="workState" property="workState" jdbcType="CHAR"*/>  
 <*result column="workID" property="workID" jdbcType="CHAR"*/>  
 <*result column="contractTerm" property="contractTerm" jdbcType="DOUBLE"*/>*

上面示例就是接口绑定的过程和调用方法，通过本次系统开发，我发现除了调用跟之前不太一样，其他并没有什么改变，那么MyBatis是如何绑定的呢？我们来看一下EmployeeMapper映射文件中的<mapper namespace="org.javaboy.vhr.mapper.EmployeeMapper">标签，当要绑定接口时就需要其中的namespace指向要绑定的接口的全路径。就是这么简单，你只需要将namespace指向要绑定的类其他事MyBatis全都帮你搞定。调用时sqlSession.getMapper(EmployeeMapper.class)然后赋值给刚才我们写的接口，然后就可以调用我们借口的方法了，这里你可能会产生疑问我们并没有写EmployeeMapper接口的实现类啊，EmployeeMapper接口是怎么实例化的啊？我们来大胆的猜一下这里可能是MyBatis的一个代理实现。也就是前面章节中描述的mybatis底层操作数据的原理-动态代理技术。

好，以上通过本次人事管理系统的一个项目案例我了解了mybatis为什么要开发人员提供一个相应数据表的Java类的接口的缘由。

3.1.2 xml文件常见sql标签对

Mybatis的另一个重要的文件，也就是XML文件的作用，提到这个文件，就不得不比较一下hibernate与mybatis的使用场景了，正如第一章1.2小节所描述的mybatis国内外使用现状的差异，之所以mybatis能够的到国人青睐，正是这个XML文件提供的灵活优化SQL语句的功能，才使得数据库在面对大用户量访问时经得住考验，这是选择用mybatis代替hibernate操作数据库的最为至关重要的缘由，在本次的人事管理系统的项目实战开发中，有关XML文件的编码大概占据了我整个项目用时的80%左右，从上一小节提供的XML案例代码也可以看到它是整个项目最为缜密的部分，毕竟有关复杂SQL语句的编写都是在这里完成的，所以这里，我很有必要花多一些篇幅来介绍XML文件中最为常用的一些SQL标签对及其包含属性的使用方式和从中得到的心得体会。

·sql – 可被其他语句引用的可重用语句块：

*<*sql id="Base\_Column\_List"*>* id, name, gender, birthday, idCard, wedlock, nationId, nativePlace, politicId, email,  
 phone, address, departmentId, jobLevelId, posId, engageForm, tiptopDegree, specialty,  
 school, beginDate, workState, workID, contractTerm, conversionTime, notWorkDate,  
 beginContract, endContract, workAge  
*</*sql*>*

·insert – 映射插入语句 ：

*<*insert id="updateEmployeeSalaryById"*>* REPLACE INTO empsalary *(*eid,sid*)* VALUES*(#{*eid*}*,*#{*sid*})  
</*insert*>*

·update – 映射更新语句

*<*update id="updateByPrimaryKey" parameterType="org.javaboy.vhr.model.Employee"*>*update employee  
set name = *#{*name,jdbcType=VARCHAR*}*,  
 gender = *#{*gender,jdbcType=CHAR*}*,

·delete – 映射删除语句

*<*delete id="deleteByPrimaryKey" parameterType="java.lang.Integer"*>* delete from employee  
 where id = *#{*id,jdbcType=INTEGER*}  
</*delete*>*

·select – 映射查询语句

*<*select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Integer"*>* select  
 *<*include refid="Base\_Column\_List"*/>* from employee  
 where id = *#{*id,jdbcType=INTEGER*}  
</*select*>*

·if

*<*select id="getTotal" resultType="java.lang.Long"*>* select *count(*\**)* from employee e  
 *<*where*>  
 <*if test="emp!=null"*>  
 <*if test="emp.name !=null and emp.name!=''"*>* and e.name like concat*(*'%',*#{*emp.name*}*,'%'*)  
 </*if*>*

·choose (when, otherwise)

*<select id="findUser"*

*resultType="User">*

*SELECT \* FROM User WHERE age = 26*

*<choose>*

*<when test="name!= null">*

*AND name like #{name}*

*</when>*

*<when test="sex!= null ">*

*AND sex like #{sex}*

*</when>*

*<otherwise>*

*AND valid = 1*

*</otherwise>*

*</choose>*

*</select>*

·trim (where, set)

*<*trim prefix="(" suffix=")" suffixOverrides=","*>  
 <*if test="id != null"*>* id,  
 *</*if*>  
 <*if test="name != null"*>* name,  
 *</*if*>  
 <*if test="gender != null"*>* gender,

·foreach

*<*foreach collection="list" separator="," item="emp"*>  
 (#{*emp.name,jdbcType=VARCHAR*}*, *#{*emp.gender,jdbcType=CHAR*}*,  
 *#{*emp.birthday,jdbcType=DATE*}*, *#{*emp.idCard,jdbcType=CHAR*}*, *#{*emp.wedlock,jdbcType=CHAR*}*,  
 *#{*emp.nationId,jdbcType=INTEGER*}*,  
 *#{*emp.nativePlace,jdbcType=VARCHAR*}*, *#{*emp.politicId,jdbcType=INTEGER*}*, *#{*emp.email,jdbcType=VARCHAR*}*,  
 *#{*emp.phone,jdbcType=VARCHAR*}*, *#{*emp.address,jdbcType=VARCHAR*}*, *#{*emp.departmentId,jdbcType=INTEGER*}*,  
 *#{*emp.jobLevelId,jdbcType=INTEGER*}*, *#{*emp.posId,jdbcType=INTEGER*}*, *#{*emp.engageForm,jdbcType=VARCHAR*}*,  
 *#{*emp.tiptopDegree,jdbcType=CHAR*}*, *#{*emp.specialty,jdbcType=VARCHAR*}*, *#{*emp.school,jdbcType=VARCHAR*}*,  
 *#{*emp.beginDate,jdbcType=DATE*}*, *#{*emp.workState,jdbcType=CHAR*}*, *#{*emp.workID,jdbcType=CHAR*}*,  
 *#{*emp.contractTerm,jdbcType=DOUBLE*}*, *#{*emp.conversionTime,jdbcType=DATE*}*, *#{*emp.notWorkDate,jdbcType=DATE*}*,  
 *#{*emp.beginContract,jdbcType=DATE*}*, *#{*emp.endContract,jdbcType=DATE*}*, *#{*emp.workAge,jdbcType=INTEGER*}  
 )  
</*foreach*>*

以上代码部分主要展示了XML文件中常用的标签对在本项目中使用的场景，这些都是学习mybatis中需要熟练掌握的关键部分。下面再详细介绍这些标签中都有哪些属性和其使用场景。

configuration：这个标签代表了这是mybatis的顶级配置标签，由它包裹所有配置标签；

settings：全局配置参数，用来配置一些mybatis的参数，例如是否使用错误处理机制，是否启用命名空间，例如：开启Mybatis支持延迟加载,name属性：“lazyLoadingEnabled” 属性value值：true，例如：属性name值：“cacheEnabled” ，属性value值："true"开启mybatis的二级缓存功能；

environments：环境集合属性对象，时数据库环境的集合。配置数据库环境，可以有默认值default。在一个配置文件中，可以有多种数据库环境集合，这样可以使mybatis将SQL同时映射至多个数据库中；

environment：环境子属性对象，具体的一个数据库环境，以mysql为例；

transactionManager：事务管理，指定mybatis的食物管理器，我们一般是使用JDBC；

dataSource：配置数据源，数据库的数据源一般有三类POOLED、UNPOOLED、JNDI，mybatis框架内部自己依照池的原理实现了自己的连接池，所以一般选用POOLED。再使用其中的type属性指定数据源的连接类型，在标签对中可以使用property属性指定数据库连接池的其他信息；

property：这个标签是配置参数，比较常见，有name、value属性，value可使用el表达式；

properties：可用来引入外部配置文件，有resource（）引当前类路径下文件）和url（以文件协议引入外部文件）两个属性，该标签可以引入外部的配置好的属性，也可以自己设置属性，该配置标签所在的同一个配置文件中的其他配置均可引用此配置中的属性；

mappers：映射器，指定映射配置文件的位置，用来告知MyBatis去哪里加载SQL映射配置，映射配置文件指的是每个dao独立的配置文件 ,属性是resource时表示对应接口使用的是xml配置，属性是class时表示对应接口使用的是注解配置；

typeAliases:类型的别名，使用typeAliases配置别名，他只能配置domain中类的别名，用来设置一些别名替代Java长类型声明（比如：java.lang.int变为int），奸商配置编码的冗余；

typeAlias:typeAliases的子标签，用于配置别名，type属性用来指明实体类全限定类名，alias属性指名别名；

package：用于指明要配置的包的别称，在指明之后该包下的实体类都会被注册别名，之后类名就是别名，且使用时不必再区分大小写；

mapper ：一个接口对应的配置文件，属性namespace说明了是哪个具体的接口

resultMap ：配置查询结果的列名和实体类属性名对应关系。可选属性，该结果集配置对应SQL结果中的每个字段名称，即将映射到java对象中的哪个属性。此属性不能与resultType同时使用；

id : 是resultMap的子标签，用于主键。SQL映射配置的唯一标识，可以代表SQL配置；

result : resultMap的子标签，代表普通属性列；

collection : resultMap的子标签，用于一个实体类中包含另一个实体而且可能是多个的情况，一般用于表与表直接是一对多，多对多关系的情况；

association : resultMap的子标签，用于一个实体类中包含另一个实体而且该实体对象是一个的情况，一般用于表与表直接是一对一，多对一关系的情况

constructor : resultMap的子标签，不常用；

discriminator : 鉴别器，resultMap的子标签，不常用，例如：男女体检时由于体检的项目有所不同，所以在表的设计时会设计一个鉴别器；

cache：表示这个Mapper对应的接口是支持二级缓存的；

select、update、delete、insert：代表数据库查询、更新、删除、插入；

id：CRUD的属性对应接口中的那个方法，和mapper 的namespace值组合在一起，mybatis的代理对象就知道找个方法；

parameterType：参数类型，可选，如果是已经在主配置文件中的package 标签配置了这个实体类型的包名，那么就不用关心大小写问题了，用来传入SQL配置中需要的参数类型的类名或别名；

resultType：返回结果类型，可选属性。用来配置SQL语句执行后期望得到的结果数据类型，配置的是结果类型的类名或别名。此属性不能与resultMap同时使用；

useCache：该条sql语句支持二级缓存功能，boolean类型，设置语句调用时，执行结果是否保存二级缓存，对select元素默认为false，resultMap：可以使用定义的；

selectKey ： select last\_insert\_id()语句的专属标签，可以用来为默认增长主键赋值。

keyproperty：selectKey 的属性，对应实体类的属性名

keycolumn：selectKey 的属性，对应数据库表中的列名

order：selectKey 的属性，设置执行优先级

3.1.3 数据库文件及模型设计

数据模型侧重点是如何全面正确地反映真实实际情况，它并不会关心数据库中是如何实现的。这种模式能够真实的反应用户的真实需求，它是一种能够被人们更好的理解的一种设计结构模式，它也是一种稳定的数据设计模式，很少会对其变动。概念性数据将用户和设计者之间搭建了沟通桥梁，它是数据库结构的设计基础。概念设计中自顶层向下的实体分析方法，即常用的实体联系模型，它对具体数据进行抽象加工，将实体集合抽象成实体模型(简称 E-R 模型)，对具体数据进行抽象加工，将实体集合抽象成实体类型。用实体间联系反映现实世界事物间的内在联系。E-R 模型是建立概念性数据模型的有力工具。

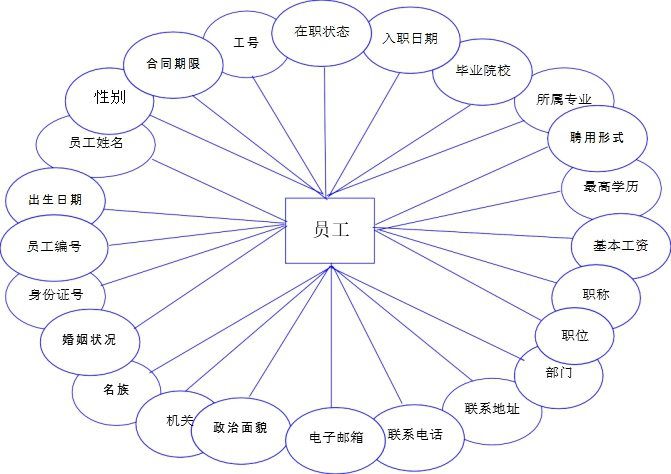


图3.1 员工培训、奖罚模型

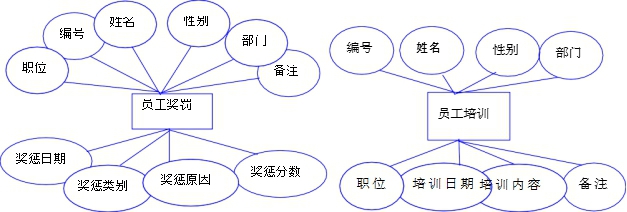


图3.2 员工培训、奖罚模型

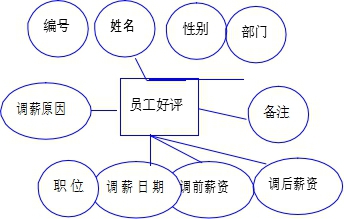


图3.3 员工好评模型

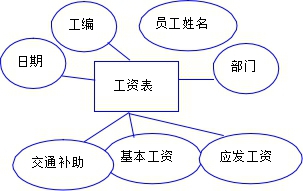


图3.4 员工工资模型

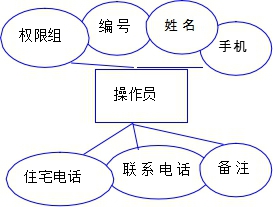


图3.5 操作员实体模型

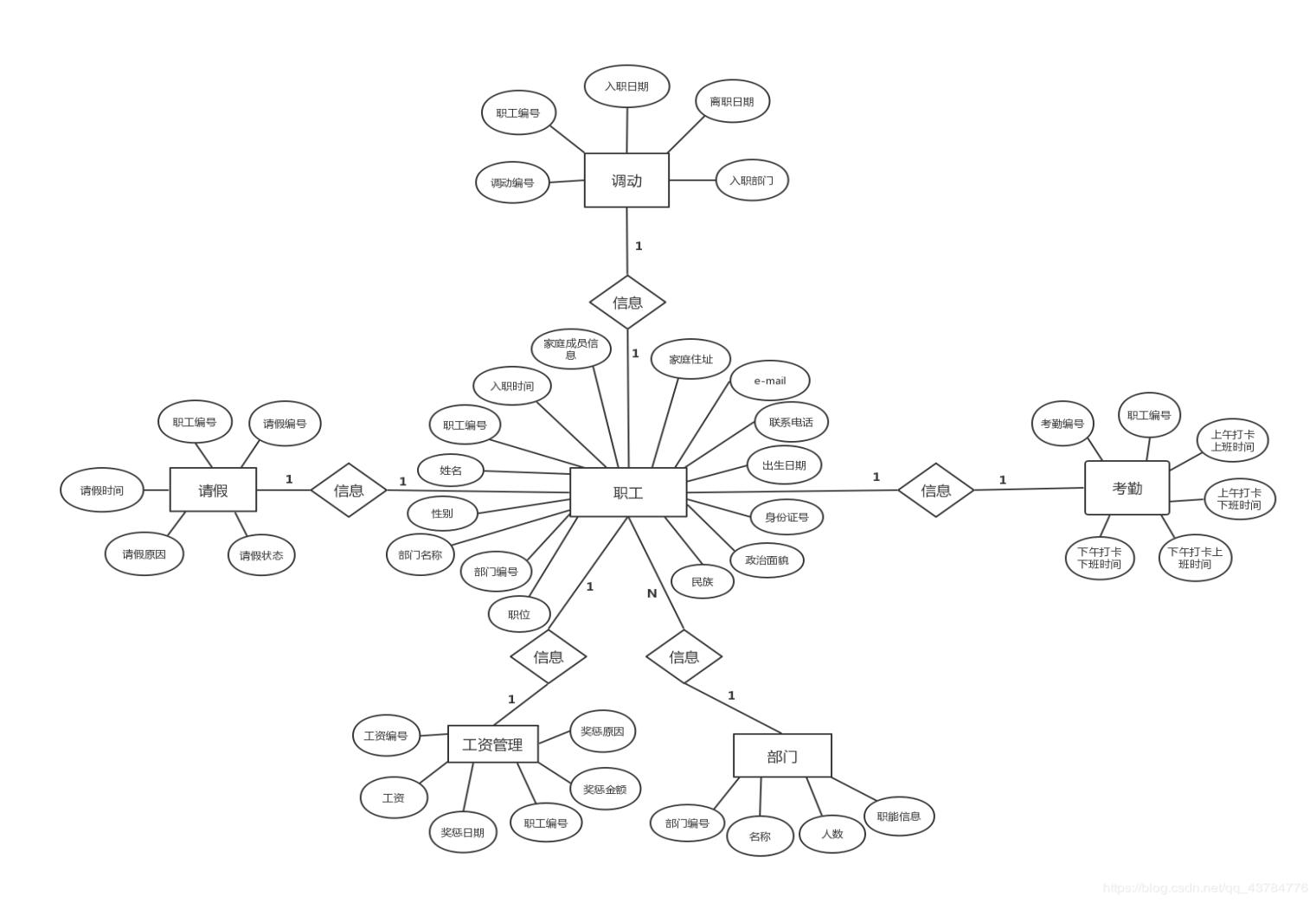


图3.6 全局E-R图

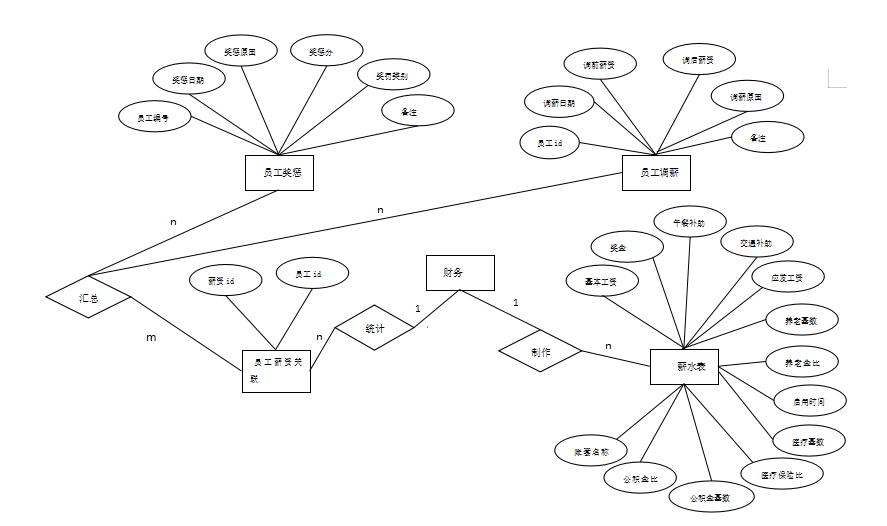


图3.7 财务E-R图

表设计时，将确定的实体对象的属性值映射为数据库表中的列的字段名称，在不同的表之间，按照名称来映射父子表之间的引用关系，同级别表之间，沿用主码、外码来映射它们之间的引用关系，之后就可以建立一个物理的数据模型。数据系统要求具有一下方面的特征：

1. 结构要合理，对人员要求建立多条操作记录；
2. 建立的数据冗余度尽可能小，独立性要强；
3. 查询、统计、修改要求快而准；
4. 有可靠性和保密性。

做到以上四个要求，数据的设计尽量做到符合第三范式的设计要求，即满足第三范式的约束：在关系模式 R(U, F)中的所有非主属性对任何候选关键字都不存在传递信赖，则称关系 R是属于第三范式的。从第一范式到第三范式是逐次“包含于”的关系。只有遵循关系型数据库设计的范式的约束才能做到更加规范化、合理化的数据结构，从而使的数据冗余减小，便于日后对各张表的操作更加简便灵活快速。部分关键性基本表的设计结果如下，表格做出了对部分数据表的字段进行了解释说明，系统的模块功能也是围绕各张表展开而来的：

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数值类型 | 可否为空 | 备注 |
| personnelID | Number(10) | NOT NULL | 员工编号，主键 |
| personnelName | Varchar2(10) | NOT NULL | 员工姓名 |
| personnelSex | Varchar2(4) | NULL | 性别 |
| birthday | Date | NULL | 出生日期 |
| ID | Number(20) | NOT NULL | 身份证号 |
| wedlock | Varchar2(4) | NULL | 婚姻状况 |
| race | Varchar2(8) | NULL | 名族 |
| nativePlace | Varchar2(20) | NULL | 籍贯 |
| politic | Varchar2(8) | NULL | 政治面貌 |
| E\_mail | Varchar2(20) | NULL | 电子邮箱 |
| phone | Number(10) | NULL | 联系电话 |
| address | Varchar2(20) | NULL | 联系地址 |
| department | Varchar2(20) | NOT NULL | 部门 |
| job | Varchar2(10) | NOT NULL | 职位 |
| duty | Varchar2(10) | NULL | 职称 |
| basisSalary | Number(10) | NULL | 基本工资 |
| engageForm | Varchar2(8) | NULL | 聘用形式 |
| TiptopDegree | Varchar2(8) | NULL | 最高学历 |
| specialty | Varchar2(8) | NULL | 所属专业 |
| school | Varchar2(8) | NULL | 毕业院校 |
| beginWorkDate | Date | NOT NULL | 入职日期 |
| workState | Varchar2(8) | NOT NULL | 在职状态 |
| workID | Number(10) | NOT NULL | 工号 |
| contractTerm | Number(4) | NULL | 合同期限 |
| beFormDate | Date | NULL | 转正日期 |
| notWorkState | Date | NULL | 离职日期 |
| beginContract | Date | NULL | 合同起始日期 |
| endContract | Date | NULL | 合同终止日期 |
| workAge | Number(4) | NULL | 工龄 |

表3.1 员工信息表(personnelInfo)/员工资料表(personnelDate)

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数值类型 | 可否为空 | 备注 |
| personnelID | Number(10) | NOT NULL | 员工编号，主键 |
| personnelEncOrChastDate | Date | NULL | 奖罚日  期 |
| personnelEncOrChastReason | Varchar2(10) | NULL | 奖罚原因 |
| personnelEncOrChastPoint | Varchar2(4) | NULL | 奖罚分 |

表 3.2 员工奖罚表(personnelEncouragementOrChastisement)

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数值类型 | 可否为空 | 备注 |
| personnelID | Number(10) | NOT NULL | 员工编号，主键 |
| trainDate | Date | NULL | 培训日期 |
| trainContent | Varchar2(8) | NULL | 培训内容 |
| remark | Varchar2(50) | NULL | 备注 |

表 3.3 员工培训表(personnelTrain)

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数值类型 | 可否为空 | 备注 |
| popedom | Varchar2(8) | NULL | 权限组 |
| opID | Number(10) | NOT NULL | 编号，主键 |
| opName | Varchar2(10) | NULL | 操作员姓名 |
| opMTele | Number(10) | NOT NULL | 手机号码 |
| opTele | Number(10) | NULL | 住宅电话 |
| opAddress | Varchar2(20) | NULL | 联系地址 |
| remark | Varchar2(50) | NULL | 备注 |

表 3.4 操作员表(op)

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| id | Integer | 主键，自增长 | hrID |
| name | String(32) |  | 姓名 |
| phone | String(11) |  | 手机号码 |
| telephone | String(16) |  | 住宅电话 |
| address | String(64) |  | 联系地址 |
| enabled | Enum | 默认值：1 | 账户是否可用 |
| username | String(255) |  | 用户名 |
| password | String(255) |  | 密码 |
| userface | String(255) |  | 用户头像 |
| remark | String(255) |  | 备注 |

表 3.5 hr表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 约束 | 说明 |
| id | Integer | 主键，自增长 | 主键 |
| name | String(32) |  | 部门名称 |
| parentId | Integer |  | 夫部门id |
| depPath | String(255) |  | 部门path |
| enabled | Enum | 默认值：1 | 是否可用 |
| isParent | Enum | 默认值：0 | 是否为夫部门 |

表3.6 deparement部门表

3.2 服务层获取持久层返回值

Java开发讲究分层制，不同的层级管理者不同的系统功能，人事管理系统也不例外，其中，最重要的逻辑开发层就是服务层的开发，包含了系统所需要的所有功能上的算法和计算过程，并与数据访问层和表现层交互。（服务端返回 数据的处理以及相应页面的变化），所以，这一层也是整个项目的难点部分。

首先，服务层通过@Service注解[1]标注在类上，表示这是一个服务层类，然后通过@Autowired注解注入操作相应业务库的数据库mapper接口类，在业务实现的的方法中，设置传入的相关的参数，在调用mapper类的相关方法时传入相关参数即可获取返回值。以查询数据库为例，如果返回值大于等于1，表明数据查询成功，已经从数据库获取到了相关记录。同时，服务层，也是对业务附赠功能操作的部分。在本此人事管理系统项目中，在查询方法的头部添加@Cacheable注解开启缓存功能，这样可以避免相同的数据库操作多次被执行，从而减轻数据库的负担，另一个拓展的新功能是，在添加新入职员工信息时，会给新入职的员工发送一封入职欢迎邮件，那么这一功能又是如何实现的呢？首先我们需要了解一个消息中间件，那就是rabbitMQ[1]消息中间件，用它来与第三方邮件服务器保持互联，通过这样来保证消息的可靠性。在使用时需要开通POP3/SMTP服务或者IMAP/SMTP服务，本项目以前者为例，当成功开启后，我们会拿到授权码，这样在项目配置文件中会使用到这个授权码。具体步骤如下：首先在配置邮件业务类中通过@Autowired注入JavaMailSender、MailProperties、TemplateEngine类，然后再方法中加入@RabbitListener注解监听服务层发送的邮件发送消息，再通过这三个类构造邮件内容和邮件发送功能，整体的邮件配置功能完成。接下来在服务层通过@Autowired注入RabbitTemplate的实例调用convertAndSend方法发送邮件发送功能。这些是服务层的拓展功能，但整体看下来，服务层主要是调用持久层用来对数据库进行操作，只不过在操作数据库时添加一些附属功能。

通过如下流程图，我们可以看到服务层Service在整体项目功能流程图中的大体作用，首先提供方法供Controller层调用，最后在方法内部通过具体实现逻辑调用数据库持久层方法实现对数据库的增删改查功能。

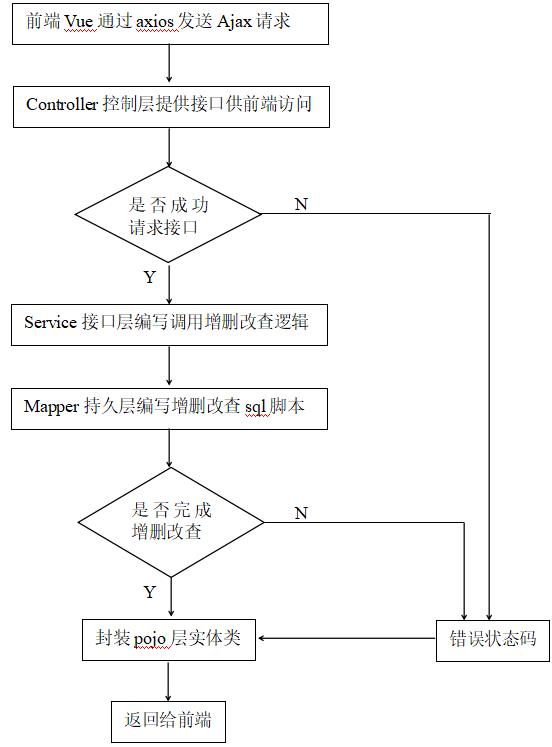


图3.1 流程图

3.3 控制层提供RESTful服务

百度百科对RESTful是这么定义的：RESTful是一种web应用程序的开发风格和设计模式，底层协议使基于[HTTP](https://baike.baidu.com/item/HTTP/243074" \t "https://baike.baidu.com/item/RESTful/_blank)，可以通过[XML](https://baike.baidu.com/item/XML/86251" \t "https://baike.baidu.com/item/RESTful/_blank)文件格式定义或[JSON](https://baike.baidu.com/item/JSON/2462549" \t "https://baike.baidu.com/item/RESTful/_blank)键值对格式定义。移动互联网企业使用RESTful风格作为业务功能接口，实现第三方[OTT](https://baike.baidu.com/item/OTT/9960940" \t "https://baike.baidu.com/item/RESTful/_blank)调用移动网络资源的功能，动作类型为新增、变更、删除所调用资源。

REST是一种Web风格软件架构，在项目设计中，可以通过@RestController注解开发一个RESTful服务，不过，SpringBoot对此提供了自动化配置方案，在开发时，只需要我们提供相关依赖就能够快速构建一个RESTful服务。有关RESTful的相关依赖是：spring-boot-starter-data-rest。RESTful服务构建成功后，默认的请求路径是实体类名小写在加上后缀。

3.4 axios请求接口的研究

首先需要分析为什么本项目需要引入axios插件，因为本项目是前后端分离项目，与传统的jsp项目引入第三方js库就可以实现AJAX请求不一样，所以，在前端与后端信息交互时就要用到axios。官网对axios是这么定义的：Axios 是一个基于 promise 的 HTTP 库[11]，可以用在浏览器和 node.js 中[10]。

axios可以执行get和post请求，请看下面例子：

// 发送get请求获取远端图片  
axios({  
 method:'get',  
 url:'http://bit.ly/2mTM3nY',  
 responseType:'stream'  
})  
 .then(function(response) {  
 response.data.pipe(fs.createWriteStream('ada\_lovelace.jpg'))  
});

// 发送 POST 请求  
axios({  
 method: 'post',  
 url: '/user/12345',  
 data: {  
 firstName: 'Fred',  
 lastName: 'Flintstone'  
 }  
});

可以看到，get方法和post方法都对应两个参数，分别是请求地址和携带的参数形式，可以通过向 axios 传递相关配置来创建请求。当然，本项目中用到的请求类型远不止上面两种形式，为方便起见，为所有支持的请求方法提供了别名 ：

axios.request(config)

axios.get(url[, config])

axios.delete(url[, config])

axios.head(url[, config])

axios.options(url[, config])

axios.post(url[, data[, config]])

axios.put(url[, data[, config]])

axios.patch(url[, data[, config]])

本项目用到的请求的详细参数配置如下：

{

// `url` 请求的服务器对应连接的 URL

url: '/user',

// `method` 以什么样的方式发送请求

method: 'get', // default，

transformRequest: [function (data, headers) {

// 对 data 数据格式进行任意转换处理

return data;

}],

// `headers` 发送的自定义请求头的设置

headers: {'X-Requested-With': 'XMLHttpRequest'},

// data 是作为请求主体被发送的数据

// data仅适用 'PUT', 'POST', 和 'PATCH'这些请求方法

// 如果请求的参数没有设置 `transformRequest` 时，规定是以下类型之一：

// - string, plain object, ArrayBuffer, ArrayBufferView, URLSearchParams

// - 浏览器专属：FormData, File, Blob

// - Node 专属： Stream

data: {

firstName: 'Fred'

}

}

上面是对axios常用请求类型和请求类型所需参数的说明，有请求必有数据响应，那么对应的响应的参数结构又是如何的呢，某个请求的响应包含以下信息 ：

{

// `data` 是服务器提供的响应信息

data: {},

// `status` 来自服务器响应的 HTTP 状态码，200表成功

status: 200,

// `statusText` 来自服务器响应的 HTTP 状态信息

statusText: 'OK',

// `headers` 服务器响应的头信息

headers: {},

// `config` 是在请求时提供的相关参数的配置信息

config: {},

request: {}

}

上面展示的是请求接口后作出响应返回的参数格式。传统的servlet+JSP的开发模式下，我们只需要引入第三方ajax库嵌入到jsp文件中可以完成对后端的请求和请求后通过jstl语法获取数据，而在前后端分离模式下，我们前端由于采用vue.js的缘故而不支持ajax请求，所以借助第三方插件axios以及nodeJS从而完成对后端接口数据的请求，这种模式下，掌握axios的请求参数书写规范和含义也就便得尤为重要了。

1. 结论与展望

## 4.1 结论

通过本次人事管理系统的设计，看到了Java系统类前后端分离开发的设计模式较传统servlet+JSP的优势，其中，后端开发不需要传奇modeview视图，而只需传递json数据给前端，前端拿到数据通过vue+element来进行数据的展示。这种后端只需传递json数据的设计优势是摒弃了后端java类代码嵌套前端代码的弊端，从而使代码看的不在混乱，也方便开发人员二次开发，大大提高了代码的耦合度。本次实验中，难点是对持久层数据库的操作，涉及到mybatis常用标签、属性整合数据库增删改查语句的编码，通过大量mybatis的xml文件的编码，掌握了mybatis在数据库操作方面的设计方式，再通过service服务层的逻辑判断和对工具类、持久层类的相关调用，获取查询并返回结果，控制层拿到这个结果数据通过@gettingRequest注解将封装好的json数据再返回给前端。

## 4.2 对进一步研究的展望

以上就是本次前后端分离项目的优势和对大致运行流程的分析，通过本次毕设项目的设计，熟练掌握了java的SSM框架的设计模式和运行流程，为日后走向工作岗位提供了技术支持和项目经验，锻炼了对java语言的编码能力。

# 参 考 文 献

[1]Spring Boot+Vue全栈开发实战 / 王松著.—北京：清华大学出版社，2019

[2]Java EE互联网轻量级框架整合开发：SSM框架（Spring MVC+Spring+MyBatis）和Redis实现/杨开振等著.—北京：电子工业出版社，2017.7

1. <https://www.github.com/> （GitHub官网）
2. <https://cn.vuejs.org/v2/guide/index.html> （Vue.js官网）
3. https://spring.io/projects/spring-boot#overview （Spring Boot 官网）
4. <https://element.eleme.io/#/zh-CN/component/icon> （Element组件官网）
5. <https://mybatis.org/mybatis-3/> （MyBatis官网）
6. <http://www.redis.cn/> （Redis中文网）
7. [https://mvnrepository.com/](https://mvnrepository.com/artifact/org.mybatis/mybatis-spring/2.0.3) （Maven仓储官网）
8. <http://nodejs.cn/> （Node.js中文网）
9. <http://www.axios-js.com/zh-cn/docs/> （axios中文网）
10. <https://spring.io/guides> （Spring学习指南）
11. <https://docs.oracle.com/javase/8/docs/api/> （Java API）
12. Spring MVC+MyBatis开发入门到项目实战 / 朱要光编著. —北京：电子 工业出版社，2018.2 ISBN 978-123-33390-3

# 致 谢

对于本次中小企业人事管理系统类的设计实现，首先最要致谢的是我的指导教师徐华老师，在她的监督指点下，我才能按时完成日常工作量。从选题开始，她就对我的选题范围和难度进行了可行性分析，帮助我顺利的固定了毕设的选题，在中途的开题报告的撰写中，也多次就毕设的内容进行过多次探讨并制定了研究工作进度计划。从前期的步步帮助就能看到，毕设的撰写就离不开老师的知识赞助，没有她的指点就不能有我顺利的固定选题和开题报告的编写，表面上毕设只是学生的任务，其实背后都有老师的认真阅读分析我们的原稿，因为她在帮我们分析哪处需要修改完善，哪处可有可无，哪部分又是多余累赘的等等，因此，再向老师道一声：老师您辛苦了！当然也要感谢校领导的支持，辅助学生去图书馆查阅相关资料以及背后的计划制定和付出，帮助我们全体毕业生这个集体进行工作质量的上的监督。

最后，感谢各位领导，老师，同学，等对给予过各类资助、指导和协助完成研究工作的曾今的好朋友，以及提供各种资源条件的个人、网站平台博主等表示致谢。

附录1

项目所需环境

1、Windows平台搭建Java开发运行环境jdk；

2、安装intelliJ IDEA和Webstorm；

3、maven搭建并配置本地仓库和安装nodeJS；

4、VMware安装并安装Linux操作系统centos；

5、centos中安装redis和rabbit MQ；

6、git安装和远程关联GitHub；

……