图书分类号：密 级：



毕业设计(论文)

基于SSH的电子邮件管理系统的设计与实现

**DESIGN AND IMPLEMENTATION OF E-MAIL MANAGEMENT SYSTEM BASE ON SSH**

学生姓名

|  |  |
| --- | --- |
| 学院名称 | 信电工程学院 |
| 学号  班级  专业名称指导教师 | 计算机科学与技术 |

2016 年 6 月 3 日

徐州工程学院学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用或参考的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。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注。

本人完全意识到本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日徐州工程学院学位论文版权协议书

本人完全了解徐州工程学院关于收集、保存、使用学位论文的规定，即：

本校学生在学习期间所完成的学位论文的知识产权归徐州工程学院所拥有。徐州工程学院有权保留并向国家有关部门或机构送交学位论文的纸本复印件和电子文档拷贝，允许论文被查阅和借阅。徐州工程学院可以公布学位论文的全部或部分内容，可以将本学位论文的全部或部分内容提交至各类数据库进行发布和检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位

|  |  |  |  |
| --- | --- | --- | --- |
| 论文作者签名： |  |  | 导师签名： |
| 日期： 年 | 月 | 日 | 日期： |

论文。

年 月 日

摘要

随着现代电子通信技术的发展，社会各机构的交流也日趋频繁。特别是企业内各部门之间的交流与协作成为企业内各部门进行良好运转的前提。为实现这一目的，企业内所有的员工都要参与到内部交流的活动中。尽管现代通信交流的工具非常多，但对于一般的大型企业往往会选择收发电子邮件的方式来完成企业内部各部门之间的交流与协作。

为了实现企业内部良好的通信，本设计针对企业员工为主体的用户对象，结合当前最优秀的Struts2+Spring+Hibernate轻量级Java EE架构，开发一个网页版的，简单易用的企业级电子邮件管理系统（Enterprise Email Management System，简称EEMS）。

本系统采用B/S模式开发，结合了UML建模技术完成了面向对象的分析建模，并严格遵循JavaEE中Servlet与JSP的开发规范。此外，本系统是以数据持久层，数据访问层，业务逻辑层与系统控制层为设计主体，以Web应用服务器为运行环境的JavaEE应用系统。

关键词 企业通信；电子邮件；Java EE

**Abstract**

With the development of modern electronic communication technology, exchanges between institutions of society are becoming increasingly frequent. Especially within the enterprise communication and collaboration between the various departments within the various departments become a prerequisite for a well-functioning enterprises. To achieve this, all employees within the enterprise to be involved in the internal communication activities. Although modern communication tool of communication very much, but for the average large companies often choose to e-mail the way to complete the communication and collaboration between various departments within the enterprise.

In order to achieve good communication within the enterprise, the design for the employees as the main user object with the current best Struts2 + Spring + Hibernate lightweight Java EE architecture, developed a web version, easy to use Enterprise Email Management System(EEMS).

The system uses the B / S model development, combined with UML modeling technology to complete object-oriented analysis model, and strictly follow the Java EE Servlet and JSP development of norms. In addition, the system is based on data persistence layer, data access layer, business logic layer and control system for the design of the main body, a Web application server runtime environment of Java EE applications. **Keywords** enterprise communication Email JavaEE

目 录

[1绪论 1](#_Toc80488)

[1.1 选题背景 1](#_Toc80489)

[1.2选题意义 1](#_Toc80490)

[1.3研究内容 1](#_Toc80491)

[2系统概述 2](#_Toc80492)

[2.1 相关技术介绍 2](#_Toc80493)

[2.1.1 Java EE平台 2](#_Toc80494)

[2.1.2 Tomcat服务器 2](#_Toc80495)

[2.1.3 Struts2.3 2](#_Toc80496)

[2.1.4 Hibernate4.3 2](#_Toc80497)

[2.1.5 Spring4.0 3](#_Toc80498)

[2.1.6 HTML、CSS、JavaScript 3](#_Toc80499)

[2.1.7 JQuery、Ajax、JSON 3](#_Toc80500)

[2.2.1 系统整体结构 4](#_Toc80501)

[2.2.2 系统各组件的调用关系 5](#_Toc80502)

[2.3 本章小结 5](#_Toc80503)

[3 面向对象的分析建模 6](#_Toc80504)

[3.1 UML与Rational Rose概述 6](#_Toc80505)

[3.2 UML用例建模 6](#_Toc80506)

[3.2.1 buc\_登录系统 7](#_Toc80507)

[3.2.2 buc\_查看邮件 7](#_Toc80508)

[3.2.3 buc\_编辑邮件 7](#_Toc80509)

[3.2.4 buc\_保存草稿 8](#_Toc80510)

[3.2.5 buc\_发送邮件 8](#_Toc80511)

[3.2.6 buc\_删除邮件 8](#_Toc80512)

[3.2.7 buc\_还原已删除邮件 9](#_Toc80513)

[3.2.8 buc\_查看个人通讯录 9](#_Toc80514)

[3.2.9 buc\_编辑个人通讯录 9](#_Toc80515)

[3.2.10 buc\_查看单位通讯录 9](#_Toc80516)

[3.3 UML动态建模 10](#_Toc80517)

[3.3.1 suc\_登录系统 10](#_Toc80518)

[3.3.2 suc\_注册新用户 11](#_Toc80519)

[3.3.3 suc\_查看收件箱邮件 13](#_Toc80520)

[3.3.4 suc\_查看已发送邮件 13](#_Toc80521)

[3.3.5 suc\_查看草稿箱邮件 14](#_Toc80522)

[3.3.6 suc\_查看已删除邮件 14](#_Toc80523)

[3.3.7 suc\_查看邮件内容 14](#_Toc80524)

[3.3.8 suc\_编辑已有邮件 15](#_Toc80525)

[3.3.9 suc\_写信 15](#_Toc80526)

[3.3.10 suc\_保存草稿 16](#_Toc80527)

[3.3.11 suc\_发送邮件 17](#_Toc80528)

[3.3.12 suc\_移到已删除 18](#_Toc80529)

[3.3.13 suc\_永久删除邮件 18](#_Toc80530)

[3.3.14 suc\_查看个人通讯录 19](#_Toc80531)

[3.3.15 suc\_删除联系人 19](#_Toc80532)

[3.3.16 suc\_添加联系人 20](#_Toc80533)

[3.3.17 suc\_查看单位通讯录 20](#_Toc80534)

[3.4 本章小结 20](#_Toc80535)

[4数据持久层的设计 21](#_Toc80536)

[4.1 提取持久化对象 21](#_Toc80537)

[4.2 持久化类之间的关系 21](#_Toc80538)

[4.3 持久化接口设计 23](#_Toc80539)

[4.3.1 实体Id接口 23](#_Toc80540)

[4.3.3 内容接口 24](#_Toc80541)

[4.3.4 附件接口 24](#_Toc80542)

[4.3.5 邮箱接口 24](#_Toc80543)

[4.3.6 用户接口 25](#_Toc80544)

[4.3.7 用户通讯录接口 25](#_Toc80545)

[4.3.8 用户通讯录接口 25](#_Toc80546)

[4.3.9 公司单位的接口 25](#_Toc80547)

[4.4 实现持久化类 26](#_Toc80548)

[4.4.1 邮件接口的实现类 26](#_Toc80549)

[4.4.2 内容接口的实现类 26](#_Toc80550)

[4.4.3 附件接口的实现类 27](#_Toc80551)

[4.4.4 邮箱接口的实现类 27](#_Toc80552)

[4.4.5 用户接口的实现类 27](#_Toc80553)

[4.4.6 用户通讯录接口的实现类 27](#_Toc80554)

[4.4.7 员工接口的实现类 28](#_Toc80555)

[4.4.8 机构单位接口的实现类 28](#_Toc80556)

[4.5 映射数据库的关系模型 28](#_Toc80557)

[4.5.1 邮件表 28](#_Toc80558)

[4.5.2 内容表 29](#_Toc80559)

[4.5.3 附件表 29](#_Toc80560)

[4.5.4 邮箱表 29](#_Toc80561)

[4.5.5 用户表 30](#_Toc80562)

[4.5.6 用户通讯录表 30](#_Toc80563)

[4.5.7 用户-用户通讯录表 30](#_Toc80564)

[4.5.8 员工表 31](#_Toc80565)

[4.5.9 机构单位表 31](#_Toc80566)

[4.6 本章小结 31](#_Toc80567)

[5数据访问层的设计 32](#_Toc80568)

[5.1 DAO接口的设计 32](#_Toc80569)

[5.1.1 BaseDao接口 32](#_Toc80570)

[5.1.2 MailDao接口 33](#_Toc80571)

[5.1.3 MailboxDao接口 33](#_Toc80572)

[5.1.4 ContentDao接口 34](#_Toc80573)

[5.1.5 AttachmentDao接口 34](#_Toc80574)

[5.1.6 UserDao接口 34](#_Toc80575)

[5.1.7 UserContactsDao接口 34](#_Toc80576)

[5.1.8 EmployeeDao接口 35](#_Toc80577)

[5.1.9 CompanyDao接口 35](#_Toc80578)

[5.2 实现DAO组件 36](#_Toc80579)

[5.2.1 实现DAO组件的基类 36](#_Toc80580)

[5.2.2 UserDaoImpl类 37](#_Toc80581)

[5.2.3 MailDaoImpl类 38](#_Toc80582)

[5.2.4 MailboxDaoImpl类 38](#_Toc80583)

[5.2.5 ContentDaoImpl类 38](#_Toc80584)

[5.2.6 AttachmentDaoImpl类 39](#_Toc80585)

[5.2.7 UserContactsDaoImpl类 39](#_Toc80586)

[5.2.8 EmployeeDaoImpl类 39](#_Toc80587)

[5.2.9 CompanyDaoImpl类 39](#_Toc80588)

[5.3 本章小结 40](#_Toc80589)

[6业务逻辑层的设计 41](#_Toc80590)

[6.1 业务接口的设计 41](#_Toc80591)

[6.1.1 登录验证的接口 41](#_Toc80592)

[6.1.2 用户注册的接口 41](#_Toc80593)

[6.1.3 加密服务的接口 41](#_Toc80594)

[6.1.4 查询邮件的接口 42](#_Toc80595)

[6.1.5 查询邮件数量的接口 42](#_Toc80596)

[6.1.6 邮件内容的服务接口 43](#_Toc80597)

[6.1.7 保存或更新邮件的接口 43](#_Toc80598)

[6.1.8 保存草稿的接口 43](#_Toc80599)

[6.1.9 发送邮件的接口 44](#_Toc80600)

[6.1.10 上传附件的接口 44](#_Toc80601)

[6.1.11 删除邮件的接口 44](#_Toc80602)

[6.1.12 还原已删除邮件的接口 45](#_Toc80603)

[6.1.13 查询通讯录的接口 45](#_Toc80604)

[6.1.14 添加通讯录联系人的接口 45](#_Toc80605)

[6.1.15 移除通讯录联系人的接口 46](#_Toc80606)

[6.2 实现Service组件 46](#_Toc80607)

[6.2.1 登录验证的服务 46](#_Toc80608)

[6.2.2 用户注册的服务 47](#_Toc80609)

[6.2.3 加密服务 48](#_Toc80610)

[6.2.4 查询邮件的服务 48](#_Toc80611)

[6.2.5 查询邮件数量的服务 50](#_Toc80612)

[6.2.6 邮件内容的服务 52](#_Toc80613)

[6.2.7 保存或更新邮件的服务 52](#_Toc80614)

[6.2.9 发送邮件的服务 55](#_Toc80615)

[6.2.10 上传附件的服务 57](#_Toc80616)

[6.2.11 删除邮件的服务 58](#_Toc80617)

[6.2.12 还原已删除邮件的服务 59](#_Toc80618)

[6.2.13 查询通讯录的服务 59](#_Toc80619)

[6.2.14 添加通讯录联系人的服务 60](#_Toc80620)

[6.2.15 移除通讯录联系人的服务 61](#_Toc80621)

[6.3 配置Spring的事务处理 61](#_Toc80622)

[6.4 本章小结 62](#_Toc80623)

[7系统控制层的设计 63](#_Toc80624)

[7.1 Action组件的设计 63](#_Toc80625)

[7.1.1 登录验证的Action 63](#_Toc80626)

[7.1.2 用户注册的Action 64](#_Toc80627)

[7.1.3 查询邮件的Action 65](#_Toc80628)

[7.1.4 查询邮件数量的Action 66](#_Toc80629)

[7.1.5 查询邮件内容的Action 67](#_Toc80630)

[7.1.6 保存草稿的Action 67](#_Toc80631)

[7.1.7 发送邮件的Action 68](#_Toc80632)

[7.1.8 删除邮件的Action 69](#_Toc80633)

[7.1.9 还原已删除邮件的Action 69](#_Toc80634)

[7.1.10 查询个人通讯录人数的Action 70](#_Toc80635)

[7.1.11 查询个人通讯录的Action 71](#_Toc80636)

[7.1.12 添加通讯录联系人的Action 71](#_Toc80637)

[7.1.13 移除通讯录联系人的Action 72](#_Toc80638)

[7.1.14 上传附件的Action 73](#_Toc80639)

[7.2 Action组件的配置 74](#_Toc80640)

[7.3 本章小结 74](#_Toc80641)

[8视图层的设计与系统实现 75](#_Toc80642)

[8.1 视图层的设计 75](#_Toc80643)

[8.2 系统的功能演示 75](#_Toc80644)

[8.2.1 用户登录 75](#_Toc80645)

[8.2.2 用户注册 76](#_Toc80646)

[8.2.3 查看收件箱邮件 77](#_Toc80647)

[8.2.4 查看邮件内容 77](#_Toc80648)

[8.2.5 回复邮件 78](#_Toc80649)

[8.2.6 保存草稿 78](#_Toc80650)

[8.2.7 发送邮件 79](#_Toc80651)

[8.2.8 删除邮件 80](#_Toc80652)

[8.2.9 添加常用联系人 81](#_Toc80653)

[8.2.10 查看个人通讯录 81](#_Toc80654)

[8.3 本章小结 81](#_Toc80655)

[结论 82](#_Toc80656)

[致谢 83](#_Toc80657)

[参考文献 84](#_Toc80658)

# 1绪论

## 1.1 选题背景

在现代的大型企业，企业内部各个机构的良好运转往往决定着一个企业运营的成败。为了使企业内部的各个机构都能良好运转，企业内部的每一个单元都需要有良好的交流与协作。尽管现代社会的交流工具非常多，但对于一般的大型企业往往会使用收发电子邮件的方式来完成企业内部各个机构的交流与协作。

试想一下以下场景：作为一个公司的领导，我需要给下面的所有员工发布一个重要的通知。同样，作为一个员工，我需要给领导提出某种请求，或者就某个任务的完成情况给领导做一个工作报告。再者，作为某个机构的部门主管，我需要联系其他机构的工作小组来对我们当前的任务给予相应的配合等等。

针对于上述情形，本设计以电子邮件管理系统的设计与实现为课题，针对企业员工为主体的用户对象，开发一个网页版的，结合当前最流行的Struts2+Spring+Hibernate的轻量级JavaEE架构实现的企业级电子邮件管理系统（Enterprise Email Management

System，简称EEMS）。

## 1.2选题意义

通过设计开发一个基于SSH架构的电子邮件管理系统来深入掌握Java面向对象分析设计方法，熟悉Java开发中常用的设计模式以和当前最流行，最优秀的Struts2+Spring +Hibernate的轻量级JavaEE应用架构及相关技术，了解企业级应用的开发流程和开发规范，对未来进入公司开发更大规模的企业级项目具有重要意义。

## 1.3研究内容

结合本系统的研究背景，针对本系统的应用架构和相关技术，本设计主要涉及以下研究内容。

* 基于UML建模的面向对象分析方法；
* 系统的分层结构设计；
* 实体对象等数据模型的设计；
* 数据访问组件的设计；
* Hibernate框架的搭建及Spring与Hibernate的整合；
* 业务逻辑组件的设计；
* 系统控制组件的设计；
* Struts2框架的搭建及Spring与Struts2的整合；
* 视图页面的设计与系统实现；

# 2系统概述

## 2.1 相关技术介绍

### 2.1.1 Java EE平台

Java EE(Java Platform,Enterprise Edition)是Sun公司在Java SE平台之上推出的Java企业级应用平台。这个版本以前称为J2EE，能够帮助我们开发和部署可移植的，健壮的，安全的JavaWeb应用。在经典的JavaEE规范中，往往以EJB(企业级JavaBean) 为核心，以开发Servlet与JSP为基本单位，以应用服务器为运行环境。这种开发模式因为分层不够清晰，组件间的耦合度较高，因此通常开发成本较高，后期维护困难。

本系统采用轻量级Java EE架构，这种架构保留了原有的经典Java EE应用架构，并且在其基础上提高了整个系统的可扩展性与可维护性，降低了系统的开发成本与以后的维护难度。

### 2.1.2 Tomcat服务器

Tomcat服务器是Apache软件基金会推出的一个开源的Web应用服务器。Tomcat服务器完全由Java实现，因此具备了平台无关性。此外，Tomcat服务器简单，易用，稳定性好，即可作为个人开发学习之用，也可发布商业型Web应用。因此，本系统也采用Tomcat 服务器为Web应用服务器，来部署与调试Java EE应用。

### 2.1.3 Struts2.3

Struts也是Apache软件基金会推出的一个基于MVC模式的Web应用框架。同时 Struts1也是最早的MVC框架，从它诞生起，一直备受Java EE应用开发者支持。但随着技术的进步，Struts1的局限性也渐渐暴露出来，并且制约着Struts1的发展。于是Struts1 结合了另一个优秀的MVC框架：WebWork，进化为全新的Struts2。Struts2在Struts1的基础上，以WebWork为核心，并且引入了WebWork优秀的设计理念，迅速成为MVC框架中的王者，并拥有极高的市场占有率。目前，Struts2框架的最新版本是Struts2.3，也是本系统所采用的版本（Struts2.3.16.1）。

### 2.1.4 Hibernate4.3

Hibernate是由GaVinKing开发的一个ORM开源框架。目前Hibernate已被Jboss应用服务器选作为数据持久层的解决方案，并且成为了当今最流行的ORM开源框架。

Hibernate实现了JPA（Java Persistent API）规范，并且对JDBC进行了轻量级的封装，通过Hibernate的Session对象去操纵持久化对象完成对关系数据库的持久化操作。

因此，Hibernate充当了面向对象程序设计与关系数据库设计的桥梁，实现了面向对象分析（OOA），面向对象设计（OOD），关系数据库设计的有机统一。本系统采用的Hibernate 版本是4.3.5 Final版。

### 2.1.5 Spring4.0

Spring是由RodJohnson为了降低企业应用开发的复杂性而开发的一个轻量级开源框架。它抽象出了大量企业应用开发的通用步骤，留给开发者的只是特定应用的相关部分，因此通过使用Spring可以大幅度提高开发效率，并且可以保证整个系统应用具有良好的设计规范。

Spring提供了非常优秀的轻量级IOC（Inversion of Control）机制，即一个对象的依赖对象都由Spring容器去注入进来，而不是由该对象去主动创建依赖对象，这种机制也称为依赖注入（Dependency Injection）。通过使用IOC机制，使得本系统中所有的控制组件，业务组件，数据访问组件以及持久化对象（Persistent Object,简称PO）都由 Spring容器生成，接受Spring容器的注入。这样，Spring向上可以与MVC框架整合，向下可以与持久层框架整合，实现了组件间解耦，提高了组件的可复用性与可扩展性。

此外，Spring还提供了非常优秀的AOP(Aspect Orient Programming)机制，即可以在一个业务组件的方法中动态织入增强处理完成业务组件的增强。本系统通过使用Spring 的AOP机制，给业务组件的业务方法中动态添加了Spring的事务处理，提高了组件的可扩展性。

本系统采用Spring的版本是4.0.4版。

### 2.1.6 HTML、CSS、JavaScript

本系统采用B/S模式架构，用户通过浏览器向Web服务器发送请求，服务器将返回HTML 页面给浏览器。本系统将采用JSP技术动态生成HTML页面。

此外，本系统还使用CSS技术给HTML页面添加样式，使用JavaScript给页面添加动态效果与事件响应，从而提高了页面的丰富度。

### 2.1.7 JQuery、Ajax、JSON

JQuery是一个优秀的轻量级JavaScript库，并兼容CSS3和众多浏览器。JQuery是基于模块的脚本库，它为开发者提供了众多的开发模块，如：选择DOM元素，操纵文档对象，添加事件处理，制作JQuery动画，Ajax等。使用JQuery可以帮助开发者更便捷地开发功能更强大的动态页面。

Ajax(Asynchronous JavaScript And XML),即异步的JavaScript与XML技术。通过采用Ajax技术，用户可以通过浏览器发送多个异步请求，无需等待服务器响应，即用户可以在发出起步请求后继续浏览页面，从而提高了页面的时间利用率。此外，当浏览器接收了服务器的响应后，浏览器也无需加载整个页面，只需更新部分页面即可，从而降低了数据的传输量，提高了用户的好感度。本系统将充分利用JQuery对Ajax的支持实现异步请求的发送。

JSON([JavaScript](http://baike.baidu.com/view/16168.htm)ObjectNotation),是JavaScript对象符号。与XML相同的是，JSON 也是平台无关的数据交换格式。但JSON语法简单易用，并且解析效率更高。通过使用JSON 作为数据交换格式，可以降低数据的传输量，提高网络传输率。本系统将充分利用Struts2 的JSON插件，完成服务器向浏览器返回JSON数据。

2.2 系统架构介绍

### 2.2.1 系统整体结构

本系统采用严格的轻量级Java EE应用架构，主要有以下几个分层：

* 视图层：由JSP页面实现；
* 系统控制层：由Struts2的核心控制器与系统控制器组成；
* 业务逻辑层：有Service组件构成；
* 数据访问层：由DAO组件构成；
* 数据持久层：由Hibernate的持久化类构成；
* 数据库服务层：使用Oracle数据库存储持久化数据；本系统的整体结构见图2-2-1：

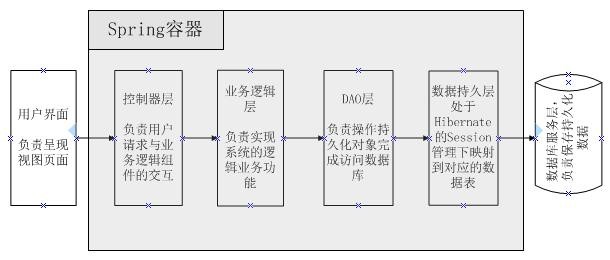


图2-2-1 系统整体结构图

由图2-2-1可见，灰色方框内的所有系统控制器，业务逻辑组件，DAO组件以及POJO 对象都由Spring容器负责生成，并管理组件的实例。

由此可见，Spring框架是整个系统的核心部分，Spring提供的IOC容器是所有系统控制器，业务逻辑组件，DAO组件，及POJO对象的大工厂。借助Spring容器的IOC机制，各组件都由Spring容器生成，并接受Spring容器的注入，完成了组件解耦，降低了系统重构的成本，极大的提高了系统的可维护性。

本设计将以本系统的整体结构为基准，以系统中每一个应用层次为设计核心，由后往前地逐一介绍每一个应用层次的组件设计，并将其部署于Web应用服务器，最终完成本系统的实现。

### 2.2.2 系统各组件的调用关系

本系统各组件的调用关系见图2-2-2：

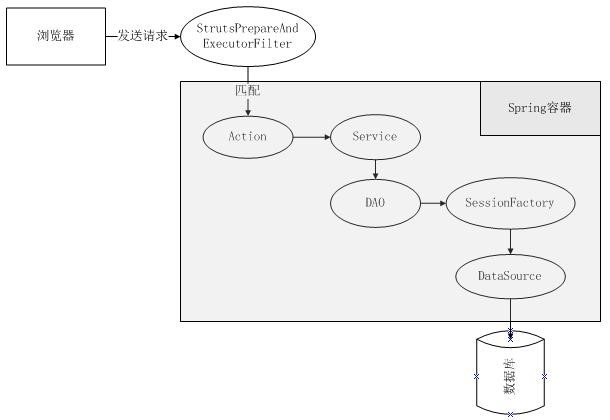


图2-2-2 系统组件调用关系图

由图2-2-2可见，当浏览器给Web服务器发送请求时，首先由Struts2的拦截器（StrutsPrepareAndExecutorFilter）拦截用户请求。然后根据Struts2的配置文件匹配到对应的系统控制组件（Action组件），对应的系统控制组件再调用相应的业务逻辑组件（Service组件）完成相应的业务逻辑操作。而Service组件又调用DAO组件去完成对数据库的访问操作。每个DAO组件又关联于Hibernate的SessionFactory，DAO组件从 SessionFactory获取Hibernate的Session对象，再由Session对象去操纵持久化对象去完成对数据库的持久化操作。

此外，系统中所有的Action组件，Service组件，DAO组件，SessionFactory，以及持久化对象都在Spring容器的管理范围之内，并且接受Spring容器的注入。Spring容器向上管理Action组件和Service组件，完成了Spring与Struts2的整合，向下管理DAO 组件和SessionFactory完成了Spring与Hibernate的整合。

## 2.3 本章小结

本章主要讲述了本系统运用到的相关技术，以及本系统的整体架构两方面。其中相关技术主要以 Java EE 平台为核心，以 tomcat 为 Web 应用服务器，结合 Struts2+Spring+Hibernate框架形成的轻量级JavaEE架构。系统整体架构方面主要讲述本系统包含的结构层次，每一个结构层次包含的系统组件，以及系统中各组件的调用关系。

# 3 面向对象的分析建模

面向对象分析（Object-Oriented Analysis）简称OOA，是指利用面向对象的概念和方法为软件系统构建模型，从而使用户的需求逐步精确化，一致化，完全化的过程。

面向对象分析中构建的模型主要有对象模型，动态模型与功能模型，面向对象分析的关键是找出问题领域的对象，在分析它们之间的相互关系后建立简洁的，精确的，可理解

的问题域模型。

## 3.1 UML与Rational Rose概述

UML(Unified Modeling Language)是一种功能强大的，面向对象的，图形化的系统分析建模语言。它为软件开发的所有阶段提供模型化，可视化的技术支持，可以帮助开发人员更好地理解系统的业务流程，建立简洁的，可靠的，完善的系统模型，从而使用户与开发人员对系统的问题描述达到一致的理解，保证了系统分析的正确性。

Rational Rose是一款全面支持UML的可视化图形建模工具，本章所有的建模图形都由Rose工具绘制。由Rose工具生成的UML建模文件另见毕业设计附件/UML建模路径下的

EEMS.mdl文件。

## 3.2 UML用例建模

UML用例建模是使用UML这种建模语言进行面向对象分析的的第一步，这一阶段主要采用用例图完成用例建模。用例图（Use Case Diagram）主要用于描述用户，需求，系统功能单元之间的关系，它展示了一个外部参与者能够观察到的系统功能模型图。

用例图主要由四部分组成：参与者（Actor），用例（UseCase），关联(Association) 系统边界，四部分组成。

本系统的业务用例图见图3-2：

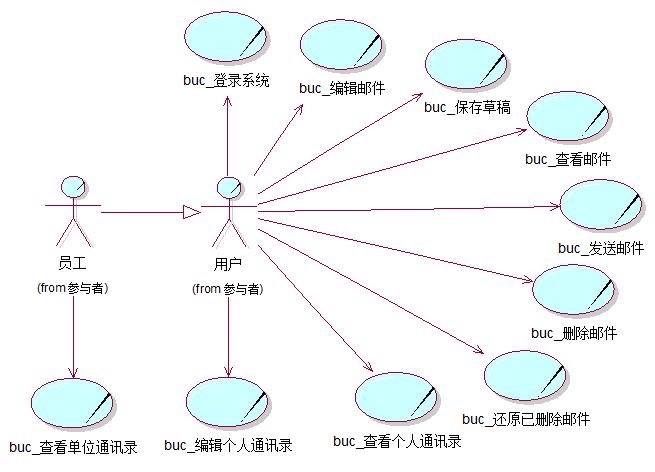


图3-2 业务用例图

接下来针对图3-2中业务用例图的每一个业务用例，完成其子系统的用例图。即对每一个业务用例进行细化，完善整个系统的需求模型。

### 3.2.1 buc\_登录系统

buc\_登录系统的子系统用例图见图3-2-1：

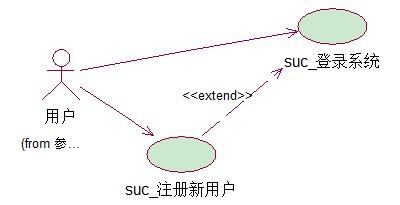


图3-2-1 buc\_登录系统的子系统用例图

### 3.2.2 buc\_查看邮件

buc\_查看邮件的子系统用例图见图3-2-2：

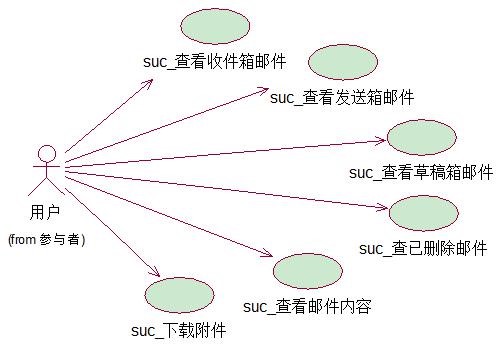


图3-2-2 buc\_查看邮件的子系统用例图

### 3.2.3 buc\_编辑邮件

buc\_编辑邮件的子系统用例图见图3-2-3：

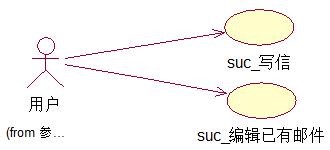


图3-2-3 buc\_编辑邮件的子系统用例图

### 3.2.4 buc\_保存草稿

buc\_保存草稿的子系统用例图见图3-2-4：

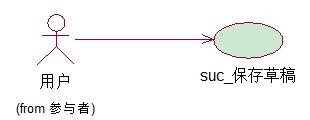


图3-2-4 buc\_保存草稿的子系统用例图

### 3.2.5 buc\_发送邮件

buc\_发送邮件的子系统用例图见图3-2-5：

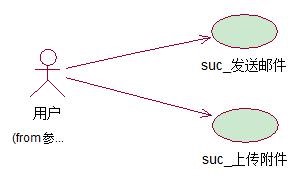


图3-2-5 buc\_发送邮件的子系统用例图

### 3.2.6 buc\_删除邮件

buc\_删除邮件的子系统用例图见图3-2-6：

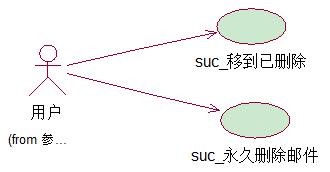


图3-2-6 buc\_删除邮件的子系统用例图

### 3.2.7 buc\_还原已删除邮件

buc\_还原已删除邮件的子系统用例图见图3-2-7：

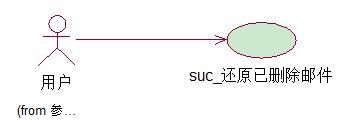


图3-2-7 buc\_还原已删除邮件的子系统用例图

### 3.2.8 buc\_查看个人通讯录

buc\_查看个人通讯录的子系统用例图见图3-2-8：

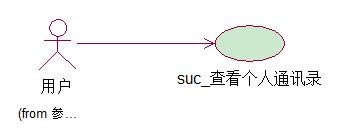


图3-2-8 buc\_查看个人通讯录的子系统用例图

### 3.2.9 buc\_编辑个人通讯录

buc\_编辑个人通讯录的子系统用例图见图3-2-9：

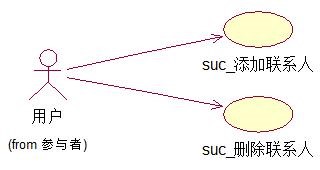


图3-2-9 buc\_编辑个人通讯录的子系统用例图

### 3.2.10 buc\_查看单位通讯录

buc\_查看单位通讯录的子系统用例图见图3-2-10：

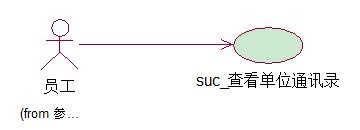


图3-2-10 buc\_查看单位通讯录的子系统用例图

## 3.3 UML动态建模

完成了UML用例建模后，我们接下来需要知道完成一个用例，参与者与系统之间如何经行交互。即需要对系统进行动态建模，在UML中通常采用活动图来完成。

### 3.3.1 suc\_登录系统

系统用例suc\_登录系统的活动图见图3-3-1。

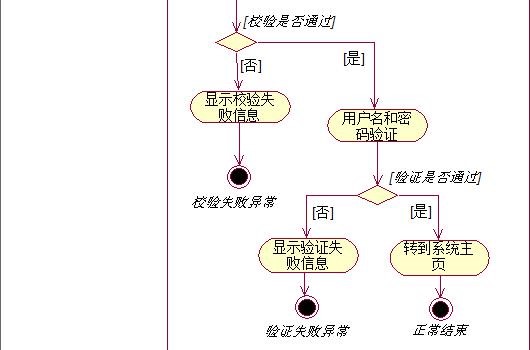
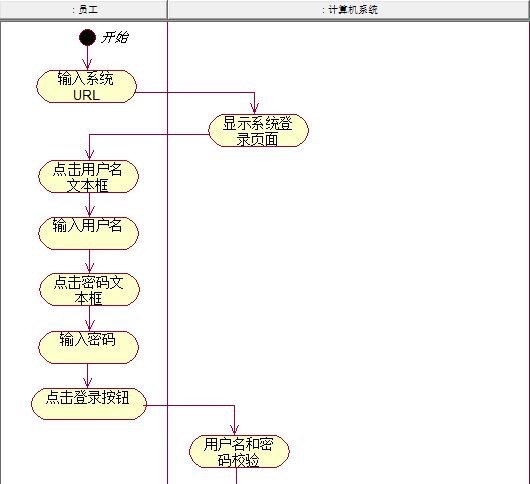


图3-3-1 suc\_登录系统的活动图

### 3.3.2 suc\_注册新用户

系统用例suc\_注册新用户的活动图见图3-3-2。

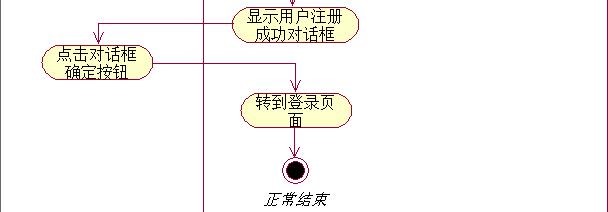
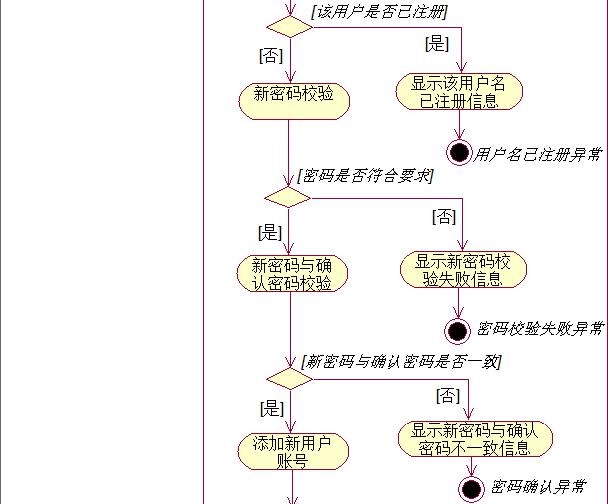
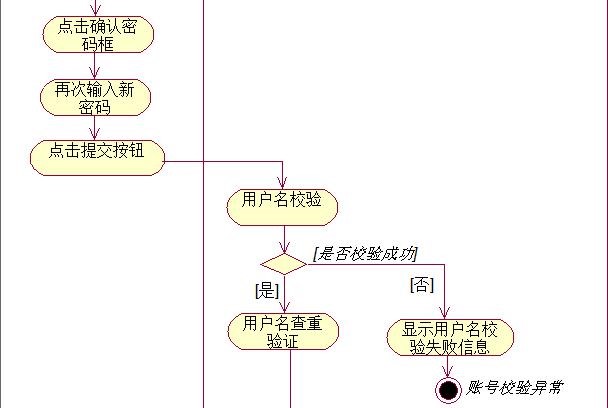
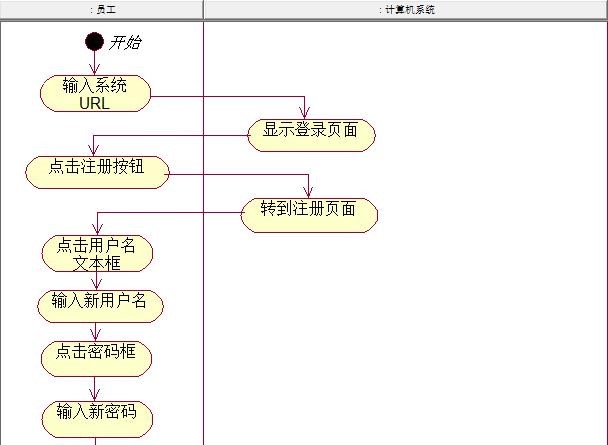


图3-3-2 suc\_注册新用户的活动图

### 3.3.3 suc\_查看收件箱邮件

系统用例suc\_查看收件箱邮件的活动图见图3-3-3。

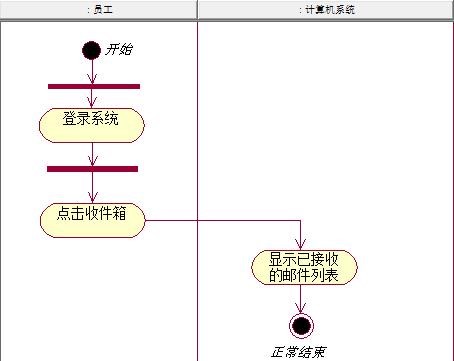


图3-3-3 suc\_查看收件箱邮件的活动图

### 3.3.4 suc\_查看已发送邮件

系统用例suc\_查看已发送邮件的活动图见图3-3-4。

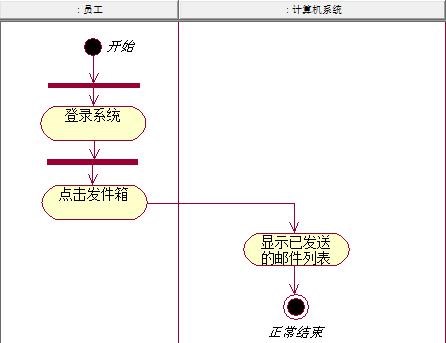


图3-3-4 suc\_查看已发送邮件的活动图

### 3.3.5 suc\_查看草稿箱邮件

系统用例suc\_查看草稿箱邮件的活动图见图3-3-5。

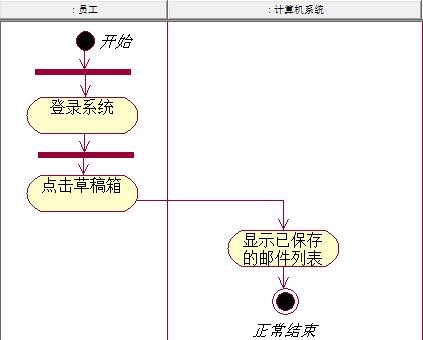


图3-3-5 suc\_查看草稿箱邮件的活动图

### 3.3.6 suc\_查看已删除邮件

系统用例suc\_查看已删除邮件的活动图见图3-3-6。



图3-3-6 suc\_查看已删除邮件的活动图

### 3.3.7 suc\_查看邮件内容

系统用例suc\_查看邮件内容的活动图见图3-3-7。

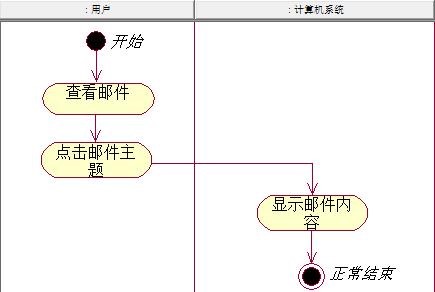


图3-3-7 suc\_查看邮件内容的活动图

### 3.3.8 suc\_编辑已有邮件

系统用例suc\_编辑已有邮件的活动图见图3-3-8。

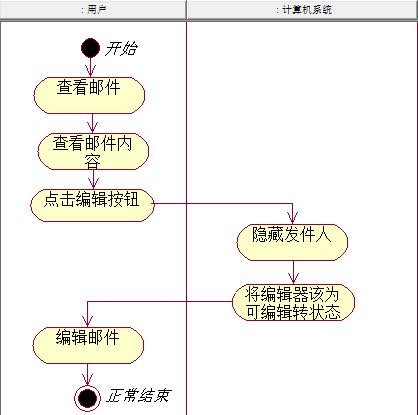


图3-3-8 suc\_编辑已有邮件的活动图

### 3.3.9 suc\_写信

系统用例suc\_写信的活动图见图3-3-9。

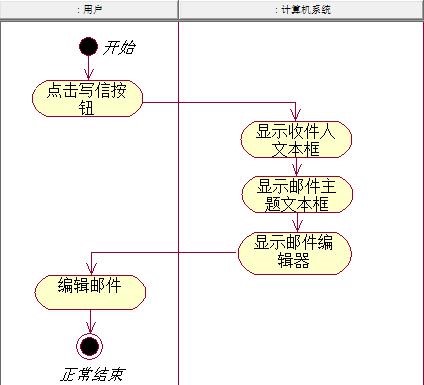


图3-3-9 suc\_写信的活动图

### 3.3.10 suc\_保存草稿

系统用例suc\_保存草稿的活动图见图3-3-10。

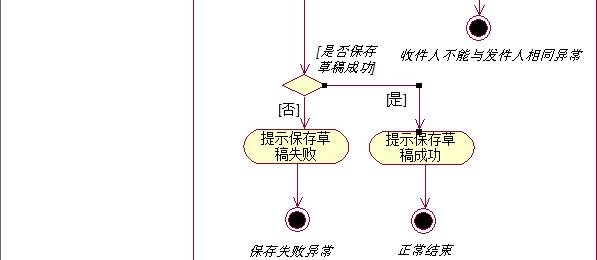
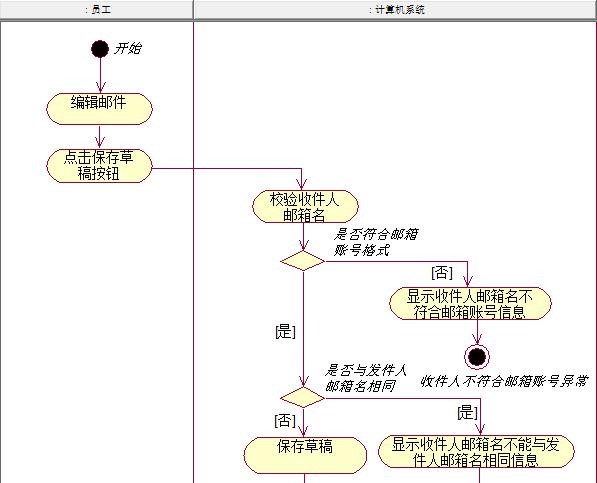


图3-3-10 suc\_保存草稿的活动图

### 3.3.11 suc\_发送邮件

系统用例suc\_发送邮件的活动图见图3-3-11。

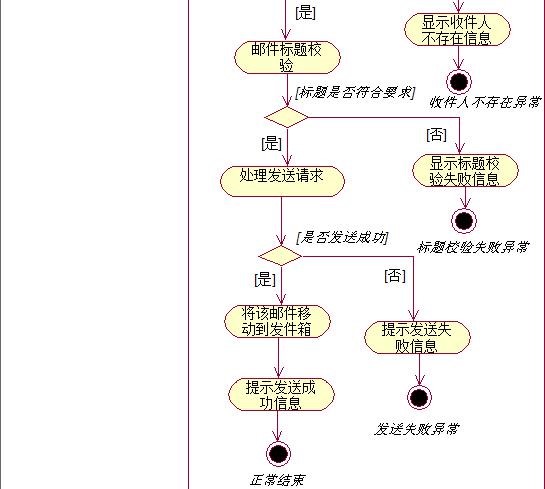
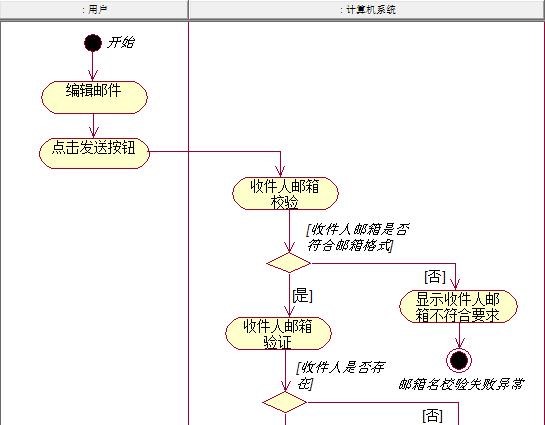


图3-3-11 suc\_发送邮件的活动图

### 3.3.12 suc\_移到已删除

系统用例suc\_移到已删除的活动图见图3-3-12。

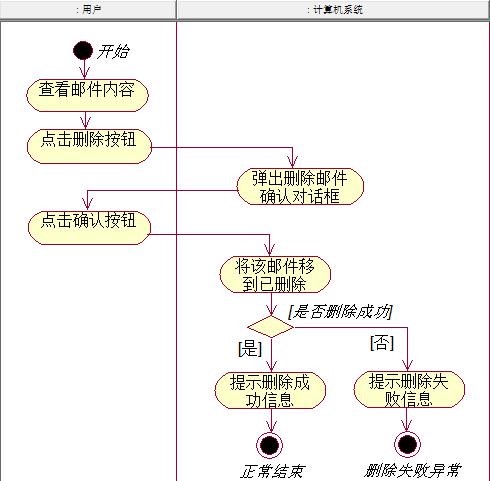


图3-3-12 suc\_移到已删除的活动图

### 3.3.13 suc\_永久删除邮件

系统用例suc\_永久删除邮件的活动图见图3-3-13。

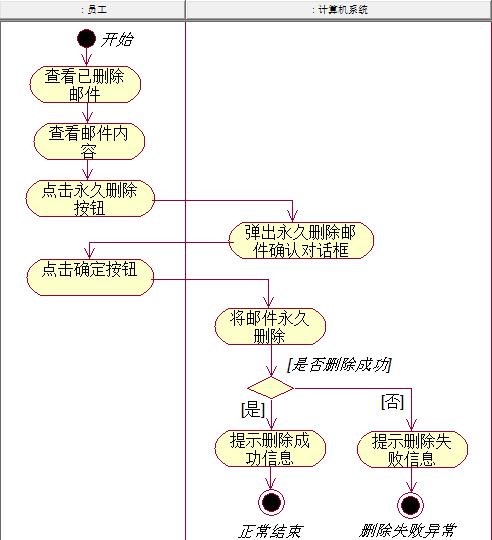


图3-3-13 suc\_永久删除邮件的活动图

### 3.3.14 suc\_查看个人通讯录

系统用例suc\_查看个人通讯录的活动图见图3-3-14。



图3-3-14 suc\_查看个人通讯录的活动图

### 3.3.15 suc\_删除联系人

系统用例suc\_删除联系人的活动图见图3-3-15。

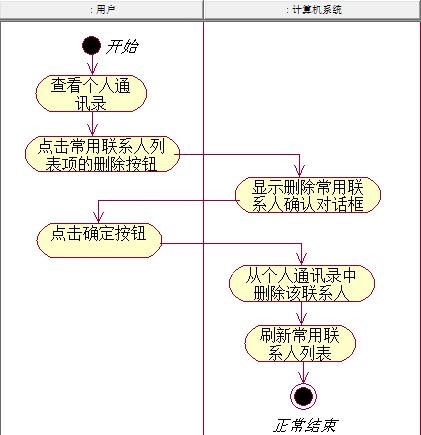


图3-3-15 suc\_删除联系人的活动图

### 3.3.16 suc\_添加联系人

系统用例suc\_添加联系人的活动图见图3-3-16。

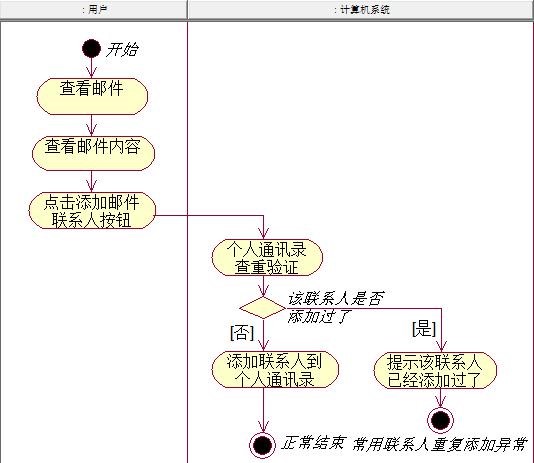


图3-3-16 suc\_添加联系人的活动图

### 3.3.17 suc\_查看单位通讯录

系统用例suc\_查看单位通讯录的活动图见图3-3-17。

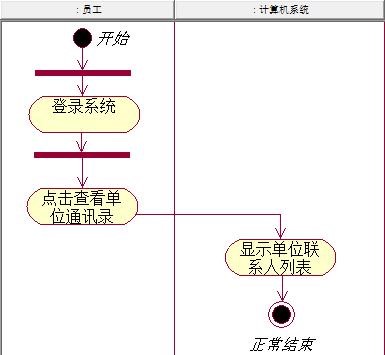


图3-3-17 suc\_查看单位通讯录的活动图

## 3.4 本章小结

本章主要讲述了采用UML这种图形建模语言完成对系统的面向对象分析建模。本系统采用用例图完成对系统的需求建模，采用活动图完成对系统的人机动态建模。此外，以后各章还会采用UML的类图完成系统的静态建模，采用时序图完成对象的交互建模。

# 4数据持久层的设计

## 4.1 提取持久化对象

面向对象的分析，是根据前面的系统需求提取出系统中的对象，将这些对象抽象成类。然而系统中需要保存的对象必须以关系模型的数据表的形式保存到数据库中。为了将系统中需要保存的实体类映射成数据库关系模型的数据表，需要借助ORM(Object/Relation

Mapping)框架完成这一映射。

本系统并没有预先设计数据库，而是完全由面向对象的分析开始，提取出需要保存的类，然后根据JPA(Java Persistence API)规范利用Hibernate框架将这些类转换成持久化类。这些持久化类对应于数据库关系模型的数据表。持久化类的实例，即持久化对象

（Persistent Object,简称PO），对应于数据表中的记录（实体）。

本系统共提取出8个需要保存的持久化对象：

* 邮件：描述现实世界中邮件的数据模型；
* 内容：描述邮件的内容；
* 附件：描述邮件的附件；
* 邮箱：描述现实世界中邮箱的数据模型；
* 用户：描述本系统参与者；
* 用户通讯录：用于记录每个用户的常用联系人；
* 员工：描述本系统参与者；
* 单位：描述员工所在的单位；

## 4.2 持久化类之间的关系

客观世界中的对象不可能是孤立存在的，它们之间必定存在着复杂的关联关系。分析对象之间的关联关系既是面向对象分析的必要步骤，也是使用Hibernate框架进行数据持久化操作的必经之路。

以下是本系统中8个PO的关联关系。

(1)对邮件与内容，一封邮件只能对应一个内容，一个内容也只能对应一封邮件。通过邮件可以找到其对应的内容，通过内容也可找到其对应的邮件。所以邮件与内容是双向一对一关联。

(2)对邮件与附件，一封邮件可以有多个附件，一个附件只能对应一封邮件。通过邮件可以找到对应的附件集，通过内容也可找到其对应的邮件。所以邮件与附件是双向一对多关联。

(3)对邮件与邮箱，邮箱有三种角色：邮件的所有者，邮件的发送者，邮件的接收者。

因此，邮件与邮箱存在三种关联。

对邮件与所有者，一封邮件只能有一个所有者，一个所有者可以有多封邮件。通过邮件可以找到其所有者，通过所有者也可以找到他的邮件集。所以邮件与所有者是双向多对一关联。

对邮件与发送者，一封邮件只能有一个发送者，一个发送者可以发送多封邮件。只能通过邮件去找其发送者，不能通过发送者去找邮件。所以邮件与发送者是单向多对一关联。

对邮件与接收者，一封邮件只能有一个接收者，一个接收者可以接收多封邮件。只能通过邮件去找其接收者，不能通过接收者去找邮件。所以邮件与接收者是单向多对一关联。

(4)用户与邮箱，一个用户只能有一个邮箱，一个邮箱也只能对应一个用户。通过用户可以找到他的邮箱，通过邮箱也可找到其对应的用户。所以用户与邮箱是双向一对一关联。

(5)对用户通讯录与用户，用户有两种角色：用户通讯录的所有者，用户通讯录中的联系人。

对用户通讯录与所有者，一个通讯录只能有一个所有者，一个所有者也只能有一个通讯录。通过通讯录可以找到其所有者，通过所有者也可以找到他的通讯录。所以用户通讯录与所有者是双向一对一关联。

对用户通讯录与联系人，一个通讯录可以有多个联系人，一个联系人的信息可以在多

个通讯录中。只能通过通讯录去找它对应的联系人集，不能通过联系人去找通讯录。所以用户通讯录与联系人是单向多对多关联。

(6)对员工与用户，员工继承于用户，即所有的员工都是用户。

(7)对员工与单位，一个员工只能在一个单位工作，一个单位可以有多个员工。通过员工可以找到他所在的单位，通过单位也可找到它的员工。所以员工与单位是双向的多对一关联。

根据上述分析，可得本系统中8个PO的关联关系图（见图4-2-1）：

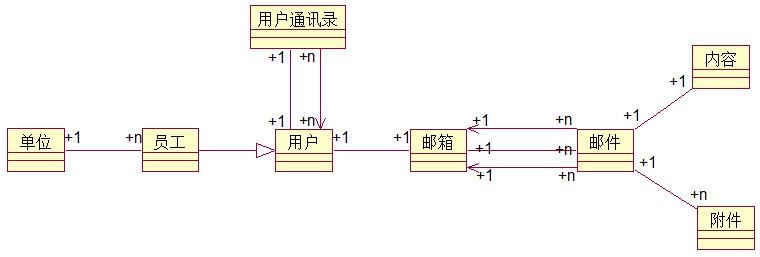


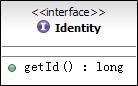
图4-2-1 PO的关联关系图

## 4.3 持久化接口设计

根据前面抽取出的持久化对象，即可设计这些持久化对象的持久化类。但本系统为了实现数据持久层的解耦，采用了面向接口的设计方法。即对于本系统中的每一个持久化类，先去定义这些持久化类的接口，然后再设计这些持久化类去实现这些接口。每一个持久化对象所关联的其他持久化对象并不是其他的持久化对象本身，而是关联这些持久化对象的接口。通过关联持久化对象的接口，实现了持久化类之间的解耦，以便以后系统的维护与升级。

根据前面抽取的持久化对象，本系统共定义9个持久化接口：

### 4.3.1 实体Id接口

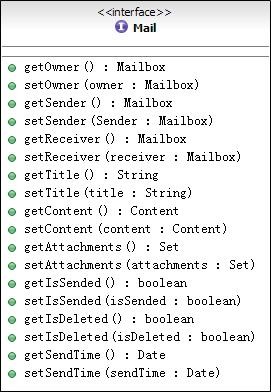
Identity：描述实体Id的接口，所有定义的持久化接口都继承该接口；如图4-3-1，Identity接口包含以下方法：

* 获取实体Id；

4.3.2 邮件接口 图4-3-1 Identity接口

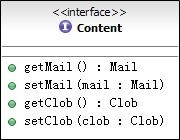
Mail：描述邮件的接口，该接口继承Identity接口。

如图4-3-2，Mail接口包含以下方法：

* 获取邮件的所有者；
* 设置邮件的所有者；
* 获取寄信人；
* 设置寄信人；
* 获取收信人；
* 设置收信人；
* 获取邮件标题；
* 设置邮件标题；
* 获取邮件内容；
* 设置邮件内容；
* 获取附件集；
* 设置附件集；
* 获取邮件的发送状态；
* 设置邮件的发送状态；
* 获取邮件的删除状态；
* 设置邮件的删除状态； 图4-3-2 Mail接口
* 获取发送的时间；
* 设置发送的时间；

### 4.3.3 内容接口

Content：描述邮件内容的接口，该接口继承Identity接口。

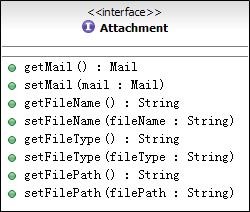
如图4-3-3，Content接口包含以下方法：

* 获取该内容对应的邮件；
* 设置该内容对应的邮件；
* 获取字符大对象；
* 设置字符大对象；

图4-3-3 Content接口

### 4.3.4 附件接口

Attachment：描述附件的接口，该接口继承Identity接口。

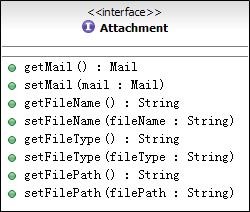
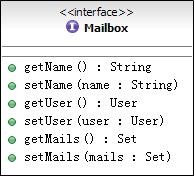
如图4-3-4，Attachment接口包含以下方法：

* 获取该附件对应的邮件；
* 设置该附件对应的邮件；
* 获取该附件的文件名称；
* 设置该附件的文件名称；
* 获取该附件的文件类型；
* 设置该附件的文件类型；
* 获取该附件的保存路径；
* 设置该附件的保存路径； 图4-3-4 Attachment接口

### 4.3.5 邮箱接口

Mailbox：描述邮箱的接口，该接口继承Identity接口。

如图4-3-5，Mailbox接口包含以下方法：

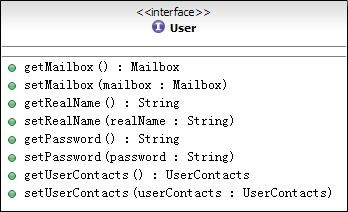


* 获取邮箱名；
* 设置邮箱名；
* 获取该邮箱的用户；
* 设置该邮箱的用户；
* 获取邮件集；
* 设置邮件集；

图4-3-5 Mailbox接口

### 4.3.6 用户接口

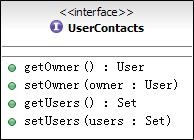
User：描述用户的接口，该接口继承Identity接口。

如图4-3-6，User接口包含以下方法：

* 获取邮箱；
* 设置邮箱；
* 获取真实姓名；
* 设置真实姓名；
* 获取密码；
* 设置密码；
* 获取个人通讯录；
* 设置个人通讯录； 图4-3-6 User接口

### 4.3.7 用户通讯录接口

UserContacts：描述用户通讯录的接口，该接口继承Identity接口。

如图4-3-7，UserContacts接口包含以下方法：

* 获取该个人通讯录下的所有者；
* 设置该个人通讯录下的所有者；
* 获取该通讯录下的用户集；
* 设置该通讯录下的用户集；

图4-3-7 UserContacts接口

### 4.3.8 用户通讯录接口

Employee：描述员工的接口，该接口继承于User接口。

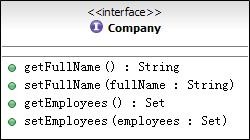
如图4-3-8，Employee接口包含以下方法：

* 获取该员工所在的单位；
* 设置该员工所在的单位;

图4-3-8 Employee接口

### 4.3.9 公司单位的接口

Company：描述公司单位的接口，该接口

继承Identity接口。如图4-3-9，Company接口包含以下方法：

* 获取公司全称；
* 设置公司全称；
* 获取该单位下的员工集；
* 设置该单位下的员工集； 图4-3-9 Company接口

## 4.4 实现持久化类

在富领域模式的设计中，这些持久化类都应该包含系统的业务逻辑方法，也就是使用领域对象来进行建模。但本系统采用贫血模型来设计持久化类，所以不打算提供任何的业务逻辑方法，而是将所有的业务逻辑方法放到业务逻辑组件中实现。

根据前面抽取出的对象以及定义的持久化接口，即可实现其相应的POJO类。然后给这些POJO类添加持久化配置，Hibernate会根据这些持久化配置将这些POJO转换成PO。

关于持久化配置的内容另见毕业设计附件/EEMS/src路径下cn.edu.xzit.mailbox.

entity.impl包中持久化类的代码。

### 4.4.1 邮件接口的实现类

MailImpl: Mail接口的实现类

由表4-4-1知，MailImpl类包含以下属性： 表4-4-1 MailImpl类

|  |
| --- |
| Class: MailImpl |
| private: id : Long private: owner : Mailbox private: sender : Mailbox private: receiver : Mailbox private: title : String private: content : Content  private: attachments : Set<Attachment> private: isSended : Boolean private: isDeleted : Boolean private: sendTime : Date |
| ...//省略所有的setter和getter方法 |

* id(邮件Id);
* owner(所有者);
* sender(发送者);
* receiver(接受者);
* title(主题);
* content(内容);
* attachments(附件集);
* isSended(是否已发送);
* isDeleted(是否已删除);
* sendTime(发送时间);

### 4.4.2 内容接口的实现类

|  |
| --- |
| Class: ContentImpl |
| private: id : Long private: mail : Mail private: clob : Clob |
| ...//省略所有的setter和getter方法 |

ContentImpl: Content接口的实现类 表4-4-2 ContentImpl类由表4-4-2可知，ContentImpl类包含以下属性：

* id(内容Id);
* mail(内容对应的邮件);
* clob(字符大对象);

### 4.4.3 附件接口的实现类

|  |
| --- |
| Class: AttachmentImpl |
| private: id : Long private: mail : Mail private: fileName : String private: fileType : String private: filePath : String |
| ...//省略所有的setter和getter方法 |

AttachmentImpl: Attachment接口的实现类由表4-4-3可知，AttachmentImpl类包含 表4-4-3 AttachmentImpl类以下属性：

* id(附件Id);
* mail(附件对应的邮件);
* fileName(文件名);
* fileType(文件类型);
* filePath (文件的相对路径);

### 4.4.4 邮箱接口的实现类

|  |
| --- |
| Class: MailboxImpl |
| private: id : Long private: name : String private: user : User Private: mails : Set<Mail> |
| ...//省略所有的setter和getter方法 |

MailboxImpl: Mailbox接口的实现类由表4-4-4可知，MailboxImpl类包含以下 表4-4-4 MailboxImpl类属性：

* id(邮箱Id);
* name(邮箱名);
* user(邮箱对应的用户);
* mails(邮箱的邮件集);

### 4.4.5 用户接口的实现类

UserImpl: User接口的实现类由表4-4-5可知，UserImpl类包含以下属性： 表4-4-5 UserImpl类

|  |
| --- |
| Class: UserImpl |
| private: id : Long private: mailbox : Mailbox private: realName : String private: password : String private: userContacts : UserContacts private: userContactses : Set<UserContacts> |
| ...//省略所有的setter和getter方法 |

* id(用户Id);
* mailbox(用户持有的邮箱);
* realName(用户的真实姓名);
* password(加密后的密码);
* userContacts(用户通讯录);
* userContactses(用户通讯录集);

### 4.4.6 用户通讯录接口的实现类

|  |
| --- |
| Class: UserContactsImpl |
| private: id : Long private: owner : User private: users : Set<User> |
| ...//省略所有的setter和getter方法 |

UserContactsImpl: UserContacts接口的实现类 表4-4-6 UserContactsImpl类由表4-4-6可知，UserContactsImpl类包含

以下属性：

* id(用户通讯录Id);
* owner(通讯录所有者);
* users(通讯录下的用户集);

### 4.4.7 员工接口的实现类

|  |
| --- |
| Class: EmployeeImpl extends UserImpl |
| private: company : Company |
| ...//省略所有的setter和getter方法 |

EmployeeImpl: Employee接口的实现类， 表4-4-7 EmployeeImpl类该类继承UserImpl类，由表4-4-7可知， EmployeeImpl类包含以下属性：

Company(员工所在的单位);

### 4.4.8 机构单位接口的实现类

|  |
| --- |
| Class: CompanyImpl |
| private: id : Long private: fullName : String private: employees : Set<Employee> |
| ...//省略所有的setter和getter方法 |

CompanyImpl: Company接口的实现类， 表4-4-8 CompanyImpl类由表4-4-8可知，CompanyImpl类包含以下属

性：

* id(机构单位Id);
* fullName(单位全称) employees(单位下的员工集);

## 4.5 映射数据库的关系模型

根据前面完成的持久化类的设计与持久化配置，通过Hibernate所提供的持久化支持

即可将持久化类映射成数据库关系模型的数据表。

以下是通过Hibernate直接将持久化类映射成数据表的表结构：

### 4.5.1 邮件表

表4-5-1为MailImpl类映射成MAIL表的表结构：

表4-5-1 MAIL表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| MAIL\_ID | 邮件Id | NUMBER(19,0) | No | Yes |
| ISDELETED | 是否已删除 | NUMBER(1,0) | Yes | No |
| ISSENDED | 是否已发送 | NUMBER(1,0) | Yes | No |
| SENDTIME | 发送时间 | TIMESTAMP(6) | Yes | No |
| TITLE | 邮件主题 | VARCHAR2(255  CHAR) | No | No |
| OWNER\_ID | 所有者邮箱Id | NUMBER(19,0) | No | No |
| RECEIVER\_ID | 接受者邮箱Id | NUMBER(19,0) | Yes | No |
| SENDER\_ID | 发送者邮箱Id | NUMBER(19,0) | Yes | No |

主键：MAIL\_ID; 外键：OWNER\_ID(参照MAILBOX表主键”MAILBOX\_ID”), RECEIVER\_ID(参照MAILBOX表主键”MAILBOX\_ID”), SENDER\_ID(参照MAILBOX表主键”MAILBOX\_ID”);

### 4.5.2 内容表

表4-5-2为ContentImpl类映射成CONTENT表的表结构：

表4-5-2 CONTENT表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| CONTENT\_ID | 内容Id | NUMBER(19,0) | No | Yes |
| CLOB | 字符大对象 | CLOB | Yes | No |
| MAIL\_ID | 邮件Id | NUMBER(19,0) | Yes | Yes |

主键：CONTENT\_ID;

外键：MAIL\_ID(参照MAIL表主键”MAIL\_ID”);

### 4.5.3 附件表

表4-5-3为AttachmentImpl类映射成ATTACHMENT表的表结构：

表4-5-3 ATTACHMENT表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| ATTACHMENT\_ID | 附件Id | NUMBER(19,0) | No | Yes |
| FILENAME | 文件名 | VARCHAR2(255  CHAR) | Yes | No |
| FILEPATH | 文件相对路径 | VARCHAR2(255  CHAR) | Yes | No |
| FILETYPE | 文件类型 | VARCHAR2(255  CHAR) | Yes | No |
| MAIL\_ID | 邮件Id | NUMBER(19,0) | No | No |

主键：ATTACHMENT\_ID; 外键：MAIL\_ID(参照MAIL表主键”MAIL\_ID”);

### 4.5.4 邮箱表

表4-5-4为MailboxImpl类映射成MAILBOX表的表结构：

表4-5-4 MAILBOX表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| MAILBOX\_ID | 邮箱Id | NUMBER(19,0) | No | Yes |
| NAME | 邮箱名 | VARCHAR2(255  CHAR) | No | No |
| USER\_ID | 用户Id | NUMBER(19,0) | Yes | Yes |

主键：MAILBOX\_ID;

外键：USER\_ID(参照USERINFO表主键”USER\_ID”);

### 4.5.5 用户表

表4-5-5为UserImpl类映射成USERINFO表的表结构：

表4-5-5 USERINFO表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| USER\_ID | 用户Id | NUMBER(19,0) | No | Yes |
| PASSWORD | 密码（已加密） | VARCHAR2(255  CHAR) | No | No |
| REALNAME | 真实姓名 | VARCHAR2(255  CHAR) | Yes | No |

主键：MAILBOX\_ID;

### 4.5.6 用户通讯录表

表4-5-6为UserContactsImpl类映射成USERCONTACTS表的表结构：

表4-5-6 USERCONTACTS表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| USERCONTACTS\_ID | 用户通讯录Id | NUMBER(19,0) | No | Yes |
| USER\_ID | 用户Id | NUMBER(19,0) | Yes | Yes |

主键：USERCONTACTS\_ID;

外键：USER\_ID(参照USERINFO表主键”USER\_ID”);

### 4.5.7 用户-用户通讯录表

表4-5-7为UserImpl类与UserContactsImpl类的连接表映射成USER\_USERCONTACTS

表的表结构：

表4-5-7 USER\_USERCONTACTS表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| USERCONTACTS\_ID | 用户通讯录Id | NUMBER(19,0) | No | No |
| USER\_ID | 用户Id | NUMBER(19,0) | No | No |

主键：USERCONTACTS\_ID，USER\_ID;

外键：USERCONTACTS\_ID(参照USERCONTACTS表的主键”USERCONTACTS\_ID”),

USER\_ID(参照USERINFO表主键”USER\_ID”);

### 4.5.8 员工表

表4-5-8为EmployeeImpl类映射成EMPLOYEE表的表结构：

表4-5-8 EMPLOYEE表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| USER\_ID | 用户Id | NUMBER(19,0) | No | Yes |
| COMPANY\_ID | 单位Id | NUMBER(19,0) | No | No |

主键：USER\_ID;

外键：USER\_ID(参照USERINFO表主键”USER\_ID”)，

COMPANY\_ID(参照COMPANY表主键”COMPANY\_ID”);

### 4.5.9 机构单位表

表4-5-9为CompanyImpl类映射成COMPANY表的表结构：

表4-5-9 COMPANY表的表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段含义 | 数据类型 | 能否为空 | 是否唯一 |
| COMPANY\_ID | 机构单位Id | NUMBER(19,0) | No | Yes |
| FULLNAME | 机构单位全称 | VARCHAR2(255  CHAR) | Yes | No |

主键：COMPANY\_ID;

## 4.6 本章小结

本章主要讲述了采用面向对象的分析方法完成本系统的数据持久层的设计。数据持久层的设计包含了从客观世界中提取持久化对象，分析持久化类之间的关系，设计持久化类的接口，实现持久化类以及Hibernate的持久化配置。最后，还讲述了Hibernate的ORM 机制实现持久化类与数据库关系模型的数据表之间的映射。

# 5数据访问层的设计

数据访问层也称DAO(DataAccessObject)层。 DAO模式是标准的JavaEE设计模式，

DAO模式的核心思想是所有的数据库访问操作都通过DAO组件来完成。DAO组件封装的对数据库进行CRUD(建表，查询，更新， 删除)等原子操作。

引入了DAO模式后，每个DAO组件都封装了对一个数据库的数据表经行CRUD操作的

API。所有的业务逻辑组件都依赖于DAO组件提供的原子操作，完成系统业务逻辑的实现。

## 5.1 DAO接口的设计

与数据持久层相同的是，DAO层也要实现与业务逻辑层的解耦，同样采用面向接口的设计方法。即先定义DAO接口，然后实现DAO接口去完成DAO组件的设计。所有的业务逻辑组件都关联于DAO接口，而不是关联与具体的DAO实现类。这样就完成了DAO层与业务逻辑层之间的分离。

### 5.1.1 BaseDao接口

为了简化DAO组件的设计，所有的DAO组件共有的方法需要提取出来，放到BaseDao 组件中去实现。其他DAO组件都继承该BaseDao组件去完成共有的DAO操作。

通常所有的DAO组件都应该提供以下方法：

* 根据ID加载持久化实体；
* 保存持久化实体；
* 更新持久化实体；
* 删除持久化实体以及根据ID删除持久化实体；
* 获取所有持久化实体；
* 获取实体总数；

因此，本系统需要提供一个BaseDao接口。

BaseDao接口的设计类图见图5-1-1。

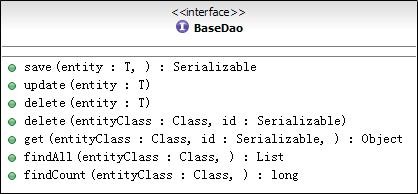


图5-1-1 BaseDao接口

### 5.1.2 MailDao接口

MailDao接口：用于访问持久化实体Mail的接口，该接口继承BaseDao接口。

MailDao接口包含以下方法：

* 根据所有者分页查询所有邮件；
* 根据所有者分页查询已删除的邮件；
* 根据所有者分页查询保存为草稿的邮件；
* 根据所有者分页查询已发送的邮件；
* 根据所有者分页查询已接收的邮件；
* 根据所有者查询邮件总数；
* 根据所有者查询已删除的邮件总数；
* 根据所有者查询保存为草稿的邮件总数；
* 根据所有者查询已发送的邮件总数； 根据所有者查询已接收的邮件总数；

根据以上接口描述，可得到该接口的设计类图（图5-1-2）。

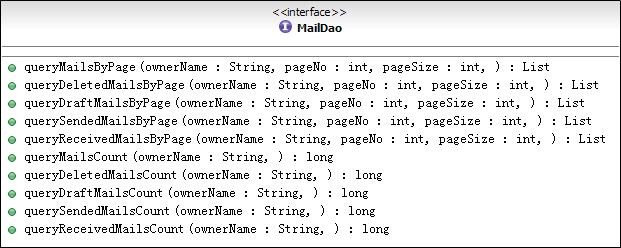


图5-1-2 MailDao接口

### 5.1.3 MailboxDao接口

MailboxDao接口：用于访问持久化实体MailboxDao的接口，该接口继承BaseDao接口。MailboxDao接口包含以下方法：

* 根据邮箱名查询邮箱；
* 根据邮箱名查询邮箱Id；

根据以上接口描述，可得到该接口的设计类图（图5-1-3）。



图5-1-3 MailboxDao接口

### 5.1.4 ContentDao接口

ContentDao接口：用于访问持久化实体Content的接口，该接口继承BaseDao接口。

ContentDao接口包含以下方法：

将给定的字符串转换成Clob； 根据邮件ID查询内容；根据以上接口描述，可得到该接口的设计类图（图5-1-4）。



图5-1-4 ContentDao接口

### 5.1.5 AttachmentDao接口

AttachmentDao接口：用于访问持久化实体Attachment的接口，该接口继承BaseDao 接口。

AttachmentDao接口包含以下方法：

根据邮件Id查询附件集；根据以上接口描述，可得到该接口的设计类图（图5-1-5）。



图5-1-5 AttachmentDao接口

### 5.1.6 UserDao接口

UserDao接口：用于访问持久化实体User的接口，该接口继承BaseDao接口。

UserDao接口包含以下方法：

根据用户名（邮箱名）查询用户；根据以上接口描述，可得到该接口的设计类图（图5-1-6）。



图5-1-6 UserDao接口

### 5.1.7 UserContactsDao接口

UserContactsDao接口：用于访问持久化实体UserContacts的接口，该接口继承 BaseDao接口。

UserContactsDao接口包含以下方法：

* 根据用户名（邮箱名）查询用户通讯录；
* 根据用户名分页查询常用联系人；
* 根据用户名查询联系人人数；

根据以上接口描述，可得到该接口的设计类图（图5-1-7）。

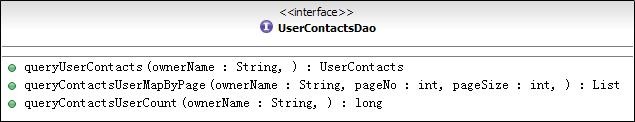


图5-1-7 UserContactsDao接口

### 5.1.8 EmployeeDao接口

EmployeeDao接口：用于访问持久化实体Employee的接口，该接口继承BaseDao接口。

EmployeeDao接口包含以下方法：

根据单位分页查询员工集；

根据以上接口描述，可得到该接口的设计类图（图5-1-8）。



图5-1-8 EmployeeDao接口

### 5.1.9 CompanyDao接口

CompanyDao接口：用于访问持久化实体Company的接口，该接口继承BaseDao接口。

根据以上接口描述，可得到该接口的设计类图（图5-1-9）。



图5-1-9 CompanyDao接口

## 5.2 实现DAO组件

完成了DAO接口的设计后，接下来就可以实现DAO接口去完成DAO组件的设计。实现了DAO接口的DAO组件可配置到Spring容器中，接受Spring容器的管理。以后可以充分利用Spring框架的IOC机制，在业务逻辑组件中注入DAO组件，实现了业务逻辑组件与

DAO组件的解耦。

以下是本系统所有DAO组件的关系类图(图5-2)。



图5-2 DAO组件的关系类图

### 5.2.1 实现DAO组件的基类

在前面的DAO接口的设计中，为了简化DAO接口的设计，定义了一个BaseDao接口。在实现DAO组件之前，同样先实现BaseDao接口去完成DAO组件基类的设计，然后所有的

DAO组件都继承该DAO基类，这样可实现DAO组件代码的重用。

在Hibernate框架中，所有的PO都必须与特定的Session对象关联才具有持久化的能力。而Hibernate的Session对象依赖于SessionFactory对象产生，所有的持久化操作都要依赖于特定的Session实现，而所有的Session对象都要在事务结束后关闭。因此，要实现DAO组件局有访问数据库的能力，必须要使每个DAO组件都与SessionFactory关

联，然后从SessionFactory中获取Session对象完成所有的持久化操作（见图5-2-1-1）。

由于前面已经定义好了 DAO 组件的基类，所以只需由 DAO 组件的基类去关联

SessionFactory即可完成所有DAO组件与SessionFactory关联。

DAO组件中每个对数据库操作方法的顺序图见图5-2-1-1。

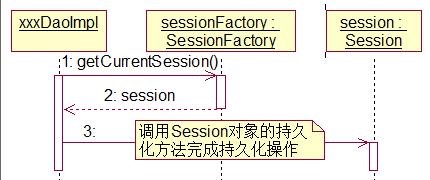


图5-2-1-1 DAO组件中对数据库操作方法的顺序图

为了使Spring框架与Hibernate框架整合，DAO组件基类的SessionFactory对象需要由Spring容器的注入。此外，所有的DAO组件都要配置在Spring容器中，然后由Spring 容器为业务逻辑组件注入 DAO 组件，完成业务逻辑组件对 DAO 组件的依赖。有关

SessionFactory 在 Spring 容器中的配置另见毕业设计附件/EEMS/WebRoot 路径下的 applicationContext.xml文件。

BaseDaoHibernate4类：该类是所有DAO组件的基类，所有的DAO组件都需要继承该基类实现共有的持久化操作。

图5-2-1-2为BaseDaoHibernate4类的设计类图，具体的实现另见毕业设计附件

/EEMS/src下cn.edu.xzit.mailbox.dao.impl.BaseDaoHibernate4类的代码。

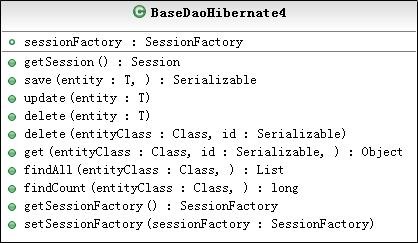


图5-2-1-2 BaseDaoHibernate4类

### 5.2.2 UserDaoImpl类

UserDaoImpl类：该类实现了UserDao接口，并继承了BaseDaoHibernate4类；图5-2-2为UserDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src下 cn.edu.xzit.mailbox.dao.impl.UserDaoImpl类的代码。



图5-2-2 UserDaoImpl类

### 5.2.3 MailDaoImpl类

MailDaoImpl类：该类是MailDao接口的实现类，并继承了BaseDaoHibernate4类；图5-2-3为MailDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src下 cn.edu.xzit.mailbox.dao.impl.MailDaoImpl类的代码。

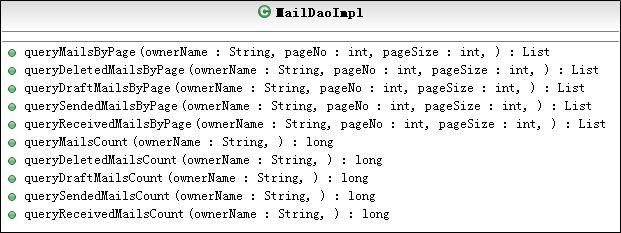


图5.2.3 MailDaoImpl类

### 5.2.4 MailboxDaoImpl类

MailboxDaoImpl类：该类是MailboxDao接口的实现类，并继承了BaseDaoHibernate4 类；

图5-2-4为MailboxDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src 下cn.edu.xzit.mailbox.dao.impl.MailboxDaoImpl类的代码。



图5-2-4 MailboxDaoImpl类

### 5.2.5 ContentDaoImpl类

ContentDaoImpl类：该类是ContentDao接口的实现类，并继承了BaseDaoHibernate4 类；

图5-2-5为ContentDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src 下cn.edu.xzit.mailbox.dao.impl.ContentDaoImpl类的代码。



图5-2-5 ContentDaoImpl类

### 5.2.6 AttachmentDaoImpl类

AttachmentDaoImpl类：该类实现了AttachmentDao接口，并继承BaseDaoHibernate4；

图 5-2-6 为 AttachmentDaoImpl 类的设计类图，具体的实现另见毕业设计附件

/EEMS/src下cn.edu.xzit.mailbox.dao.impl.AttachmentDaoImpl类的代码。



图5-2-6 AttachmentDaoImpl类

### 5.2.7 UserContactsDaoImpl类

UserContactsDaoImpl 类 ： 该 类 实 现 了 UserContactsDao 接 口 ， 并 继 承

BaseDaoHibernate4类；

图5-2-7为UserContactsDaoImpl类的设计类图，具体的实现另见毕业设计附件

/EEMS/src下cn.edu.xzit.mailbox.dao.impl.UserContactsDaoImpl类的代码。

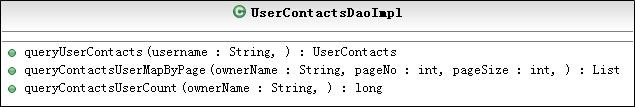


图5-2-7 UserContactsDaoImpl类

### 5.2.8 EmployeeDaoImpl类

EmployeeDaoImpl类：该类实现了EmployeeDao接口，并继承了BaseDaoHibernate4 类；

图5-2-8为EmployeeDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src 下cn.edu.xzit.mailbox.dao.impl.EmployeeDaoImpl类的代码。



图5-2-8 EmployeeDaoImpl类

### 5.2.9 CompanyDaoImpl类

CompanyDaoImpl类：该类实现了CompanyDao接口，并继承了BaseDaoHibernate4类；图5-2-9是CompanyDaoImpl类的设计类图，具体的实现另见毕业设计附件/EEMS/src 下cn.edu.xzit.mailbox.dao.impl.CompanyDaoImpl类的代码。



图5-2-9 CompanyDaoImpl类

## 5.3 本章小结

本章主要讲述了基于DAO模式的数据访问层的设计，即通过设计DAO组件去封装所有的数据库访问操作。数据访问层的设计包含DAO接口的设计与DAO组件的实现。此外，本系统所有的DAO组件都关联于Hibernate的SessionFactory对象，并且所有的DAO组件与Hibernate的SessionFactory都配置在Spring容器中，接受Spring容器的管理，实现了Spring与Hibernate的整合。

# 6业务逻辑层的设计

业务逻辑层也称为Service层，即主要为系统控制器提供相应服务的业务层次。在经典的Java EE项目中，所有的业务逻辑方法都在Service层实现，然后由MVC框架的系统控制器去调用Service层的业务逻辑方法处理用户的请求。

## 6.1 业务接口的设计

与DAO层相同的是，Service层也要实现与系统控制层的解耦，同样采用面向接口的设计方法。即先定义业务接口（Service接口），然后实现业务接口去完成Service组件的设计。所有的MVC框架的系统控制器都关联于业务接口，而不是关联与具体的业务逻辑实现类。这样就完成了业务逻辑层与系统控制层之间的分离。

### 6.1.1 登录验证的接口

LoginService接口：提供登录验证服务；

LoginService接口包含以下方法：

根据用户名（邮箱名）和密码判断用户是否登录成功；根据以上描述，可得到该接口的设计类图（图6-1-1）。



图6-1-1 LoginService接口

### 6.1.2 用户注册的接口

RegisterService接口：提供用户注册服务；

RegisterService接口包含以下方法：

根据给定的注册参数完成用户注册，返回是否注册成功；根据以上描述，可得到该接口的设计类图（图6-1-2）。



图6-1-2 RegisterService接口

### 6.1.3 加密服务的接口

EncryptService接口：提供加密服务；

EncryptService接口包含以下方法：

对给定的字符串进行加密，返回加密后的字符串；

根据以上描述，可得到该接口的设计类图（图6-1-3）。



图6-1-3 EncryptService接口

### 6.1.4 查询邮件的接口

SearchMailsService接口：提供查询邮件的服务；

SearchMailsService接口包含以下方法：

* 根据邮件ID查询邮件；
* 根据所有者（邮箱名）分页查询邮件；
* 根据所有者（邮箱名）分页查询已删除邮件；
* 根据所有者（邮箱名）分页查询草稿邮件；
* 根据所有者（邮箱名）分页查询已发送邮件； 根据所有者（邮箱名）分页查询已接收邮件；

根据以上描述，可得到该接口的设计类图（图6-1-4）。

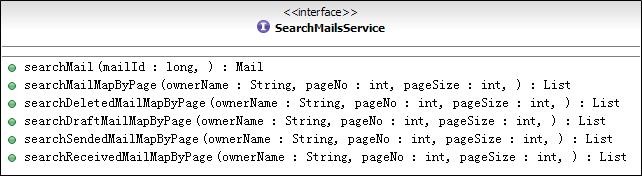


图6-1-4 SearchMailsService接口

### 6.1.5 查询邮件数量的接口

SearchMailsCountService接口：提供查询邮件数量的服务；

SearchMailsCountService接口包含以下方法：

* 根据所有者查询邮件数量；
* 根据所有者查询已删除邮件的数量；
* 根据所有者查询草稿邮件的数量；
* 根据所有者查询已发送邮件的数量；
* 根据所有者查询已接收邮件数量；

根据以上描述，可得到该接口的设计类图（图6-1-5）。

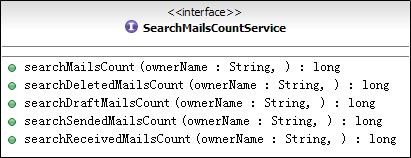


图6-1-5 SearchMailsCountService接口

### 6.1.6 邮件内容的服务接口

MailContentService接口：提供查询邮件内容的服务；

MailContentService接口包含以下方法：

* 根据邮件Id查询邮件内容；
* 根据给定的邮件内容转换成字符串；根据以上描述，可得到该接口的设计类图（图6-1-6）。



图6-1-6 SearchContentService接口

### 6.1.7 保存或更新邮件的接口

SaveOrUpdateMailService接口：提供保存或更新邮件的接口；

SaveOrUpdateMailService接口包含以下方法：

根据给定的邮件Map保存或更新邮件；根据以上描述，可得到该接口的设计类图（图6-1-7）。



图6-1-7 SaveOrUpdateMailService接口

### 6.1.8 保存草稿的接口

SaveDraftService接口：提供保存草稿的服务；

SaveDraftService接口包含以下方法：

根据给定的邮件参数保存草稿；根据以上描述，可得到该接口的设计类图（图6-1-8）。



图6-1-8 SaveDraftService接口

### 6.1.9 发送邮件的接口

SendMailService接口：提供发送邮件的服务；

SendMailService接口包含以下方法：

根据给定的邮件参数保存发送者的邮件； 根据给定的邮件参数保存接受者的邮件；

根据以上描述，可得到该接口的设计类图（图6-1-9）。



图6-1-9 SendMailService接口

### 6.1.10 上传附件的接口

UploadService接口：提供上传附件的服务；

UploadService接口包含以下方法：

* 根据邮件上传给定的附件；
* 根据邮件上传给定的附件集；

根据以上描述，可得到该接口的设计类图（图6-1-10）。



图6-1-10 UploadService接口

### 6.1.11 删除邮件的接口

DeleteMailService接口：提供删除邮件的服务；

DeleteMailService接口包含以下方法：

* 根据邮件ID将邮件移到已删除；
* 根据邮件ID永久删除邮件；

根据以上描述，可得到该接口的设计类图（图6-1-11）。



图6-1-11 DeleteMailService接口

### 6.1.12 还原已删除邮件的接口

RestoreMailService接口：提供还原已删除邮件的服务；

RestoreMailService接口包含以下方法：

根据邮件Id还原已删除邮件；根据以上描述，可得到该接口的设计类图（图6-1-12）。



图6-1-12 RestoreMailService接口

### 6.1.13 查询通讯录的接口

SearchContactsService接口：提供查询个人通讯录的服务；

SearchContactsService接口包含以下方法：

根据用户名（邮箱名）分页查询常用联系人； 查询当前用户的个人通讯录下的人数；根据以上描述，可得到该接口的设计类图（图6-1-13）。



图6-1-13 SearchContactsService接口

### 6.1.14 添加通讯录联系人的接口

AddContactsUserService接口：提供添加通讯录联系人的服务；

AddContactsUserService接口包含以下方法：

将给定邮箱名的联系人添加到当前用户的个人通讯录中；根据以上描述，可得到该接口的设计类图（图6-1-14）。



图6-1-14 AddContactsUserService接口

### 6.1.15 移除通讯录联系人的接口

RemoveContactsUserService接口：提供移除通讯录联系人的服务；

RemoveContactsUserService接口包含以下方法;

将给定邮箱名的联系人从当前用户的个人通讯录中移除；根据以上描述，可得到该接口的设计类图（图6-1-15）。



图6-1-15 RemoveContactsUserService接口

## 6.2 实现Service组件

完成了业务接口的设计后，接下来就可以实现业务接口完成业务组件（Service组件）

的设计。为了实现Spring与Struts2的整合，所有的Service组件都要配置在Spring容器中，然后由Spring容器给系统控制器（Action组件）注入Service组件，完成了系统控制器对业务组件的依赖。

### 6.2.1 登录验证的服务

LoginServiceImpl类：该类是LoginService接口的实现类；与之关联的接口：UserDao；

表6-2-1 LoginServiceImpl类

|  |
| --- |
| class: LoginServiceImpl |
| private: userDao : UserDao |
| public: validate(username : String, password : String) : boolean  ...//省略所有的setter和getter方法 |

由表6-2-1可以看出，该类实现了LoginService的validate方法。该方法的时序图见图6-2-1：



图6-2-1 validate方法的时序图

### 6.2.2 用户注册的服务

RegisterServiceImpl类：该类是RegisterService接口的实现类；与之关联的接口：EncryptService，UserDao, MailboxDao, User, Mailbox,

表6-2-2 RegisterServiceImpl类

|  |
| --- |
| class: RegisterServiceImpl |
| private: encryptService : EncryptService private: userDao : UserDao private: mailboxDao : MailboxDao  private: user : User private: mailbox : Mailbox |
| public: register(registerParams : Map<String, String>) : boolean ...//省略所有的setter和getter方法 |

由表6-2-2可以看出，该类实现了RegisterService的register方法。该方法的时序图见图6-2-2：

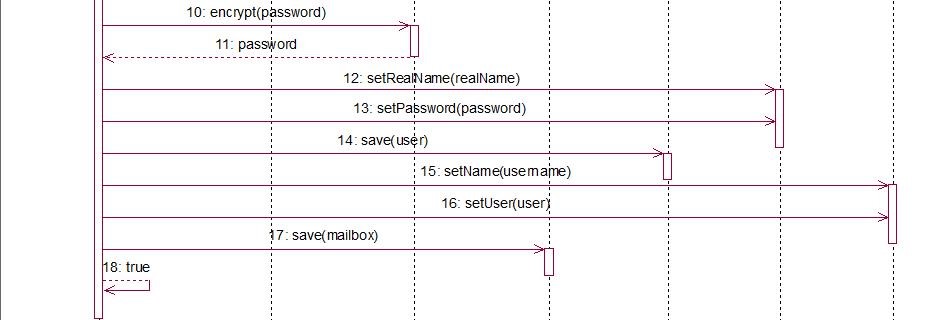
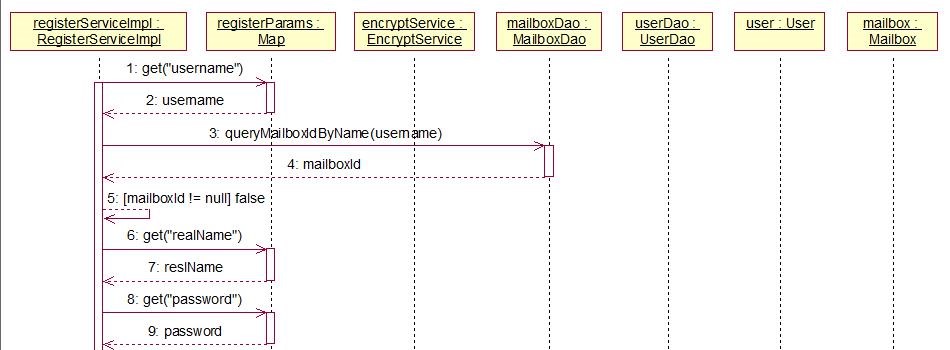


图6-2-2 register方法的时序图

### 6.2.3 加密服务

EncryptServiceImpl类：该类是EncryptService接口的实现类；

表6-2-3 EncryptServiceImpl类

|  |
| --- |
| class: EncryptServiceImpl |
|  |
| public: encrypt(msg : String) : String |

由表6-2-3可以看出，该类实现了EncryptService的encrypt方法。该方法的时序图见图6-2-3：

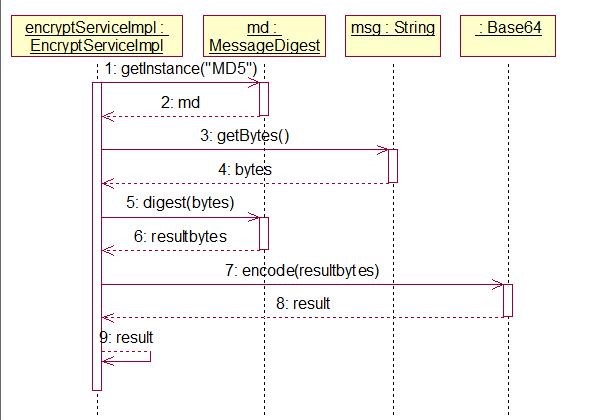


图6-2-3 encrypt方法的时序图

### 6.2.4 查询邮件的服务

SearchMailsServiceImpl类：该类是SearchMailsService接口的实现类；与之关联的接口：MailDao

表6-2-4 SearchMailsServiceImpl类

|  |
| --- |
| class: SearchMailsServiceImpl |
| private: mailDao : MailDao |
| public: searchMail(mailId : long) : Mail  public: searchMailMapByPage(ownername : String, pageNo : int, pageSize : int) : list public: searchDeletedMailMapByPage(ownername : String, pageNo : int, pageSize : int) : list public: searchDraftMailMapByPage(ownername : String, pageNo : int, pageSize : int) : list public: searchSendedMailMapByPage(ownername : String, pageNo : int, pageSize : int) : list public: searchReceivedMailMapByPage(ownername : String, pageNo : int, pageSize : int) : list ...//省略所有的setter和getter方法 |

由表6-2-4可以看出该类实现了SearchMailsService的6个方法。

以下分别该6个方法时序图；

其中searchMail方法的时序图见图6-2-4-1：

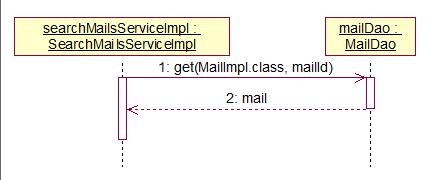


图6-2-4-1 searchMail方法的时序图

其中searchMailMapByPage方法的时序图见图6-2-4-2：

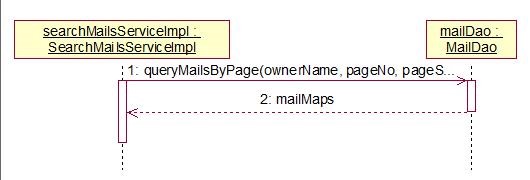


图6-2-4-2 searchMailMapByPage方法的时序图其中searchDeletedMailMapByPage方法的时序图见图6-2-4-3：

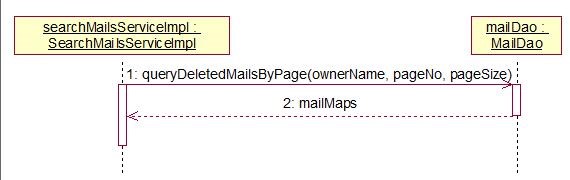


图6-2-4-3 searchDeletedMailMapByPage方法的时序图

其中searchDraftMailMapByPage方法的时序图见图6-2-4-4：

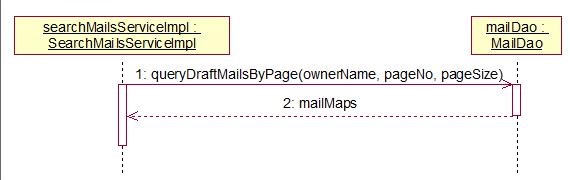


图6-2-4-4 searchDraftMailMapByPage方法的时序图其中searchSendedMailMapByPage方法的时序图见图6-2-4-5：



图6-2-4-5 searchSendedMailMapByPage方法的时序图

其中searchReceivedMailMapByPage方法的时序图见图6-2-4-6：

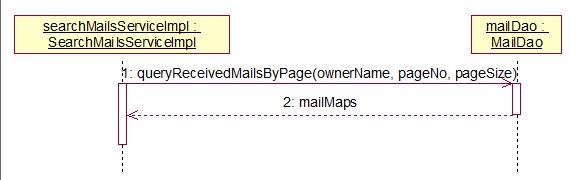


图6-2-4-6 searchReceivedMailMapByPage方法的时序图

### 6.2.5 查询邮件数量的服务

SearchMailsCountServiceImpl类：该类是SearchMailsCountService接口的实现类；与之关联的接口：MailDao

表6-2-5 SearchMailsCountServiceImpl类

|  |
| --- |
| class: SearchMailsCountServiceImpl |
| private: mailDao : MailDao |
| public: searchMailsCount(ownerName: String) : long; public: searchDeletedMailsCount(ownerName: String) : long; public: searchDraftMailsCount(ownerName: String) : long; public: searchSendedMailsCount(ownerName: String) : long;  public: searchReceivedMailsCount(ownerName: String) : long;  ...//省略所有的setter和getter方法 |

由表6-2-5可知，该类实现了SearchMailsCountService的5个方法。

以下分别是该5个方法时序图；

其中searchMailsCount方法的时序图见图6-2-5-1：

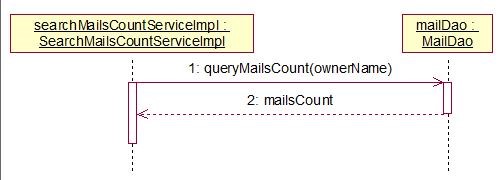


图6-2-5-1 searchMailsCount方法的时序图

其中searchDeletedMailsCount方法的时序图见图6-2-5-2：

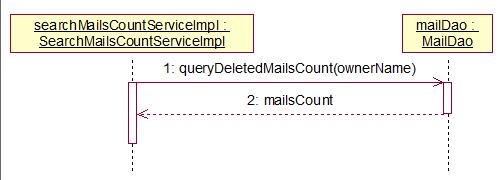


图6-2-5-2 searchDeletedMailsCount方法的时序图

其中searchDraftMailsCount方法的时序图见图6-2-5-3：

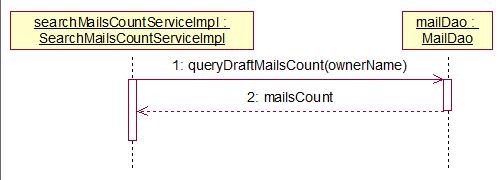


图6-2-5-3 searchDraftMailsCount方法的时序图其中searchSendedMailsCount方法的时序图见图6-2-5-4：

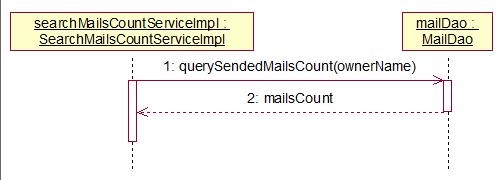


图6-2-5-4 searchSendedMailsCount方法的时序图其中searchReceivedMailsCount方法的时序图见图6-2-5-5：

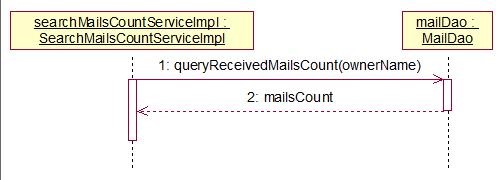


图6-2-5-5 searchReceivedMailsCount方法的时序图

### 6.2.6 邮件内容的服务

MailContentServiceImpl类：该类是MailContentService接口的实现类；与之关联的接口：ContentDao

表6-2-6 MailContentServiceImpl类

|  |
| --- |
| class: MailContentServiceImpl |
| private: contentDao: ContentDao |
| public: searchContent(ownerName: String) : Content; public: convertToString(content: Content) : String; ...//省略所有的setter和getter方法 |

由表6-2-6可知，该类实现了SearchContentService接口的 searchContent方法与

convertToString方法。其中searchContent方法的时序图见图6-2-6-1:

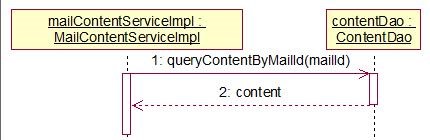


图6-2-6-1 searchContent方法的时序图

其中convertToString方法的时序图见图6-2-6-2:

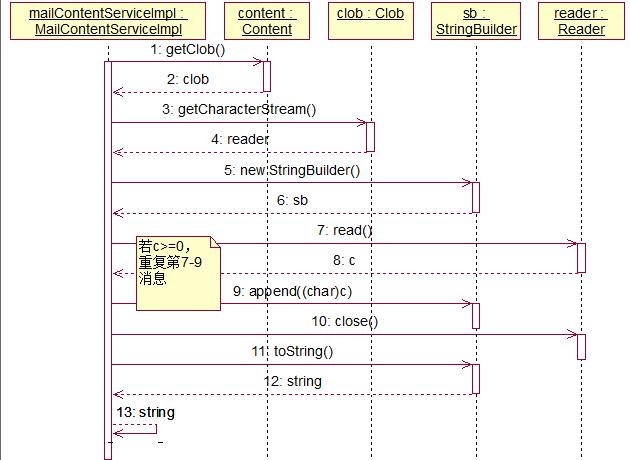


图6-2-6-2 convertToString方法的时序图

### 6.2.7 保存或更新邮件的服务

SaveOrUpdateMailServiceImpl类：该类是SaveOrUpdateMailService接口的实现类；与之关联的接口：ContentDao, MailDao

表6-2-7 SaveOrUpdateMailServiceImpl类

|  |
| --- |
| class: SaveOrUpdateMailServiceImpl |
| private: mailDao : MailDao private: contentDao : ContentDao |
| public:saveOrUpdateMail(mailMap: Map) : Content; ...//省略所有的setter和getter方法 |

由表6-2-7可知，该类实现了SaveOrUpdateMailService接口的SaveOrUpdateMail 方法。

SaveOrUpdateMail方法的时序图见图6-2-7:

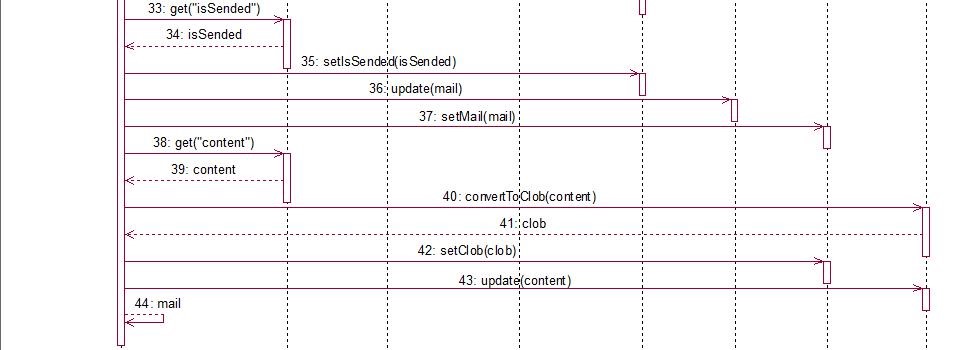
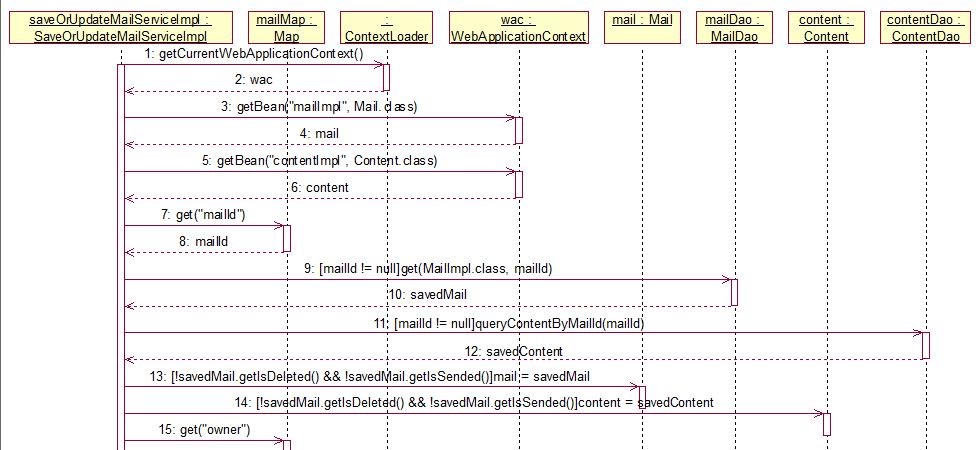


图6-2-7 SaveOrUpdateMail方法的时序图

6.2.8 保存草稿的服务

SaveDraftServiceImpl类：该类是SaveDraftService接口的实现类；与之关联的接口：SaveOrUpdateMailService, MailboxDao

表6-2-8 SaveDraftServiceImpl类

|  |
| --- |
| class: SaveDraftServiceImpl |
| private: saveOrUpdateMailService : SaveOrUpdateMailService private: mailboxDao : MailboxDao |
| public:saveDraft(mailParams: Map) : Mail; ...//省略所有的setter和getter方法 |

由表6-2-8可知，该类实现了SaveDraftService接口的saveDraft方法， saveDraft方法的时序图见图6-2-8:

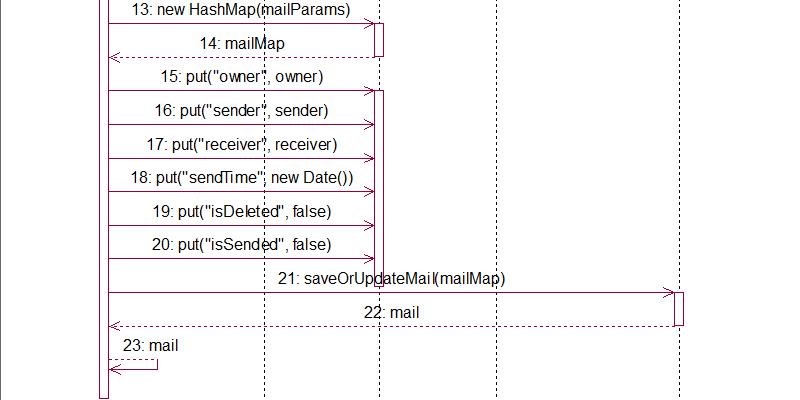
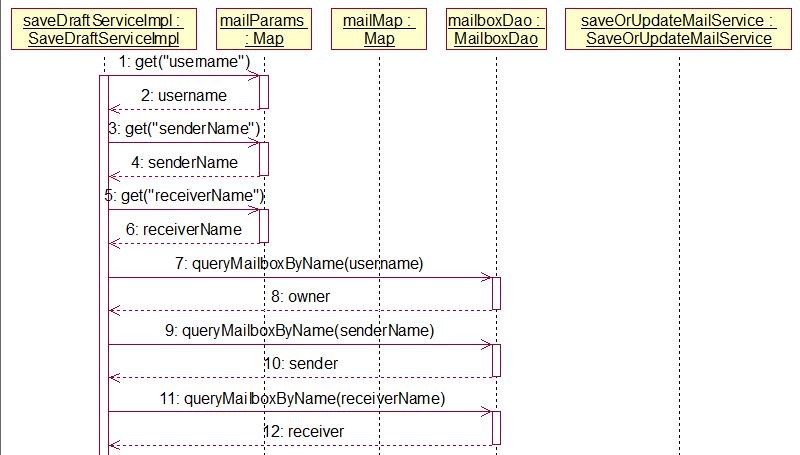


图6-2-8 saveDraft方法的时序图

### 6.2.9 发送邮件的服务

SendMailServiceImpl类：该类是SendMailService接口的实现类；与之关联的接口：SaveOrUpdateMailService, MailboxDao

表6-2-9 SendMailServiceImpl类

|  |
| --- |
| class: SendMailServiceImpl |
| private: saveOrUpdateMailService : SaveOrUpdateMailService private: mailboxDao : MailboxDao |
| public: saveOrUpdateMail(mailParams: Map, owner: Mailbox) : Mail; public: saveOrUpdateSenderMail(mailParams: Map) : Mail;  public: saveReceiverMail(mailParams: Maps) : Mail;  ...//省略所有的setter和getter方法 |

由表6-2-9可知，该类实现了SendMailService的saveOrUpdateSenderMail方法与

saveReceiverMail方法。其中saveOrUpdateSenderMail方法的时序图见图6-2-9-1。

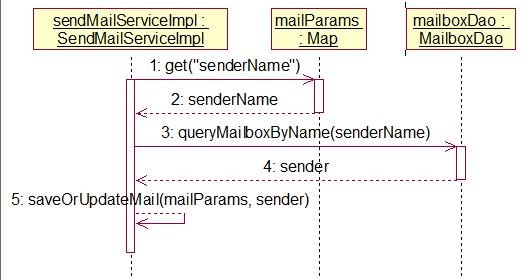


图6-2-9-1 saveOrUpdateSenderMail方法的时序图

其中saveReceiverMail方法的时序图见图6-2-9-2。

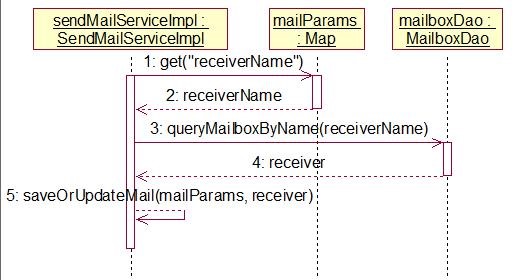


图6-2-9-2 saveReceiverMail方法的时序图

由图6-2-9-1与图6-2-9-2可看出，SendMailServiceImpl类实现的方法都调用了本

类的saveOrUpdateMail方法。 saveOrUpdateMail方法的时序图见图6-2-9-3。

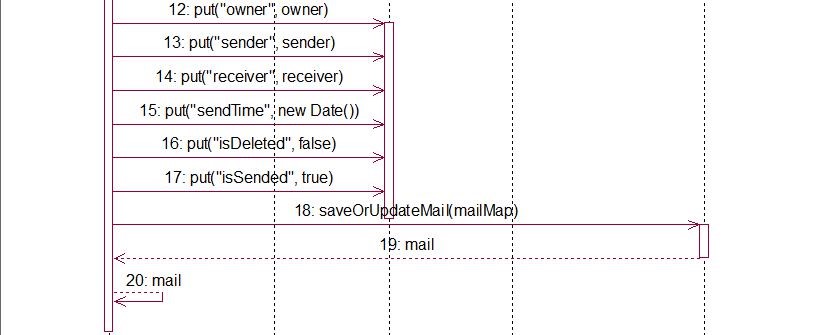
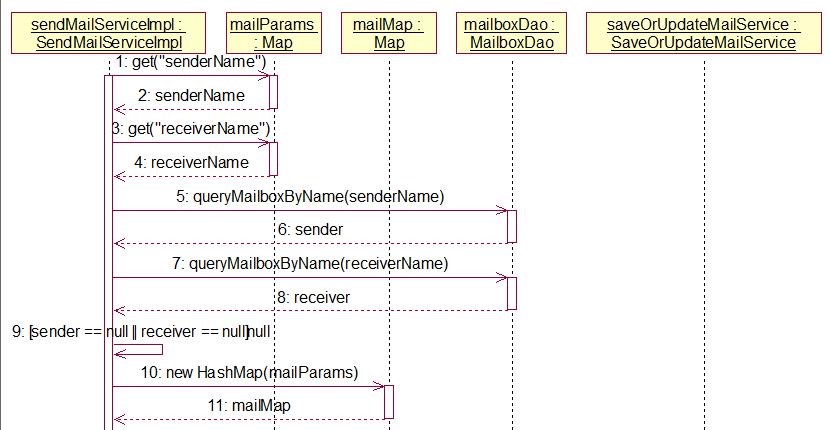


图6-2-9-3 saveOrUpdateMail方法的时序图

### 6.2.10 上传附件的服务

UploadServiceImpl类：该类是UploadService接口的实现类；与之关联的接口：AttachmentDao

表6-2-10 UploadServiceImpl类

|  |
| --- |
| class: UploadServiceImpl |
| private: attachmentDao: AttachmentDao |
| public: upload(upload: File, uploadCotentType: String, uploadFileName: String, saveDirectory: String, mail: Mail); public: upload(uploads: List<File>, uploadCotentTypes: List<String>, uploadFileNames:  List<String>, saveDirectorys: List<String>, mail: Mail);  ...//省略所有的setter和getter方法 |

由表6-2-10可知，该类实现了UploadService的两个重载的upload方法。其中第一个upload方法的时序图见图6-2-10-1。

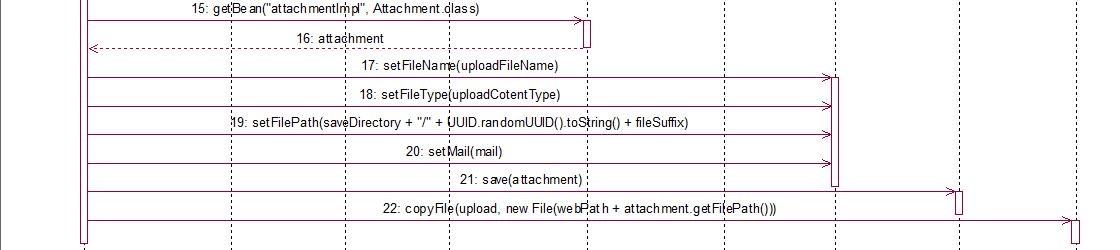
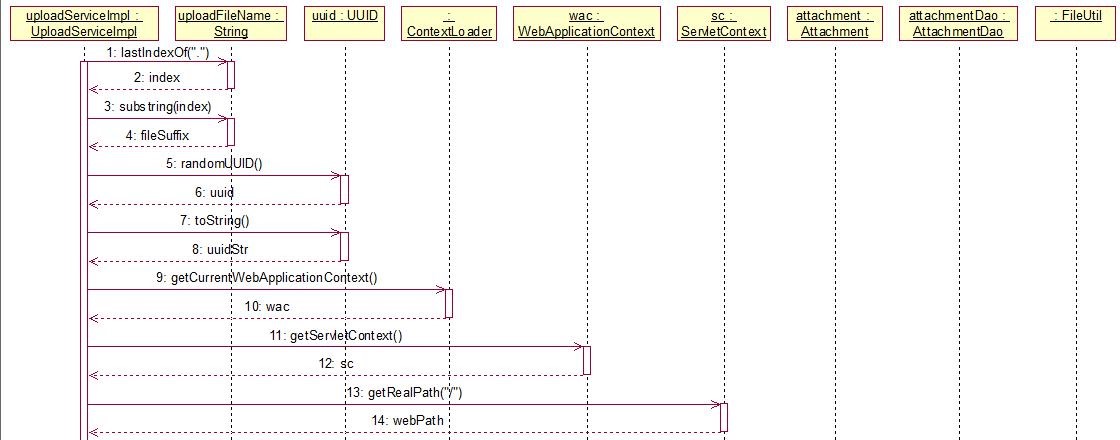


图6-2-10-1第一个upload方法的时序图

### 6.2.11 删除邮件的服务

DeleteMailServiceImpl类：该类是DeleteMailService接口的实现类；与之关联的接口：MailDao, ContentDao, AttachmentDao

表6-2-11 DeleteMailServiceImpl类

|  |
| --- |
| class: DeleteMailServiceImpl |
| private: mailDao : MailDao private: contentDao : ContentDao private: attachmentDao: AttachmentDao |
| public: deleteMail(mailId: long);  public: deleteMailPermanently(mailId: long);  ...//省略所有的setter和getter方法 |

由表 6-2-11 可知，该类实现了 DeleteMailService 的 deleteMail 方法与 deleteMailPermanently方法。其中deleteMail方法的时序图见图6-2-11-1。

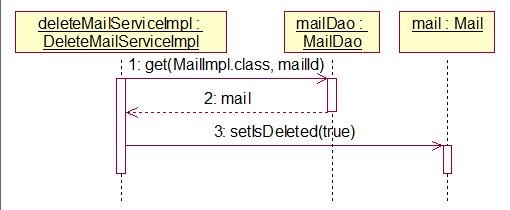


图6-2-11-1 deleteMail方法的时序图

其中deleteMailPermanently方法的时序图见图6-2-11-2。

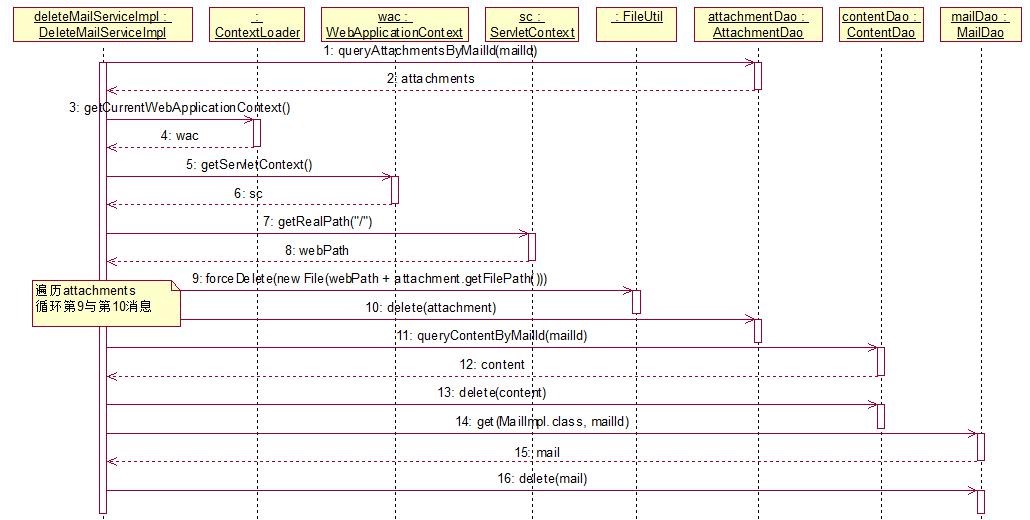


图6-2-11-2 deleteMailPermanently方法的时序图

### 6.2.12 还原已删除邮件的服务

RestoreMailServiceImpl类：该类是RestoreMailService接口的实现类；与之关联的接口：MailDao

表6-2-12 RestoreMailServiceImpl类

|  |
| --- |
| class: RestoreMailServiceImpl |
| private: mailDao: MailDao |
| public: restoreMail(mailId: long);  ...//省略所有的setter和getter方法 |

由表6-2-12可知，该类实现了RestoreMailService接口的restoreMail方法。 restoreMail方法的时序图见图6-2-12。

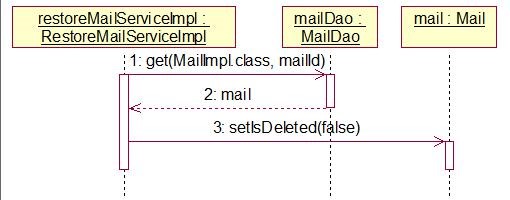


图6-2-12 restoreMail方法的时序图

### 6.2.13 查询通讯录的服务

SearchContactsServiceImpl类：该类是SearchContactsService接口的实现类；与之关联的接口：UserContactsDao

表6-2-13 SearchContactsServiceImpl类

|  |
| --- |
| class: SearchContactsServiceImpl |
| private: userContactsDao: UserContactsDao |
| public: searchContactsUserMap(username: String,pageNo: int,pageSize: int): List; public:searchContactsUserCount(username: String): Long  ...//省略所有的setter和getter方法 |

由表6-2-13可知，该类实现了RestoreMailService接口的searchContactsUserMap

方法与searchContactsUserCount方法。其中searchContactsUserMap方法的时序图见图6-2-13-1。



图6-2-13-1 searchContactsUserMap方法的时序图

其中searchContactsUserCount方法的时序图见图6-2-13-2。

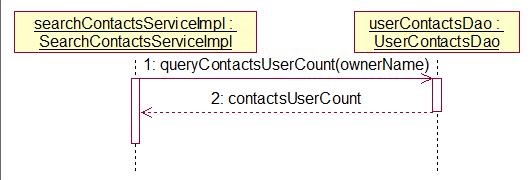


图6-2-13-2 searchContactsUserCount方法的时序图

### 6.2.14 添加通讯录联系人的服务

AddContactsUserServiceImpl类：该类是AddContactsUserService接口的实现类；与之关联的接口：UserDao, UserContactsDao

表6-2-14 AddContactsUserServiceImpl类

|  |
| --- |
| class: AddContactsUserServiceImpl |
| private: userDao: UserDao  private: userContactsDao: UserContactsDao |
| public: addContactsUser(username: String, mailboxName: String): boolean; ...//省略所有的setter和getter方法 |

由表6-2-14可知，该类实现了AddContactsUserService接口的addContactsUser方法。 addContactsUser方法的时序图见图6-2-14。

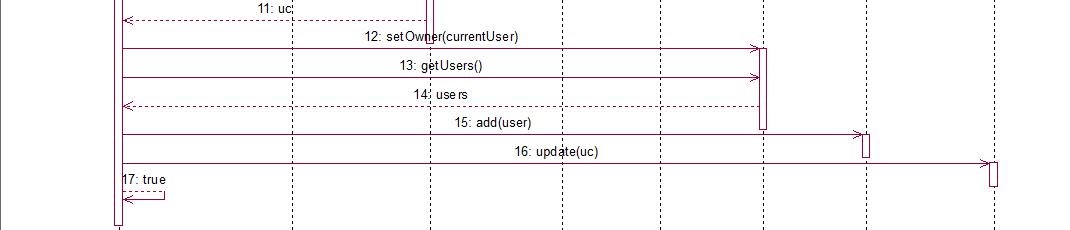


图6-2-14 addContactsUser方法的时序图

### 6.2.15 移除通讯录联系人的服务

RemoveContactsUserServiceImpl类：该类是RemoveContactsUserService接口的实现类；

与之关联的接口：UserDao

表6-2-15 RemoveContactsUserServiceImpl类

|  |
| --- |
| class: RemoveContactsUserServiceImpl |
| private: userDao: UserDao |
| public: removeContactsUser(username: String, mailboxName: String): boolean; ...//省略所有的setter和getter方法 |

由 表 6-2-15 可 知 ， 该 类 实 现 了 RemoveContactsUserService 接 口 的 removeContactsUser方法。

removeContactsUser方法的时序图见图6-2-15。



图6-2-15 removeContactsUser方法的时序图

## 6.3 配置Spring的事务处理

本系统业务逻辑层中的每个业务逻辑方法几乎都包含一系列对数据库的操作，这些操作可能会修改数据库中的多条记录。那么这一系列的操作应该是一个整体，而不是仅完成一部分操作而修改数据库其中的部分记录。因此，本系统需要为每一个业务逻辑方法添加事务处理机制，从而保证了数据库数据的完整性。

此外，本系统需要实现业务逻辑代码与事务操作代码的解耦，不会直接向业务方法中添加事务操作代码。而是采用Spring的AOP机制，在业务方法中动态织入Spring的事务处理，从而实现了业务逻辑方法的事务增强处理。

为了实现这一目的，首先需要在Spring容器中配置Hibernate的局部事务管理器。然后需要在Spring容器中配置事务增强处理的切面Bean，并且该切面Bean必须指定 Hibernate的局部事务管理器，即本系统所有的业务方法都需要动态添加Hibernate的局部事务处理操作。最后还需要在Spring容器中配置aop元素，并指定所有的业务操作方法为事务增强处理的切入点，在切入点中动态织入事务处理切面Bean的增强处理。其实质是动态生成AOP的事务代理对象，然后调用事务代理对象的业务方法实现业务组件的事务增强处理（见图6-3）。事务配置的详细代码另见毕业设计附件/EEMS/WebRoot路径下的applicationContext.xml文件。

Spring事务配置的原理图见图6-3。

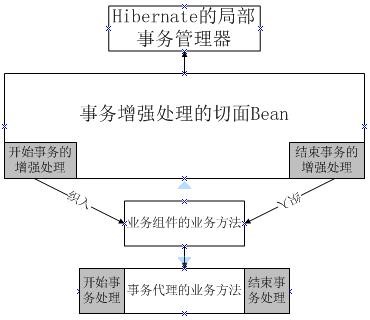


图6-3 Spring事务配置的原理图

## 6.4 本章小结

本章主要讲述了基于Service组件的业务逻辑层的设计。业务逻辑层的设计包含业务接口的设计，业务组件（Service组件）的实现两方面。此外，本章还介绍了基于Spring

AOP的事务处理机制，通过配置Spring的事务处理，实现了业务组件的事务增强。

# 7系统控制层的设计

系统控制层也称为系统Web层，通常采用控制器（Controller）实现。在经典的Java EE项目中，控制器主要由Servlet实现。根据MVC的思想，浏览器向Web服务器发起请求首先要匹配到特定的控制器，然后由特定的控制器调用特定的模型组件（Model）来处理用户的请求，最后由特定的控制器返回指定的视图页面（View）响应用户的请求。从以上的MVC模式中可以看出，无论是哪一个流程，控制器都参与其中，并起到了最核心的作用。

本系统采用Struts2框架实现Web应用的MVC分层设计。根据MVC的思想，本系统采用JSP页面充当Web应用的视图层，由业务逻辑组件，DAO组件，持久化对象充当Web应用的模型层，由Struts2的StrutsPrepareAndExecutorFilter充当核心控制器，把开发的Action组件充当系统控制器。核心控制器主要负责拦截用户从浏览器发出的请求，并且根据Struts2的配置文件匹配到对应的系统控制器，然后由系统控制器调用模型组件处理用户的请求。

为了实现Spring整合Struts2,所有的系统控制器也应该由Spring容器去管理，因此，所有的Action组件都要配置在Spring容器中。然后由Struts2的核心控制器去调用Spring 容器中特定的Action组件来处理用户的请求。为实现这一目的，需要借助Struts2整合 Spring的插件：struts2-spring-plugin-2.3.16.1.jar。借助该插件，可实现Struts2的核心控制器与系统控制器的解耦，完成了Spring与Struts2整合。

## 7.1 Action组件的设计

系统中所有的Action组件都包含一个execute方法。当Struts2的核心控制器匹配到特定的Action时，默认会调用该execute方法处理用户的请求。此外，本系统还将利用Struts2的内置的类型转换器与基于ONGL的类型转换机制完成HTTP请求参数的解析与请求参数从转换器到Action组件的传递。

### 7.1.1 登录验证的Action

LoginAction类：该类是登录验证的Action组件，继承了Struts2的ActionSupport 类。

LoginAction类应包含以下属性：

* loginService(登录服务);
* encryptService(加密服务);
* loginParams(登录请求的参数);

根据以上描述，可得到LoginAction的设计类图（表7-1-1）。

表7-1-1 LoginAction类

|  |
| --- |
| class: LoginAction |
| private: loginService : LoginService private: encryptService : EncryptService private: loginParams : Map<String, String> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

LoginAction中execute方法的时序图见图7-1-1：

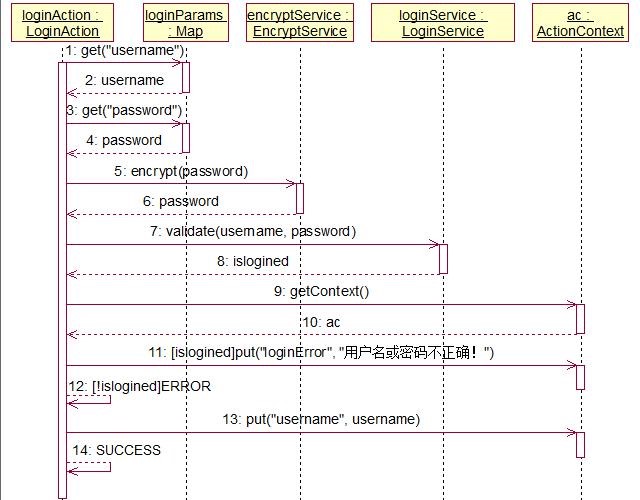


图7-1-1 LoginAction中execute方法的时序图

### 7.1.2 用户注册的Action

RegisterAction类：该类是用户注册的Action组件，继承Struts2的ActionSupport 类。

RegisterAction类应包含以下属性：

registerService(注册服务); registerParams (注册参数); 根据以上描述，可得到RegisterAction的设计类图（表7-1-2）。

表7-1-2 RegisterAction类

|  |
| --- |
| class: RegisterAction |
| private: registerService : RegisterService private: registerParams : Map<String, String> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

RegisterAction类中execute方法的时序图见图7-1-2：

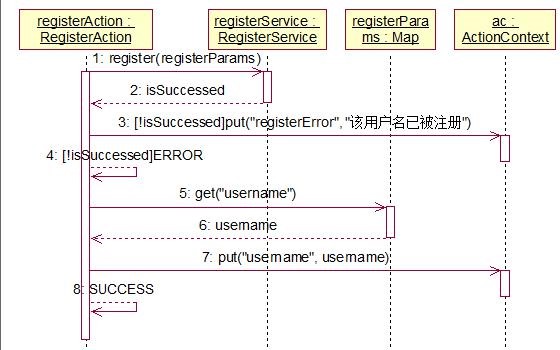


图7-1-2 RegisterAction类中execute方法的时序图

### 7.1.3 查询邮件的Action

SearchMailsAction类：该类是分页查询所有邮件的Action组件，继承了Struts2的

ActionSupport类。

SearchMailsAction类应包含以下属性：

* searchMailsService(查询邮件的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SearchMailsAction的设计类图（表7-1-3）。

表7-1-3 SearchMailsAction类

|  |
| --- |
| class: SearchMailsAction |
| private: searchMailsService : SearchMailsService private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchMailsAction中execute方法的时序图见图7-1-3：

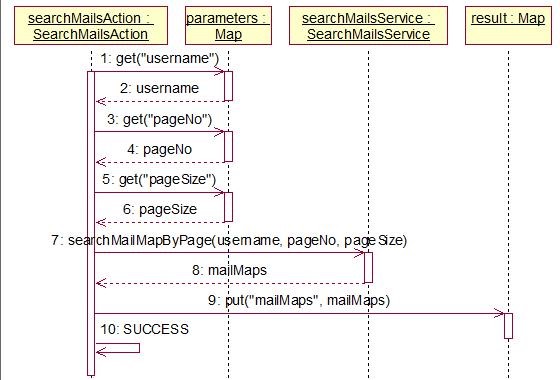


图7-1-3 SearchMailsAction中execute方法的时序图

注：本系统中查询已删除邮件的Action(SearchDeletedMailsAction)，查询草稿邮件

的Action(SearchDraftMailsAction)，查询已发送的邮件的Action(SearchSendedMails Action)，查询已接收邮件的Action(SearchReceivedMailsAction)都与本Action的设计非常相似。不同的仅仅是调用了searchMailsService对应的业务逻辑方法，为此本章不再赘述这四个Action组件的设计。

### 7.1.4 查询邮件数量的Action

SearchMailsCountAction 类：该类是查询所有邮件数量的 Action 组件，继承了

Struts2的ActionSupport类。

SearchMailsCountAction类应包含以下属性：

* searchMailsCountService(查询邮件数量的服务);
* username(封装当前用户名的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SearchMailsCountAction的设计类图（表7-1-4）。

表7-1-4 SearchMailsCountAction类

|  |
| --- |
| class: SearchMailsCountAction |
| private: searchMailsCountService: SearchMailsCountService private: username: String  private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchMailsCountAction中execute方法的时序图见图7-1-4：

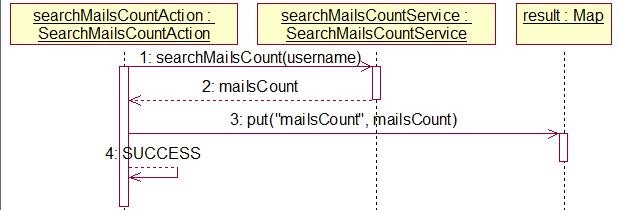


图7-1-4 SearchMailsCountAction中execute方法的时序图

注：本系统中查询已删除邮件数量的Action(SearchDeletedMailsCountAction)，查询草稿邮件数量的 Action(SearchDraftMailsCountAction)，查询已发送的邮件数量

Action(SearchSendedMailsCountAction)查询已接收邮件数量的Action(SearchReceived

MailsCountAction)都与本 Action 的设计非常相似。不同的仅仅是调用了 searchMailsCountService对应的业务逻辑方法，为此本章不在赘述这四个Action组件的设计。

### 7.1.5 查询邮件内容的Action

SearchContentAction 类：该类是查询邮件内容的 Action 组件，继承了

SearchMailsAction类。

SearchContentAction类应包含以下属性： MailContentService(邮件内容的服务); 根据以上描述，可得到SearchContentAction的设计类图（表7-1-5）。

表7-1-5 SearchContentAction类

|  |
| --- |
| class: SearchContentAction |
| private: mailContentService: MailContentService |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchContentAction中execute方法的时序图见图7-1-5：

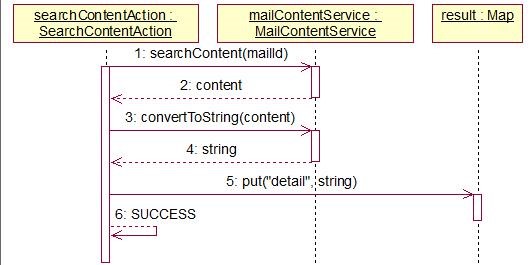


图7-1-5 SearchContentAction中execute方法的时序图

### 7.1.6 保存草稿的Action

SaveDraftAction 类：该类是保存草稿的 Action 组件，继承了 Struts2 的

ActionSupport类。

SaveDraftAction类应包含以下属性：

* saveDraftService(保存草稿的服务);
* mailParams(封装查询邮件的请求参数);
* result(保存结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SaveDraftAction的设计类图（表7-1-6）。

表7-1-6 SaveDraftAction类

|  |
| --- |
| class: SaveDraftAction |
| private: saveDraftService: SaveDraftService private: mailParams: Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SaveDraftAction中execute方法的时序图见图7-1-6。



图7-1-6 SearchMailsAction中execute方法的时序图

### 7.1.7 发送邮件的Action

SendMailAction类：该类是发送邮件的Action组件，继承了Struts2的ActionSupport 类。SendMailAction类应包含以下属性：

* sendMailService(发送邮件的服务);
* mailParams(封装查询邮件的请求参数);
* result(发送结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SendMailAction的设计类图（表7-1-7）。

表7-1-7 SendMailAction类

|  |
| --- |
| class: SendMailAction |
| private: sendMailService: SendMailService private: mailParams: Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchMailsAction中execute方法的时序图见图7-1-7：

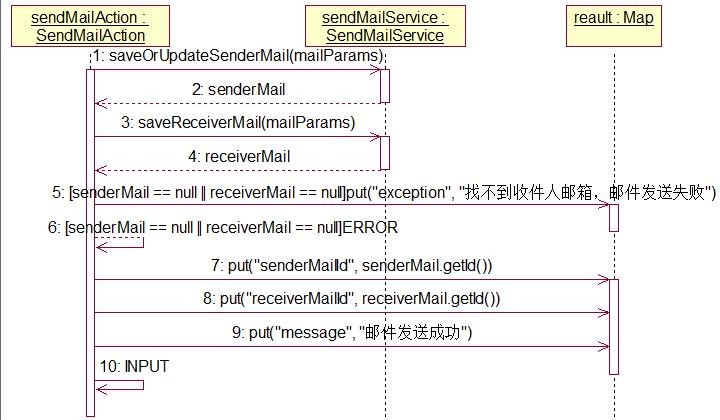


图7-1-7 SearchMailsAction中execute方法的时序图

### 7.1.8 删除邮件的Action

DeleteMailAction 类：该类是删除邮件的 Action 组件，继承了 Struts2 的

ActionSupport类。

DeleteMailAction类应包含以下属性：

* deleteMailService(删除邮件的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到DeleteMailAction的设计类图（表7-1-8）。

表7-1-8 DeleteMailAction类

|  |
| --- |
| class: DeleteMailAction |
| private: deleteMailService: DeleteMailService private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

DeleteMailAction中execute方法的时序图见图7-1-8：

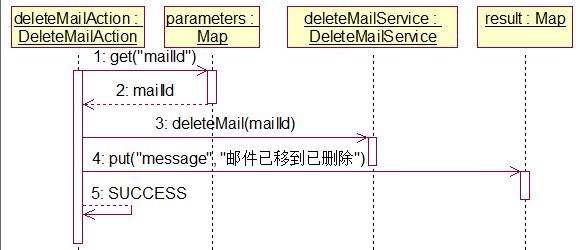


图7-1-8 DeleteMailAction中execute方法的时序图

注：本系统中永久删除邮件的Action(DeleteMailPermanentlyAction)与本Action的设计非常相似，不同的仅仅是调用了deleteMailService的deleteMailPermanently方法和封装了“邮件已永久删除”的信息返回结果。因此，本章不在赘述永久删除邮件的Action 组件的设计。

### 7.1.9 还原已删除邮件的Action

RestoreMailAction类：该类是还原已删除邮件的Action组件，继承了Struts2的

ActionSupport类。

RestoreMailAction类应包含以下属性：

* restoreMailService(还原已删除邮件的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到RestoreMailAction的设计类图（表7-1-9）。

表7-1-9 RestoreMailAction类

|  |
| --- |
| class: RestoreMailAction |
| private: restoreMailService: RestoreMailService private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

RestoreMailAction中execute方法的时序图见图7-1-9：



图7-1-9 RestoreMailAction中execute方法的时序图

### 7.1.10 查询个人通讯录人数的Action

SearchContactsUsersCountAction类：该类是还原已删除邮件的Action组件，继承了Struts2的ActionSupport类。

SearchContactsUsersCountAction类应包含以下属性：

* searchContactsService(查询个人通讯录的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SearchContactsUsersCountAction的设计类图（表7-1-10）。

表7-1-10 SearchContactsUsersCountAction

|  |
| --- |
| class: SearchContactsUsersCountAction |
| private: searchContactsService: SearchContactsService  private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchContactsUsersCountAction中execute方法的时序图见图7-1-10：

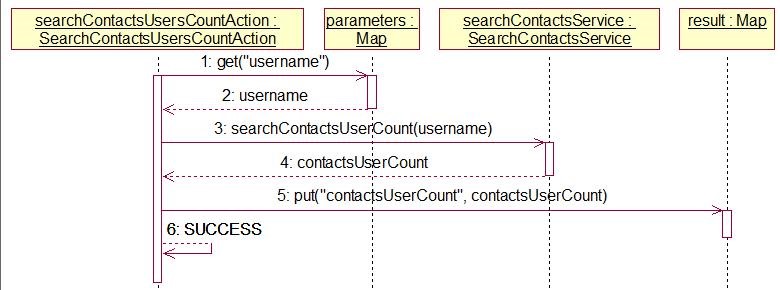


图7-1-10 SearchContactsUsersCountAction中execute方法的时序图

### 7.1.11 查询个人通讯录的Action

SearchContactsUsersAction类：该类是查询用户通讯录的Action组件，继承了 Struts2的ActionSupport类。

SearchContactsUsersAction类应包含以下属性：

* searchContactsService(查询个人通讯录的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到SearchContactsUsersAction的设计类图（表7-1-11）。

表7-1-11 SearchContactsUsersAction类

|  |
| --- |
| class: SearchContactsUsersAction |
| private: searchContactsService: SearchContactsService private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

SearchContactsUsersAction中execute方法的时序图见图7-1-11：

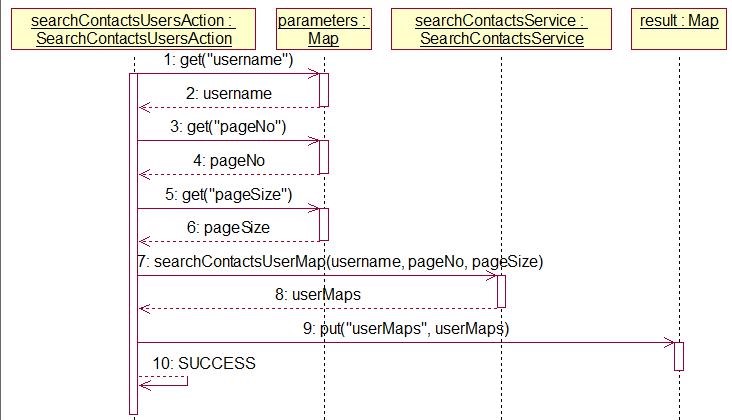


图7-1-11 SearchContactsUsersAction中execute方法的时序图

### 7.1.12 添加通讯录联系人的Action

AddContactsUserAction类：该类是查询用户通讯录的Action组件，继承了Struts2 的ActionSupport类。

AddContactsUserAction类应包含以下属性：

* AddContactsUserService(添加通讯录联系人的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到AddContactsUserAction的设计类图（表7-1-12）。

表7-1-12 AddContactsUserAction类

|  |
| --- |
| class: AddContactsUserAction |
| private: addContactsUserService: AddContactsUserService  private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

AddContactsUserAction中execute方法的时序图见图7-1-12：

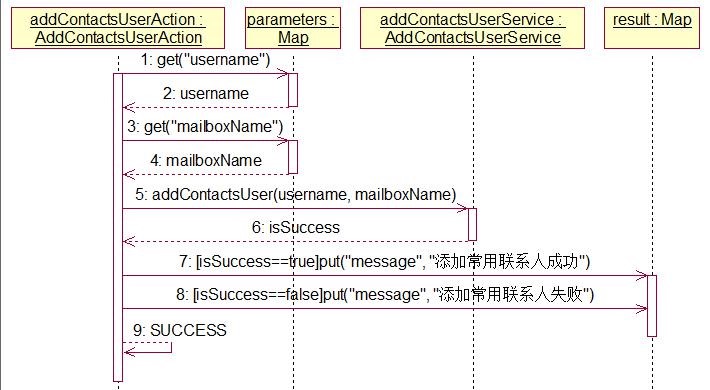


图7-1-12 AddContactsUserAction中execute方法的时序图

### 7.1.13 移除通讯录联系人的Action

RemoveContactsUserAction 类：该类是查询用户通讯录的 Action 组件，继承了

Struts2的ActionSupport类。

RemoveContactsUserAction类应包含以下属性：

* removeContactsUserService(移除通讯录联系人的服务);
* parameters(封装查询邮件的请求参数);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到RemoveContactsUserAction的设计类图（表7-1-13）。

表7-1-13 RemoveContactsUserAction类

|  |
| --- |
| class: RemoveContactsUserAction |
| private: removeContactsUserService: RemoveContactsUserService  private: parameters : Map<String, String> private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

RemoveContactsUserAction中execute方法的时序图见图7-1-13：

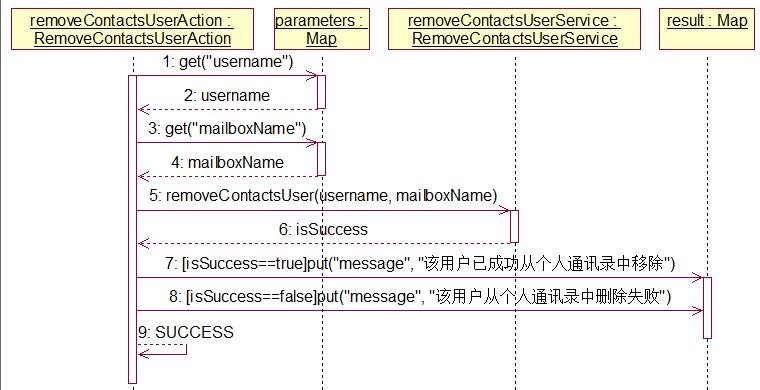


图7-1-13 RemoveContactsUserAction中execute方法的时序图

### 7.1.14 上传附件的Action

UploadFileAction类：该类是查询用户通讯录的Action组件，继承了Struts2的

ActionSupport类。

UploadFileAction类应包含以下属性：

* searchMailsService(查询邮件的服务);
* uploadService(上传附件的服务)；
* upload(上传的文件);
* uploadContentType(上传的文件类型);
* uploadFileName(上传的文件名称);
* saveDirectory(上传文件的保存目录);
* mailId(上传文件对应的邮件Id);
* result(查询结果，会被Struts2的json插件序列化成json字符串); 根据以上描述，可得到UploadFileAction的设计类图（表7-1-14）。

表7-1-14 UploadFileAction类

|  |
| --- |
| class: UploadFileAction |
| private: searchMailsService: SearchMailsService private: uploadService: UploadService private: upload: File  private: uploadContentType: String private: uploadFileName: String private: saveDirectory: String private: mailId: Long  private: result : Map<String, Object> |
| public: execute() : String  ...//省略了所有的setter和getter方法 |

UploadFileAction中execute方法的时序图见图7-1-14：

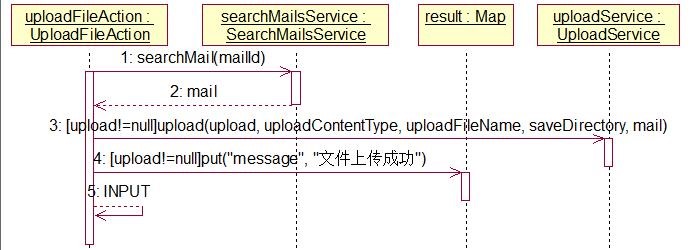


图7-1-14 UploadFileAction中execute方法的时序图

## 7.2 Action组件的配置

实现了Action组件后，接下来需要在struts.xml中配置Action组件了。配置Action 组件就是让Struts2的核心控制器知道那个Action组件处理那个用户的请求，并返回什么样的结果。即需要完成Action组件与用户请求，与返回结果的对应关系。可以说，Action 组件是Struts2的基本单位，也是系统控制层的基本单位。

此外，本系统为了实现Spring与Struts2的整合，采用了基于注解的组件扫描的方式将所有的Action组件都配置在Spring容器中。借助Struts2整合Spring的插件， struts.xml文件中所有的Action只需指定Spring容器中Action组件的Id即可完成

Action组件与用户请求之间的对应关系。

本章不会详细介绍本系统中每一个Action组件的配置，配置Acion组件的详细内容另见毕业设计附件/EEMS/src/路径下的struts.xml文件。

## 7.3 本章小结

本章主要讲述了基于Struts2框架的系统控制层的设计。系统控制层的设计的核心是 Action组件的设计。因此，本章对本系统中Action组件的设计内容做了详细介绍，采用类图描述Action组件的类模型，采用时序图来描述每个Acion组件的处理用户请求的方法。最后，本章还略带讲述了Action组件的配置等内容。

# 8视图层的设计与系统实现

## 8.1 视图层的设计

视图层主要用于向用户展现用户界面与数据，是MVC设计中的一部分。本系统主要采用JSP技术，结合HTML、CSS、JavaScript技术向用户展现用户界面。

此外，本系统还会采用两种技术发送两种请求。一是采用Struts2的标签库技术简化用户界面的设计，发送普通的用户请求，并结合Struts2的ONGL表达式与数据标签呈现用户数据。本系统中的登录和注册页面就采用这种设计方式来发送登录请求与注册请求。二是结合JQuery技术简化JavaScript的设计，并结合提供Ajax支持发送Ajax请求，返回JSON类型的数据。

本系统的视图层设计了5个JSP页面。

* exception.jsp(异常处理页面);
* login.jsp(登录页面);
* register.jsp(注册页面);
* registerSuccess.jsp(注册成功页面);
* main.jsp(主页面，本系统的主要操作都集中在该页面完成);

本章不会详细介绍着5个JSP页面的设计。关于其设计内容请另见毕业设计附件

/EEMS/WebRoot/WEB-INF/content的目录下的JSP文件。

## 8.2 系统的功能演示

### 8.2.1 用户登录

当用户在浏览器地址栏输入http://localhost:8080/EEMS/后，系统会发送进入登录页面的请求，进入登录页面。本系统的登录页面见图8-2-1-1。

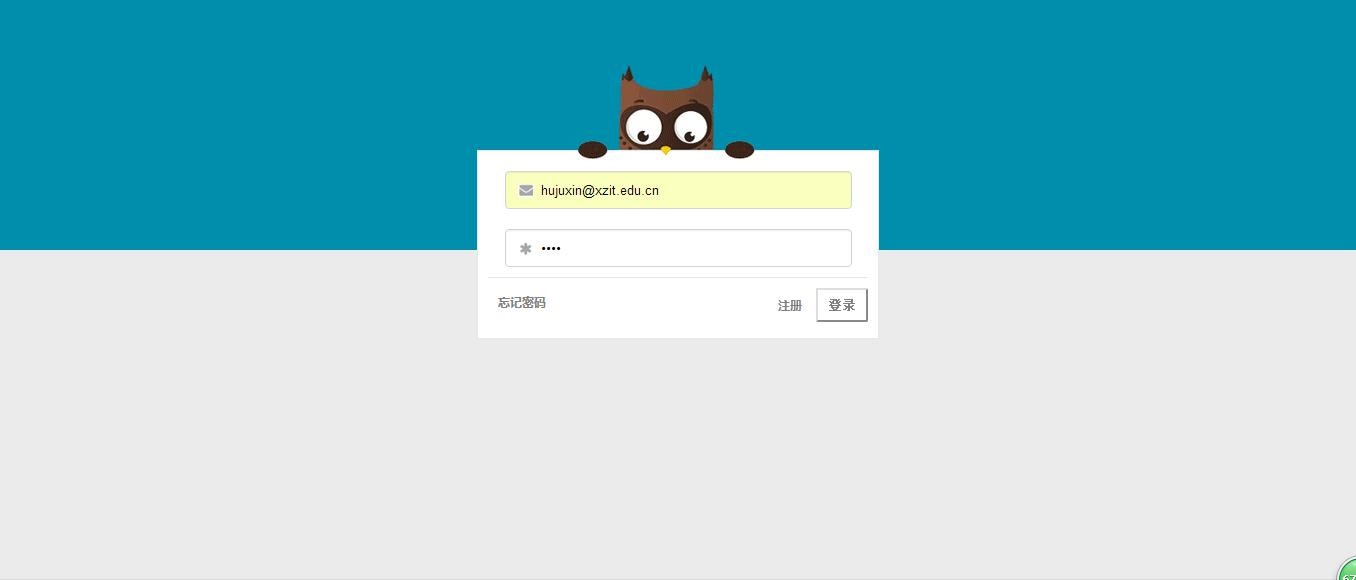


图8-2-1-1 系统的登录页面

当用户输入正确的用户名和密码后，点击登录按钮，系统会发出登录请求。登录成功后，进入系统的主页面。本系统的主页面见图8-2-1-2。



图8-2-1-2 系统的主页面

### 8.2.2 用户注册

当用户在登录页面（图8-2-1-1）点击注册按钮时，系统会发出进入注册页面的请求，进入注册页面。本系统的注册页面见图8-2-2-1.

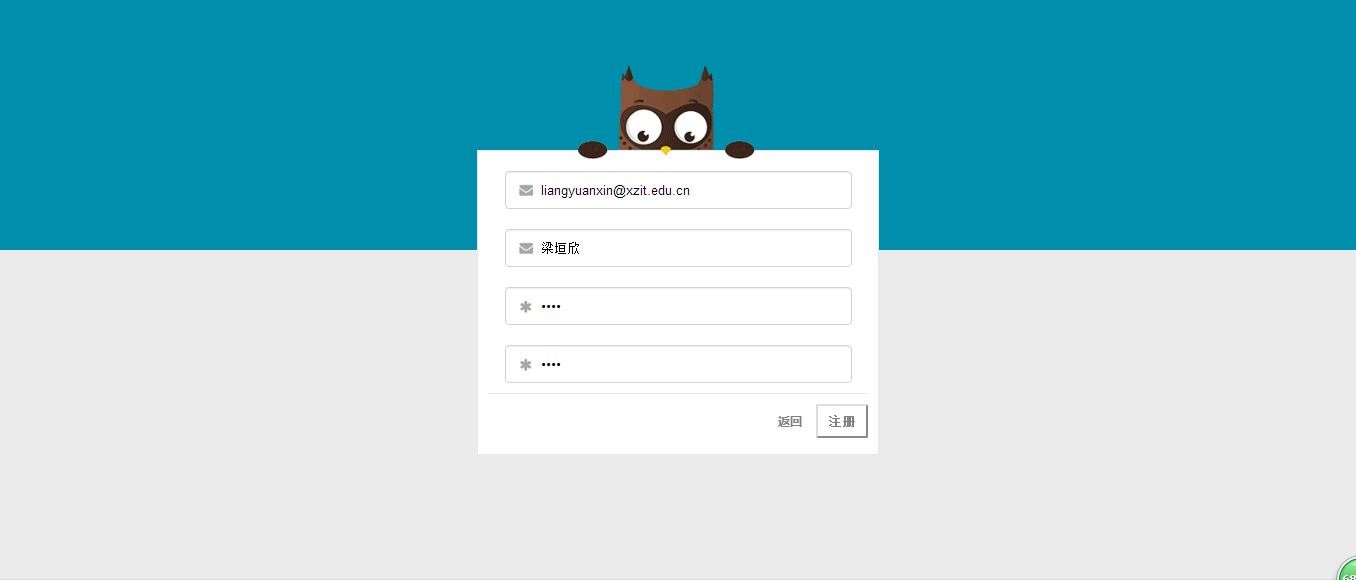


图8-2-2-1 本系统的注册页面

当用户输入了正确的注册信息后，系统会发送用户注册的请求。注册成功后，进入注册成功的页面。用户注册成功的页面见图8-2-2-2。

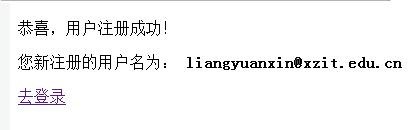


图8-2-2-2 用户注册成功的页面

当点击注册成功页面的去登录链接时，系统会回到登录页面。

### 8.2.3 查看收件箱邮件

当用户在主页面点击收件箱时，系统会发出查看已接收邮件的Ajax请求。当浏览器接收到服务器的响应时，浏览器会解析接收到的JSON类型的数据，动态生成收件箱邮件的列表。生成收件箱邮件列表的主页面见图8-2-3。



图8-2-3 生成收件箱邮件列表的主页面

### 8.2.4 查看邮件内容

当用户点击邮件列表中任意一封邮件的主题时，系统会发送查看邮件类容的Ajax请求。此时主页面会隐藏当期邮件列表，显示邮件编辑器。当浏览器接收到服务器的响应时，浏览器会解析接收到的JSON类型的数据，并在邮件编辑器中显示该邮件的内容。

显示邮件内容的主页面见图8-2-4。

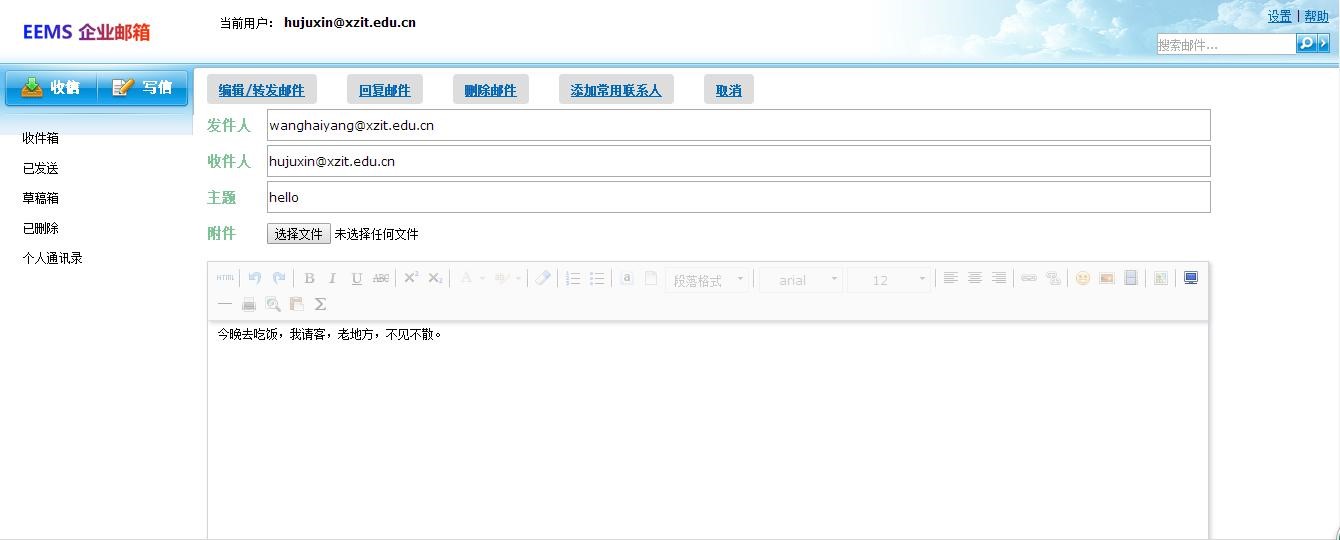


图8-2-4 显示邮件内容的主页面

注意，此时当前页面的所有文本框及编辑器都是不可编辑的。只有用户点击编辑/转发邮件或回复邮件的按钮时，才能编辑邮件。

### 8.2.5 回复邮件

当用户点击回复邮件的按钮时，即可编辑对发送人的回信。

编辑回信的主页面见图见图8-2-5。

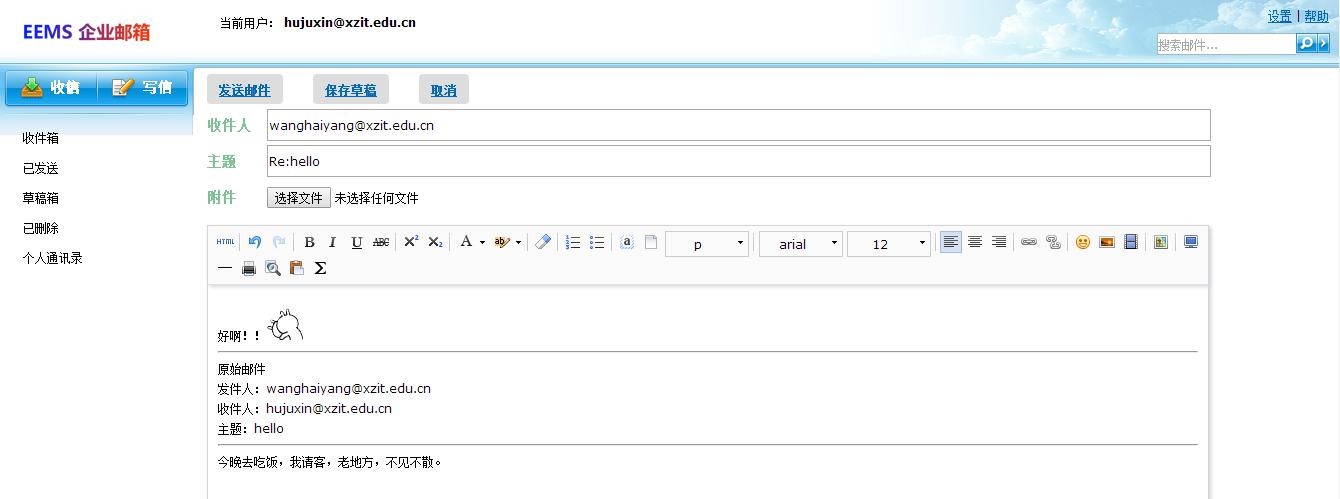


图8-2-5 编辑回信的主页面

### 8.2.6 保存草稿

当用户点击保存草稿的按钮时，可将当前邮件保存为草稿。

保存草稿成功的主页面见图8-2-6-1。



图8-2-6-1 保存草稿成功的主页面

当用户点击确定按钮时，会关闭邮件编辑器。此时点击草稿箱即可找到刚才保存的草稿邮件，点击该邮件主题同样可查询到邮件内容（见图8-2-5）。

生成草稿箱邮件列表的主页面见图8-2-6-2。



图8-2-6-2 生成草稿箱邮件列表的主页面

### 8.2.7 发送邮件

当用户点击编辑邮件的主页面（图8-2-5）的发送邮件按钮时，即可给对方用户发送邮件。发送邮件成功的主页面见图8-2-7-1。

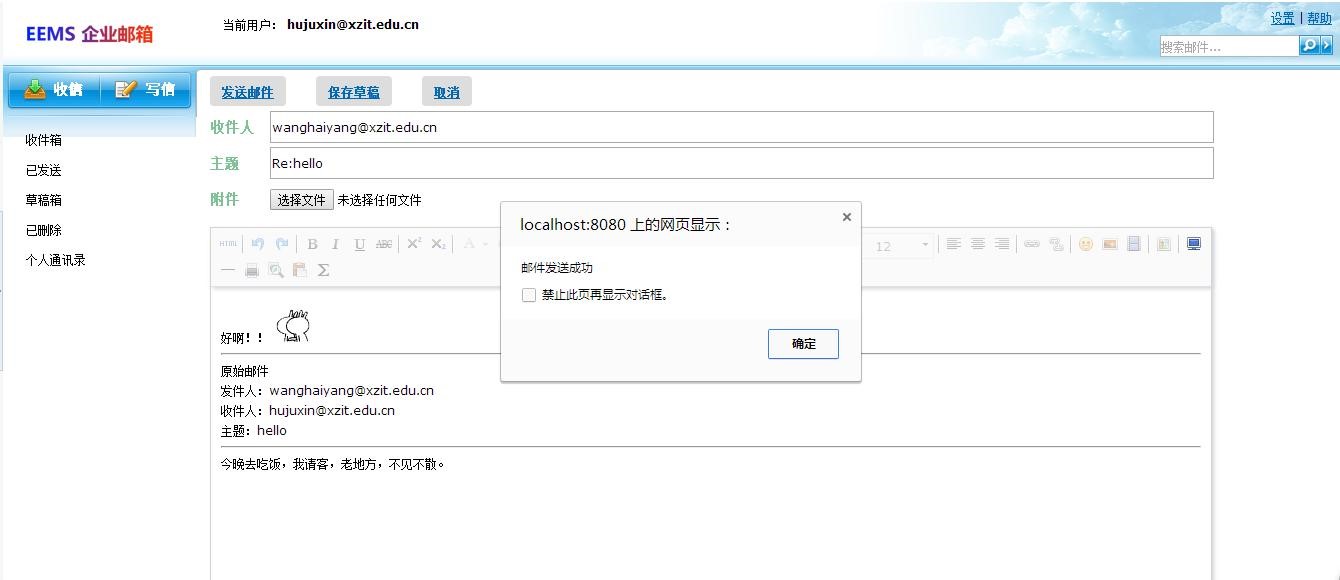


图8-2-7-1 发送邮件成功的主页面

当用户点击确定按钮时，会关闭邮件编辑器。如果发送的邮件原来是草稿邮件，那么该邮件会被移到已发送的区域。同样登录对方账号也可找到刚才发送的邮件。

生成已发送邮件列表的主页面见图8-2-7-2。

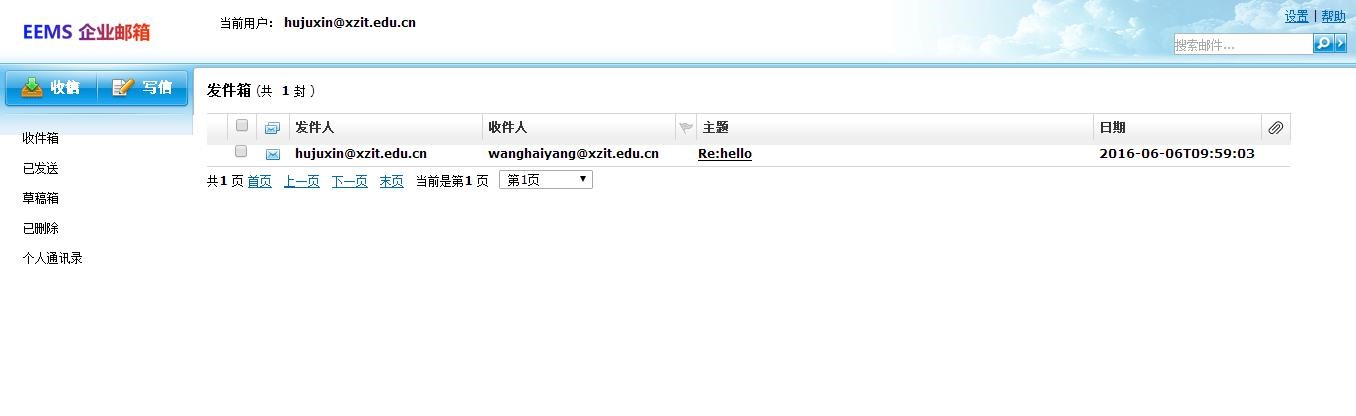


图8-2-7-2 生成已发送邮件列表的主页面对方账号的收件箱邮件列表的主页面见图8-2-7-3。



图8-2-7-3 对方账号的收件箱邮件列表的主页面

### 8.2.8 删除邮件

在查看邮件内容的主页面（图8-2-4）上，用户可将当前邮件删除，删除后的邮件被移到已删除。此时点击左侧列表的已删除即可查看到刚才已删除的邮件。

删除邮件成功的主页面见图8-2-8-1。



图8-2-8-1

生成已删除邮件列表的主页面见图8-2-8-2。此时点击邮件主题仍然可查看邮件内容。

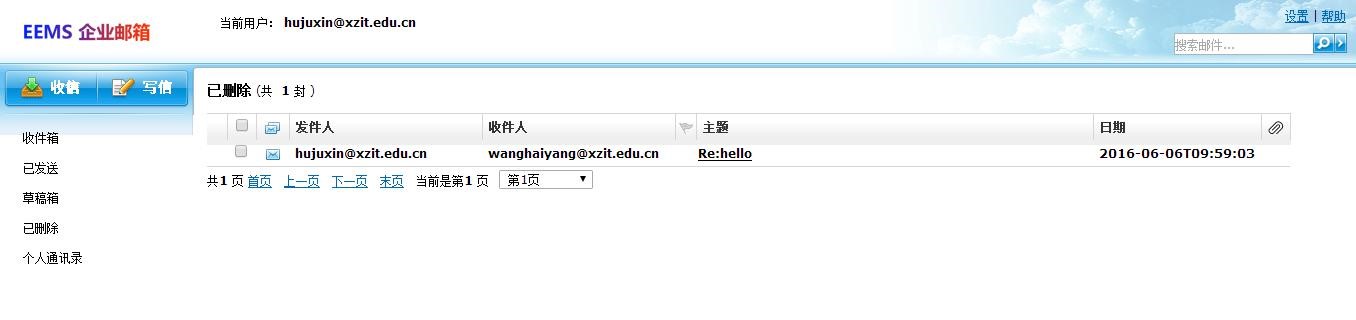


图8-2-8-2

查看已删除邮件内容的主页面见图8-2-8-3。与之前查看邮件内容不同的是，此时用户还可执行还原邮件和永久删除邮件的操作。

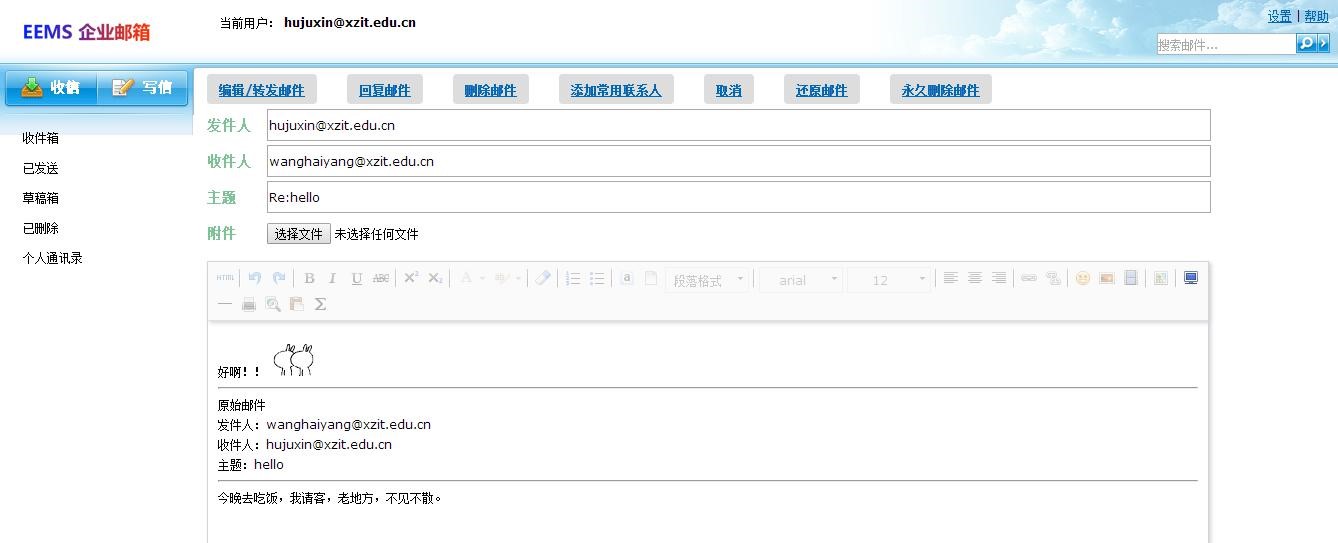


图8-2-8-3

### 8.2.9 添加常用联系人

在查看邮件内容的主页面（图8-2-4）上，用户可将当前邮件的对方用户添加到个人通讯录中。

添加常用联系人的主页面见图8-2-9。



图8-2-9 添加常用联系人的主页面

### 8.2.10 查看个人通讯录

当用户点击个人通讯录时可查看到该用户的常用联系人。

查看个人通讯录的主页面见图8-2-10。



图8-2-10 查看个人通讯录的主页面

此时用户可执行给常用联系人写信和删除联系人等操作。

## 8.3 本章小结

本章主要讲述了本系统的实现内容，还略带介绍视图层的设计。本系统视图层的设计主要采用JSP技术实现。此外，本章重点讲述本系统的核心功能的演示，完成了本系统核心功能的系统实现。

# 结论

经过数据持久层，数据访问层，业务逻辑层，系统控制层以及视图层的设计后，本系统已基本实现了一个普通电子邮箱的所具备功能。其中包括查询邮件，编辑邮件，保存邮件，发送邮件，删除邮件，还原邮件，查看个人通讯录，编辑个人通讯录等核心功能，以及几乎所有管理系统都具备的登录，注册功能。通过本设计的功能实现，本系统已经能够解决企业内部各部门，各单元之间的通信问题，从而保证了企业内部具有良好的交流与协作，为企业的良好运转提供了有力支持。

当然，本系统实现得还不够完善，有部分扩展功能尚未实现。其一，本系统是一个面向企业员工的电子邮件管理系统，因此，本系统还需要提供完善用户员工信息，以及查询单位通讯录的功能。普通用户可不用完善员工信息，但作为企业员工，只有完善了员工信息才能查询单位通讯录。其二，本系统还尚未完全实现上传附件与下载附件的功能。对于一个电子邮箱来说，实现附件的上传和下载也是收发邮件所必须具备的功能之一。但目前由于我个人能力的限制，此功能留到以后有待实现。其三，本系统还不能与互联网上的其他邮箱平台进行通信，即还不能与其他平台上的邮箱账号进行收发邮件。此功能也是一个电子邮箱要求具备的重要功能，因此以后也须实现。

经过本系统复杂的设计，我更深刻地学习了基于UML建模的面向对象分析方法，熟悉了企业级应用的分层结构与开发规范，以及掌握了Struts2+Spring+Hibernate的轻量级 Java EE应用架构以及相关技术。我相信本设计会对我以后进入软件公司，开发更高层次的，更复杂的企业级应用带来深刻的影响。

# 致谢

大学是学校与社会的分水岭，也是校园生涯和人生历程中最为精彩，最为辉煌的舞台。四年的大学生活即将结束，而我的求学生涯也即将画上圆满的句号。然而，这只是一个阶段的结束，同时也是另一个阶段的开始。如何步入社会，如何找到一份满意的工作，如何构建新的交友关系，同事关系，社会关系，以及不久的将来如何找到自己的终生伴侣，构建新的家庭等等，都是我即将要面对的问题。

大学其实可以说是一个微型的社会。当年我们一同参加过全国统一高考，然后凭着一张相同的录取通知书，来自五湖四海的我们聚集在一起。刚踏进大学校门的那一刻，我们所看到的都是陌生的面孔，听到的都是不同地方的口音，然后我们在一系列障碍中逐渐构建起了朋友关系，师生关系，人际关系。这不就是我们步入社会后必须要面对的问题吗？因此，可以说大学生活就是一个通往社会的过渡期。在这个微型的社会圈子里，我们尝试了与不同的人进行交往，参加了多样化的社团活动，了解了很多丰富的故事，学习了很多优秀的知识，掌握了很多处事的经验，同时应对了很多艰难的挑战，经历了许多失败的沮丧与成功的喜悦。不知不觉中，我们学习了很多，收获了很多，改变了很多，或许我们自己还没有察觉，但事实上，我们正一步步地走向社会。四年的大学时光，说长也不长，说短也不短，但或多或少也留下了很多珍贵的回忆。

回想起四年前的初秋，那时我高考失利，无赖之下，不远千里来到了江苏本科大学排名靠尾的徐州工程学院。当时我仍然处在高考失利的阴影下，并没有发现自己有什么特长与亮点，对自己的前途非常的茫然。但学校并没有放弃我，辅导员与班主任也在耐心的开导我，身边同学们也在积极地鼓励我，使我慢慢地走出高考失利的阴影。不知道是否是命运的安排，学校帮我选择了决定着我未来命运与前途的专业--计算机科学与技术。当时对于没有什么突出的特长的我来说，学习一门技术，掌握一项本领可不是一件容易的事。

但是在学长学姐以及老师的帮助下，我渐渐爱上了编程，爱上了软件开发，爱上了计算机技术。从此，我真正起培养其了自己的兴趣，找到了自己最为突出的亮点。四年下来的大学生活有着无数的困惑，失败的沮丧，前景的迷茫，但都无法摧毁我坚定的决心。一个个问题的解决，一次次成功的喜悦，使我渐渐相信了一句话：即使不优秀的学校也会出最优秀的学生。现在，我已经成功入职了一家从事于银行软件系统开发的大型外包公司，从事着软件开发工程师的职责，并接受着公司良好的薪资待遇。我能有现在这样的成绩，离不开身边鼓励我的同学们与老师，离不开悉心开导的辅导员和班主任，更离不开帮助我找到自身亮点与未来命运的学校。为此，我要向他们表达我最深挚的感谢。

除此之外，我还有感谢我的指导老师，王海洋老师。他在百忙之中还抽出时间给我知道指导本系统的设计与论文的修改，帮助我开拓思路，指点迷津，精心指导，最终使我顺利通过毕业设计。为此，我也要向他表示我最深挚的谢意。

# 参考文献

1. 李刚著，疯狂Java讲义（第3版）[M].电子工业出版社，2014
2. 李刚著，疯狂HTML5/CSS3/JavaScript讲义[M].电子工业出版社，2012
3. 李刚著，轻量级JavaEE企业应用实战（第4版）[M].电子工业出版社，2014
4. 李刚著，Struts 2.x 权威指南（第3版）[M].电子工业出版社，2012
5. 李刚著，疯狂Ajax讲义（第3版）[M].电子工业出版社，2013
6. 埃克尔著，陈昊鹏译.Java编程思想[M].机械工业出版社，2007
7. DavidM．Kroenke著，冯飞译.数据库原理[M].清华大学出版社，2008
8. 秦靖，刘存勇著，Oracle从入门到精通[M].机械工业出版社，2011

[9]MichaelBlaha，JamesRumbaug著，UML面向对象建模与设计（第二版）[M].人民邮电出版社，2011

1. 耿祥义，张跃平著.Java设计模式[M].清华大学出版社，2009
2. 软件开发技术联盟著，Java Web开发实战[M].清华大学出版社，2013
3. 贾蓓.镇明敏.杜磊著，Java Web整合开发实战[M].清华大学出版社，2013
4. 李运莉著，web数据库应用系统性能优化[M].北京人民邮电出版社，2011
5. 钱雪忠著，数据库原理与应用（第二版）[M].[北京邮电大学出版社，](https://www.baidu.com/s?wd=%E5%8C%97%E4%BA%AC%E9%82%AE%E7%94%B5%E5%A4%A7%E5%AD%A6%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLuyFBryF-mHmsnjIBnHm10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHmdP1mLnW01)2007
6. 缪忠剑著，基于Spring的集成化Web开发平台的研究与实现[M]．北京机械工业出版社，2013
7. 宋佳颖，基于Java的邮件接收系统分析[J].无线互联科技，2015
8. 盛仲飙，乔道迹，基于Java的通讯软件设计与实现[J].福建电脑，2015
9. 陈衍席，基于Java网络聊天系统的设计及实现研究[J].通讯世界，2016
10. 彭晓青，MVC模式的应用架构系统的研究与实现[J].电子工业出版社，2013
11. 赵争东，基于Java的数据库应用框架的研究设计和探索[J].信息系统工程，2015