[1.绪论 1](#_Toc357060295)

[1.1可行性分析 1](#_Toc357060296)

[1.2怎样是一个有效的在线办公系统： 1](#_Toc357060297)

[2.技术 3](#_Toc357060298)

[3.需求分析 4](#_Toc357060299)

[3.1 系统的实现目标 4](#_Toc357060300)

[3.2系统功能 5](#_Toc357060301)

[3.3 用例模型 6](#_Toc357060302)

[4总体概要设计 8](#_Toc357060303)

[5.系统详细设计与实现 15](#_Toc357060304)

[实现 31](#_Toc357060305)

[总结和展望 31](#_Toc357060306)

**1 绪论**

1.1 背景介绍

1.2可行性分析

对于一些中小型企业而言，日常办公常常重复和效率低，开发一套有效安全的办公系统成为大势所趋。

办公系统的可行性研究包括两部分：

1）经济可行性：科学化、系统化的范畴，历史所趋。

2）技术可行性：网站日后的维护。

1.3怎样是一个有效的在线办公系统：

校友会的在线办公系统是一个实际的项目，怎样做出有效，人人使用的办公系统成了第一个问题。首先，这个系统需要满足校友会办公职员的日常工作，比如上传文件，发布通告等；再有，要怎样满足他们的需求，我阅读了一些文献，上面提到系统应易用性、可用性、有效性，简单的词语确涵盖了很多具体的问题，比如易用性就要考虑，用户可否容易用系统完成某项任务，级联菜单是否设计合理等；最后，系统应该不断测试，用户错误就是系统错误，比如缺乏反馈页面，难懂的文字等。

所以，下文中我尽量写前期的一些研究准备，和提示自己一些设计上的点。不过我要始终提醒自己，测试包括远程的，纸质原型测试，用户体验测试。我开始由于缺乏和用户的沟通，做出了一套“自以为”的办公系统，界面很漂亮，就是增加通知和职工，结果我同学就评价觉得系统没什么用处，客户也认为我完全不懂什么是办公系统。错误中成长啊，没有工作观察和情景访谈，结果就是做无用功。

**1.2.1研究准备**

a.观察目标人群，有助于指定设计决策。事实上，大部分客户都不是产品设计师，他们无法表达需要什么样的产品实现目标，而且通常无法准确的拿捏自己对产品的感觉。比较好的办法是工作观察，可用性测试的目标是找出也用户界面相关的问题。当跟踪某个用户时，目标是找出需要创建什么样的产品，或是如何在现有基础上改进现有产品。

我就提给了自己这些问题，用户是否在某项任务上花费了大量的时间，用户是否被迫记住某项任务的操作步骤，细节等可以用电脑完成的事情。情景访谈更是需要，除了我今年看到的之外你是否还经常要做别的什么任务，如果完成任务还需要一些别的信息，你会怎么做等。用户研究能帮助我们做出更好的预测，但他不能排除所有的不确定性。有些时候，还要做些尝试，然后看看用户是否认为它有用。由此确定需要的功能

b.然后创建了用户的模型，对用户的讨论更加容易，而且深入考虑目标用户，改善设计流程，使其更加以人为中心。比如权限1,2,3,4个负责什么事情，功能更加确定。

制定使用手册，这样迫使自己了解系统的工作原理。没有什么能像这项工作迫使自己思考设计的细节。如果出现难以解释的东西，可能会很难用。

之后便是设计流程图和实例模型。选择最为重要的用户目标，然后考虑这些目标所需的步骤。

设计注意的问题。

c.少用一些文字。北卡罗来纳州立大学心理学院的文章failure to recognize fake internet popup warning messages的论文，分析了人们如金额处理虚警信息。文中说到，“从问卷调查结果数据来看，遇到弹出信息时点击ok按钮的人中，有接近一半的人（42%）说他们这样做只是想让弹出的信息消失。”所以如果发生了一项错误，我有办法让产品不打扰用户的情况下自行修复，就不要让用户知道，如确定用户要访问那个界面，就直接跳到那个页面，自动刷新；又或者出现了用户的操作将出现严重的问题，允许用户撤销。如果无可避免的通知用户，先假定用户不会看你的文字的情况设计界面，避免把按钮设为“确定”和“取消”，而是设定成“删除文件”和“取消”。

根据雅各布尼尔森的给出的文字设计准则。

使用对用户有用的词汇；

每段只阐明一个主旨，并放在句首；

使用能概括段义的标题；

加量关键字；

使用简短的生活化的文字；

语句表达清晰。

d.拿故事版或者纸质原型，设计的草图定期给客户看，并设计具体的任务，让客户自行解决。这是早期的可用性的测试，为了避免开发系统时做无用功，修改草图比改代码方面的多，也助于提高系统的可用性，并制图。

以上是研究准备时需要注意的问题，帮助我更好的设计系统，当然我希望能更好的做出一个系统，但是仅仅要做出一个很好的界面说起来可能要说一本书，所有考虑的点也不太全面， 时反应度可以改进。删除功能考虑的问题，使用率不高可能是设计目的的问题还是功能本身的问题，还是名字没起好。

**2 相关技术**

1.1 背景介绍

2.1开发环境要求

服务器端：因为这是具体的项目，所以要满足我自己电脑的条件和远程链接学校的服务器的条件。

操作系统：windows 7，或者window xp

Web服务器：Tomcat 6.0

Java开发包：JDK1.5以上

数据库：oracle 11g

浏览器：火狐，sougou，IE 6.0以上

分辨率：

客户端：已进行过测试

浏览器：火狐，Google Chore，世界之窗，IE6.0以上

MVC经典设计模式

MVC模式(模型、视图、控制),这个模式能够有效的将业务与视图分开，从而将开发人员与网页美工人员分离开，有利于分工合作，但Model2模式也有它致命的弱点，由于所有的用户请求都通过控制器来与后台业务交互，但每一个用户请求又不同，这样我们就必须为每一个请求写一个控制器(servlet)，如果一旦业务庞大，就可能有很多的控制器，这样会给系统维护和扩展都带来巨大的不便，为了解决这一问题基于MVC模式的开源框架Struts悄然问世，在这个框架中我们所有的请求都到一个中央控制器(ActionServlet)统一处理,这样既有利于我们的维护工作，也增强了系统的扩展性。

Servlet

Servlet是一个JAVA编写的程序，此程序是基于Http协议的，在服务端运行的，是按照servlet规范编写的一个JAVA类。Servlet的主要作用是接受并处理客户端的请求并将其结果发送到客户端的。

运行时，当servlet容器收到一个请求是，servlet容器会针对请求而创建servletRequest和serveltResponse，然后调用service()方法。不管这个请求是post提交还是get提交的，最终这个请求都会是service（）处理。

2.2 Struts 1,2

Struts是Apache软件基金会（ASF）赞助的一个开源项目。它采用了Java Servlet/JSP技术，实现了基于Java EE web应用的MVC设计模式的应用框架，是MVC经典设计模式中的一个经典产品。

在struts中，已经有一个名为ActionServlet的servlet充当控制器角色，根据描述模型、视图、控制器对应关系的struts-config.xml的配置文件，转发视图（View）的请求，组装相应数据模型。两个主要的子系统是系统的内部数据状态对应struts中的ActionForm类，改变数据状态的逻辑动作对应struts中的Action类。这里，struts可以与各种标准的数据访问技术结合在一起，包括EJB，JDBC等。在struts的视图（View）中，除了使用标准的JavaServer Pages(JSP)以外，还提供了大量的标签库使用，同时也可以与其他表现层组件技术（产品）进行整合，比如Velocity Templates，XSLT等。通过应用Strtus的框架，最终用户可以把大部分时间的关注点放在自己的业务逻辑（Action）和映射关系的配置文件（Struts-config.xml）中。

对于编程者而言，需要知晓的问题有

1.Struts1.2要求Action类要扩展一个抽象基类。Struts是面向抽象类编程的。

2.struts1.2的action类依赖servlet的HttpServletRequest和HttpServletRequest为execute方法，提供参数。

3.struts1.2使用ActionForm对象来捕获输入，像Action一样，所有的ActionForm必须拓展基类。其他的javaBean不能作为ActionForm使用，但是开发者可以重新描述已经存在的JavaBean。

4.Struts1.2支持手动验证凭借ActionForm的validate方法，或者通过拓展的工用验证期。

5.Struts1.2为每个模型配一个独立的请求处理器，到是所有在模型中的Action必须共享同一个生命周期。这与struts2很不同。

ＦＣＫ上传包

DAO

DAO(Data Access Object)是一个数据访问接口，数据访问：就是与数据库打交道。夹在业务逻辑与数据库资源中间。

DAO层一般有接口和该接口的实现类。 接口用于规范实现类，实现类一般用于用于操作数据库。一般操作修改，添加，删除数据库操作的步骤很相似，就写了一个公共类DAO类 ，修改，添加，删除数据库操作时 直接调用公共类DAO类。

Ajax

Ajax(即异步javascipt和XML)是一种Web应用程序开发手段，它采用客户端脚本和Web服务器交互数据。所以，不必用中断交互完整页面刷新，就可以动态的更新Web页面。

传统的web应用允许用户填写表单，提交到服务器，然后返回一个新的页面。而Ajax采用异步交互的过程。Ajax在用户和服务器之间引入了一个中间媒介，从而消除了网络交互过程中的处理——等待——处理——等待的缺陷，即相当于在用户和服务器之间加了一个中间层，使用户操作和服务器相应异步化。从而到达响应速度加快的目的。

此外应注意的是客户端界面和Ajax引擎都是在客户端运行的，实现所谓的“富客户端”。

JQuery

Jquery是一个Javascript库。它兼容CSS3和很多的浏览器，Jquery使用户能更方便的处理HTML，document,events等，最大的优点是各种应用的表达非常详细，还有很多成熟的函数插件，也就是说可以用jquery实现一个成熟的功能，更加灵活。

JavaMail API

JavaMail API是一个用于阅读、编写和发送电子邮件的可选包。

JavaMail体系分成三层，分别是抽象层、Internet邮件实现层、协议实现层。

抽象层定义了用于邮件处理功能的抽象类，接口和抽象方法。位于JavaMail顶级包（java.mail）内。

Internet邮件实现层实现了部分抽象层元素，遵循了Internet的标准，如MIME。Internet邮件实现层的接口可实现在java.mail.interent包内。

协议实现层有服务商提供对特定协议的支持。

其中主要的类是Session，Message，Address，Authentcator，Transport，Store和Folder。

SMTP协议

SMTP协议：这是“简单邮件传输协议”，最初互联网上的电子邮件就是通过这个协议在不同的主机上传播的。此协议只支持7bit的字节流（每个字节的最高位将被强制转化成0），这就决定了SMTP协议只能传输简单的ASCII字符集的纯文本英文邮件。

MIME协议

MIME协议：这是“多用途互联网邮件扩展”的简称。由于SMTP协议只能收发纯文本方式的英文邮件极大的影响了邮件的可用性，而MIME协议就是将复杂形式的邮件内容组织成为符合SMTP协议要求的形式，然后通过已有的SMTP协议进行传输。

MIME协议主要解决了两个问题，一是将非ASCII码的字节流转换成SMTP服务器可以接受的7bit的字节流；二是在邮件中组织正文及附件等。

3 **系统需求分析**

首先系统的范围和边界，本办公系统的使用人员为校友会的全体职工，完成他们日常的工作，如活动通知，日程安排，设备租借和上传文件等功能，从而实现办公系统的无纸化，网络化等等，提高企业的工作和行政效率。

3.1 系统的实现目标

一般来说，办公系统实现的目标是：

* 合理的分配权限；
* 严格审查存入的数据；
* 实行增加、删除、修改、按需求显示；
* 运行稳定、安全可靠；
* 界面友好，操作简单；

3.2系统的功能需求

经调查得知（参见概述），本系统要实现的系统功能如下。

本系统的权限主要分为管理员权限1，普通用户权限0和特殊用户权限2。

1）管理员，普通用户，特殊用户权限相同的功能模块

A．通知管理

任何职工都可以添加，和查看通知，并删除，更新自己发布的通知。并且任何用户都可以回复通知。任何用户还可以查看未读通知。

B．邮件管理

职工之间可以通过发送邮件来进一步沟通。可以发送一条信息给其中的一个职工或多个职工，当收信人登录系统时，可以在他的未读邮件里看到未读邮件的数量，然后登录邮箱查看。

C．校友查询

任何职工可以增加校友信息记录，并且按条件查询校友信息（比如）。

D．文件中心

任何用户可以上传文件到服务器的文件中心（文件夹：public），并且下载该文件。此外，用户可以删除自己上传成功的文件。

2）管理员权限功能模块

A．通知管理

除了和其他用户相同的功能之外，管理员有权限更新和删除任何通知。

B．职工管理模块

管理员可以增加，删除用户信息，对用户信息进行更新，并对其进行授权。每个用户会有他自己的角色。

C．客户管理模块

管理员可以增加，删除，和更新客户信息，并且查看客户信息。

D．设备管理模块

管理员可以增加，删除，更新和查看设备信息。并且可以租金设备，登记归还记录，查看设备租借记录，超期记录等。

E．会议日程管理

管理员可以增加，删除，更新会议日程，并且以日历表的形式或者表格的形式查看会议日程记录。此外，还可以查看今日会议日程记录。

F．个人日程管理

管理员可以对自己的个人日程进行增、删、改、查操作。除此之外，用户可以查看当日日程安排。以表格形式查看日程或者以日历表格的形式查看日程。

查看特别用户的个人日程，并且可以更新和删除。

G．文件中心

除了和其他用户相同的功能之外，管理员有权限可以删除任何文件。

4）普通用户权限管理模块

A．个人日程管理

普通用户可以对自己的个人日程进行增、删、改、查操作。除此之外，用户可以查看当日日程安排。以表格形式查看日程或者以日历表格的形式查看日程。这一点和普通用户相同。

但是普通用户可以查看特别用户的个人日程，但是不可以更新和删除。

B．职工管理模块

可以更新职工信息，但是无法自己授权。

C．客户管理模块

普通用户只可以查看

3）特别用户权限功能模块

A．个人日程管理

与普通用户不同的是，其他用户的日程只可以自己查看的，而特别用户的个人日程可以让所有用户查看，但是不可以被除了特别用户和管理员之外的用户修改。

满足上述需求的系统主要包括以下子系统模块

* 通知管理子系统
* 文件中心子系统
* 邮件管理子系统
* 职工管理字系统
* 客户管理字系统
* 个人日程管理子系统
* 会议日程管理子系统
* 设备管理子系统
* 校友查询子系统

3.3 用例模型

3.3.1 识别参与者

用户的使用者为校友会的职工，他们是行政人员，有一定的计算机基础。熟悉Microsoft Office等相关软件。

确定系统用例的第一步是确定系统的参与者，本系统的参与者包括以下几种

系统管理员：负责维护一些信息，如职工和客户信息

普通用户：负责查看一些信息，上传和下载文件和租借设备

刘校长（特殊用户）：每个职工需要知道校长的行程和安排，以便安排活动。



3.3.2构建用例模型

任何用例都必须由某一个参与者出发后才能触发活动，所以当确定系统的参与者后，就可以从系统的参与者开始确定系统的用例。此系统分成三个部分绘制。

1. 管理员用例图

管理员能通过系统做如下的活动

维护职工、客户、文件、设备信息。管理员能够维护职工，客户和设备的信息，添加，删除，更新和查看；设备信息还可以预定设备，取消预定，归还设备及超期提醒。

管理员能够管理会议日程表，个人日程表和刘校长的个人日程表的信息，添加，删除和修改，查看并且查看今日日程。

群件群发和接收。管理员能够发送和接收邮件。



1. 用户信息管理：管理员通过验证后对职工信息的添加、删除、更新和查看操作。管理员通过更新用户信息来为用户分配权限（是管理员，还是特殊用户还是普通用户），每个用户必须有一个角色。用户基本信息包括：用户名、姓名、邮箱、密码、邮箱密码和电话。
2. 通知管理：管理员可以添加、删除、更新和查看任何通知。还可以显示未读的通知，和某条通知的阅读量及未读通知的名单。
3. 通知的回复管理：管理员可以对某条通知添加回复。
4. 客户信息管理：管理员对客户信息的添加、删除、更新和查看操作。客户信息包括：客户名称、邮箱和电话。
5. 设备基本信息管理：管理员可以对设备信息进行添加、删除、更新和查看操作。设备的信息包括设备名称、设备类别、置入时间和用途。
6. 设备租赁管理：管理员通过验证后可以对设备进行租借，取消租借，更新租借内容，归还，查看租借信息和超期提醒。租借的信息包括设备名称、租借人、租出时间、归还时间和状态。
7. 文件管理：管理员通过验证后可以对文件进行上传，下载和删除操作。
8. 日程管理：管理员可以对日程增加，删除和修改。
9. 验证：判断当前用户的权限，是管理员则通过，可以执行操作。

注：上文出现了很多通过验证后的字眼，因为管理员与其他用户是同样一起登陆的，需要验证判断其权限值，做出相应的操作。所以，本例中，判断出是管理员，才可以执行管理员可执行的操作。

1. 普通用户用例图
2. 特殊用户用例图

4 **系统概要设计**

从本章开始，介绍校友会在线办公系统的设计与实现。从总体设计到详细设计，最后实现，本章给出了概要设计。开始本章之前，作者解释了一下的问题。什么是概要设计，需要考虑什么，重点是什么。

概要设计该给出进行详细设计前的提纲和注意点，以及一些公共模块的设计。所以本章先阐释了该办公系统的系统体系结构，模块结构，以方便详细设计的有侧重的和模块化设计。接着阐释了数据库设计，以明确处理的数据目标，然后是公共模块的设计，在详细设计前考虑公共模块（如web.xml，struts.xml，基类，公共类）等，是为了设计多考虑继承，简化代码规模，让开发者更专注于整体和方便排错。在本章偏后，作者思考了整体设计中可能会在详细设计中遇到的问题。

4.1 系统体系结构设计

本系统采用B/S（browser/server，浏览器与服务器）架构，在这种架构下，用户的工作界面通过浏览器来实现，主要事务在服务器上实现，极少事务在本机的前端上实现。从而大大简化了客户端电脑的负荷，方便系统升级和用户的总体成本减低。

4.1.1 MVC模式下的struts1.2

本系统遵循了(Model-View-Controller)MVC设计模式下的struts1.2应用框架。从第一章可知道，struts1.2框架主要由控制器（servlet）,业务逻辑Action和映射关系配置文件(Struts-config.xml)构成。

基本工作流程：

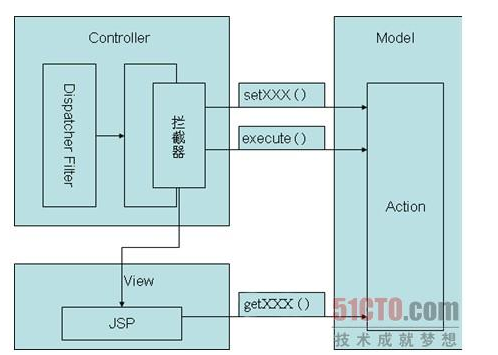
1.一个请求到来，在内存中找到与之匹配的ActionMapping对象，如果没有找到这提示请求路径错误信息，如果找到进入第二步；

2.找到ActionMapping对象对应的formBean,并创建一个实例把请求信息填充到对象中；

3.根据配置的validate属性决定是否进行表单验证，如果需要则执行formbean的validate方法，如果返回null或者不包含ActionMessage的ActionErrors对象，则表明验证通过，如果不通过则根据配置的input属性的值跳到相应的错误页面；

4.通过验证后配置的映射对象并创建该Action对象，执行execute方法完成业务逻辑的操作，返回一个ActionForward对象，ActionServlet根据返回的ActionForward对象，生成动态页面并响应给客户端。

最终程序员可以把大部分时间的关注点放在自己的业务逻辑（Action）和映射关系的配置文件（Struts-config.xml）中。



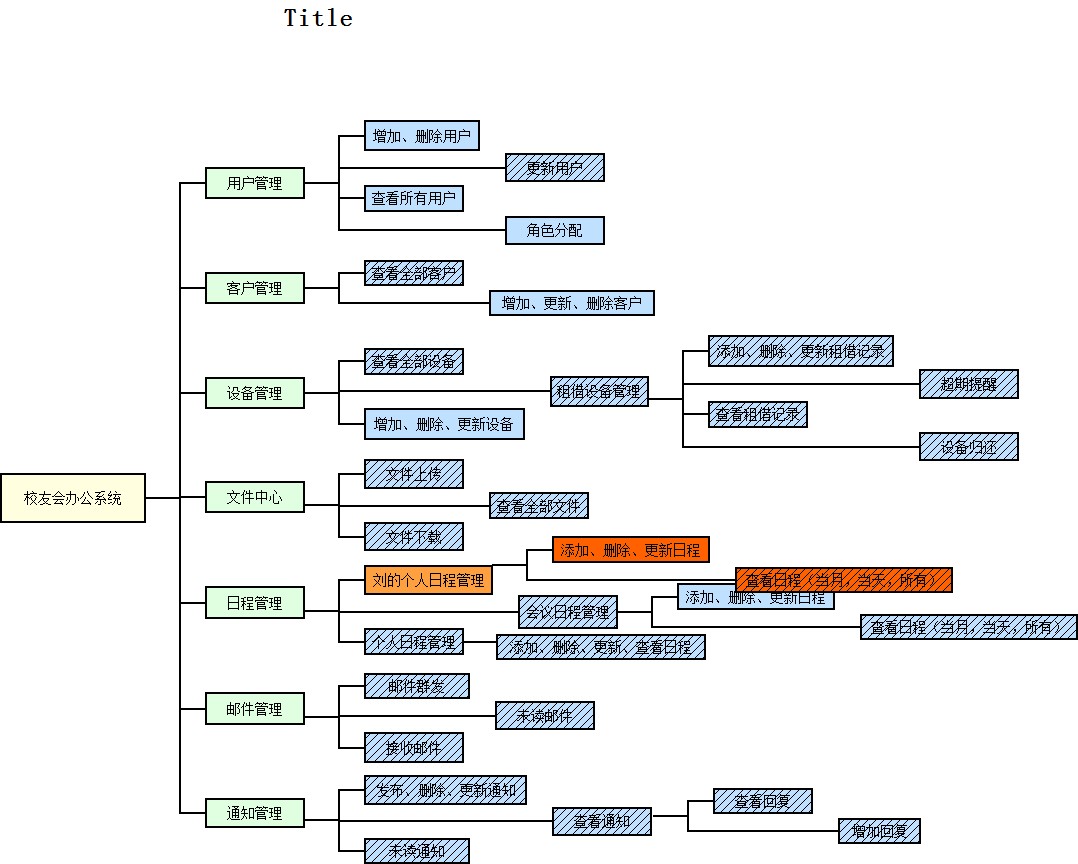
4.1.2 Ajax

另外，Ajax的实现过程是1.客户端触发Ajax事件。2.创建XMLHttpRequest对象的事例，中间指定了相应函数和发送请求。3.向服务器做出请求，可以调用Servlet等服务器端技术。4.在服务器端根据请求做出相应的处理，如访问数据库定。5.请求返回到浏览器。6，XMLHttpRequest对象接受到相应后，按照配置调用callBack()函数处理返回结果，检查readyState等。重点是callBack()在前面操作完成后，最后执行。

4.2系统的模块结构图

根据需求分析，本系统可以分成十个子系统，而且这十个子系统中有些是继承同一个基本系统，具体是通知管理模块、文件中心模块、邮件管理模块、职工管理模块、客户管理模块、个人日程管理模块、会议日程管理模块、设备管理模块、校友查询模块和登录模块。如其中个人日程管理模块和会议日程管理模块的内容几乎完全相同，只是名称不同。

其结构图如下所示。



图

4.3数据库设计

任何系统都需要一个完善，合理的数据库表的支持，本系统也不例外。由于本办公系统，校友会手机APP系统及校友会后台管理系统需要构成一个的整体。所以，数据库表与其他相关同学进行讨论，确定如下：

4.3.1 数据库概念设计

数据库中的属性表

用户信息表：（用户ID，用户名，用户姓名，用户密码，权限，邮箱地址，邮箱密码，联系方式）。

客户信息表：（客户ID，客户名称，邮箱地址，联系方式）。

设备信息表：（设备ID，设备全程，设备类别，用途，购入时间）。

设备租借表：（租借ID，设备ID，用户ID，租借开始时间，租借结束时间，当前状态）。

通知信息表：（通知ID，用户ID，通知标题，通知内容，通知时间）。

通知阅读表：（通知ID，用户ID，已读）。

通知评论表：（用户ID，通知ID，评价标题，评价内容，评价时间）。

会议日程表：（日程ID，日程日期，开始时间，结束时间，完成状态，日程标题，日程内容）。

个人日程表：（日程ID，日程日期，开始时间，结果时间，完成状态，日程标题，日程内容）。

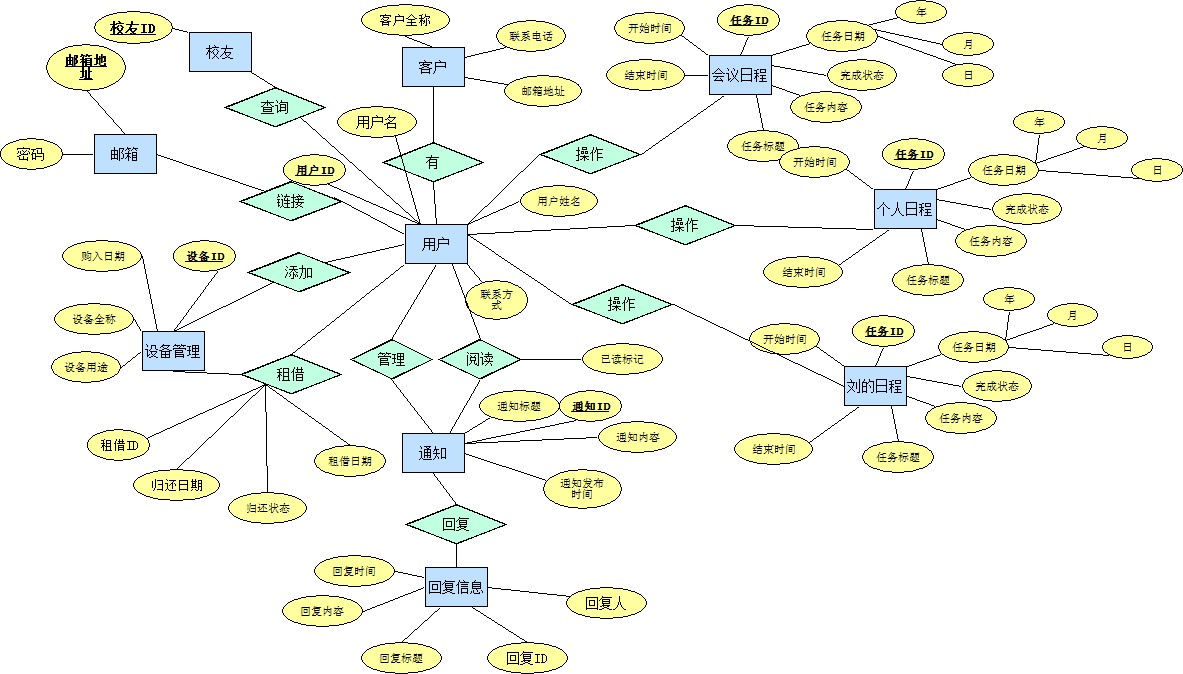
文件信息表：（文件ID，文件题目，用户ID）。

校友信息表：（校友ID，校友用户名，密码，校友名字，性别，生日，ID卡号，入学年份，户籍，大学，班级号，学号，电话号码，qq号，电子邮件，公司，工作，邮政地址，邮编）。

4.3.2 数据库表逻辑设计

数据表关系图

E-R图表示出了数据库中的表之间的关系。关系模型E-R图如果所示。



4.3.3数据表

系统数据库中的表有如下：记录用户的用户信息表，客户信息表等。所有的上述表的描述如下。

用户信息表（users）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| userID | Int | 主键自增 | 用户ID |
| username | Varchar(20) | 唯一 | 用户名 |
| Password | Varchar(100) | 非空 | 登录密码 |
| Status | Int | 非空 | 角色标识 |
| Name | Varchar(100) | 非空 | 真实姓名 |
| email | Varchar(100) | 可以空 | 邮箱地址 |
| Emailpass | Varchar(100) | 可以空 | 邮箱密码 |
| mobileTel | Varchar(100) | 可以空 | 电话 |

注：status中，1为管理员，2为特殊用户，0是普通用户

客户信息表（tb\_customer）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键自增 | 客户ID |
| Name | Varchar(20) | 唯一 | 客户全称 |
| Email | Varchar(100) | 非空 | 邮箱地址 |
| mobileTel | Varchar(100) | 非空 | 联系方式 |

通知信息表（message）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键自增 | 用户ID |
| Subject | Varchar(20) | 唯一 | 通知题目 |
| Day | Varchar(100) | 非空 | 发布日期 |
| Time | Int | 非空 | 发布时间 |
| Content | Varchar(100) | 非空 | 通知内容 |
| Userid | Varchar(2000) | 外键 | 作者Id |

注：userid为users表的主键，本表的外键

阅读信息关系表（visit）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键自增 | ID |
| PID | Int | 外键 | 用户ID |
| MID | Int | 外键 | 通知ID |
| Isread | Int | 非空 | 已读标识 |

注：isread取值为0，1。0是未读，1是已读，默认是0。PID指向users表，MID指向message表。

通知回复表（reply）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键，自增 | 回复ID |
| Mid | Int | 外键 | 通知ID |
| Pid | Int | 外键 | 用户ID |
| Subject | Varchar(100) | 非空 | 回复标题 |
| Time | Varchar(100) | 非空 | 回复时间 |
| Content | Varchar(1000) | 可以空 | 内容 |

注：mid指向message表，pid指向users表

会议日程信息表（task）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键，自增 | 用户ID |
| Year | Varchar(20) | 非空 | 年 |
| Month | Varchar(100) | 非空 | 月 |
| Day | Int | 非空 | 日期 |
| Start\_time | Varchar(100) | 可以空 | 开始时间 |
| End\_time | Varchar(100) | 可以空 | 结束时间 |
| Subject | Varchar(100) | 非空 | 任务主题 |
| Description | Varchar(1000) | 可以空 | 任务内容 |
| location | Varchar(100) | 可以空 | 任务地点 |
| Userid | Int | 外键 | 用户id |

注：Userid指向users表，是本表的外键

个人日程信息表（task1）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键，自增 | 用户ID |
| Year | Varchar(20) | 非空 | 年 |
| Month | Varchar(100) | 非空 | 月 |
| Day | Int | 非空 | 日期 |
| Start\_time | Varchar(100) | 可以空 | 开始时间 |
| End\_time | Varchar(100) | 可以空 | 结束时间 |
| Subject | Varchar(100) | 非空 | 任务主题 |
| Description | Varchar(100) | 可以空 | 任务内容 |
| location | Varchar(100) | 可以空 | 任务地点 |
| Userid | Int | 外键 | 用户id |

注：Userid指向users表，是本表的外键

设备信息表（tb\_equipment）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键，自增 | 用户ID |
| Etype | Varchar(20) | 非空 | 设备类别 |
| Ename | Varchar(100) | 唯一 | 设备全程 |
| Eusage | Int | 可以空 | 设备用途 |
| Buy\_time | Varchar(100) | 可以空 | 购入时间 |

注：设备的Etype是枚举数组里的对象，有“打印机”等。

设备租借关系表（borrow）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| ID | Int | 主键自增 | 租借编号 |
| Eid | Int | 外键 | 设备编号 |
| Name | Varchar(100) | 非空 | 租借人名 |
| Start\_date | Varchar(100) | 非空 | 开始日期 |
| End\_date | Varchar(100) | 非空 | 结束日期 |
| Eusage | Varchar(100) | 可以空 | 用途 |
| Status | Int | 非空 | 归还状态 |

注：Status是设备的归还状态，1为已归还，0是未归还。Eid指向equipment表。

校友信息表（alumni）

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 约束 | 字段描述 |
| userID | Int | 主键 | 用户ID |
| username | Varchar(20) | 唯一 | 用户名 |
| Password | Varchar(100) | 非空 | 登录密码 |
| realName | Varchar(30) | 非空 | 真实姓名 |
| gender | Varchar(20) | 可以空 | 性别 |
| Birthday | Varchar(20) | 可以空 | 出生日期 |
| IDcardnum | Varchar(30) | 可以空 | ID卡号 |
| EnterYear | Varchar(20) | 可以空 | 入学年份 |
| LocationCity | Varchar(30) | 可以空 | 户籍 |
| College | Varchar(200) | 可以空 | 大学名称 |
| ClassNum | Varchar(200) | 可以空 | 班号 |
| StuID | Varchar(20) | 可以空 | 学号 |
| PhoneNum | Varchar(30) | 可以空 | 电话号码 |
| QqNum | Varchar(30) | 可以空 | Qq号码 |
| EmailAddress | Varchar(200) | 可以空 | 电子邮件 |
| Company | Varchar(200) | 可以空 | 公司 |
| Job | Varchar(100) | 可以空 | 职位 |
| MailAddress | Varchar(300) | 可以空 | 邮政地址 |
| PostCode | Varchar(30) | 可以空 | 邮编 |

4.4 公共模块设计

在本系统中，公用模块所设计的类可以在本系统中任意调用，主要有数据库操作类，转换类定义和加密解密类。

其中，数据库操作类包含了数据库的连接，对两个数据表banktable和SelectMoney的查询、修改、删除、添加，以及面向本系统的专有操作。

转换类指编码的转换，由于编码不同，容易造成乱码。

加密解密类指对密码的加密，为了系统的安全性考量。例如，本系统采取了将登陆密码用MD5加密的方法，将密码存入数据库中。

用了struts框架，配置文件需要修改。Web.xml

4.4 窗体风格设计

为了使一个系统风格统一，本文使用了iframe和include来设计大体的窗体布局。

其中Iframe浮框，在浏览器窗口嵌套子窗口，在其中显示页面内容。

<iframe >

本系统使用的网页结构，即分区如下：

头部页面主体功能区尾部

4.5 前段技术分析

去除图形边框 border=0

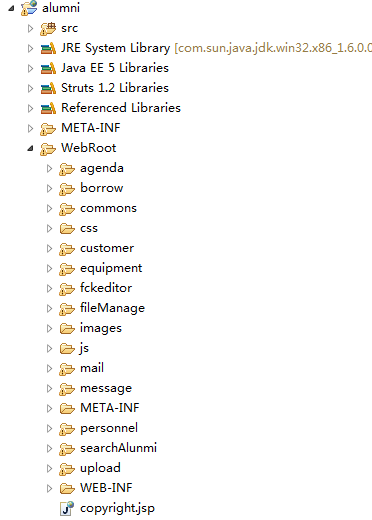
5 **系统的详细设计与实现**

详细设计阶段需要确定怎样实现代码，并分配相应的设计工作，本阶段是设计出程序蓝图，一边接下来翻译成代码。在这里，虽然是设计部分，但是考虑了一些设计代码时的注意点，比如用户密码的加密等。

从需求和概要设计入手，根据软件开发的基本过程，按照模块的顺序，先绘出模块的流程图，并设计出数据库中表间的关系，再给出草绘界面，此界面设计是与客户进行反复沟通后的达成一致的结果。在分析各个模块时，将考虑设计此模块时需要注意的问题。

5.1文件结构图

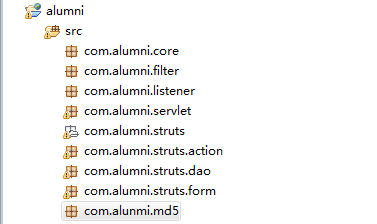
每个文件夹有对应的功能文件。 Src存放了JAVA的封装结构，CSS文件夹中存放了层叠样式表。Images里存放里系统里需要的图，JS里存放了前端的Javascript文件,Webroot下也存放了模块的JSP页面文件，根据名称对应不同的系统模块。 这样分类存放的好处是使文件结构清晰。部署后目录结构如下图：



src封装结构

本系统主要使用了Struts1.2框架，其中有部分模块使用了JSP+servlet的框架。对于使用了Struts1.2的部分，分为action、form、dao。对于JSP+servlet的部分，也有相应的servlet。

项目包的结构如下：



各包的功能如下：

Com.alumni.struts.dao：封装了DAO的实现，处于系统的业务层，实现对象与关系数据库的数据操作。

Com.alumni.struts.action：封装了业务逻辑实现，处于系统的业务逻辑层层，实现了对整个系统的业务逻辑的处理。

Com.alumni.struts.form：封装HTTP请求中的数据，实现了将用户提交的数据自动填充(在相应的action的execute方法被调用之前)到form的属性中。

Com.alumni.sevrlet：封装了JSP+servlet模块下的Servlet的实现，为系统的控制器，实现了对系统业务逻辑的处理。

Com.alumni.core：封装了数据库的链接和对数据库基本操作的实现，使用数据库必须使用这个包。

Com.alumni.md5：封装了MD5加密解密的算法。

Com.alumni.filter：封装了过滤器。

Com.alumni.lister：封装了监听器。实现了监听客户端、服务端的操作，自动激发一些操作等。

5.2 公用模块实现

5.2.1 数据库

5.2.2配置文件

5.3 功能实现

5.3.1登录和登出

5.3.1.1功能描述和流程设计

登录模块，所有用户不管是管理员还是普通用户，用一个界面，输入用户名和密码登录系统，身份认证成功后，获取每个登录用户的职能角色。根据每个用户的角色不同，进入不同的jsp页面。

登出模块，所有用户退出系统，并清除踪迹，显示离线。

5.3.1.2 界面设计

登录部分包含有logo图标，用户名和用户密码的输入框，提交和取消按钮。这里的登陆框参考了网上的登录模板和图片，然后按照本办公系统的整体风格修改。

5.3.1.3具体实现

在本模块中，用JSP+Servlet+JavaBean技术来操作数据库，在index.jsp输入用户名和密码后跳转到<http://localhost:8080/alumni/servlet/logincheck>进行身份验证和职能认证，最后分别跳转到系统不同页面。在其他页面，如果session失效了，将重新跳转到登录页面。

配置文件web.xml：根据web.xml中的servlet路径值，跳转到相应的jsp界面。

|  |
| --- |
| <servlet>  <servlet-name>loginCheck</servlet-name>  <servlet-class>com.alumni.servlet.loginCheck</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>loginCheck</servlet-name>  <url-pattern>/servlet/loginCheck</url-pattern>  </servlet-mapping> |

JSP登录页面，发出请求。

|  |
| --- |
| <form id=*"frmLogin"* name=*"frmLogin"* action=*"servlet/loginCheck"* method=*"post"* onSubmit=*"return isValid(this);"*>  <label for=*"Id"*>登录用户名：</label>  <input type=*"text"* value=*""* name=*"username"* />  <label for=*"password"*>登录用户名：</label>  <input type=*"password"* value=*""* name=*"pwd"* />  <input type=*"submit"* value=*"提交"* name=*"Submit"* /> |

Servlet：按照配置文件找到相应的servlet，处理业务需求。

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  String username = request.getParameter("username").trim();  String pwd = request.getParameter("pwd").trim();  MD5 oMD5=**new** MD5();pwd=oMD5.getMD5ofStr(pwd); //对密码进行MD5加密  String sql = "select \* from users where username = '" + username + "' and password ='" + pwd + "'";  PersonnelForm personnel = **null**;  *userBean*.type=personnel.getStatus();  *userBean*.ID=personnel.getID();  **try** { …….//省略的内容是检查username=username，password=pwd的用户是否在users中，并封装在personnel}conn.close();  } **catch** (SQLException e) {  PrintWriter out = response.getWriter();  out.println("<script>alert('服务器错！'); location.replace('http://hp-pc:8080/alumni/index.jsp') </script>");  e.printStackTrace();  }  **if**(personnelList.isEmpty()){  PrintWriter out = response.getWriter();  out.println("<script>alert('登录错！'); location.replace('http://hp-pc:8080/alumni/index.jsp') </script>");    }**else**{personnel=personnelList.get(0);  **if**(personnel.getStatus()==1){//判断是管理员  PrintWriter out = response.getWriter();  out.println("<script>alert('管理员登陆成功！'); </script>");  *userBean*.type=1;  HttpSession session = request.getSession();  session.setAttribute("id", *userBean*.ID);  session.setAttribute("status", *userBean*.type);  response.sendRedirect("http://hp-pc:8080/alumni/main.jsp");  }  **else**{  PrintWriter out = response.getWriter();  out.println("<script>alert('login successfully'); </script>");  HttpSession session = request.getSession();  session.setAttribute("id", *userBean*.ID);  session.setAttribute("status", *userBean*.type);  response.sendRedirect("http://hp-pc:8080/alumni/main.jsp");  }  }  } |

5.3.2用户基本信息管理

5.3.2.1功能描述与流程设计

此功能为添加、删除、更新用户和查看所有用户

添加：管理员可以增加用户信息，用户名必填，真实姓名，初始密码，权限必填，联系电话，邮箱，邮箱密码可为空。

更新：管理员可修改所有用户的所有信息，普通用户和特殊用户只可以更新自己的用户名，真实姓名，联系电话，邮箱，邮箱密码和密码。

删除：管理员可以删除用户信息。

流程设计

在personnel.jsp中查看员工列表，选择操作（增或删或改），然后操作完后跳回personnel.jsp。增：personnel\_add.jsp填入表单，然后通过action选择add，在personnelAction中对应add的操作，更新数据库，成功即进入personnel\_ok？para=1，在personnel.jsp中输出list。

流程图为：

5.3.2.2 界面设计

从数据库中读取所有用户信息，并可以适时的时候按钮添加，删除和修改。

图用户基本信息管理主页面

5.3.2.3具体实现

Struts-config.xml：根据Action中的不同返回值，跳转到相应的jsp界面。

|  |
| --- |
| <action  attribute=*"personnelForm"*name=*"personnelForm"*path=*"/personnel"*  scope=*"request"*  type=*"com.alumni.struts.action.PersonnelAction"* validate=*"true"*>  <forward name=*"personnelQuery"* path=*"/personnel/personnel.jsp"* />  <forward name=*"personnelAdd"* path=*"/personnel/personnel\_ok.jsp?para=1"* />  <forward name=*"personnelQueryModify"* path=*"/personnel/personnel\_modify.jsp"* />  <forward name=*"personnelModify"* path=*"/personnel/personnel\_ok.jsp?para=2"* />  <forward name=*"personnelDel"* path=*"/personnel/personnel\_ok.jsp?para=3"* />  <forward name=*"error"* path=*"/error.jsp"* />  </action> |

Jsp：界面中为提交表单和网页跳转的按钮指定Action的函数，/personnel.do对应着跳转到action包中的PersonnelAction，点击后执行，下面以添加用户为例。为input name=”name”的值设定职工users的username属性，这样，提交表单时，先执行前端的javascript的checkform(form1)函数，检查时候填入是否合法，如果合法，提交后转到Action包中PersonnelAction.java，执行execute函数，执行action=personnelADD的部分。

|  |
| --- |
| <form name=*"form1"* action=*"*<%=basePath %>*personnel.do?action=personnelAdd"* method=*"post"* onSubmit=*"return checkform(form1)"*>  <tr>  <td>用户名：</td>  <td><input name=*"username"* type=*"text"* id=*"username"* size=*"20"*>  \*</td> </tr>  <tr>……</tr>  <input name=*"Submit"* type=*"submit"* class=*"btn\_grey"* value=*"提交"*> |

Action：提交后，根据\\*.do?action=XX转到com.alumni.struts.action包中的相应文件，并含有参数action=personnelAdd，如本例中是personnel.do?action=personnelAdd，转到action包中的personnelAction.java，由于personnelAction继承了Action.java，所以自动执行execute，并执行有action=personnelAdd的部分代码。

添加用户部分，为了安全性考虑，将用户的密码加密，而且已加密后的形式存到数据库中，使用了MD5加密算法。其原理在详述。

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  String action = request.getParameter("action");  **if**("personnelAdd".equals(action)){  **return** personnelAdd(mapping, form, request,response);}  .......  }  **private** ActionForward personnelAdd(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) {  PersonnelForm personnelForm = (PersonnelForm) form;  personnelForm.setUsername(chStr.*toChinese*(personnelForm.getUsername()));  ……//设置personnelForm并转码，省略personnelForm.setXX(personnelForm.getXX())  **int** ret = personnelDAO.insert(personnelForm);  **if** (ret == 1) {//插入成功  **return** mapping.findForward("personnelAdd");  }  request.setAttribute("error","error");  **return** mapping.findForward("error");  }} |

权限控制：管理员与普通用户有不同的操作权限。本系统中实现的方法是利用权限值，if...else...控制界面的显示实现。

|  |
| --- |
| <%**if**(Utype==1){ %>  <a href=*"personnel.do?action=personnelModifyQ&id=*<%=id%>*"*>修改</a>  <a href=*"personnel.do?action=personnelDel&id=*<%=id%>*"*>删除</a>  <%}**else**{ %>  <font color=*"#f5f5f5"*>修改</font>  <font color=*"#f5f5f5"*>删除</font>  <%} %> |

5.3.2.4实现难点

JS邮箱的合法性

流程图

风格设计

技术与难点

具体代码

5.3.3客户管理模块

5.3.3.1功能描述与流程设计

此功能为添加、删除、更新客户信息，并查看所有客户信息。

查看：所有用户可以查看客户的信息，以表格的形式呈现的。

添加：管理员可以添加客户的信息，客户全称，邮箱地址和联系电话是必填的。而且客户全称是唯一的，邮箱地址要合法。

编辑：管理员可以更新客户的全称，邮箱地址和联系电话，要注意全称的唯一性，邮箱的合法性。

删除：管理员可以删除客户信息，将其从数据库和页面中删除。

5.3.3.2界面设计

5.3.3.3具体实现

Struts-config.xml：根据Action中的不同的返回值，跳转到JSP界面

|  |
| --- |
| <action  attribute=*"customerForm"* name=*"customerForm"* path=*"/customer"* scope=*"request"*  type=*"com.alumni.struts.action.CustomerAction"* validate=*"true"*>  <forward name=*"customerQuery"* path=*"/customer/customer.jsp"* />  <forward name=*"customerAdd"*path=*"/customer/customer\_ok.jsp?para=1"* />  <forwardname=*"customerQueryModify"* path=*"/customer/customer\_modify.jsp"*/>  <forward name=*"customerModify"*path=*"/customer/customer\_ok.jsp?para=2"*/>  <forward name=*"customerDel"* path=*"/customer/customer\_ok.jsp?para=3"* />  <forward name=*"error"* path=*"/error.jsp"* />  </action> |

JSP：页面中点击网页跳转或提交的按钮，转入com.alunmi.struts.action中的相应Java文件，并执行与action值相关的函数，注意，前面的action是指src中的包名，后面的action是相应客户端要执行的动作传来的参数。下面以更新客户信息为例，涉及两个主要的jsp文件，一个是请求更新页面，这个页面中，客户的ID通过url参数传到action包中的java文件，即CustomerAction.java接收到action=customerModifyQ和客户的ID值。

|  |
| --- |
| <tr>  <td style=*"padding:5px;"*><%=name%></td>  <td align=*"center"*>  <a href=*"customer.do?action=customerModifyQ&id=*<%=id%>*"*>修改</a></td>  </tr> |

另一个是提交更新的页面，这个页面中，已经得到了待更新的客户信息的原始记录，然后通过提交表单，执行checkform，验证填入数据的合法性，提交到action包中customerAction.java，action参数值是customerModify。

|  |
| --- |
| <form name=*"form1"* action=*"*<%=basePath %>*customer.do?action=customerModify"* method=*"post"* onSubmit=*"return checkform(form1)"*>  <%**if**(list.size()==1){  CustomerForm c=(CustomerForm)list.get(0);  %>  <tr>  <td width=*"20%"* height=*"27"* align=*"center"*>客户全称：</td>  <td width=*"80%"*><input name=*"name"* type=*"text"* id=*"name"* value=*"*<%=c.getName()%>*"* size=*"50"*>\*</td>  </tr>  <tr>...</tr>  <input name=*"Submit"* type=*"submit"* class=*"btn\_grey"* value=*"提交"*>  <%**}**%>  </form> |

Action：传值成功后，执行Action包中CustomerAction.java中的execute函数，根据action值的判断，执行函数体内不同的函数，为了完成更新操作，需要两次访问CustomerAction.java，传入两次action值，action=customerModifyQ是获取数据库在待更新记录的原数据，而action=customerModify是更新记录。以customerModifyQ为例，在execute中对action值判断后，转入customerQueryModify函数，该函数的作用是根据ID获取数据库中该条记录，customerDAO.query(id)是获取对应的客户信息，然后把客户信息存到放request的属性中，供转发后的页面获取request属性值。最后将它转发到更新界面。

而customerModify中，执行customerModify函数，函数参数form中存放了jsp中提交的客户信息，包括客户ID，该函数的作用是将form中的数据更新到数据库中，dao.update。

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  String action = request.getParameter("action");  **if**("customerModifyQ".equals(action)){  **return** customerQueryModify(mapping, form, request,response);  }**else** **if**("customerModify".equals(action)){  **return** customerModify(mapping, form, request,response);  }**else**  request.setAttribute("error", "error");  **return** mapping.findForward("error");  }  **private** ActionForward customerQueryModify(ActionMapping mapping,ActionForm form, HttpServletRequest request, HttpServletResponse response) {  Int id=Integer.*parseInt*(request.getP arameter("id")); request.setAttribute("customerQuery",customerDAO.query(id));  **return** mapping.findForward("customerQueryModify");  }  **private** ActionForward customerModify(ActionMapping mapping,ActionForm form, HttpServletRequest request, HttpServletResponse response) {  CustomerForm customerForm=(CustomerForm) form;  //更新customerForm对象  customerForm.setName(chStr.*toChinese*(customerForm.getName()));  customerForm.setEmail(chStr.*toChinese*(customerForm.getEmail()));  customerForm.setMobileTel(chStr.*toChinese*(customerForm.getMobileTel()));  **int** ret=customerDAO.update(customerForm);  **if**(ret==0){  request.setAttribute("error","error");  **return** mapping.findForward("error");  }**else**{  **return** mapping.findForward("customerModify");  }  } |

5.3.3.4难点：

Customer\_ok。

5.3.4设备管理模块

此模块分两个子块功能，一个是设备信息管理，此功能为添加、删除、更新设备信息，并查看所有设备信息；另一个是设备租借管理，租借已有设备，更新租借，取消租借，归还设备，查看租借记录（包括已归还和未归还的）和超期提醒。

事实上，要实现设备租借管理，需要有设备才行，要先实现设备信息管理。

5.3.4.1设备信息管理

5.3.4.1.1功能描述与流程设计

查看所有设备：所有用户可以查看所有设备信息，以表格的形式呈现。设备全程，设备类别、设备用途、购入时间。

添加：管理员可以添加设备的信息，设备全称、设备类别是必填的，用途和购入时间可以空。而且设备全称是唯一的。

更新：管理员可以更新设备全称，设备类别、用途和购入时间，要注意全称的唯一性。

删除：管理员可以删除设备信息，将其从数据库和页面中删除。

5.3.4.1.2界面设计

5.3.4.1.3具体实现

对于设备信息管理模块

Struts-config.xml：根据Action中的不同的返回值，跳转到JSP界面

|  |
| --- |
| <action  attribute=*"equipmentForm"* name=*"equipmentForm"* path=*"/equipment"*  scope=*"request"*type=*"com.alumni.struts.action.EquipmentAction"*validate=*"true"*>  <forward name=*"equipmentQuery"* path=*"/equipment/equipment.jsp"* />  <forward name=*"equipmentAdd"* path=*"/equipment/equipment\_ok.jsp?para=1"* />  <forward name=*"equipmentQueryModify"* path=*"/equipment/equipment\_modify.jsp"* />  <forward name=*"equipmentModify"* path=*"/equipment/equipment\_ok.jsp?para=2"* />  <forward name=*"equipmentDel"* path=*"/equipment/equipment\_ok.jsp?para=3"* />  <forward name=*"error"* path=*"/error.jsp"* />  </action> |

JSP：页面中点击网页跳转或提交的按钮，转入com.alunmi.struts.action中的相应Java文件，并执行与action值相关的增删改函数。下面以删除设备信息为例，这个页面中，设备的ID通过url参数传到action包中的java文件，即CustomerAction.java接收到action=equipmentDel和设备的ID值。

|  |
| --- |
| <td style=*"padding:5px;"*><%=ename%></td>  <td align=*"center"*>  <a href=*"*<%=basePath %>*equipment.do?action=equipmentDel&id=*<%=id%>*"*>删除</a></td> |

Action：传值成功后，执行Action包中EquipmentAction.java中的execute函数，根据action值的判断，执行函数体内不同的函数，为了完成更新操作，以action=equipmentDel为例，在execute中对action值判断后，转入equipmentDel()函数，该函数的作用是从数据库中删除设备id是id值的设备信息。

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  @SuppressWarnings("unused")  EquipmentForm equipmentForm = (EquipmentForm) form;  String action=request.getParameter("action");  **if**("equipmentDel".equals(action)){  **return** equipmentDel(mapping,form,request,response);  }  request.setAttribute("error", "error");  **return** mapping.findForward("error");  }  **private** ActionForward equipmentDel(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  EquipmentForm equipmentForm = **new** EquipmentForm();  **int** id=Integer.*parseInt*(request.getParameter("id"));  equipmentForm.setId(id);  **int** ret=equipmentDAO.delete(equipmentForm);  **if**(ret==1)  **return** mapping.findForward("equipmentDel");  } |

5.3.4.2设备租借管理

5.3.4.2.1功能描述与流程设计

查看租借记录和超期设备提醒：租借记录包含已归还的和未归还的。所有用户可以查看租借的信息，以表格的形式呈现。可以查看到设备租借的信息，包括租借设备，租借人，借出日期，应还日期，用途和当前状态。

租借设备：所有的用户可以租借设备，设备全程，设备类型，租借人，租赁时间，归还时间，状态为必填，用途可以空，设备全程和设备类型可以在左侧的选择菜单中选择并单击，也可以手动填入。

更新租借：所有的用户可以更新租借，包括设备全称、设备类别和购入时间，要注意全称的唯一性。

取消租借：所有用户可以删除租借，将其从数据库和页面中删除。

5.3.4.2.2界面设计

5.3.4.2.3实现思想

和设备信息管理模块相似，租借设备就等同于设备信息管理中的增加设备；选择归还，其实就是更新数据库，把设备状态由未还更新成已还，和设备管理中的更新设备相似，不同的是更新只一个信息数据；查看租借记录和超期提醒与设备信息管理中的查看原理相似，不同的就是设备信息管理是全部查看，而设备信息管理是有条件的查看。下面，作者具体解释如何实现超期提醒功能，特殊的原理清楚了，那全部查看就更明白了，所以做了粗略解释。

5.3.4.2.4具体实现

对于设备信息管理模块

Struts-config.xml：根据Action中的不同的返回值，跳转到JSP界面

|  |
| --- |
| <action  attribute=*"borrowForm"* name=*"borrowForm"* path=*"/borrow"* scope=*"request"* type=*"com.alumni.struts.action.BorrowAction"* validate=*"true"*>  <forward name=*"borrowQuery"* path=*"/borrow/borrow.jsp"* />  //跳转到查看全部租借记录，其中包括未归还租借和已归还租借  <forward name=*"borrowAdd"* path=*"/borrow/borrow\_ok.jsp?para=1"* />  //设备添加成功，将跳转到borrow\_ok.jsp?para=1，弹出添加成功提醒，转到查看全部记录。  <forward name=*"borrowQueryModify"* path=*"/borrow/borrow\_modify.jsp"* />  //更新某个租借信息，得到租借的原始记录成功后，跳转到更新界面borrow\_modify.jsp  <forward name=*"borrowModify"* path=*"/borrow/borrow\_ok.jsp?para=2"* />  <forward name=*"borrowDel"* path=*"/borrow/borrow\_ok.jsp?para=3"* />  <forward name=*"borrowReturn"* path=*"/borrow/borrow\_ok.jsp?para=4"* />  //设备归还，更新数据库中的记录后，跳转到borrow\_ok.jsp?para=4，弹出已归还。  <forward name=*"borrowRestore"* path=*"/borrow/borrow\_ok.jsp?para=5"* />  //设备归还撤销，更新数据库中的记录后，跳转到borrow\_ok.jsp?para=5，弹出取消归还  <forward name=*"overdue"* path=*"/borrow/overdue.jsp"* />  //超期提醒，超期记录查完后存入并保持后，跳转到overdue.jsp  <forward name=*"error"* path=*"/error.jsp"* />  </action> |

JSP：页面中点击网页跳转或提交的按钮，转入com.alunmi.struts.action中的相应Java文件，并执行与action值相关的增删查函数。下面以超期提醒信息为例，首先，用户提出查看超期提醒请求，转入Action包中action=overdue中处理。JSP文件中是：

|  |
| --- |
| <a href=*"http://hp-pc:8080/alumni/borrow.do?action=overdue"* target=*"main"* ><font size=*"2"*>&middot;超期提醒</font></a> |

Action：传值成功后，执行Action包中BorrowAction.java中的execute函数，根据action值的判断，执行函数体内不同的函数，为了完成查询操作，与action=overdue的函数overdue()执行的功能是将已归还和未归还的租借记录的数组链表存入到request中。

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  String action = request.getParameter("action");  **if**("overdue".equals(action)){  **return** overdue(mapping, form, request,response);  }  request.setAttribute("error", "error");  **return** mapping.findForward("error");  }  **private** ActionForward overdue(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  request.setAttribute("borrowQuery", borrowDAO.query(0));  //query(0)查询所有租借记录，返回数组链表类型，setAttribute将此链表保存为名为borrowQuery，供request.getAttribute使用。  request.setAttribute("borrowReQuery", borrowDAO.queryRe());  //queryRe()查询所有已归还记录  request.setAttribute("borrowUreQuery", borrowDAO.queryUre());  //queryUre()查询所有未归还记录  **return** mapping.findForward("overdue");  } |

DAO：实现与数据库操作相关的实现，提供相应的接口。本例中详细介绍一下List queryRe()函数。该函数的功能是查询所有未归还的记录。

|  |
| --- |
| **public** List<BorrowForm> queryUre() {  List<BorrowForm> borrowList = **new** ArrayList<BorrowForm>();  BorrowForm cF = **null**;  String sql= "SELECT \* FROM borrow WHERE status=0";  ResultSet rs = conn.executeQuery(sql);  **try** {  **while** (rs.next()) {  cF = **new** BorrowForm();  ……cF.setXX(rs.getYY(n));  borrowList.add(cF);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  conn.close();  **return** borrowList;  } |

jsp文件，完成Action中的操作后，将按照struts-config.xml返回到overdue.jsp。 该jsp页面从request取得数据，并显示。Jsp中，控制的数据显示方式。

|  |
| --- |
| List listUre=(List)request.getAttribute("borrowUreQuery");  //通过迭代方式显示数据  Iterator itUre=listUre.iterator();  **while**(itUre.hasNext()){  BorrowForm c=(BorrowForm)itUre.next();  **if**(c.getEnd\_date().compareToIgnoreCase(thisDate)<0) {%>  %>  <tr>  <td align=*"center"*><%=c.getEname()%></td>  <td>......</ td>  <td align=*"center"*>超期</td>  <%}%>  </tr> |

难点：

Request.setAttribute

条件显示。

5.3.5日程管理

5.3.5.1功能描述与流程设计

日程管理包括个人日程管理，会议日程管理和刘的日程管理，这三个子模块只有操作权限上的差异，数据库内部设计不同外，完全是相同的，原理自然相同。所以，作者详细介绍会议日程管理，以会议日程管理为例，推及到其他的日程管理模块。

会议日程管理是以日程表的形式，添加，更新，删除和查看任务，本模块还要实现的功能有表格的形式显示当日会议日程任务，当月会议日程任务。

添加任务：管理员可以通过点击日历上某一天的数字来为这一天添加任务，任务日期为选择的这一天日期，可以修改。任务表里的任务标题，任务状态必填，任务描述，任务开始时间，结束时间，地点可以空。

更新租借：管理员可以通过点击日历上的任务结点更新任务，而任务表里的任务标题，任务日期，任务状态必填、可修改，任务描述，任务开始时间，结束时间，地点可以空、可修改。

取消租借：管理员可以通过点击日历上的任务结点删除任务，从而从数据库删除。

查看日历表：所有用户可以查看日历上的所有日程，并点击查看任务详细说明。

查看当月，当天日程：所有用户可以以表格的形式查看当月，当天的任务。

5.3.5.2界面设计

5.3.5.3实现思想

界面而言，首先要生成动态日历，日历的结点创建等用Jquery编辑。然后在每个日期结点的数字框上加载事件：点击创建日程，提交表单给服务器处理，并返回任务标题和日期，这些动作完成后，执行回调函数：在这一日期的日期框里创建一个任务提醒框结点，点击就可以编辑和删除，编辑和删除过程同添加。

页面设计

5.3.5.4具体实现

按照Ajax的具体实现过程来解释日程表的实现，以添加任务为例。

JSP界面——制作动态日历

考虑一个月的最大情况，先制作7\*6的表格，标记第一个格子的id= calBox0，横向以此类推。详见代码。这样是一是为了实现动态的日历，使用算法将日期与星期对应；而是为了实现在这一日期的格子中添加这一天的任务结点，使用算法找到格子。 一个单元格的结构如下

|  |
| --- |
| <tr>  <td class=*"calBox "* id=*"calBox28"*>  <div class='date' id='2009-3-13’ onclick='openAddBox(this)'>  <div id='task3' class='task' onclick='editTask(this)'>subject</div>  </td>  </tr> |

算法：根据当前年月填充单元格，计算填充开始位置

|  |
| --- |
| **function** fillBox() {  **var** dayCounter = 1; //设置天数计数器并初始化为1  **var** cal = **new** Date(year, month, 1); //以当前年月第一天为参数创建日期对象  **var** startDay = cal.getDay(); //计算填充开始位置  //计算填充结束位置  **var** endDay = startDay + getDays(cal.getMonth(), cal.getFullYear()) - 1;  //如果显示的是今日所在月份的日程，设置day变量为今日日期  **var** day = -1;  **if** (today.year == year && today.month == month) {  day = today.day;  }  //从startDay开始到endDay结束，在每日单元格内填入日期信息  **for** (**var** i=startDay; i<=endDay; i++) {  **if** (dayCounter==day) {  $("#calBox" + i).html("<div class='date today' id='" + year + "-" + (month + 1) + "-" + dayCounter + "' onclick='openAddBox(this)'>" + dayCounter + "</div>");  } **else** {  $("#calBox" + i).html("<div class='date' id='" + year + "-" + (month + 1) + "-" + dayCounter + "' onclick='openAddBox(this)'>" + dayCounter + "</div>");  }  dayCounter++;  }} |

向服务器提出请求：通过调用JQuery.post(url,[date],[callback,[type

]])，远程HTTP POST请求载入信息，将数据date传给地址为url的服务器来处理。以添加任务为例，用户触发了添加事件，请求向服务器提交新任务信息。

|  |
| --- |
| **function** addTask() {  **var** subject = $("#subject").val(); //获取任务信息  ……  $.post("calendar.jsp", //服务器页面地址  {  action: "add", //action参数  subject:subject,  ……  },  **function**(taskId) { //回调函数  buildTask(wholeDate, taskId, subject); //建立任务节点  closeAddBox(); //关闭新建任务box  }  );  }  } |

在服务器器上做出相应的处理：获取客户端传来的数据，对数据库进行操作，以本例为例。将数据插入到数据库中，得到这条记录的taskID，out.print（taskID）为向客户端输出流。

|  |
| --- |
| String action = request.getParameter("action"); //获取action信息  **if** ("add".equals(action)) {  Task task=**new** Task();  task.setSubject(request.getParameter("subject"));……  out.print(meetingDao.addTask(task)); |

如果需要向客户端输出数据很多，而且每次传过来都要客户端解析转换，这个问题用JSON数据很方便，从 web 服务器上读取 JSON 数据（作为文件或作为 HttpRequest），将 JSON 数据转换为 JavaScript 对象，然后在网页中使用该数据。

|  |
| --- |
| **if** ("getTasks".equals(action)) { //获取整月任务信息  JSONArray array= **new** JSONArray();  String month = request.getParameter("month");  String year = request.getParameter("year");  month=String.valueOf(Integer.parseInt(month)+1);  List<Task> list=(List<Task>)meetingDao.getTasks(year,month);  Iterator<Task> it=list.iterator();  Task task=**null**;  **while**(it.hasNext()){  JSONObject obj = **new** JSONObject();  task=(Task)it.next(); //转入添加到JSON  obj.put("id",task.getId());  obj.put("subject",task.getSubject());  obj.put("wholeDate",task.getYear()+"-"+task.getMonth()+"-"+task.getDay());  obj.put("start\_time",task.getStart\_time());  array.add(obj); //添加到JSONArray  }  out.print(array.toString()); |

JSON：使用JSON转换数据时，在项目中需要加入json-lib-2.3-jdk15.jar、ezmorph-1.0.6.jar、commons-lang-2.5.jar、commons-collection-3.2.1.jar和commonsbeanutil-1.8.3jar，少了一个都会出错。

回调函数处理返回结果：回调函数是Ajax的精髓，因为它要等到前面的任务执行完才执行回调函数，因此实现了异步传输，更新页面内容。以添加任务为例，客户端解析服务端的输出流，然后为这个任务创建结点。

|  |
| --- |
| **function**(taskId) { //回调函数  $("#" + wholeDate).parent().append("<div id='task" + taskId + "' class='task' onclick='editTask(this)'>" + subject + "</div>");//建立任务节点  closeAddBox(); //关闭新建任务box  } |

注：$("#" + wholeDate).parent().append,是jQuery中返回id= wholeDate的父标记的id元素的结尾加入内容。

对应的，客户端处理JSONArray的数据。JSON直接让数据变成了JavaScript 对象，可以在网页中直接使用该数据。

|  |
| --- |
| **function**(json) { //回调函数  //遍历JSON数组，建立任务信息  $(json).each(**function**(i){  $("#" + json[i]).parent().append("<div id='task" + json[i].id + "' class='task' onclick='editTask(this)'>" + subject + "</div>");  });  } |

5.3.5.5难点：

JSON数据的数据

Month的变化

Out.clear();

5.3.6邮件管理模块

5.3.6.1功能描述和流程设计

所有职工可以发送一条信息给其中的一个职工或多个职工，当收信人登录系统时，可以在他的未读邮件里看到未读邮件的数量，然后登录邮箱查看明细。

5.3.6.2界面设计

5.3.6.3具体实现：

在本例中，还是使用struts1.2的框架，主要使用JavaMail来发送邮件。用户在JSP页面，填写发送邮件的信息封装成sendMailForm并传到SendmailAction.java中进行处理。在Javamail中，首先获取form中的用户提交的信息；然后设置发送邮件的基本信息，最后使用SimpleMailSender类的sendTextMail(””)来发送邮件。

SendMailDAO:：主要用来设置发送邮件的基本信息，发送邮件。主要代码如下：

|  |
| --- |
| **public** **int** sendMail(SendMailForm s) {  **int** ret = 0;  String from = s.getAddresser();String to = s.getAddressee();  String subject = s.getTitle();String content = s.getContent();  String password = s.getPwd();String path = s.getAdjunct();  **try** {  String mailserver="smtp."+to.substring(to.indexOf('@')+1,to.length());  Properties prop = **new** Properties();  prop.put("mail.smtp.host", mailserver);  prop.put("mail.smtp.auth", "true");//添加验证  MyAuthenticator authen = **new** MyAuthenticator(from,password);  //需要身份验证，创建一个密码验证器  Session sess = Session.*getDefaultInstance*(prop,authen);  //根据邮件回话属性和密码验证器构造一个发送邮件的session  MimeMessage message = **new** MimeMessage(sess);  message.setFrom(**new** InternetAddress(from));  String toArr[]=to.split(",");  InternetAddress[] to\_mail=**new** InternetAddress[toArr.length];  **for**(**int** i=0;i<toArr.length;i++){  to\_mail[i]=**new** InternetAddress(toArr[i]);  }  message.setRecipients(Message.RecipientType.*TO*,to\_mail);  message.setSubject(subject);  Multipart mul = **new** MimeMultipart();  BodyPart mdp = **new** MimeBodyPart();  mdp.setContent(content, "text/html;charset=utf-8");  mul.addBodyPart(mdp);  **if**(!path.equals("") && path!=**null**){  ……附件添加 }  message.setContent(mul);  //设置邮件内容  Transport transport = sess.getTransport("smtp");  transport.connect(mailserver, from, password); //发送邮件  transport.sendMessage(message, message.getAllRecipients());  transport.close(); ret = 1;  } **catch** (Exception e) {  ret = 0;  } **return** ret;  } |

注：myAuthenticator是一个继承了Authenticator的类，它带便了一盒可以为网络连接获取认证信息的对象，通常通过提示用户输入用户名和密码来收集认证信息，是连接可以访问受保护资源。

Struts-config.xml：根据Action中的不同的返回值，跳转到JSP界面。

|  |
| --- |
| <action  attribute=*"sendMailForm"*name=*"sendMailForm"*path=*"/sendMail"*scope=*"request"*  type=*"com.alumni.struts.action.SendMailAction"* validate=*"true"*>  <forward name=*"addMail"* path=*"/mail/sendMail.jsp"* />  <forward name=*"sendMail"* path=*"/mail/sendMail\_ok.jsp"* />  <forward name=*"error"* path=*"/error.jsp"* />  </action> |

JSP：在JSP页面中填写发送邮件的信息，并提交到Action包中的java文件处理。

|  |
| --- |
| <form action=*"*<%=basePath %>*sendMail.do?action=sendMail"* method=*"post"* name=*"form1"* onSubmit=*"return checkform(form1)"*>  <tr><td>收件人：</td>  <td>  <textarea name=*"addressee"* cols=*"58"* rows=*"3"* id=*"addressee"* title=*"收件人"*></textarea></td></tr>  <tr>……</tr>  <td><input name=*"Submit"* type=*"submit"* value=*"发送"*>  </form> |

Action：主要用来处理用户提交的信息，设置邮件

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping,ActionForm form,HttpServletRequest request,HttpServletResponse response){  String action = request.getParameter("action");  **if**("sendMail".equals(action)){  **return** sendMail(mapping, form, request,response);  }……  }  **private** ActionForward sendMail(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response){  SendMailForm sendMailForm=(SendMailForm) form;  sendMailForm.setTitle(chStr.*toChinese*(sendMailForm.getTitle()));  sendMailForm.setAdjunct(chStr.*toChinese*(sendMailForm.getAdjunct()));  sendMailForm.setContent(chStr.*toChinese*(sendMailForm.getContent()));  **int** ret=sendMailDAO.sendMail(sendMailForm);  **return** mapping.findForward("sendMail");  } } |

5.3.6.4难点一、邮件中有附件

我们平时发送的邮件主要可以分解成2大部分，一个是发信人,接信人,主题等邮件标头,另外一部分是邮件内容，它包括了邮件的附件。我们在发送普通邮件的时候content设置的类型是 "text/html "，带上附件后，我们需要把content的类型设置成Multipart，这时content包括了附件和 "text/html "类型的正文。有附件的内容的的代码如下。

|  |
| --- |
| message.setSubject(subject);//设置邮件主题  ……//设置发信人，接信人，主题等标头  //设置有附件的邮件内容内容  Multipart mul = **new** MimeMultipart();// MiniMultipart类是一个容器类，包含MimeBodyPart类型的对象  BodyPart mdp = **new** MimeBodyPart();  mdp.setContent(content, "text/html;charset=utf-8");//设置文本内容  mul.addBodyPart(mdp); //创建一个BodyPart  **if**(!path.equals("") && path!=**null**){ //附件处理  mdp = **new** MimeBodyPart();  String adjunctname = **new** String(path.getBytes("utf-8"), "ISO-8859-1");  path = (System.*getProperty*("java.io.tmpdir") + "/" +path).replace("\\", "/");//对邮件的转码  FileDataSource fds = **new** FileDataSource(path);  DataHandler handler = **new** DataHandler(fds);  mdp.setFileName(adjunctname);  mdp.setDataHandler(handler);  mul.addBodyPart(mdp);  }  message.setContent(mul); //将MiniMultipart设置为邮件内容 |

5.3.6.5难点二、导入联系人列表

在struts1.2中，导入联系人列表，可以将联系人的信息存在List中，通过再Action业务逻辑层设置request的参数，名称为”personnnelList”、”customerList”等。

JSP：发出发邮件请求，其实就是发送导人联系人列表的请求。

|  |
| --- |
| <a href=*"http://hp-pc:8080/alumni/sendMail.do?action=addMail"* target=*"main"* ><font size=*"2"*>&middot;邮件群发</font></a> |

Action：业务逻辑层的功能是将联系人的信息从数据库中取出，存放在List中，再把list的内容放在request的参数中，转发到目标页面。

|  |
| --- |
| **public** ActionForward execute(ActionMapping mapping,ActionForm form,HttpServletRequest request,HttpServletResponse response){  String action = request.getParameter("action");  **if** ("addMail".equals(action)) {  **return** addMail(mapping, form, request,response);  }.....  }  **private** ActionForward addMail(ActionMapping mapping, ActionForm form,  HttpServletRequest request, HttpServletResponse response) {  request.setAttribute("personnelQuery",personnelDAO.query(0));  request.setAttribute("customerQuery",customerDAO.query(0));  **return** mapping.findForward("addMail");  } |

JSP：显示联系人表。并实现当点击联系人姓名时，邮箱地址出现在收件人框中。

|  |
| --- |
| <%  List<PersonnelForm> list\_p=(List<PersonnelForm>)request.getAttribute("personnelQuery");  List<CustomerForm> list\_c=(List<CustomerForm>)request.getAttribute("customerQuery");  **if**(list\_p.size()>0){%>  <table>  <tr>  <td height=*"27"*><a onclick="javascript:ShowTR(img\_2,OpenRep2)" href=*"#"* ><img src=*"Images/jia.gif"* alt=*"展开"* name=*"img\_2"* width=*"10"* height=*"10"* border=*"0"*> 员工列表</a></td>  </tr>  <tr id=*"OpenRep2"* style=*"display:none"*>  <td height=*"27"*>  <table width=*"100%"* border=*"0"* cellspacing=*"0"* cellpadding=*"0"*>  <% **for**(**int** i=0;i<list\_p.size();i++){  PersonnelForm p=(PersonnelForm)list\_p.get(i);%>  <tr>  <td>&nbsp;<a href=*"#"* onclick=*"add('*<%=p.getEmail()%>*')"*><%=p.getName()%></a></td>  </tr>  <%}%>  </table> |

JS：显示联系人表。并实现当点击联系人姓名时，邮箱地址出现在收件人框中。

|  |
| --- |
| **function** add(email){  **if**(checkemail(email)){//检查邮箱格式是否合法，非法或空返回false  s=form1.addressee.value;  **if**(checkemail(s)){  arrS=s.split(",");  flag=**false**; //标记是否已经添加  **for**(i=0;i<arrS.length;i++){  **if**(arrS[i]==email){ //判断该Email地址是否已经添加  flag=**true**;  **break**;  }  }  **if**(!flag){  form1.addressee.value=s+","+email;  }}**else**{  form1.addressee.value=email;  }}} |

5.3.7文件上传下载模块

5.3.7.1功能描述和流程设计

在这里使用commons-fileupload组件实现文件上传功能，用户可以在fileUp.jsp页面里选择要上传的文件，然后提交到Servlet(FIleUpload)中，并将上传的文件写入服务器的硬盘中和数据库中。

5.3.7.2界面设计

5.3.7.3具体实现

JSP：fileUp.jsp进入数据库中，弹出文件上传成功字样。

|  |
| --- |
| <FORM ACTION=*"*<%=basePath %>*servlet/fileUpload"* METHOD=*POST* ENCTYPE=*"multipart/form-data"*>  <td>文件：</td>  <td><input name=*"file1"* type=*"file"* id=*"subno"* size=*"32"* ></td>  <INPUT type=*"submit"* value=*"上传"* id=*submit* name=*submit*> |

Servlet：先在init()中设定文件上传的路径地址，本来中dir=””，再用MultipartRequest实例化设定上传参数，然后传回所有文件输入类型的名称，用方法得上上传的文件的真正的文件名，fileName是文件输入类型的名称。上传成功了，就在返回文件属性，存入数据库中，弹出文件上传成功字样。

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  **int** id=(Integer)session.getAttribute("id");  **try** { MultipartRequest multi =  **new** MultipartRequest(request, dirName, 10\*1024\*1024,  "utf-8", **new** DefaultFileRenamePolicy());  Enumeration files = multi.getFileNames();  **while** (files.hasMoreElements()) {  String name = (String)files.nextElement();  String filename = multi.getFilesystemName(name);  String originalFilename = multi.getOriginalFileName(name);  String type = multi.getContentType(name);  File f = multi.getFile(name);  **if** (f != **null**) {  fileForm cF=**new** fileForm();  cF.setFname(filename);  cF.setPid(id);  DataSource ds = **null**;  FileDAO fileDao=**new** FileDAO();  **try**{  InitialContext ctx=**new** InitialContext();  ds=(DataSource)ctx.lookup(Constants.*DB\_TOMCAT\_CONTEXT*);  **int** n = fileDao.insert(cF);  **if**(n > 0){  out.println("<script language='javascript'>");  out.println("alert('上传成功');"); |

5.3.7.4难点

路径乱码

校友查询模块

统计在线人数

MD5加密算法

5.3.8通知管理模块

5.3.8.1功能描述与流程设计

此功能为添加、删除、更新用户和查看通知，回复通知，查看未读通知，查看通知的阅读量和未读某条通知职工的列表。

添加：职工可以增加通知，通知标题必填，日期，内容可以为空，时间可以是默认的系统时间。通知内容包括：通知标题，通知内容，通知发布日期和时间

更新：管理员和添加这条通知的职工可修改通知，所以管理员可修改所有通知。

删除：管理员和添加这条通知的职工可以删除通知，所以管理员可以删除所有通知。

未读通知查看：任何职工若查看通知的详细内容，未读列表该记录消失。通知的创建职工默认已读此条通知。

查看通知的阅读量和未读该通知的职工列表：管理员在查看全部通知的页面可以查看通知的阅读量，点击这个数字，跳转到未读该通知的职工信息列表。

回复通知：任何职工可以对某条通知回复，回复的标题必填，内容 可以为空。回复内容包括回复标题，回复内容，和回复时间，其中回复时间为当前时间。

流程设计

5.3.8.2界面设计：

5.3.8.3实现思路

关于 通知的增加、删除和更细，在前文中做了详细的解释。具体阐解未读通知提醒、阅读量和通知回复。

未读通知提醒message\_unread.jsp的实现思路

创建一张访问联系数据表(visit)，表内存放用户ID，通知ID，已读标记（1是已读，0是未读）。

首先初始化访问联系数据表，一是每创建一个新用户的同时，创建1\*n条访问联系数据记录(n是通知的总条数)，并且每条记录的isread值为0，即（ID，用户ID(userID)，通知ID(1~n), isread(0)）。二是每创建一个通知的同时，创建1\*m条访问联系数据表记录(m是用户的总个数)，并且每条记录的isread值为0，即（ID，用户ID(1~m)，通知ID(messageID), isread(0)）。

然后，用户进入look.jsp，即用户玷辱查看通知的详细信息，isread被更新为1。

5.3.8.4具体实现：

初始化访问联系数据表。每创建一个用户后和创建以一个通知后，都要初始化访问联系数据表。以每创建一个新用户后初始化为例：首先添加成功用户后，在PersonnelAction.java中，将用户添加成功后，初始化访问联系数据表。

|  |
| --- |
| **int** ret = personnelDAO.insert(personnelForm);  **if** (ret == 1) {//ret==1是插入数据库成功，添加用户成功  VisitDAO visitDao=**new** VisitDAO();  Message message=**null**;  List list=**new** MessageDAO().listMessage();//获得所有通知  **for**(**int** i=0;i<list.size();i++){  Visit visit=**new** Visit();  message=(Message) list.get(i);  visit.setMid(message.getId());  visit.setUid(personnelDAO.findItId(personnelForm));  visit.setIsRead(0);  visitDao.addMessage(visit);//将访问记录的初始化插入数据库  }  **return** mapping.findForward("personnelAdd");  } |

DAO：未读通知的列表查询：

|  |
| --- |
| **public** List<Visit> unVisitMessageList(**int** uid){  Connection conn=**new** ConnDB().getConnection();  String sql="select \* from visit where userid=? and isread=?";  Object[] obj={uid,0};  List<Visit> list = **null**;  **try** {  list=(List<Visit>)runner.~~query~~(conn, sql, obj, **new** BeanListHandler<Visit>(Visit.**class**));  } **catch** (SQLException e) {  e.printStackTrace();  }  **return** list;  } |

5.3.8.5难点一、文本编辑器：FckEditor

FckEditor是众多优秀HTML在线文本编辑器之一，以其支持多语言的优势而受到众多开发者的亲睐。

要实现FckEditor的编辑文本，插入图片，需要在Web-INF目录中的lib目录中有commons-fileUpload-1.2.1.jar，commons-io1.3.2.jarFckEditor/

java/core2.4.jar和slf4j-simple152.jar。而且需要下载FckEditor\_2.66.zai解压后将根目录中的FckEditor文件拷贝到项目的WebRoot下。并且添加标签定义。

有三种方法调用FckEditor，分别是FckEditor自定义标签，script脚本语言，FCKeditor API调用。在通知管理模块通知添加出的通知内容使用了FckEditor自定义标签。

在message\_add.jsp

|  |
| --- |
| <%@ taglib uri=*"http://fckeditor.net/tags-fckeditor"* prefix=*"FCK"* %>  <jsp:useBean id=*"message"* class=*"com.alumni.struts.form.Message"* scope=*"request"*/>  <script src=*"*${ctx}*/fckeditor/fckeditor.js"*></script>  <script type=*"text/javascript"* src=*"fckeditor/fckeditor.js"*></script>  <tr>  <td >内容：</td>  <td><script type=*"text/javascript"*>  **var** oFCKeditor = **new** FCKeditor('content');  oFCKeditor.BasePath = "${ctx}/fckeditor/";  oFCKeditor.Height = 300;  oFCKeditor.Value = '${message.content==null?" ":message.content}';  oFCKeditor.Create();  </script></td></tr> |

5.3.8.6难点二、乱码问题的总结：

在JSP开发中，经常会遇到乱码，为解决乱码，需要知道问题的根源所在。而乱码的原因有两个方面：jsp和java本身编译是产生的乱码；其他媒介交互产生的乱码问题。

在这一部分，出现了java程序与其他存储媒介交互时产生的乱码问题，数据库、文件、六等存储方式都是基于字节流的，java程序与这些媒介交互时产生了字符与字节间的转换，如果转换过程中的编码方式和字节原有的不一样，就可能出现乱码。

JSP与页面参数间的乱码

JSP获取页面参数时使用系统的默认编码方式，如果页面编码方式与系统不同，可能就会出现乱码。

解决方式就是在页面获取参数之前，强制指定request获取参数的编码方式。



但是如果每个页面都写这句很麻烦，所以可以利用servlet中的过滤器指定编码。

首先在web.xml中配置过滤器

|  |
| --- |
| <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>com.alumni.filter.SetCharacterEncodingFilter  </filter-class>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

Filter包下的代码是：具体的设计了request编码格式

|  |
| --- |
| **public** **void** doFilter(ServletRequest request, ServletResponse response,  FilterChain chain) **throws** IOException, ServletException {  // Conditionally select and set the character encoding to be used  **if** (ignore || (request.getCharacterEncoding() == **null**)) {  request.setCharacterEncoding("utf-8");  response.setCharacterEncoding("utf-8");  }  // Pass control on to the next filter  chain.doFilter(request, response);  } |

1. JSP与数据库之间的乱码

如果数据库的编码与jsp的编码不一致，那得到的数据很可能是乱码。由于大部分数据库支持Unicode编码方式，所以可以直接使用Unicode与数据库互动，可以在驱动的url参数中指定。

1. Java与文件流之间的乱码

Java读写文件最常用的的类是FileInputStream/FileOutputStream和FileReader/FileWriter，前面的两个是基于字节流的额，读写二进制文件，而后两个是基于字符的。

但是这两个类的构造函数使用默认系统的编码方式，所以可能造成乱码。

解决方法是要么改变系统的编码方式，要么使用FileReader/FileWriter的父类InputStreamReader/OutputStreamWriter，他们的父类有可以制定编码方式的构造函数。InputStreamReader(InputStream in,Charset cs)

1. get的乱码

使用get方式提交中文，出现乱码的原因是Tomcat内部编码格式ISO8859-1导致的。Tomcat会以get的默认编码方式ISO8859-1对汉字进行编码，然后追加到url，导致接受页面得到的参数时乱码。

解决方法是在JSP页面对接受的数据进行编码转换。

String str=new String(request.getParameter(”something”).

getBytes(“ISO-8859-1”),”utf-8”);

或者修改tomcatd的配置文件，又或者用post提交。

6 测试

6.1 实现和测试

实现的算法

难点

常见问题和重点

6 结论与展望

6.1 论文结论

做的事情，

优点

展望

缺点和提高

云计算，它的概念。保障了安全性，文件可以随时随地得到

用户注册可以用Ajax

可以多些公共类，例如权限就可以独立编写公共类使得更加结构化处理

Ajax，速度快

今天,也许只有"现用现学"是解决这种情况的最好办法.

可是面对这些东西,开始觉得那句:"现用现学"是多么恐怖的一种境界.这种东西现学了能马上会用,用不是问题,问题是用这些新东西还能算出工程进度,还能编出可以维护的代码.如果再能保证代码的健壮性,那........

这里不由想到另一个问题:为什么现用现学还能用呢?如果是全新的东西,快速的学习肯定神仙也做不了.所以看起来全新的知识应该有相应的共通点.

是不是因为面向对象的思想呢?个人猜测,可能性比较大,因为一旦变成对象了,就可以不管他们的实现方式,一律归到已有的面向对象的思路中来,无非是多了几个新元素要记而已.要是这样,真应该感谢面向对象.