目 录

第一章 绪论

1.1 系统的目的和意义

系统的功能

作业提交系统基于 B/S 架构，服务器端采用 Tomcat8.0+Mysql，浏览器通过 Web Server 同数据库进行数据交互。这样就大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量，降低了用户的总体成本(TCO)。作业提交系统主要便于老师或同学收集文件，能对文件的上传的格式与大小进行限制，并且能够统一命名方式。整个系统需要根据不同的用户类型提供不同的功能，主要分为管理员和普通用户，管理员能够进行所有操作，普通用户有一定的操作限制。在系统的首页提供统计当前在线人数、文件数、不同用户数量的功能。登录时使用 JS 提供验证码验证功能。

目的和意义

Internet 的蓬勃发展，使学生作业资料的传播方式发生了巨大的变化，传统的传播如纸质形式的作业资料等已经不再是作业资料传播的主要方式，人们更多的开始关注电子化的作业。由于互联网所容纳的信息量大、内容丰富、信息及时、准确，更有相关信息的全面的介绍与比较，大大的方便了人们的阅读，因此在短短几年的时间里，互联网便济身与众多媒体之间，并有相当一部分媒体人群。

这个作业提交系统旨在帮助收集和备份文件，减轻老师与学习委员的负担，提高实验报告的质量，可以作为班级作业收集系统的一个雏形。

第二章 作业提交系统需求分析

2.1 引言

需求分析主要详细说明系统要实现的主要内容和功能有那些，首先对整体做一个需求分析概述,然后分模块进行详细需求分析。需求分析主要通过文字描述，对功能进行一个详细的分析。本系统主要实现以下几项功能：用户登录，用户管理，文件上传下载与管理。本系统的用户注册部分由管理员执行，即管理员增加用户信息之后自动生成用户的登录信息，不允许用户自行注册

2.2 总体需求分析

1. 系统提供登录的功能模块，登录时需要提供验证码识别
2. 提供识别是否为登录用户的安全监测 Filter，用于过滤绕过登录模块直接访问的非正常访问。
3. 根据不同的用户权限提供不同的功能,分为管理员和普通用户。
4. 管理员可以实现对用户的管理，比如添加，删除，修改和模糊查询，普通用户只能进行查询操作
5. 管理员可以实现文件的上传，下载，删除以及模糊查询操作，普通用户只能进行文件的上传和查询操作
6. 限制文件上传格式和大小，并可以在配置文件中修改限制。
7. 可以根据输入的描述信息完成对文件的重命名，统一命名方式
8. 通过 HttpSession 对象实现统计当前在线人数。可以统计当前所有的文件数，管理员数，普通用户人数

2.3 登录部分需求分析

①　登录部分主要为用户提供登录功能,本系统的用户注册部分由管理员执行,即管理员增加用户信息之后自动生成用户的登录信息,不允许用户自行注册。

②　登录时需要提供验证码验证功能。

③　提供识别是否为登录用户的安全监测 Filter，用于过滤绕过登录模块直接访问的非正常访问。

④　登录后端的代码需要根据不同的用户类型跳转到不同的前端界面,本系统主要分为管理员和普通用户

2.4 用户管理需求分析

用户管理部分主要分为管理员与普通用户

管理员可以查看当前所有的用户和管理员，对用户和管理员的信息进行修改，删除和增加管理员以及普通用户

普通用户只能查看当前所有的用户和管理员

2.5 文件管理需求分析

系统需要实现对上传文件的格式与大小进行限制，而且为了便于管理，需要对上传的文件统一命名格式

文件管理部分主要分为管理员与普通用户

管理员可以查看系统中所有的文件，可以进行上传和下载操作，还可以删除文件。管理员可以新建文件夹，对文件夹进行打包下载。

普通用户只能查看系统中所有的文件，可以进行上传操作

2.6 其他功能需求分析

(1) 通过 HttpSession 对象实现用户会话管理，实现统计当前在线用户人数；

(2) 整个系统采用 JSP Model2 实现基于 MVC 设计模式的软件架构。通过 JSP+Servlet+JavaBean 的模式实现系统基础架构。

(3) 整个系统采用 UTF-8 编码，数据库也采用 UTF-8 格式存储，避免乱码问题

(4) 客户端界面要求简洁美观。既要体现系统功能，也需要符合一般网站的美工与布局要求

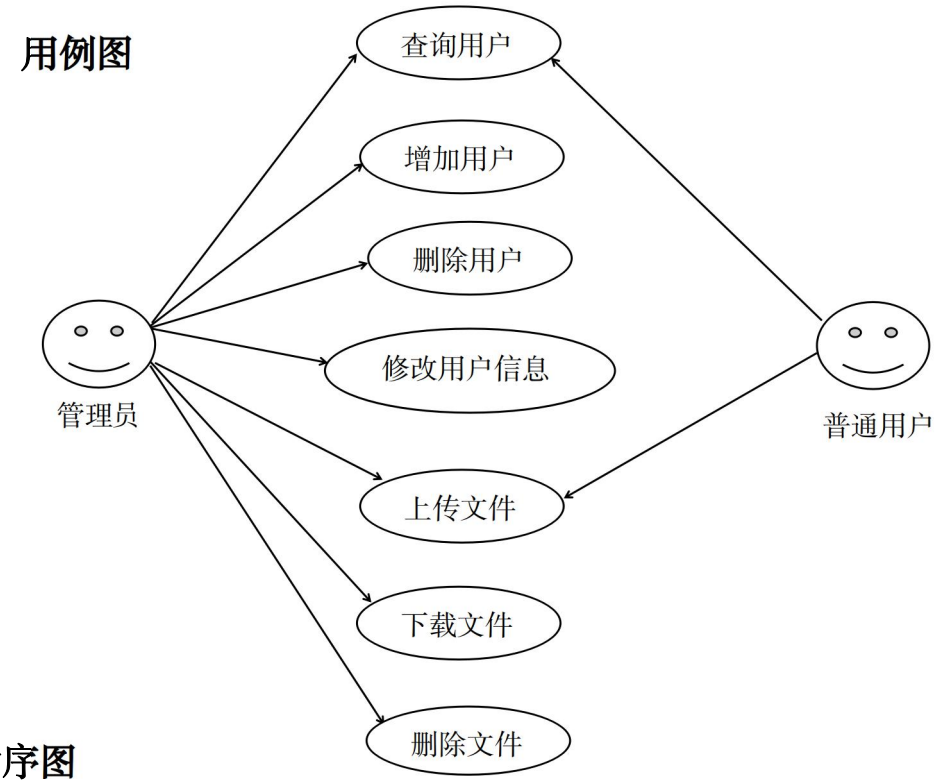
第三章 作业提交系统结构分析与设计

3.1 引言

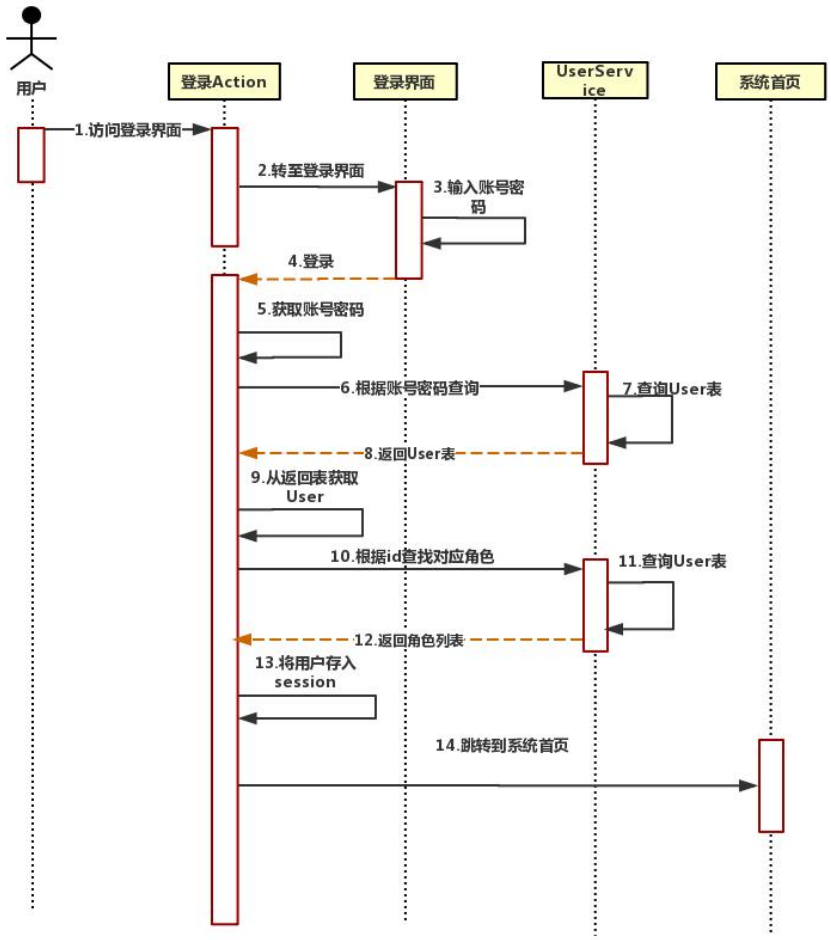
本阶段的基本目标就是解决系统如何实现问题，也叫做概要设计，本阶段主要任务是划分出系统的物理元素及设计软件的结构，完成软件定义时期的任务之后就应该对系统进行总体设计，即根据系统分析产生的分析结果来确定这个系统由哪些系统和模块组成，这些系统和模块又如何有机的结合在一起，每个模块的功能如何实现。系统设计的目标是使系统实现拥有所要求的功能，同时，力争达到高效率、高可靠性、可修改性，并且容易掌握和使用。

系统分析主要采用用例图，时序图和类结构图来体现系统的功能以及业务逻辑。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。时序图通过描述对象之间发送消息的时间顺序显示多个对象之间的动态协作。类图(Class diagram)是显示了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。

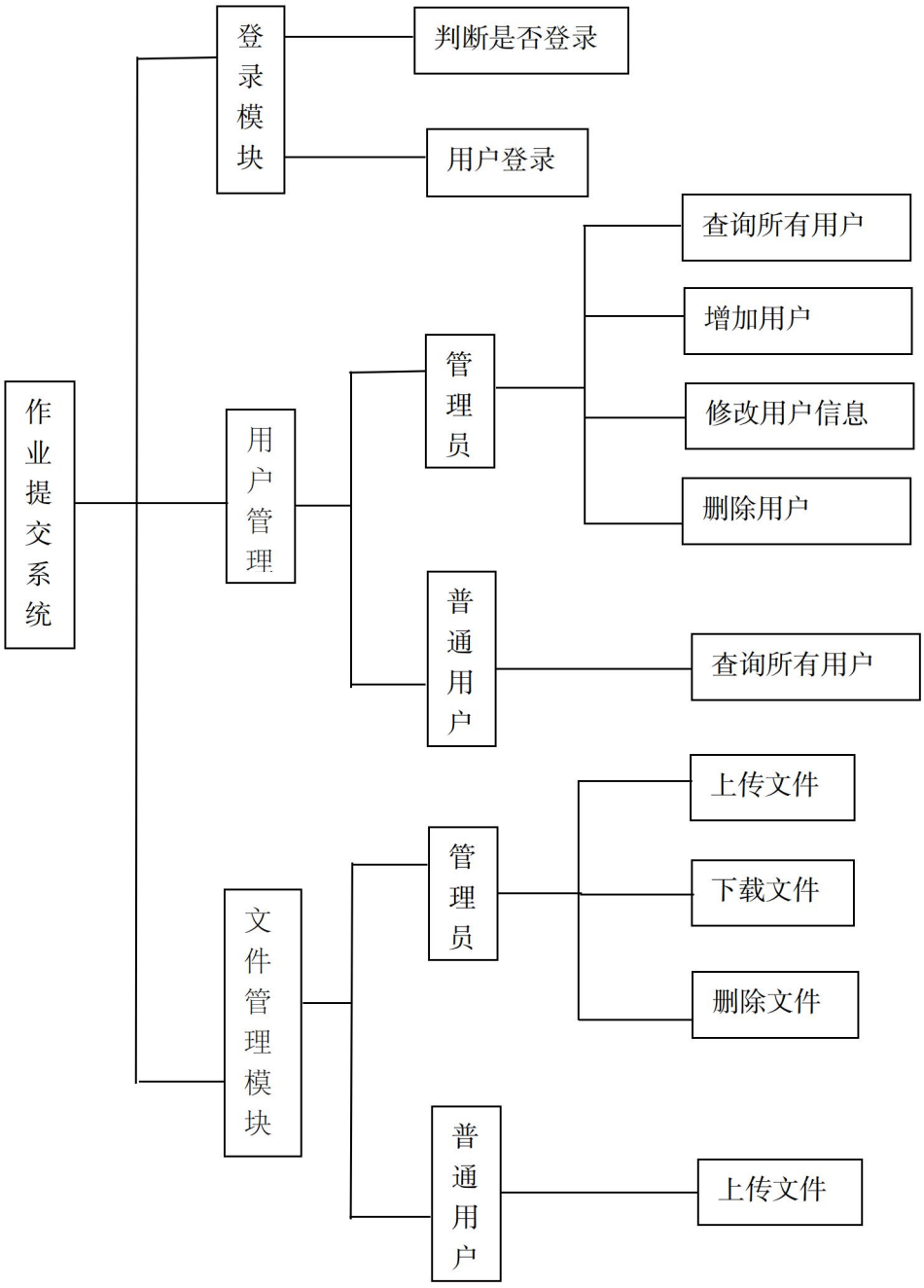
用例图


查询用户
增加用户
删除用户
修改用户信息
上传文件
下载文件
删除文件
管理员
普通用户

时序图

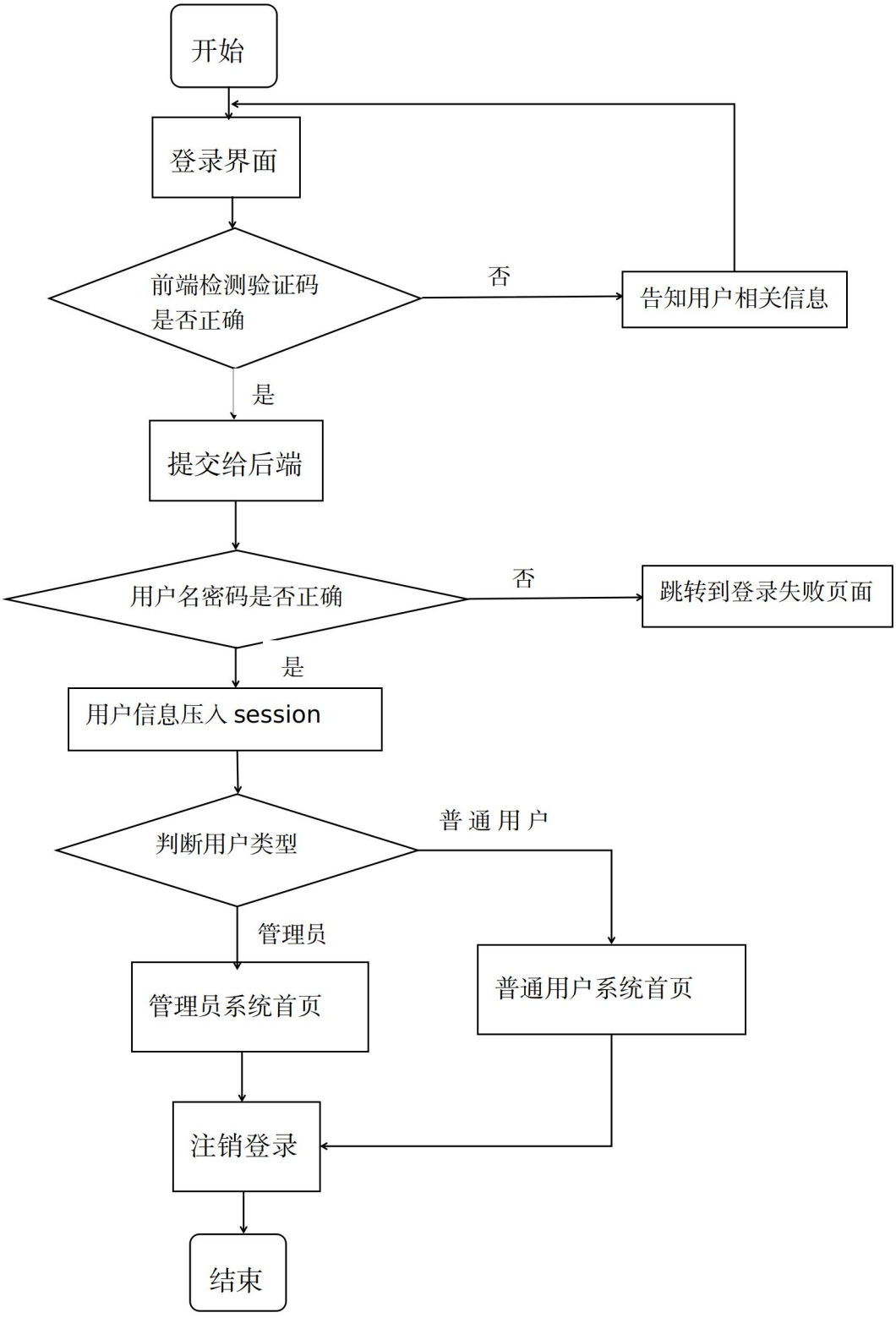

用户
登录Action
登录界面
UserServ ice
系统首页
1.访问登录界面
2.转至登录界面
3.输入账号密码
4.登录
5.获取账号密码
6.根据账号密码查询
7.查询User表
8.返回User表
9.从返回表获取User
10.根据id查找对应角色
11.查询User表
12.返回角色列表
13.将用户存入session
14.跳转到系统首页

3.2 系统模块结构图

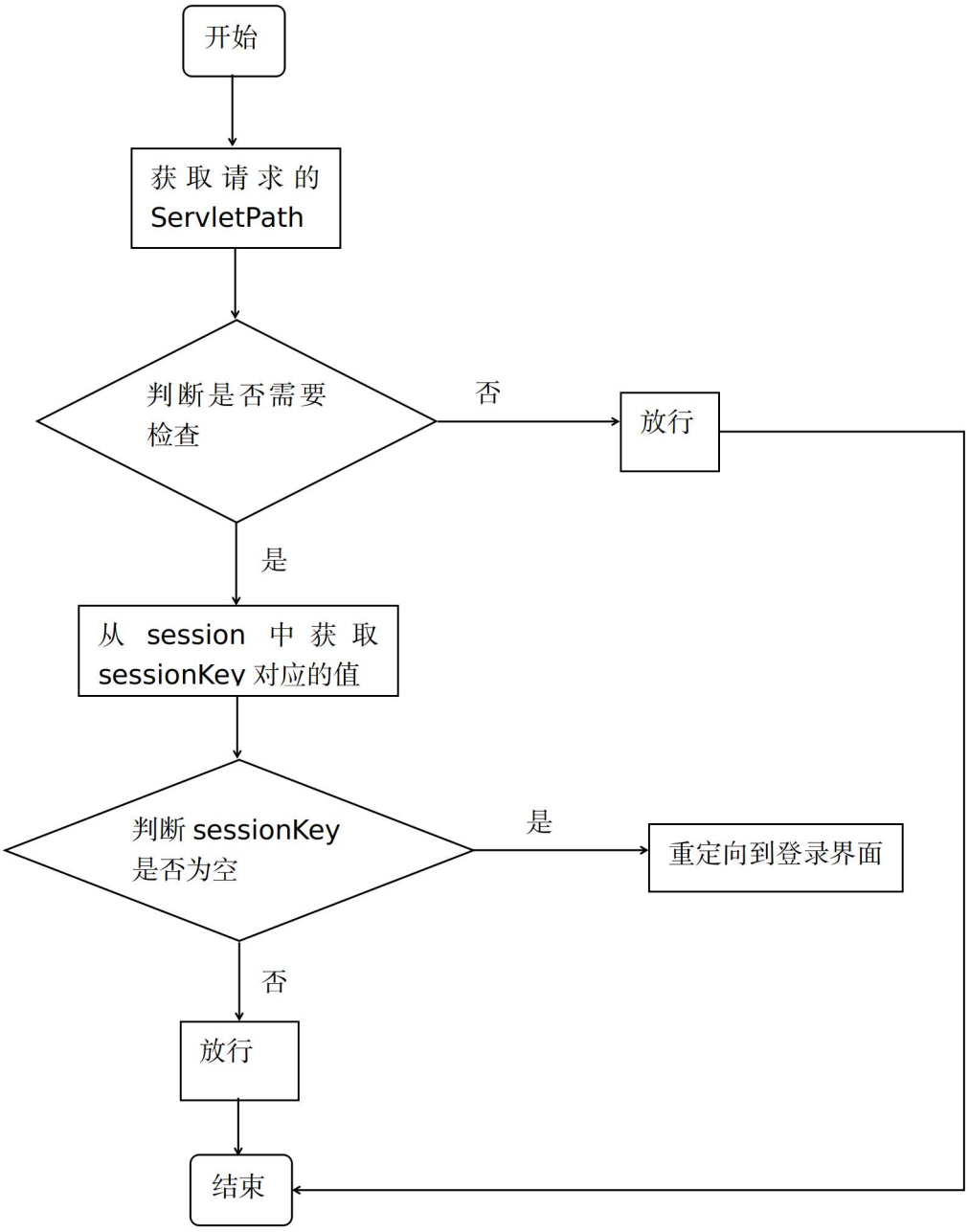

作业提交系统
登录模块
判断是否登录
用户登录
用户管理
管理员
查询所有用户
增加用户
修改用户信息
删除用户
普通用户
查询所有用户
文件管理模块
管理员
上传文件
下载文件
删除文件
普通用户
上传文件

3.3 登录模块

这一模块是登录模块，主要功能包括用户的登录，注销以及通过 Filter 检测用户是否登录。以下是登录注销的流程图。



Filter 检测用户是否登录的流程图

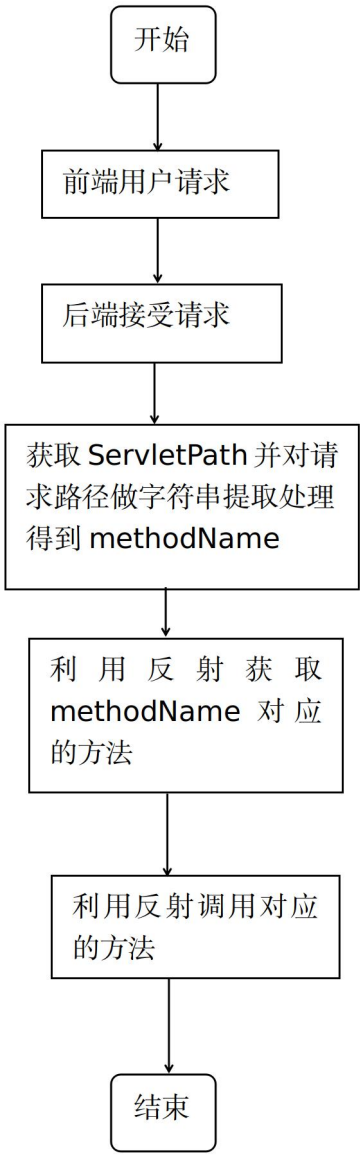


这个模块是整个系统的开始，这个模块是每个系统都必不可少的内容，用来验证用户的身份，判断用户是否属于本站。验证码功能是用前端 JS 代码完成的，验证成功之后才转发到后端进行用户名和密码的验证。验证成功之后会将用户信息存入 session 中，然后根据不同的用户类型跳转到不同的前端页面。

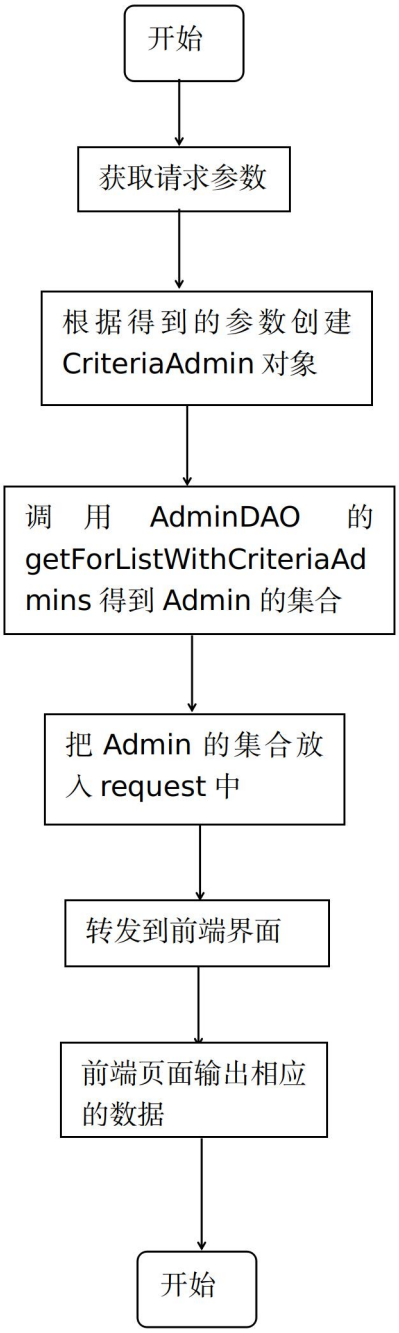
3.4 用户管理模块

用户管理模块主要包含四个部分，查询当前所有用户，添加用户，更新用户信息，删除用户。其中管理员具有全部功能，普通用户只能进行查询操作。四个功能都是用反射机制实现的，利用反射获取并调用对应的方法。由于删除操作比较简单这里不做说明。

用户管理整体流程图



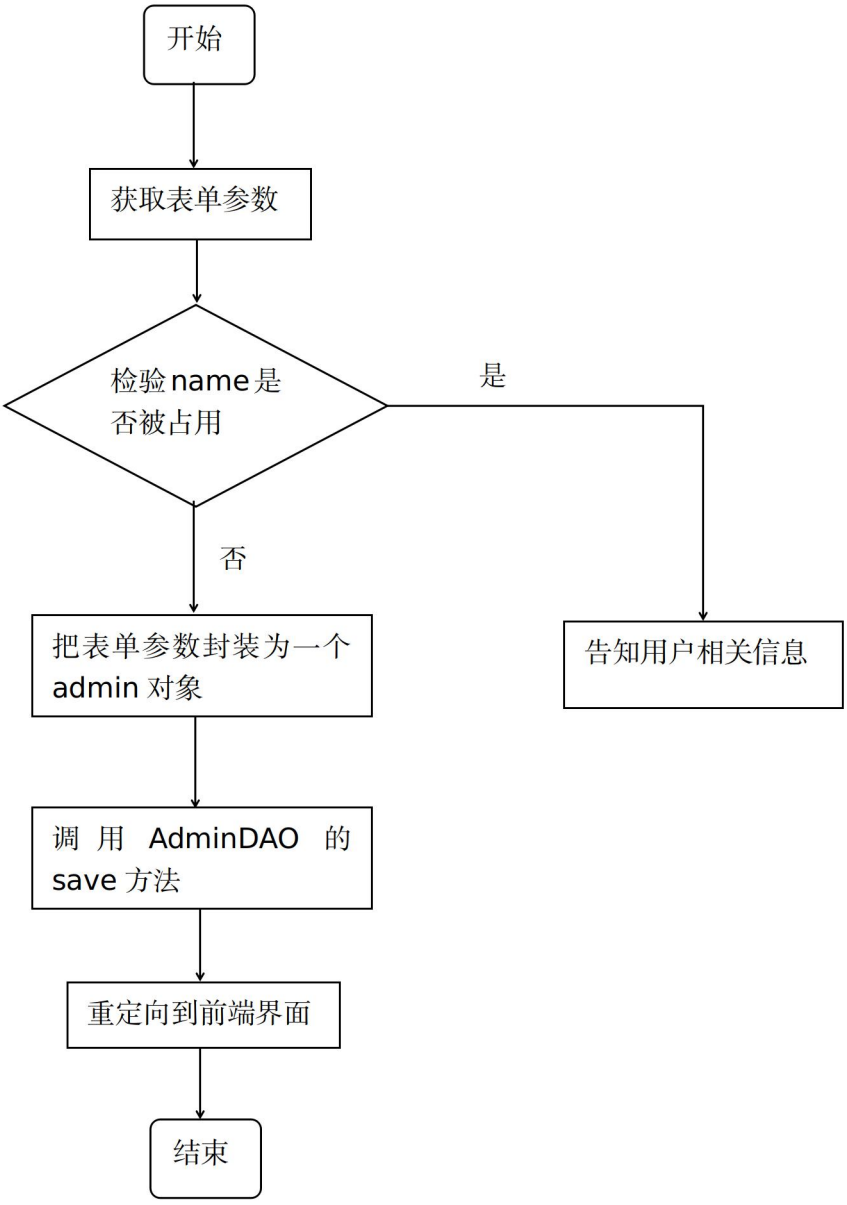
查询操作



CriteriaAdmin 主要用来做模糊查询，修改 GET 方法将获取到的参数添加模糊查询所需要的通配符。

getForListWithCriteriaAdmins 通过调用 DAO 层的 getForList 方法获取 Admin 的集合

增加操作

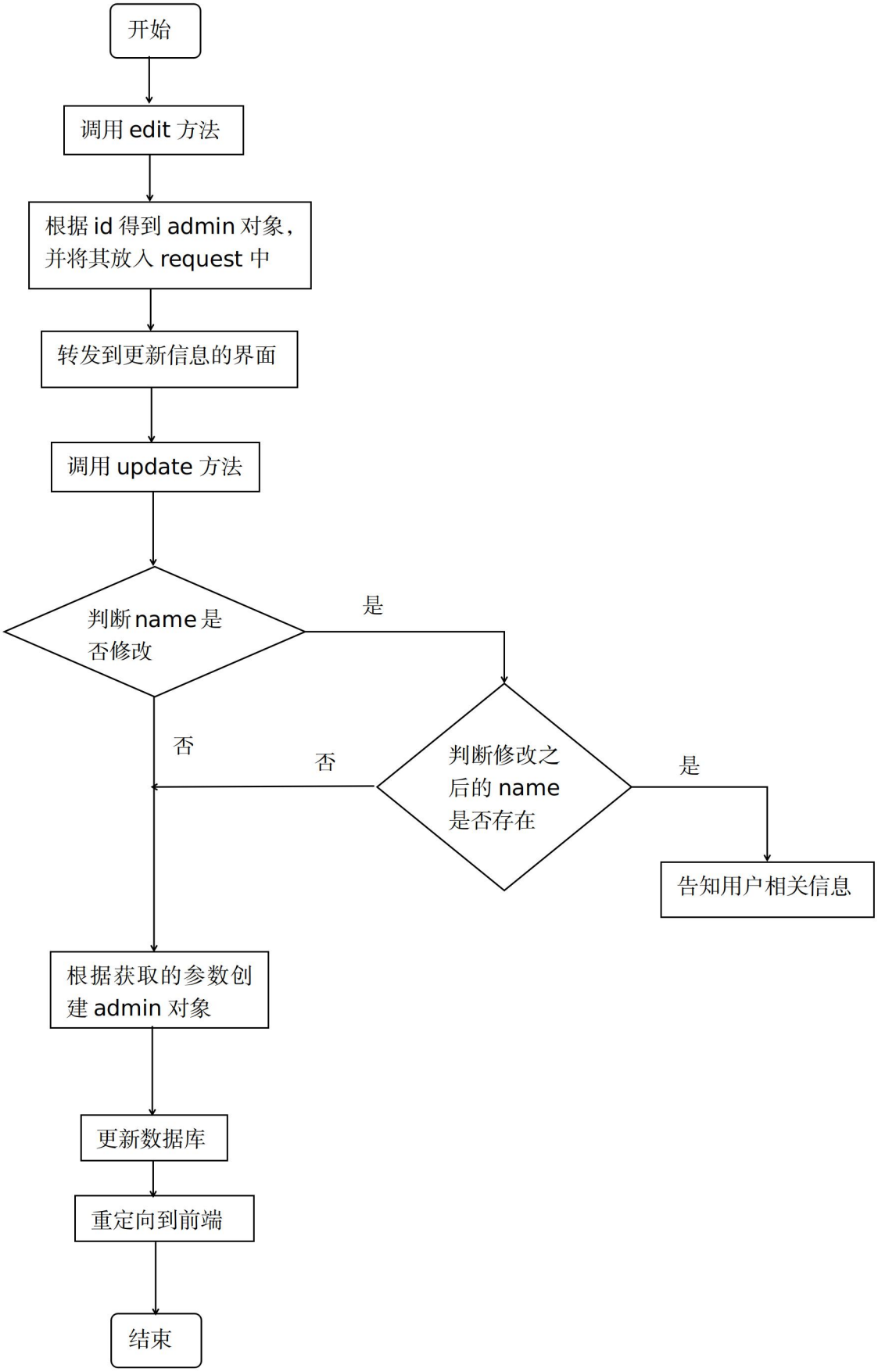


添加的时候前端 js 代码会验证输入数据的合法性，合法之后才会提交给后端处理

检验 name 是否被占用用到了 AdminDAO 的 getCountWithName(name)方法，由于为 name 字段设置了唯一约束，通过调用 DAO 的 getForValue 方法查询数据库中是否存在相同的 name 即可。

AdminDAO 的 save 方法就是通过调用 DAO 的 update 方法将信息存入数据库。

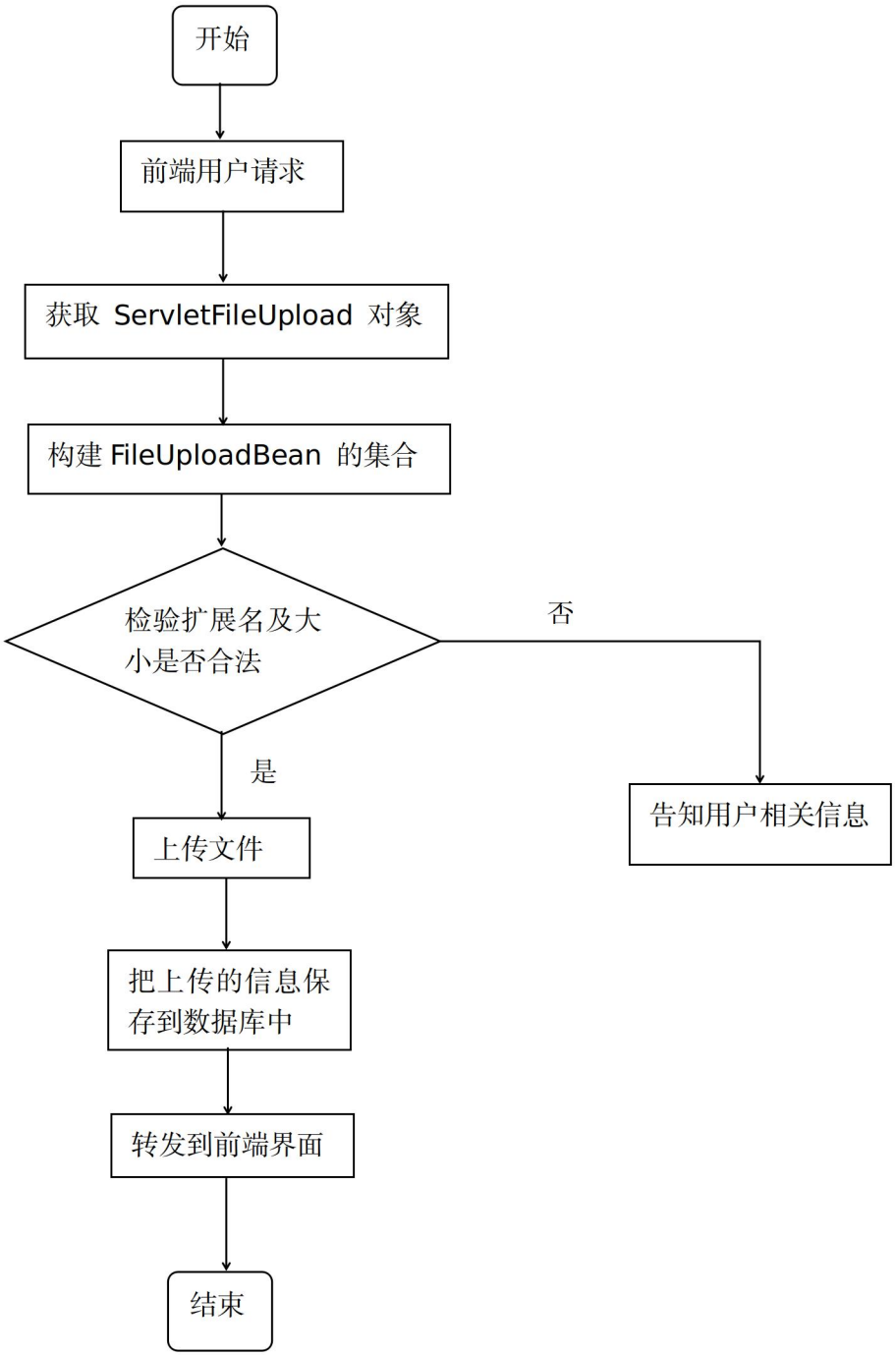
修改操作


开始
调用 edit 方法
根据 id 得到 admin 对象，并将其放入 request 中
转发到更新信息的界面
调用 update 方法
判断 name 是否修改
是
否
判断修改之后的 name 是否存在
否
是
告知用户相关信息
根据获取的参数创建 admin 对象
更新数据库
重定向到前端
结束

3.5 文件模块

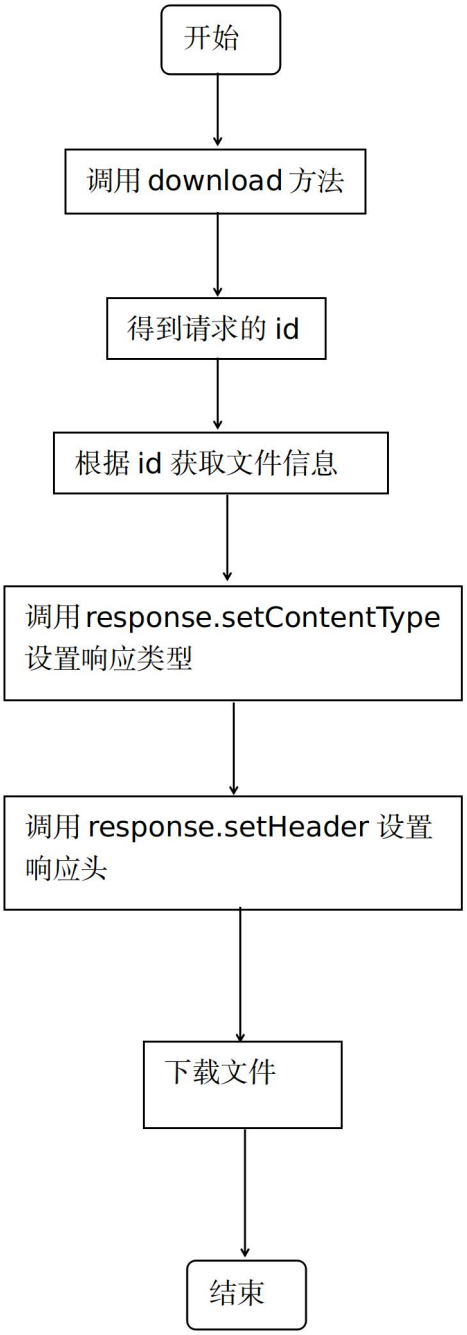
文件模块主要包含文件的上传与下载，查询当前所有文件以及删除文件。同样管理员具有全部功能，普通用户只能进行文件上传和查询。

文件上传



文件下载，文件查询，文件删除是用反射机制实现的，利用反射获取并调用对应的方法。整体流程跟上面的用户管理类似，这里不再赘述。文件删除操作只是删除了数据库中的信息，以后还值得改进。

文件下载流程图



第四章 关键技术研究

4.1 实现技术路线

总体技术路线

整个系统采用 JSPModel2 实现基于 MVC 设计模式的软件架构。通过 JSP+Servlet+JavaBean 的模式实现系统基础架构。作业提交系统基于 B/S 架构，浏览器通过 Web Server 同数据库进行数据交互。这样就大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量，降低了用户的总体成本(TCO)。

采用 MVC 设计模式的原因是 MVC 具有以下优点 1、MVC 模式实现显示模块与功能模块的分离。提高了程序的可维护性、可移植性、可扩展性与可重用性，降低了程序开发难度，降低了后期维护的难度；2、一个模型可以适用于多个视图中；3、视图与控制器的可插拔性；4、模型的可移植性，模型是独立于视图的

系统的开发工具与环境

整个系统的开发平台为 Ubuntu 操作系统。采用的语言是 java，jdk 版本为 openjdk 11.0.3，使用的开发工具为 eclipse。数据库使用的是 MySQL，版本为 Ver 14.14 Distrib 5.7.26，服务器端使用的是 tomcat 服务器，版本为 8.0.53。

Tomcat 是一个免费的开源 Servlet 容器，在 Tomcat 中，应用程序的部署很简单，只需要将 WAR 包放到 Tomcat 的 webapps 目录下，Tomcat 会自动检测这个文件，并将其解压，只是由于 Tomcat 要将 JSP 转化为 Servlet 文件，所以第一次访问有点慢

4.2 关键技术研究

4.2.1 关键技术一

第一个关键技术我想谈一下关于用户管理部分。用户管理主要是基于 MVC 设计模式，实现用户的增删改查。虽然功能比较简单，但也遇到过不少问题。整个流程我在上面的用户管理模块已经提及，这里不再概述。

首先是 DAO 层的设计，DAO 层主要是定义一些方法，供其他类调用。我的项目比较简单，不涉及复杂操作。没有业务层，直接由 Servlet 调用 DAO，所以也没有事务操作。可以直接在 DAO 中直接获取 Connection 对象。数据连接部分用到了 C3P0 连接池。

然后是 Servlet 的设计，Servlet 使用了 java 的反射机制来获取并调用相应的方法。以下是相关的代码实现。文件的查询和下载以及删除操作也是用的反射

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("utf-8");
        //获取ServletPath
    String servletPath = request.getServletPath();
    System.out.println(servletPath);
        //去除/和.do
    String methodName = servletPath.substring(7);
    methodName = methodName.substring(0, methodName.length()-3);


    Method method;
    try {
        //利用反射获取methodName对应的方法
        method = getClass().getDeclaredMethod(methodName, HttpServletRequest.class,HttpServletResponse.class);
         //利用反射调用对应的方法
        method.invoke(this, request,response);

    } catch (Exception e) {
        // TODO Auto-generated catch block
        //e.printStackTrace();
        System.out.println(1);
        response.sendRedirect("error.jsp");
    }

    }
```

Web.xml 的相关配置

```
<servlet>
  <description></description>
  <display-name>AdminServlet</display-name>
  <servlet-name>AdminServlet</servlet-name>
  <servlet-class>cn.edu.swu.servlet.AdminServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>AdminServlet</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
<servlet>
  <description></description>
  <display-name>FileServlet</display-name>
  <servlet-name>FileServlet</servlet-name>
  <servlet-class>cn.edu.swu.fileupload.servlet.FileServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>FileServlet</servlet-name>
  <url-pattern>*.is</url-pattern>
</servlet-mapping>
```

在进行管理的信息修改的时候用到了隐藏域，为了提升用户体验，将原来的信息显示在修改界面。由于 name 字段设置了唯一约束，在修改信息提交到后端之后会将修改之前的 oldname 和修改之后的 name 都传到后端，然后判断是否进行了修改，以及修改之后的 name 在数据库中是否存在，如若存在，便会告知用户。

```
private void update(HttpServletRequest request,HttpServletResponse response)throws ServletException, IOException {
    //System.out.println(1);
    String id = request.getParameter("id");
    String name = request.getParameter("name");
    String phone = request.getParameter("phone");
    String mail = request.getParameter("mail");
    String characters = request.getParameter("characters");
    String pass = request.getParameter("pass");
    String oldName = request.getParameter("oldName");
    //System.out.println(characters);
    //System.out.println(2);
    if(!oldName.equalsIgnoreCase(name)) {
        //System.out.println(3);
        long count = adminDAO.getCountWithName(name);
        if(count > 0 ) {
            //System.out.println(4);
            request.setAttribute("message", "用户名" + name + "已经被占用，请重新选择");
            request.getRequestDispatcher("admin-edit.jsp").forward(request, response);
            return;
        }}
        //System.out.println(5);
        Admin admin = new Admin(name, phone, mail, characters, pass);
        //System.out.println(characters);
        admin.setId(Integer.parseInt(id));
        //System.out.println(characters);
        adminDAO.update(admin);
        response.sendRedirect("query.do");
```

4.2.2 关键技术二

这一部分主要关于在使用 filter 判断用户登录的过程中遇到的一个问题，用户在第一次登录时候会抛出一个异常，然后在登录即可成功。抛出的异常为:Cannot call sendRedirect() after the response has been committed,当时这个问题困扰了我很久。下面贴出 filter 和 web.xml 的代码

LoginFileter.java

```
public void  doFilter(HttpServletRequest request,HttpServletResponse response ,FilterChain chain) throws IOException, ServletException{
    //1.获取请求的servletPah
    String servletPath = request.getServletPath();
    //2.检查1获取的servletPath是否为不需要检查的URL中的一个，若是，则直接放行，方法结束
    List<String> urls = Arrays.asList(uncheckedUrls.split(","));
    if(urls.contains(servletPath)) {
        chain.doFilter(request,response);
        return ;
    }
    //3.从session中获取sessionKey对应的值，若值不存在，则重定向到redirectUrl
    Object user = request.getSession().getAttribute(sessionKey);
    if(user == null) {
        response.sendRedirect(request.getContextPath() + redirecUrl);
    }
    //4.若存在，则放行，允许访问
    chain.doFilter(request, response);
    }
```

Web.xml

```
<filter>
  <filter-name>loginFilter</filter-name>
  <filter-class>cn.edu.swu.filter.login.LoginFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/admin/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/users/*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>*.html</url-pattern>
</filter-mapping>
```

修改之前的 web.xml

```
<filter>
  <filter-name>loginFilter</filter-name>
  <filter-class>cn.edu.swu.filter.login.LoginFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

首先我们需要明白 Filter 的生命周期同 servlet 的生命周期是一样的。它们都提供了 init(FilterConfig arg0)和 destroy()方法来控制。当 web 容器启动的时候，就会自动调用 init(FilterConfig arg0)来对 filter 进行初始化，当关闭 web 容器，关机，或者 reload 整个应用时，都会调用 destroy()来关闭 filter。也就是说，当 web 容器启动时，filter 就被加载到内存，并在 destroy()调用之前都常驻内存。

修改之前的 web.xml 配置的 url-pattern 为/*，这个表示项目下的所有资源包括请求。当我们第一次登录的时候，filter 检测到 login.jsp 不需要检测，直接放行。然后 LoginServlet 登录成功之后会跳转到后端的 index 页面，这个时候 filter 发现 index.jsp 需要检查，然后检查 session 中是否存在这个用户，由于第一次登录 Session 还没有创建所以会重定向到 login.jsp 界面，这也就解释了为什么第一次登录会失败。然后第二次登录的时候由于第一次已经创建了 session,所以即可成功访问。

后来我改变了 web.xml 中 filter 的 url-pattern,选择只过滤页面，这样登录就没有什么问题了。

4.2.3 关键技术三

第一部分主要是关于文件上传，对于上传多个文件存在文件的文件名与描述信息不匹配的问题。对传入 FileItem 的集合进行遍历，得到得到文件域的那些 FileItem 对象。构建的 FileUploadBean 对象，并填充 beans 和 uploadFiles

```
        for(int i = 0; i < items.size(); i++){
            FileItem item = items.get(i);
            if(item.isFormField()){
                //desc1 或 desc2 ...
                String fieldName = item.getFieldName();
                String desc = item.getString("UTF-8");
                descs.put(fieldName, desc);
            }
        }

        for(int i = 0; i < items.size(); i++){
            FileItem item = items.get(i);
            FileUploadBean bean = null;
            if(!item.isFormField()){
                String fieldName = item.getFieldName();
                String descName = "desc" + fieldName.substring(fieldName.length() - 1);
                String desc = descs.get(descName);

                //对应文件名
                String fileName = item.getName();
                String filePath = getFilePath(desc,fileName);

                bean = new FileUploadBean(fileName, filePath, desc, null);
                beans.add(bean);
                uploadFiles.put(bean.getFile_path(), item);
            }
```

第二部分主要是关于给文件命名的问题，上传文件的时候会附带一个文件描述信息，我根据这个描述信息给文件命名。主要用到了 java 的 split 方法来分割字符串，然后将分割的字符串作为文件的命名方式。可以分割中文逗号，英文逗号，以及空格。

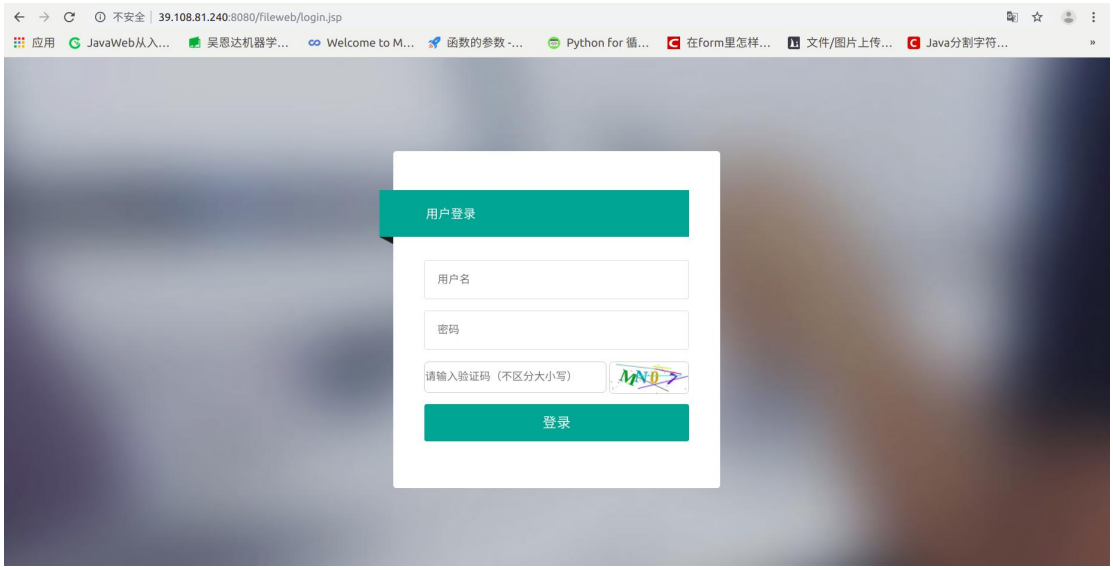
```
private String getFilePath(String desc, String fileName) {
    String extName = fileName.substring(fileName.lastIndexOf("."));
    //Random random = new Random();

    String [] tempname = desc.split(", |,| ",3);
    String descName="";
    for (int i = 0; i < tempname.length; i++) {
        descName =descName + tempname[i];
        System.out.println(descName);
    }
    String filePath = getServletContext().getRealPath(FILE_PATH) + "/" + descName + extName;
    System.out.println(filePath);
    return filePath;
}
```

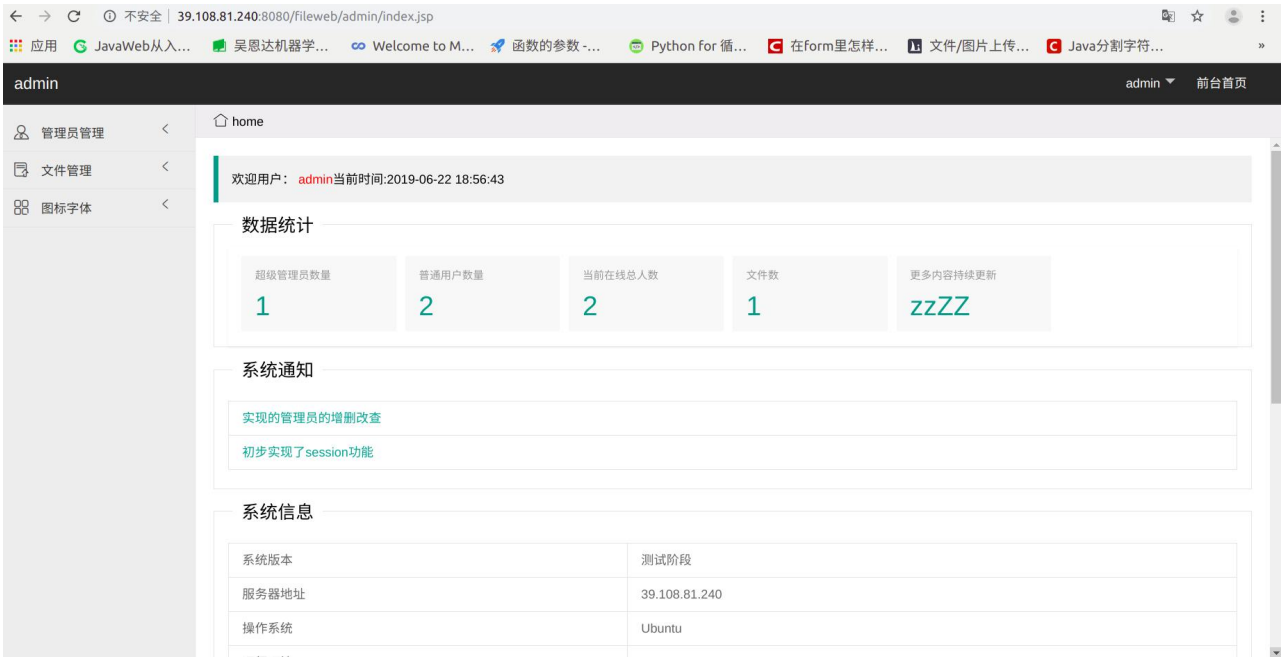
第五章 系统实现

5.1 系统实现介绍

登录界面实现验证码功能

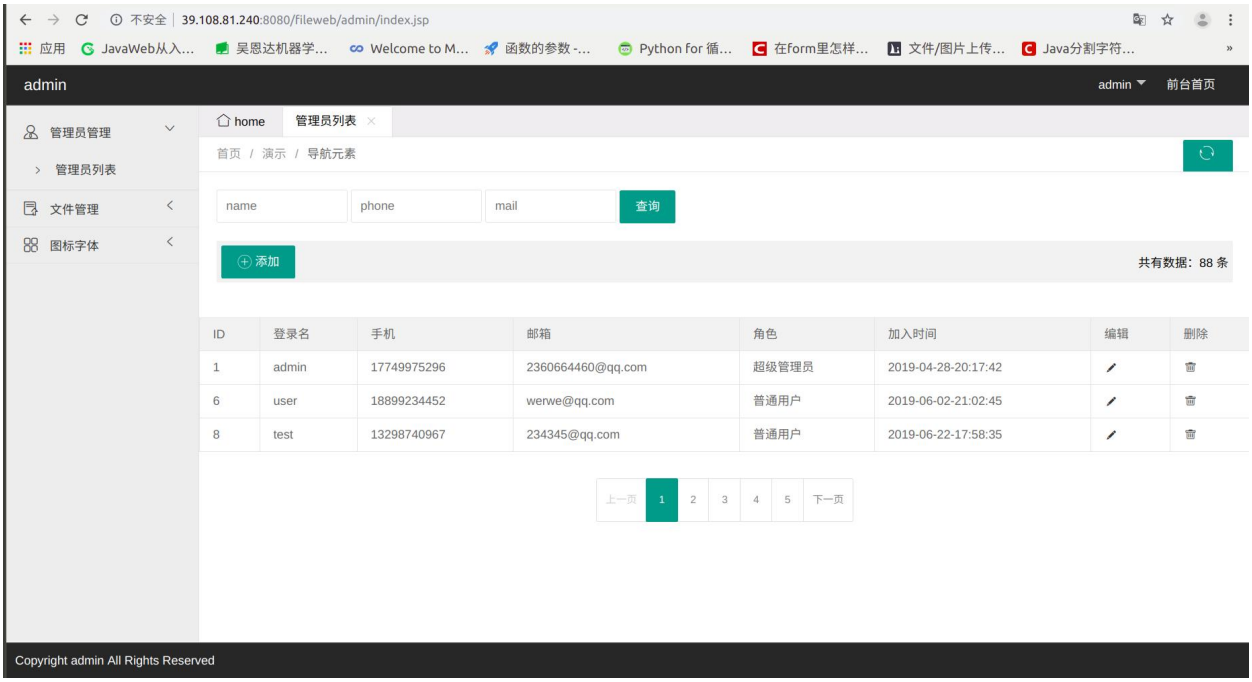


登录成功之后显示当前在线人数，统计系统文件数，不同用户数量

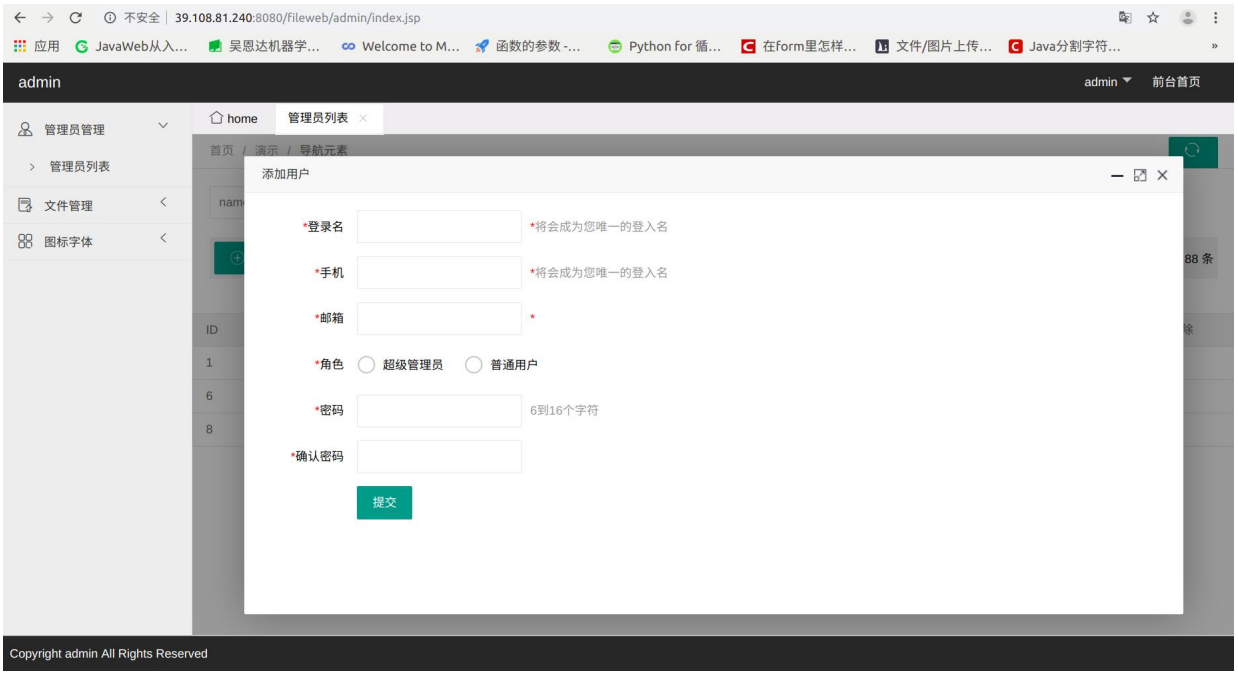


管理员可以查询当前所有用户，增加管理员或者普通用户，更新用户的信息，删除用户。普通用户只能查询当前所有用户。

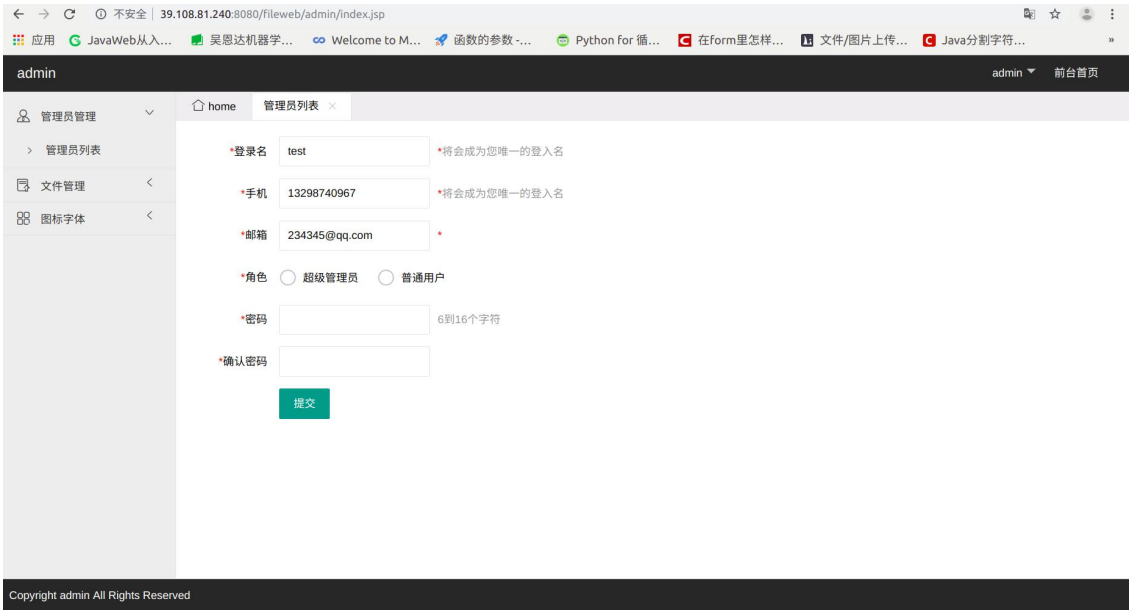
查询当前所有用户



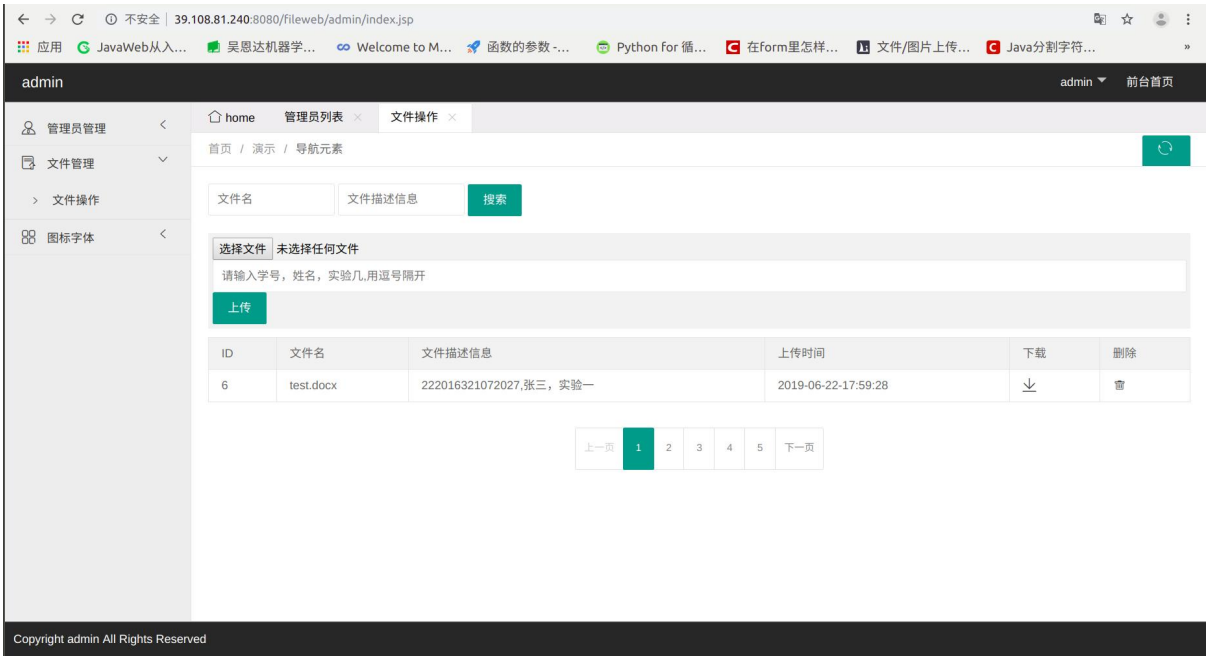
增加用户界面



修改用户信息



管理员可以上传文件，下载文件，查看当前所有文件，删除文件；普通用户只能上传文件，查看当前所有文件。



5.2 系统实现的不足

由于作者能力有限，本系统只实现了非常基础的功能，可以作为作业收集系统的一个雏形。本系统存在许多不足，主要有以下几点：

1、未能实现功能要求的分页功能

2、文件上传部分，当用户上传的文件类型或者大小超出限制的时候，只是单纯的回调到原界面，不能给用户输出提示信息。主要是因为，要输出提示信息就要求用转发,但转发属于同一个请求,上传失败再次点击查询的时候就会报错。原因是我获取请求方式的方法是字符串分割，转发之后的路径发生变化，分割字符串的时候就会发生错误，所以我就直接用重定向了。

3、文件上传不能上传到用户制定的目录，比如用户可以选择新建文件夹之类的。

4、未能实现打包下载，既然作为文件收集系统，肯定是为了方便收集文件，但目前只能实现文件的单个下载，未能实现整个目录打包下载。

5、数据库中存的只是是文件的信息，当用户删除文件，但是数据库的信息未删除时，就会下载错误，未能实现数据库的信息与文件实际信息实时更新。

6、由于本人实力有限，系统的安全性，健壮性之类的功能还需要进一步测试，目前只是用了 filter 判断用户是否登录，不知是否存在许多漏洞。