仿 AirDrop 文件传输及共享剪切板

一、软件是什么

1.1 软件功能简介

该软件是基于 Web 页面实现的（由于各种限制，只支持 Google 浏览器）。主要的功能有：文件传输、共享剪切板。

文件传输功能实现流程：当手机和电脑处于同一个局域网下时，手机（电脑）访问发布出来的 Web 页面的网址，点击上传文件即可将手机（电脑）中的文件上传到服务器中，上传的文件也可以通过访问下载页面的网址，点击下载。

共享剪切板的实现流程：当手机和电脑处于同一个局域网下时手机（电脑）可以通过表单将需要复制的内容提交到服务器端储存，在通过客户端读取服务器端储存的内容，实现手机和电脑之间的数据同步，接着通过 clipboard API 实现将传来的内容写入剪切板实现复制粘贴功能。

1.2 界面展示

软件界面如下：

图 1 PC 端文件传输及剪切板页面

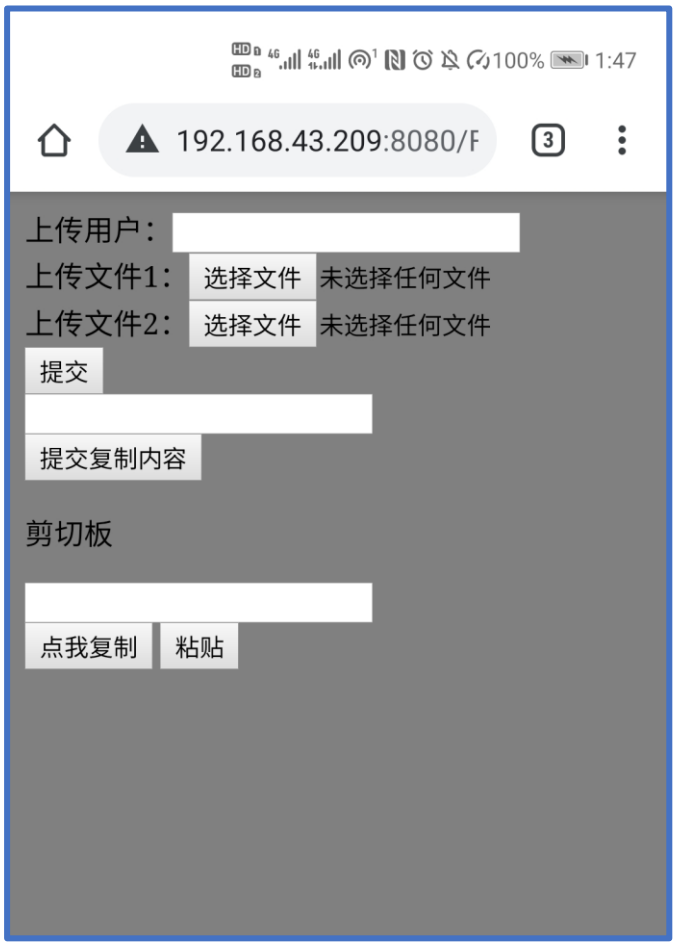

图 2 手机端文件传输及剪切板页面

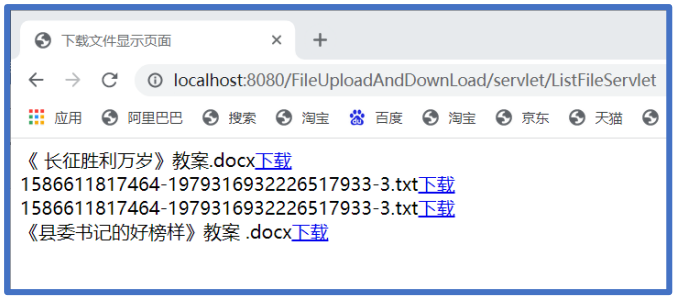

图 3 PC 端文件下载页面

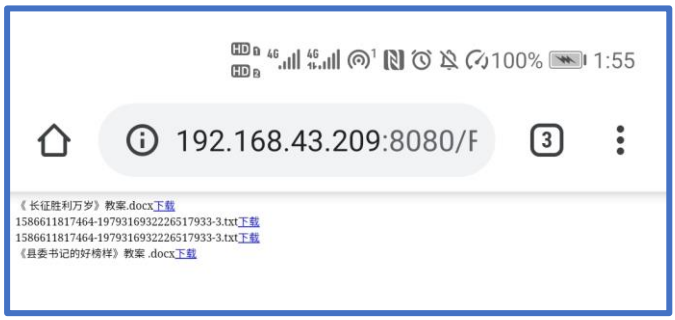

图 4 手机端文件下载页面

二、使用方式

2.1 文件传输及下载

2.1.1 文件传输

填入用户名，点击上传文件，从手机（PC 端）已有的文件中选择需要上传的文件，点击提交按钮即可上传。由于设置了单个上传文件的大小的最大值，目前是设置为 1024*1024 字节，因此传输的单个文件最大为 1MB。PC 端流程如下(手机端的流程与之相似)：

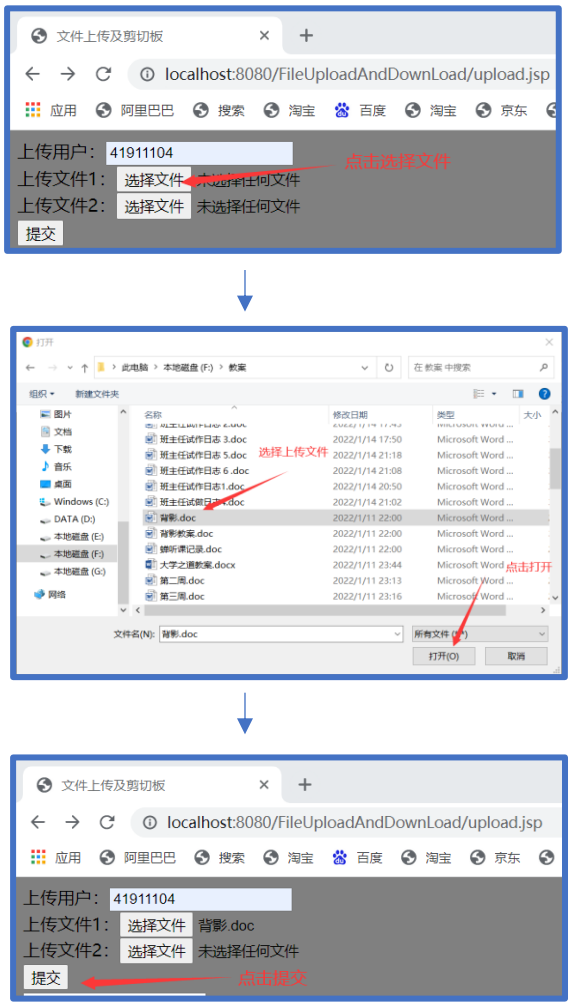

图 5 文件传输流程

2.1.2 文件下载

在手机（电脑）谷歌浏览器的网址栏中输入下载页面的网址，即会出现下载的网页，显

示出所有上传到服务器的文件，选择所需要的文件点击下载，即可下载到本地。

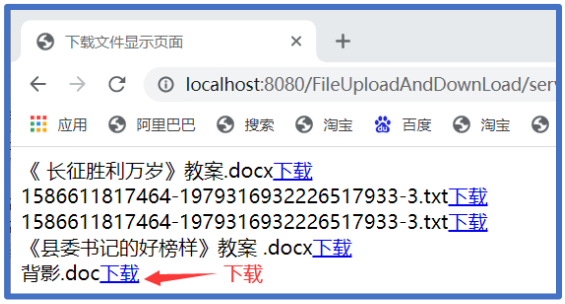

图 6 下载文件流程图

2.2 共享剪切板

2.2.1 pc 端剪切板

单独在电脑可实现的剪切板功能：功能一：在表单填入需要复制的内容，点击“提交复制内容”，将要复制的内容传到下方剪切板中（也可直接在剪切板的文本框中写入要复制的内容，该功能主要是为了实现共享剪切板），点击“点我复制”，即可将复制内容存入剪切板，点击“粘贴”实现在本网页上显示复制内容（流程见图 7）。功能二：能获得在电脑其他页面复制的内容在本页面粘贴，只需点击粘贴，即可将其它页面复制的内容粘贴在页面上。

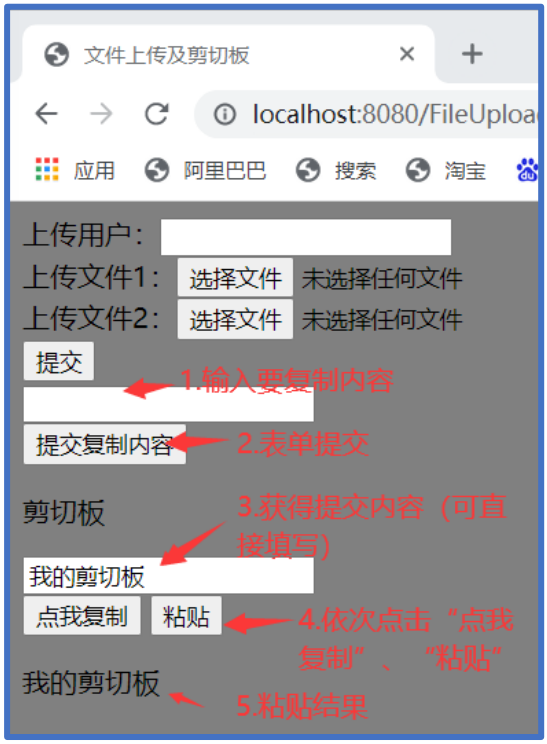

图 7 PC 端剪切板使用流程图

2.2.2 共享剪切板

共享剪切板可以实现手机端（PC 端）提交复制内容 PC 端（手机端）可以得到它并进行复制粘贴。具体的实现手机端提交复制内容 PC 端粘贴的使用流程如下（PC 端传手机端操作类似）：首先在手机端打开软件网页，按照图 7 流程图中的步骤一、二输入复制内容、提交表单数据；接着再 PC 端按照图 7 流程图的步骤四、五将上传内容复制到剪切板中进行粘贴。

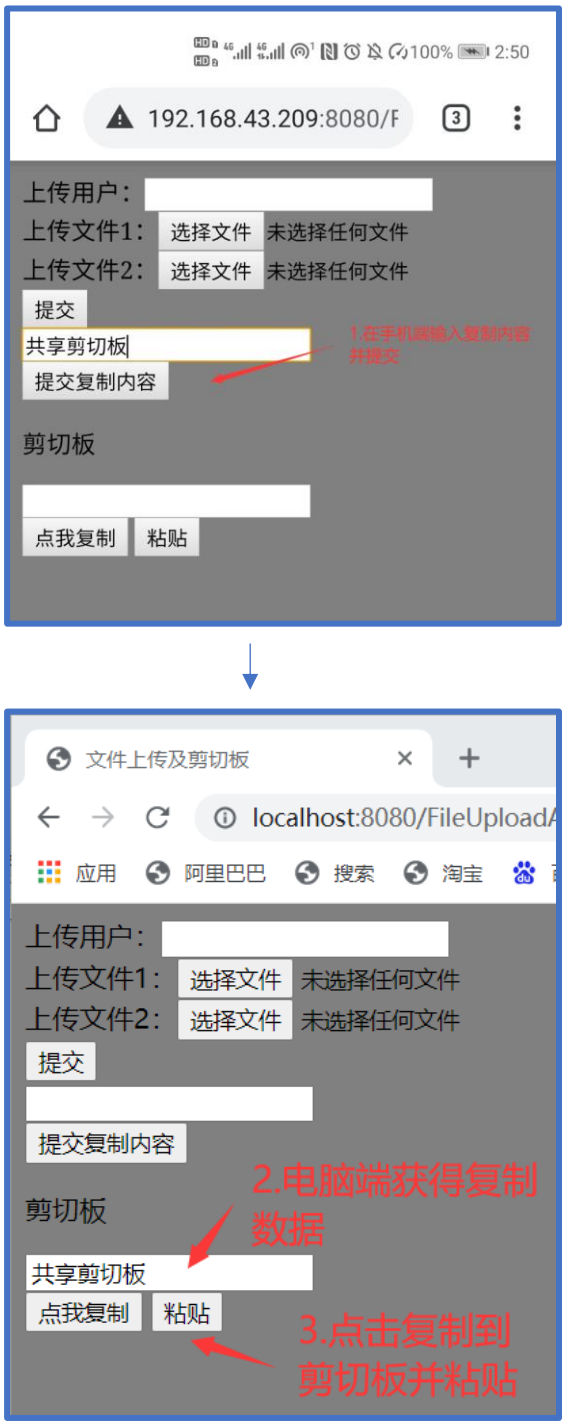

图 8 共享剪切板实现流程

三、关键代码

3.1 网页端代码 upload.jsp

upload.jsp 是实现这个软件主要前端界面的代码。主要用来让用户上传文件、实现表单提交及剪切板。

```
<%@ page language="java" pageEncoding="UTF-8" import="java.io.*"%>
 <!DOCTYPE HTML>
 <html>
 <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>文件上传及剪切板</title>
    <link rel="stylesheet" href="css/styl.css">
 </head>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/clipboard.js/1.7.1/clipboard.min.js"></script>
 <%

 File file=new File("E:/log1.txt");
 String p=null;
try {
//2构建文件输入流对象
       InputStream fileInoutStream=new FileInputStream(file);
//3、构建读取的容器
byte[] bytes=new byte[1024];
       StringBuilder stringBuilder=new StringBuilder();
int len=-1;//表示每次读取的字节长度
//read 传入数组 把数据读入到数组中，并返回读取的字节数，当不等于-1时，表示读取到数据，等于-1，表示读完了
while ((len=fileInoutStream.read(bytes))!=-1){
//stringBuilder.append(new String(bytes));
           stringBuilder.append(new String(bytes,0,len));
        }
        System.out.println(stringBuilder.toString());
   if(stringBuilder.toString()!=null){
       p=stringBuilder.toString();
   }else{
       p="请输入要复制内容";
   }
        fileInoutStream.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
 %>
 <body>
 <form action="UploadHandleServlet" enctype="multipart/form-data" method="post">
 上传用户：<input type="text" name="username"><br/>
 上传文件1：<input type="file" name="file1"><br/>
 上传文件2：<input type="file" name="file2"><br/>
  <input type="submit" value="提交">
 </form>
 <form action="ClipServlet" method="post">
 <input type="text" name="deliver" id="deliver"><br/>
 <input type="submit" value="提交复制内容">
 </form>
    <p>剪切板</p>
    <input type="text" id="copy-text" value=<%=p%>><br>

    <button id="click-copy">点我复制</button>
    <button id="click-paste">粘贴</button><br>
    <p class="paste-space"></p>
    <p class="paste-space"></p>
    <p class="paste-space"></p>
</body>
<script src="js/clip.js"></script>
</html>
```

图 9 upload.jsp 代码截图

3.2UploadHandleServlet.java

该 servlet 实现从客户端获得得到要上传文件的保存目录，将上传的文件存放于 WEB-INF 目录下，不允许外界直接访问。同时将限制传输文件的单个大小限制为 1MB；通过文件的扩展名来判断上传的文件类型是否合法；为防止文件覆盖的现象发生，通过文件名的 hash 值为文件分配内容地址、为上传文件产生一个唯一的文件名。

```
//获取item中的上传文件的输入流
InputStream in = item.getInputStream();
//得到文件保存的名称
String saveFilename = makeFileName(filename);
//得到文件的保存目录
System.out.println( "saveFilename:"+saveFilename);
String realSavePath = makePath(saveFilename, savePath);
//创建一个文件输出流
System.out.println( "realSavePath:"+realSavePath);
FileOutputStream out = new FileOutputStream(realSavePath + "\\" + saveFilename);
//创建一个缓冲区
byte buffer[] = new byte[1024];
  int len = 0;
//循环将输入流读入到缓冲区当中，(len=in.read(buffer))>0就表示in里面还有数据
while((len=in.read(buffer))>0){
 //使用FileOutputStream输出流将缓冲区的数据写入到指定的目录(savePath + "\\" + filename)当中


    //System.out.print(buffer);
 out.write(buffer, 0, len);
}
//关闭输入流
in.close();
//关闭输出流
out.close();    //删除处理文件上传时生成的临时文件
item.delete();    message = "文件上传成功！";
}
}
}catch (FileUploadBase.FileSizeLimitExceededException e) {
e.printStackTrace();
```

图 10 UploadHandleServlet.java 核心上传代码

3.3 ListFileServlet.java

该 servlet 主要实现将上传到服务器的文件及其下载的链接发送到客户端，使用户能够点击下载上传的文件。

```java
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
//获取上传文件的目录
String uploadFilePath = this.getServletContext().getRealPath("/WEB-INF/upload");
System.out.println( uploadFilePath);
//存储要下载的文件名
Map<String,String> fileNameMap = new HashMap<String,String>();
//递归遍历filepath目录下的所有文件和目录，将文件的文件名存储到map集合中
listfile(new File(uploadFilePath),fileNameMap);//File既可以代表一个文件也可以代表一个目录
//将Map集合发送到listfile.jsp页面进行显示
request.setAttribute("fileNameMap", fileNameMap);
request.getRequestDispatcher("/listfile.jsp").forward(request, response);
}

public void listfile(File file,Map<String,String> map){
//如果file代表的不是一个文件，而是一个目录
if(!file.isFile()){
//列出该目录下的所有文件和目录
File files[] = file.listFiles();
//遍历files[]数组
for(File f : files){
//递归
listfile(f,map);
}
}else{
/**
* 处理文件名，上传后的文件是以uuid_文件名的形式去重新命名的，去除文件名的uuid_部分
file.getName().indexOf("_")检索字符串中第一次出现"_"字符的位置，如果文件名类似于：9349249849-88343-8344_阿_凡_达.avi
那么file.getName().substring(file.getName().indexOf("_")+1)处理之后就可以得到阿_凡_达.avi部分
*/
String realName = file.getName().substring(file.getName().indexOf("_")+1);
//file.getName()得到的是文件的原始名称，这个名称是唯一的，因此可以作为key，realName是处理过后的名称，有可能会重复
map.put(file.getName(), realName);
}
}
```

图 11ListFileServlet.java 核心上传代码

3.4clip.js

该 js 通过 clipboard API 实现实时监听剪切板内容，进行复制粘贴操作。代码如下：

```javascript
const copy = function () {
    navigator.clipboard.writeText(document.getElementById('copy-text').value)
      .then(function () {
            console.log("CopyOK!");
        }, function () {
            console.log('CopyNoOK!');
        });
};

const copybutton = document.getElementById('click-copy');
  copybutton.addEventListener('click', function(){
    copy();
  }, false);


const paste = function(){
navigator.clipboard.readText()
.then(function (text) {
    countUp();
    document.getElementsByClassName('paste-space')[CountValue].textContent = text;
    console.log("剪切板的内容为" + text )
}, function () {
  console.log('粘贴失败');
    });
};

const pastebutton = document.getElementById('click-paste');
  pastebutton.addEventListener('click', function(){
    paste();
  }, false);
```

图 12 clip.js

四、单元测试

通过 eclipse 的 JunitTest 对进行单元测试 ListFileServlet.java 以及 UploadHandleServlet.java 进行单元测试 ，测试成功。测试结果如下图：

图 13 UploadHandleServlet.java 测试结果

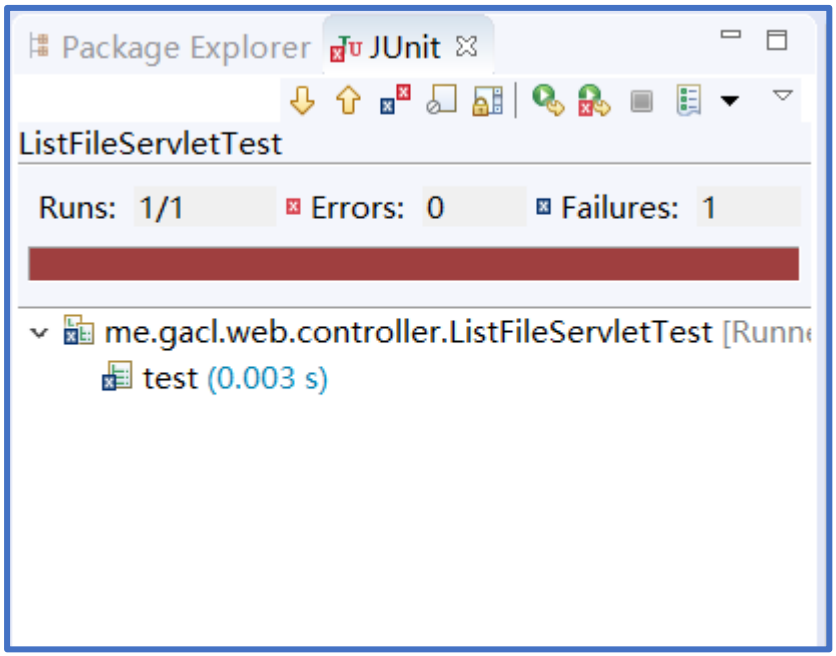

图 14 ListFileServlet.java 测试结果

五、持续集成

用 git 命令将整个源代码提交到 GitHub 的个人仓库上在通过 GitHub 自带的 GitHub Action 持续集成工具进行集成。通过 Github Action 集成的做法:在个人仓库的根目录下，创建一个 .github 文件夹,里面放一个 *.yaml 文件，通过 Yaml 文件配置 Github Action 所用的文件。

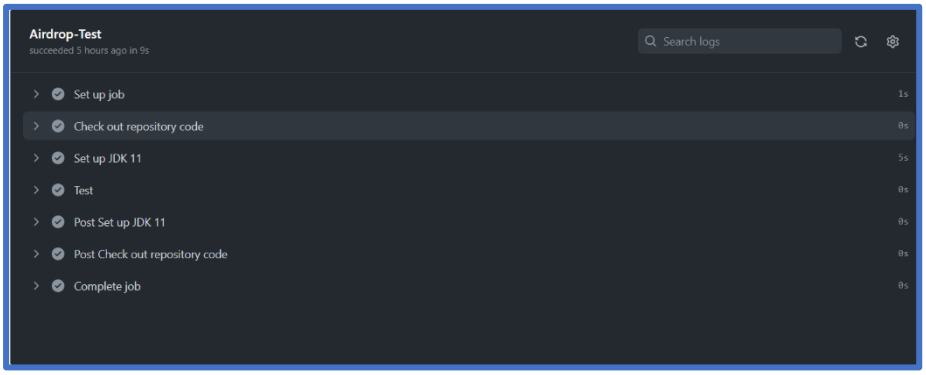

图 15 Github Action 集成结果

六、亮点

本软件有两个亮点，其中之一是在文件上传的过程中通过文件的 hash 值分配地址，为文件设置唯一的文件名解决了文件覆盖的问题。第二个是在网页写的剪切板可以获得在本设备其它部分写入剪切板的内容，从而在网页端可以粘贴出来。

七、总结与反思

本项目在实现仿 AirDrop 文件上传基本没有问题，但在实现共享剪切板时任有问题存在。首先在剪切板功能中使用了 clipboard API 对复制、粘贴操作进行监听，由于非 https 的网页（除 localhost 外）不能使用 clipboard API，而通过 Tomcat 发布的网页只能通过 http 访问，因此在手机端的剪贴板功能不能实现。其次由于浏览器的兼容问题，文件传输功能及剪切板只能在谷歌浏览器运行。最后剪切板的共享功能只能通过表单提交实现，并不能实现理想状态下的手机端复制内容，电脑端直接就可以粘贴的功能。因此本软件使用的方案不是个很好的方案，仍有很大的改进空间。