Java Web文件下载1——文件下载实现和乱码处理

<https://blog.csdn.net/sheen1991/article/details/39084251?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task>

文件上传和下载是web开发中常遇到的问题，这几天在做一个项目又用到了文件下载，之前也零零散散记了些笔记，今天来做一下整理。文件上传还有待进一步测试，这里先说一下文件下载。

一、文件下载处理流程

文件下载处理流程其实很清晰，即：

1、根据文件名或者文件路径定位文件，具体的策略主要根据自己的需求，总之需要系统能找到的文件全路径。

2、获取输入流，从目标文件获取输入流。

3、获取输出流，从response中获取输出流。

4、从输入流读入文件，通过输出流输出文件。这是真正的下载执行过程。

5、关闭IO流。

主要流程就是这个，另外就是一些必要的属性设置，比如比较重要的有设置文件的contentType类型等。

二、不啰嗦了了，上代码

我是用Springmvc做的，但其实用其他的也一样，主要需要HttpServletResponse对象和有效的目标文件。

1、前台代码

|  |
| --- |
| /\*  \* 下载上传的文件  \*/  function downloadFromUpload(fileName){  window.location.href = path + "/download?dir=upload&fileName="+encodeURI(encodeURI(fileName));  }  /\*  \* 普通下载  \*/  function download(fileName){  window.location.href = path + "/download?dir=download&fileName="+encodeURI(encodeURI(fileName));  } |

2、controller代码

|  |
| --- |
| /\*\*  \* 文件下载(从上传路径下载)  \*  \* @param request  \* @param response  \* @throws IOException  \*/  @ResponseBody  @RequestMapping(value = "/download")  public void downloadFile(HttpServletRequest request,  HttpServletResponse response, FileModel model) throws Exception {  String fileName = URLDecoder.decode(model.getFileName(), "UTF-8");  /\*  \* 限制只有upload和download文件夹里的文件可以下载  \*/  String folderName = "download";  if (!StringUtils.isEmpty(model.getDir())  && model.getDir().equals("upload")) {  folderName = "upload";  } else {  folderName = "download";  }  String fileAbsolutePath = request.getSession().getServletContext()  .getRealPath("/")  + "/WEB-INF/" + folderName + "/" + fileName;  FileTools.downloadFile(request, response, fileAbsolutePath);  log.warn("用户Id："  + (Integer) (request.getSession().getAttribute("userId"))  + ",用户名："  + (String) (request.getSession().getAttribute("username"))  + ",下载了文件:" + fileAbsolutePath);  } |

这里的下载逻辑是，前台只需要请求/download，并给出文件名参数即可。为了避免中文乱码，前台的文件名在作为参数时，使用了js的encodeURI（）将其变为Unicode码，然后后台解码转换为中文。另外由于项目的特殊性，我这里要下载的文件可能会在upload和download两个文件夹中，所以这里多了一部分判断逻辑。另外，我这里将文件名和请求的文件夹名称都封装在了FileModel中。

3、下载逻辑实现。这里没有用service了，直接用的静态方法实现。

|  |
| --- |
| /\*\*  \* 下载文件时指定下载名  \*  \* @param request  \* HttpServletRequest  \* @param response  \* HttpServletResponse  \* @param filePath  \* 文件全路径  \* @param fileName  \* 指定客户端下载时显示的文件名  \* @throws IOException  \*/  public static void downloadFile(HttpServletRequest request,  HttpServletResponse response, String filePath, String fileName)  throws IOException {  BufferedInputStream bis = null;  BufferedOutputStream bos = null;    bis = new BufferedInputStream(new FileInputStream(filePath));  bos = new BufferedOutputStream(response.getOutputStream());    long fileLength = new File(filePath).length();    response.setCharacterEncoding("UTF-8");  response.setContentType("multipart/form-data");  /\*  \* 解决各浏览器的中文乱码问题  \*/  String userAgent = request.getHeader("User-Agent");  byte[] bytes = userAgent.contains("MSIE") ? fileName.getBytes()  : fileName.getBytes("UTF-8"); // fileName.getBytes("UTF-8")处理safari的乱码问题  fileName = new String(bytes, "ISO-8859-1"); // 各浏览器基本都支持ISO编码  response.setHeader("Content-disposition",  String.format("attachment; filename=\"%s\"", fileName));    response.setHeader("Content-Length", String.valueOf(fileLength));  byte[] buff = new byte[2048];  int bytesRead;  while (-1 != (bytesRead = bis.read(buff, 0, buff.length))) {  bos.write(buff, 0, bytesRead);  }  bis.close();  bos.close();    }    /\*\*  \* 下载文件时不指定下载文件名称  \*  \* @param request  \* HttpServletRequest  \* @param response  \* HttpServletResponse  \* @param filePath  \* 文件全路径  \* @throws IOException  \*/  public static void downloadFile(HttpServletRequest request,  HttpServletResponse response, String filePath) throws IOException {  File file = new File(filePath);  downloadFile(request, response, filePath, file.getName());  } |

这里提供了重载的下载方法，解决有时需要指定客户端下载的文件名的需求。

三、注意事项

1、关于MIME类型的选择

之前对MIME类型不是很了解，发现网上有很多下载的源码的MIME类型设置的不一样。即这句

|  |
| --- |
| response.setContentType("multipart/form-data"); |

查了下这里设置MIME类型的一个作用是告诉客户端浏览器以什么格式处理要下载的文件。具体的对应网上有很多讲解，这I类设置成这种格式，一般会自动匹配格式。

2、指定客户端下载文件名

有时我们可能需要指定客户端下载文件时的文件名，即这句代码

|  |
| --- |
| response.setHeader("Content-disposition", String.format("attachment; filename=\"%s\"", fileName)); |

中的fileName，可以自定义。前面的部分一般不要动。

3、中文乱码问题的解决

中文文件乱码太常见了，在项目系统架构刚搭建时，就应该统一所有的中文编码，包括编辑器中、页面中以及数据库中，推荐UTF-8编码。如果用的Spring，还可以配置Spring的字符集过滤器，进一步避免中文乱码。

（1）客户端下载请求过程文件名乱码

有时我们会遇到，前台页面显示中文文件名下载列表时正常的，但我们到后台发现请求中的文件名乱码了，这时采用前面所说的encodeURI可以解决。

（2）客户端下载执行时文件名乱码

在实际测试中发现，在其他浏览器都可以执行的情况下，ie下中文文件名可能会出现乱码。在网上看到了这样一段代码，经测试，完美解决了不同浏览器的中文乱码问题

|  |
| --- |
| /\*  \* 解决各浏览器的中文乱码问题  \*/  String userAgent = request.getHeader("User-Agent");  byte[] bytes = userAgent.contains("MSIE") ? fileName.getBytes()  : fileName.getBytes("UTF-8"); // fileName.getBytes("UTF-8")处理safari的乱码问题  fileName = new String(bytes, "ISO-8859-1"); // 各浏览器基本都支持ISO编码  response.setHeader("Content-disposition",  String.format("attachment; filename=\"%s\"", fileName)); |

（3）服务器上文件乱码

不同的服务器可能因平台的不同编码方式也不同，这里也需要注意。具体的解决方案请参见之前写过的一篇文章：文件下载过程中中文乱码处理

原创作品，转载请注明出处。

————————————————

版权声明：本文为CSDN博主「sheen1991」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/sheen1991/article/details/39084251