

# 文件的上传和下载

讲师：王振国

## 今日内容

## 文件的上传和下载

文件的上传和下载，是非常常见的功能。很多的系统中，或者软件中都经常使用文件的上传和下载。

比如：QQ 头像，就使用了上传。

邮箱中也有附件的上传和下载功能。

OA 系统中审批有附件材料的上传。

## 1、文件的上传介绍（\*\*\*\*\*重点）

- 1、要有一个 form 标签，method=post 请求
- 2、form 标签的 encType 属性值必须为 multipart/form-data 值
- 3、在 form 标签中使用 input type=file 添加上传的文件
- 4、编写服务器代码（Servlet 程序）接收，处理上传的数据。

encType=multipart/form-data 表示提交的数据，以多段（每一个表单项一个数据段）的形式进行拼接，然后以二进制流的形式发送给服务器

## 1.1、文件上传，HTTP 协议的说明。

### 文件上传时发送的HTTP协议内容

```
POST /09_EL_JSTL/uploadServlet HTTP/1.1
Host: 192.168.31.74:8080
Connection: keep-alive
Content-Length: 4647
Cache-Control: max-age=0
Origin: http://192.168.31.74:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryc12joVsAFxzitHaW
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://192.168.31.74:8080/09_EL_JSTL/upload.jsp
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: JSESSIONID=588D36868550E65F52155C005F86CFDC
```

空行

-----WebKitFormBoundaryc12joVsAFxzitHaW → 表示一段数据的开始

Content-Disposition: form-data; name="username"

空行

wzg168 → 当前表单项的值

-----WebKitFormBoundaryc12joVsAFxzitHaW → 表示另一段数据的开始

Content-Disposition: form-data; name="photo"; filename="d.jpg"

Content-Type: image/jpeg

空行

上传的文件的数据

-----WebKitFormBoundaryc12joVsAFxzitHaW-- → 多了两个减号的分隔符，表示数据的结束标记

Content-Type 表示提交的数据类型

multipart/form-data 表示提交的数据，以多段（每一个表单项一个数据段）的形式进行拼接，然后以二进制流的形式发送给服务器

boundary 表示每段数据的分隔符

----WebKitFormBoundaryc12joVsAFxzitHaW是由浏览器每次都随机生成。它就是每段数据的分界符。

## 1.2、commons-fileupload.jar 常用 API 介绍说明

commons-fileupload.jar 需要依赖 commons-io.jar 这个包，所以两个包我们都要引入。

第一步，就是需要导入两个 jar 包：

commons-fileupload-1.2.1.jar

commons-io-1.4.jar

commons-fileupload.jar 和 commons-io.jar 包中，我们常用的类有哪些？

ServletFileUpload 类，用于解析上传的数据。

FileItem 类，表示每一个表单项。

```
boolean ServletFileUpload.isMultipartContent(HttpServletRequest request);
```

判断当前上传的数据格式是否是多段的格式。

```
public List<FileItem> parseRequest(HttpServletRequest request)
```

解析上传的数据

```
boolean FileItem.isFormField()
```

判断当前这个表单项，是否是普通的表单项。还是上传的文件类型。

true 表示普通类型的表单项

false 表示上传的文件类型

```
String FileItem.getFieldName()
```

获取表单项的 name 属性值

```
String FileItem.getString()
```

获取当前表单项的值。

```
String FileItem.getName();
```

获取上传的文件名

```
void FileItem.write( file );
```

将上传的文件写到 参数 file 所指向抽硬盘位置 。

## 1.3、fileupload 类库的使用：

上传文件的表单：

```
<form action="http://192.168.31.74:8080/09_EL_JSTL/uploadServlet" method="post"
enctype="multipart/form-data">
    用户名: <input type="text" name="username" /> <br>
    头像: <input type="file" name="photo" > <br>
    <input type="submit" value="上传">
</form>
```

解析上传的数据的代码：

```
/**
 * 用来处理上传的数据
 * @param req
 * @param resp
 * @throws ServletException
 * @throws IOException
 */
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

    //1 先判断上传的数据是否多段数据（只有是多段的数据，才是文件上传的）
    if (ServletFileUpload.isMultipartContent(req)) {
//        创建 FileItemFactory 工厂实现类
        FileItemFactory fileItemFactory = new DiskFileItemFactory();
        // 创建用于解析上传数据的工具类 ServletFileUpload 类
        ServletFileUpload servletFileUpload = new ServletFileUpload(fileItemFactory);
        try {
            // 解析上传的数据，得到每一个表单项 FileItem
            List<FileItem> list = servletFileUpload.parseRequest(req);
            // 循环判断，每一个表单项，是普通类型，还是上传的文件

```

```
for (FileItem fileItem : list) {

    if (fileItem.isFormField()) {
        // 普通表单项

        System.out.println("表单项的 name 属性值: " + fileItem.getFieldName());
        // 参数 UTF-8. 解决乱码问题
        System.out.println("表单项的 value 属性值: " + fileItem.getString("UTF-8"));

    } else {
        // 上传的文件
        System.out.println("表单项的 name 属性值: " + fileItem.getFieldName());
        System.out.println("上传的文件名: " + fileItem.getName());

        fileItem.write(new File("e:\\\" + fileItem.getName()));
    }
}
} catch (Exception e) {
    e.printStackTrace();
}
}
```

## 3、文件下载

下载的常用 API 说明:

```
response.getOutputStream();
servletContext.getResourceAsStream();
servletContext.getMimeType();
response.setContentType();
```

```
response.setHeader("Content-Disposition", "attachment; fileName=1.jpg");
```

这个响应头告诉浏览器。这是需要下载的。而 `attachment` 表示附件，也就是下载的一个文件。`fileName=`后面，表示下载的文件名。

完成上面的两个步骤，下载文件是没问题了。但是如果我们要下载的文件是中文名的话。你会发现，下载无法正确显示出正确的中文名。

原因是在响应头中，不能包含有中文字符，只能包含 ASCII 码。

文件下载示例：

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//    1、获取要下载的文件名
String downloadFileName = "2.jpg";
//    2、读取要下载的文件内容（通过ServletContext 对象可以读取）
ServletContext servletContext = getServletContext();
//    获取要下载的文件类型
String mimeType = servletContext.getMimeType("/file/" + downloadFileName);
System.out.println("下载的文件类型：" + mimeType);
//    4、在回传前，通过响应头告诉客户端返回的数据类型
resp.setContentType(mimeType);
//    5、还要告诉客户端收到的数据是用于下载使用（还是使用响应头）
//    Content-Disposition 响应头，表示收到的数据怎么处理
//    attachment 表示附件，表示下载使用
//    filename= 表示指定下载的文件名
resp.setHeader("Content-Disposition", "attachment; filename=" + downloadFileName);
/**
 * /斜杠被服务器解析表示地址为 http://ip:port/工程名/ 映射 到代码的Web 目录
 */
InputStream resourceAsStream = servletContext.getResourceAsStream("/file/" +
downloadFileName);
//    获取响应的输出流
OutputStream outputStream = resp.getOutputStream();
//    3、把下载的文件内容回传给客户端
//    读取输入流中全部的数据，复制给输出流，输出给客户端
IOUtils.copy(resourceAsStream,outputStream);
}
```

## 附件中文名乱码问题解决方案：

### 方案一：URLEncoder 解决 IE 和谷歌浏览器的 附件中 中文名问题。

如果客户端浏览器是 IE 浏览器 或者 是谷歌浏览器。我们需要使用 URLEncoder 类先对中文名进行 UTF-8 的编码操作。

因为 IE 浏览器和谷歌浏览器收到含有编码后的字符串后会以 UTF-8 字符集进行解码显示。

```
// 把中文名进行 UTF-8 编码操作。
```

```
String str = "attachment; fileName=" + URLEncoder.encode("中文.jpg", "UTF-8");
```

```
// 然后把编码后的字符串设置到响应头中  
response.setHeader("Content-Disposition", str);
```

## 方案二: BASE64 编解码 解决 火狐浏览器的附件中文名问题

如果客户端浏览器是火狐浏览器。 那么我们需要对中文名进行 BASE64 的编码操作。

这时候需要把请求头 **Content-Disposition: attachment; filename=中文名**  
编码成为: **Content-Disposition: attachment; filename=?charset?B?xxxxx?=?**

**?charset?B?xxxxx?=?** 现在我们对这段内容进行一下说明。

<b>=?</b>	表示编码内容的开始
<b>charset</b>	表示字符集
<b>B</b>	表示 BASE64 编码
<b>xxxx</b>	表示文件名 BASE64 编码后的内容
<b>?=?</b>	表示编码内容的结束

### BASE64 编解码操作:

```
public static void main(String[] args) throws Exception {  
    String content = "这是需要 Base64 编码的内容";  
    // 创建一个 Base64 编码器  
    BASE64Encoder base64Encoder = new BASE64Encoder();  
    // 执行 Base64 编码操作  
    String encodedString = base64Encoder.encode(content.getBytes("UTF-8"));  
  
    System.out.println( encodedString );  
    // 创建 Base64 解码器  
    BASE64Decoder base64Decoder = new BASE64Decoder();  
    // 解码操作  
    byte[] bytes = base64Decoder.decodeBuffer(encodedString);  
  
    String str = new String(bytes, "UTF-8");  
  
    System.out.println(str);  
}
```

因为火狐使用的是 BASE64 的编解码方式还原响应中的汉字。所以需要使用 BASE64Encoder 类进行编码操作。

```
// 使用下面的格式进行 BASE64 编码后
String str = "attachment; fileName=" + "?utf-8?B?"
           + new BASE64Encoder().encode("中文.jpg".getBytes("utf-8")) + "?=";
// 设置到响应头中
response.setHeader("Content-Disposition", str);
```

那么我们如何解决上面两种不同编解码方式呢。我们只需要通过判断请求头中 `User-Agent` 这个请求头携带过来的浏览器信息即可判断出是什么浏览器。

如下：

```
String ua = request.getHeader("User-Agent");
// 判断是否是火狐浏览器
if (ua.contains("Firefox")) {
    // 使用下面的格式进行 BASE64 编码后
    String str = "attachment; fileName=" + "?utf-8?B?"
               + new BASE64Encoder().encode("中文.jpg".getBytes("utf-8")) + "?=";
    // 设置到响应头中
    response.setHeader("Content-Disposition", str);
} else {
    // 把中文名进行 UTF-8 编码操作。
    String str = "attachment; fileName=" + URLEncoder.encode("中文.jpg", "UTF-8");
    // 然后把编码后的字符串设置到响应头中
    response.setHeader("Content-Disposition", str);
}
```