文件的上传与下载

文件的上传介绍

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

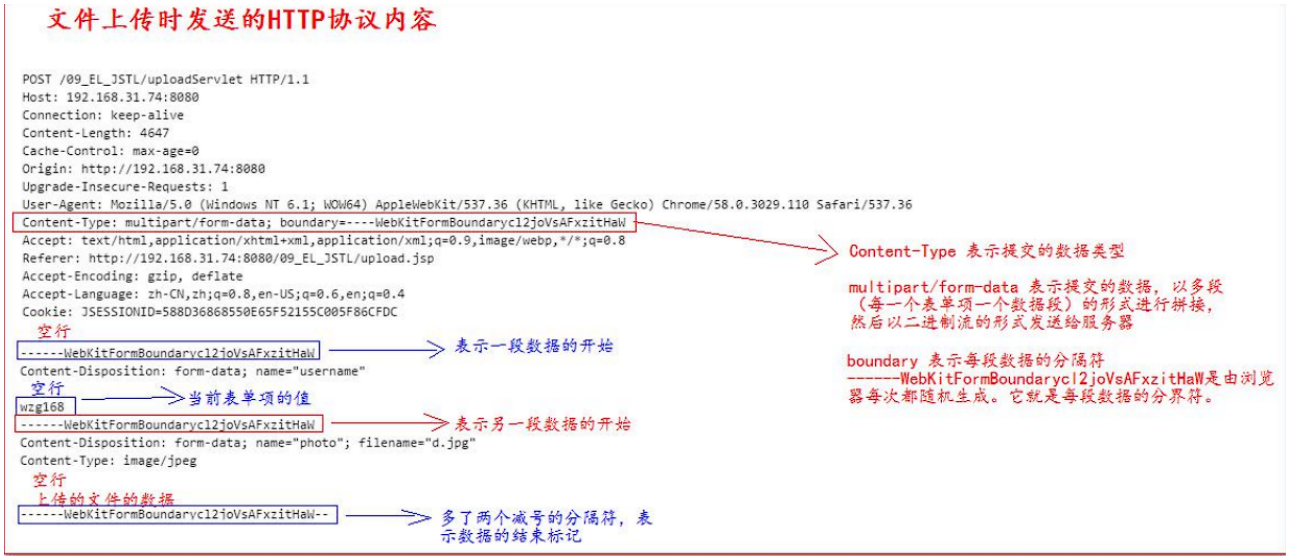
4、编写服务器代码（Servlet 程序）接收，处理上传的数据。

```
public class UploadServlet extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
        System.out.println("Yes!");
    }
}
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="http://192.168.233.1:8080/EJ/us" 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>
</body>
</html>
```

```xml
    <servlet>
        <servlet-name>UploadServlet</servlet-name>
        <servlet-class>com.UploadServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>UploadServlet</servlet-name>
        <url-pattern>/us</url-pattern>
    </servlet-mapping>
```

上传的HTTP协议内容介绍



上传和用到的类和方法的介绍

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

- 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 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.getFieldName()

- 获取表单项的 name 属性

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

使用fileupload解析上传的数据

Servlet服务器端的实现代码

```
1  package com;
2
3  import org.apache.commons.fileupload.FileItem;
4  import org.apache.commons.fileupload.FileItemFactory;
5  import org.apache.commons.fileupload.FileUploadException;
6  import org.apache.commons.fileupload.disk.DiskFileItemFactory;
7  import org.apache.commons.fileupload.servlet.ServletFileUpload;
8
9  import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
```

```
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13 import java.io.File;
14 import java.io.IOException;
15 import java.util.List;
16
17 /**
18  * JunXing
19  * 2023/3/22 17:01
20  * IntelliJ IDEA
21  */
22 public class UploadServlet extends HttpServlet{
23
24     @Override
25     protected void doPost(HttpServletRequest req,
   HttpServletResponse resp) throws ServletException, IOException {
26         //判断上传的数据是否多段数据（只有是多端数据，才是文件上传）
27         if (ServletFileUpload.isMultipartContent(req)) {
28             //创建FileItemFactory工厂实现类
29             FileItemFactory fileItemFactory = new
   DiskFileItemFactory();
30             //创建用于解析上传数据的工具类ServletFileUpload类
31             ServletFileUpload servletFileUpload = new
   ServletFileUpload(fileItemFactory);
32             //解析上传的数据，得到每一个表单项FileItem
33             try {
34                 List<FileItem> list =
   servletFileUpload.parseRequest(req);
35                 //循环判断 每一个表单项 是普通类型 还是 上传文件
36                 for (FileItem fileItem : list) {
37                     if (fileItem.isFormField()) {
38                         //普通表单项
39                         System.out.println("表单项的name属性值" +
   fileItem.getFieldName());
40                         //参数UTF-8解决乱码问题
41                         System.out.println("表单项的value下、属性
   值" + fileItem.getString("UTF-8"));
42                     } else {
43                         //上传的为文件
```

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

文件下载的实现

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

```
package com;
```

```
2
3  import org.apache.commons.io.IOUtils;
4
5  import javax.servlet.ServletContext;
6  import javax.servlet.ServletException;
7  import javax.servlet.ServletOutputStream;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import java.io.IOException;
12 import java.io.InputStream;
13
14 /**
15  * JunXing
16  * 2023/3/23 11:33
17  * IntelliJ IDEA
18  */
19 public class Download extends HttpServlet {
20     @Override
21     protected void doGet(HttpServletRequest req,
   HttpServletResponse resp) throws ServletException, IOException {
22         //1.获取要下载的文件名
23         String downloadFileName = "2.jpg";
24         //2.读取要下载的文件内容（通过ServletContext对象可以读取）
25         ServletContext servletContext = getServletContext();
26         //获取要下载的文件类型
27         String mimeType = servletContext.getMimeType("/file" +
   downloadFileName);
28         //4.在回传前，通过响应头告诉客户端返回的数据类型
29         resp.setContentType(mimeType);
30
31         /**
32         //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）
33         //Content-Disposition 响应头，表示收到的数据怎么处理
34         //attachment 表示附件，表示下载使用
35         //filename 表示指定下载的文件名
36          */
37
```

```
38          resp.setHeader("Content-Disposition", "attachment;
    filename" + downloadFileName);
39
40          //斜杠被服务器解析表示地址为 http://ip:prot/工程名/ 映射到代码的
    web目录
41
42          InputStream resourceAsStream =
    servletContext.getResourceAsStream("/file/" + downloadFileName);
43          //获取响应的输出流
44          ServletOutputStream outputStream =
    resp.getOutputStream();
45          //3.把下载的文件内容回传给客户端
46          //读取输入流中全部的数据，复制给输出流，输出给客户端
47          IOUtils.copy(resourceAsStream, outputStream);
48      }
49  }
```

中文名乱码问题解决方案

方案一：URLEncoder 解决Edge和谷歌浏览器的附件中文名问题

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

```
1  // 把中文名进行 UTF-8 编码操作。
2  String str = "attachment; fileName=" + URLEncoder.encode("中
    文.jpg", "UTF-8");
3  // 然后把编码后的字符串设置到响应头中
4  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。

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

```
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

```
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)
}
```

返回文首