# 文件上传和下载

## 课前默写

1、写出JSP的脚本语法
2、写出JSP的三大指令的格式和作用
3、写出JSP的动态包含和静态包含的区别
4、写出JSP的常用内置对象
5、写出EL表达式的格式和作用
6、写出JSTL的常用标签和作用

## 课程回顾

1、JSP的脚本语法
2、JSP的指令
3、JSP的内置对象
4、JSP的标签
5、EL表达式
6、JSTL标签库

## 今日内容

1、什么是过滤器文件上传
2、文件下载

## 教学目标

1、掌握文件上传
2、掌握文件下载

> 主要教学内容:利用file-upload工具包进行文件上传的处理!

## 第一章 文件上传

### 1.1 文件上传三要素

- 提供form表单,method必须是post!
- form表单的enctype必须是multipart/form-data
- 提供 input type="file" 类型长传输入

### 1.2 实现文件上传

#### 1.2.1 项目准备

导入: file-upload的jar包

#### 1.2.2 编写上传页面

```
<%@ page language="java" pageEncoding="UTF-8"%>
```

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>文件上传</title>
  </head>

  <body>
    <form action="${pageContext.request.contextPath}/servlet/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>
  </body>
</html>
```

#### 1.2.3 编写处理代码

```
package me.gacl.web.controller;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

public class UploadHandleServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //得到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，保证上传
文件的安全
        String savePath = this.getServletContext().getRealPath("/WEB-INF/upload");
        File file = new File(savePath);
        //判断上传文件的保存目录是否存在
        if (!file.exists() && !file.isDirectory()) {
            System.out.println(savePath+"目录不存在，需要创建");
            //创建目录
            file.mkdir();
        }
        //消息提示
        String message = "";
        try{
            //使用Apache文件上传组件处理文件上传步骤:
            //1、创建一个DiskFileItemFactory工厂
```

```
                    DiskFileItemFactory factory = new DiskFileItemFactory();
                    //2、创建一个文件上传解析器
                    ServletFileUpload upload = new ServletFileUpload(factory);
                     //解决上传文件名的中文乱码
                    upload.setHeaderEncoding("UTF-8");
                    //3、判断提交上来的数据是否是上传表单的数据
                    if(!ServletFileUpload.isMultipartContent(request)){
                        //按照传统方式获取数据
                        return;
                    }
                    //4、使用ServletFileUpload解析器解析上传数据，解析结果返回的是一个List<FileItem>集合，每一个FileItem对应一个Form表单的输入项
                    List<FileItem> list = upload.parseRequest(request);
                    for(FileItem item : list){
                        //如果fileitem中封装的是普通输入项的数据
                        if(item.isFormField()){
                            String name = item.getFieldName();
                            //解决普通输入项的数据的中文乱码问题
                            String value = item.getString("UTF-8");
                            //value = new String(value.getBytes("iso8859-1"),"UTF-8");
                            System.out.println(name + "=" + value);
                        }else{//如果fileitem中封装的是上传文件
                            //得到上传的文件名称，
                            String filename = item.getName();
                            System.out.println(filename);
                            if(filename==null || filename.trim().equals("")){
                                continue;
                            }
                            //注意：不同的浏览器提交的文件名是不一样的，有些浏览器提交上来的文件名是带有路径的，如：  c:\a\b\1.txt，而有些只是单纯的文件名，如：1.txt
                            //处理获取到的上传文件的文件名的路径部分，只保留文件名部分
                            filename = filename.substring(filename.lastIndexOf("\\")+1);
                            //获取item中的上传文件的输入流
                            InputStream in = item.getInputStream();
                            //创建一个文件输出流
                            FileOutputStream out = new FileOutputStream(savePath + "\\" + filename);
                            //创建一个缓冲区
                            byte buffer[] = new byte[1024];
                            //判断输入流中的数据是否已经读完的标识
                            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
                                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 (Exception e) {
                    message= "文件上传失败！";
                    e.printStackTrace();

                }
                request.setAttribute("message",message);
                request.getRequestDispatcher("/message.jsp").forward(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        doGet(request, response);
    }
}
```

#### 1.2.4 配置

```
<servlet>
    <servlet-name>UploadHandleServlet</servlet-name>
    <servlet-class>me.gacl.web.controller.UploadHandleServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>UploadHandleServlet</servlet-name>
    <url-pattern>/servlet/UploadHandleServlet</url-pattern>
</servlet-mapping>
```

### 1.3 文件上传细节注意

上述的代码虽然可以成功将文件上传到服务器上面的指定目录当中，但是文件上传功能有许多需要注意的小细节问题，以下列出的几点需要特别注意的

**1、为保证服务器安全，上传文件应该放在外界无法直接访问的目录下，比如放于WEB-INF目录下。**

**2、为防止文件覆盖的现象发生，要为上传文件产生一个唯一的文件名。**

**3、为防止一个目录下面出现太多文件，要使用hash算法打散存储。**

**4、要限制上传文件的最大值。**

**5、要限制上传文件的类型，在收到上传文件名时，判断后缀名是否合法。**

最终处理代码改进为:

```
public class UploadHandleServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
                //得到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，保证上传文件的安全

                String savePath = this.getServletContext().getRealPath("/WEB-INF/upload");
```

```
                //上传时生成的临时文件保存目录
                String tempPath = this.getServletContext().getRealPath("/WEB-INF/temp");
                File tmpFile = new File(tempPath);
                if (!tmpFile.exists()) {
                    //创建临时目录
                    tmpFile.mkdir();
                }
                //消息提示
                String message = "";
                try{
                    //使用Apache文件上传组件处理文件上传步骤:
                    //1、创建一个DiskFileItemFactory工厂
                    DiskFileItemFactory factory = new DiskFileItemFactory();
                    //设置工厂的缓冲区的大小，当上传的文件大小超过缓冲区的大小时，就会生成一个临时文件存放到指定的临时目录当中。
                    factory.setSizeThreshold(1024*100);//设置缓冲区的大小为100KB，如果不指定，那么缓冲区的大小默认是10KB
                    //设置上传时生成的临时文件的保存目录
                    factory.setRepository(tmpFile);
                    //2、创建一个文件上传解析器
                    ServletFileUpload upload = new ServletFileUpload(factory);
                    //监听文件上传进度
                    upload.setProgressListener(new ProgressListener(){
                        public void update(long pBytesRead, long pContentLength, int arg2) {
                            System.out.println("文件大小为：" + pContentLength + ",当前已处理：" + pBytesRead);
                            /**
                             * 文件大小为：14608,当前已处理：4096
                               文件大小为：14608,当前已处理：7367
                               文件大小为：14608,当前已处理：11419
                               文件大小为：14608,当前已处理：14608
                             */
                        }
                    });
                     //解决上传文件名的中文乱码
                    upload.setHeaderEncoding("UTF-8");
                    //3、判断提交上来的数据是否是上传表单的数据
                    if(!ServletFileUpload.isMultipartContent(request)){
                        //按照传统方式获取数据
                        return;
                    }
                    //设置上传单个文件的大小的最大值，目前是设置为1024*1024字节，也就是1MB
                    upload.setFileSizeMax(1024*1024);
                    //设置上传文件总量的最大值，最大值=同时上传的多个文件的大小的最大值的和，目前设置为10MB
                    upload.setSizeMax(1024*1024*10);
                    //4、使用ServletFileUpload解析器解析上传数据，解析结果返回的是一个List<FileItem>集合，每一个FileItem对应一个Form表单的输入项
                    List<FileItem> list = upload.parseRequest(request);
                    for(FileItem item : list){
                        //如果fileitem中封装的是普通输入项的数据
                        if(item.isFormField()){

                            String name = item.getFieldName();
```

```
                            //解决普通输入项的数据的中文乱码问题
                            String value = item.getString("UTF-8");
                            //value = new String(value.getBytes("iso8859-1"),"UTF-8");
                            System.out.println(name + "=" + value);
                        }else{//如果fileitem中封装的是上传文件
                            //得到上传的文件名称,
                            String filename = item.getName();
                            System.out.println(filename);
                            if(filename==null || filename.trim().equals("")){
                                continue;
                            }
                            //注意：不同的浏览器提交的文件名是不一样的，有些浏览器提交上来的文件名是带有路径的，如：  c:\a\b\1.txt，而有些只是单纯的文件名，如：1.txt
                            //处理获取到的上传文件的文件名的路径部分，只保留文件名部分
                            filename = filename.substring(filename.lastIndexOf("\\")+1);
                            //得到上传文件的扩展名
                            String fileExtName =
filename.substring(filename.lastIndexOf(".")+1);
                            //如果需要限制上传的文件类型，那么可以通过文件的扩展名来判断上传的文件类型是否合法
                            System.out.println("上传的文件的扩展名是："+fileExtName);
                            //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
                            String realSavePath = makePath(saveFilename, savePath);
                            //创建一个文件输出流
                            FileOutputStream out = new FileOutputStream(realSavePath + "\\" +
saveFilename);
                            //创建一个缓冲区
                            byte buffer[] = new byte[1024];
                            //判断输入流中的数据是否已经读完的标识
                            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
                                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();

                    request.setAttribute("message", "单个文件超出最大值！！！");
```

```
                request.getRequestDispatcher("/message.jsp").forward(request, response);
                return;
            }catch (FileUploadBase.SizeLimitExceededException e) {
                e.printStackTrace();
                request.setAttribute("message", "上传文件的总的大小超出限制的最大值！！！");
                request.getRequestDispatcher("/message.jsp").forward(request, response);
                return;
            }catch (Exception e) {
                message= "文件上传失败！";
                e.printStackTrace();
            }
            request.setAttribute("message",message);
            request.getRequestDispatcher("/message.jsp").forward(request, response);
    }

    /**
    * @Method: makeFileName
    * @Description: 生成上传文件的文件名，文件名以：uuid+"_"+文件的原始名称
    * @Anthor:孤傲苍狼
    * @param filename 文件的原始名称
    * @return uuid+"_"+文件的原始名称
    */
    private String makeFileName(String filename){  //2.jpg
        //为防止文件覆盖的现象发生，要为上传文件产生一个唯一的文件名
        return UUID.randomUUID().toString() + "_" + filename;
    }

    /**
     * 为防止一个目录下面出现太多文件，要使用hash算法打散存储
    * @param filename 文件名，要根据文件名生成存储目录
    * @param savePath 文件存储路径
    * @return 新的存储目录
    */
    private String makePath(String filename,String savePath){
        //得到文件名的hashCode的值，得到的就是filename这个字符串对象在内存中的地址
        int hashcode = filename.hashCode();
        int dir1 = hashcode&0xf;  //0--15
        int dir2 = (hashcode&0xf0)>>4;  //0-15
        //构造新的保存目录
        String dir = savePath + "\\" + dir1 + "\\" + dir2;  //upload\2\3  upload\3\5
        //File既可以代表文件也可以代表目录
        File file = new File(dir);
        //如果目录不存在
        if(!file.exists()){
            //创建目录
            file.mkdirs();
        }
        return dir;
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);

    }
```

```
}
```

## 第二章 文件下载

> 我们要将Web应用系统中的文件资源提供给用户进行下载，首先我们要有一个页面列出上传文件目录下的所有文件，当用户点击文件下载超链接时就进行下载操作，编写一个ListFileServlet，用于列出Web应用系统中所有下载文件

### 2.1 获取文件列表

```
package me.gacl.web.controller;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ListFileServlet extends HttpServlet {
    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");
        //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);
        }
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

这里简单说一下ListFileServlet中listfile方法，listfile方法是用来列出目录下的所有文件的，listfile方法内部用到了递归，在实际开发当中，我们肯定会在数据库创建一张表，里面会存储上传的文件名以及文件的具体存放目录，我们通过查询表就可以知道文件的具体存放目录，是不需要用到递归操作的，这个例子是因为没有使用数据库存储上传的文件名和文件的具体存放位置，而上传文件的存放位置又使用了散列算法打散存放，所以需要用到递归，在递归时，将获取到的文件名存放到从外面传递到listfile方法里面的Map集合当中，这样就可以保证所有的文件都存放在同一个Map集合当中。

### 2.2 配置

在Web.xml文件中配置ListFileServlet

```
<servlet>
     <servlet-name>ListFileServlet</servlet-name>
     <servlet-class>me.gacl.web.controller.ListFileServlet</servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>ListFileServlet</servlet-name>
    <url-pattern>/servlet/ListFileServlet</url-pattern>
</servlet-mapping>
```

### 2.3 下载页面

展示下载文件的listfile.jsp页面如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE HTML>
<html>
  <head>
    <title>下载文件显示页面</title>
  </head>
  <body>
     <!-- 遍历Map集合 -->
    <c:forEach var="me" items="${fileNameMap}">
        <c:url value="/servlet/DownLoadServlet" var="downurl">
            <c:param name="filename" value="${me.key}"></c:param>
        </c:url>
        ${me.value}<a href="${downurl}">下载</a>
        <br/>
    </c:forEach>

  </body>
```

```
</html>
```

### 2.4 实现文件下载

编写一个用于处理文件下载的Servlet，DownLoadServlet的代码如下：

```
public class DownLoadServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //得到要下载的文件名
        String fileName = request.getParameter("filename");  //23239283-92489-阿凡达.avi
        fileName = new String(fileName.getBytes("iso8859-1"),"UTF-8");
        //上传的文件都是保存在/WEB-INF/upload目录下的子目录当中
        String fileSaveRootPath=this.getServletContext().getRealPath("/WEB-INF/upload");
        //通过文件名找出文件的所在目录
        String path = findFileSavePathByFileName(fileName,fileSaveRootPath);
        //得到要下载的文件
        File file = new File(path + "\\" + fileName);
        //如果文件不存在
        if(!file.exists()){
            request.setAttribute("message", "您要下载的资源已被删除！！");
            request.getRequestDispatcher("/message.jsp").forward(request, response);
            return;
        }
        //处理文件名
        String realname = fileName.substring(fileName.indexOf("_")+1);
        //设置响应头，控制浏览器下载该文件
        response.setHeader("content-disposition", "attachment;filename=" +
URLEncoder.encode(realname, "UTF-8"));
        //读取要下载的文件，保存到文件输入流
        FileInputStream in = new FileInputStream(path + "\\" + fileName);
        //创建输出流
        OutputStream out = response.getOutputStream();
        //创建缓冲区
        byte buffer[] = new byte[1024];
        int len = 0;
        //循环将输入流中的内容读取到缓冲区当中
        while((len=in.read(buffer))>0){
            //输出缓冲区的内容到浏览器，实现文件下载
            out.write(buffer, 0, len);
        }
        //关闭文件输入流
        in.close();
        //关闭输出流
        out.close();
    }

    public String findFileSavePathByFileName(String filename,String saveRootPath){
        int hashcode = filename.hashCode();
        int dir1 = hashcode&0xf;  //0--15
        int dir2 = (hashcode&0xf0)>>4;  //0-15
        String dir = saveRootPath + "\\" + dir1 + "\\" + dir2;  //upload\2\3  upload\3\5

        File file = new File(dir);
```

```
        if(!file.exists()){
            //创建目录
            file.mkdirs();
        }
        return dir;
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

### 2.5 配置

```
<servlet>
      <servlet-name>DownLoadServlet</servlet-name>
      <servlet-class>me.gacl.web.controller.DownLoadServlet</servlet-class>
</servlet>

<servlet-mapping>
      <servlet-name>DownLoadServlet</servlet-name>
      <url-pattern>/servlet/DownLoadServlet</url-pattern>
</servlet-mapping>
```

## 作业题

1、完善图书图片的上传和下载功能

## 面试题

1、文件上传的页面表单必须要设置那些属性