1. 文件上传下载依赖

依赖于common-fileupload.jar, 而此包又依赖于common-io.jar

1. 文件上传代码之前台

<form action="text.action" 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>

1. 文件上传代码之后台

public class UploadHandleServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){

String savePath = this.getServletContext().getRealPath("/WEB-INF/upload"); //上传文件目录

String tempPath = this.getServletContext().getRealPath("/WEB-INF/temp"); //临时文件目录

File tmpFile = new File(tempPath);

if (!tmpFile.exists()) {

tmpFile.mkdir(); //创建临时目录

}

String message = ""; //消息提示

try {

DiskFileItemFactory factory = new DiskFileItemFactory(); //使用Apache文件上传组件处理文件上传步骤：

//设置缓冲区的大小，当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，默认是10KB

factory.setRepository(tmpFile); //设置上传时生成的临时文件的保存目录

ServletFileUpload upload = new ServletFileUpload(factory); //创建一个文件上传解析器

upload.setProgressListener(new ProgressListener() { //监听文件上传进度

public void update(long pBytesRead, long pContentLength, int arg2) {

System.out.println("文件大小为：" + pContentLength + ",当前已处理：" + pBytesRead);

}

});

upload.setHeaderEncoding("UTF-8"); //解决上传文件名的中文乱码

if (!ServletFileUpload.isMultipartContent(request)) { //判断提交上来的数据是否是上传表单的数据

return; //按照传统方式获取数据

}

upload.setFileSizeMax(1024 \* 1024); //设置上传单个文件的大小的最大值

upload.setSizeMax(1024 \* 1024 \* 10); //设置上传多个文件的大小和

//使用ServletFileUpload解析器解析上传数据，返回List<FileItem>集合，每元素对应一个Form表单的输入项

List<FileItem> list = upload.parseRequest(request);

for (FileItem item : list) {

if (item.isFormField()) { //如果fileitem中封装的是普通输入项的数据

String name = item.getFieldName();

String value = item.getString("UTF-8"); //解决普通输入项的数据的中文乱码问题

System.out.println(name + "=" + value);

} else { //如果fileitem中封装的是上传文件

String filename = item.getName(); //得到上传的文件名称

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

filename = filename.substring(filename.lastIndexOf("\\") + 1);

String fileExtName = filename.substring(filename.lastIndexOf(".") + 1); //得到上传文件的扩展名

//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);

InputStream in = item.getInputStream(); //获取item中的上传文件的输入流

String saveFilename = makeFileName(filename); //得到文件保存的名称

String realSavePath = makePath(saveFilename, savePath); //得到文件的保存目录

FileOutputStream out = new FileOutputStream(realSavePath + "\\" + saveFilename); //创建文件输出流

byte buffer[] = new byte[1024]; //创建一个缓冲区

int len = 0;

while ((len = in.read(buffer)) > 0) {

out.write(buffer, 0, len);

}

in.close();

out.close();

//item.delete(); //删除上传时生成的临时文件

message = "文件上传成功！";

}

}

} catch (FileUploadBase.FileSizeLimitExceededException e) {

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

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

return;

} catch (FileUploadBase.SizeLimitExceededException e) {

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！";

}

request.setAttribute("message", message);

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

}

//生成上传文件的文件名，文件名以：uuid+"\_"+文件的原始名称

private String makeFileName(String filename) { //2.jpg

return UUID.randomUUID().toString() + "\_" + filename; //产生唯一文件名

}

//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

private String makePath(String filename, String savePath) {

int hashcode = filename.hashCode();

int dir1 = hashcode & 0xf; //0--15

int dir2 = (hashcode & 0xf0) >> 4; //0-15

String dir = savePath + "\\" + dir1 + "\\" + dir2; //构造新的保存目录

File file = new File(dir); //File既可以代表文件也可以代表目录

if (!file.exists()) {

file.mkdirs();

}

return dir;

}

}

1. 文件上传之细节
   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
2. 文件下载之前台

上传文件名

1. 文件下载之后台

public class DownLoadServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){

String fileName = request.getParameter("filename"); //得到要下载的文件名

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");

String path = findFileSavePathByFileName(fileName, fileSaveRootPath); //通过文件名找出文件的所在目录

File file = new File(path + "\\" + fileName); //得到要下载的文件

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;

}

String realname = fileName.substring(fileName.indexOf("\_") + 1); //处理文件名

//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"));

FileInputStream in = new FileInputStream(path + "\\" + fileName); //读取要下载的文件，保存到文件输入流

OutputStream out = response.getOutputStream(); //创建输出流

byte buffer[] = new byte[1024];

int len = 0;

while ((len = in.read(buffer)) > 0) {

out.write(buffer, 0, len);

}

in.close();

out.close();

}

//通过文件名和存储上传文件根目录找出要下载的文件的所在路径

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;

File file = new File(dir);

if (!file.exists()) {

file.mkdirs();

}

return dir;

}

}