文件上传在web应用中非常普遍，要在jsp环境中实现文件上传功能是非常容易的，因为网上有许多用java开发的文件上传组件，本文以commons-fileupload组件为例，为jsp应用添加文件上传功能。  
common-fileupload组件是apache的一个开源项目之一，可以从**http://jakarta.apache.org/commons/fileupload/** (http://jakarta.apache.org/commons/fileupload/)下载。   
用该组件可实现一次上传一个或多个文件，并可限制文件大小。

**代码**

下载后解压zip包，将commons-fileupload-1.0.jar复制到tomcat的webapps你的webappWEB-INFlib下，目录不存在请自建目录。  
新建一个servlet: Upload.java用于文件上传：  
import java.io.\*;  
import java.util.\*;  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
import org.apache.commons.fileupload.\*;  
  
public class Upload extends HttpServlet {  
  
private String uploadPath = "C:upload"; // 上传文件的目录  
private String tempPath = "C:uploadtmp"; // 临时文件目录  
  
public void doPost(HttpServletRequest request,  
HttpServletResponse response)  
throws IOException, ServletException  
{  
}  
}  
在doPost()方法中，当servlet收到浏览器发出的Post请求后，实现文件上传。以下是示例代码：  
public void doPost(HttpServletRequest request,  
HttpServletResponse response)  
throws IOException, ServletException  
{  
try {  
DiskFileUpload fu = new DiskFileUpload();  
// 设置最大文件尺寸，这里是4MB  
fu.setSizeMax(4194304);  
// 设置缓冲区大小，这里是4kb  
fu.setSizeThreshold(4096);  
// 设置临时目录：  
fu.setRepositoryPath(tempPath);  
  
// 得到所有的文件：  
List fileItems = fu.parseRequest(request);  
Iterator i = fileItems.iterator();  
// 依次处理每一个文件：  
while(i.hasNext()) {  
FileItem fi = (FileItem)i.next();  
// 获得文件名，这个文件名包括路径：  
String fileName = fi.getName();  
// 在这里可以记录用户和文件信息  
// ...  
// 写入文件，暂定文件名为a.txt，可以从fileName中提取文件名：  
fi.write(new File(uploadPath + "a.txt"));  
}  
}  
catch(Exception e) {  
// 可以跳转出错页面  
}  
}  
如果要在配置文件中读取指定的上传文件夹，可以在init()方法中执行：  
public void init() throws ServletException {  
uploadPath = ....  
tempPath = ....  
// 文件夹不存在就自动创建：  
if(!new File(uploadPath).isDirectory())  
new File(uploadPath).mkdirs();  
if(!new File(tempPath).isDirectory())  
new File(tempPath).mkdirs();  
}  
编译该servlet，注意要指定classpath，确保包含commons-upload-1.0.jar和tomcatcommonlibservlet-api.jar。  
配置servlet，用记事本打开tomcatwebapps你的webappWEB-INFweb.xml，没有的话新建一个。  
典型配置如下：  
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE web-app  
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
"http://java.sun.com/dtd/web-app\_2\_3.dtd">  
  
<web-app>  
<servlet>  
<servlet-name>Upload</servlet-name>  
<servlet-class>Upload</servlet-class>  
</servlet>  
  
<servlet-mapping>  
<servlet-name>Upload</servlet-name>  
<url-pattern>/fileupload</url-pattern>  
</servlet-mapping>  
</web-app>  
配置好servlet后，启动tomcat，写一个简单的html测试：  
<form action="fileupload" method="post"  
enctype="multipart/form-data" name="form1">  
<input type="file" name="file">  
<input type="submit" name="Submit" value="upload">  
</form>  
注意action="fileupload"其中fileupload是配置servlet时指定的url-pattern。   
  
  
下面是某个大虾的代码：  
  
这个Upload比smartUpload好用多了.完全是我一个个byte调试出来的,不象smartUpload的bug具多.  
调用方法:  
Upload up = new Upload();  
up.init(request);  
/\*\*  
此处可以调用setSaveDir(String saveDir);设置保存路径  
调用setMaxFileSize(long size)设置上传文件的最大字节.  
调用setTagFileName(String)设置上传后文件的名字(只对第一个文件有效)  
\*/  
up. uploadFile();  
  
然后String[] names = up.getFileName();得到上传的文件名,文件绝对路径应该是  
保存的目录saveDir+"/"+names[i];  
可以通过up.getParameter("field");得到上传的文本或up.getParameterValues("filed")  
得到同名字段如多个checkBox的值.  
其它的自己试试.  
  
源码:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
package com.inmsg.beans;  
  
import java.io.\*;  
import java.util.\*;  
import javax.servlet.\*;  
import javax.servlet.http.\*;  
  
public class Upload {  
private String saveDir = "."; //要保存文件的路径  
private String contentType = ""; //文档类型  
private String charset = ""; //字符集  
private ArrayList tmpFileName = new ArrayList(); //临时存放文件名的数据结构  
private Hashtable parameter = new Hashtable(); //存放参数名和值的数据结构  
private ServletContext context; //程序上下文,用于初始化  
private HttpServletRequest request; //用于传入请求对象的实例  
private String boundary = ""; //内存数据的分隔符  
private int len = 0; //每次从内在中实际读到的字节长度  
private String queryString;  
private int count; //上载的文件总数  
private String[] fileName; //上载的文件名数组  
private long maxFileSize = 1024 \* 1024 \* 10; //最大文件上载字节;  
private String tagFileName = "";  
  
public final void init(HttpServletRequest request) throws ServletException {  
this.request = request;  
boundary = request.getContentType().substring(30); //得到内存中数据分界符  
queryString = request.getQueryString();  
}  
  
public String getParameter(String s) { //用于得到指定字段的参数值,重写request.getParameter(String s)  
if (parameter.isEmpty()) {  
return null;  
}  
return (String) parameter.get(s);  
}  
  
public String[] getParameterValues(String s) { //用于得到指定同名字段的参数数组,重写request.getParameterValues(String s)  
ArrayList al = new ArrayList();  
if (parameter.isEmpty()) {  
return null;  
}  
Enumeration e = parameter.keys();  
while (e.hasMoreElements()) {  
String key = (String) e.nextElement();  
if ( -1 != key.indexOf(s + "||||||||||") || key.equals(s)) {  
al.add(parameter.get(key));  
}  
}  
if (al.size() == 0) {  
return null;  
}  
String[] value = new String[al.size()];  
for (int i = 0; i < value.length; i++) {  
value[i] = (String) al.get(i);  
}  
return value;  
}  
  
public String getQueryString() {  
return queryString;  
}  
  
public int getCount() {  
return count;  
}  
  
public String[] getFileName() {  
return fileName;  
}  
  
public void setMaxFileSize(long size) {  
maxFileSize = size;  
}  
  
public void setTagFileName(String filename) {  
tagFileName = filename;  
}  
  
public void setSaveDir(String saveDir) { //设置上载文件要保存的路径  
this.saveDir = saveDir;  
File testdir = new File(saveDir); //为了保证目录存在,如果没有则新建该目录  
if (!testdir.exists()) {  
testdir.mkdirs();  
}  
}  
  
public void setCharset(String charset) { //设置字符集  
this.charset = charset;  
}  
  
public boolean uploadFile() throws ServletException, IOException { //用户调用的上载方法  
setCharset(request.getCharacterEncoding());  
return uploadFile(request.getInputStream());  
}  
  
private boolean uploadFile(ServletInputStream servletinputstream) throws //取得央存数据的主方法  
ServletException, IOException {  
String line = null;  
byte[] buffer = new byte[256];  
while ( (line = readLine(buffer, servletinputstream, charset)) != null) {  
if (line.startsWith("Content-Disposition: form-data;")) {  
int i = line.indexOf("filename=");  
if (i >= 0) { //如果一段分界符内的描述中有filename=,说明是文件的编码内容  
String fName = getFileName(line);  
if (fName.equals("")) {  
continue;  
}  
if (count == 0 && tagFileName.length() != 0) {  
String ext = fName.substring( (fName.lastIndexOf(".") + 1));  
fName = tagFileName + "." + ext;  
}  
tmpFileName.add(fName);  
count++;  
while ( (line = readLine(buffer, servletinputstream, charset)) != null) {  
if (line.length() <= 2) {  
break;  
}  
}  
File f = new File(saveDir, fName);  
FileOutputStream dos = new FileOutputStream(f);  
long size = 0l;  
while ( (line = readLine(buffer, servletinputstream, null)) != null) {  
if (line.indexOf(boundary) != -1) {  
break;  
}  
size += len;  
if (size > maxFileSize) {  
throw new IOException("文件超过" + maxFileSize + "字节!");  
}  
dos.write(buffer, 0, len);  
}  
dos.close();  
}  
else { //否则是字段编码的内容  
String key = getKey(line);  
String value = "";  
while ( (line = readLine(buffer, servletinputstream, charset)) != null) {  
if (line.length() <= 2) {  
break;  
}  
}  
while ( (line = readLine(buffer, servletinputstream, charset)) != null) {  
  
if (line.indexOf(boundary) != -1) {  
break;  
}  
value += line;  
}  
put(key, value.trim(), parameter);  
}  
}  
}  
if (queryString != null) {  
String[] each = split(queryString, "&");  
for (int k = 0; k < each.length; k++) {  
String[] nv = split(each[k], "=");  
if (nv.length == 2) {  
put(nv[0], nv[1], parameter);  
}  
}  
}  
fileName = new String[tmpFileName.size()];  
for (int k = 0; k < fileName.length; k++) {  
fileName[k] = (String) tmpFileName.get(k); //把ArrayList中临时文件名倒入数据中供用户调用  
}  
if (fileName.length == 0) {  
return false; //如果fileName数据为空说明没有上载任何文件  
}  
return true;  
}  
  
private void put(String key, String value, Hashtable ht) {  
if (!ht.containsKey(key)) {  
ht.put(key, value);  
}  
else { //如果已经有了同名的KEY,就要把当前的key更名,同时要注意不能构成和KEY同名  
try {  
Thread.currentThread().sleep(1); //为了不在同一ms中产生两个相同的key  
}  
catch (Exception e) {}  
key += "||||||||||" + System.currentTimeMillis();  
ht.put(key, value);  
}  
}  
  
/\*  
调用ServletInputstream.readLine(byte[] b,int offset,length)方法,该方法是从ServletInputstream流中读一行  
到指定的byte数组,为了保证能够容纳一行,该byte[]b不应该小于256,重写的readLine中,调用了一个成员变量len为  
实际读到的字节数(有的行不满256),则在文件内容写入时应该从byte数组中写入这个len长度的字节而不是整个byte[]  
的长度,但重写的这个方法返回的是String以便分析实际内容,不能返回len,所以把len设为成员变量,在每次读操作时  
把实际长度赋给它.  
也就是说在处理到文件的内容时数据既要以String形式返回以便分析开始和结束标记,又要同时以byte[]的形式写到文件  
输出流中.  
\*/  
private String readLine(byte[] Linebyte,  
ServletInputStream servletinputstream, String charset) {  
try {  
len = servletinputstream.readLine(Linebyte, 0, Linebyte.length);  
if (len == -1) {  
return null;  
}  
if (charset == null) {  
return new String(Linebyte, 0, len);  
}  
else {  
return new String(Linebyte, 0, len, charset);  
}  
  
}  
catch (Exception \_ex) {  
return null;  
}  
  
}  
  
private String getFileName(String line) { //从描述字符串中分离出文件名  
if (line == null) {  
return "";  
}  
int i = line.indexOf("filename=");  
line = line.substring(i + 9).trim();  
i = line.lastIndexOf("");  
if (i < 0 || i >= line.length() - 1) {  
i = line.lastIndexOf("/");  
if (line.equals("""")) {  
return "";  
}  
if (i < 0 || i >= line.length() - 1) {  
return line;  
}  
}  
return line.substring(i + 1, line.length() - 1);  
}  
  
private String getKey(String line) { //从描述字符串中分离出字段名  
if (line == null) {  
return "";  
}  
int i = line.indexOf("name=");  
line = line.substring(i + 5).trim();  
return line.substring(1, line.length() - 1);  
}  
  
public static String[] split(String strOb, String mark) {  
if (strOb == null) {  
return null;  
}  
StringTokenizer st = new StringTokenizer(strOb, mark);  
ArrayList tmp = new ArrayList();  
while (st.hasMoreTokens()) {  
tmp.add(st.nextToken());  
}  
String[] strArr = new String[tmp.size()];  
for (int i = 0; i < tmp.size(); i++) {  
strArr[i] = (String) tmp.get(i);  
}  
return strArr;  
}  
}  
  
下载其实非常简单，只要如下处理，就不会发生问题。  
  
public void downLoad(String filePath,HttpServletResponse response,boolean isOnLine)  
throws Exception{  
File f = new File(filePath);  
if(!f.exists()){  
response.sendError(404,"File not found!");  
return;  
}  
BufferedInputStream br = new BufferedInputStream(new FileInputStream(f));  
byte[] buf = new byte[1024];  
int len = 0;  
  
response.reset(); //非常重要  
if(isOnLine){ //在线打开方式  
URL u = new URL("file:///"+filePath);  
response.setContentType(u.openConnection().getContentType());  
response.setHeader("Content-Disposition", "inline; filename="+f.getName());  
//文件名应该编码成UTF-8  
}  
else{ //纯下载方式  
response.setContentType("application/x-msdownload");   
response.setHeader("Content-Disposition", "attachment; filename=" + f.getName());   
}  
OutputStream out = response.getOutputStream();  
while((len = br.read(buf)) >0)  
out.write(buf,0,len);  
br.close();  
out.close();  
}