**（2）通过修改服务器端代码。前提是利用jar包。**

**commons-fileupload-1.2.2.jar和commons-io-1.4.jar**

**//创建工厂**

**DiskFileItemFactory factoy=new DiskFileItemFactory();**

**//factory工厂类, 可以设置上传文件的大小!**

**//创建解析器**

**ServletFileUpload sfu=new ServletFileUpload(factoy);**

**sfu.setSizeMax(int size) //设置文件总的大小, 也可以setFileSizeMax设置单个文件的大小**

**//解析request**

**try {**

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

**for (FileItem fileItem : list) {**

**if(fileItem.isFormField()){**

**//FileItem类型的对象里有一个方法isFormField(),**

**//可以判断表单项是表单域还是文件域**

**String name = fileItem.getFildName();**

**String value = fileItem.getString();**

**//表单域拿到键值对**

**}else{**

**//否则是文件域,就取得文件的信息和文件的流**

**String fieldname = fileItem.getFildName();**

**String filename = fileItem.getName();**

**String contentType = fileItem.getContentType();**

**....**

**}**

**}**

**} catch (FileUploadException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**DiskFileItemFactory类的常用方法：**

**构造方法**

**1) public DiskFileItemFactory()**

**2) public DiskFileItemFactory(int sizeThreshold,File repository)**

**3) void setSizeThreshold(int sizeThreshold)**

**Apache文件上传组件在解析上传数据中的每个字段内容时，需要临时保存解析出的数据，以便在后面进行数据的进一步处理（保存在磁盘特定位置或插入数据库）。因为Java虚拟机默认可以使用的内存空间是有限的，超出限制时将会抛出“java.lang.OutOfMemoryError”错误。如果上传的文件很大，例如800M的文件，在内存中将无法临时保存该文件内容，Apache文件上传组件转而采用临时文件来保存这些数据；但如果上传的文件很小，例如600个字节的文件，显然将其直接保存在内存中性能会更加好些。**

**setSizeThreshold方法用于设置是否将上传文件已临时文件的形式保存在磁盘的临界值（以字节为单位的int值），如果从没有调用该方法设置此临界值，将会采用系统默认值10KB。对应的getSizeThreshold() 方法用来获取此临界值。**

**4) void setRepository(File repository)**

**setRepositoryPath方法用于设置当上传文件尺寸大于setSizeThreshold方法设置的临界值时，将文件以临时文件形式保存在磁盘上的存放目录。有一个对应的获得临时文件夹的 File getRespository() 方法。**

**注意：当从没有调用此方法设置临时文件存储目录时，默认采用系统默认的临时文件路径，可以通过系统属性 java.io.tmpdir 获取。如下代码：**

**System.getProperty("java.io.tmpdir");**

**Tomcat系统默认临时目录为“<tomcat安装目录>/temp/”。**

**FileItem类的常用方法：**

**1. boolean isFormField()**

**isFormField方法用于判断FileItem类对象封装的数据是一个普通文本表单字段，还是一个文件表单字段，如果是普通表单字段则返回true，否则返回false。因此，可以使用该方法判断是否为普通表单域，还是文件上传表单域。**

**2. String getName()**

**getName方法用于获得文件上传字段中的文件名。**

**注意IE或FireFox中获取的文件名是不一样的，IE中是绝对路径，FireFox中只是文件名。**

**3. String getFieldName()**

**getFieldName方法用于返回表单标签name属性的值。如上例中<input type="text" name="column" />的value。**

**4. void write(File file)**

**write方法用于将FileItem对象中保存的主体内容保存到某个指定的文件中。如果FileItem对象中的主体内容是保存在某个临时文件中，该方法顺利完成后，临时文件有可能会被清除。该方法也可将普通表单字段内容写入到一个文件中，但它主要用途是将上传的文件内容保存在本地文件系统中。**

**5. String getString()**

**getString方法用于将FileItem对象中保存的数据流内容以一个字符串返回，它有两个重载的定义形式：**

**public java.lang.String getString()**

**public java.lang.String getString(java.lang.String encoding)**

**throws java.io.UnsupportedEncodingException**

**前者使用缺省的字符集编码将主体内容转换成字符串，后者使用参数指定的字符集编码将主体内容转换成字符串。如果在读取普通表单字段元素的内容时出现了中文乱码现象，请调用第二个getString方法，并为之传递正确的字符集编码名称。**

**6. String getContentType()**

**getContentType 方法用于获得上传文件的类型，即表单字段元素描述头属性“Content-Type”的值，如“image/jpeg”。如果FileItem类对象对应的是普通表单字段，该方法将返回null。**

**7. boolean isInMemory()**

**isInMemory方法用来判断FileItem对象封装的数据内容是存储在内存中，还是存储在临时文件中，如果存储在内存中则返回true，否则返回false。**

**8. void delete()**

**delete方法用来清空FileItem类对象中存放的主体内容，如果主体内容被保存在临时文件中，delete方法将删除该临时文件。**

**尽管当FileItem对象被垃圾收集器收集时会自动清除临时文件，但及时调用delete方法可以更早的清除临时文件，释放系统存储资源。另外，当系统出现异常时，仍有可能造成有的临时文件被永久保存在了硬盘中。**

**9. InputStream getInputStream()**

**以流的形式返回上传文件的数据内容。**

**10. long getSize()**

**返回该上传文件的大小（以字节为单位）。**

**写一个helloUploadFile简单例子来看看! 在资料中有java代码文件**

**上传文件前应该想到是注意项:**

**/\*\***

**\* 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。**

**\*/**

**基本思路:**

**①获取和创建保存文件的最终目录和临时目录**

**②创建一个DiskFileItemFactory工厂**

**③创建一个文件上传解析器**

**④判断提交上来的数据是否是上传表单的数据,是不是Multipart编码方式ServletFileUpload.isMultipartContent(request)**

**⑤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**

**⑥判断普通域还是文件域**

**⑦将普通表单域的键值对显示出来**

**⑧验证后缀的合法性**

**⑨将文件流写入保存的目录中(生成新的文件名,避免一个目录中文件太多而生成新的存储目录)**