# SmartUpload上传组件

## 认识SmartUpload组件

1. SmartUpload组件是由www.jspsmart.com网站（但该网站已经关闭）开发的一套文

件上传下载组件包，可以轻松的实现文件的上传与下载功能，并且可以实现上传文件类型的限制，还可以轻松地取得上传的文件的名称、后缀、大小等信息。

1. SmartUpload组件提供方已经关闭，但由于SmartUpload组件在非框架的开发中较为 好用，所以至今仍有人使用。

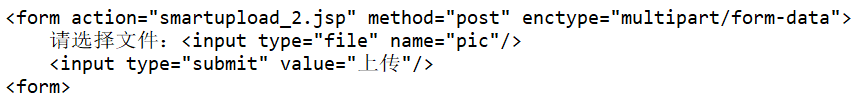
## 安装SmartUpload组件

SmartUpload组件本身是一个jar包，用户将此包放到classpath指定的目录即可，也可以直接放在tomcat安装目录的lib文件夹中。

## 上传单个文件

1）要想上传文件，必须使用HTML中提供的file控件，而且<form>标签必须设置enctype

属性将该表单封装，表示该表单不再是分别提交，而是将所有的内容都按照二进制的方式进行提交；且表单的提交方式必须只能是post，因为文件通常情况下都大于get提交方式中地址栏的参数长度。



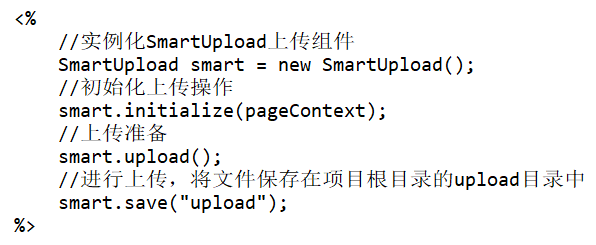
2）选择文件点击提交按钮进行提交后，在服务端的上传步骤如：

·声明SmartUpload对象

·初始化上传

·上传准备

·进行上传，并指定保存文件的路径



在实例化SmartUpload实例时记得通过page指令的import属性导入SmartUpload组件的包；

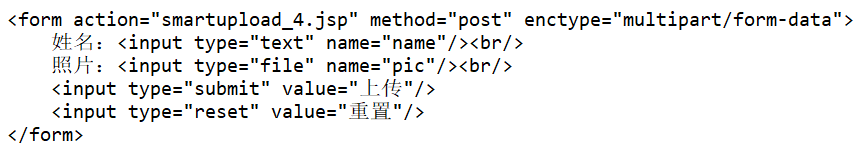
在进行上传时，必须保证保存上传文件的目录（已根目录为起点）已经存在，如不存在，可以手动创建或通过代码创建；

上传完成后，保存的文件名称与上传的文件名称是一样的，所以当上传了同样文件的名称时将会出现覆盖的情况。

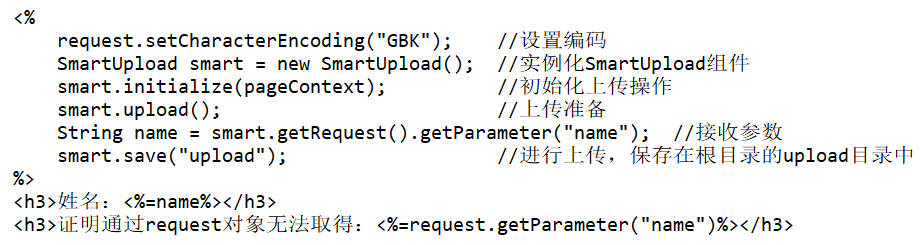
## 混合表单

在实际开发中，一个表单中往往同时包含了参数传递和文件上传的操作（即混合表单），由于文件上传必须使用enctype属性对表单进行封装，但当表单使用enctype属性进行封装后，其他的非文件类的表单控件传递的参数就无法通过request内置对象调用getParameter( )方法取得， 此时就必须通过SmartUpload类的getRequest( ).getParameter( )方法取得。并且，smart.getRequest( ).Parameter( )方法一定要在执行完upload( )方法后才可以使用，否则无法取得参数值。

**stmartupload\_3.jsp：**



**smartupload\_4.jsp：**



注意：使用SmartUpload对象调用getRequest( ).getParameter( )接收请求参数必须在调用upload( )方法之后才可以使用。

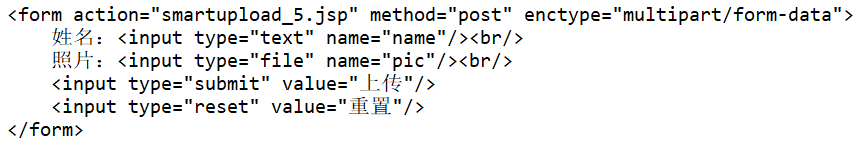
## 解决文件覆盖问题

在上传操作中，如果多个用户上传的文件的名称是一样的，则会发生覆盖的情况。为了解决这个问题，可以为上传的文件自动命名；为了防止重名，自动命名可以采用以下格式：

**IP地址 + 时间戳 + 三位随机数**

当然，采用何种格式命名，应根据实际开发要求，只要文件名称不重名即可，以防止文件覆盖情况。

**smartuupload\_3.jsp：上传文件页**



**smartupload\_5.jsp：进行文件上传的后端代码**



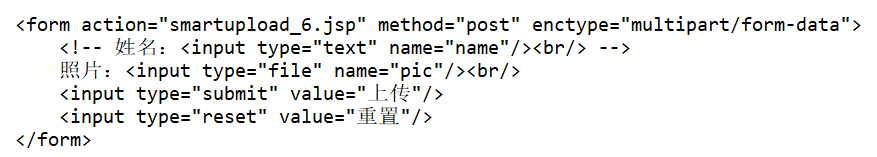
在代码中，SmartUpload实例调用getFiles( )方法可以取得上传的所有文件；调用getFile( )方法可以根据索引值取得上传的文件中指定索引位置的文件；文件调用getFileExt( )方法可以取得该文件的后缀名，如jps、txt等；文件调用saveAs( )方法表示手动保存文件。

## 限制上传文件

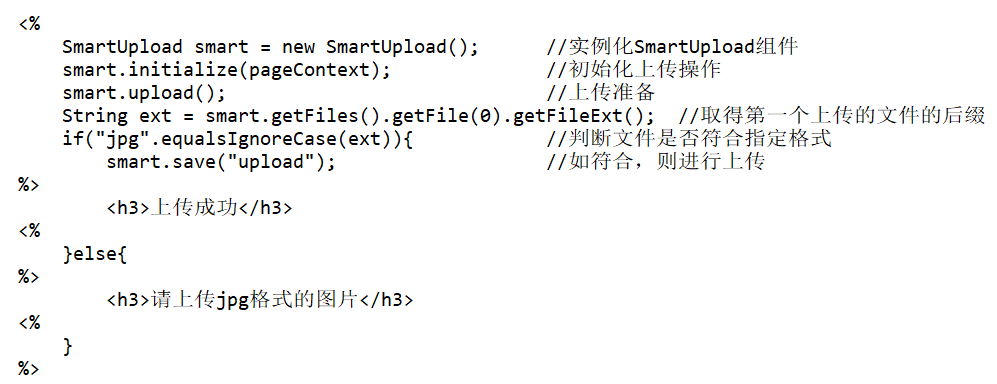
如果没有对上传的文件进行限制，如文件大小、文件类型等，那么用户可以上传任意的文件，但从实际开发角度讲，往往需要对上传的文件进行限制。

例如：采集用户头像时，需要图片格式的文件，此时就需要对用户上传的文件的类型进行限制，可通过文件的后缀名来判断。

**smartupload\_3.jsp：文件上传页面**



**smartupload\_6.jsp：负责接收文件并进行上传**



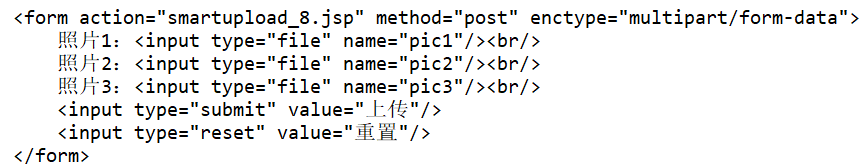
以上代码先取出上传文件的后缀，判断其后缀是否符合图片格式。

在实际开发中，除了显示上传文件的类型，往往还需要限制上传文件的大小。

## 上传多个文件

如果一次要上传多个文件，在处理上传的文件时可以通过循环迭代依次进行上传。通过SmartUpload实例调用getFiles( ).getCount( )可以取得上传的文件个数。

**smartupload\_7.jsp：上传文件页**



**smartupload\_8.jsp：接收上传的文件并进行上传**



SmartUpload组件使用简单，所以在普通的Web开发中使用SmartUpload完成文件的上传操作是一个明智的选择。

# FileUpload上传组件

## 认识FileUpload组件

FileUpload组件是Apache（www.apache.org）提供的免费上传组件，可以直接从Apache站点上下载，下载地址如下：

<http://commons.apache.org/fileupload/>

但是FileUpload组件需要依赖于Commons组件包，所以从Apache下载FileUpload组件时还需要连同Commons组件的IO包一起下载，下载地址如下：

<http://commons.apache.org/io/>

Commons组件包在很多的框架开发中都可以直接使用，而且此包中提供大量的开发类可作为Java的有力补充，值得研究。

从实际开发角度来讲，FileUpload组件使用起来会很麻烦，没有SmartUpload组件方

便。

## 安装FileUpload组件

下载后fileupload组件和commons组件后，将目录中的commons-fileupload.jar和commons-io.jar包配置到Tomcat安装目录下的lib目录中，表示配置到classpath路径中。

## 接收上传内容

只要是文件上传操作，都需要依靠HTML的file控件。FileUpload的具体上传操作与SmartUpload相比有着很高的复杂度。在使用FileUpload组件接收文件时，所有提交的内容都会通过upload.parseRequest( )方法返回，这其中就包括了传递的参数和上传的文件，所以需要依次判断每一次上传的内容是文件还是普通文本。下面是FileUpload接收上传内容的基本步骤，：

1. 创建磁盘工厂：DiskFileItemFactory factory = new DiskFileItemFactory( ) ;
2. 创建处理工具：ServletFileUpload upload = new ServletFileUpload( factory ) ;
3. 设置上传文件大小：upload.setFileSizeMax( 3145728 ) ;
4. 接收全部内容：List<FileItem> items = upload.parseRequest( request ) ;
5. 迭代接受的全部内容，判断是传递的参数还是上传的文件，如果是上传的文件，则进行上传操作，如果是传递的参数，则取出参数值。

具体示例请参考教材书。

## 保存上传内容

上节已经学习了如何接受上传的全部内容，但是上传的内容并没有保存在服务器上，而要想进行文件的保存，还需要通过java.io包中的InputStream和OutputStream两个类完成。大致步骤如下：将接收到的内容（上一节已经学习了如何接受上传的内容）进行迭代，判断每个内容是否是普通参数还是上传的文件，如果是上传的文件，则使用输出流进行上传保存操作；如果是普通参数，则取出参数值。（具体示例请参考教材书）

## 开发FileUpload组件工具

学习完FileUpload组件的基本组件后，会发现使用FileUpload组件存在以下几点不便之

处：

1. 无法像使用request.getParameter( )方法那样准确地取得提交的参数；
2. 无法像使用request.getParameterValues( )方法那样准确地取得一组提交参数；
3. 所有的上传的内容都需要进行依次的判断，才能够分别保存，不能一次性批量保存。

为此，如果想要解决以上问题，可以自己进行代码的扩充，编写一个FileUpload组件操作的工具类-----FileUploadToos。教材书上已经给了这个工具类的源代码，直接拿来使用即可，如果需要扩充，可以根据自己的需要进行扩充。

## 认识FileUpload组件

1. FileUpload组件是Apache组织提供的文件上传下载组件，可以直接从Apache站点 下载（http://commons.apache.org/fileupload/）；

2）但是FileUpload组件还需要依赖于Commons组件包，所以还需要下载Commons的 IO组件包（下载地址：http:commons.apache.org/io/）。

## 使用FileUpload上传文件

1. 不管是使用SmartUpload组件或是FileUpload组件进行上传操作，都必须依靠HTML 的file控件，且表单必须封装。
2. 使用FileUpload组件上传文件的基本步骤：

·创建磁盘工厂：DiskFileItemFactory factory =new DiskFileItemFactory( ) ;

·设置临时文件夹：factory.setRepository( new File( 文件路径 ) ) ;

·创建处理工具：ServletFildUpload upload= new ServletFileUpload( factory ) ;

·设置上传文件大小：upload.setFileSizeMax( 31457248 ) ;

·接收全部内容：List<FileItem> items = upload.parseRequest(request) ;

·遍历控件并判断：Iterator< FileItem > iter=items.iterator( ) ;

在接收全部内容时，会将所有的上传内容一起接收（普文件控件和其它控件）， 所以需要依次判断每一个上传的内容时文件还是普通文本。

## 开发FileUpload组件操作类

FileUpload组件存在以下几点不便之处：

1. 无法像使用request.getParameter( )方法那样准确得取得提交的参数值
2. 无法像使用request.getParameterValues( )方法那样准确的提交的一组参数值
3. 所有的上传文件都需要进行依次的判断，才能够保存，不能一次性批量保存。

如果要解决以上问题，则可以自己进行代码的扩充，编写一个FileUpload操作的工具类。