# 文件的上传和下载

* 文件的上传和下载在web应用中是非常常用，也是非常有用的功能。
  + 例如：发送电子邮件时可以同过上传附件发送文件，OA系统中可以通过上传文件来提交公文，社交网站通过上传图片来自定义头像等等。
  + 例如：下载实际上只要资源放在用户可访问的目录中用户就可以直接通过地址下载，但是一些资源是存放到数据库中的，还有一些资源需要一定权限才能下载，这里就需要我们通过Servlet来完成下载的功能。
* 可以说上传和下载是每一个web应用都需要具有的一个功能，所以需要我们掌握。

## 文件的上传

* 文件的上传主要分成两个步骤
  + 用户在页面中选择要上传的文件，然后将请求提交到Servlet
  + Servlet收到请求，解析用户上传的文件，然后将文件存储到服务器

### 创建上传文件的表单

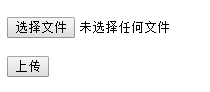
1. 创建一个form表单

|  |
| --- |
| <form action=*""* method=*"post"* enctype=*"multipart/form-data"*>  <input type=*"file"* name=*"file"* /><br /><br />  <input type=*"submit"* value=*"上传"* />  </form> |

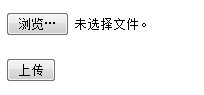
* + 文件上传的表单和之前的表单类似，但有以下内容需要注意
    - 表单的method属性必须为post
    - 表单的enctype属性必须为multipart/form-data
    - 上传文件的控件是intput，type属性为file
  + 该表单打开后是如下效果：
    - IE



* + - Chrome



* + - 火狐



1. 编写Servelet。
   * 页面的表单控件创建好以后，选中文件点击上传按钮请求将会提交到指定的Servlet来处理。
   * 注意：这里不能再像以前的Servlet中那样，通过request.getParamter()来获取请求参数了，当enctype=*"multipart/form-data"* 时，再使用getParamter()获取到内容永远为空。因为浏览器发送请求的方式已经改变。
   * 既然以前的方法不能使用了，这里我们必须要引入一个新的工具来解析请求中的参数和文件，这个工具就是commons-fileupload。

#### commons-fileupload

* + commons-fileupload是Apache开发的一款专门用来处理上传的工具，它的作用就是可以从request对象中解析出，用户发送的请求参数和上传文件的流。
  + commons-fileupload包依赖commons-io，两个包需要同时导入。
  + 核心类：
    1. DiskFileItemFactory
       - 工厂类，用于创建ServletFileUpload，设置缓存等
       - 该类一般直接使用构造器直接创建实例
       - 方法：
         * public void setSizeThreshold(int sizeThreshold)

用于设置缓存文件的大小（默认值10kb）

* + - * + public void setRepository(File repository)

用于设置缓存文件位置（默认系统缓存目录）

* + 1. ServletFileUpload
       - 该类用于解析request对象从而获取用户发送的请求参数（包括普通参数和文件参数）
       - 该类需要调用有参构造器创建实例，构造器中需要一个DiskFileItemFactory作为参数
       - 方法：
         * public List<FileItem> parseRequest(HttpServletRequest request)

解析request对象，获取请求参数，返回的是一个List，List中保存的是一个FileItem对象，一个对象代表一个请求参数。

* + - * + public void setFileSizeMax(long fileSizeMax)

设置单个文件的大小限制，单位为B

如果上传文件超出限制，会在parseRequest()抛出异常

FileSizeLimitExceededException。

* + - * + public void setSizeMax(long sizeMax)

限制请求内容的总大小，单位为B

如果上传文件超出限制，会在parseRequest()抛出异常

SizeLimitExceededException。

* + 1. FileItem
       - 该类用于封装用户发送的参数和文件，也就是用户发送来的信息将会被封装成一个FileItem对象，我们通过该对象获取请求参数或上传文件的信息。
       - 该类不用我们手动创建，由ServletFileItem解析request后返回。
       - 方法：
         * String getFieldName()

获取表单项的名字，也就是input当中的name属性的值。

* + - * + String getName();

获取上传的文件名，普通的请求参数为null。

* + - * + String getString(String encoding);

获取内容

若为文件，将文件的流转换为字符串。

若为请求参数，则获取请求参数的value。

encoding参数需要指定一个字符集

* + - * + boolean isFormField();

判断当前的FileItem封装的是普通请求参数，还是一个文件。

如果为普通参数返回：true

如果为文件参数返回：false

* + - * + String getContentType();

获取上传文件的MIME类型

* + - * + long getSize();

获取内容的大小

* + 实例代码，创建一个Servlet并在doPost()方法中编写如下代码:

|  |
| --- |
| //创建工厂类  DiskFileItemFactory factory = **new** DiskFileItemFactory();  //创建请求解析器  ServletFileUpload fileUpload = **new** ServletFileUpload(factory);  //设置上传单个文件的的大小  fileUpload.setFileSizeMax(1024\*1024\*3);  //设置上传总文件的大小  fileUpload.setSizeMax(1024\*1024\*3\*10);  //设置响应内容的编码  response.setContentType("text/html;charset=utf-8");  **try** {  //解析请求信息，获取FileItem的集合  List<FileItem> items = fileUpload.parseRequest(request);  //遍历集合  **for** (FileItem fileItem : items) {  //如果是普通的表单项  **if**(fileItem.isFormField()){  //获取参数名  String fieldName = fileItem.getFieldName();  //获取参数值  String value = fileItem.getString("utf-8");  System.*out*.println(fieldName+" = "+value);  //如果是文件表单项  }**else**{  //获取文件名  String fileName = fileItem.getName();  //获取上传路径  String realPath = getServletContext().getRealPath("/WEB-INF/upload");  //检查upload文件夹是否存在，如果不存在则创建  File f = **new** File(realPath);  **if**(!f.exists()){  f.mkdir();  };  //为避免重名生成一个uuid作为文件名的前缀  String prefix = UUID.*randomUUID*().toString().replace("-", "");  //将文件写入到服务器中  fileItem.write(**new** File(realPath+"/"+prefix+"\_"+fileName));  //清楚文件缓存  fileItem.delete();  }  }  } **catch** (Exception e) {  **if**(e **instanceof** SizeLimitExceededException){  //文件总大小超出限制  response.getWriter().print("上传文件的总大小不能超过30M");  }**else** **if**(e **instanceof** FileSizeLimitExceededException){  //单个文件大小超出限制  response.getWriter().print("上传单个文件的大小不能超过3M");  }  }  response.getWriter().print("上传成功"); |

## 文件的下载

* + 文件下载最直接的方法就是把文件直接放到服务器的目录中，用户直接访问该文件就可以直接下载。
  + 但是实际上这种方式并不一定好用，比如我们在服务器上直接放置一个MP3文件，然后通过浏览器访问该文件的地址，如果是IE浏览器可能就会弹出下载窗口，而如果是FireFox和Chrome则有可能直接播放。再有就是有一些文件我们是不希望用户可以直接访问到的，这是我们就要通过Servlet来完成下载功能。
  + 下载文件的关键是几点：
    1. 服务器以一个流的形式将文件发送给浏览器。
    2. 发送流的同时还需要设置几个响应头，来告诉浏览器下载的信息。
       - 具体响应头如下：
         * Content-Type

下载文件的MIME类型

可以通过servletContext. getMimeType(String file)获取

也可以直接手动指定

使用response.setContentType(String type);

响应头样式：

Content-Type: audio/mpeg

* + - * + Content-Disposition

下载文件的名字，主要作用是提供一个默认的用户名

通过response.setHeader("Content-Disposition", disposition)设置

响应头样式：

Content-Disposition: attachment; filename=xxx.mp3

* + - * + Content-Length

下载文件的长度，用于设置文件的长处（不必须）

通过response. setContentLength(int len)设置。

设置后样式：

Content-Length: 3140995

* + 1. 接下来需要以输入流的形式读入硬盘上的文件
       - FileInputStream is = new FileInputStream(file);
       - 这个流就是我们一会要发送给浏览器的内容
    2. 通过response获取一个输出流，并将文件（输入流）通过该流发送给浏览器
       - 获取输出流
         * ServletOutputStream out = response.getOutputStream();
       - 通过输出流向浏览器发送文件（不要忘了关闭输入流）

|  |
| --- |
| **byte**[] b = **new** **byte**[1024];  **int** len = 0;  **while**((len=is.read(b))> 0){  out.write(b, 0, len);  }  is.close(); |

### 步骤：

* + 一下步骤都是在同一个Servlet的doGet()方法中编写的
  + 我所下载的文件是放在WEB-INF下mp3文件夹中的文件
  + 具体步骤
    1. 获取文件的流：

|  |
| --- |
| String realPath = getServletContext().getRealPath("/WEB-INF/mp3/中国话.mp3");  //获取文件的File对象  File file = **new** File(realPath);  //获取文件的输入流  FileInputStream is = **new** FileInputStream(file); |

* + 1. 获取头信息

|  |
| --- |
| //获取文件的MIME信息  String contentType = getServletContext().getMimeType(realPath);  //设置下载文件的名字  String filename = "zhongguohua.mp3";  //创建Content-Disposition信息  String disposition = "attachment; filename="+ filename ;  //获取文件长度  **long** size = file.length(); |

* + 1. 设置头信息

|  |
| --- |
| //设置Content-Type  response.setContentType(contentType);  //设置Content-Disposition  response.setHeader("Content-Disposition", disposition);  //设置文件长度  response.setContentLength((**int**)size); |

* + 1. 发送文件

|  |
| --- |
| //通过response获取输出流，用于向浏览器输出内容  ServletOutputStream out = response.getOutputStream();  //将文件输入流通过输出流输出  **byte**[] b = **new** **byte**[1024];  **int** len = 0;  **while**((len=is.read(b))> 0){  out.write(b, 0, len);  }  //最后不要忘记关闭输入流，输出流由Tomcat自己处理，我们不用手动关闭  is.close(); |

### 乱码：

* + 至此实际上文件下载的主要功能都已经完成。但是还有一个问题我们这里没有体现出来，因为目前我们的文件名使用的是纯英文的，没有乱码问题。这里如果我们要使用中文文件名的话，毫无疑问会出现乱码问题。
  + 解决此问题的方法很简单，在获取文件名之后为文件名进行编码：

|  |
| --- |
| filename = java.net.URLEncoder.*encode*(filename,"utf-8"); |

* + 但是注意这里火狐浏览器比较特殊，因为他默认是以BASE64解码的，所以这块如果需要考虑火狐的问题的话还需要特殊处理一下。
    1. 先要获取客户端信息（通过获取请求头中的User-Agent信息）

|  |
| --- |
| //获取客户端信息  String ua = request.getHeader("User-Agent"); |

* + 1. 然后判断浏览器版本，做不同的处理（通过判断头信息中是否包含Firefox字符串来判断浏览器版本）

|  |
| --- |
| //判断客户端是否为火狐  **if**(ua.contains("Firefox")){  //若为火狐使用BASE64编码  filename = "=?utf-8?B?"+**new** BASE64Encoder()  .encode(filename.getBytes("utf-8"))+"?=";  }**else**{  //否则使用UTF-8  filename = URLEncoder.*encode*(filename,"utf-8");  } |