# day22 上传下载JavaMail

## 文件上传概述

###### 1　文件上传的作用

例如网络硬盘！就是用来上传下载文件的。

在智联招聘上填写一个完整的简历还需要上传照片呢。

###### 2　文件上传对页面的要求

上传文件的要求比较多，需要记一下：

1. 必须使用表单，而不能是超链接；
2. 表单的method必须是POST，而不能是GET；
3. 表单的enctype必须是multipart/form-data；
4. 在表单中添加file表单字段，即<input type=”file” name=”xxx” />

|  |
| --- |
| <form action=*"*${pageContext.request.contextPath }*/FileUploadServlet"* method=*"post"* enctype=*"multipart/form-data"*>  用户名：<input type=*"text"* name=*"username"*/><br/>  文件1：<input type=*"file"* name=*"file1"*/><br/>  文件2：<input type=*"file"* name=*"file2"*/><br/>  <input type=*"submit"* value=*"提交"*/>  </form> |

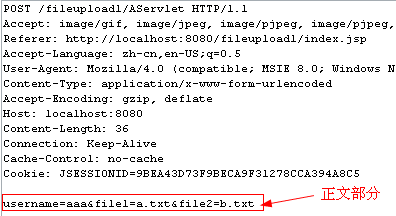
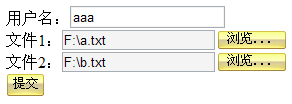
###### 3　比对文件上传表单和普通文本表单的区别

通过httpWatch查看“文件上传表单”和“普通文本表单”的区别。

* 文件上传表单的enctype=”multipart/form-data”，表示多部件表单数据；
* 普通文本表单可以不设置enctype属性：
* 当method=”post”时，enctype的默认值为application/x-www-form-urlencoded，表示使用url编码正文；
* 当method=”get”时，enctype的默认值为null，没有正文，所以就不需要enctype了。

**对普通文本表单的测试**：

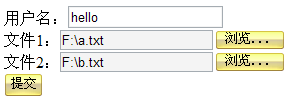
|  |
| --- |
| <form action=*"*${pageContext.request.contextPath }*/FileUploadServlet"* method=*"post"*>  用户名：<input type=*"text"* name=*"username"*/><br/>  文件1：<input type=*"file"* name=*"file1"*/><br/>  文件2：<input type=*"file"* name=*"file2"*/><br/>  <input type=*"submit"* value=*"提交"*/>  </form> |



　　通过httpWatch测试，查看表单的请求数据正文，我们发现请求中只有文件名称，而没有文件内容。也就是说，当表单的enctype不是multipart/form-data时，请求中不包含文件内容，而只有文件的名称，这说明普通文本表单中input:file与input:text没什么区别了。

**对文件上传表单的测试**：

|  |
| --- |
| <form action=*"*${pageContext.request.contextPath }*/FileUploadServlet"* method=*"post"* enctype=*"multipart/form-data"*>  用户名：<input type=*"text"* name=*"username"*/><br/>  文件1：<input type=*"file"* name=*"file1"*/><br/>  文件2：<input type=*"file"* name=*"file2"*/><br/>  <input type=*"submit"* value=*"提交"*/>  </form> |





通过httpWatch测试，查看表单的请求数据正文部分，发现正文部分是由多个部件组成，每个部件对应一个表单字段，每个部件都有自己的头信息。头信息下面是空行，空行下面是字段的正文部分。多个部件之间使用随机生成的分隔线隔开。

文本字段的头信息中只包含一条头信息，即Content-Disposition，这个头信息的值有两个部分，第一部分是固定的，即form-data，第二部分为字段的名称。在空行后面就是正文部分了，正文部分就是在文本框中填写的内容。

　　文件字段的头信息中包含两条头信息，Content-Disposition和Content-Type。Content-Disposition中多出一个filename，它指定的是上传的文件名称。而Content-Type指定的是上传文件的类型。文件字段的正文部分就是文件的内容。

　　请注意，因为我们上传的文件都是普通文本文件，即txt文件，所以在httpWatch中是可以正常显示的，如果上传的是exe、mp3等文件，那么在httpWatch看到的就是乱码了。

###### 4　文件上传对Servlet的要求

当提交的表单是文件上传表单时，那么对Servlet也是有要求的。

首先我们要肯定一点，文件上传表单的数据也是被封装到request对象中的。

**request.getParameter(String)方法获取指定的表单字段字符内容，但文件上传表单已经不在是字符内容，而是字节内容，所以失效。**

这时可以使用request的getInputStream()方法获取ServletInputStream对象，它是InputStream的子类，这个ServletInputStream对象对应整个表单的正文部分（从第一个分隔线开始，到最后），这说明我们需要的解析流中的数据。当然解析它是很麻烦的一件事情，而Apache已经帮我们提供了解析它的工具：commons-fileupload。

　　可以尝试把request.getInputStream()这个流中的内容打印出来，再对比httpWatch中的请求数据。

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  InputStream in = request.getInputStream();  String s = IOUtils.*toString*(in);  System.*out*.println(s);  } |
| -----------------------------7ddd3370ab2  Content-Disposition: form-data; name="username"  hello  -----------------------------7ddd3370ab2  Content-Disposition: form-data; name="file1"; filename="a.txt"  Content-Type: text/plain  aaa  -----------------------------7ddd3370ab2  Content-Disposition: form-data; name="file2"; filename="b.txt"  Content-Type: text/plain  bbb  -----------------------------7ddd3370ab2-- |

## commons-fileupload

为什么使用fileupload：

上传文件的要求比较多，需要记一下：

* 必须是POST表单；
* 表单的enctype必须是multipart/form-data；
* 在表单中添加file表单字段，即<input type=”file” name="xxx" />

Servlet的要求：

* 不能再使用request.getParameter()来获取表单数据；
* 可以使用request.getInputStream()得到所有的表单数据，而不是一个表单项的数据；
* 这说明不使用fileupload，我们需要自己来对request.getInputStream()的内容进行解析！！！

###### 1　fileupload概述

fileupload是由apache的commons组件提供的上传组件。它最主要的工作就是帮我们解析request.getInputStream()。

fileupload组件需要的JAR包有：

* commons-fileupload.jar，核心包；
* commons-io.jar，依赖包。

###### 2　fileupload简单应用

　　fileupload的核心类有：DiskFileItemFactory、ServletFileUpload、FileItem。

使用fileupload组件的步骤如下：

1. 创建工厂类DiskFileItemFactory对象：DiskFileItemFactory factory = new DiskFileItemFactory()
2. 使用工厂创建解析器对象：ServletFileUpload fileUpload = new ServletFileUpload(factory)
3. 使用解析器来解析request对象：List<FileItem> list = fileUpload.parseRequest(request)

隆重介绍FileItem类，它才是我们最终要的结果。一个FileItem对象对应一个表单项（表单字段）。一个表单中存在文件字段和普通字段，可以使用FileItem类的isFormField()方法来判断表单字段是否为普通字段，如果不是普通字段，那么就是文件字段了。

* String getName()：获取文件字段的文件名称；
* String getString()：获取字段的内容，如果是文件字段，那么获取的是文件内容，当然上传的文件必须是文本文件；
* String getFieldName()：获取字段名称，例如：<input type=”text” name=”username”/>，返回的是username；
* String getContentType()：获取上传的文件的类型，例如：text/plain。
* int getSize()：获取上传文件的大小；
* boolean isFormField()：判断当前表单字段是否为普通文本字段，如果返回false，说明是文件字段；
* InputStream getInputStream()：获取上传文件对应的输入流；
* void write(File)：把上传的文件保存到指定文件中。

###### 3　简单上传示例

写一个简单的上传示例：

* 表单包含一个用户名字段，以及一个文件字段；
* Servlet保存上传的文件到uploads目录，显示用户名，文件名，文件大小，文件类型。

第一步：

完成index.jsp，只需要一个表单。注意表单必须是post的，而且enctype必须是mulitpart/form-data的。

|  |
| --- |
| <form action=*"*${pageContext.request.contextPath }*/FileUploadServlet"* method=*"post"* enctype=*"multipart/form-data"*>  用户名：<input type=*"text"* name=*"username"*/><br/>  文件1：<input type=*"file"* name=*"file1"*/><br/>  <input type=*"submit"* value=*"提交"*/>  </form> |

第二步：

完成FileUploadServlet

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  // 因为要使用response打印，所以设置其编码  response.setContentType("text/html;charset=utf-8");    // 创建工厂  DiskFileItemFactory dfif = **new** DiskFileItemFactory();  // 使用工厂创建解析器对象  ServletFileUpload fileUpload = **new** ServletFileUpload(dfif);  **try** {  // 使用解析器对象解析request，得到FileItem列表  List<FileItem> list = fileUpload.parseRequest(request);  // 遍历所有表单项  **for**(FileItem fileItem : list) {  // 如果当前表单项为普通表单项  **if**(fileItem.isFormField()) {  // 获取当前表单项的字段名称  String fieldName = fileItem.getFieldName();  // 如果当前表单项的字段名为username  **if**(fieldName.equals("username")) {  // 打印当前表单项的内容，即用户在username表单项中输入的内容  response.getWriter().print("用户名：" + fileItem.getString() + "<br/>");  }  } **else** {//如果当前表单项不是普通表单项，说明就是文件字段  String name = fileItem.getName();//获取上传文件的名称  // 如果上传的文件名称为空，即没有指定上传文件  **if**(name == **null** || name.isEmpty()) {  **continue**;  }  // 获取真实路径，对应${项目目录}/uploads，当然，这个目录必须存在  String savepath = **this**.getServletContext().getRealPath("/uploads");  // 通过uploads目录和文件名称来创建File对象  File file = **new** File(savepath, name);  // 把上传文件保存到指定位置  fileItem.write(file);  // 打印上传文件的名称  response.getWriter().print("上传文件名：" + name + "<br/>");  // 打印上传文件的大小  response.getWriter().print("上传文件大小：" + fileItem.getSize() + "<br/>");  // 打印上传文件的类型  response.getWriter().print("上传文件类型：" + fileItem.getContentType() + "<br/>");  }  }  } **catch** (Exception e) {  **throw** **new** ServletException(e);  }  } |

## 文件上传之细节

###### 1　把上传的文件放到WEB-INF目录下

如果没有把用户上传的文件存放到WEB-INF目录下，那么用户就可以通过浏览器直接访问上传的文件，这是非常危险的。

假如说用户上传了一个a.jsp文件，然后用户在通过浏览器去访问这个a.jsp文件，那么就会执行a.jsp中的内容，如果在a.jsp中有如下语句：Runtime.getRuntime().exec(“shutdown –s –t 1”);，那么你就会…

通常我们会在WEB-INF目录下创建一个uploads目录来存放上传的文件，而在Servlet中找到这个目录需要使用ServletContext的getRealPath(String)方法，例如在我的upload1项目中有如下语句：

ServletContext servletContext = this.getServletContext();

String savepath = servletContext.getRealPath(“/WEB-INF/uploads”);

其中savepath为：F:\tomcat6\_1\webapps\upload1\WEB-INF\uploads。

###### 2　文件名称（完整路径、文件名称）

**上传文件名称可能是完整路径**：

IE6获取的上传文件名称是完整路径，而其他浏览器获取的上传文件名称只是文件名称而已。浏览器差异的问题我们还是需要处理一下的。

|  |
| --- |
| String name = file1FileItem.getName();  response.getWriter().print(name); |

使用不同浏览器测试，其中IE6就会返回上传文件的完整路径，不知道IE6在搞什么，这给我们带来了很大的麻烦，就是需要处理这一问题。

处理这一问题也很简单，无论是否为完整路径，我们都去截取最后一个“\\”后面的内容就可以了。

|  |
| --- |
| String name = file1FileItem.getName();  **int** lastIndex = name.lastIndexOf("\\");//获取最后一个“\”的位置  **if**(lastIndex != -1) {//注意，如果不是完整路径，那么就不会有“\”的存在。  name = name.substring(lastIndex + 1);//获取文件名称  }  response.getWriter().print(name); |

###### 3　中文乱码问题

**上传文件名称中包含中文**：

当上传的谁的名称中包含中文时，需要设置编码，commons-fileupload组件为我们提供了两种设置编码的方式：

* request.setCharacterEncoding(String)：这种方式是我们最为熟悉的方式了；
* fileUpload.setHeaderEncdoing(String)：这种方式的优先级高与前一种。

**上传文件的文件内容包含中文：**

通常我们不需关心上传文件的内容，因为我们会把上传文件保存到硬盘上！也就是说，文件原来是什么样子，到服务器这边还是什么样子！

但是如果你有这样的需求，非要在控制台显示上传的文件内容，那么你可以使用fileItem.getString(“utf-8”)来处理编码。

**文本文件内容和普通表单项内容使用FileItem类的getString(“utf-8”)来处理编码。**

###### 4　上传文件同名问题（文件重命名）

通常我们会把用户上传的文件保存到uploads目录下，但如果用户上传了同名文件呢？这会出现覆盖的现象。处理这一问题的手段是使用UUID生成唯一名称，然后再使用“\_”连接文件上传的原始名称。

例如用户上传的文件是“我的一寸照片.jpg”，在通过处理后，文件名称为：“891b3881395f4175b969256a3f7b6e10\_我的一寸照片.jpg”，这种手段不会使文件丢失扩展名，并且因为UUID的唯一性，上传的文件同名，但在服务器端是不会出现同名问题的。

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  DiskFileItemFactory dfif = **new** DiskFileItemFactory();  ServletFileUpload fileUpload = **new** ServletFileUpload(dfif);  **try** {  List<FileItem> list = fileUpload.parseRequest(request);  //获取第二个表单项，因为第一个表单项是username，第二个才是file表单项  FileItem fileItem = list.get(1);  String name = fileItem.getName();//获取文件名称    // 如果客户端使用的是IE6，那么需要从完整路径中获取文件名称  **int** lastIndex = name.lastIndexOf("\\");  **if**(lastIndex != -1) {  name = name.substring(lastIndex + 1);  }    // 获取上传文件的保存目录  String savepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads");  String uuid = CommonUtils.*uuid*();//生成uuid  String filename = uuid + "\_" + name;//新的文件名称为uuid + 下划线 + 原始名称    //创建file对象，下面会把上传文件保存到这个file指定的路径  //savepath，即上传文件的保存目录  //filename，文件名称  File file = **new** File(savepath, filename);    // 保存文件  fileItem.write(file);  } **catch** (Exception e) {  **throw** **new** ServletException(e);  }  } |

###### 5　一个目录不能存放过多的文件（存放目录打散）

一个目录下不应该存放过多的文件，一般一个目录存放1000个文件就是上限了，如果在多，那么打开目录时就会很“卡”。你可以尝试打印C:\WINDOWS\system32目录，你会感觉到的。

也就是说，我们需要把上传的文件放到不同的目录中。但是也不能为每个上传的文件一个目录，这种方式会导致目录过多。所以我们应该采用某种算法来“打散”！

打散的方法有很多，例如使用日期来打散，每天生成一个目录。也可以使用文件名的首字母来生成目录，相同首字母的文件放到同一目录下。

日期打散算法：如果某一天上传的文件过多，那么也会出现一个目录文件过多的情况；

首字母打散算法：如果文件名是中文的，因为中文过多，所以会导致目录过多的现象。

我们这里使用hash算法来打散：

1. 获取文件名称的hashCode：int hCode = name.hashCode();；
2. 获取hCode的低4位，然后转换成16进制字符；
3. 获取hCode的5~8位，然后转换成16进制字符；
4. 使用这两个16进制的字符生成目录链。例如低4位字符为“5”

这种算法的好处是，在uploads目录下最多生成16个目录，而每个目录下最多再生成16个目录，即256个目录，所有上传的文件都放到这256个目录下。如果每个目录上限为1000个文件，那么一共可以保存256000个文件。

例如上传文件名称为：新建 文本文档.txt，那么把“新建 文本文档.txt”的哈希码获取到，再获取哈希码的低4位，和5~8位。假如低4位为：9，5~8位为1，那么文件的保存路径为uploads/9/1/。

|  |
| --- |
| **int** hCode = name.hashCode();//获取文件名的hashCode  //获取hCode的低4位，并转换成16进制字符串  String dir1 = Integer.*toHexString*(hCode & 0xF);  //获取hCode的低5~8位，并转换成16进制字符串  String dir2 = Integer.*toHexString*(hCode >>> 4 & 0xF);  //与文件保存目录连接成完整路径  savepath = savepath + "/" + dir1 + "/" + dir2;  //因为这个路径可能不存在，所以创建成File对象，再创建目录链，确保目录在保存文件之前已经存在  **new** File(savepath).mkdirs(); |

###### 6　上传的单个文件的大小限制

限制上传文件的大小很简单，ServletFileUpload类的setFileSizeMax(long)就可以了。参数就是上传文件的上限字节数，例如servletFileUpload.setFileSizeMax(1024\*10)表示上限为10KB。

一旦上传的文件超出了上限，那么就会抛出FileUploadBase.FileSizeLimitExceededException异常。我们可以在Servlet中获取这个异常，然后向页面输出“上传的文件超出限制”。

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  DiskFileItemFactory dfif = **new** DiskFileItemFactory();  ServletFileUpload fileUpload = **new** ServletFileUpload(dfif);  // 设置上传的单个文件的上限为10KB  fileUpload.setFileSizeMax(1024 \* 10);  **try** {  List<FileItem> list = fileUpload.parseRequest(request);  //获取第二个表单项，因为第一个表单项是username，第二个才是file表单项  FileItem fileItem = list.get(1);  String name = fileItem.getName();//获取文件名称    // 如果客户端使用的是IE6，那么需要从完整路径中获取文件名称  **int** lastIndex = name.lastIndexOf("\\");  **if**(lastIndex != -1) {  name = name.substring(lastIndex + 1);  }    // 获取上传文件的保存目录  String savepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads");  String uuid = CommonUtils.*uuid*();//生成uuid  String filename = uuid + "\_" + name;//新的文件名称为uuid + 下划线 + 原始名称    **int** hCode = name.hashCode();//获取文件名的hashCode  //获取hCode的低4位，并转换成16进制字符串  String dir1 = Integer.*toHexString*(hCode & 0xF);  //获取hCode的低5~8位，并转换成16进制字符串  String dir2 = Integer.*toHexString*(hCode >>> 4 & 0xF);  //与文件保存目录连接成完整路径  savepath = savepath + "/" + dir1 + "/" + dir2;  //因为这个路径可能不存在，所以创建成File对象，再创建目录链，确保目录在保存文件之前已经存在  **new** File(savepath).mkdirs();    //创建file对象，下面会把上传文件保存到这个file指定的路径  //savepath，即上传文件的保存目录  //filename，文件名称  File file = **new** File(savepath, filename);    // 保存文件  fileItem.write(file);  } **catch** (Exception e) {  // 判断抛出的异常的类型是否为FileUploadBase.FileSizeLimitExceededException  // 如果是，说明上传文件时超出了限制。  **if**(e **instanceof** FileUploadBase.FileSizeLimitExceededException) {  // 在request中保存错误信息  request.setAttribute("msg", "上传失败！上传的文件超出了10KB！");  // 转发到index.jsp页面中！在index.jsp页面中需要使用${msg}来显示错误信息  request.getRequestDispatcher("/index.jsp").forward(request, response);  **return**;  }  **throw** **new** ServletException(e);  }  } |

###### 7　上传文件的总大小限制

上传文件的表单中可能允许上传多个文件，例如：



有时我们需要限制一个请求的大小。也就是说这个请求的最大字节数（所有表单项之和）！实现这一功能也很简单，只需要调用ServletFileUpload类的setSizeMax(long)方法即可。

例如fileUpload.setSizeMax(1024 \* 10);，显示整个请求的上限为10KB。当请求大小超出10KB时，ServletFileUpload类的parseRequest()方法会抛出FileUploadBase.SizeLimitExceededException异常。

###### 8　缓存大小与临时目录

大家想一想，如果我上传一个蓝光电影，先把电影保存到内存中，然后再通过内存copy到服务器硬盘上，那么你的内存能吃的消么？

所以fileupload组件不可能把文件都保存在内存中，fileupload会判断文件大小是否超出10KB，如果是那么就把文件保存到硬盘上，如果没有超出，那么就保存在内存中。

**10KB是fileupload默认的值，我们可以来设置它。**

**当文件保存到硬盘时，fileupload是把文件保存到系统临时目录，当然你也可以去设置临时目录。**



|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  request.setCharacterEncoding("utf-8");  DiskFileItemFactory dfif = **new** DiskFileItemFactory(1024\*20, **new** File("F:\\temp"));  ServletFileUpload fileUpload = **new** ServletFileUpload(dfif);    **try** {  List<FileItem> list = fileUpload.parseRequest(request);  FileItem fileItem = list.get(1);  String name = fileItem.getName();  String savepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads");    // 保存文件  fileItem.write(path(savepath, name));  } **catch** (Exception e) {  **throw** **new** ServletException(e);  }  }    **private** File path(String savepath, String filename) {  // 从完整路径中获取文件名称  **int** lastIndex = filename.lastIndexOf("\\");  **if**(lastIndex != -1) {  filename = filename.substring(lastIndex + 1);  }    // 通过文件名称生成一级、二级目录  **int** hCode = filename.hashCode();  String dir1 = Integer.*toHexString*(hCode & 0xF);  String dir2 = Integer.*toHexString*(hCode >>> 4 & 0xF);  savepath = savepath + "/" + dir1 + "/" + dir2;  // 创建目录  **new** File(savepath).mkdirs();    // 给文件名称添加uuid前缀  String uuid = CommonUtils.*uuid*();  filename = uuid + "\_" + filename;    // 创建文件完成路径  **return** **new** File(savepath, filename);  } |

## 文件下载

###### 2　通过Servlet下载1

被下载的资源必须放到WEB-INF目录下（只要用户不能通过浏览器直接访问就OK），然后通过Servlet完成下载。

在jsp页面中给出超链接，链接到DownloadServlet，并提供要下载的文件名称。然后DownloadServlet获取文件的真实路径，然后把文件写入到response.getOutputStream()流中。

download.jsp

|  |
| --- |
| <body>  This is my JSP page. <br>  <a href=*"*<c:url value=*'/DownloadServlet?path=a.avi'*/>*"*>a.avi</a><br/>  <a href=*"*<c:url value=*'/DownloadServlet?path=a.jpg'*/>*"*>a.jpg</a><br/>  <a href=*"*<c:url value=*'/DownloadServlet?path=a.txt'*/>*"*>a.txt</a><br/>  </body> |

DownloadServlet.java

|  |
| --- |
| **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {  String filename = request.getParameter("path");  String filepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads/" + filename);  File file = **new** File(filepath);  **if**(!file.exists()) {  response.getWriter().print("您要下载的文件不存在！");  **return**;  }  IOUtils.*copy*(**new** FileInputStream(file), response.getOutputStream());  } |

上面代码有如下问题：

* 可以下载a.avi，但在下载框中的文件名称是DownloadServlet；
* 不能下载a.jpg和a.txt，而是在页面中显示它们。



###### 3　通过Servlet下载2

下面来处理上一例中的问题，让下载框中可以显示正确的文件名称，以及可以下载a.jpg和a.txt文件。

通过添加content-disposition头来处理上面问题。当设置了content-disposition头后，浏览器就会弹出下载框。

而且还可以通过content-disposition头来指定下载文件的名称！

|  |
| --- |
| String filename = request.getParameter("path");  String filepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads/" + filename);  File file = **new** File(filepath);  **if**(!file.exists()) {  response.getWriter().print("您要下载的文件不存在！");  **return**;  }  response.addHeader("content-disposition", "attachment;filename=" + filename);  IOUtils.*copy*(**new** FileInputStream(file), response.getOutputStream()); |







　　虽然上面的代码已经可以处理txt和jpg等文件的下载问题，并且也处理了在下载框中显示文件名称的问题，但是如果下载的文件名称是中文的，那么还是不行的。

###### 3　通过Servlet下载3

下面是处理在下载框中显示中文的问题！

其实这一问题很简单，只需要通过URL来编码中文即可！

download.jsp

|  |
| --- |
| <a href=*"*<c:url value=*'/DownloadServlet?path=这个杀手不太冷.avi'*/>*"*>这个杀手不太冷.avi</a><br/>  <a href=*"*<c:url value=*'/DownloadServlet?path=白冰.jpg'*/>*"*>白冰.jpg</a><br/>  <a href=*"*<c:url value=*'/DownloadServlet?path=说明文档.txt'*/>*"*>说明文档.txt</a><br/> |

DownloadServlet.java

|  |
| --- |
| String filename = request.getParameter("path");  // GET请求中，参数中包含中文需要自己动手来转换。  // 当然如果你使用了“全局编码过滤器”，那么这里就不用处理了  filename = **new** String(filename.getBytes("ISO-8859-1"), "UTF-8");    String filepath = **this**.getServletContext().getRealPath("/WEB-INF/uploads/" + filename);  File file = **new** File(filepath);  **if**(!file.exists()) {  response.getWriter().print("您要下载的文件不存在！");  **return**;  }  // 所有浏览器都会使用本地编码，即中文操作系统使用GBK  // 浏览器收到这个文件名后，会使用iso-8859-1来解码  filename = new String(filename.getBytes("GBK"), "ISO-8859-1");  response.addHeader("content-disposition", "attachment;filename=" + filename);  IOUtils.*copy*(**new** FileInputStream(file), response.getOutputStream()); |

# JavaMail

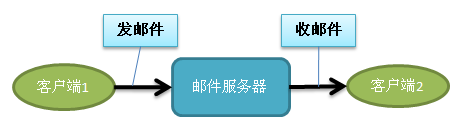
今日内容

* 邮件协议
* telnet访问邮件服务器
* JavaMail

## 邮件协议

###### 1　收发邮件

　　发邮件大家都会吧！发邮件是从客户端把邮件发送到邮件服务器，收邮件是把邮件服务器的邮件下载到客户端。



我们在163、126、QQ、sohu、sina等网站注册的Email账户，其实就是在邮件服务器中注册的。这些网站都有自己的邮件服务器。

###### 2　邮件协议概述

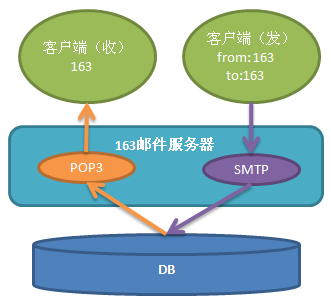
与HTTP协议相同，收发邮件也是需要有传输协议的。

* SMTP：（Simple Mail Transfer Protocol，简单邮件传输协议）发邮件协议；
* POP3：（Post Office Protocol Version 3，邮局协议第3版）收邮件协议；
* IMAP：（Internet Message Access Protocol，因特网消息访问协议）收发邮件协议，我们的课程不涉及该协议。

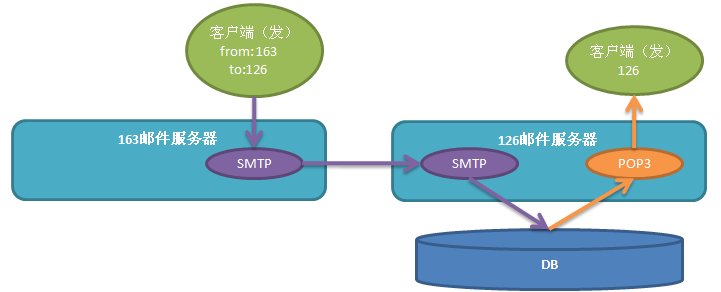
###### 3　理解邮件收发过程

其实你可以把邮件服务器理解为邮局！如果你需要给朋友寄一封信，那么你需要把信放到邮筒中，这样你的信会“自动”到达邮局，邮局会把信邮到另一个省市的邮局中。然后这封信会被送到收信人的邮箱中。最终收信人需要自己经常查看邮箱是否有新的信件。

　　其实每个邮件服务器都由SMTP服务器和POP3服务器构成，其中SMTP服务器负责发邮件的请求，而POP3负责收邮件的请求。



当然，有时我们也会使用163的账号，向126的账号发送邮件。这时邮件是发送到126的邮件服务器，而对于163的邮件服务器是不会存储这封邮件的。



###### 4　邮件服务器名称

smtp服务器的端口号为25，服务器名称为smtp.xxx.xxx。

pop3服务器的端口号为110，服务器名称为pop3.xxx.xxx。

例如：

* 163：smtp.163.com和pop3.163.com；
* 126：smtp.126.com和pop3.126.com；
* qq：smtp.qq.com和pop3.qq.com；
* sohu：smtp.sohu.com和pop3.sohu.com；
* sina：smtp.sina.com和pop3.sina.com。

## telnet收发邮件

###### 1　BASE64加密

BASE64是一种加密算法，这种加密方式是可逆的！它的作用是使加密后的文本无法用肉眼识别。Java提供了sun.misc.BASE64Encoder这个类，用来对做Base64的加密和解密，但我们知道，使用sun包下的东西会有警告！甚至在eclipse中根本使用不了这个类（需要设置），所以我们还是听sun公司的话，不要去使用它内部使用的类，我们去使用apache commons组件中的codec包下的Base64这个类来完成BASE64加密和解密。

|  |
| --- |
| **package** cn.itcast;  **import** org.apache.commons.codec.binary.Base64;  **public** **class** Base64Utils {  **public** **static** String encode(String s) {  **return** *encode*(s, "utf-8");  }    **public** **static** String decode(String s) {  **return** *decode*(s, "utf-8");  }    **public** **static** String encode(String s, String charset) {  **try** {  **byte**[] bytes = s.getBytes(charset);  bytes = Base64.*encodeBase64*(bytes);  **return** **new** String(bytes, charset);  } **catch** (Exception e) {  **throw** **new** RuntimeException(e);  }  }  **public** **static** String decode(String s, String charset) {  **try** {  **byte**[] bytes = s.getBytes(charset);  bytes = Base64.*decodeBase64*(bytes);  **return** **new** String(bytes, charset);  } **catch** (Exception e) {  **throw** **new** RuntimeException(e);  }  }  } |

###### 2　telnet发邮件

连接163的smtp服务器：；

连接成功后需要如下步骤才能发送邮件：

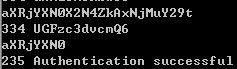
1. 与服务器打招呼：**ehlo你的名字**



1. 发出登录请求：**auth login**



1. 输入加密后的邮箱名：(itcast\_cxf@163.com)**aXRjYXN0X2N4ZkAxNjMuY29t**
2. 输入加密后的邮箱密码：(itcast)**aXRjYXN0**



1. 输入谁来发送邮件，即from：**mail from:<itcast\_cxf@163.com>**



1. 输入把邮件发给谁，即to：**rcpt to:<itcast\_cxf@126.com>**



1. 发送填写数据请求：**data**



1. 开始输入数据，数据包含：from、to、subject，以及邮件内容，如果输入结束后，以一个“.”为一行，表示输入结束：

**from:<zhangBoZhi@163.com>**

**to:<itcast\_cxf@sina.com>**

**subject: 我爱上你了**

**我已经深深的爱上你了，我是张柏芝。**

**.**

注意，在标题和邮件正文之间要有一个空行！当要退出时，一定要以一个“.”为单行，表示输入结束。

　　9　最后一步：**quit**



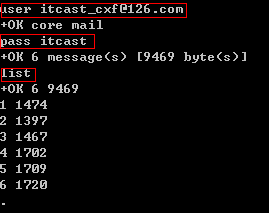
## telnet收邮件

###### 1　telnet收邮件的步骤

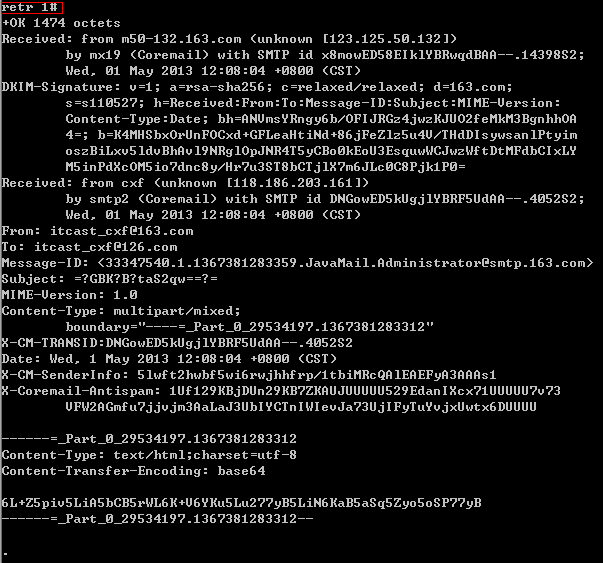
pop3无需使用Base64加密！！！

收邮件连接的服务器是pop3.xxx.com，pop3协议的默认端口号是110。请注意！这与发邮件完全不同。如果你在163有邮箱账户，那么你想使用telnet收邮件，需要连接的服务器是pop3.163.com。

* 连接pop3服务器：telnet pop3.163.com 110
* user命令：user 用户名，例如：user itcast\_cxf@163.com；
* pass命令：pass 密码，例如：pass itcast；
* stat命令：stat命令用来查看邮箱中邮件的个数，所有邮件所占的空间；
* list命令：list命令用来查看所有邮件，或指定邮件的状态，例如：list 1是查看第一封邮件的大小，list是查看邮件列表，即列出所有邮件的编号，及大小；
* retr命令：查看指定邮件的内容，例如：retr 1#是查看第一封邮件的内容；
* dele命令：标记某邮件为删除，但不是马上删除，而是在退出时才会真正删除；
* quit命令：退出！如果在退出之前已经使用dele命令标记了某些邮件，那么会在退出是删除它们。







## JavaMail

###### 1　JavaMail概述

Java Mail是由SUN公司提供的专门针对邮件的API，主要Jar包：mail.jar、activation.jar。

在使用MyEclipse创建web项目时，需要小心！如果只是在web项目中使用java mail是没有什么问题的，发布到Tomcat上运行一点问题都没有！

但是如果是在web项目中写测试那就出问题了。

在MyEclipse中，会自动给web项目导入javax.mail包中的类，但是不全（其实是只有接口，而没有接口的实现类），所以只靠MyEclipse中的类是不能运行java mail项目的，但是如果这时你再去自行导入mail.jar时，就会出现冲突。

处理方案：到下面路径中找到javaee.jar文件，把javax.mail删除！！！

D:\Program Files\MyEclipse\Common\plugins\com.genuitec.eclipse.j2eedt.core\_10.0.0.me201110301321\data\libraryset\EE\_5

###### 2　JavaMail中主要类

java mail中主要类：javax.mail.Session、javax.mail.internet.MimeMessage、javax.mail.Transport。

Session：表示会话，即客户端与邮件服务器之间的会话！想获得会话需要给出账户和密码，当然还要给出服务器名称。在邮件服务中的Session对象，就相当于连接数据库时的Connection对象。

MimeMessage：表示邮件类，它是Message的子类。它包含邮件的主题（标题）、内容，收件人地址、发件人地址，还可以设置抄送和暗送，甚至还可以设置附件。

Transport：用来发送邮件。它是发送器！

###### 3　JavaMail之Hello World

在使用telnet发邮件时，还需要自己来处理Base64编码的问题，但使用JavaMail就不必理会这些问题了，都由JavaMail来处理。

**第一步：获得Session**

Session session = Session.getInstance(Properties prop, Authenticator auth);

其中prop需要指定两个键值，一个是指定服务器主机名，另一个是指定是否需要认证！我们当然需要认证！

Properties prop = new Properties();

prop.setProperty(“mail.host”, “smtp.163.com”);//设置服务器主机名

prop.setProperty(“mail.smtp.auth”, “true”);//设置需要认证

其中Authenticator是一个接口表示认证器，即校验客户端的身份。我们需要自己来实现这个接口，实现这个接口需要使用账户和密码。

Authenticator auth = new Authenticator() {

public PasswordAuthentication getPasswordAuthentication () {

new PasswordAuthentication(“itcast\_cxf”, “itcast”);//用户名和密码

}

};

通过上面的准备，现在可以获取得Session对象了：

Session session = Session.getInstance(prop, auth);

**第二步：创建MimeMessage对象**

创建MimeMessage需要使用Session对象来创建：

MimeMessage msg = new MimeMessage(session);

然后需要设置发信人地址、收信人地址、主题，以及邮件正文。

msg.setFrom(new InternetAddress(“itcast\_cxf@163.com”));//设置发信人

msg.addRecipients(RecipientType.TO, “itcast\_cxf@qq.com,itcast\_cxf@sina.com”);//设置多个收信人

msg.addRecipients(RecipientType.CC, “itcast\_cxf@sohu.com,itcast\_cxf@126.com”);//设置多个抄送

msg.addRecipients(RecipientType.BCC, ”itcast\_cxf@hotmail.com”);//设置暗送

msg.setSubject(“这是一封测试邮件”);//设置主题（标题）

msg.setContent(“当然是hello world!”, “text/plain;charset=utf-8”);//设置正文

**第三步：发送邮件**

Transport.send(msg);//发送邮件

###### 4　JavaMail发送带有附件的邮件（了解）

一封邮件可以包含正文、附件N个，所以正文与N个附件都是邮件的一个部份。

上面的hello world案例中，只是发送了带有正文的邮件！所以在调用setContent()方法时直接设置了正文，如果想发送带有附件邮件，那么需要设置邮件的内容为MimeMultiPart。

MimeMulitpart parts = new MimeMulitpart();//多部件对象，可以理解为是部件的集合

msg.setContent(parts);//设置邮件的内容为多部件内容。

然后我们需要把正文、N个附件创建为“主体部件”对象（MimeBodyPart），添加到MimeMuiltPart中即可。

MimeBodyPart part1 = new MimeBodyPart();//创建一个部件

part1.setCotnent(“这是正文部分”, “text/html;charset=utf-8”);//给部件设置内容

parts.addBodyPart(part1);//把部件添加到部件集中。

下面我们创建一个附件：

MimeBodyPart part2 = new MimeBodyPart();//创建一个部件

part2.attachFile(“F:\\a.jpg”);//设置附件

part2.setFileName(“hello.jpg”);//设置附件名称

parts.addBodyPart(part2);//把附件添加到部件集中

注意，如果在设置文件名称时，文件名称中包含了中文的话，那么需要使用MimeUitlity类来给中文编码：

part2.setFileName(MimeUitlity.encodeText(“美女.jpg”));

