




软件开发环境

主讲教师 刘凡

fanliu@hhu.edu.cn



第十章

JSP的文件操作



本章主要内容

- ◆10.1 File类
- ◆10.2 读写文件的常见流
- ◆10.3 RandomAccessFile类
- ◆10.4 文件上传
- ◆10.5 文件下载

概述

JSP通过Java的输入/输出流来实现文件的读写操作。

本章采用MVC的设计模式来学习文件的操作。

- 输入/输出流

- 应用：文件上传、文件下载

10.1 File类

- File类的对象主要用来获取文件本身的一些信息，例如文件所在的目录、文件的长度、文件读写权限等，不涉及对文件的读写操作。
- 创建一个File对象的构造方法有3个：

File(String filename);

File(String directoryPath,String filename);

File(File f, String filename);

10.1 File类

经常使用File类的下列方法获取文件本身的一些信息：

- public String **getName()** —获取文件的名字
- public boolean **canRead()** —判断文件是否是可读的
- public boolean **canWrite()** —判断文件是否可被写入
- public boolean **exists()** —判断文件是否存在
- public long **length()** —获取文件的长度（单位是字节）
- public String **getAbsolutePath()** —获取文件的绝对路径
- public String **getParent()** —获取文件的父目录
- public boolean **isFile()** —判断是否是正常文件，而不是目录
- public boolean **isDirectory()** —判断文件是否是一个目录
- public boolean **isHidden()** —判定是否隐藏文件
- public long **lastModified()** —获取最后修改时间

10.1 File类

example8_1.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*"%>
<HTML><body bgcolor=cyan><font Size=2>
<% String jspPage=request.getServletPath(); //请求的页面的名称
String webDir = request.getContextPath();//获取当前Web服务目录的名称
webDir = webDir.substring(1); //去掉名称前面的目录符号: /
jspPage =jspPage.substring(1); //去掉名称前面的目录符号: /
File f= new File(""); //该文件认为在Tomcat引擎启动的目录中, 即bin目录中
String path = f.getAbsolutePath();
int index = path.indexOf("bin");
String tomcatDir = path.substring(0,index);//tomcat的安装目录
File file=new File(tomcatDir+"/webapps/"+webDir,jspPage);
%>
文件<%=file.getName()%>是可读的吗?<b><%=file.canRead()%></b>
<br>文件<%=file.getName()%>的长度<b><%=file.length()%>字节</b>
<br><%=file.getName()%>的父目录是:<br><b><%=file.getParent()%></b>
<br><%=file.getName()%>的绝对路径是: <br><b><%=file.getAbsolutePath()%>
</font></body></HTML>
```

10.1 File类

1. 创建目录

- File对象调用方法 **public boolean mkdir()** 创建一个目录
- 如果创建成功就返回 **true**
- 否则返回 **false**
- 如果该目录已经存在也将返回 **false**

10.1 File类

example8_2.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*"%>
<HTML><body bgcolor=#EEFFAD><font Size=2>
<%
```

String webDir = request.getContextPath();//获取当前Web服务目录的名称

webDir = webDir.substring(1);//去掉名称前面的目录符号: /

File f= new File("");//该文件认为在Tomcat引擎启动的目录中, 即bin目录中

String path = f.getAbsolutePath();

int index = path.indexOf("bin");

String tomcatDir = path.substring(0,index);//tomcat的安装目录

File dir=new File(tomcatDir+"/webapps/"+webDir+"/image");

```
%>
```

**
** 在**<%=webDir%>**下创建一个新的目录: **image**,**
**成功创建了吗?

<%=dir.mkdir()%>

**
** **image**是目录吗? **<%=dir.isDirectory()%>**

```
</font></body></HTML>
```

10.1 File类

2. 列出目录中的文件

- 如果File对象是一个目录，那么该对象可以调用下面的方法获得目录下的文件和子目录
- **public String [] list()** 用字符串形式返回目录下的全部文件.
- **public File [] listFiles()** 用File对象形式返回目录下的全部文件。

10.1 File类

3. 列出指定类型的文件

- **public String [] list(FilenameFilter obj)** 该方法用字符串形式返回目录下的指定类型的所有文件。
- **public File [] listFiles(FilenameFilter obj)** 该方法用File对象返回目录下的指定类型的所有文件。

10.1 File类

example8_3.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import="java.io.*"%>
<%! class FileJSP implements FilenameFilter{
    String str=null;
    FileJSP(String s){
        str="."+s;
    }
    public boolean accept(File dir,String name){
        return name.endsWith(str);
    }
}
%>
```

10.1 File类

example8_3.jsp

```
<HTML><body bgcolor=#EEFFAD><font Size=2>
<%
    String name="jsp";
    String webDir = request.getContextPath();
    webDir = webDir.substring(1);
    File f= new File("");
    String path = f.getAbsolutePath();
    int index = path.indexOf("bin");
    String tomcatDir = path.substring(0,index); //tomcat的安装目录
    File dir=new File(tomcatDir+"/webapps/"+webDir);
%>
<br> 在<%=dir%>下<%=name%>文件:
<%
    FileJSP file_jsp=new FileJSP(name);
    String file_name[]=dir.list(file_jsp);
    for(int i=0;i<file_name.length;i++)
        out.print("<br>" +file_name[i]);
%></font></body></HTML>
```

10.1 File类

4. 删除文件和目录

- File对象调用方法 `public boolean delete()` 删除当前对象代表的文件或目录.
- 如果对象表示的是一个目录, 则该目录必须是空目录.
- 删除成功返回 `true`.

10.2 读写文件的常用流

- Java的I/O流提供一条通道程序，可以使用这条通道把源中的数据送给目的地。
- 输入流的指向称做源，程序从指向源的输入流中读取源中的数据。
- 输出流的指向是数据要去的一个目的地，程序通过向输出流中写入数据把信息传递到目的地

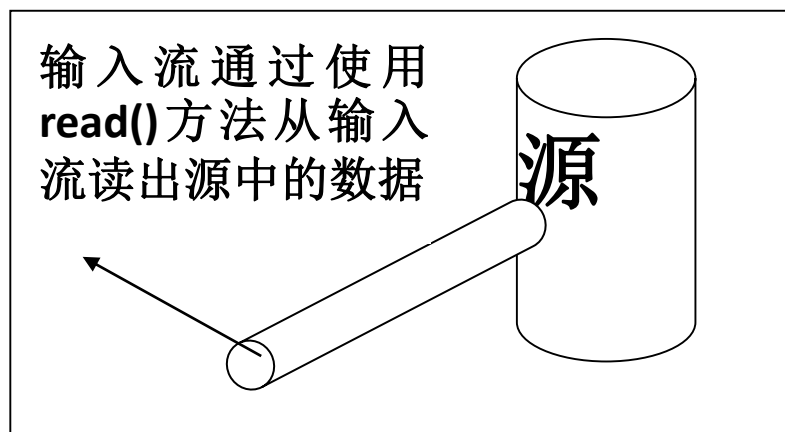


图8.2 输入流示意图

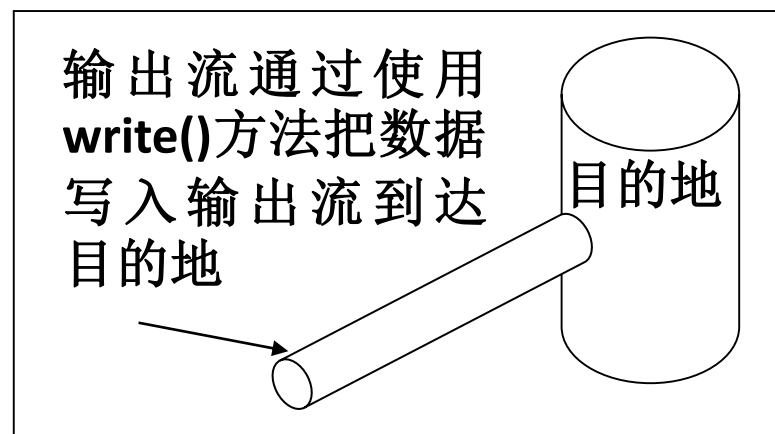


图8.3 输出流示意图

10.2 读写文件的常用流

1. 字节输入流

- 如果对文件读取需求比较简单，那么可以使用 `FileInputStream` 类（文件字节输入流），该类是 `InputStream` 类的子类（以字节为单位读取文件），该类的实例方法都是从 `InputStream` 类继承来的。
- 使用输入流通常包括4个基本步骤：
 1. 设定输入流的源
 2. 创建指向源的输入流
 3. 让输入流读取源中的数据
 4. 关闭输入流。

10.2 读写文件的常用流

1. 字节输入流-构造方法

FileInputStream(String name);

FileInputStream(File file);

- 第一个构造方法使用给定的文件名name创建FileInputStream流，第二个构造方法使用File对象创建FileInputStream流。

参数name和file指定的文件称为输入流的源。

```
File f = new File("hello.txt"); //指定输入流的源
```

```
try {
```

```
FileInputStream in = new FileInputStream(f); //创建指向源的输入流
```

```
}
```

```
catch (IOException e) {
```

```
    System.out.println("File read error:"+e );
```

```
}
```

10.2 读写文件的常用流

1. 字节输入流-使用输入流读取字节

- 输入流的目的是提供读取源中数据的通道，程序可以通过这个通道读取源中的数据。文件字节流可以调用从父类继承的 `read` 方法顺序地读取文件，只要不关闭流，每次调用 `read` 方法就顺序地读取文件中的其余内容，直到文件的末尾或文件字节输入流被关闭。
- **`int read()`** 读取单个字节的数据，该方法返回字节值（0~255之间的一个整数），如果未读出字节就返回-1。
- **`int read(byte b[])`** 试图读取 `b.length` 个字节到字节数组 `b` 中。
- **`int read(byte b[], int off, int len)`** 试图读取 `len` 个字节到字节数组 `b` 中，参数 `off` 指定从字节数组的某个位置开始存放读取的数据。

10.2 读写文件的常用流

1. 字节输入流-关闭流

- 输入流都提供了关闭方法`close()`，尽管程序结束时会自动关闭所有打开的流，但是当程序使用完流后，显式地关闭任何打开的流仍是一个良好的习惯。

10.2 读写文件的常用流

2. 字节输出流

- 如果对文件写入需求比较简单，那么可以使用 `FileOutputStream` 类（文件字节输出流），它是 `OutputStream` 类的子类（以字节为单位向文件写入内容），该类的实例方法都是从 `OutputStream` 类继承来的。
- 使用输出流通常包括4个基本步骤：
 1. 给出输出流的目的地
 2. 创建指向目的地的输出流
 3. 让输出流把数据写入到目的地
 4. 关闭输出流

10.2 读写文件的常用流

2. 字节输出流-构造方法

FileOutputStream(String name);

FileOutputStream(File file);

- 第一个构造方法使用给定的文件名name创建FileOutputStream流，第二个构造方法使用File对象创建FileOutputStream流。参数name和file指定的文件称为输出流的目的地。
- 如果输出流指向的文件不存在，java就创建该文件，如果已经存在，输出流将刷新该文件，使其长度为0.

10.2 读写文件的常用流

2. 字节输出流-选择是否具有刷新功能的构造方法

FileOutputStream(String name, boolean append);

FileOutputStream(File file, boolean append);

- 如果append=true, 输出流不会刷新所指向的文件, write方法将从文件末尾开始向文件写入数据; 如果append=false, 输出流将刷新所指向的文件

```
File f=new File("destin.txt"); //指定输出流的目的地
try {
    FileOutputStream out = new FileOutputStream(f); //创建指向目的地的输出流
}
catch (IOException e) {
    System.out.println("Filewrite:"+e );
}
```

10.2 读写文件的常用流

2. 字节输出流-使用输出流写字节

- 输出流的目的是提供通往目的地的通道，程序可以通过这个通道将程序中的数据写入到目的地。文件字节流可以调用从父类继承的write方法顺序地写文件。FileOutputStream流顺序地向文件写入内容，即只要不关闭流，每次调用write方法就顺序地向文件写入内容，直到流被关闭。
- **void write(int n)** 输出流调用该方法向目的地写入单个字节
- **void write(byte b[])** 输出流调用该方法向目的地写入一个字节数组。
- **void write(byte b[],int off,int len)** 给定字节数组中起始于偏移量off处取len个字节写到目的地。

10.2 读写文件的常用流

2. 字节输出流-关闭流

- 在操作系统把程序所写到输出流上的那些字节保存到磁盘上之前，有时被存放在内存缓冲区中，通过调用 `close()` 方法，可以保证操作系统把流缓冲区的内容写到它的目的地，即关闭输出流可以把该流所用的缓冲区的内容冲洗掉（通常冲洗到磁盘文件上）

10.2 读写文件的常用流

3. 字符流

- 与FileInputStream、FileOutputStream字节流相对应的是FileReader、FileWriter字符流（文件字符输入、输出流），FileReader和FileWriter分别是Reader和Writer的子类. 字符输入流和输出流的read和write方法使用字符数组读写数据，即以字符为基本单位处理数据。

构造方法分别是：

FileReader(String filename);

FileReader(File filename);

FileWriter (String filename);

FileWriter (File filename);

FileWriter (String filename,boolean append);

FileWriter (File filename,boolean append);

10.2 读写文件的常用流

4. 缓冲流

- `BufferedReader`类和`BufferedWriter`的构造方法分别是：

`BufferedReader(Reader in);`

`BufferedWriter (Writer out);`

- `BufferedReader`流能够读取文本行，方法是`readLine()`。
`BufferedReader`和`BufferedWriter`类创建的对象称为缓冲输入、输出流，二者增强了读写文件的能力。需要注意的是二者的源和目的地必须是字符输入流和字符输出流

`FileReader inOne=new FileReader("studen.txt");` //创建字符输入流

`BufferedReader inTwo=new BufferedReader(inOne);` //创建缓冲输入流

`FileWriter tofile=new FileWriter("hello.txt");` //创建字符输出流

`BufferWriter out=new BufferWriter(tofile);` //创建缓冲输入流

10.2 读写文件的常用流

4. 缓冲流

- `readLine()` //读取一行
- `write(String s, int off, int len)` //把字符串s写到流中，参数off是s开始处的偏移量，len是写入的字符数量
- `newLine()` //写入一个回车符
- 可以把`BufferedReader`和`BufferedWriter`称为上层流，把它们指向的字符流称为底层流。

10.2 读写文件的常用流

4. 缓冲流

- Java采用缓存技术将上层流和底层流连接。底层字符输入流首先将数据读入缓存，BufferedReader流再从缓存读取数据；BufferedWriter流将数据写入缓存，底层字符输出流会不断地将缓存中的数据写入到目的地。
- 当BufferedWriter流调用flush()刷新缓存或调用close()方法关闭时，即使缓存没有益满，底层流也会立刻将缓存的内容写入目的地。

10.2 读写文件的常用流

example8_4_choice.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><BODY bgcolor=cyan><font size=2>
  <form action="helpReadFile" method="post" name="form">
    输入文件的路径(如:d:/2000):
    <input type="text" name="filePath" size=12>
    <br>输入文件的名字(如:Hello.java):
    <input type="text" name="fileName" size=9>
    <br><input type="submit" value="读取" name="submit">
  </form>
</font></BODY></HTML>
```

10.2 读写文件的常用流

<servlet>

<servlet-name>helpReadFile </servlet-name>

<servlet-class>myservlet.control.Example8_4_Servlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>helpReadFile </servlet-name>

<url-pattern>/helpReadFile</url-pattern>

</servlet-mapping>

10.2 读写文件的常用流

Example8_4_Bean.java

```
package mybean.data;  
  
public class Example8_4_Bean {  
    String filePath,fileName,fileContent;  
    long fileLength;  
    public void setFilePath(String str){filePath=str;}  
    public String getFilePath(){return filePath; }  
    public void setFileName(String str){fileName=str;}  
    public String getFileName(){return fileName; }  
    public void setFileContent(String str){fileContent=str;}  
    public String getFileContent(){return fileContent; }  
    public void setFileLength(long len){fileLength=len; }  
    public long getFileLength(){return fileLength; }  
}
```

10.2 读写文件的常用流

Example8_4_Servlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
```

```
    Example8_4_Bean fileBean=new Example8_4_Bean();
```

```
    request.setAttribute("fileBean",fileBean);
```

```
    String filePath=request.getParameter("filePath");
```

```
    String fileName=request.getParameter("fileName");
```

```
    fileBean.setFilePath(filePath); //将数据存储在fileBean中
```

```
    fileBean.setFileName(fileName);
```

```
    try{    File f=new File(filePath,fileName);
```

```
        long length=f.length();
```

```
        fileBean.setFileLength(length);
```

```
        FileReader in=new FileReader(f) ;
```

```
        BufferedReader inTwo=new BufferedReader(in);
```

```
        StringBuffer stringbuffer=new StringBuffer();
```

```
        String s=null;
```

```
        while ((s=inTwo.readLine())!=null)
```

```
            stringbuffer.append("\n"+s);
```

```
        String content=new String(stringbuffer);
```

```
        fileBean.setFileContent(content);
```

```
    }
```


10.2 读写文件的常用流

Example8_4_Servlet.java

```
catch(IOException exp){  
    fileBean.setFileContent("读取失败"+exp.toString());  
}  
RequestDispatcher dispatcher=  
    request.getRequestDispatcher("example8_4_showFile.jsp");  
dispatcher.forward(request, response);  
}
```

10.2 读写文件的常用流

example8_4_showFile.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="fileBean"
  type="mybean.data.Example8_4_Bean" scope="request"/>
<HTML><body bgcolor=#EEFFAE><font size=2>
  文件的位置: <jsp:getProperty name="fileBean"
    property="filePath"/>,
  文件的名字: <jsp:getProperty name="fileBean"
    property="fileName"/>,
  文件的长度: <jsp:getProperty name="fileBean"
    property="fileLength"/> 字节。
  <br>文件的内容:
  <br><TextArea rows="6" cols="60">
    <jsp:getProperty name="fileBean" property="fileContent"/>
  </TextArea>
</font></body></HTML>
```

10.3 RandomAccessFile类

- RandomAccessFile类创建的流与前面的输入、输出流不同。RandomAccessFile类既不是输入流类InputStream类的子类，也不是输出流类OutputStream类的子类。
- 习惯上，仍然称RandomAccessFile类创建的对象为一个流。RandomAccessFile流的指向既可以作为源也可以作为目的地。换句话说，当想对一个文件进行读写操作时，可以创建一个指向该文件的RandomAccessFile流，这样既可以从这个流读取文件的数据，也可以通过这个流向文件写入数据。

10.3 RandomAccessFile类

1. 构造方法

- RandomAccessFile类的两个构造方法:

RandomAccessFile(String name,String mode)

RandomAccessFile(File file,String mode)

- 参数name用来确定一个文件名，参数file是一个File对象，给出创建的流的源（也是流的目的地）。参数mode取“r”（只读）或“rw”（可读写），决定创建的流对文件的访问权利。

10.3 RandomAccessFile类

2. 读写方法

- `readLine()` 从文件中读取一个文本行
- `readUTF()` 从文件中读取一个UTF字符串
- `seek(long a)` 定位当前流在文件中的读写的位置
- `write(byte b[])` 写**b.length**个字节到文件
- `writeDouble(double v)` 向文件写入一个双精度浮点值
- `writeInt(int v)` 向文件写入一个int值
- `writeUTF(String s)` 写入一个UTF字符串
- `getFilePointer()` 获取当前流在文件中的读写的位置

10.4 文件上传

- 用户通过一个JSP页面上传文件给服务器时，该JSP页面必须含有File类型的表单，并且表单必须将ENCTYPE的属性值设成multipart/form-data。

```
<form action="接受上传文件的页面"  
method="post" ENCTYPE="multipart/form-data"  
<input type="File" name="picture" >  
</form>
```

10.4 文件上传

example8_5.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body>
  <P>选择要上传的文件: <BR>
    <form action="example8_5_accept.jsp" method="post"
      ENCTYPE="multipart/form-data">
      <input type=FILE name="boy" size="38">
      <br><input type="submit" name ="g" value="提交">
    </form>
  </body></HTML>
```

10.4 文件上传

example8_5_accpet.jsp

```
<%@ page contentType="text/html;charset=GB2312" %>
<%@ page import ="java.io.*" %>
<HTML><body>
    <%try{ InputStream in=request.getInputStream();
        File dir=new File("C:/1000");
        dir.mkdir();
        File f=new File(dir,"B.txt");
        FileOutputStream o=new FileOutputStream(f);
        byte b[]=new byte[1000];
        int n;
        while((n=in.read(b))!=-1)
            o.write(b,0,n);
        o.close();
        in.close();
        out.print("文件已上传");
    }
    catch(IOException ee){out.print("上传失败"+ee);}
%> </body></HTML>
```


10.4 文件上传

例子6中，通过输入、输出流技术获取文件的内容，即去掉表单的信息。

1. 不同用户的session对象互不相同这一特点，将用户提交的全部信息首先保存成一个临时文件，该临时文件的名字是用户的session对象的id.
2. 然后读取该临时文件的第2行，因为这一行中含有用户上传的文件的名字，再获取第4行结束的位置，以及倒数第6行结束的位置，因为这两个位置之间的内容是上传文件的内容，然后将这部分内容存入文件，该文件的名字和用户上传的文件的名字保持一致
3. 最后删除临时文件。

10.4 文件上传

example8_6.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import ="java.io.*" %>
<jsp:useBean id="fileBean"
  class="mybean.data.Example8_6_Bean" scope="request"/>
<HTML><body>
  <P>选择要上传的文件: <BR>
  <form action="upFile" method="post" ENCTYPE="multipart/form-
    data">
    <input type=FILE name="boy" size="45">
    <br> <input type="submit" name ="boy" value="提交">
  </form>
  <br> 上传的文件名字:
  <jsp:getProperty name="fileBean" property="fileName"/>
  <br> 上传反馈:
  <jsp:getProperty name="fileBean" property="mess"/>
```

10.4 文件上传

example8_6.jsp

```
<%  
    String name=fileBean.getFileName();  
    boolean boo =name.endsWith(".jpg");  
    boo = boo || name.endsWith(".gif");  
    if(boo) {  
        %>    <image src = "image/<%=name%>" width=200 height  
            =200><%=name %></image>  
        <% }  
        else {  
            %>    <%=name %>  
            <% }  
        %>  
    </body></HTML>
```

10.4 文件上传

Example8_6_Bean.java

```
package mybean.data;
public class Example8_6_Bean {
    String fileName="";
    String mess="";
    public void setFileName(String str){
        fileName=str;
    }
    public String getFileName(){
        return fileName;
    }
    public void setMess(String str){
        mess=str;
    }
    public String getMess(){
        return mess;
    }
}
```

10.4 文件上传

<servlet>

<servlet-name>upFile</servlet-name>

<servlet-class>myservlet.control.Example8_6_Servlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>upFile</servlet-name>

<url-pattern>/upFile</url-pattern>

</servlet-mapping>

10.4 文件上传

Example8_6_Servlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{
    request.setCharacterEncoding("gb2312");
    Example8_6_Bean fileBean=new Example8_6_Bean();
    request.setAttribute("fileBean",fileBean);
    String fileName=null;
    HttpSession session=request.getSession(true);
    try{
        //用客户的session对象的Id建立一个临时文件
        String tempFileName=(String)session.getId();
        String webDir = request.getContextPath();
        webDir = webDir.substring(1);
        File f= new File("");
        String path = f.getAbsolutePath();
        int index = path.indexOf("bin");
        String tomcatDir = path.substring(0,index);//tomcat的安装目录
        File dir=new File(tomcatDir+"/webapps/"+webDir+"/image");
        dir.mkdir();
```

10.4 文件上传

Example8_6_Servlet.java

```
//建立临时文件f1
File f1=new File(dir,tempFileName);
FileOutputStream o=new FileOutputStream(f1);
//将客户上传的全部信息存入f1
InputStream in=request.getInputStream();
byte b[]=new byte[10000];
int n;
while( (n=in.read(b))!=-1){
    o.write(b,0,n);
}
o.close();
in.close();
//读取临时文件f1，从中获取上传文件的名字和上传文件的内容
RandomAccessFile randomRead=new RandomAccessFile(f1,"r");
//读出f1的第2行，析取出上传文件的名字
int second=1;
String secondLine=null;
```

10.4 文件上传

Example8_6_Servlet.java

```
while(second<=2) {  
    secondLine=randomRead.readLine();  
    second++;  
}  
//获取f1中第2行中"filename"之后 "="出现的位置:  
int position=secondLine.lastIndexOf("=");  
//客户上传的文件的名字是  
fileName=secondLine.substring(position+2,secondLine.length()-1);  
randomRead.seek(0); //再定位到文件f1的开头  
//获取第4行回车符号的位置  
long forthEndPosition=0;  
int forth=1;  
while((n=randomRead.readByte())!=-1&&(forth<=4)){  
    if(n=='\n'){  
        forthEndPosition=randomRead.getFilePointer();  
        forth++;  
    }  
}
```


10.4 文件上传

Example8_6_Servlet.java

//根据客户上传文件的名称，将该文件存入磁盘

```
byte cc[]=fileName.getBytes("ISO-8859-1");
```

```
fileName=new String(cc);
```

```
File f2= new File(dir,fileName);
```

```
RandomAccessFile randomWrite=new RandomAccessFile(f2,"rw");
```

//确定出文件f1中包含客户上传的文件的内容的最后位置，即倒数第6行

```
randomRead.seek(randomRead.length());
```

```
long endPosition=randomRead.getFilePointer();
```

```
long mark=endPosition;
```

```
int j=1;
```

```
while((mark>=0)&&(j<=6)) {
```

```
    mark--;
```

```
    randomRead.seek(mark);
```

```
    n=randomRead.readByte();
```

```
    if(n=='\n'){
```

```
        endPosition=randomRead.getFilePointer();
```

```
        j++;
```

```
    }
```

```
}
```

10.4 文件上传

Example8_6_Servlet.java

```
//将randomRead流指向文件f1的第4行结束的位置
    randomRead.seek(forthEndPosition);
    long startPoint=randomRead.getFilePointer();
//从f1读出客户上传的文件存入f2（读取第4行结束位置和倒数第6行之间的内容）
    while(startPoint<endPosition-1){
        n=randomRead.readByte();
        randomWrite.write(n);
        startPoint=randomRead.getFilePointer();
    }
    randomWrite.close();
    randomRead.close();
    fileBean.setMess("上传成功");
    fileBean.setFileName(fileName);
    f1.delete(); //删除临时文件
} catch (Exception ee) {
    fileBean.setMess("没有选择文件或上传失败"); }
RequestDispatcher dispatcher=
request.getRequestDispatcher("example8_6.jsp");
dispatcher.forward(request, response);
}
```

10.5 文件下载

- JSP 内置对象 `response` 调用方法 `getOutputStream()` 可以获取一个指向用户的输出流，服务器将文件写入这个流，用户就可以下载这个文件了。
- 当提供下载功能时，应当使用 `response` 对象向用户发送 HTTP 头信息，这样用户的浏览器就会调用相应的外部程序打开下载的文件，`response` 调用 `setHeader` 方法添加下载头的格式如下：

```
response.setHeader("Content-  
disposition","attachment;filename="下载文件名");
```

10.5 文件下载

example8_7.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<HTML><body bgcolor=#FFBBFE>
<form action="loadFile" method=post name=form>
  选择要下载的文件: <br>
  <Select name="filePath" size=3>
    <Option Selected value="d:/2000/E.java">E.java
    <Option value="d:/2000/first.jsp">first.jsp
    <Option value="d:/2000/book.zip">book.zip
    <Option value="d:/2000/A.txt">A.txt
  </Select>
  <br><INPUT TYPE="submit" value="提交" >
</form>
</body></HTML>
```

10.5 文件下载

```
<servlet>
  <servlet-name>loadFile</servlet-name>
  <servlet-class>myservlet.control.Example8_7_Servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>loadFile</servlet-name>
  <url-pattern>/loadFile</url-pattern>
</servlet-mapping>
```

10.5 文件下载

Example8_7_Servlet.java

```
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{
    request.setCharacterEncoding("gb2312");
    String filePath=request.getParameter("filePath");
    String fileName=filePath.substring(filePath.lastIndexOf("/") + 1);
    response.setHeader("Content-
disposition","attachment;filename="+fileName);
    try{ //读取文件,并发送给用户下载:
        File f=new File(filePath);
        FileInputStream in=new FileInputStream(f);
        OutputStream out=response.getOutputStream();
        int n=0;
        byte b[]=new byte[500];
        while((n=in.read(b))!=-1)
            out.write(b,0,n);
        out.close();
        in.close();
    }
    catch(Exception exp){}
}
```

小结

- 输入流的指向称为源，程序从指向源的输入流中读取源中的数据。而输出流的指向是数据要去的目的地，程序通过向输出流中写入数据把信息送往目的地。
- FileInputStream和FileReader流都顺序地读取文件，只要不关闭流，每次调用read方法就顺序地读取源中其余的内容，直到源的末尾或流被关闭。二者的区别是，FileInputStream流以字节（byte）为单位读取文件；FileReader流以字符（char）为单位读取文件。

小结

- `FileOutputStream`流和`FileWriter` 顺序地写文件，只要不关闭流，每次调用`writer`方法就顺序地向输出流写入内容，直到流被关闭。二者的区别是，`FileOutputStream`流以字节（byte）为单位写文件；`FileWriter`流以字符（char）为单位写文件。
- `RandomAccessFile`流的指向既可以作为源也可以作为目的地，在读写文件时可以调用`seek`方法改变读写位置。