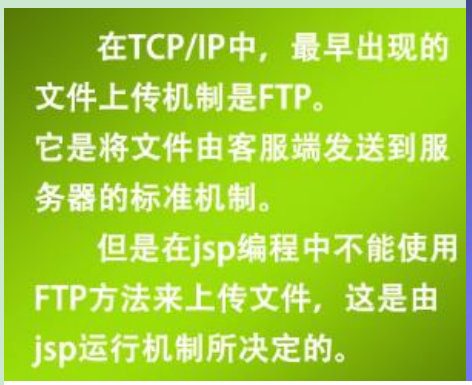
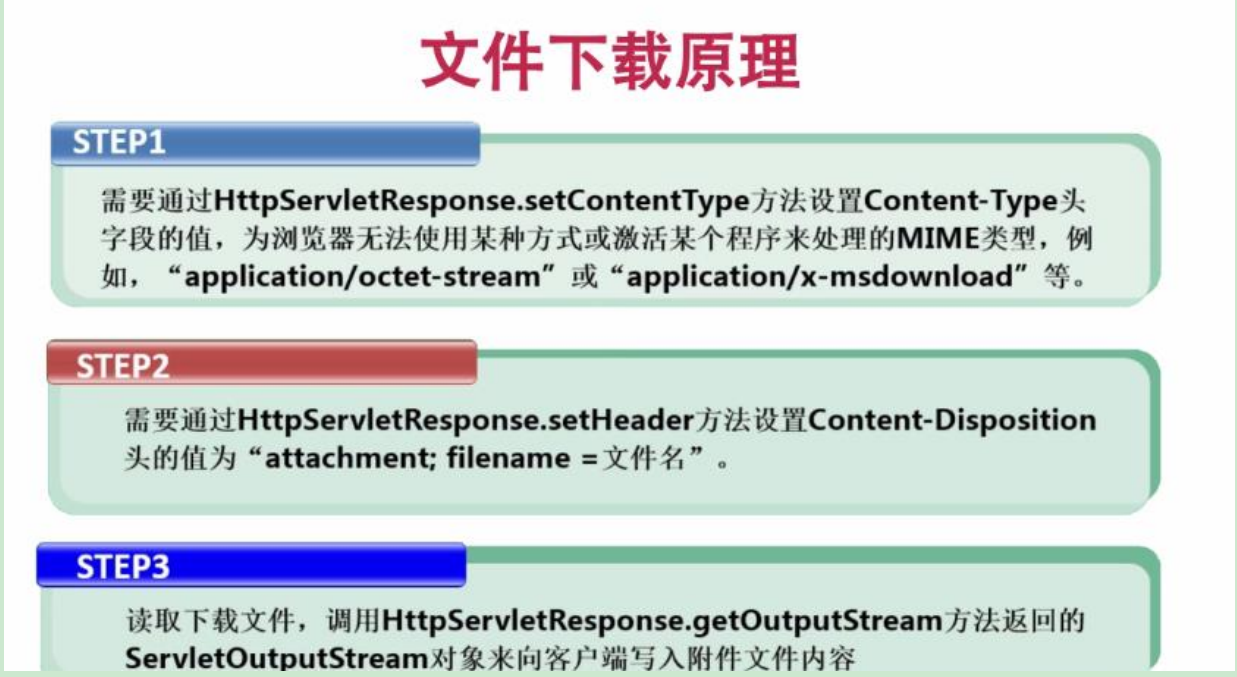
文件上传下载原理

文件上传的本质就是将文件序列化成字节流，然后通过网络协议（比如：HTTP/TCP），在网络中传输，到达指定的目标机器之后再将对应的字节流解析成对应的文件内容。
JavaWeb编程中的文件上传和下载的功能，主要通过浏览器来完成前端表单提交的内容的序列化工作，然后，后端服务器端的程序来反序列化。下载的过程则相反。





文件上传下载的原理

1：目前是通过浏览器将表单中的文件内容以二进制流的形式传递到后台服务器中
2：要求在表单中使用
1）file标签
2）表单的发送请求的方法是post
3）表单的enctype属性值是"multipart/form-data"

文件下载的原理：
1：服务器端提供文件流
2：服务器端的响应信息，告诉浏览器传递过来的字节流是一个文件
3：浏览器来接收对应的字节流不，并且将其反序列化为对应的文件的内容
下面我们是在JSP页面中使用到的js文件
全是jq，处理起来的确是非常的方便

```
$(function(){
$(".thumbs a").click(function(){
var largePath  = $(this).attr("href");
var largeAlt = $(this).attr("title");
$("#largeImg").attr({
src : largePath,
alt : largeAlt
      });
return false;
});

$("#myfile").change(function(){
$("#previewImg").attr("src","file:///" + $("#myfile").val());
});

var la = $("#large");
la.hide();

$("#previewImg").mousemove(function(e){
      la.css({
top : e.pageY,
left : e.pageX
}).html('<img src = "' + this.src + '" />').show();
}).mouseout(function(){
      la.hide();
});
});
```

其实只是需要一些上传的IO流来处理就好
其中使用了form表单 关于form表单有个上传的选项选定好了，可以选择提交这个选项，与servlet的后台处理，如果在jsp页面处理的话也可以，
servlet

```
//从request当中获取流信息
InputStream fileSource =req.getInputStream();
String  tempFileName="E://tempFile";

//tempFile指向临时文件
File tempFile=new File(tempFileName);
//outputSteam文件输出流指向这个临时文件
FileOutputStream outputStream=new FileOutputStream(tempFile);
byte b[]=new byte[1024];
int n;
while(( n = fileSource.read(b)) != -1){
    outputStream.write(b, 0, n);
}
//关闭输出流, 输入流
outputStream.close();
fileSource.close();
System.out.println("已接受到请求");
```

这是dopost的servlet处理方式

这里介绍了一个工具，是IE上的
是HttpWatch

这是上传与显示的Servlet
这种是最复杂的，首先，要获取流，其中我们主要用到的是RamdomAccessFile这个类，用来读取上传的文件。，用的是readbyte这个读取字节的，这样子我们就通过俩个指针来获取开始与结束的位置，，然后主要是通过上面的指针来来写入内容，，然后保存到指定的路径，然后该关的关，提示的提示一下，跳转回来，结束了（这里的指针来回跳转是比较麻烦的）

```
public void doPost(HttpServletRequest req, HttpServletResponse resp)
```

```
public void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
//从request当中获取流信息
InputStream fileSource = req.getInputStream();
String tempFileName = "E:/tempFile";
//tempFile指向临时文件
File tempFile = new File(tempFileName);
//outputStram文件输出流指向这个临时文件
FileOutputStream outputStream = new FileOutputStream(tempFile);
    byte b[] = new byte[1024];
    int n;
    while(( n = fileSource.read(b)) != -1){
        outputStream.write(b, 0, n);
}
//关闭输出流、输入流
outputStream.close();
fileSource.close();

//获取上传文件的名称
RandomAccessFile randomFile = new RandomAccessFile(tempFile,"r");
randomFile.readLine();
String str = randomFile.readLine();
    int beginIndex = str.lastIndexOf("\\") + 1;
    int endIndex = str.lastIndexOf("\"");
String filename = str.substring(beginIndex, endIndex);
System.out.println("filename:" + filename);

//重新定位文件指针到文件头
randomFile.seek(0);
    long startPosition = 0;
    int i = 1;
//获取文件内容 开始位置
while(( n = randomFile.readByte()) != -1 && i <=4){
if(n == '\n'){
            startPosition = randomFile.getFilePointer();
i ++;
}
    }
    startPosition = randomFile.getFilePointer() -1;
//获取文件内容 结束位置
randomFile.seek(randomFile.length());
    long endPosition = randomFile.getFilePointer();
    int j = 1;
    while(endPosition >=0 && j<=2){
        endPosition--;
        randomFile.seek(endPosition);
        if(randomFile.readByte() == '\n'){
            j++;
}
    }
    endPosition = endPosition -1;

//设置保存上传文件的路径
String realPath = getServletContext().getRealPath("/") + "images";
File fileupload = new File(realPath);
    if(!fileupload.exists()){
        fileupload.mkdir();
}
    File saveFile = new File(realPath,filename);
RandomAccessFile randomAccessFile = new RandomAccessFile(saveFile,"rw");
//从临时文件当中读取文件内容（根据起止位置获取）
randomFile.seek(startPosition);
    while(startPosition < endPosition){
        randomAccessFile.write(randomFile.readByte());
        startPosition = randomFile.getFilePointer();
}
//关闭输入输出流、删除临时文件
randomAccessFile.close();
randomFile.close();
tempFile.delete();

req.setAttribute("result", "上传成功！");
RequestDispatcher dispatcher = req.getRequestDispatcher("jsp/01.jsp");
dispatcher.forward(req, resp);
}
```

这是下载的servlet
下载的非常简单，就是设置头部信息，然后下载

```
//获取文件下载路径
String path = getServletContext().getRealPath("/") + "images/";
String filename = req.getParameter("filename");
File file = new File(path + filename);
if(file.exists()){
//设置相应类型application/octet-stream
resp.setContentType("application/x-msdownload");
//设置头信息
resp.setHeader("Content-Disposition", "attachment;filename=\"" + filename + "\"");
InputStream inputStream=new FileInputStream(file);
ServletOutputStream  outputstream=resp.getOutputStream();
    byte b[]=new byte[1024];
   int n;
   while ((n=inputStream.read(b))!=-1){
      outputstream.write(b,0,n);
}
//关闭流、释放资源
outputstream.close();
inputStream.close();


}else{
   req.setAttribute("errorResult", "文件不存在下载失败！");
RequestDispatcher dispatcher = req.getRequestDispatcher("jsp/01.jsp");
dispatcher.forward(req, resp);
}
```

SmartUpload简介，
1：可以用它限制文件的类型、大小、后缀等等
2：框架、组件等等一般都是以JAR的形式存在的引用起来相当简单
3：SmartUpload能够实现文件的批量上传
下面贴代码：

```
String filePath=getServletContext().getRealPath("/")+"images";
File file=new File(filePath);
    if(!file.exists()){
        file.mkdir();
}
    SmartUpload su = new SmartUpload();
// 初始化对象
su.initialize(getServletConfig(),req,resp);
//设置上传文件的大小
su.setMaxFileSize(1024*1024*10);
//设置所有文件的大小
su.setTotalMaxFileSize(1024*1024*100);
//设置允许上传文件类型
su.setAllowedFilesList("txt,jpg,gif");
String result;
    try {
//设置不需要上传的文件类型
su.setDeniedFilesList("rar,jsp,js");
result="上传成功!";
//上传文件
int count=su.save(filePath);
System.out.println("上传成功了 "+count+"个文件");
} catch (Exception e) {
        result="上传失败";
        if(e.getMessage().indexOf("1015")!=-1){
            result="上传失败：上传类型不正确";
}else if(e.getMessage().indexOf("1010")!=-1){
            result="上传失败：上传类型不正确";
}else if(e.getMessage().indexOf("1105")!=-1){
            result="上传失败：上传文件的大小小于允许的最大值";
}
else if(e.getMessage().indexOf("1110")!=-1){
            result="上传失败：上传文件的总大小小于允许的最大值";
}

        e.printStackTrace();
```

```
}

for (int i=0;i<su.getFiles().getCount();i++){
        com.jspsmart.upload.File tempfile=su.getFiles().getFile(i);
System.out.println("----------------------");
System.out.println("表单中的name的值"+tempfile.getFieldName());
System.out.println("上传文件名"+tempfile.getFieldName());
System.out.println("上传文件的大小"+tempfile.getSize());
System.out.println("上传文件的扩展名"+tempfile.getFileExt());
System.out.println("上传文件的全名"+tempfile.getFilePathName());
System.out.println("----------------------");
}
    req.getRequestDispatcher("jsp/02.jsp").forward(req,resp);
}
```

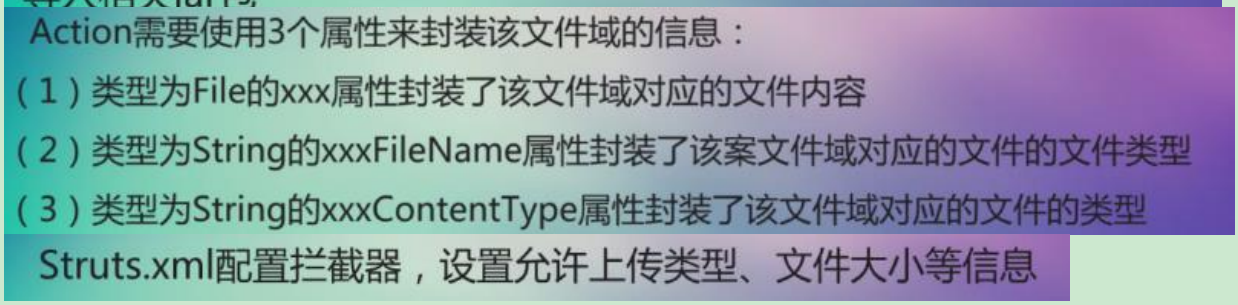
```
protected void doPost(HttpServletRequest request, HttpServletResponse response) thro
ws ServletException, IOException {
    String filename = request.getParameter("filename");
SmartUpload su= new SmartUpload();
su.initialize(getServletConfig(),request,response);
su.setContentDisposition(null);
    try{
        su.downloadFile("/images/"+ filename);
}catch (Exception e){
        e.printStackTrace();
}

}
```

```
//常见异常
if(m_deniedFilesList.contains(s5))
    //1015 文件拓展名禁止上传
    throw new SecurityException("The extension of the file is denied to be uploaded (1015).");
if(!m_allowedFilesList.isEmpty() && !m_allowedFilesList.contains(s5))
    //1010 文件拓展名不是允许的上传类型
    throw new SecurityException("The extension of the file is not allowed to be uploaded (1010).");
if(m_maxFileSize > 0L && (long)((m_endData - m_startData) + 1) > m_maxFileSize)
    //1105 大小超过单个文件允许的最大值
    throw new SecurityException("Size exceeded for this file : " + s4 + " (1105).");
l += (m_endData - m_startData) + 1;
if(m_totalMaxFileSize > 0L && l > m_totalMaxFileSize)
    //1110 上传文件总大小超过所有文件总大小允许的最大值
    throw new SecurityException("Total File Size exceeded (1110).");
```

批量下载的代码如下：
这是没有涉及Smart这个Jar包，

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws Servl
etException, IOException {
resp.setContentType("application/x-msdownload");
resp.setHeader("Content-Disposition", "attachment;filename=test.zip");
String path = getServletContext().getRealPath("/") + "images/";
String[] filenames=req.getParameterValues("filename");
String str="";
String rt="\r\n";
ZipOutputStream zos = new ZipOutputStream(resp.getOutputStream());
    for (String filename:filenames){
        str+=filename+rt;
File file=new File(path+filename);//文件的位置
zos.putNextEntry(new ZipEntry(filename));
FileInputStream fis = new FileInputStream(file);
        byte b[]=new byte[1024];
        int n=0;
        while ((n=fis.read(b))!=-1){
            zos.write(b,0,n);
}
        zos.flush();
        fis.close();
}
}
```

Struts2的上传与下载



从Struts2中设置上传下载的话，主要是通过配置一个Struts2.xml的文件来在里面进行配置就好了，
非常的智能，可以说得上非常的智能，我们配置好struts2的框架后，声明一个action，在这里我们定义好我们在struts2.xml中需要的字段
这几个字段在struts的官方文档中都有说明
inputName就是配置我们返回出来的对应的字段名，就是我们需要下载的具体处理的地方
这个inputPath就是很明显，就是下载的地址，
其中底下的filename的属性中配置的就是我们下载的时候这个文件的名字
这里我们采用的是自动获取action中的配置的，用的是OGNL表达式，来自动获取，这样子好改
同时，还有也可以在jsp页面申请下载的时候，我们申请的时候指定下载文件
这样子的话，我们也直接在action类中直接声明这么一样的名字的一个名字的字段就好了，这样子我们上面定义的inputpath就不用了，但是struts就是这么智能，也帮我们拦截了，也管了，不需要其他配置，

```
<a href="download.action?filename=img3-lg.jpg"></a>
```

```
<param name="inputPath">/images/img2-lg.jpg</param>
```

```
<param name="inputName">inputStream</param>
<param name="contentDisposition">attachment;filename="${downloadFilename}"</param>
```

比如这俩个，struts2,就会在自动的帮我们找到，配置好，我们只是要把字段名起对了就ok，另外包括其他的配置也都是在xml中进行配置的，
下面贴代码：

```
<constant name="struts.enable.DynamicMethodInvocation" value="false"></constant>
 <constant name="struts.devMode" value="true"></constant>
<constant name="struts.custom.i18n.resources" value="app"></constant>
 <package name="default" namespace="/" extends="struts-default">
  <action name="upload" class="com.imooc.servlet.FileUploadAction">
        <result>/jsp/03.jsp</result>
        <result name="input">/jsp/error.jsp</result>
<!--配置拦截器的信息，上传文件的类
        型与大小-->

<interceptor-ref name="fileUpload">
             <param name="allowedTypes">image/bmp,image/x-png,image/gif,image/jpeg<
/param>
              <param name="maximumSize">2M</param>
       </interceptor-ref>
    <interceptor-ref name="defaultStack"></interceptor-ref>
  </action>
    <action name="download" class="com.imooc.servlet.FileDownAction">
        <param name="inputPath">/images/img2-lg.jpg</param>
            <result name="success" type="stream" >
             <param name="contentType">appliaction/octet-stream</param>
             <param name="inputName">inputStream</param>
```

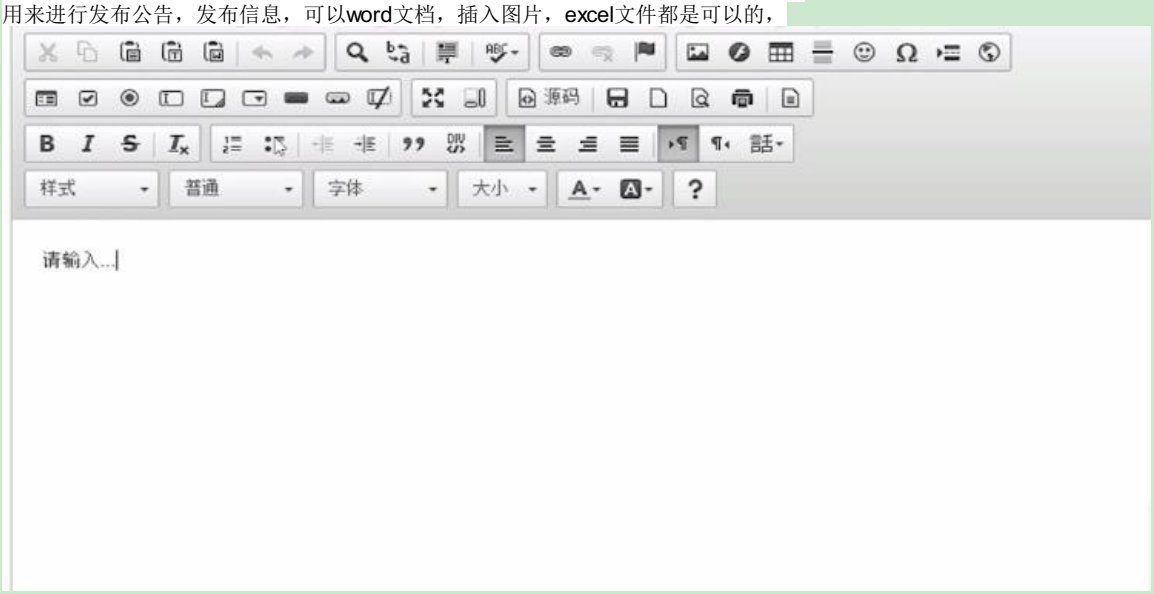
```
            <param name="contentDisposition">attachment;filename="${downloadFilena
me}"</param>
            <param name="bufferSize">8192</param>
        </result>
    </action>
```

这就是上传与下载，我们可以在xml中指定好下载的东西，这样真的是非常的简单了，如果指定好的话

```
public InputStream getInputStream()   {
return  ServletActionContext.getServletContext().getResourceAsStream(inputPath);
}
```

直接把我们需要当前路径配置一下，加上我们上面也在xml中配置我们需要下载的文件字段就行了，

下面这个是富文本编辑器的使用，这个是跟其他的上传下载是不一样的东西，这个是
用来进行发布公告，发布信息，可以word文档，插入图片，excel文件都是可以的，



这个是富文本编辑器的具体代码

如下：

```
public class Ckaction extends ActionSupport {

private  String editor;

    public String getEditor() {
return editor;
}

public void setEditor(String editor) {
this.editor = editor;
}

@Override
public String execute() throws Exception {
        System.out.println("富文本信息: "+editor);

        int beginIndex=editor.indexOf("<a");
        while(beginIndex!=-1) {
int endIndex = editor.indexOf(">", beginIndex) + 1;
String beginStr = editor.substring(0, beginIndex);
String endStr = editor.substring(endIndex);
String str = editor.substring(beginIndex, endIndex);
String filename = str.substring(str.lastIndexOf("/") + 1, str.lastIndexOf("\""));
String replace = "<a href=download.action?filename=" + filename + " /a>";
editor = beginStr + replace + endStr;
beginIndex=editor.indexOf("<a",endIndex);
}
return SUCCESS;
}
```

```
<action name="cktest" class="com.imooc.action.Ckaction">
    <result>show.jsp</result>
</action>
```

```
<action name="download" class="com.imooc.action.FileDownloadAction">
    <param name="inputPath">/images/img2-lg.jpg</param>
    <result name="success" type="stream">
        <param name="contentType">application/octet-stream</param>
        <param name="inputName">inputStream</param>
        <param name="contentDisposition">attachment;filename="${downloadFileName}"</
param>
        <param name="bufferSize">8192</param>
    </result>
</action>
```

```
<script type="application/javascript" src="<%=request.getContextPath()%>/ckeditor/ck
editor.js"></script>
  <script type="application/javascript" src="<%=request.getContextPath()%>/ckfinder/
ckfinder.js"></script>
  <title>$Title$</title>
</head>
<body>

<form action="cktest.action" method="post">
  <textarea rows="10" cols="80" id="editor" name="editor"
class="ckeditor"
>请输入...</textarea>
  <input type="submit" value="保存"/>
```

然后加入我们需要的 

文件
还有jar包，好的，我又失败了，等以后尝试

文件上传下载Java web
1.文件上传下载原理，通过为表单元素设置Method="potist",enctype="mulpart/form- data"属性，让表单提交数据以二进制编码方式提交，在接受此请求时用二进制流来获取内容
text/plain：主要适用于直接通过表单发送邮件
（图片文件类型：jsp,css,js)
示例：<link rel="stylesheet""type="text/css"herf="css/common.css"/>
2.图片浏览：
a.创建upload servlet
b.Form的method设置为Post
c.保存上传文件：record
获取request当中的流信息，保存至临时文件，从临时文件中得到长传文件名，及文件内容起止位置，读取上传文件内容，保存至本地。
3.文件下载
a.通过超链接方式发起文件下载请求
b.配置Web.xml创建Download Servlet
c.后台Servlet,设置响应类型及响应头输出流写入文件内容
4.Struts 2
5.富文本编辑器
该方法是从ServletInputstream流中读一行 到指定的byte数组,为了保证能够容纳一行,该byte[]b不应该小于256,重写的readLine中,调用了一个成员变量len为，实际读到的字节数(有的行不满256),则在文件内容写入时，应该从byte数组中写入这个len长度的字节而不是整个byte的长度,但重写的这个方法返回的是String以便分析实际内容,不能返回len,所以把len设为成员变量,在每次读操作时，把实际长度赋给它。
也就是说在处理到文件的内容时，数据既要以String形式返回以便分析开始和结束标记,又要同时以[byte]的形式写到文件输出流中.