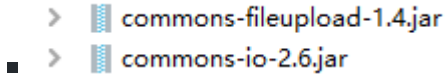
01_文件上传

- 本质就是将一台电脑中的文件根据网络协议通过io流传递到另外一台电脑(服务器)上。
- 文件上传三要素
  - 表单提交方式为post
  - 表单中需要文件上传项
  - enctype=multipart/form-data
- 文件上传代码实现
  - 1,导入jar包

  - 2,文件上传项
  - 3,编写Servlet
    - 创建磁盘文件项工厂对象DiskFileItemFactory
    - 创建核心解析类ServletFileUpload
    - 解析请求，获取到所有的文件项

```
DiskFileItemFactory diskFileItemFactory = new DiskFileItemFactory();
ServletFileUpload servletFileUpload = new
ServletFileUpload(diskFileItemFactory);
try {
    List<FileItem> fileItems = servletFileUpload.parseRequest(request);
    for (FileItem fileItem : fileItems) {
        if (fileItem.isFormField()){
            String desc = fileItem.getString();
        }else {
            InputStream inputStream = fileItem.getInputStream();
            BufferedInputStream bis = new BufferedInputStream(inputStream);
            String contextPath =
request.getServletContext().getRealPath("upload");
            File contextFile = new File(contextPath);
            if (!contextFile.exists()){
                contextFile.mkdir();
            }
            String path = contextPath+File.separator+fileItem.getName();
            BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(path));
            byte[] bys = new byte[8192];
            int len = -1;
            while((len = bis.read(bys)) != -1){
                bos.write(bys,0,len);
            }
            bis.close();
            bos.close();

        }
    }
} catch (Exception e) {
```

```
            e.printStackTrace();
        }
```

02_文件上传API

- ServletFileUpload:核心解析类
  - parseRequest(HttpServletRequest request):解析请求，并获取相关文件项
  - setHeaderEncoding(String encoding):解决中文文件名乱码
- FileItem:文件项
  - boolean isFormField()
    - 返回为true,普通字段。返回为false，就是文件。
  - String getFieldName()
    - 获取表单字段
  - String getString(String encoding)
    - 根据指定编码格式获取字段值
  - String getName()
    - 获取上传文件名称
  - InputStream getInputStream()
    - 获取上传文件对应的输入流
- 上传文件名称重复问题

03_文件上传结合数据库

04_文件下载

- 本质就是将一台电脑(服务器)中的文件根据网络协议通过io流传递到另外一台电脑上。
- 文件下载两种形式
  - 超链接
    - 如果浏览器支持这个格式的文件.可以在浏览器中打开.如果浏览器不支持这个格式的文件才会提示下载.
  - 手动编写代码的方式下载
- 手动编写代码实现下载
  - 设置媒体类型
  - 设置下载窗口
  - 开始读写

```
String fileName = "ceshi.txt";
String realPath = request.getServletContext().getRealPath("/download"+
File.separator+fileName);
BufferedInputStream bis = new BufferedInputStream(new FileInputStream(realPath));
//设置媒体类型
response.setContentType(request.getServletContext().getMimeType(fileName));
//设置下载窗口
response.setHeader("Content-Disposition","attachment;filename="+fileName);
byte[] bys = new byte[8192];
int len = -1;
```

```
BufferedOutputStream bos = new BufferedOutputStream(response.getOutputStream());
//开始读写
while((len = bis.read(bys)) != -1){
    bos.write(bys,0,len);
}
bis.close();
bos.close();
```

- 解决下载文件名中文乱码问题
  - 设置窗口中的文件名称

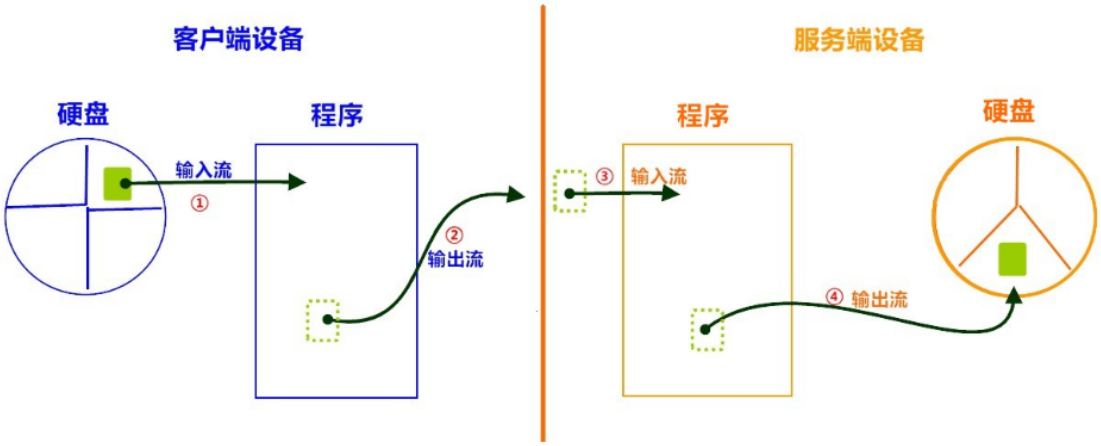
```
String newFileName = null;
String userAgent = request.getHeader("User-Agent");
if("Chrome".equals(userAgent)){
    //谷歌
    newFileName = URLEncoder.encode(fileName, "utf-8");
}else {
    newFileName = base64EncodeFileName(fileName);
}
//设置下载窗口
response.setHeader("Content-Disposition","attachement;filename="+newFileName);

//base64编码
public String base64EncodeFileName(String fileName) {
    BASE64Encoder base64Encoder = new BASE64Encoder();
    try {
        return "=?UTF-8?B?"
                + new String(base64Encoder.encode(fileName
                .getBytes("UTF-8"))) + "?=";
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}
```

05_自定义文件上传

- A.文件上传分析图解
  - 【客户端】输入流，从硬盘读取文件数据到程序中。
  - 【客户端】输出流，写出文件数据到服务端。
  - 【服务端】输入流，读取文件数据到服务端程序。

- 【服务端】输出流，写出文件数据到服务器硬盘中。



- B.客户端实现

```
Socket socket = null;
try {
    socket = new Socket(InetAddress.getByName("127.0.0.1"), 6666);
    OutputStream outputStream = socket.getOutputStream();
    BufferedOutputStream bos = new BufferedOutputStream(outputStream);
    BufferedInputStream bis = new BufferedInputStream(new
FileInputStream("C:\\Users\\qiuzhiwei\\Desktop\\a.pptx"));
    byte[] bys = new byte[8192];
    int len = -1;
    while ((len = bis.read(bys)) != -1) {
        bos.write(bys, 0, len);
    }
    System.out.println("已上传");
} catch (Exception e) {
    e.printStackTrace();
} finally {

    try {
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

- C.服务端实现

```
ServerSocket serverSocket = null;
try {
     serverSocket = new ServerSocket(6666);
    Socket socket = serverSocket.accept();
    InputStream inputStream = socket.getInputStream();
    BufferedInputStream bis = new BufferedInputStream(inputStream);
    BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream("C:\\Users\\qiuzhiwei\\Desktop\\copy.pptx"));
    byte[] bys = new byte[8192];
```

```
            int len = -1;
            while ((len = bis.read(bys)) != -1) {
                bos.write(bys, 0, len);
            }
            System.out.println("已保存");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                serverSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

- D.优化
    - a. **文件名称写死的问题**
    - 服务端，保存文件的名称如果写死，那么最终导致服务器硬盘，只会保留一个文件，建议使用系统时间优化，保证文件名称唯一，代码如下：

```
FileOutputStream fis = new FileOutputStream(System.currentTimeMillis()+".jpg") // 文件名称
BufferedOutputStream bos = new BufferedOutputStream(fis);
```

    - b. **循环接收的问题**

- 服务端，指保存一个文件就关闭了，之后的用户无法再上传，这是不符合实际的，使用循环改进，可以不断的接收不同用户的文件，代码如下：

```
```java
// 每次接收新的连接,创建一个Socket
while（true）{
    Socket accept = serverSocket.accept();
    ......
}
```

* 3. **效率问题**

* 服务端，在接收大文件时，可能耗费几秒钟的时间，此时不能接收其他用户上传，所以，使用多线程技术优化，代码如下：
```
while（true）{
    Socket accept = serverSocket.accept();
    // accept 交给子线程处理.
    new Thread(() -> {
        ......
        InputStream bis = accept.getInputStream();
        ......
    }).start();
```
```

```
}
```