



JSP 2.3 & Servlet 3.1

파일업로드

- 김근형 -

1. 파일 업로드의 원리

- ▶ form 태그에서 파일을 업로드 시 사용하는 옵션

```
<form method="post" enctype="multipart/form-data">  
  <input type="file" name="filename">  
</form>
```

- ▶ multipart/form-data : 이 타입이 지정이 안될 경우 파일 선택 박스에서 선택된 파일 객체가 전송되는 것이 아닌 파일 이름만 문자열 형태로 서버에 전송된다.

2. 업로드 모듈 COS 라이브러리

- ▶ 가장 널리 쓰이는 업로드 모듈 COS 라이브러리
- ▶ 다운로드 : <http://www.servlets.com>

Servlet Software
com.oreilly.servlet

The essential toolkit for Servlet and JSP developers

- [Download the package](#)
- [Read the documentation](#)






Download

This is a .zip readable by "jar", newer releases are at the top.
To be notified when new versions release, [subscribe here](#).
Be sure to check out the [FAQ](#) and [Javadocs](#) below.

Version	Comments
cos-26Dec2008.zip	File upload improvements: <ul style="list-style-type: none">• Added support for Servlets 2.4 and Java 5.• Added an ExceededSizeException type to make catching easier.• Added support for EBCDIC machines.• Added a workaround for browsers that send Content-Length of -1.• Added a workaround for Opera missing parameter names.

2. 업로드 모듈 COS 라이브러리

- ▶ 가장 널리 쓰이는 업로드 모듈 COS 라이브러리
 - ▶ 해당 파일의 압축을 풀어 lib 폴더에 있는 cos.jar를 WEB-INF > lib에 복사

- ▼  WebContent
 - >  META-INF
 -  upload
 - ▼  WEB-INF
 - ▼  lib
 -  cos.jar

2. 업로드 모듈 COS 라이브러리

▶ MultipartRequest 클래스의 생성자

```
MultipartRequest(HttpServletRequest request,  
                  String saveDirectory,  
                  int maxPostSize,  
                  String encoding,  
                  FileRenamePolicy policy)
```

2. 업로드 모듈 COS 라이브러리

▶ MultipartRequest 클래스의 생성자

인자	설명
request	MultipartRequest와 연결될 request 객체를 의미
saveDirectory	서버 컴퓨터에 파일이 실질적으로 저장될 경로를 의미
maxPostSize	한번에 업로드할 수 있는 최대 파일 크기
encoding	파일의 인코딩 방식을 의미
policy	파일 이름 중복 처리를 위한 클래스 객체를 의미

2. 업로드 모듈 COS 라이브러리

▶ MultipartRequest 클래스의 메서드

메소드	설명
getParameterNames()	폼에서 전송된 파라미터의 타입이 file이 아닌 파라미터들의 이름들도 Enumeration 타입으로 반환한다
getParameterValues()	폼에서 전송된 파라미터 값들을 배열로 받아온다
getParameter()	request 객체에 있는 지정된 이름의 파라미터 값을 가져온다.
getFileNames()	파일을 여러 개 업로드할 경우 타입이 file인 파라미터 이름들을 Enumeration 타입으로 반환한다.
getFileSystemName()	서버에 실제로 업로드된 파일의 이름을 반환한다

2. 업로드 모듈 COS 라이브러리

▶ MultipartRequest 클래스의 메서드

메소드	설명
getOriginalFileName()	클라이언트가 업로드한 파일의 원본 이름을 반환한다.
getContentType()	업로드된 파일의 마임타입을 반환한다.
getFile()	서버에 업로드 된 파일 객체 자체를 반환한다.

3. MultipartRequest 클래스를 이용한 파일 업로드 구현

▶ 파일 업로드 폼 작성하기

▶ fileUploadForm.jsp

```
<section id = "uploadFormArea">
<form action="fileUpload.jsp" method="post" enctype="multipart/form-data">
<table>
  <tr>
    <td colspan="2" class = "td_title" >파일 업로드 폼</td>
  </tr>
  <tr>
    <td><label for = "name">올린 사람 : </label></td><td><input type="text" name="name" id = "name"></td>
  </tr>
  <tr>
    <td><label for = "subject">제목 : </label></td><td><input type="text" name="subject" id = "subject"></td>
  </tr>
  <tr>
    <td><label for = "fileName1">파일명1 : </label></td><td><input type="file" name="fileName1" id = "fileName1"></td>
  </tr>
  <tr>
    <td><label for = "fileName2">파일명2 : </label></td><td><input type="file" name="fileName2" id = "fileName2"></td>
  </tr>
  <tr>
    <td colspan=2 align=center><input type="submit" value="전송"></td>
  </tr>
</table>
</form>
</section>
```

3. MultipartRequest 클래스를 이용한 파일 업로드 구현

▶ 업로드 페이지 작성하기

▶ fileUpload.jsp

```
<%
String uploadPath=request.getRealPath("/upload");

int size = 10*1024*1024;
String name="";
String subject="";
String filename1="";
String filename2="";
String origfilename1="";
String origfilename2="";

try{
    MultipartRequest multi=new MultipartRequest(request,
        uploadPath,
        size,
        "UTF-8",
        new DefaultFileRenamePolicy());

    name=multi.getParameter("name");
    subject=multi.getParameter("subject");

    Enumeration files=multi.getFileNames();

    String file1 =(String)files.nextElement();
    filename1=multi.getFilesystemName(file1);
    origfilename1= multi.getOriginalFileName(file1);

    String file2 =(String)files.nextElement();
    filename2=multi.getFilesystemName(file2);
    origfilename2=multi.getOriginalFileName(file2);

}catch(Exception e){
    e.printStackTrace();
}
%>
```

3. MultipartRequest 클래스를 이용한 파일 업로드 구현

- ▶ 업로드 페이지 작성하기

- ▶ fileUpload.jsp

```
><form name="filecheck" action="fileCheck.jsp" method="post">
    <input type="hidden" name="name" value="<%=name%>">
    <input type="hidden" name="subject" value="<%=subject%>">
    <input type="hidden" name="filename1" value="<%=filename1%>">
    <input type="hidden" name="filename2" value="<%=filename2%>">
    <input type="hidden" name="origfilename1" value="<%=origfilename1%>">
    <input type="hidden" name="origfilename2" value="<%=origfilename2%>">
</form>
<a href="#" onclick="javascript:filecheck.submit()">업로드 확인 및 다운로드 페이지 이동</a>
</body>
```

3. MultipartRequest 클래스를 이용한 파일 업로드 구현

- ▶ 업로드의 확인 및 다운로드 페이지 작성하기
 - ▶ 업로드 확인 페이지 작성하기(fileCheck.jsp)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%
    request.setCharacterEncoding("UTF-8");
    String name=request.getParameter("name");
    String subject=request.getParameter("subject");
    String filename1=request.getParameter("filename1");
    String filename2=request.getParameter("filename2");
    String origfilename1=request.getParameter("origfilename1");
    String origfilename2=request.getParameter("origfilename2");
%>
<html>
<head>
<title>파일 업로드 확인 및 다운로드</title>
</head>
<body>
출린 사람 : <%=name %><br>
제목 : <%=subject %><br>
파일명1 : <a href="file_down.jsp?file_name=<%=filename1 %>"><%=origfilename1 %></a><br>
파일명2 : <a href="file_down.jsp?file_name=<%=filename2 %>"><%=origfilename2 %></a><p>
</body>
</html>
```

3. MultipartRequest 클래스를 이용한 파일 업로드 구현

- ▶ 업로드의 확인 및 다운로드 페이지 작성하기
 - ▶ 브라우저에서 실행되는 파일에 대해서도 다운로드 박스가 출력되게 처리하는 페이지 코드 작성 (file_down.jsp)

```
<%  
String fileName = request.getParameter("file_name");  
  
String savePath = "upload";  
ServletContext context = getServletContext();  
String sDownloadPath = context.getRealPath(savePath);  
String sFilePath = sDownloadPath + "\\" + fileName;  
byte b[] = new byte[4096];  
FileInputStream in = new FileInputStream(sFilePath);  
String sMimeType = getServletContext().getMimeType(sFilePath);  
System.out.println("sMimeType>>>" + sMimeType);  
  
if (sMimeType == null)  
    sMimeType = "application/octet-stream";  
  
response.setContentType(sMimeType);  
String agent = request.getHeader("User-Agent");  
boolean ieBrowser = (agent.indexOf("MSIE") > -1) || (agent.indexOf("Trident") > -1);  
  
if (ieBrowser) {  
    fileName = URLEncoder.encode(fileName, "UTF-8").replaceAll("\\+", "%20");  
} else {  
    fileName = new String(fileName.getBytes("UTF-8"), "iso-8859-1");  
}  
  
response.setHeader("Content-Disposition", "attachment; filename= " + fileName);  
  
ServletOutputStream out2 = response.getOutputStream();  
int numRead;  
  
while ((numRead = in.read(b, 0, b.length)) != -1) {  
    out2.write(b, 0, numRead);  
}  
out2.flush();  
out2.close();  
in.close();  
%>
```

4. Part 인터페이스를 사용한 업로드

▶ Part 인터페이스

- ▶ part 인터페이스는 multipart/form-data 형태로 전송된 POST 요청의 항목 데이터를 다루는 기능들이 정의된 인터페이스이다.

메소드	설명
delete()	Part에 담겨있는 파일 항목을 관련된 임시 디렉토리를 포함하여 삭제한다.
getContentType()	Part 객체의 콘텐츠 타입을 String 형태로 반환한다.
getHeader(String name)	인자로 지정된 헤더의 정보를 String 형태로 반환한다.
getHeaderNames()	Part 객체의 헤더 정보들을 Collection<String> 형태로 반환한다.
getHeaders(String name)	인자로 지정된 헤더의 정보들을 Collection<String> 형태로 반환한다.

4. Part 인터페이스를 사용한 업로드

▶ Part 인터페이스

- ▶ part 인터페이스는 multipart/form-data 형태로 전송된 POST 요청의 항목 데이터를 다루는 기능들이 정의된 인터페이스이다.

메소드	설명
getInputStream()	Part의 내용을 읽어 들일 수 있는 InputStream 타입의 객체를 반환한다.
getName()	Part 객체의 이름을 String 타입으로 반환한다.
getSize()	파일의 크기를 바이트 단위의 long 타입으로 반환한다.
write(String filename)	Part 객체의 파일을 인자로 지정된 파일 이름으로 디스크 상에 출력한다.

4. Part 인터페이스를 사용한 업로드

▶ MultipartConfig 어노테이션

- ▶ MultipartConfig 어노테이션은 이 어노테이션이 지정된 서블릿 객체가 multipart/form-data 형태의 요청 데이터를 처리할 수 있게 해주는 어노테이션이다.
- ▶ 이 어노테이션이 지정된 서블릿 클래스 객체에서는 request 객체의 `getPart(String name)` 메소드나 `getParts()` 메소드를 호출하여 객체를 얻을 수 있다.

4. Part 인터페이스를 사용한 업로드

▶ MultipartConfig 어노테이션

옵션항목	설명
fileSizeThreshold	파일이 업로드될 때 임시 디렉토리에 저장되기 시작할 파일의 바이트 크기. 이 크기가 넘으면 임시 디렉토리에 저장되기 시작한다. 이 크기를 넘지 않으면 메모리에 저장된다. 기본값은 0이며 데이터 타입은 int이다.
location	업로드된 파일이 저장될 디렉토리를 String 타입으로 저장한다.
maxFileSize	업로드할 수 있는 최대 파일의 바이트 크기로 데이터 타입은 long이다.
maxRequestSize	하나의 요청에서 업로드할 수 있는 최대 바이트 수이다. 데이터 타입은 long이다.

4. Part 인터페이스를 사용한 업로드

- ▶ Part 인터페이스를 사용한 업로드 구현
 - ▶ partUploadForm1.jsp

```
<form action = "partUploadPro1" method = "POST" enctype="multipart/form-data">  
  <label for = "writer">작성자 : </label><input type = "text" name = "writer" id = "writer"><br>  
  <label for = "partFile1">업로드 파일 : </label><input type = "file" name = "partFile1" id = "partFile1"><br>  
  <input type = "submit" value = "단일업로드"/>  
</form>
```

4. Part 인터페이스를 사용한 업로드

- ▶ Part 인터페이스를 사용한 업로드 구현
 - ▶ UploadPro1Servlet.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    // TODO Auto-generated method stub  
    request.setCharacterEncoding("UTF-8");  
    String writer = request.getParameter("writer");  
    Part part = request.getPart("partFile1");  
    response.setContentType("text/html;charset=UTF-8");  
    PrintWriter out = response.getWriter();  
    String contentDisposition = part.getHeader("content-disposition");  
    String uploadFileName = getUploadFileName(contentDisposition);  
    part.write(uploadFileName);  
    out.println("작성자 " + writer + "님이 " + uploadFileName + " 파일을 업로드 하였습니다." );  
}
```

4. Part 인터페이스를 사용한 업로드

- ▶ Part 인터페이스를 사용한 업로드 구현
 - ▶ UploadPro1Servlet.java

```
//사용 브라우저가 IE인 경우
private String getUploadFileName(String contentDisposition) {
    String uploadFileName = null;
    String[] contentSplitStr = contentDisposition.split(";");
    int lastPathSeparatorIndex = contentSplitStr[2].lastIndexOf("\\");
    int lastQutosIndex = contentSplitStr[2].lastIndexOf("\"");
    uploadFileName = contentSplitStr[2].substring(lastPathSeparatorIndex + 1, lastQutosIndex);
    return uploadFileName;
}
```

4. Part 인터페이스를 사용한 업로드

- ▶ 다중 파일 업로드를 위한 클라이언트 코드 작성
 - ▶ partUploadForm2.jsp

```
<form action = "partUploadPro2" method = "POST" enctype="multipart/form-data">  
  <label for = "writer">작성자 : </label><input type = "text" name = "writer" id = "writer"><br>  
  <label for = "partFile1">업로드 파일1 : </label><input type = "file" name = "partFile1" id = "partFile1"><br>  
  <label for = "partFile2">업로드 파일2 : </label><input type = "file" name = "partFile2" id = "partFile2"><br>  
  <input type = "submit" value = "다중업로드"/>  
</form>
```

4. Part 인터페이스를 사용한 업로드

▶ 다중 파일 업로드를 위한 클라이언트 코드 작성

▶ PartUploadPro2Servlet.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String writer = request.getParameter("writer");
    String uploadFileNameList = "";
    for(Part part: request.getParts()){
        if(!part.getName().equals("writer")){
            String contentDisposition = part.getHeader("content-disposition");
            System.out.println("contentDisposition = " + contentDisposition);
            String uploadFileName = getUploadFileName(contentDisposition);
            part.write(uploadFileName);
            uploadFileNameList += " " + uploadFileName;
        }
    }
    out.println("작성자 " + writer + "님이 " + uploadFileNameList + " 파일을 업로드 하였습니다." );
}
```