



# JSP 5강 – 자료실

---

양 명 속

**[[now4ever7@gmail.com](mailto:now4ever7@gmail.com)]**



# 목차

---

- 파일 업로드
  - 업로드 컴포넌트 – cos.jar
- 자료실



파일 업로드

---



# 파일 업로드를 위한 폼 형태

- 웹 브라우저를 통해서 파일을 전송하기 위한 폼 구성
  - [1] 파일을 전송하기 위한 파일 선택창을 사용하기 위해 <input type="file"> 태그 사용
  - [2] 선택된 파일을 업로드하기 위해서는 form 태그의 method="post", enctype="multipart/form-data" 지정

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

- post방식일 경우 2가지 인코딩

- [1] application/x-www-form-urlencoded : 디폴트, 파일 이름만 전송됨

- [2] multipart/form-data : 파일 이름과 함께 파일 데이터가 전송됨



# 파일 업로드를 위한 폼 형태

- `<form method="post">`의 형태로 전송한 폼에 담겨진 파라미터들은 request 객체를 통해서 이름에 해당하는 값을 얻어낼 수 있으나
- `enctype="multipart/form-data"`로 지정한 폼을 전송했을 경우 request 객체로 얻어낸 파라미터의 이름으로 값을 얻어 낼 수 없음
- `enctype="multipart/form-data"`로 전송한 폼에 담겨진 파라미터들에 대한 이름과 값을 가져오고, `<input type="file">`로 지정된 파일을 서버상의 한 폴더에 업로드하기 위해서 특별한 컴포넌트가 필요함
- 업로드를 하기 위한 여러 가지 컴포넌트 중 [www.servlets.com](http://www.servlets.com)에서 제공하는 cos.jar 파일에서 필요한 컴포넌트를 선택하여 업로드 수행

web-inf부분에 붙이기



# 파일 업로드를 위한 방법

---

- [1] method 속성 : post
- [2] enctype : multipart/form-data
- [3] 폼 컨트롤 : <input type=file>
- [4] 업로드 컴포넌트 이용 : cos.jar => MultipartRequest



# 파일을 업로드하고 폼 데이터를 분석하는 도구 - cos.jar

---

- cos.jar 파일 다운로드 및 설치
  - [1] <http://www.servlets.com>
    - 왼쪽 메뉴 - com.oreilly.servlet 링크 클릭
  - [2] cos-26Dec2008.zip 다운로드 -> 압축 해제
    - 소스 : src\com\oreilly\servlet\multipart\
    - API 문서 : doc\
    - lib\ 클래스 파일을 다시 압축한 cos.jar 파일
      - cos.jar : 파일 업로드와 enctype="multipart/form-data" 로 넘겨져 오는 파라미터에 대한 이름과 값을 얻어낼 수 있음
  - [3] lib\cos.jar 파일 복사해서 톰캣 홈\webapps\mystudy\WEB-INF\lib 폴더에 붙여넣기



# 파일을 업로드하고 폼 데이터를 분석하는 도구 - cos.jar

---

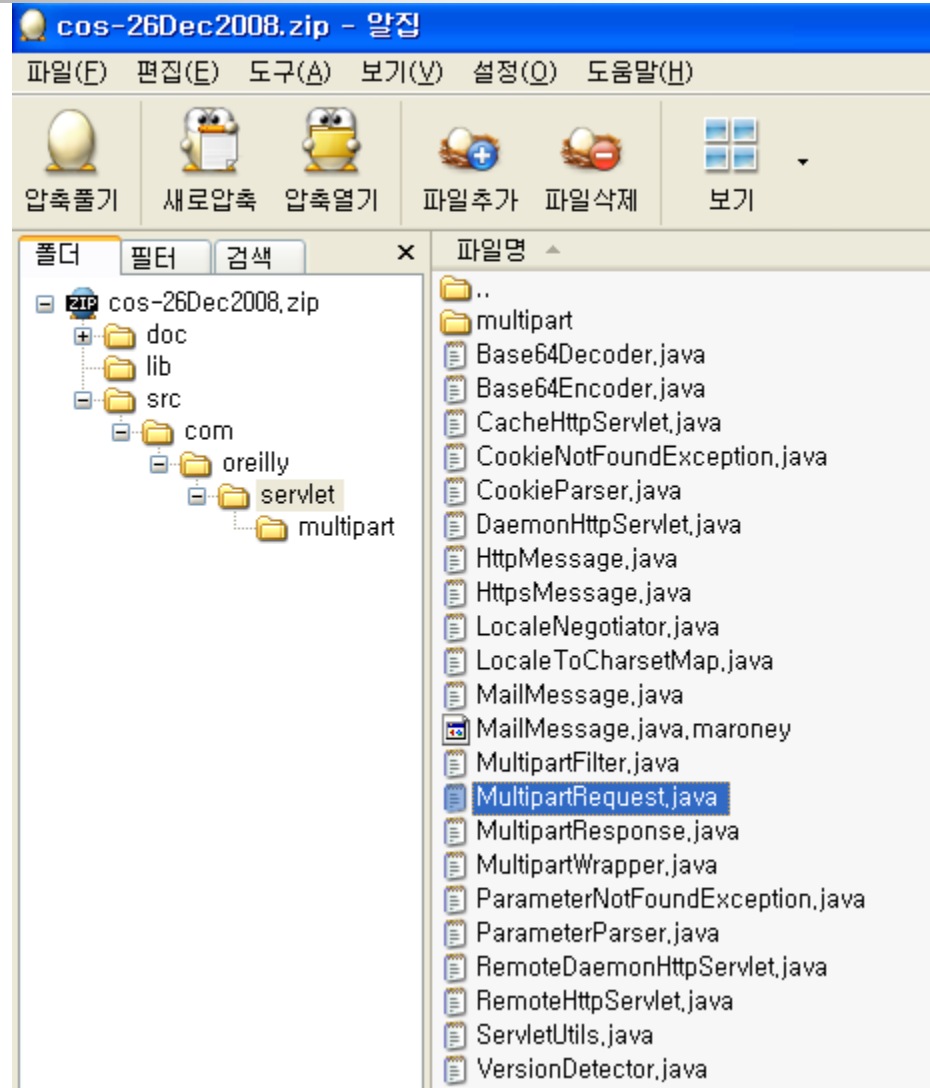
- [4] 톰캣 홈\Wwebapps\Wmystudy 폴더에 업로드될 파일을 저장할 upload 폴더 만들기



# 파일 업로드 및 폼 요소 처리를 위한 MultipartRequest 클래스

## ■ 1. cos.jar 파일의 구성

- cos.jar 파일은 패키지들로 구성
- 각각의 패키지에는 클래스 파일들이 존재
- cos.jar 파일을 웹 어플리케이션 폴더의 하위 폴더인 lib 에 넣어두고,
- 필요한 클래스 파일을 import 해주면 업로드에 관련한 해당 클래스 파일을 사용할 수 있음





# MultipartRequest 클래스

---

- MultipartRequest 클래스를 사용해서 파일 업로드 수행
  - 이 클래스에서 제공되는 API 이용-파일 업로드 및 폼 요소에 대한 처리 가능
- MultipartRequest 생성자
  - `MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, int maxPostSize, java.lang.String encoding, FileRenamePolicy policy)`



# MultipartRequest 의 생성자

- 파일 업로드 및 폼의 요소 처리
  - [1] 폼의 enctype 속성 값을 multipart/form-data 로 지정한 폼으로 부터 파라미터를 읽어오기 위해서 MultipartRequest 클래스의 객체 생성
  - [2] 해당 객체를 통해서 MultipartRequest 클래스에서 제공하는 메소드들을 사용
- MultipartRequest의 생성자
  - 한글 인코딩이나 업로드 되는 파일이 기존의 파일명과 중복될 때 파일명을 변경해서 똑같은 파일이 있을 때 덮어쓰는 단점을 해결하는 생성자

```
MultipartRequest mr = new MultipartRequest(request,  
                                             fileDirectory,  
                                             1024*5,  
                                             "utf-8",  
                                             new DefaultFileRenamePolicy());
```



# MultipartRequest 의 생성자

---

- 첫 번째 매개변수 : request 객체
- 두 번째 매개변수 : 업로드된 파일이 저장될 파일의 경로
- 세 번째 매개변수 : 업로드할 파일의 최대크기 (1024\*5 이므로 5KB)
- 네 번째 매개변수 : 인코딩 타입을 지정
- 다섯 번째 매개변수 : 업로드될 파일명이 기존에 업로드된 파일명과 이름이 같을 경우 덮어쓰기 되는 것을 방지하기 위해 설정하는 부분
- => 이 객체를 통해서 폼에서 지정한 파일이 웹 서버상의 지정된 폴더에 업로드됨

# MultipartRequest 의 메서드

return type	method
String	<code>getContentType(String name)</code> 업로드된 파일의 콘텐츠 타입을 리턴, 업로드된 파일이 없으면 null 을 리턴
File	<code>getFile(String name)</code> 서버상에 업로드된 파일의 파일 객체를 리턴, 업로드된 파일이 없다면 null 을 리턴
Enumeration	<code>getFileNames()</code> 폼 요소 중 input 태그 속성이 file로 된 파라미터의 이름들을 리턴, upload 된 파일이 없으면 비어 있는 Enumeration 을 리턴
String	<code>getFileName(String name)</code> 사용자가 지정해서 서버상에 실제로 업로드된 파일명을 리턴
String	<code>getOriginalFileName(String name)</code> 사용자가 지정해서 서버상에 업로드된 파일명을 리턴. 이때의 파일명은 파일의 중복을 고려한 파일명 변경 전의 이름
String	<code>getParameter(String name)</code> 스트링으로 주어진 이름에 대한 값을 리턴, 값 없이 파라미터가 전송되었거나, 해당되는 이름의 파라미터가 전송되지 않았을 경우 null을 리턴



# MultipartRequest 의 메서드

return type	method
Enumeration	<code>getParameterNames()</code> 모든 파라미터 이름을 Enumeration 으로 리턴
String[]	<code>getParameterValues(String name)</code> 주어진 이름에 대한 값을 스트링 배열로 리턴, 파라미터가 전송되지 않았을 때는 null 을 리턴

doc\ index.html 파일 참고



# MultipartRequest 의 메서드

- `getParameterNames()`

```
Enumeration params = mr.getParameterNames();
```

- 폼으로부터 전송된 파라미터들의 이름을 Enumeration 타입으로 리턴
- 입력 폼에 있는 input 태그 중 type 속성 값이 file 이 아닌 모든 파라미터들의 이름을 리턴

- `getParameter(String name)`

```
String pValue= mr.getParameter(String name);
```

- request 객체에서 사용되는 `getParameter()` 메서드와 같이 파라미터의 이름을 매개변수로 받아서 그 파라미터의 이름에 해당하는 값을 리턴하는 메서드



# MultipartRequest 의 메서드

- getFileNames()

```
Enumeration files = mr.getFileNames();
```

- 폼 요소 중 input 태그의 type속성이 file 인 파라미터의 이름을 Enumeration 객체 타입으로 리턴

- getFileName(String name)

```
String filename = mr.getFileName(name);
```

- type 속성값이 file 로 지정된 input 태그에 의해 서버상에 실제로 업로드된 파일명을 String 객체 타입으로 리턴
- 사용자가 선택한 파일이 실제로 서버상의 폴더에 저장되었을 때의 파일명을 리턴( 업로드시 파일명이 변경될 수 있음)





# MultipartRequest 의 메서드

- `getOriginalFileName(String name)`

```
String original = mr.getOriginalFileName(name);
```

- 사용자가 입력 폼에서 직접 지정한 파일명을 리턴.
- 이때의 파일명은 파일의 중복을 고려한 **파일명 변경 전의 이름**
- `getFileSystemName()` – 파일명이 중복되었을 경우 변경된 파일명 리턴
  - 예) 업로드한 파일명:abc.txt, 같은 파일을 중복해서 업로드하면 abc1.txt로 변경

- `getContentType(String name)`

```
String type = mr.getContentType(name);
```

- 업로드된 파일의 콘텐츠 타입을 리턴



# MultipartRequest 의 메서드

- getFile(String name)

```
File file = mr.getFile(name);
```

- 서버상에 업로드된 파일에 대한 파일 객체를 리턴
- File 객체 타입으로 리턴됨

- getParameterValues(String name)

- 주어진 이름에 대한 값을 스트링 배열로 리턴, 파라미터가 전송되지 않았을 때는 null 을 리턴



# uploadTest.jsp

---

```
<%@ page contentType="text/html; charset=utf-8"%>

<html>
<head>
<title>파일 업로드 예제</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>

<body>
<form name="frm" method="post" enctype="multipart/form-data" action="uploadTest_ok.jsp">
    작성자:      <input type="text" name="writer"><br>
    제 목:      <input type="text" name="title"><br>
    파일명:      <input type="file" name="uploadFile"><br>
    <input type="submit" value="파일 업로드"><br>
</form>
</body>
</html>
```

# uploadTest\_ok.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
<%@ page import="com.oreilly.servlet.MultipartRequest"%>
<%@ page import="com.oreilly.servlet.multipart.DefaultFileRenamePolicy"%>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<% String realFolder = ""; //웹 어플리케이션상의 절대 경로
//파일이 업로드되는 폴더를 지정한다.
String uploadFolder = "pds_upload";
String encType = "utf-8"; //인코딩타입
int maxSize = 5*1024*1024; //최대 업로드될 파일크기 5MB
```

파일 업로드시 기존 파일과 동일한 파일이 있을 때 덮어쓰기 방지  
하는 **DefaultFileRenamePolicy** 클래스 사용하기 위해

```
ServletContext context = getServletContext();
//현재 jsp페이지의 웹 어플리케이션상의 절대 경로를 구한다
// 주어진 가상경로에 대한 실제 경로를 리턴함
realFolder = context.getRealPath(uploadFolder);
out.println("the realpath is : " + realFolder+"<br>");
try{
    MultipartRequest mr = null;
    //전송을 담당할 콤포넌트를 생성하고 파일을 전송한다.
    //전송할 파일명을 가지고 있는 객체, 서버상의 절대경로, 최대 업로드될 파일크기, 문자코드, 기본 보
    안 적용(파일 업로드시 기존 파일과 동일한 파일이 있을 때 덮어쓰기 방지)
```

•하나의 웹 어플리케이션이 실행되는 환경은 하나의 **Context**에 대응됨,  
•이 **Context**는 **ServletContext interface**로 구현되는 한 객체로 **getServletContext()** 메서드로 얻음



# uploadTest\_ok.jsp

```
mr = new MultipartRequest(request,realFolder,maxSize,encType,new DefaultFileRenamePolicy());
//Form의 파라미터 목록을 가져온다
Enumeration params = mr.getParameterNames();
//파라미터를 출력한다
while(params.hasMoreElements()){
    String name = (String)params.nextElement(); //전송되는 파라미터이름
    String value = mr.getParameter(name); //전송되는 파라미터값
    out.println(name + " : " + value + "<br>");
}
out.println("-----<br>");

//전송한 파일 정보를 가져와 출력한다
Enumeration files = mr.getFileNames();

//파일 정보가 있다면
while(files.hasMoreElements()){
    //input 태그의 속성이 file인 태그의 name 속성값 :파라미터이름
    String name = (String)files.nextElement();

    //서버에 저장된 파일 이름
    String filename = mr.getFilesystemName(name);
```



# uploadTest\_ok.jsp

```
//전송전 원래의 파일 이름
String original = mr.getOriginalFileName(name);
//전송된 파일의 내용 타입
String type = mr.getContentType(name);

//전송된 파일 속성이 file인 태그의 name 속성값을 이용해 파일 객체 생성
File file = mr.getFile(name);
out.println("파라미터 이름 : " + name + "<br>");
out.println("실제 파일 이름 : " + original + "<br>");
out.println("저장된 파일 이름 : " + filename + "<br>");
out.println("파일 타입 : " + type + "<br>");

if(file!=null){
    out.println("크기 : " + file.length());
    out.println("<br>");
}
} //while
} catch(IOException ioe){
    System.out.println(ioe);
} catch(Exception ex){
    System.out.println(ex);
} %>
```

- 하나의 웹 어플리케이션이 실행되는 환경은 하나의 **Context**에 대응됨,
- 이 **Context**는 **ServletContext** interface로 구현되는 한 객체로 **getServletContext()** 메서드로 얻음

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@page import="com.oreilly.servlet.multipart.DefaultFileRenamePolicy"%>
<%@ page import="com.oreilly.servlet.MultipartRequest"%>
<%@page import="java.io.File"%>
<%

    //MultipartRequest 객체 생성 성공 => 업로드 완료
    //실패 => IOException 발생(파일 용량 초과 등)

    //업로드할 폴더의 절대경로 구하기
    //String saveDirectory = config.getServletContext().getRealPath("/upload");
    //String saveDirectory="D:\ \ java\ \ workspace\ \ mystudy\ \ upload";
    String saveDirectory=application.getRealPath("/upload");
    int maxSize=2*1024*1024; //2메가=>업로드 최대용량 2메가로 제한
    String encoding="utf-8";

    out.print("upload 디렉토리 : " + saveDirectory + "<br>");

    //업로드시 동일한 파일명이 있을 경우 나중에 업로드한 파일에 번호를 붙여 구분해줌
    DefaultFileRenamePolicy df = new DefaultFileRenamePolicy();
    MultipartRequest multi = new MultipartRequest(request, saveDirectory, maxSize, encoding, df);
    out.print("업로드 완료!<br>");

    String id = request.getParameter("id");
    out.print("id : "+ id + "<br>");

    id = multi.getParameter("id");
    out.print("id : "+ id + "<br>");
```

abc.txt => abc1.txt => abc2.txt

파일 업로드시 기존 파일과 동일한 파일이 있을 때 덮어쓰기 방지  
하는 **DefaultFileRenamePolicy** 클래스

```
String filename = multi.getFilesystemName("fileName");
out.print("업로드 된 파일명(변경후) : "+ filename + "<br>");
```

```
String filename1 = multi.getOriginalFileName("fileName");
out.print("업로드 된 파일명 원본 : "+ filename1 + "<br>");
```

```
String contentType = multi.getContentType("fileName");
out.print("contentType : "+ contentType + "<br>");
```

```
File myFile = multi.getFile("fileName");
out.print("file size : "+ myFile.length() + "<br>");
```

```
%>
```

```
<form name="frm1" method="post" action="upload_ok.jsp" enctype="multipart/form-data" >
  아이디 : <input type="text" name="id"><br>
  파일 업로드 : <input type="file" name="fileName" size="30"><br>
  <input type="submit" value="등록"><br>
</form>
```

파일 업로드 연습

아이디 :

파일 첨부 :



# 파일 업로드 테스트2

아이디

주소

파일첨부

UUID.txt

후기1.png

스프링스케줄러.docx

업로드된 파일 이름: 스프링스케줄러1.docx

원본 파일 이름: 스프링스케줄러.docx

파일사이즈: 334433

업로드된 파일 이름: 후기11.png

원본 파일 이름: 후기1.png

파일사이즈: 15994

업로드된 파일 이름: UUID1.txt

원본 파일 이름: UUID.txt

파일사이즈: 21311

아이디: null, 아이디2:hong

주소: 서울

```
<%@ page language="java" contentType="text/html; charset=UTF-8"    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head><meta charset="UTF-8"></head>
<body>
    <h1>파일 업로드 테스트2</h1>
    <form name="frm1" method="post" action="uploadTest2_ok.jsp"
        enctype="multipart/form-data">
        아이디 <input type="text" name="id"/><br>
        주소 <input type="text" name="address"/><br>
        파일첨부<br>
        <input type="file" name="upfile1" /><br>
        <input type="file" name="upfile2" /><br>
        <input type="file" name="upfile3" /><br><br>

        <input type="submit" value="전송" />

    </form>

</body>
</html>
```

```

<% String upDir="pds_upload";
    //실제 물리적인 경로 구하기
    //String saveDir=config.getServletContext().getRealPath(upDir);
    //saveDir=application.getRealPath(upDir);

String saveDir="D:\\jsp_ws\\mystudy\\WebContent\\pds_upload";
System.out.println("업로드 경로:"+saveDir);

int maxSize=2*1024*1024; //2M
String encoding="utf-8";
DefaultFileRenamePolicy policy = new DefaultFileRenamePolicy();
//=> 동일한 이름이 있는 경우 이름 변경  abc.txt => abc1.txt
try{
    MultipartRequest mr = new MultipartRequest(request, saveDir, maxSize, encoding, policy);
    //=> 파일 업로드 완료됨
    System.out.println("파일 업로드 완료!!");

    //업로드된 파일의 정보 읽어오기
    Enumeration fileNames = mr.getFileNames();
    //=> 여러개 파일을 업로드하는 경우 파일이름 목록을 리턴
    while(fileNames.hasMoreElements()){
        String fName = (String)fileNames.nextElement();
        //=> upfile1, upfile2, upfile3
        String fileName =mr.getFilesystemName(fName);
        //=> 업로드된 파일의 이름(변경된 파일 이름)
    }
}

```

```
String originalName = mr.getOriginalFileName(fName);
```

```
//=> 변경전 원래 파일 이름
```

```
File myfile = mr.getFile(fName);
```

```
long fileSize=0;
```

```
if(myfile!=null){
```

```
    fileSize=myfile.length();
```

```
}
```

```
out.println("업로드된 파일 이름: "+fileName+"<br>");
```

```
out.println("원본 파일 이름: "+originalName+"<br>");
```

```
out.println("파일 사이즈: "+fileSize+"<br><br>");
```

```
}//while
```

```
//사용자가 입력한 요청 파라미터 읽어오기
```

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

```
String id2 = mr.getParameter("id");
```

```
String address=mr.getParameter("address");
```

```
out.println("<hr>아이디: "+id+", 아이디2:"+id2+"<br>");
```

```
out.println("주소: "+address+"<br>");
```

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

```
    System.out.println("2M이상의 파일은 업로드할 수 없습니다!");
```

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

```
}
```

```
%>
```



자료실

---



# 자료실 Table

---

```
create table upboard
(
    no            number      primary key,    --번호
    name          varchar2(20) not null,      --이름
    pwd           varchar2(20) not null,      --비밀번호
    title         varchar2(100) not null,     --제목
    email         varchar2(80) null,          --이메일
    regdate       date        default sysdate, --작성일
    readcount     number      default 0 null,  --조회수
    content       clob        null,           --내용
    groupNo       number      default 0,      --그룹번호
    step          number      default 0,      --글의 단계
    sortNo        number      default 0,      --글의 정렬순서
    delFlag       char        default 'N',    --삭제 Flag
    fileName      varchar2(50) null,          --업로드파일명
    fileSize      number      default 0,      --파일사이즈
    downCount     number      default 0      --다운수
);

create sequence upboard_seq
increment by 1
start with 1
nocache;
```



# 업로드 폴더 준비하기

---

- 파일이 업로드될 폴더를 하나 만들어서 지정
  - 폴더의 이름: pds\_upload
  - 위치 : 홈디렉토리/pds\_upload
  - 업로드된 파일들이 저장될 폴더임

# 글쓰기 페이지

- 글 쓰기 페이지 -write.jsp

글 쓰는 폼에 업로드할 파일을 선택할 수 있도록  
[file] 컨트롤 추가

## 자료실 글쓰기

제목

작성자

비밀번호

이메일

내용

첨부파일

찾아보기...

(최대 2M)

등록

글목록





# write.jsp

---

```
<form name="frm1" method="post" enctype="multipart/form-data">
...
<div>
    <label for="content">내용</label>
    <textarea id="content" name="content" rows="12" cols="40"></textarea>
</div>
<div>
    <label for="upfile">첨부파일</label>
    <input type="file" id="upfile" name="fileName" /> (최대 2M)
</div>
```

<%

```
String saveDirectory=application.getRealPath("/pds_upload");
int maxSize=2*1024*1024; //2메가
String encoding="utf-8";
try{
    DefaultFileRenamePolicy df = new DefaultFileRenamePolicy();
    MultipartRequest multi
        = new MultipartRequest(request, saveDirectory, maxSize, encoding, df);
    System.out.println("파일 업로드 완료!!");

    String fileName = multi.getFilesystemName("fileName");

    long fileSize=0;
    if(fileName!=null){ //파일이 첨부된 경우만
        File myFile = multi.getFile("fileName");
        fileSize = myFile.length();
    }

    String name=multi.getParameter("name");
    String title=multi.getParameter("title");
    String pwd=multi.getParameter("pwd");
    String content=multi.getParameter("content");
    String email=multi.getParameter("email");

    UpBoardBean bean = new UpBoardBean();
    .....
    bean.setFileName(fileName);
    bean.setFileSize(fileSize);

    UpBoardDAO dao = new UpBoardDAO();
    int result = dao.insert(bean);
}
```

write\_ok.jsp

```

        if(result>0){
            out.print("<script>");
            out.print("alert('등록 성공!!');");
            out.print("location.href='list.jsp';");
            out.print("</script>");
        }else{
            out.print("<script>");
            out.print("alert('등록 실패!!');");
            out.print("history.back()");
            out.print("</script>");
        }
    }catch(IOException e){
        String str = "<script>";
        str += "alert('2M 이상의 파일은 업로드할 수 없습니다!');";
        str += "history.back();</script>";
        out.print(str);
    }
}

```

%>

```
public class UpBoardDAO {
    ConnectionPoolMgr pool;

    Connection con;
    PreparedStatement ps;
    ResultSet rs;



    public UpBoardDAO(){ //생성자
        pool = ConnectionPoolMgr.getInstance();
    }
    public int insert(UpBoardBean bean) throws SQLException{
        try{
            con = pool.getConnection();
            String sql="insert into upboard(no, name, pwd, title, email," +
                "content, groupno, filename, filesize) values("+
                "upboard_seq.nextval,?,?,?,?,?, upboard_seq.nextval,?,?)";
            ps=con.prepareStatement(sql);

            ps.setString(1,bean.getName());
            ps.setString(2,bean.getPwd());
            ps.setString(3,bean.getTitle());
            ps.setString(4,bean.getEmail());
            ps.setString(5,bean.getContent());
            ps.setString(6,bean.getFileName());
            ps.setLong(7,bean.getFileSize());
        }
    }
}
```

```
        int n=ps.executeUpdate();  
        return n;  
    }finally{  
        pool.dbClose(ps, con);  
    }  
}  
}  
} //class
```

# 목록보기 페이지

## 자료실

번호	제목	작성자	작성일	조회수
8	 이미지 파일 첨부	홍길순	2015-04-19	15
7	질문!!	김길순	2015-04-19	3
6	문의합니다	홍길동	2015-04-19	1
5	공지합니다	관리자	2015-04-19	2
4	 파일첨부합니다	이길동	2015-04-19	13

[1] 2 [3]

제목 ▼

검색

글쓰기

### ■ 자료실의 목록 보기

- 일반형 게시판 목록에서 첨부파일에 대한 부분만 추가
- 글이 파일을 첨부하고 있다는 의미로 파일이 첨부된 글은 글의 제목 앞에 간단한 이미지를 붙여 보여줌

list.jsp

```
<%=Utility.displayFile(bean.getFileName())%>  
<%=Utility.displayRe(bean.getStep())%>
```

utility.java

```
public static String displayFile(String fileName)  
{  
    String result="";  
    if (fileName !=null)  
    {  
        result= "<img src = '../images/file.gif' border='0'>";  
    }  
    return result;  
}
```

# 상세보기 페이지

## 글 상세보기

제목	파일 첨부합니다
작성자	홍길동
등록일	2015-09-26 22:43:30.0
조회수	1
첨부파일	 UUID2.txt (20KB) 다운 : 0

하나  
둘  
셋

[수정](#) | [삭제](#) | [답변](#) | [목록](#)

- 자료실의 본문 내용보기
  - 첨부된 파일의 정보를 보여주고
  - 그 링크를 클릭하면 파일의 다운로드가 가능하게 하는 기능 추가



# detail.jsp

.....

//파일이 첨부된 경우 파일 정보 보여주기

```
String fileInfo="";
```

```
//파일이미지 + 파일명 (파일크기 KB) + 다운 : 다운로드수
```

```
if(vo.getFileName()!=null && !vo.getFileName().isEmpty()){
```

```
    //파일이 첨부된 경우에만
```

```
    fileInfo="<img src = '../images/file.gif' border='0'> "
```

```
        + vo.getFileName() + " ("
```

```
        + (vo.getFileSize()/1024)+ "KB) 다운 : "
```

```
        + vo.getDownCount();
```

```
}
```

```
<div>
```

```
    <span class="sp1">첨부파일</span>
```

```
    <span><a href="downCount.jsp?no=<%=no%>&fileName=<%=vo.getFileName()%>">
```

```
        <%=fileInfo %></a></span>
```

```
</div>
```

```
<div class="lastDiv">
```

```
    <p class="content"><%=content %></p>
```

```
</div>
```

```
<% //다운로드 수 증가시키기
int no = Integer.parseInt(request.getParameter("no"));
String fileName = request.getParameter("fileName");
UpBoardDAO dao = new UpBoardDAO();
int n = dao.updateDownCount(no);
String dirPath = application.getRealPath("/pds_upload");
//파일이름을 인코딩한다.(euc-kr => ISO-8859-1)
//=> url전송시에는 모든 문자가 ISO-8859-1로 인코딩 되기 때문에
//강제 다운로드 창 띄우기
try{
    File myfile = new File(dirPath, fileName);

    //page의 설정을 바꾸기 위해서 response를 다 날려버림
    response.reset();

    //setContentType는 MIME 타입을 지정-octet-stream으로 지정시, 형식을 지정하지 않겠다는 것
    response.setContentType("application/octet-stream");

    //브라우저 파일 확장자를 포함하여 모든 확장자의 파일들에 대해 다운로드시 무조건 파일다운로드
    대화상자가 뜨도록 하는 헤더속성
    response.setHeader("Content-Disposition", "attachment;filename="
        + new String(fileName.getBytes("euc-kr"),"ISO-8859-1")+"");
    //url 전송시 ISO-8859-1로 인코딩되므로 한글 처리 위해 인코딩
```

```
//response.setContentLength((int)myfile.length());
//바이너리 데이터를 아스키 텍스트 형식으로 변환하기 위한 방법
//response.setHeader("Content-Treansper-Encoding", "binary");
```

```
//cache에서 해당 페이지 읽기방지, 로딩시마다 새로고침한 것
//response.setHeader("Pargma","no-cache");
//cache 막기
//response.setHeader("Expires","-1");
```

java.lang.IllegalStateException: getOutputStream() has already been called for this response

```
out.clear();
out=pageContext.pushBody();
```

이를 생략하면 프로그램 상엔 이상이 없으나 이미 존재하고 있는 **out**객체로 바이트 기반의 스트림 작업을 명시하면서 오류가 발생함

```
byte[] data = new byte[1024 * 1024];
BufferedInputStream bis = new BufferedInputStream(new FileInputStream(myfile));
BufferedOutputStream bos = new BufferedOutputStream(response.getOutputStream());
```

```
int count = 0;
while((count = bis.read(data))!= -1){
    bos.write(data);
}
```

**jsp**에서는 **servlet**으로 변환될때 내부적으로 **out** 객체가 자동으로 생성되기 때문에(**JSP** 에서 내장객체인 **JspWriter out** 이라는 변수가 선언 되어 있음) 따로 **out** 객체를 만들면 충돌이 일어나서 **IllegalStateException** 예외가 발생함

```
if(bis !=null) bis.close();
if(bos != null) bos.close();
}catch(Exception e){
    e.printStackTrace();
}
```



# UpBoardDAO

---

```
public int updateDownCount(int no)throws SQLException{
    int n=0;
    try{
        con = pool.getConnection();
        String sql="update upboard set downcount = downcount+1 where no=?";

        ps=con.prepareStatement(sql);

        ps.setInt(1, no);

        n=ps.executeUpdate();
    }finally{
        pool.dbClose(ps, con);
    }
    return n;
}
```



# 삭제하기

- 테이블의 데이터를 삭제하는 기능 이외에 첨부파일이 있는 경우 등록된 파일을 삭제하는 코드가 새롭게 추가

• detail.jsp

<a href

```
= "delete.jsp?no=<%=no%>&step=<%=bean.getStep()%>&groupno=<%=bean.getGroupNo()%>&fileName=<%=bean.getFileName()%>"> 삭제 </a> |
```

• delete.jsp

```
<input type="hidden" name="fileName" value='<%=request.getParameter("fileName")%>'>
```



# delete\_ok.jsp

---

<%

... ..

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

```
UpBoardDAO dao = new UpBoardDAO();
```

```
int result=0;
```

```
if(dao.checkPwd(no, pwd)){
```

```
    result = dao.deleteByNo(no, step, groupNo);
```

```
    if(result>0){
```

```
        //파일관련 처리
```

```
        String dirPath = application.getRealPath("/pds_upload");
```

```
        File myfile = new File(dirPath, fileName);
```

```
        //기존 파일이 있으면 파일삭제
```

```
        if (myfile.exists())
```

```
        {
```

```
            myfile.delete();
```

```
        }
```

```
        out.print("<script>");
```

```
        out.print("alert('삭제성공!!');");
```

```
        ... .
```

# 수정 페이지

## 글수정


제목 파일 첨부합니다

작성자 홍길동

비밀번호

이메일 h@nate.com

첨부파일  선택된 파일 없음

첨부파일목록  UUID2.txt (20KB)  
첨부파일을 새로 지정할 경우 기존 파일은 삭제됩니다.

내용  
하나  
둘  
셋

수정

글목록



# 수정 페이지

- 수정하기 페이지에서 새로 파일을 선택한 경우
  - 이전에 파일이 첨부된 경우엔 그 파일을 삭제
    - 이 파일이 존재하는가를 체크해서 존재한다면 삭제

```
//기존 파일은 삭제
File oldFile = new File(saveDirectory, multi.getParameter("oldFileName"));
if (oldFile.exists()){
    oldFile.delete();
}
```

- 새로운 파일을 서버에 저장
- 새로 파일을 선택하지 않은 경우
  - 이전 파일명과 사이즈를 그대로 넘겨줌





# edit.jsp

```
<input type="hidden" name="oldFileName" value='<%=bean.getFileName()%>'>
<input type="hidden" name="oldFileSize" value='<%=bean.getFileSize()%>'>
```

```
String fileInfo="";
//파일이 첨부된 경우 파일 정보를 보여주기 위한 처리
String fileName=vo.getFileName();
if(fileName!=null && !fileName.isEmpty()){
    fileInfo=Utility.displayFile(fileName)+" "
        + fileName + " (" + (vo.getFileSize()/1024)+"KB)"
        + "<br><span style='color:green;font-weight:bold'>"
        + "첨부파일을 새로 지정할 경우 기존 파일은 삭제됩니다.</span>";
}else{
    fileName="";
}
}
```

```
<div>
    <label for="upfile">첨부파일</label>
    <input type="file" id="upfile" name="upfile"/>
</div>
<div>
    <span class="sp1">첨부파일목록</span>
    <p><%=fileInfo %>
    </p>
</div>
```



# edit\_ok.jsp

---

```
<%
String saveDirectory=application.getRealPath("/pds_upload");
int maxSize=2*1024*1024; //2메가
String encoding="utf-8";
try{
    DefaultFileRenamePolicy df = new DefaultFileRenamePolicy();
    MultipartRequest multi
        = new MultipartRequest(request, saveDirectory, maxSize, encoding, df);
    String fileName = multi.getFilesystemName("fileName");

    long fileSize=0;
    if(fileName!=null){ //새로운 파일 첨부
        File myFile = multi.getFile("fileName");
        fileSize = myFile.length();
        //기존 파일은 삭제
        File oldFile = new File(saveDirectory, multi.getParameter("oldFileName"));
        if (oldFile.exists()) oldFile.delete();
    }else{
        fileName=multi.getParameter("oldFileName");
        fileSize=Long.parseLong(multi.getParameter("oldFileSize"));
    }
}
```



# edit\_ok.jsp

---

```
int no=Integer.parseInt(multi.getParameter("no"));
String name=multi.getParameter("name");

....
UpBoardBean bean = new UpBoardBean();
.....
bean.setFileName(fileName);
bean.setFileSize(fileSize);

UpBoardDAO dao = new UpBoardDAO();
int result = dao.update(bean);
if(result>0){
    out.print("<script>");
    out.print("alert('수정성공!!');");
    ....
}
}catch(IOException e){
    String str = "<script>";
    str += "alert('2M 이상의 파일은 업로드할 수 없습니다!');";
    str += "history.back();</script>";
    out.print(str);
}
```

%>

```

public int updateReBoard(ReBoardVO vo) throws SQLException{
    //글 수정
    Connection conn=null;
    PreparedStatement ps=null;
    int cnt=0;
    try{
        //1,2 conn
        conn=pool.getConnection();

        //3. ps
        String sql="update reboard";
        sql+=" set title=?, name=?, email=?, content=?";
        //새로 파일이 첨부된 경우에만 파일명, 파일크기 업데이트
        if(vo.getFileName()!=null && !vo.getFileName().isEmpty()){
            sql+=" , filename=?, filesize=?";
        }
        sql+=" where no=?";
        ps=conn.prepareStatement(sql);
        ps.setString(1, vo.getTitle());
        ps.setString(2, vo.getName());
        ps.setString(3, vo.getEmail());
        ps.setString(4, vo.getContent());
        if(vo.getFileName()!=null && !vo.getFileName().isEmpty()){
            ps.setString(5, vo.getFileName());
            ps.setLong(6, vo.getFileSize());
            ps.setInt(7, vo.getNo());
        }
    }
}

```

```

        }else{
            ps.setInt(5, vo.getNo());
        }

        //4. exec
        cnt= ps.executeUpdate();
        System.out.println("글 수정 cnt="+cnt+", 입력값 vo="+vo);
    }finally{
        pool.dbClose(ps, conn);
    }
    return cnt;
}

```