

## 8장. jsp 파일 업로드



*FileUpload*



# 파일 업로드(file upload)

## 파일 업로드(file upload)

- 웹 브라우저에서 서버로 파일을 전송하여 서버에 저장하는 것을 말한다.
- JSP 페이지에 폼 태그를 이용하여 서버에 전송하고 오픈 라이브러리를 이용해야 한다.

```
<form action="jsp파일" method="post" enctype="multipart/form-data">  
  <input type="file" name="요청파라미터 이름">  
</form>
```

- ① form 태그의 method 속성은 반드시 POST 방식으로 설정해야 한다.
- ② form 태그의 enctype 속성은 반드시 multipart/form-data 설정해야 한다.

## 파일 업로드 처리 방법

종류	특징	필수 라이브러리
MultipartRequest 이용	가장 간단한 방법	cos.jar
아파치 API 이용하기	편리하고 강력한 API 제공	commons-fileupload.jar commons-io-2.6.jar



# 파일 업로드(file upload)

## MultipartRequest 메서드

- 웹 브라우저에서 전송되는 요청 파라미터 중 일반 데이터는 `getParameter()`로 값을 받지만, 파일의 경우는 **`getFileNames()`**를 이용한다.

메소드	반환유형	설 명
<code>getContentType(String name)</code>	<code>String</code>	업로드된 파일의 콘텐츠 유형을 반환함
<code>getParameter(String name)</code>	<code>String</code>	요청 파라미터 이름이 name인 값을 전달받음
<code>getParameterNames()</code>	<code>java.util.Enumeration</code>	모든 요청 파라미터의 이름과 값을 Enumeration 객체 타입으로 전달 받음
<code>getFile(String name)</code>	<code>java.io.File</code>	서버에 업로드된 파일에 대한 파일 객체 반환
<code>getFileNames()</code>	<code>java.util.Enumeration</code>	폼 페이지에 input 태그 내 type 속성 값이 file로 설정된 요청 파라미터의 이름을 반환
<code>getFileSystemNames()</code>	<code>String</code>	사용자가 설정하여 서버에 실제로 업로드된 파일명을 반환.

# 파일 업로드(file upload)

## MultipartRequest 클래스 생성

- cos(com.oreilly.servlet) 패키지에 포함되어 있는 파일 업로드 컴포넌트로 오픈 라이브러리 cos.jar를 다운로드해서 사용한다.

```
MultipartRequest(javax.servlet.http.HttpServletRequest request,  
                  java.lang.String saveDirectory,  
                  int maxPostSize,  
                  java.lang.String encoding,  
                  FileRenamePolicy policy)
```

MultipartRequest의 생성자의 매개변수

종류	특징
saveDirectory	서버의 파일 저장 경로 설정
maxPostSize	파일의 최대 크기
encoding	인코딩 유형 설정
policy	파일명 변경 정책 설정, 덮어쓰기 여부 설정



# 파일 업로드(file upload)

## cos.jar를 이용한 파일 업로드

- 배포 사이트 : <http://servlets.com/cos>

의 요약 | [servlets.com/cos/](http://servlets.com/cos/)

### com.oreilly.servlet

Home of com.oreilly.servlet

There's no sense in reinventing the wheel--here are some servlet support packages. The most famous is the file upload package MultipartRequest and MultipartParser.

[View the Readme](#)  
[View the License](#)  
[View the FAQ](#)

#### View class documentation

##### Class Index

- class [Base64Decoder](#)
- class [Base64Encoder](#)
- class [CacheHttpServlet](#)
- class [DaemonHttpServlet](#)
- class [RemoteDaemonHttpServlet](#) (implements java.rmi.Remote)
- class [RemoteHttpServlet](#) (implements java.rmi.Remote)
- class [HttpMessage](#)
- class [HttpsMessage](#)
- class [LocaleNegotiator](#)
- class [LocaleToCharsetMap](#)
- class [MailMessage](#)
- class [MultipartRequest](#)
- class [MultipartParser](#)
- class [FilePart](#)
- class [ParamPart](#)

com.oreilly.servlet

## Class MultipartRequest

java.lang.Object

extended by [com.oreilly.servlet.MultipartRequest](#)

```
public class MultipartRequest  
extends java.lang.Object
```

A utility class to handle multipart/form-data requests, the kind of requests that support file uploads. It uses a "push" model where any incoming files are read and saved to a database, use the "pull" model [MultipartParser](#) instead.

This class can receive arbitrarily large files (up to an artificial limit you can set), a multipart content. It **can** now with the latest release handle internationalized content.

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.

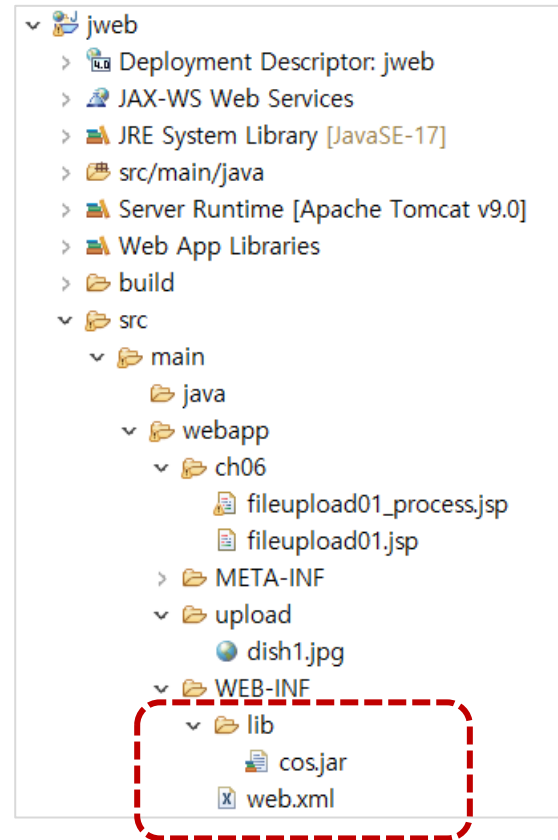
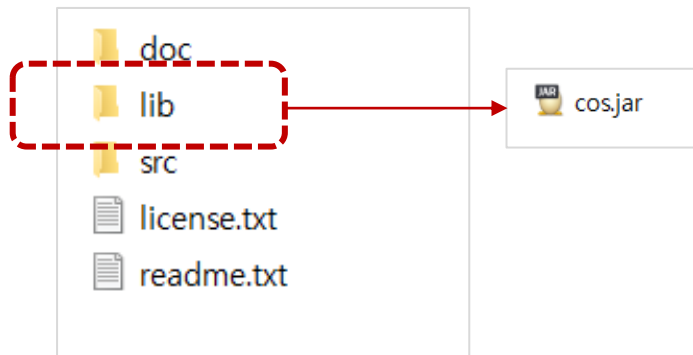
Version	Comments
	File upload improvements: <ul style="list-style-type: none"><li>Modernized the Javadoc and copyrights</li><li>Added support for Servlets 2.4 and Java 5.</li><li>Added an ExceededSizeException type to make catching easier.</li><li>Added support for EBCDIC machines.</li><li>Added a workaround for browsers that send Content-Length of -1.</li><li>Added a workaround for Opera missing parameter names.</li></ul>
<a href="#">cos-22.05.zip</a>	



# 파일 업로드(file upload)

## cos.jar를 이용한 파일 업로드

- cos-20.08.zip 압축 풀고 > lib 폴더 > cos.jar(복사) > WEB-INF/lib에 붙여 넣기



# 파일 업로드(file upload)

name=안산  
title=한국의 쌀밥

=====

요청 파라미터 이름: fileName

저장 파일 이름: dish1.jpg

실제 파일 이름: dish1.jpg

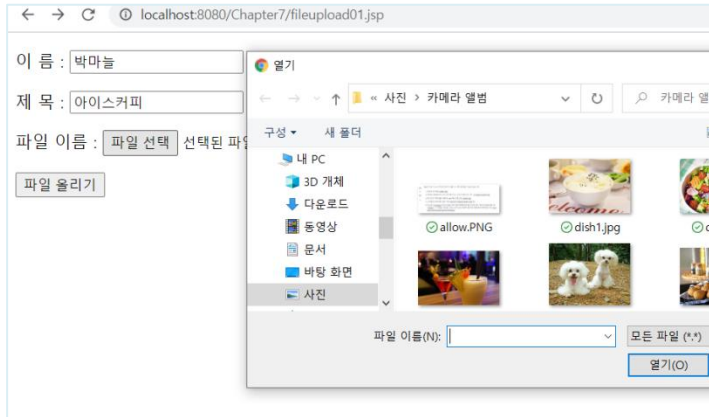
파일 콘텐츠 유형: image/jpeg

파일 크기: 55078B

[이미지 보기](#)



# 파일 업로드(file upload)



fileupload01.jsp

```
<!-- 파일을 업로드할 경우 enctype 속성에 "multipart/form-data"를 명시해야함-->
<form action="fileupload01_process.jsp" method="post"
      enctype="multipart/form-data">
  <p>이름: <input type="text" name="name">
  <p>제목: <input type="text" name="title">
  <p>파일이름: <input type="file" name="fileName">
  <p><input type="submit" value="파일올리기">
</form>
```



# 파일 업로드(file upload)

fileupload01\_process.jsp

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

    String realFolder = "C:\\\\Dev\\\\jspwork\\\\jweb\\\\src\\\\main\\\\webapp\\\\upload";

    MultipartRequest multi = new MultipartRequest(request, realFolder, 5*1024*1024,
        "utf-8", new DefaultFileRenamePolicy()); //5MB 한도

    //요청 파라미터 name(file name은 제외) 속성 가져오기
    Enumeration<String> params = multi.getParameterNames(); //이름 객체 생성
    while(params.hasMoreElements()){ //요청 파라미터 중 파일이 있다면 반복
        String name = params.nextElement(); //파일 가져오기
        String value = multi.getParameter(name); //이름으로 값 가져오기
        out.println(name + "=" + value + "<br>");
    }
    out.println("=====<br>");
```



# 파일 업로드(file upload)

fileupload01\_process.jsp

```
//요청 파라미터 중 file name 속성만 가져옴
Enumeration<String> files = multi.getFileNames();
while(files.hasMoreElements()){
    String name = files.nextElement();
    String fileName = multi.getFilesystemName(name);
    String original = multi.getOriginalFileName(name);
    String type = multi.getContentType(name);
    File file = multi.getFile(name);

    out.println("요청 파라미터 이름: " + name + "<br>");
    out.println("저장 파일 이름: " + fileName + "<br>");
    out.println("실제 파일 이름: " + original + "<br>");
    out.println("파일 콘텐츠 유형: " + type + "<br>");

    if(file != null){
        out.println("파일 크기: " + file.length() + "B");
    }
}

%>
<p>이미지 보기</p>
<p>
<% } %> <!-- while 닫기 -->
```

//서버에 업로드된 파일 가져오기  
//전송된 파일이 서버에 저장되기 전의 파일이름  
//파일의 콘텐츠 유형 가져오기  
//전송된 파일 가져오기



# 파일 업로드(file upload)

## 톰캣 서버의 server.xml 에 등록

업로드 폴더의 절대경로를 등록해야 이미지가 표시됨

```
<Context docBase="/jweb" path="/jweb" reloadable="true" source="org.eclipse.jst.jee.server" />  
<Context docBase="C:\\Dev\\jspwork\\jweb\\src\\main\\webapp\\upload" path="/jweb/upload" />  
</Host>  
</Engine>  
</Service>
```



# 파일 업로드(file upload)

## File 클래스

- 파일 자체의 경로나 정보를 알 수 있고, 파일을 생성할 수 있는 클래스
- java.io 패키지에 속해있고, 생성자는 File(String pathname)이다.

```
public class FileTest {  
  
    public static void main(String[] args) throws IOException {  
        File file = new File("C:\\filetest\\newFile.txt"); //경로 설정  
        file.createNewFile(); //실제 파일 생성  
  
        System.out.println(file.isFile());  
        System.out.println(file.getName());  
        System.out.println(file.getPath());  
        System.out.println(file.canRead());  
        System.out.println(file.canWrite());  
  
        //file.delete(); //파일 삭제  
    }  
}
```

```
true  
newFile.txt  
C:\\filetest\\newFile.txt  
true  
true
```

로컬 디스크 (C:) > filetest

이름

newFile.txt



# 파일 업로드(file upload)

## 다중 파일 업로드 및 정보 출력하기

← → ↻ ⓘ localhost:8181/Chapter12/fileupload02.jsp

이 름1 :  제 목1 :  파 일1 :  coffee.jpg

이 름2 :  제 목2 :  파 일2 :  fruit.jpg

이 름3 :  제 목3 :  파 일3 :  rose.jpg

```
<form action="fileupload02_process.jsp" method="post"
      enctype="multipart/form-data">
  <p>이 름1 : <input type="text" name="name1">
    제 목1 : <input type="text" name="title1">
    파 일1 : <input type="file" name="filename1">
  <p>이 름2 : <input type="text" name="name2">
    제 목2 : <input type="text" name="title2">
    파 일2 : <input type="file" name="filename2">
  <p>이 름3 : <input type="text" name="name3">
    제 목3 : <input type="text" name="title3">
    파 일3 : <input type="file" name="filename3">
  <p><input type="submit" value="파일올리기">
</form>
```



# 파일 업로드(file upload)

## 다중 파일 업로드 및 정보 출력하기

```
<%  
    String realFolder = "C:\\JspTest\\Chapter12\\WebContent\\upload";  
  
    MultipartRequest multi = new MultipartRequest(request, realFolder,  
        5*1024*1024, "utf-8", new DefaultFileRenamePolicy());  
  
    //폼 데이터 가져오기  
    String name1 = multi.getParameter("name1");  
    String title1 = multi.getParameter("title1");  
  
    String name2 = multi.getParameter("name2");  
    String title2 = multi.getParameter("title2");  
  
    String name3 = multi.getParameter("name3");  
    String title3 = multi.getParameter("title3");  
  
    //파일 정보 가져오기(스택 자료구조)  
    Enumeration<String> files = multi.getFileNames();  
  
    String file3 = files.nextElement();  
    String filename3 = multi.getFilesystemName(file3);  
  
    String file2 = files.nextElement();  
    String filename2 = multi.getFilesystemName(file2);
```



# 파일 업로드(file upload)

## 다중 파일 업로드 및 정보 출력하기

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

%>
<table>
  <tr>
    <th>이 름</th>
    <th>제 목</th>
    <th>파 일</th>
  </tr>
  <tr>
    <td>
      out.print("<tr><td>" + name1 + "</td>");
      out.print("<td>" + title1 + "</td>");
      out.print("<td>" + filename1 + "</td></tr>");

      out.print("<tr><td>" + name2 + "</td>");
      out.print("<td>" + title2 + "</td>");
      out.print("<td>" + filename2 + "</td></tr>");

      out.print("<tr><td>" + name3 + "</td>");
      out.print("<td>" + title3 + "</td>");
      out.print("<td>" + filename3 + "</td></tr>");
    </td>
  </tr>
</table>

%>
```

```
<style type="text/css">
  table{width: 300px;}
  table, th, td{
    border:1px solid #ccc;
    border-collapse: collapse;
    padding: 10px;
  }
</style>
```

이 름	제 목	파 일
김대리	커피	coffee.jpg
이대리	과일	fruit.jpg
박대리	장미	rose.jpg

