

Chapter 16
파일 업로드

파일 업로드의 개요

파일 업로드의 개요

■ 파일 업로드 라이브러리

- servlet과 JSP에서는 주로 오픈 소스 라이브러리를 사용해 파일의 업로드를 구현하고 있음
 - ▶ Java는 기본적으로 파일 업로드 기능을 제공하고 있지 않음
- 주요 사용 라이브러리의 종류
 - ▶ commons-fileupload 라이브러리
 - Apache Commons Library에서 제공하는 파일 업로드 라이브러리
 - ▶ cos(Com Oreilly Servlet) 라이브러리
 - <http://www.servlets.com>에서 제공하는 파일 업로드 라이브러리

파일 업로드의 개요

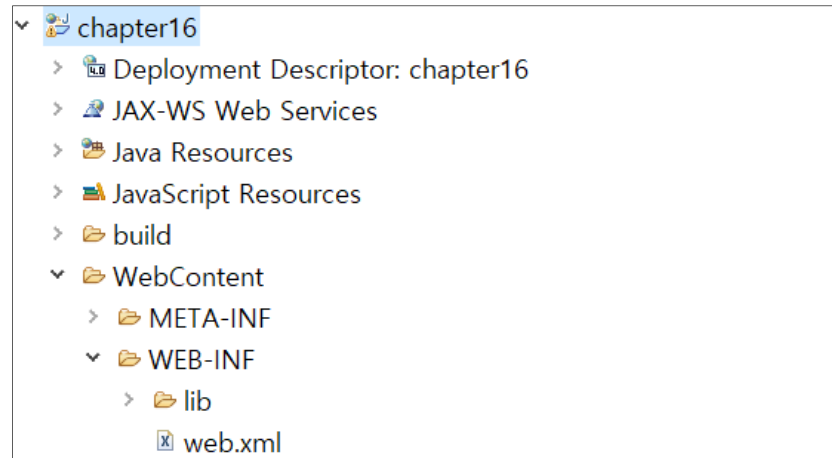
■ 업로드 구현 방법

- 대부분의 어플리케이션에서는 업로드된 파일을 별도의 폴더(디렉터리)에 저장함
 - 데이터베이스 내에 파일을 저장하지 않고 서버 내 특정 폴더에 업로드된 파일들을 유지함
 - 개발자는 업로드된 파일들이 유지되는 폴더를 생성하고 라이브러리를 사용해 업로드를 구현함
- 파일에 대한 정보만을 데이터베이스에 저장함
 - 파일의 이름, 크기, 종류 등의 정보만 데이터베이스에 저장함
 - 데이터베이스 내의 파일 정보와 폴더 내의 실제 파일과 매핑하여 파일을 관리함
- 업로드되는 파일들이 유지되는 위치
 - 일반적으로 해당 어플리케이션의 루트(/)에 폴더를 생성하여 사용
 - 모든 어플리케이션마다 별도의 폴더를 생성해 사용

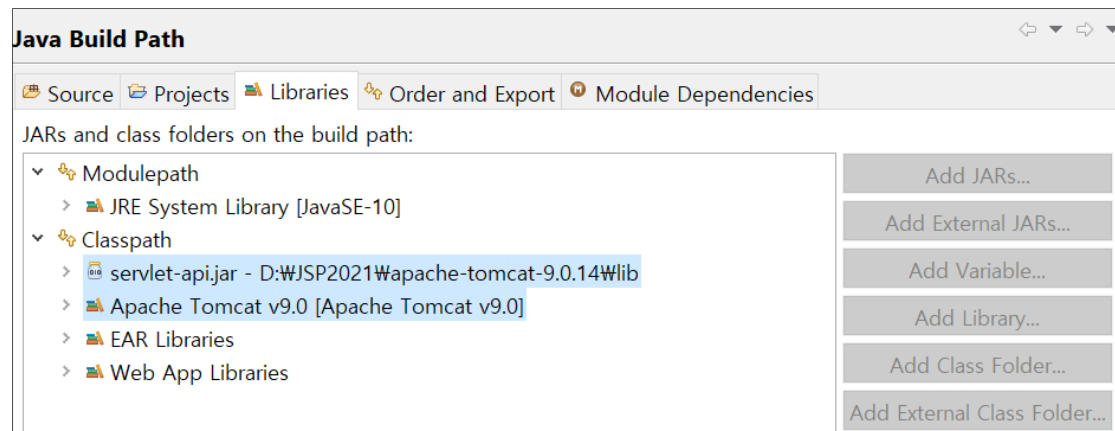
파일 업로드의 개요

■ 프로젝트 생성

- 프로젝트 이름 : chapter16
 - ▶ web.xml 파일은 반드시 생성해야 함



- servlet-api 라이브러리 추가



파일 업로드의 개요

■ 업로드 파일이 유지되는 폴더 생성

- 업로드 되는 파일들이 유지되는 폴더는 해당 어플리케이션의 루트에 생성해 사용함
 - ▶ 개발 환경인 이클립스에서 생성하는 것이 아니라 어플리케이션의 루트에 직접 생성해야 함
- 어플리케이션의 루트를 구하는 방법
 - ▶ `getRealPth()` 메서드를 사용하면 해당 어플리케이션 루트의 절대 경로를 구할 수 있음
 - ▶ 어플리케이션 루트에 파일이 업로드 되는 폴더를 생성함

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   ServletContext context = getServletContext();
5   String realPath = context.getRealPath("/");
6   out.println(realPath);
7 %>
```

← → ↻ ⓘ localhost:8080/chapter16/realPath.jsp ☆

D:\JSP2021\JSP_WORK\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\chapter16\

파일 업로드의 개요

- 업로드 되는 파일들이 저장되는 폴더의 이름을 [uploaded_files]로 지정하는 경우

로컬 디스크 (D:) > JSP2021 > JSP_WORK > .metadata > .plugins > org.eclipse.wst.server.core > tmp0 > wtpwebapps > chapter16 >				
이름	수정한 날짜	유형	크기	
META-INF	2021-05-16 오후 9:19	파일 폴더		
uploaded_files	2021-05-16 오후 9:28	파일 폴더		
WEB-INF	2021-05-16 오후 9:19	파일 폴더		
realPath.jsp	2021-05-16 오후 9:20	JSP 파일	1KB	

- uploaded_files 폴더의 절대 경로를 구하는 방법

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   ServletContext context = getServletContext();
5   String realPath = context.getRealPath("upload_files");
6   out.println(realPath);
7 %>
```

localhost:8080/chapter16/realPath.jsp

D:\JSP2021\JSP_WORK\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\chapter16\upload_files

commons fileupload 라이브러리

라이브러리 다운로드와 설치

■ 다운로드

- <https://commons.apache.org/index.html>에서 다운로드



- 문서 중앙의 [FileUpload] 구성요소를 선택

FileUpload	File upload capability for your servlets and web applications.	maven-central v1.4	2019-01-16
Functor	A functor is a function that can be manipulated as an object, or an object representing a single, generic function.	1.0	2011-??-??
Geometry	Space and coordinates.	maven-central v1.0-beta1	2020-07-19
Imaging (previously called Sanselan)	A pure-Java image library.	maven-central v1.0-alpha2	2020-08-01

라이브러리 다운로드와 설치

- Full Releases에서 FileUpload 1.4를 선택

Full Releases

FileUpload 1.4 - 23 December 2018

- Download the binary and source distributions from a mirror site [here](#)

FileUpload 1.3.3 - 13 June 2017

- Download the binary and source distributions from a mirror site [here](#)

- 바이너리 zip 버전 파일을 시스템의 임의의 위치에 다운로드











Apache Commons FileUpload 1.4 (requires Java 1.6 or later)

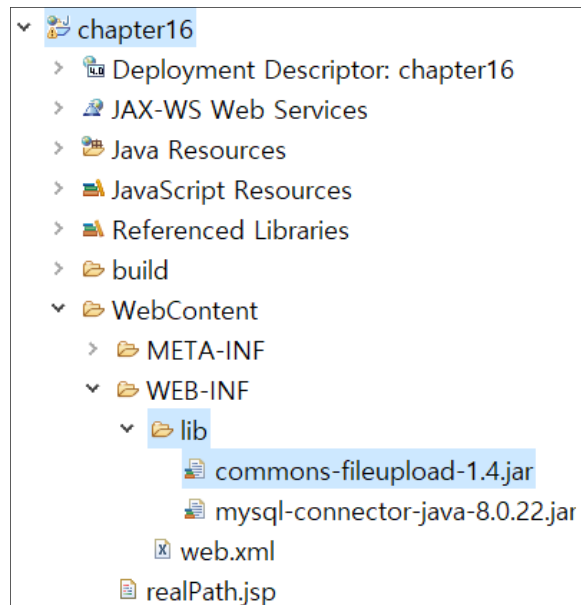
Binaries

commons-fileupload-1.4-bin.tar.gz	sha256	pgp
commons-fileupload-1.4-bin.zip	sha256	pgp

라이브러리 다운로드와 설치

- commons-fileupload-1.4.jar 파일을 복사하여 이클립스 프로젝트의 lib 폴더에 저장

 apidocs	✓	파일 폴더
 site	✓	파일 폴더
 commons-fileupload-1.4	✓	ALZip JAR File
 commons-fileupload-1.4-javadoc	✓	ALZip JAR File
 commons-fileupload-1.4-sources	✓	ALZip JAR File
 commons-fileupload-1.4-tests	✓	ALZip JAR File
 commons-fileupload-1.4-test-sources	✓	ALZip JAR File
 LICENSE	✓	텍스트 문서
 NOTICE	✓	텍스트 문서
 RELEASE-NOTES	✓	텍스트 문서



라이브러리 다운로드와 설치

- 문서 중앙의 [IO] 구성요소를 선택

IO	Collection of I/O utilities.	maven-central v2.8.0	2020-09-05
JCI	Java Compiler Interface	maven-central v1.1	2013-10-14

- 바이너리 zip 버전 파일을 시스템의 임의의 위치에 다운로드

Releases

Commons IO 2.8.0 (requires Java 8)

Commons IO 2.8.0 requires a minimum of Java 8 - [Download now!](#)

View the [Release Notes](#) and [Javadoc API documents](#)

Apache Commons IO 2.8.0 (requires Java 8)

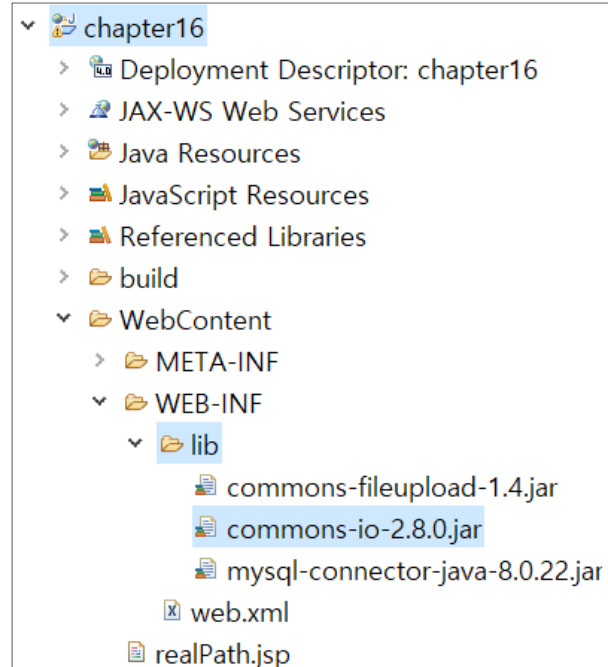
Binaries

commons-io-2.8.0-bin.tar.gz	sha512	pgp
commons-io-2.8.0-bin.zip	sha512	pgp

라이브러리 다운로드와 설치

- commons-io-2.8.0.jar 파일을 복사하여 이클립스 프로젝트의 lib 폴더에 저장

docs	✓	파일 폴더
commons-io-2.8.0	✓	ALZip JAR File
commons-io-2.8.0-javadoc	✓	ALZip JAR File
commons-io-2.8.0-sources	✓	ALZip JAR File
commons-io-2.8.0-tests	✓	ALZip JAR File
commons-io-2.8.0-test-sources	✓	ALZip JAR File
LICENSE	✓	텍스트 문서
NOTICE	✓	텍스트 문서
RELEASE-NOTES	✓	텍스트 문서



파일 업로드를 위한 <form> 요소 지정

■ <FORM> 태그를 이용한 파일 입력 양식 작성 시 유의 사항

- action 속성
 - ▶ 반드시 POST 방식이어야 함
- enctype 속성
 - ▶ 반드시 multipart/form-data 방식으로 지정해야 함
 - ▶ 지정하지 않거나 application/x-www-form-urlencoded으로 지정하면 파일 전송이 불가능함
- [사용 예]

```
<form action="commonFileUpload.jsp" method="post" enctype="multipart/form-data">
```

파일 업로드에 사용되는 클래스

■ DiskFileItemFactory 클래스

- 업로드된 파일을 저장할 폴더와 관련된 클래스
 - 파일의 크기와 파일의 저장 위치를 지정하는 메서드를 제공
- 주요 메서드

메서드	설명
setSizeThreshold(int)	<ul style="list-style-type: none">- 메모리에 저장할 수 있는 최대 크기를 바이트 단위로 지정- 기본 값은 10KB임- 지정된 크기 이상의 파일은 setRepositoy()에서 지정하는 위치를 임시 공간으로 사용
setRepositoy(File)	<ul style="list-style-type: none">- setSizeThreshold()가 지정한 크기 이상의 파일이 업로드된 경우 사용할 임시 위치를 File 객체로 지정- 지정하지 않을 경우 시스템이 사용하는 임시 디렉터리를 사용- 임시 파일로 저장되었던 파일은 이후에 FileItem의 write()메서드를 통해 실제 파일로 변경됨

파일 업로드에 사용되는 클래스

■ ServletFileUpload 클래스

- HttpServletRequest 객체에서 multipart/form-data 형식으로 넘어온 데이터를 다루기 쉽게 변환
- 주요 메서드

메서드	설명
parseRequest(request)	- multipart/form-data 형식으로 넘어온 데이터를 FileItem 형식으로 변환
setSizeMax(int)	- 업로드 할 수 있는 전체 파일의 최대 용량을 바이트 단위로 지정 - 무한대일 경우 -1로 지정
setFileSizeMax(int)	- 파일 개별 단위로 업로드 할 수 있는 최대 용량을 바이트 단위로 지정 - 무한대일 경우 -1로 지정

파일 업로드에 사용되는 클래스

■ FileItem 클래스

- multipart/form-data로 전송된 파라미터 또는 파일 정보를 처리하는 클래스
- 주요 메서드

메서드	반환타입	설명
isFormField()	boolean	파일이 아닌 일반적인 입력 파라미터일 경우 true를 리턴
getFieldName()	string	파라미터 이름을 반환
getString()	string	기본 캐릭터 셋을 사용하여 파라미터의 값을 반환
getString(String encoding)	string	지정한 인코딩을 사용하여 파라미터의 값을 반환
getName()	string	업로드 한 파일의 이름을 반환
getSize()	long	업로드한 파일의 크기를 반환
write(File file)	void	업로드 한 파일을 file이 나타내는 파일로 저장
delete()	void	임시 파일이 저장된 메모리를 반환, 임시 폴더의 파일을 제거

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9     <form action="commonFileUpProc.jsp" method="post" enctype="multipart/form-data">
10         <table border=1 cellspacing=0 cellpadding=3>
11             <tr>
12                 <td width=60 align=center>제 목</td>
13                 <td width=300><input type="text" name="userTitle"></td>
14             </tr>
15             <tr>
16                 <td width=60 align=center>파 일1</td>
17                 <td width=300><input type="file" size=60 name="userFile1"></td>
18             </tr>
19             <tr>
20                 <td width=60 align=center>파 일2</td>
21                 <td width=300><input type="file" size=60 name="userFile2"></td>
22             </tr>
23             <tr align=center>
24                 <td colspan="2"><input type="submit" value="업로드" ></td>
25             </tr>
26         </table>
27     </form>
28 </body>
29 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.io.File" %>
4 <%@ page import="java.util.List" %>
5 <%@ page import="org.apache.commons.fileupload.FileItem" %>
6 <%@ page import="org.apache.commons.fileupload.disk.DiskFileItemFactory" %>
7 <%@ page import="org.apache.commons.fileupload.servlet.ServletFileUpload" %>
8 <%
9   request.setCharacterEncoding("utf-8");
10
11   //uploaded_files 폴더에 대한 절대경로를 추출
12   String realFolder = getServletContext().getRealPath("uploaded_files");
13
14   //DiskFileItemFactory 객체 생성
15   DiskFileItemFactory factory = new DiskFileItemFactory();
16
17   //메모리에 저장할 수 있는 임시 파일의 크기를 1MByte로 지정
18   //생략가능(기본값이 10K로 지정됨)
19   factory.setSizeThreshold(1024*1024);
20
21   //임시파일이 setSizeThreshold에서 지정한 1MByte이상일 경우 임시위치를 지정
22   //생략가능 (시스템 기본 임시폴더를 사용 )
23   File currentDirPath = new File(realFolder);
24   factory.setRepository(currentDirPath);
25
26   //ServletFileUpload 객체를 생성
27   //ServletFileUpload 객체로 다루는 파일의 파일의 한글이름 인코딩
28   ServletFileUpload upload = new ServletFileUpload(factory);
29   upload.setHeaderEncoding("utf-8");
```

```
30
31  try {
32
33      // multipart/form-data 형식으로 전달된 모든 데이터를 리스트에 저장
34      List<FileItem> items = upload.parseRequest(request);
35
36
37      for (int i = 0; i < items.size(); i++) {
38
39          //fileItem 리스트로부터 순차적으로 데이터를 추출
40          FileItem fileItem = (FileItem) items.get(i);
41
42          if (fileItem.isFormField()) {
43              // 데이터가 파일이 아닐 경우 파라미터의 이름과 값을 출력(인코딩 포함)
44              out.println(fileItem.getFieldName() + "=" + fileItem.getString("utf-8") + "<br>");
45
46          } else {
47
48              //데이터가 파일을 경우 처리
49              if (fileItem.getSize() > 0) {
50
51                  // 파일의 이름을 추출하고 출력
52                  String fileName = (String) fileItem.getName();
53                  out.println("fileName : " + fileName+ "<br>");
54                  // 파일의 실제 저장 위치를 지정하고 출력
55                  String DirAndFilename = realFolder + "\\\" + fileName;
56                  out.println("DirAndFilename : " + DirAndFilename + "<br>");
```

```
57
58         // 파일을 uploaded_files 폴더에 저장
59         File uploadFile = new File(DirAndFilename);
60         fileItem.write(uploadFile);
61
62         //파일의 크기 출력
63         out.println("파일크기:" + fileItem.getSize() + "bytes" + "<br><br>");
64     }
65
66 }
67
68 }
69
70 } catch (Exception e) {
71
72     e.printStackTrace();
73
74 }
75
76 %>
```

← → ↻ ⓘ localhost:8080/chapter16/commonFileUpForm.jsp

제 목	<input type="text"/>
파 일 1	<input type="button" value="파일 선택"/> 선택된 파일 없음
파 일 2	<input type="button" value="파일 선택"/> 선택된 파일 없음
<input type="button" value="업로드"/>	

← → ↻ ⓘ localhost:8080/chapter16/commonFileUpForm.jsp

제 목	<input type="text" value="배경 화면 모음입니다."/>
파 일 1	<input type="button" value="파일 선택"/> bg1.png
파 일 2	<input type="button" value="파일 선택"/> bg2.png
<input type="button" value="업로드"/>	

← → ↻ ⓘ localhost:8080/chapter16/commonFileUpProc.jsp ☆

```

userTitle=배경 화면 모음입니다.
fileName : bg1.png
DirAndFilename : D:\JSP2021\JSP_WORK\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\chapter16\uploaded_files\bg1.png
파일크기:14789bytes

fileName : bg2.png
DirAndFilename : D:\JSP2021\JSP_WORK\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\chapter16\uploaded_files\bg2.png
파일크기:427658bytes
    
```

로컬 디스크 (D:) > JSP2021 > JSP_WORK > .metadata > .plugins > org.eclipse.wst.server.core > tmp0 > wtpwebapps > chapter16 > uploaded_files



cos 라이브러리

COS 라이브러리의 개요

■ COS 라이브러리의 개요

- **MultipartRequest 클래스를 사용해 파일 업로드를 수행**
 - ▶ 클라이언트로부터 전달되는 파일은 물론 일반 파라미터(텍스트 데이터를 가진)도 MultipartRequest 객체로 처리함
 - request 객체가 아님
 -
- **동일한 파일 이름 처리에 용이**
 - ▶ 서버에 같은 이름의 파일이 이미 존재할 경우, 새로 업로드되는 파일에 숫자를 추가해 업로드함
 - ▶ 한글 파일 이름 처리가 쉬움
- **파일이 업로드 되는 시점**
 - ▶ MultipartRequest 객체가 생성되는 순간 파일 업로드가 발생함

COS 라이브러리의 개요

■ cos 라이브러리 다운로드

• 다운로드

- ▶ <http://www.servlets.com/cos>에서 다운로드
- ▶ 사이트 하단의 **cos-20.08.zip** 파일을 임의의 위치에 다운로드
 - 다운 받은 파일을 압축 해제

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.

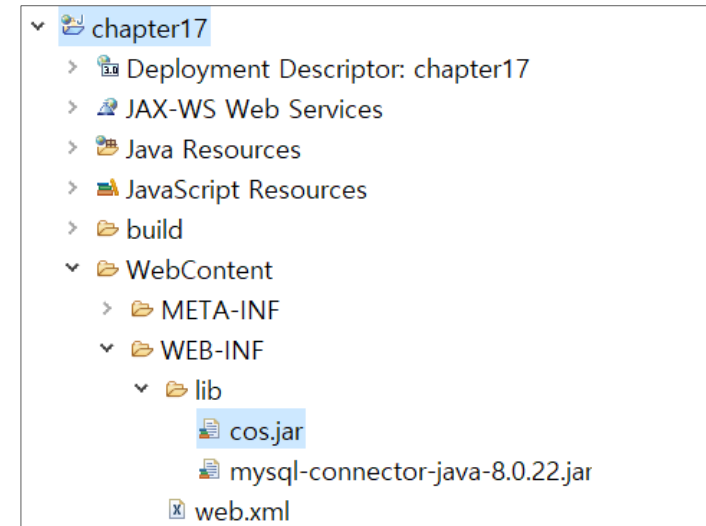
Version	Comments
cos-20.08.zip	File upload improvements: <ul style="list-style-type: none">• Added support for Servlets 2.4 and Java 5.• Added an ExceededSizeException type to make catching easier• Added support for EBCDIC mach• Added a workaround for browsers• Added a workaround for Opera m

이름	유형	크기
doc	파일 폴더	
lib	파일 폴더	
src	파일 폴더	
license	텍스트 문서	4KB
readme	텍스트 문서	2KB

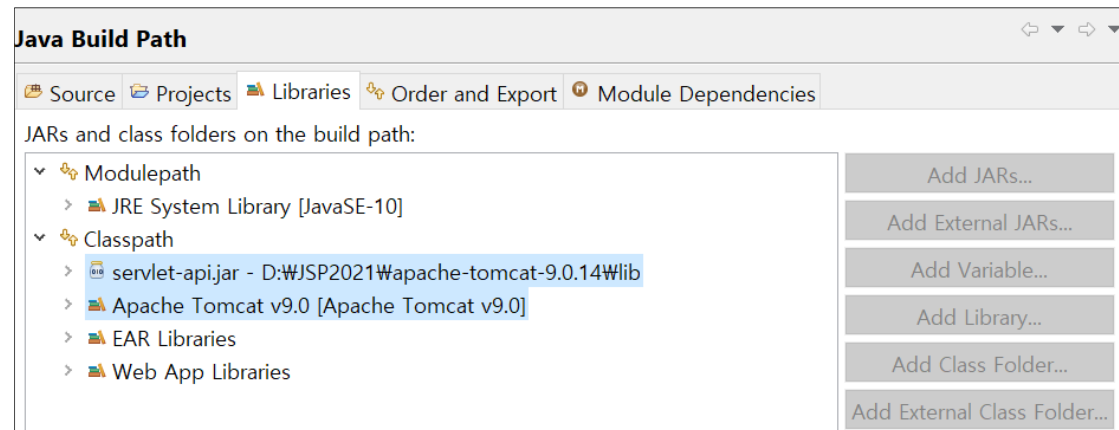
COS 라이브러리의 개요

■ 프로젝트 생성

- 프로젝트 이름 : chapter17
 - ▶ web.xml 파일은 반드시 생성해야 함
- 다운 받은 cos-20.08.zip 파일을 압축 해제
 - ▶ lib 폴더의 cos.jar 파일을 WebContent/WEB-INF/lib에 복사



- servlet-api 라이브러리 추가



MultipartRequest 클래스

■ MultipartRequest 클래스의 생성자

연번	생 성 자
1	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory)
2	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, int maxPostSize)
3	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, int maxPostSize, FileRenamePolicy policy)
4	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, int maxPostSize, java.lang.String encoding)
5	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, int maxPostSize, java.lang.String encoding, FileRenamePolicy policy)
6	MultipartRequest(javax.servlet.http.HttpServletRequest request, java.lang.String saveDirectory, java.lang.String encoding)
7	MultipartRequest(javax.servlet.ServletRequest request, java.lang.String saveDirectory)
8	MultipartRequest(javax.servlet.ServletRequest request, java.lang.String saveDirectory, int maxPostSize)

MultipartRequest 클래스

인자	설명
request	MultipartRequest 클래스와 연결되는 request 기본 객체 (클라이언트에서 전달되는 request 객체)
saveDirectory	업로드 되는 파일이 저장되는 서버 내의 폴더 (어플리케이션의 루트에 생성되어 있어야 함)
maxPostSize	POST 방식으로 업로드되는 파일의 최대 크기를 바이트 단위로 지정 (지정된 크기를 초과한 파일을 업로드할 경우 IOException가 발생)
encoding	업로드되는 파일의 이름에 대한 문자 집합을 지정 (파일의 이름이 한글인 경우 반드시 'UTF-8'을 사용해야 함)
policy	업로드하고자 하는 파일과 동일한 이름을 가진 파일이 서버에 이미 존재할 경우, 업로드되는 파일의 이름을 변경하여 서버에 저장 동일한 이름의 파일이 존재할 경우 업로드되는 파일의 이름 뒤에 숫자를 붙여 서버에 저장 (사용하지 않을 경우 서버에 존재하는 중복된 이름의 파일은 업로드되는 새로운 파일로 대체됨)

MultipartRequest 클래스의 주요 메서드

■ MultipartRequest 클래스의 주요 메서드

메서드	반환 타입	설명
getFileNames()	Enumeration	파일 입력 양식의 이름을 Enumeration 형식으로 반환
getContentType(String name)	String	업로드되는 파일의 형식의 MIME 타입을 반환
getFile(String name)	File	업로드되는 파일의 File 객체를 반환
getOriginalFileName(String name)	String	업로드되는 파일의 이름을 반환
getFileSystemName(String name)	String	업로드되어 서버에 저장되는 파일의 이름을 반환
getParameter(String name)	String	MultipartRequest 객체로부터 전달된 데이터를 반환

MultipartRequest 클래스의 주요 메서드

■ getFileNames()

- <FORM>내에 존재하는 FILE 입력 양식의 이름을 Enumeration 형식으로 반환
 - ▶ 예를 들어, 입력 양식이 <INPUT TYPE=FILE NAME='userfile'>로 지정된 경우 userfile이 반환
 - 파일을 업로드하지 않은 경우 비어있는 Enumeration 객체를 반환

■ getContentType(String name)

- 업로드하고자 하는 파일 형식의 MIME 타입을 반환
 - ▶ 예를 들어, EXE 파일을 업로드할 경우 application/octet-stream을 반환
 - 파일을 업로드하지 않은 경우 null을 반환

■ getFile(String name)

- 업로드되는 파일의 File 객체를 반환
 - ▶ 이 객체로부터 파일의 크기와 같은 정보를 얻을 수 있음
 - 파일을 업로드하지 않은 경우 null을 반환

MultipartRequest 클래스의 주요 메서드

■ getOriginalFileName(String name)

- 업로드되는 파일의 이름을 반환
 - ▶ 파일을 업로드하지 않은 경우 null을 반환

■ getFileSystemName(String name)

- 업로드되어 서버에 저장되는 파일의 이름을 반환
 - ▶ `FileRenamePolicy`를 사용해 중복되는 파일을 처리할 경우, 변경된 파일의 이름을 반환
 - 만일, 파일을 업로드하지 않은 경우 null을 반환
- [예] happy.txt라는 파일이 업로드되어 있는 상황에서 happy.txt라는 파일을 다시 업로드할 경우
 - ▶ 업로드되는 파일의 이름은 happy1.txt로 변경
 - `getOriginalFileName()`은 happy.txt를 반환하고, `getFileSystemName()`는 happy1.txt를 반환

MultipartRequest 클래스의 주요 메서드

■ getParameter(String name)

- MultipartRequest 객체로부터 전달된 파라미터를 반환

- ▶ application/x-www-form-urlencoded 인코딩 방식을 사용할 경우
 - request 기본 객체의 getParameter() 메소드를 적용해 전달된 파라미터를 추출함
- ▶ multipart/form-data 인코딩 방식과 COS 라이브러리를 사용할 경우
 - request 기본 객체가 아닌 MultipartRequest 객체를 사용해야 함

- 파라미터 name을 추출해 myname 변수에 저장하는 예

- ▶ application/x-www-form-urlencoded 인코딩을 사용하는 경우

```
String myname = request.getParameter("name")
```

- ▶ multipart/form-data 인코딩과 COS 라이브러리를 사용하는 경우

```
MultipartRequest multi = new MultipartRequest( .... );  
String myname = multi.getParameter("name")
```


파일 업로드 실습

- 파일 업로드 폴더 (uploaded_files) 생성

JSP_WORK > .metadata > .plugins > org.eclipse.wst.server.core > tmp0 > wtpwebapps > chapter17 > uploaded_files					
이름	상태	수정한 날짜	유형	크기	
이 폴더는 비어 있습니다.					

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9   <form action="cosFileUpProc.jsp" method="post" enctype="multipart/form-data">
10     <table border=1 cellspacing=0 cellpadding=3>
11       <tr>
12         <td width=60 align=center>제 목</td>
13         <td width=300><input type="text" name="userTitle"></td>
14       </tr>
15       <tr>
16         <td width=60 align=center>파 일1</td>
17         <td width=300><input type="file" size=60 name="userFile1"></td>
18       </tr>
19       <tr>
20         <td width=60 align=center>파 일2</td>
21         <td width=300><input type="file" size=60 name="userFile2"></td>
22       </tr>
23       <tr align=center>
24         <td colspan="2"><input type="submit" value="업로드" ></td>
25       </tr>
26     </table>
27   </form>
28 </body>
29 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <%@ page import="java.io.File" %>
5 <%@ page import="java.io.IOException" %>
6 <%@ page import="java.util.Enumeration" %>
7 <%@ page import="java.util.List" %>
8
9 <%@ page import="com.oreilly.servlet.MultipartRequest" %>
10 <%@ page import="com.oreilly.servlet.multipart.DefaultFileRenamePolicy" %>
11
12 <%
13   request.setCharacterEncoding("utf-8");
14   response.setContentType("text/html; charset=utf-8");
15
16   // 첨부한 원본 파일의 이름
17   String org_filename = null;
18
19   // 중복 파일 정책에 따라 변경된 파일 이름
20   String fs_filename = null;
21
22   //파일의 크기를 저장하는 변수 선언
23   int filesize=0;
24
25   //업로드되는 폴더(어플리케이션 루트 내부)
26   String realFolder = "";
27   realFolder = getServletContext().getRealPath("uploaded_files");
28
```

```
29 //MultipartRequest 클래스의 인자
30 int sizeLimit = 10*1024*1024;
31 String encType="utf-8";
32 DefaultFileRenamePolicy policy = new DefaultFileRenamePolicy();
33
34 try {
35
36     // MultipartRequest 객체 생성. 객체가 생성되는 순간 업로드 발생
37     MultipartRequest multi
38         = new MultipartRequest(request, realFolder, sizeLimit, encType, policy);
39
40     // 파라미터 출력
41     String title = multi.getParameter("userTitle");
42     out.println("파라미터 :" + title + "<br><hr><br>");
43
44     // 파일 입력 양식의 이름을 추출
45     Enumeration e = multi.getFileNames();
46
47     while(e.hasMoreElements()) {
48
49         String userfile = (String)e.nextElement();
50
51         // 파일 입력 양식에 입력된 파일의 원래 이름을 추출
52         org_filename = multi.getOriginalFileName(userfile);
53
54         // 파일이 업로드 된 경우 처리
55         if(org_filename != null) {
56
57             // 파일 입력 양식에 입력된 파일의 원래 이름을 출력
58             out.println("Original FileName :" + org_filename + "<br>");
```

```
59
60         // 서버에 저장되는 파일의 이름을 출력 (파일 이름 중복 처리 수행)
61         fs_filename = multi.getFilesystemName(userfile);
62         out.println("Filesystem FileName : " + fs_filename + "<br>");
63
64         // 입력된 파일의 객체 생성
65         File myfile = multi.getFile(userfile);
66
67         //파일의 크기 추출
68         filesize = (int)myfile.length();
69
70         // 파일의 크기와 종류 추출
71         if(filesize >0){
72             out.println("파일의 크기 : " + filesize + "<br> ");
73             out.println("파일의 종류 : " + multi.getContentType(userfile) + "<br> ");
74         }
75         out.println("<br><hr><br>");
76     }
77
78 }
79
80 } catch(Exception e) {
81
82     e.printStackTrace();
83 }
84
85 %>
```

파일 업로드 실습

cosFileUpProc.jsp

← → ↻ ⓘ localhost:8080/chapter17/cosFileUpForm.jsp ☆

제 목	<input type="text" value="강의 자료입니다."/>	
파 일 1	<input type="button" value="파일 선택"/>	Chapter01-오...환경구축.pdf
파 일 2	<input type="button" value="파일 선택"/>	Chapter02-S...SP의 이해.pdf
<input type="button" value="업로드"/>		



← → ↻ ⓘ localhost:8080/chapter17/cosFileUpProc.jsp ☆

파라미터 :강의 자료입니다.

Original FileName :Chapter02-Servlet과 JSP의 이해.pdf
 Filesystem FileName :Chapter02-Servlet과 JSP의 이해.pdf
 파일의 크기 : 705659
 파일의 종류 : application/pdf

Original FileName :Chapter01-오리엔테이션과 개발환경구축.pdf
 Filesystem FileName :Chapter01-오리엔테이션과 개발환경구축.pdf
 파일의 크기 : 3119668
 파일의 종류 : application/pdf

> JSP_WORK > .metadata > .plugins > org.eclipse.wst.server.core > tmp0 > wtpwebapps > chapter17 > uploaded_files

이름	상태	수정한 날짜	유형	크기
 Chapter01-오리엔테이션과 개발환경구축	✓	2021-05-19 오후 9:38	Adobe Acrobat 문서	3,047KB
 Chapter02-Servlet과 JSP의 이해	✓	2021-05-19 오후 9:38	Adobe Acrobat 문서	690KB

수고하셨습니다