

14. 파일 업로드

1. 파일 업로드의 개요
2. COS 라이브러리
3. MultipartRequest 클래스
4. Part 인터페이스
5. Commons 라이브러리
6. 파일 업로드 예제 작성

1. 파일 업로드의 개요

- 파일을 업로드하고 폼을 분석하는 라이브러리
 - ▣ cos 라이브러리(<http://servlets.com/cos/>)
 - cos.jar
 - ▣ commons 라이브러리(<http://commons.apache.org/>)
 - commons-fileupload-1.4.jar
 - commons-io-2.6.jar

1. 파일 업로드의 개요

□ 파일 업로드용 폼의 형태

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

- 파일을 입력받는 폼의 method 및 enctype 타입 설정
 - `method="post" enctype="multipart/form-data"`
- 파일을 업로드 하기 위해서는 input type 을 file 로 지정하고 enctype 속성을 설정
- 업로드 할 파일이 저장될 폴더 작성
 - [fileSave] 폴더

1. 파일 업로드의 개요

□ 파일 업로드용 폼의 형태

▣ 작성 방법

- 방법1 : <form> 태그의 속성을 사용하는 방법

```
<form name="formName" method="post"
      enctype="multipart/form-data">
```

- 방법2 : <input type="submit">에서 별도의 속성을 지정하는 방법
 - <input type="submit">에 HTML5에서 새로 추가된 *formenctype*, *formmethod* 속성을 사용

```
<input type="submit" formmethod="post",
      enctype="multipart/form-data" value="업로드 전송">
```

□ 파일 업로드 형태

▣ 1. FORM태그를 통째로 SUBMIT

```
<form id="frm" action="/fileupload" method="post">  
<input type="file" name="uploadfile" />  
<input type="file" name="uploadfile" />  
<input type="button" id="uploadbutton" value="클릭" />  
</form>
```

▣ 2. key,value로 append하면서 SUBMIT

```
<input type="file" name="uploadfile" />  
<input type="file" name="uploadfile" />  
<input type="button" id="uploadbutton" value="파일업로드" />
```

1. 파일 업로드 개요

□ 파일 업로드용 폼의 형태(jQuery Form Plugin 기반)

▣ 코딩 방법

■ HTML 태그

```
<form id="formid" method="post" action="upPro.jsp"
      enctype="multipart/form-data">
  <input type="text" name="title">
  <input type="file" name="selectfile">
  <input type="submit" value="전송">
</form>
```

■ 자바스크립트 코드

```
$("#formid").ajaxForm({
  success: function(data, status){
    $("#result").html(data);
  });
```

1. 파일 업로드 개요

□ 다중 파일 업로드용 폼의 형태

▣ 코딩 방법

■ HTML 태그

```
<input type="file" id="file1" name="file1" class="multi"
      maxlength="3">
```

■ 자바스크립트 코드

```
$("#formid").ajaxForm({
  success: function(data, status){
  }});
```

1. 파일 업로드 개요

□ 다중 파일 업로드용 폼의 형태

▣ 작성 방법

- 방법3 : HTML5에서 추가된 multiple속성을 사용
 - 코딩 방법

```
<input type="file" name="selectfile" multiple>
```

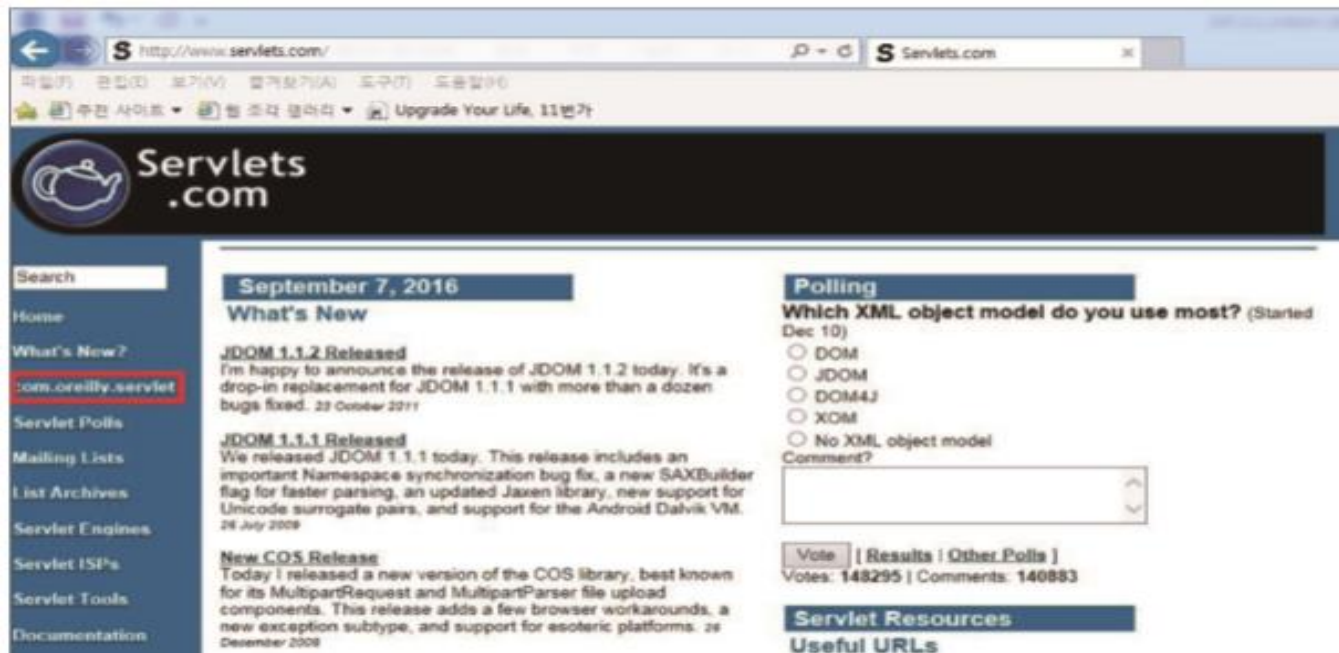

2. cos 라이브러리

□ cos.jar

- ▣ 파일을 업로드하고 폼으로부터 넘어오는 데이터를 얻는데 사용

□ 다운로드 및 설치

- ▣ <http://www.servlets.com/cos> 사이트에서 => com.oreilly.servlet 클릭



2. cos 라이브러리

□ 다운로드 및 설치

- ▣ 다운로드 영역에서 cos-20.08.zip 다운로드

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 machines.• Added a workaround for browsers that send Content-Length of -1.• Added a workaround for Opera missing parameter names.

- ▣ 프로젝트의 라이브러리 폴더([프로젝트]-[WebContent]-[WEB-INF]-[lib])에 복사

3. MultipartRequest 클래스

□ multipartRequest 클래스

- 파일 업로드 및 폼 요소를 처리하는 클래스
- COS 라이브러리에서 가장 핵심적인 역할을 수행하는 클래스
- 파일 업로드를 담당하는 생성자와 여러 메소드를 포함하고 있음
- 다른 업로드 라이브러리보다 **안정적**이며 **파일 중복 처리** 가능
- 생성자

```
MultipartRequest(javax.servlet.http.HttpServletRequest request, //request 객체  
    java.lang.String saveDirectory, //파일 업로드할 폴더  
    int maxPostSize, //업로드 할 파일의 최대크기  
    java.lang.String encoding, //인코딩 방식  
    FileRenamePolicy policy //같은 파일 덮어쓰기 방지 설정  
);
```

3. MultipartRequest 클래스

□ MultipartRequest 생성자의 파라미터

인자	설명
request	MultipartRequest와 연결될 request 객체를 의미한다.
saveDirectory	서버 컴퓨터에 파일이 실질적으로 저장될 경로를 의미한다.
maxPostSize	한 번에 업로드할 수 있는 최대 파일 크기를 의미한다.
encoding	파일의 인코딩 방식을 의미한다.
policy	파일 이름 중복 처리를 위한 클래스 객체를 의미한다.

□ 객체 생성의 예

```
MultipartRequest upload = new MultiPartRequest(  
    request,  
    fileSave,  
    1024*5,  
    "utf-8",  
    new DefaultFileRenamePolicy( )  
);
```

3. MultipartRequest 클래스

□ MultipartRequest 클래스의 메소드

메소드	설명
getParameterNames()	폼에서 전송된 파라미터의 타입이 file이 아닌 파라미터들의 이름들을 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.
getFileName()	서버에 실제로 업로드된 파일의 이름을 반환한다.
getOriginalFileName()	클라이언트가 업로드한 파일의 원본 이름을 반환한다.
getContentType()	업로드된 파일의 마임 타입을 반환한다.
getFile()	서버에 업로드된 파일 객체 자체를 반환한다.

파일 업로드 예1

```
<body>
<form action="upload.do" method="post" enctype="multipart/form-data">
글쓴이 : <input type="text" name="name"> <br>
제 &nbsp;   목 : <input type="text" name="title"> <br>
파일 지정하기 : <input type="file" name="uploadFile"> <br>
<input type="submit" value="전송">
</form>
</body>
```

```

@WebServlet("/upload.do")
public class UploadServlet extends HttpServlet {
    ...
    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        String savePath = "upload"; // 여기를 바꿔주면 다운받는 경로가 바뀜
        int uploadFileSizeLimit = 5 * 1024 * 1024; // 최대 업로드 파일 크기 5MB로 제한
        String encType = "UTF-8";
        ServletContext context = getServletContext();
        String uploadFilePath = context.getRealPath(savePath);
        System.out.println("서버상의 실제 디렉토리 :");
        System.out.println(uploadFilePath);
        try {
            MultipartRequest multi = new MultipartRequest(request, // request 객체
                uploadFilePath, // 서버상의 실제 디렉토리
                uploadFileSizeLimit, // 최대 업로드 파일 크기
                encType, // 인코딩 방법
                new DefaultFileRenamePolicy()); // 동일한 이름이 존재하면 새로운 이름이 부여됨

            String fileName = multi.getFilesystemName("uploadFile"); // 업로드된 파일의 이름 얻기
            if (fileName == null) { // 파일이 업로드 되지 않았을때
                System.out.print("파일 업로드 되지 않았음");
            } else { // 파일이 업로드 되었을때
                out.println("<br> 글쓴이 : " + multi.getParameter("name"));
                out.println("<br> 제 &nbsp;   목 : " + multi.getParameter("title"));
                out.println("<br> 파일명 : " + fileName);
            } // else
        } catch (Exception e) {System.out.print("예외 발생 : " + e); } // catch
    }
}

```

```
<body>
<form action="upload2.do" method="post" enctype="multipart/form-data">
  1. 파일 지정하기 : <input type="file" name="uploadFile01"> <br>
  2. 파일 지정하기 : <input type="file" name="uploadFile02"> <br>
  3. 파일 지정하기 : <input type="file" name="uploadFile03"> <br>
  <input type="submit" value="전송">
</form>
</body>
```



```
protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    String savePath = "upload";
    int uploadFileSizeLimit = 5 * 1024 * 1024;
    String encType = "UTF-8";
    ServletContext context = getServletContext();
    String uploadFilePath = context.getRealPath(savePath);
    try {
        MultipartRequest multi = new MultipartRequest(request,
            uploadFilePath, uploadFileSizeLimit, encType,
            new DefaultFileRenamePolicy());
        Enumeration files = multi.getFileNames();
        while (files.hasMoreElements()) {
            String file = (String) files.nextElement();
            String file_name = multi.getFilesystemName(file);
            // 중복된 파일을 업로드할 경우 파일명이 바뀐다.
            String ori_file_name = multi.getOriginalFileName(file);
            out.print("<br> 업로드된 파일명 : " + file_name);
            out.print("<br> 원본 파일명 : " + ori_file_name);
            out.print("<hr>");
        }
    } catch (Exception e) {
        System.out.print("예외 발생 : " + e);
    }
}
```

4. Part 인터페이스

□ part 인터페이스란?

- multipart/form-data 형태로 전송된 POST 요청의 항목 데이터를 다루는 기능들이 정의된 인터페이스
- √ Part 인터페이스에서 제공되는 메소드들

메소드	설명
delete()	Part에 담겨있는 파일 항목을 관련된 임시 디렉토리를 포함하여 삭제한다.
getContentType()	Part 객체의 콘텐츠타입을 String 형태로 반환한다.
getHeader(java.lang.String name)	인자로 지정된 헤더의 정보를 String 형태로 반환한다.
getHeaderNames()	Part 객체의 헤더 정보들을 Collection<String> 형태로 반환한다.
getHeaders(java.lang.String name)	인자로 지정된 헤더의 정보들을 Collection<String> 형태로 반환한다.
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(java.lang.String fileName)	Part 객체의 파일을 인자로 지정된 파일 이름으로 디스크 상에 출력한다.

4. Part 인터페이스

□ MultipartConfig 어노테이션

- 이 어노테이션이 지정된 서블릿 객체가 multipart/form-data 형태의 요청 데이터를 처리할 수 있게 해 주는 어노테이션.
- 이 어노테이션이 지정된 서블릿 클래스 객체에서는 request 객체의 `getPart(String name)` 메소드나 `getParts()` 메소드를 호출하여 Part 객체를 얻을 수 있음
- MultipartConfig 어노테이션의 옵션

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 0이다. 데이터 타입은 int이다.
location	업로드된 파일이 저장될 디렉토리를 String 타입으로 지정한다.
maxFileSize	업로드할 수 있는 최대 파일의 바이트 크기로 데이터 타입은 long이다.
maxRequestSize	하나의 요청에서 업로드할 수 있는 최대 바이트 수이다. 데이터 타입은 long이다.

5. Commons 라이브러리

1.1 파일 업로드 라이브러리 설치

1. jakarta.apache.org로 접속한 후 왼쪽 메뉴에서 Commons를 클릭.

**The Apache Jakarta Project**
[http:// jakarta.apache.org/](http://jakarta.apache.org/)

Support

- [License](#)
- [Mailing Lists](#)
- [Jakarta Wiki](#)

Ex-Jakarta

- [Ant](#)
- [Avalon](#)
- [BCEL](#)
- [BSF](#)
- [Commons](#)
- [DB](#)
- [Excalibur](#)
- [Gump](#)
- [HiveMind](#)
- [HttpComponents](#)
- [James](#)
- [JCS](#)
- [JMeter](#)

Welcome to The Apache Jakarta™ Project

Founded in 1999, the Jakarta Project housed a diverse set of Jakarta subprojects began to become full top-level Apache projects, join other TLPs (Commons), or in some cases

News
Latest Jakarta News

- [21 December 2011 - Jakarta Retired](#)
- [26 October 2011 - JMeter becomes a top level project](#)
- [03 October 2011 - Apache JMeter 2.5.1 Released](#)
- [11 September 2011 - BSF moves to Apache Commons](#)
- [17 August 2011 - Apache JMeter 2.5 Released](#)
- [05 August 2011 - Cactus moves to Apache Attic](#)
- [25 June 2011 - JCS moves to Apache Commons](#)
- [25 June 2011 - BCEL moves to Apache Commons](#)
- [17 April 2011 - Regexp is retired](#)

5. Commons 라이브러리

2. 페이지 왼쪽 중간쯤에 위치한 FileUpload를 클릭.

DbUtils	JDBC helper library.
Digester	XML-to-Java-object mapping utility.
Email	Library for sending e-mail from Java.
Exec	API for dealing with external process execution and environment management in Java.
FileUpload	File upload capability for your servlets and web applications.
Functor	A functor is a function that can be manipulated as an object, or an object representing a single, generic function.
Geometry	Space and coordinates.
Imaging (previously called Sanselan)	A pure-Java image library.
IO	Collection of I/O utilities.

5. Commons 라이브러리

3. FileUpload 1.3.3 버전을 찾아서 here를 클릭합니다.

Downloading

Full Releases

FileUpload 1.3.3 - 13 June 2017

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

FileUpload 1.3.2 - 26 May 2016

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

5. Commons 라이브러리

4. commons-fileupload-1.3.3-bin.zip을 클릭해 다운로드.

Apache Commons FileUpload 1.3.3 (requires Java 1.5)

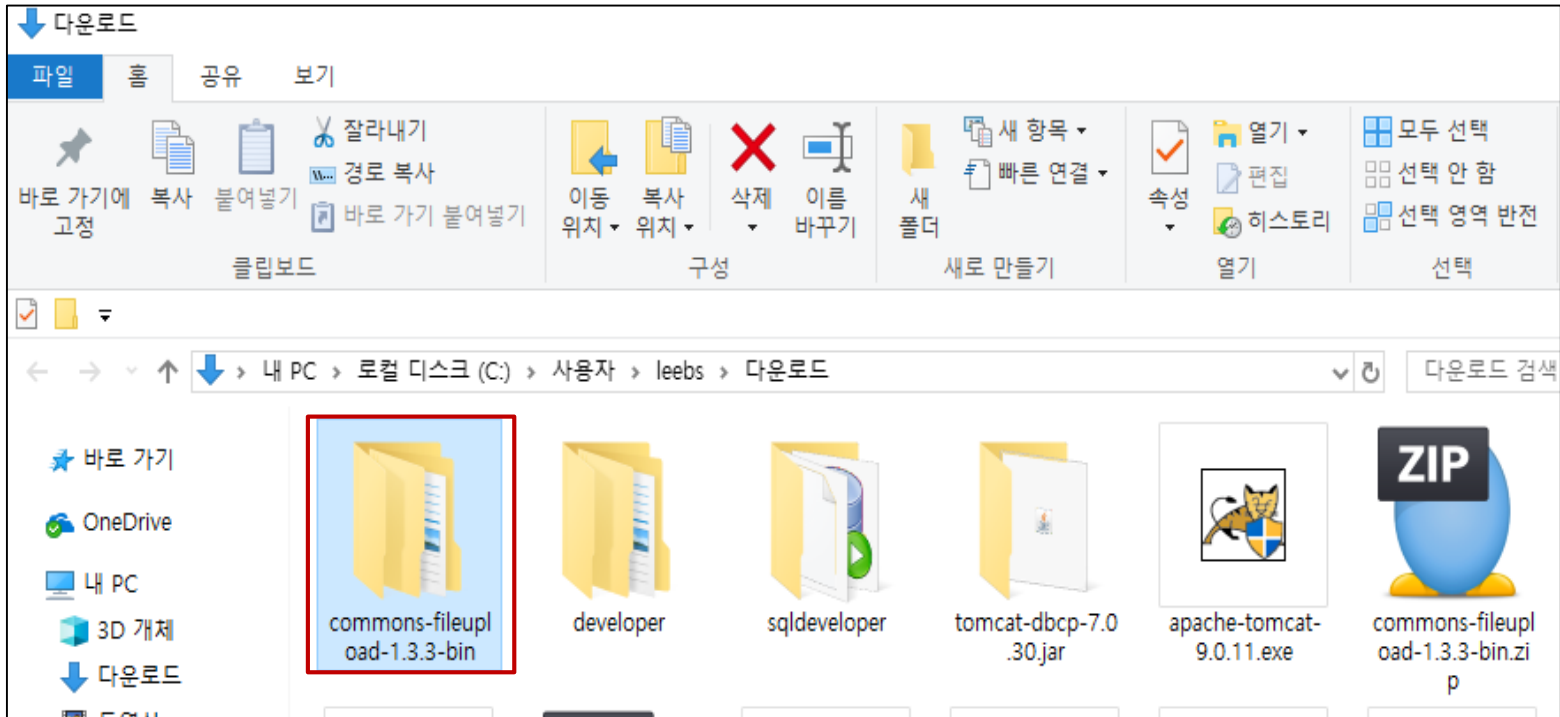
Binaries

[commons-fileupload-1.3.3-bin.tar.gz](#)

[commons-fileupload-1.3.3-bin.zip](#)

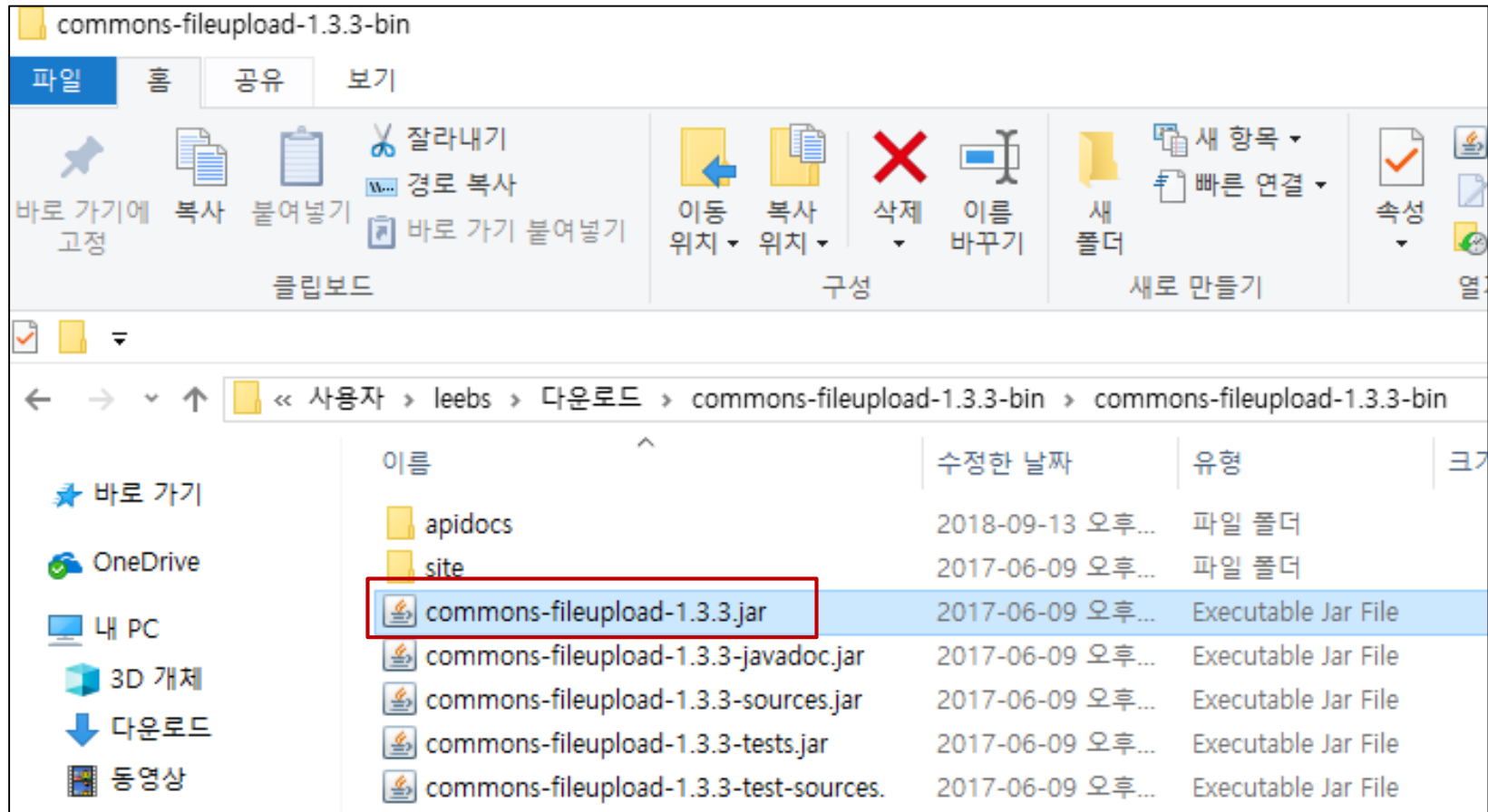
5. Commons 라이브러리

5. zip 파일의 압축을 푼다.



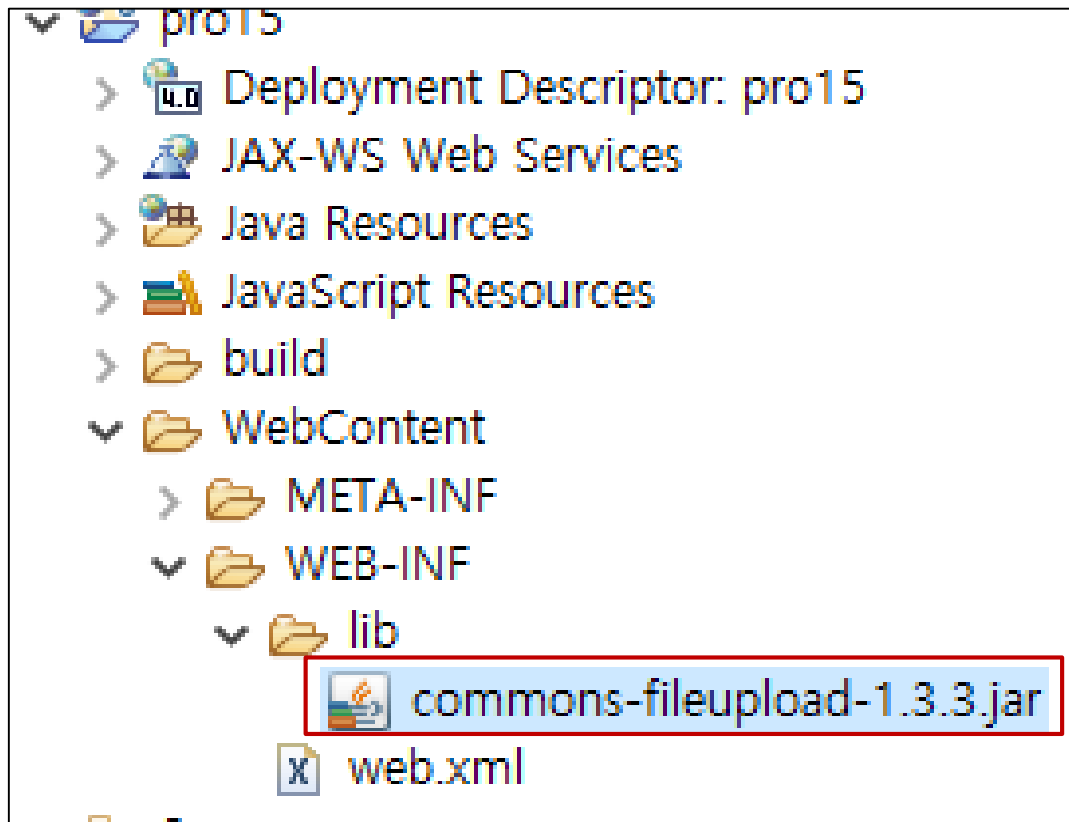
5. Commons 라이브러리

6. commons-fileupload-1.3.3-bin에서 commons-fileupload-1.3.3.jar 파일을 복사



5. Commons 라이브러리

7. 프로젝트 pro15의 WEB-INF 하위에 있는 lib 폴더에 붙여 넣습니다.



5. Commons 라이브러리

1.2 commons-io-2.6.jar 파일 설치

1. 다음 링크로 접속한 후 commons-io-2.6-bin.zip을 클릭해 다운로드.

https://commons.apache.org/proper/commons-io/download_io.cgi

Apache Commons IO 2.6 (requires JDK 1.7+)

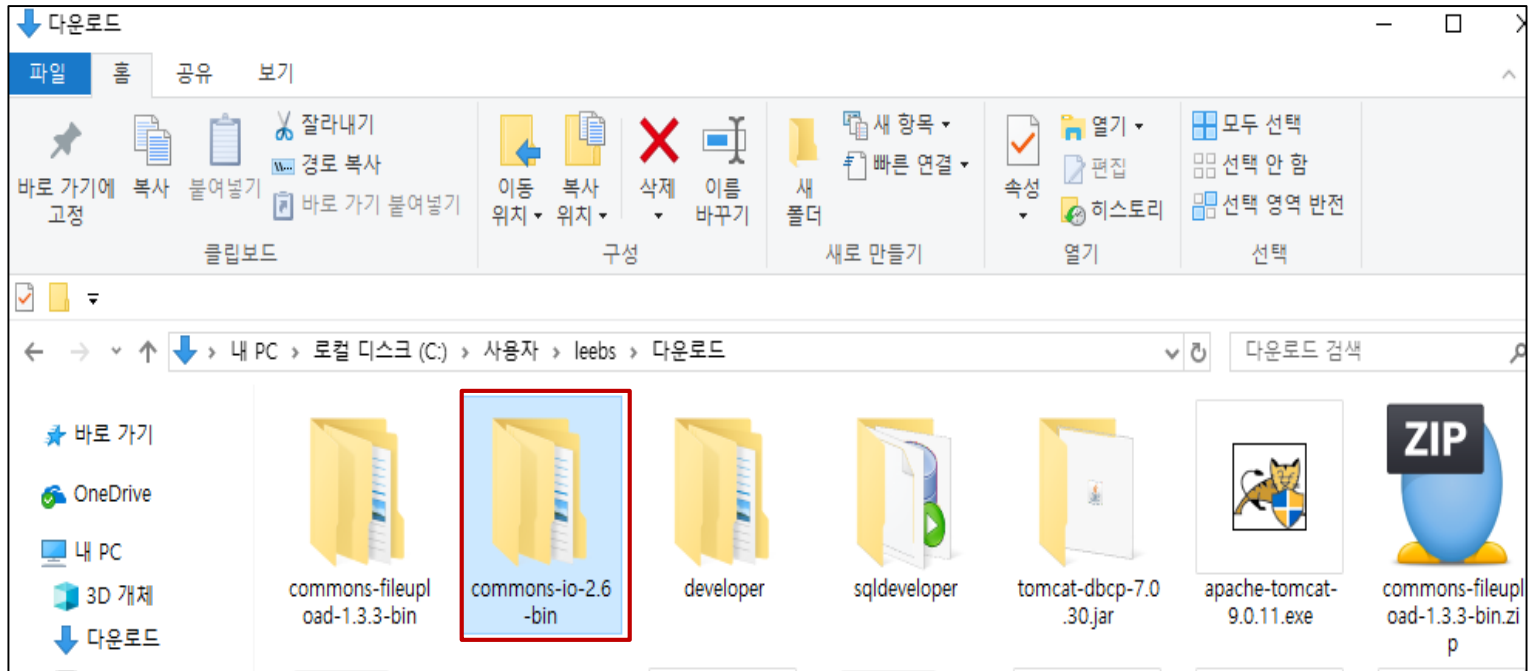
Binaries

[commons-io-2.6-bin.tar.gz](#)

[commons-io-2.6-bin.zip](#)

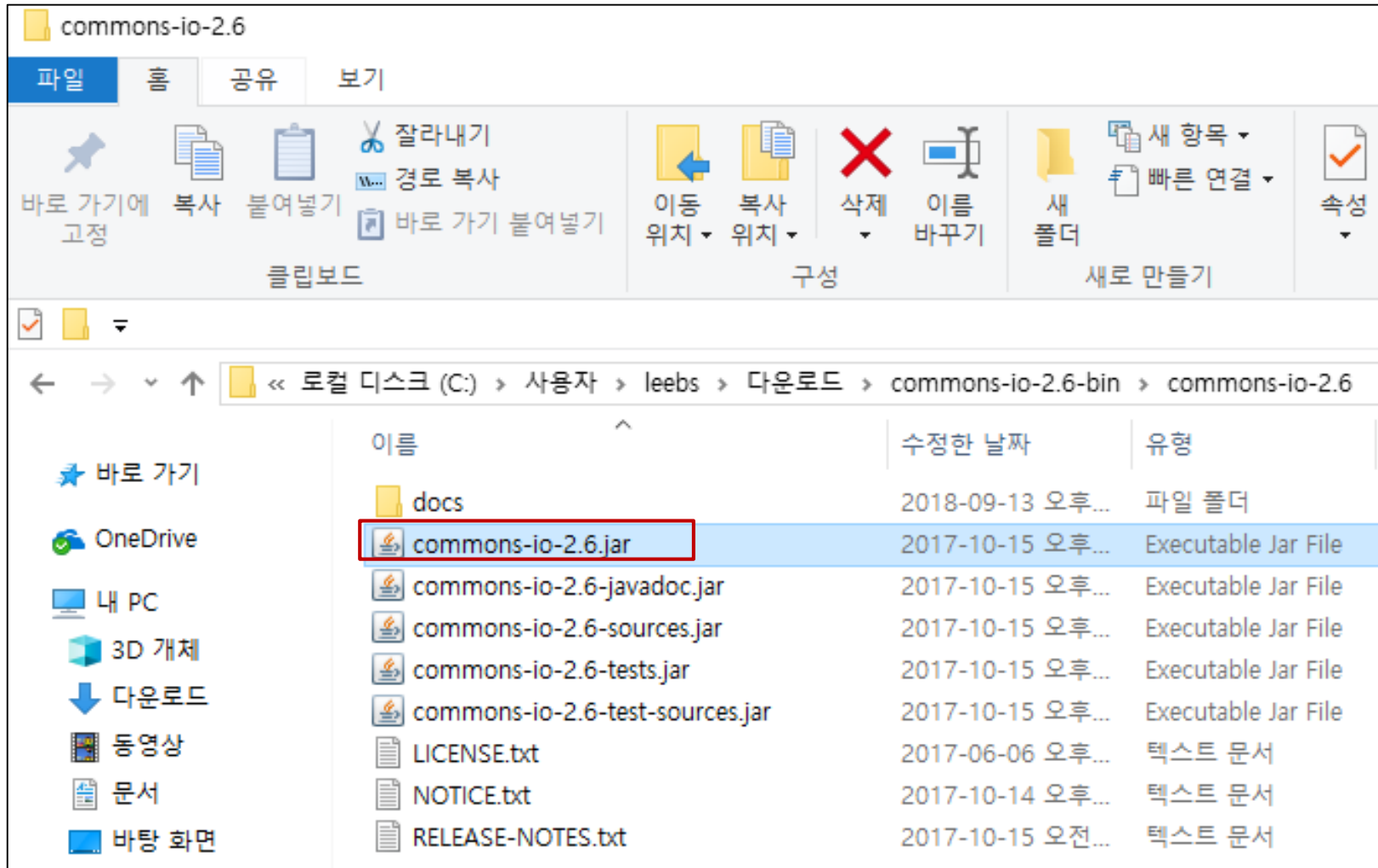
5. Commons 라이브러리

2. 로컬 PC의 여러분이 원하는 폴더에 zip 파일의 압축을 푼다.

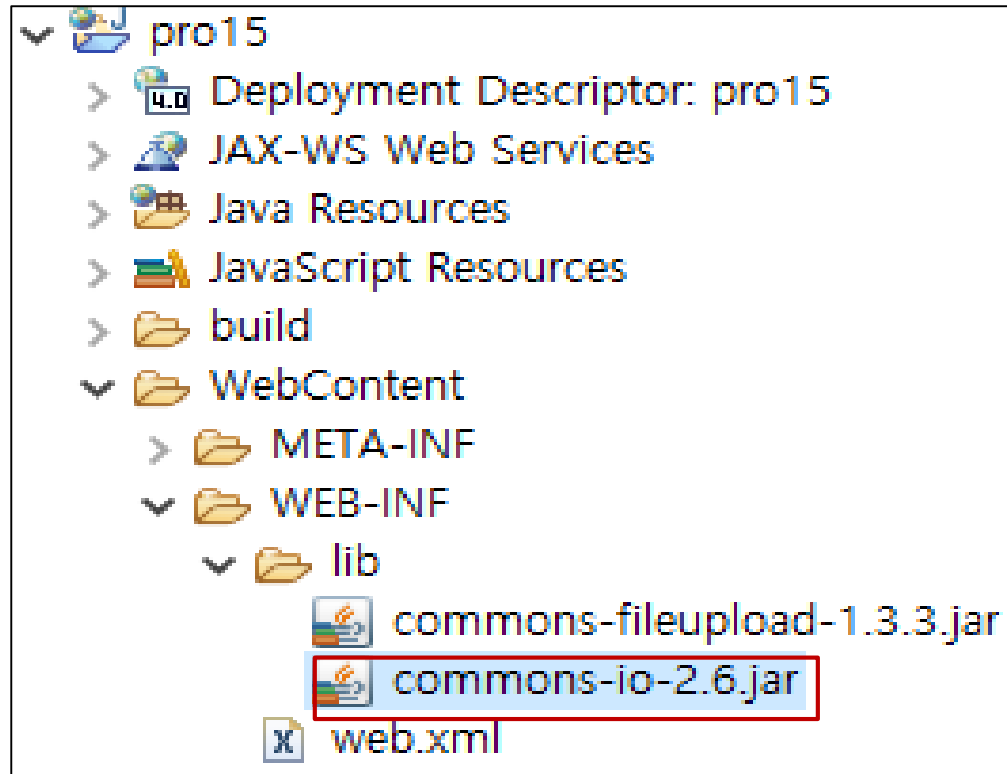


5. Commons 라이브러리

3. commons-io-2.7-bin 폴더로 이동한 후 commons-io-2.6.jar 파일을 복사해 이클립스 프로젝트의 WEB-INF/lib 폴더에 붙여 넣는다.



5. Commons 라이브러리



5. Commons 라이브러리

1.3 파일 업로드 관련 API

DiskFileItemFactory 클래스가 제공하는 메서드

메서드	기능
setRepository()	파일을 저장할 디렉토리를 설정합니다.
setSizeThreshold()	최대 업로드 가능한 파일 크기를 설정합니다.

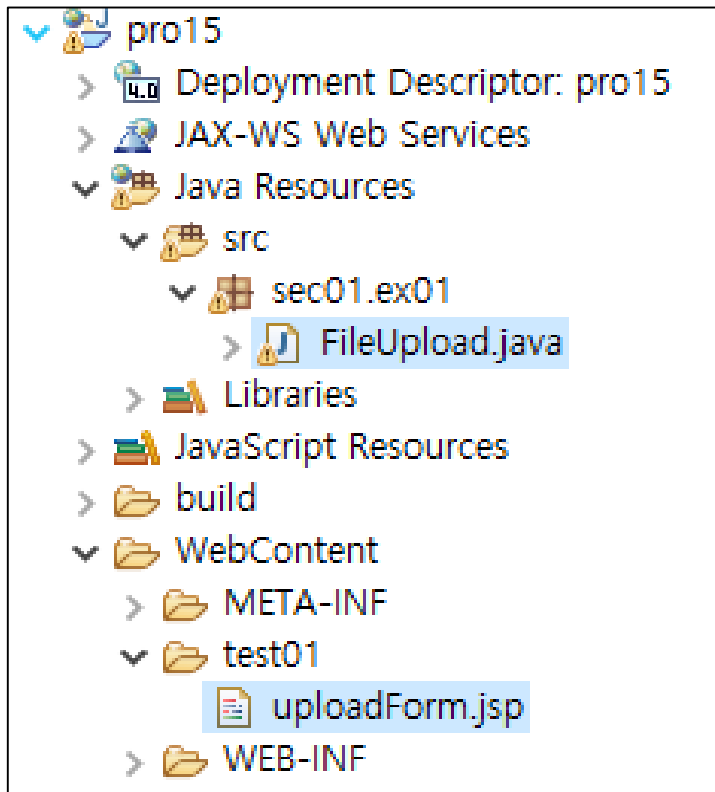
ServletFileUpload 클래스가 제공하는 메서드

메서드	기능
parseRequest()	전송된 매개변수를 List 객체로 얻습니다.
getItemIterator()	전송된 매개변수를 Iterator 타입으로 얻습니다

5. Commons 라이브러리-fileUpload

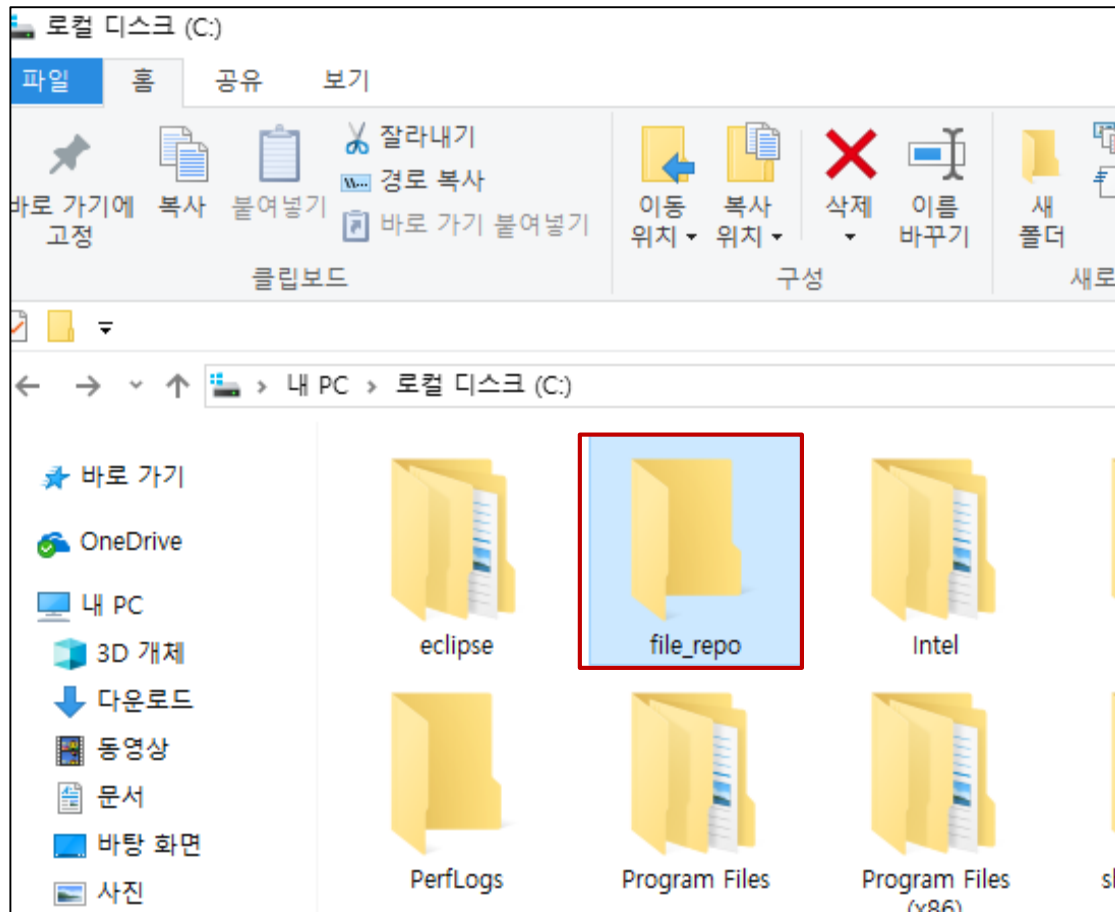
1.4 JSP 페이지에서 파일 업로드

1. sec01.ex01 패키지를 만들고 FileUpload 클래스를 생성. 또 test01 폴더를 생성하고실습 파일 uploadForm.jsp를 추가.



5. Commons 라이브러리-fileUpload

2. 파일을 업로드할 때 사용할 저장소를 다음과 같이 C 드라이브 아래에 file_repo로 작성



5. Commons 라이브러리-fileUpload

3. uploadForm.jsp를 다음과 같이 작성

코드 15-1 pro15/WebContent/test01/uploadForm.jsp

...

<body>

<form action="\${contextPath}/upload.do"

method="post" enctype="multipart/form-data" >

파일1: <input type="file" name="file1" >

파일2: <input type="file" name="file2" >

매개변수1: <input type="text" name="param1" >

매개변수2: <input type="text" name="param2" >

매개변수3: <input type="text" name="param3" >

<input type="submit" value="업로드" >

</form>

</body>

서블릿에 요청해 파일을 업로드합니다.

파일 업로드 시 반드시 encType을 multipart/form-data로 설정해야 합니다.

5. Commons 라이브러리-fileUpload

4. 파일 업로드를 처리하는 서블릿인 FileUpload 클래스 작성.

```
...  
public private void doHandle(HttpServletRequest request, HttpServletResponse response)  
throws ServletException, IOException {  
    request.setCharacterEncoding("utf-8");  
    String encoding="utf-8";  
    File currentDirPath =new File("C:\\file_repo");  
    DiskFileItemFactory factory = new DiskFileItemFactory();  
    factory.setRepository(currentDirPath );  
    factory.setSizeThreshold(1024*1024);  
    ServletFileUpload upload=new ServletFileUpload(factory);  
    try{  
        List items = upload.parseRequest(request);  
        for(int i=0; i < items.size();i++) {  
            FileItem fileItem = (FileItem) items.get(i);  
            if(fileItem.isFormField()) {  
                System.out.println(fileItem.getFieldName()+ "=" +fileItem.getString(encoding));  
            }else{  
                System.out.println("매개변수이름:"+fileItem.getFieldName());  
                System.out.println("파일이름:"+fileItem.getName());  
                System.out.println("파일크기:"+fileItem.getSize( ) + "bytes");  
                if(fileItem.getSize() > 0) {  
                    int idx = fileItem.getName().lastIndexOf("\\");  
                    if(idx ==-1) {  
                        idx = fileItem.getName().lastIndexOf("/");  
                    }  
                }  
            }  
        }  
    }  
}
```

업로드할 파일 경로를 지정합니다.

파일 경로를 설정합니다.

최대 업로드 가능한 파일 크기를 설정합니다.

request 객체에서 매개변수를 List로 가져옵니다.

파일 업로드창에서 업로드된 항목들을 하나씩 가져옵니다.

폼 필드이면 전송된 매개변수 값을 출력합니다.

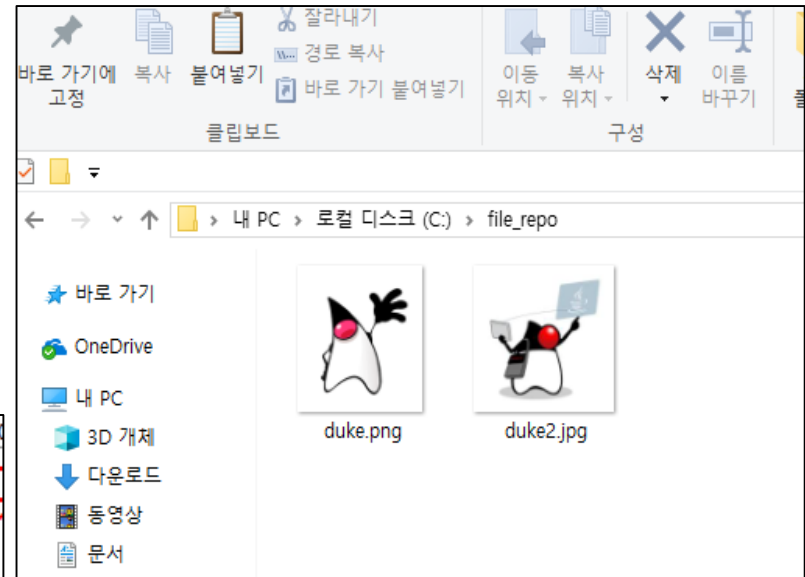
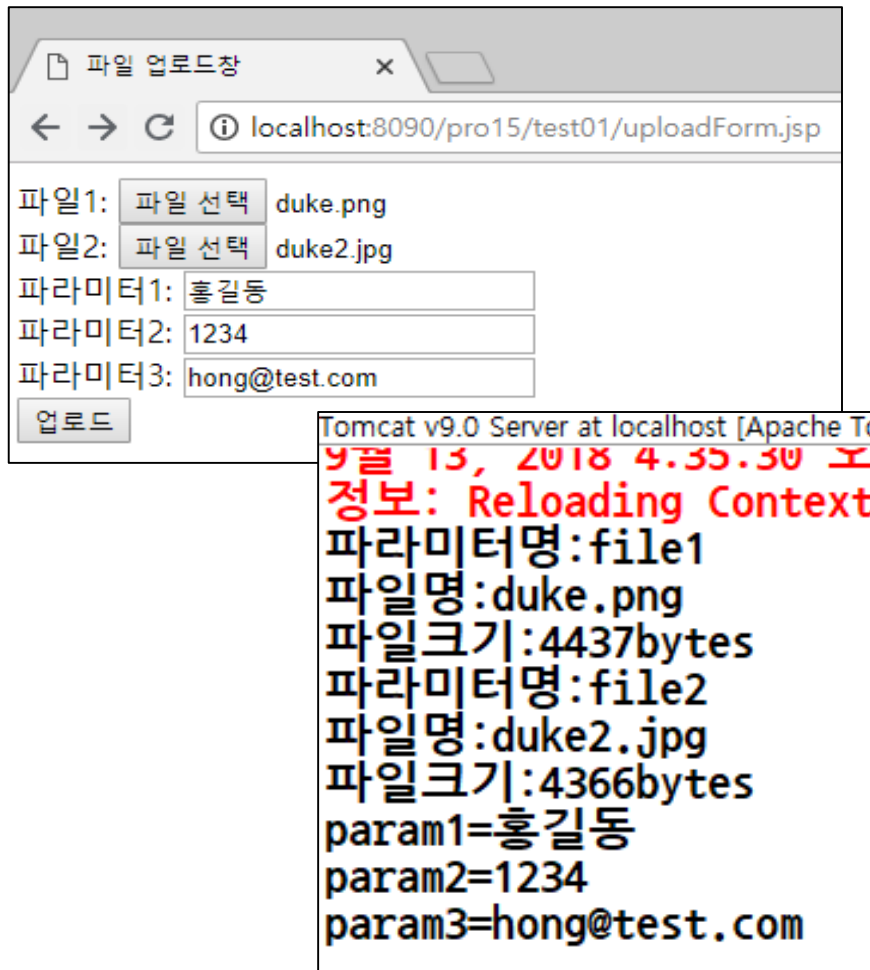
5. Commons 라이브러리-fileUpload

```
    }  
    String fileName = fileItem.getName().substring(idx+1);  
    File uploadFile = new File(currentDirPath + "\\\" + fileName);  
    fileItem.write(uploadFile);  
} //end if  
} //end if  
} //end for  
} catch(Exception e) {  
    e.printStackTrace();
```

- 폼 필드가 아니면 파일 업로드 기능을 수행합니다.
- 업로드한 파일 이름을 가져옵니다.
- 업로드한 파일 이름으로 저장소에 파일을 업로드합니다.

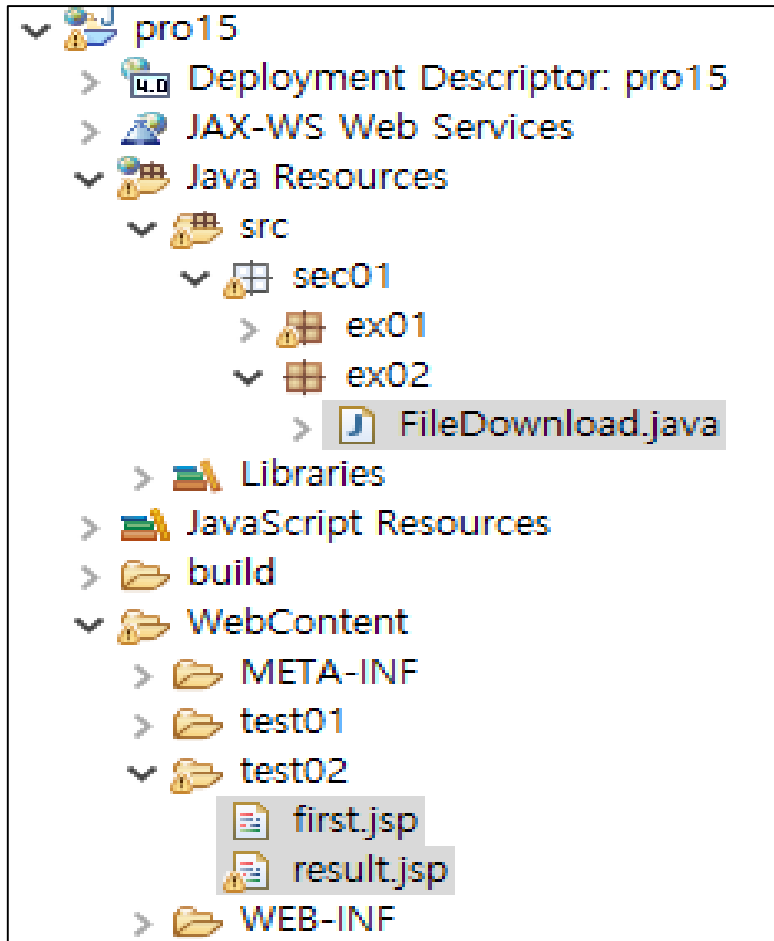
5. Commons 라이브러리-fileUpload

5. uploadForm.jsp로 파일 업로드 요청.
6. 2번 과정에서 만든 파일 저장소(C:\file_repo)에 업로드된 파일들을 볼 수 있다.
7. 이클립스의 Console 탭에서 업로드한 매개변수 정보와 파일 정보가 출력된 것을 확인 가능.



5. Commons 라이브러리- fileDownload

1. 다음과 같이 sec01.ex02 패키지를 만들고 FileDownload 서블릿을 생성합니다. 이어서 test02 폴더를 만들고 실습 파일 first.jsp와 result.jsp를 추가합니다.



5. Commons 라이브러리-fileUpload

2. 첫 번째 JSP에서 다운로드할 이미지 파일 이름을 두 번째 JSP로 전달하도록 first.jsp를 작성

코드 15-3 pro15/WebContent/test01/first.jsp

...

<body>

<form method="post" action="result.jsp" >

<input type="hidden" name="param1" value="duke.png" />

<input type="hidden" name="param2" value="duke2.jpg" />

<input type="submit" value="이미지 다운로드">

</form>

</body>

다운로드할 파일 이름을 매개변수로 전달합니다
(duke.png나 duke2.png가 아닌 다른 파일을 업로드
했다면 해당 파일 이름으로 수정하세요).

5. Commons 라이브러리-fileUpload

3. 두 번째 JSP인 result.jsp를 다음과 같이 작성.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    isELIgnored="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="contextPath" value="${pageContext.request.contextPath}" />
<%
    request.setCharacterEncoding("utf-8");
%>
<html>
<head>
    <meta charset="UTF-8">
    <c:set var="file1" value="${param.param1}" />
    <c:set var="file2" value="${param.param2}" />

    <title>이미지 파일 출력하기</title>
</head>
<body>
    매개변수 1 :
    <c:out value="${file1}" /><br>
    매개변수 2 :
    <c:out value="${file2}" /><br>
```

다운로드할 파일 이름을 가져옵니다.

5. Commons 라이브러리-fileUpload

```
<c:if test="${not empty file1 }">
```

```
  <br>
```

```
</c:if>
```

```
<br>
```

파일 이름으로 서블릿에서 이미지를 다운로드해 표시합니다.

```
<c:if test="${not empty file2 }">
```

```
  <br>
```

```
</c:if>
```

파일 이름으로 서블릿에서 이미지를 다운로드해 표시합니다.

```
파일 내려받기 :<br>
```

```
<a href="${contextPath}/download.do?fileName=${file2}">
```

이미지를 파일로 다운로드합니다.

```
  파일 내려받기 </a><br>
```

```
</body>
```

```
</html>
```

5. Commons 라이브러리-fileUpload

4. 파일 다운로드 기능을 할 서블릿인 FileDownload 클래스를 다음과 같이 작성.

```
package sec02.ex01;

...

protected private void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html; charset=utf-8");
    String file_repo="C:\\file_repo";
    String fileName = (String)request.getParameter("fileName");
    System.out.println("fileName="+fileName);
    OutputStream out = response.getOutputStream();
    String downFile=file_repo+"\\ "+fileName;
    File f=new File(downFile);
    response.setHeader("Cache-Control","no-cache");
    response.addHeader("Content-disposition", "attachment; fileName="+fileName);
}
```

매개변수로 전송된 파일 이름을 읽어옵니다.

response에서 OutputStream 객체를 가져옵니다.

파일을 다운로드할 수 있습니다.

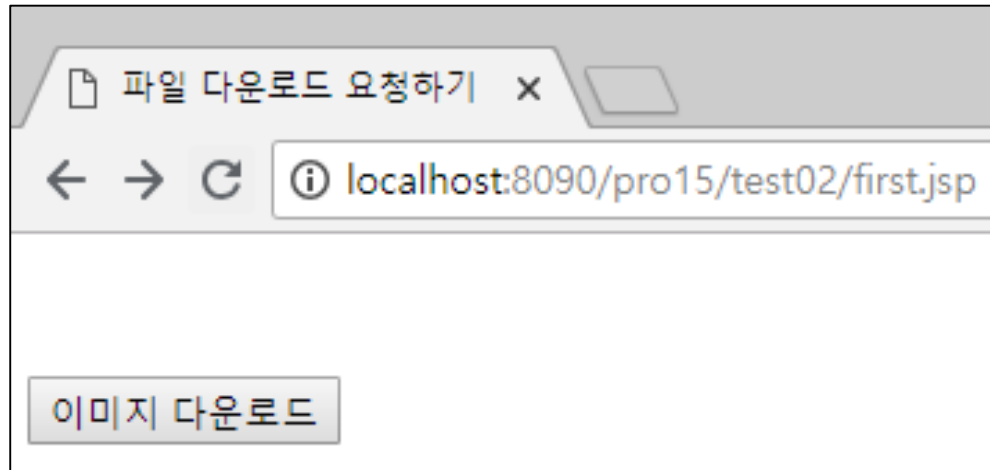
5. Commons 라이브러리-fileUpload

```
FileInputStream in=new FileInputStream(f);
byte[] buffer=new byte[1024*8];
while(true) {
    int count=in.read(buffer);
    if(count==-1)
        break;
    out.write(buffer,0,count);
}
in.close();
out.close();
}
```

버퍼 기능을 이용해 파일에서 버퍼로 데이터를 읽어와 한꺼번에 출력합니다.

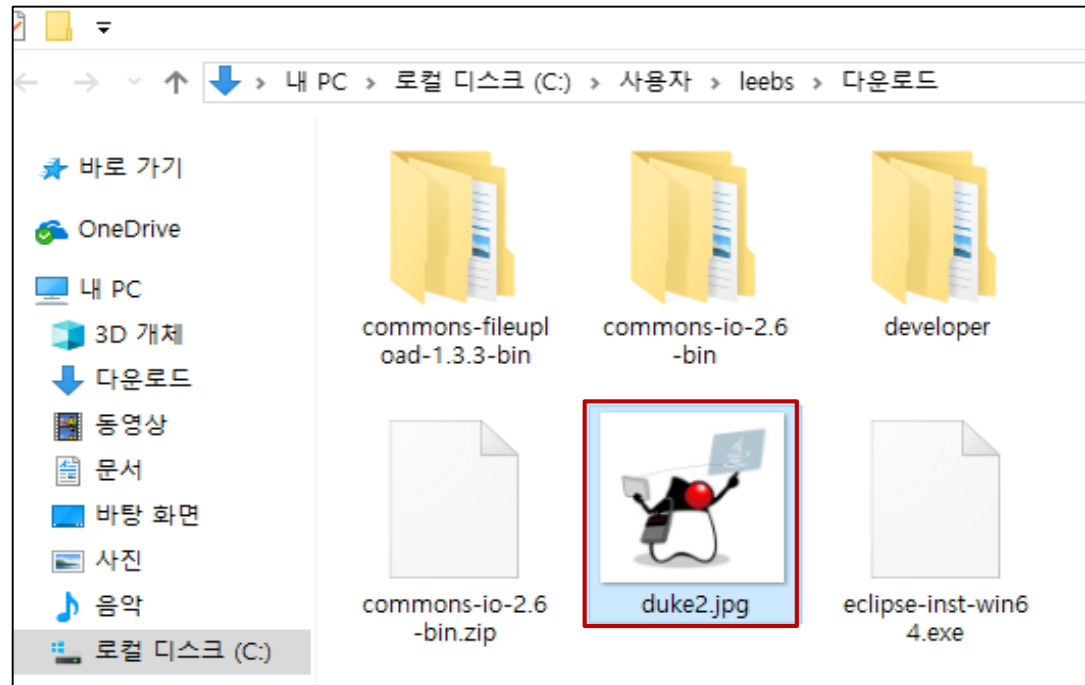
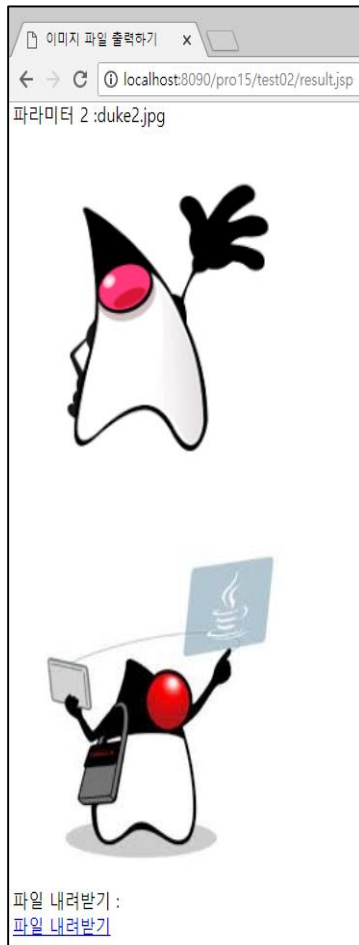
5. Commons 라이브러리-fileUpload

5. <http://localhost:8090/pro15/test02/first.jsp>로 요청한 후 이미지 다운로드를 클릭



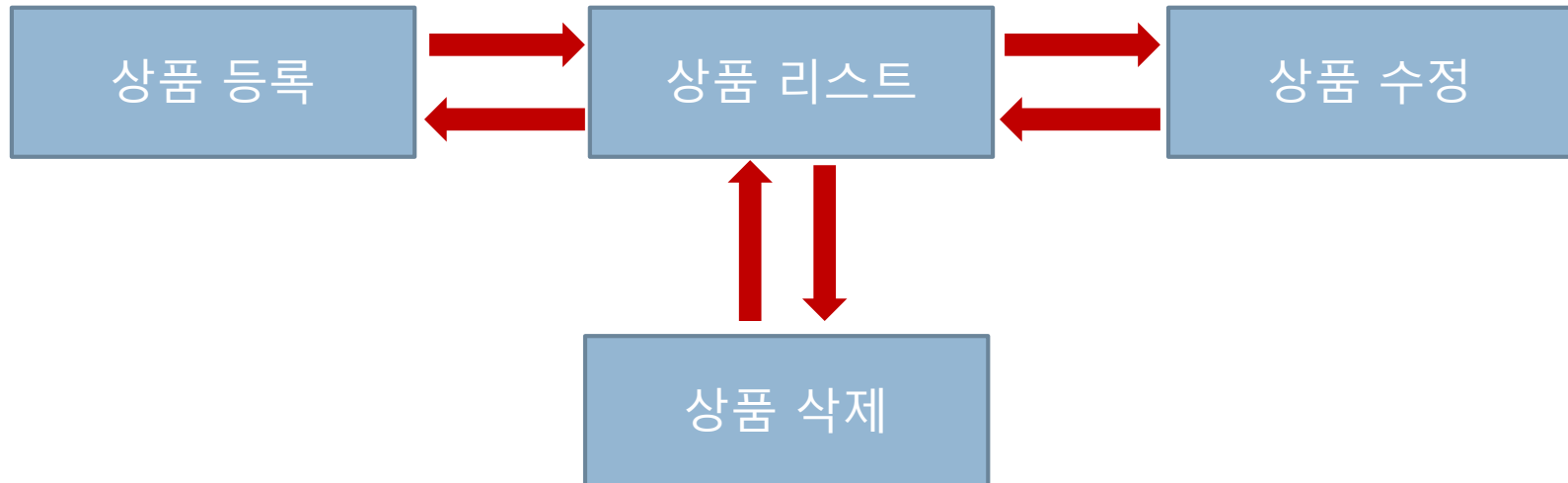
5. Commons 라이브러리-fileUpload

6. 업로드한 이미지가 브라우저에 출력되면 **파일 내려받기**를 클릭해 로컬 PC에 파일을 저장.



6. 파일 업로드 예제 작성

□ 쇼핑몰 관리자 페이지



6. 파일 업로드 예제 작성

□ 데이터베이스 구현

컬럼명	데이터 타입	제약조건
code	number	primary key
name	varchar2(100)	not null
price	number	
pictureurl	varchar2(50)	
description	varchar2(1000)	

6. 파일 업로드 예제 작성

□ vo 클래스(src/vo)

파일명	설명
ProductVO.java	상품 정보를 저장하는 클래스

□ dao 클래스(src/dao)

파일명	설명
ProductDao.java	데이터베이스 테이블과 연동해서 작업하는 데이터베이스 처리 클래스

□ 서블릿 클래스(src/controller)

파일명	설명	요청 URL패턴
ProductListServlet.java	상품 전체 정보를 DB에서 얻어 옴	productList.do
ProductWriteServlet.java	입력된 상품 정보를 DB에 추가	productWrite.do
ProductUpdateServlet.java	입력된 정보로 DB 상품 저오 수정	productUpdate.do
ProductDeleteServlet.java	DB에 상품 정보 삭제	productDelete.do

6. 파일 업로드 예제 작성

□ 자바 스크립트(WebContnets/script)

파일명	설명
product.js	폼에 입력된 데이터 유효성 검사

□ css 스크립트(WebContnets/css)

파일명	설명
shopping.css	폼에 입력된 데이터 유효성 검사

□ JSP 파일(WebContent/product)

파일명	설명
productList.jsp	상품 리스트 페이지
productWrite.jsp	상품 등록 페이지
productUpdate.jsp	상품 수정 페이지
productDelete.jsp	상품 삭제 페이지

6. 파일 업로드 예제 작성- 실행 결과

앱 Gmail YouTube 지도 hadoop 1강 Cento... 부산IT교육센터 test.html STEP_관리자 [알고리즘] 코딩, 알... DNA codon table -...

»

상품 리스트 - 관리자 페이지

상품 등록

번호	이름	가격	수정	삭제
2	데이터 베이스	25000 원	상품 수정	상품 삭제
1	java	300 원	상품 수정	상품 삭제

상품 등록 - 관리자 페이지

상 품 명	<input type="text"/>
가 격	<input type="text"/> 원
사 진	<div>파일 선택 선택된 파일 없음</div> <div>(주의사항 : 이미지를 변경하고자 할때만 선택하시오)</div>
설 명	<div></div>

등록

다시작성

목록

상품 수정 - 관리자 페이지

누구도 알려주지 않았던 시스템 개발 현장의 128가지 해결책

IT 아키텍트가 하지 말아야 할 128가지

설계, 방법론, 구축·테스트, 운용, 보안
나케이스스템즈 역량 1 회석기 옮김

**상
품
명**

데이터 베이스

**가
격**

25000 원

**사
진**

**설
명**

파일 선택 선택된 파일 없음
(주의사항 : 이미지를 변경하고자 할때만 선택하시오)

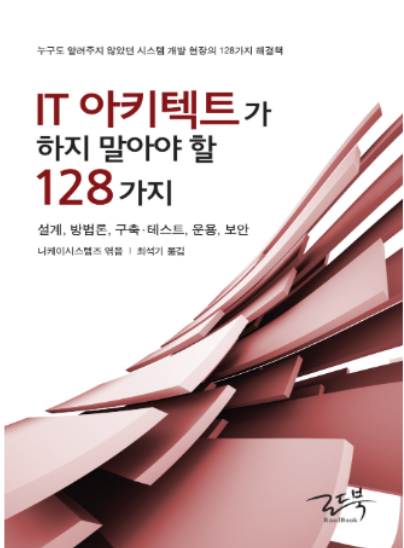
오라클 데이터베이스 사용법 설명하고 있음
환경설정, select 구문 사용법, DDL, DML 사용법에 대하여 설명하고 있다

수정

다시작성

목록

상품 삭제 - 관리자 페이지

	상 품 명	데이터 베이스
	가 격	25000원
	설 명	오라클 데이터베이스 사용법 설명하고 있음 환경설정, select 구문 사용법, DDL, DML 사용법에 대하여 설명하고 있다

[삭제](#)[목록](#)