7장 파일 업로드와 간의 웹하드 구현

동의과학대학교
컴퓨터정보과
김진숙

내용

1. 파일 업로드
2. 파일 다운로드
3. 배포

파일 업로드 실습 순서

1. 데이터베이스에 테이블 작성
2. 업로드 파일 저장 폴더 생성
3. 이클립스의 퍼블리싱 기능 해제
4. 업로드 폼 작성
5. 업로드 처리 서블릿 작성

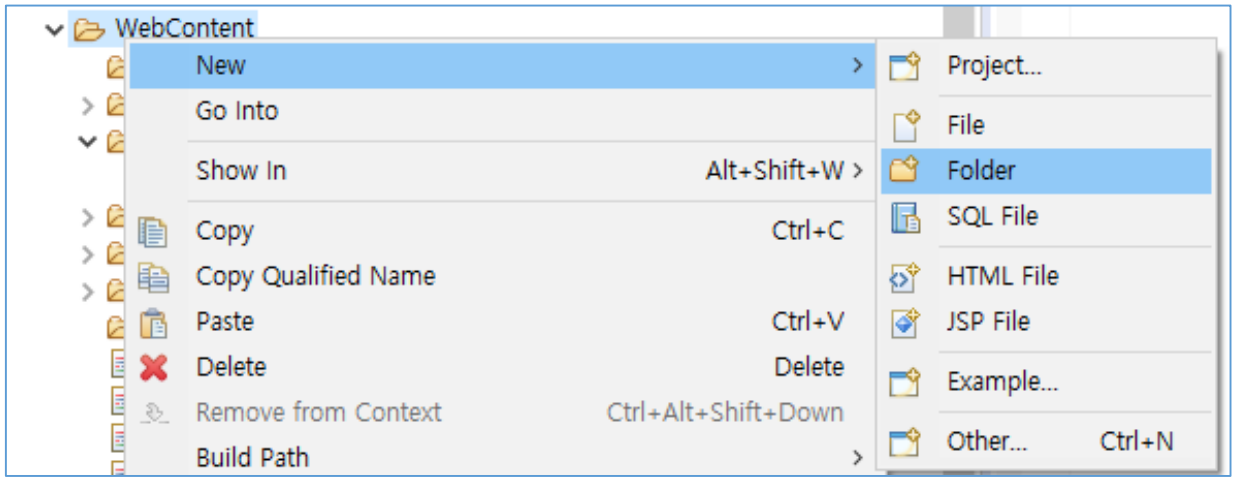
1. 데이터베이스에 테이블 작성

- 데이터베이스 테이블(UPLOADFILES)

```
CREATE TABLE uploadfiles(
    num INT PRIMARY KEY auto_increment,
    author VARCHAR(100),
    title VARCHAR(100),
    filename VARCHAR(100)
);
COMMIT;
```

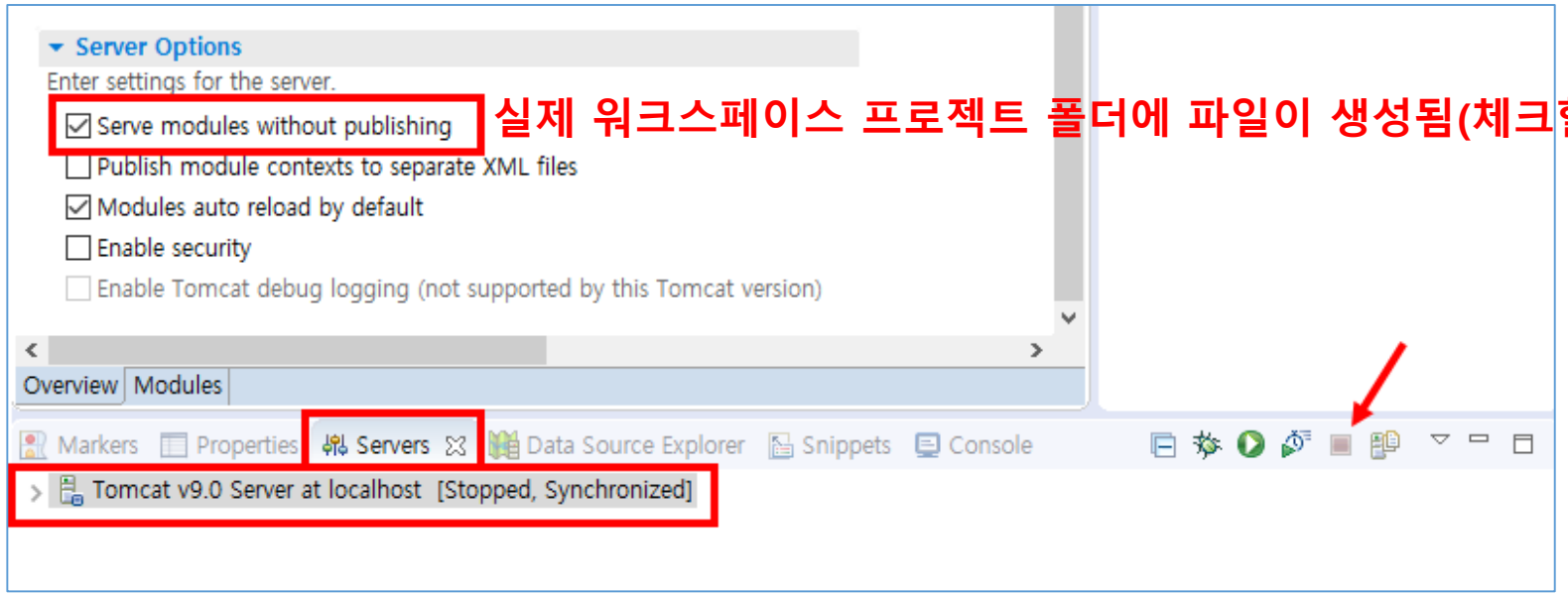
2. 업로드 파일 저장 폴더 생성

- webapp 하위 폴더로 uploadfiles 생성



3. 이클립스 퍼블리싱 기능 해제

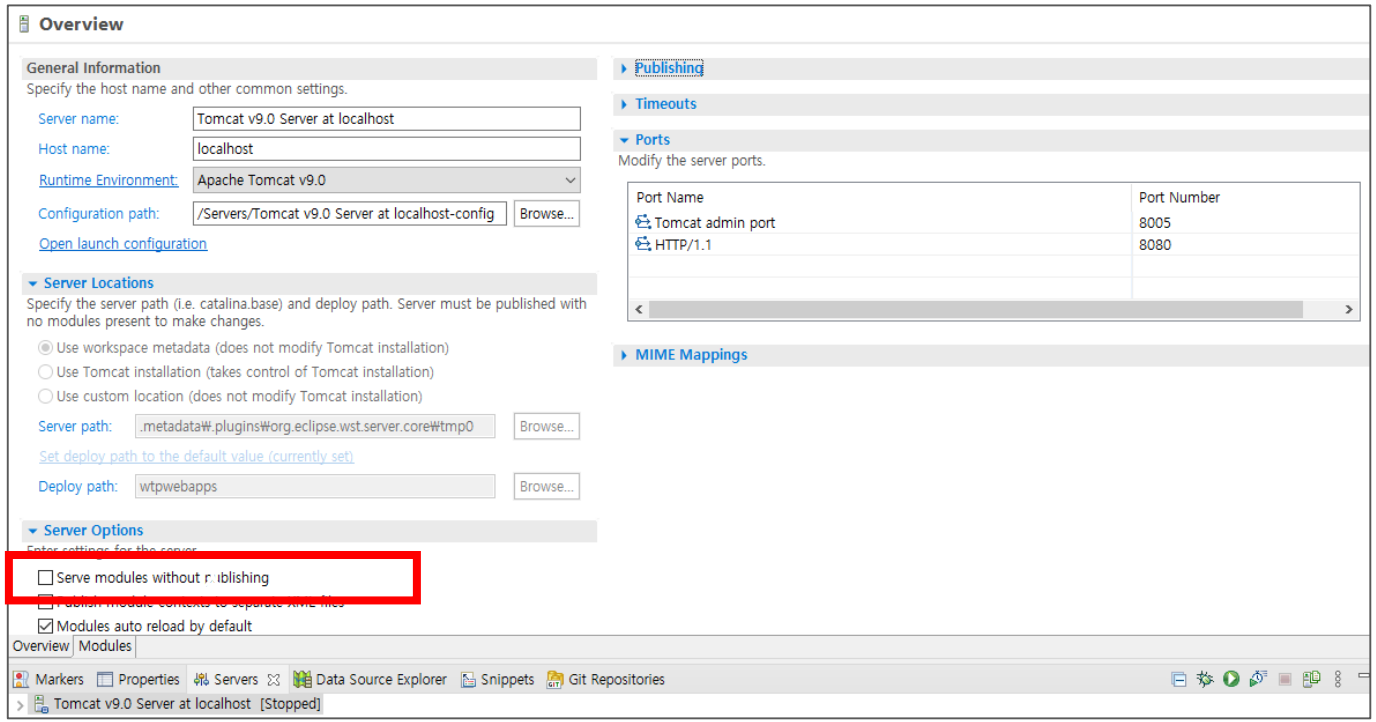
- 퍼블리싱 기능 :
  - 웹 애플리케이션을 실행할 때, 이클립스는 웹 애플리케이션의 원본 폴더는 건드리지 않고, 원본을 복사한 실행용 폴더를 다른 위치에서 만들어서 사용
  - 파일 업로드를 할 때에는 물리적으로 저장된 폴더를 찾기 어렵다는 문제가 발생하여 해제



실제 워크스페이스 프로젝트 폴더에 파일이 생성됨(체크할 것)

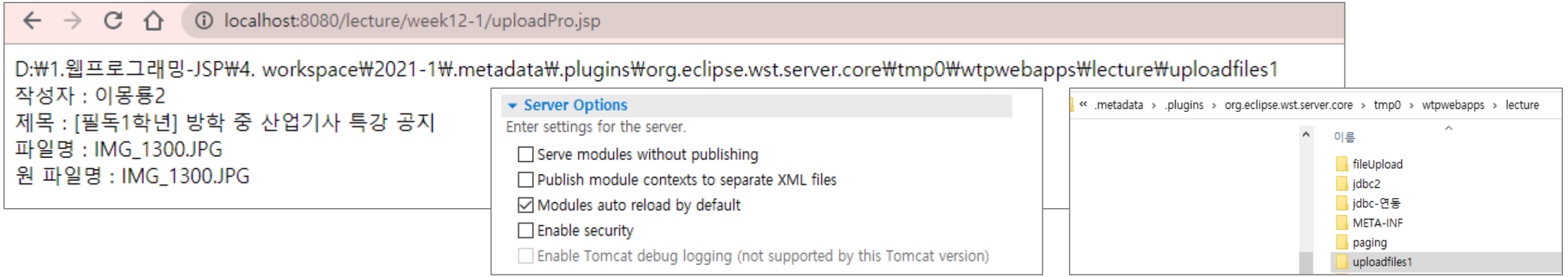
서버 설정

- 이클립스의 퍼블리싱 기능
  - 톰캣 서버가 동작 중이라면 중지하고 설정
  - 퍼블리싱(publishing)
    - 웹 애플리케이션을 실행할 때, 웹 애플리케이션의 원본 폴더는 건드리지 않고, 원본을 복사한 실행용 폴더를 만들어서 사용(이클립스 가상환경에서 실행)



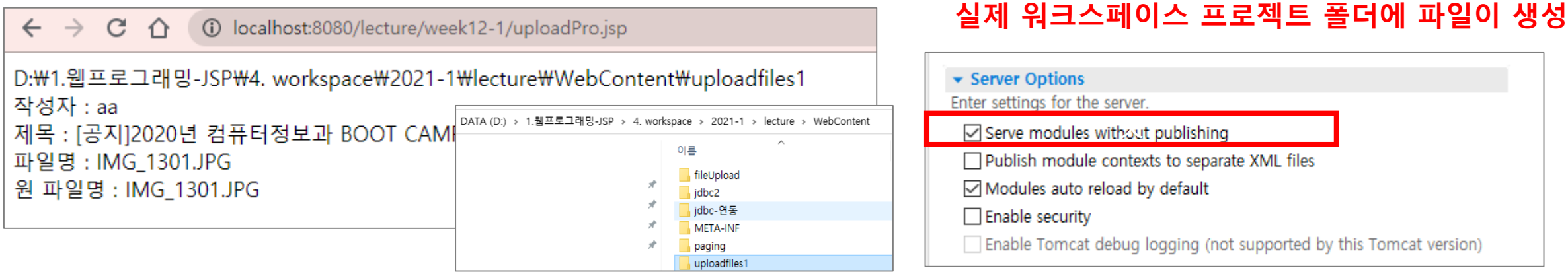
서버 설정

1. Serve modules without publishing 체크 안함 **가상 환경을 사용한다는 의미**



2. Server modules without publishing 체크(사용할 것)
   - 메인 프로젝트는 의존하는 모듈(프로젝트 등)을 퍼블리싱 하지 않고 직접 참조

실제 워크스페이스 프로젝트 폴더에 파일이 생성됨



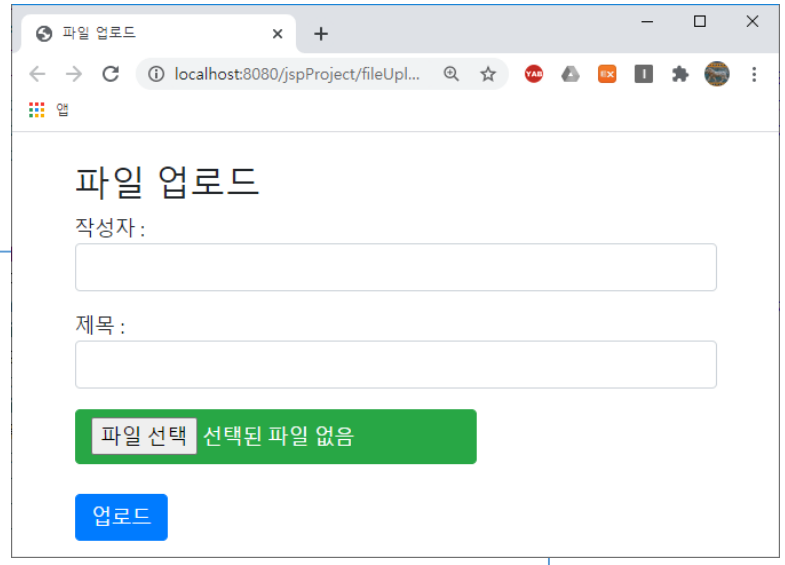
4. 업로드 폼 작성

[reference] : https://lena-chamna.netlify.app/post/http_multipart_form-data/

- 폼의 enctype 주요 속성값
  - application/x-www-form-urlencoded
    - 요청 Http의기본 Content-Type의 기본값으로 모든 문자들을 서버로 보내기 전 인코딩되는 것을 명시
    - 메시지 body에 들어가는 데이터 타입을 명시, 기본 값
  - **multipart/form-data**
    - <form> 요소가 파일이나 이미지를 서버로 전송할 때 반드시 사용
    - POST 전송방식일 때만 사용 가능
    - 한 폼 내의 두개의 input 요소가 다른 Content-type으로 데이터를 전송할 때 사용
      - 예를 들어 사진 파일을 전송할 때 하나는 사진 설명 데이터, 두번째는 사진 자체를 위한 이진 파일 데이터
      - 하나의 request body 내에 이 두 종류의 데이터를 구분해 넣어주는 방법

<form>에서 enctype속성을 변경해주지 않으면 기본값인 application/x-www-form-urlencoded로 데이터를 전송하게 되고 여러가지 타입의 데이터를 동시에 전송할 수 없기 때문에 텍스트 데이터만 전송하게 됨

- fileUploadForm.jsp




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>파일 업로드</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</head>
<body>
<br>
  <div class="container">
    <h3>파일 업로드</h3>
    <form action="fileUploadPro.jsp" method="post" enctype="multipart/form-data">
      <div class="form-group">
      작성자 : <input type="text" name="author" class="form-control">
      </div>
      <div class="form-group">
      제목 : <input type="text" name="title" class="form-control">
      </div>
      <input type="file" name="filename" class="btn btn-success"><br><br>
      <input type="submit" value="업로드" class="btn btn-primary">
    </form>
  </div>
</body>
</html>
```

4. 업로드 폼 작성

- 폼의 enctype 주요 속성값
  - multipart/form-data으로 지정된 폼에서 만들어진 HTTP request 메시지
    - boundary를 기준으로 여러개의 부분(Part)으로 나뉘어짐
    - 서버에서 이를 Part객체로 만들어 확인할 것임
    - request header

```
request.getContentType()
```

```
Part.getHeader("Content-Disposition");
Part.getSubmittedFileName()
```

```
POST http://localhost:8080/jspProject/fileupload HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 2049708
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryg3IbmadDo87Bmh2R
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like  Gecko)
Chrome/75.0.3770.142 Safari/537.36

------WebKitFormBoundaryg3IbmadDo87Bmh2R
Content-Disposition: form-data; name="file1"; filename="메이븐.pptx"
Content-Type: application/vnd.openxmlformats-officedocument.presentationml.presentation

ppt 파일에 대한바이너리 데이터...
```

request body

5. 업로드 처리 서블릿 작성

- Part 인터페이스
  - Part 클래스는 multipart/form-data Post 요청에서 전달된 부분 또는 폼 데이터를 처리
  - Servlet 3.0 이상 지원

    https://javaee.github.io/javaee-spec/javadocs/index.html?javax/servlet/http/package-summary.html

5. 업로드 처리 서블릿 작성

- Part 인터페이스의 주요 메서드

메소드	설명
Public String **getContentType()**	Content-Type을 리턴
public String **getName()**	파라미터명을 리턴
public String **getSubmittedFileName()**	업로드한 파일명을 리턴 servlet 3.1부터 사용 가능
public long **getSize();**	파일의 크기를 byte단위로 리턴
public void **write(String fileName)** throws IOException	임시저장되어 있는 파일 데이터를 복사하여 fileName에 지정한 경로로 저장 임시저장 되어있는 파일데이터가 메모리상에 있든 디스크에 있든 신경쓰지 않아도 됨
public void **delete()** throws IOException	임시저장된 파일 데이터를 제거 HTTP요청이 처리되고 나면 자동으로 제거되지만 그 전에 메모리나 디스크 자원을 아끼고 싶다면 수동으로 제거할 수 있음
public String **getHeader(String name)**	Part로부터 지정한 name헤더값을 리턴

5. 업로드 처리 서블릿 작성

- Part 인터페이스와 관련된 HttpServletRequest 객체 메소드

메소드	설명
Collection<Part> **getParts()** throws IOException, ServletException	multipart/form-data 타입이 제공하는 이 요청의 모든 Part 구성요소들을 얻어냄
Part **getPart(String name)** throws IOException, ServletException	요청된 Part의 이름으로 Part를 얻어냄

5. 업로드 처리 서블릿 작성

- 임시저장 경로 및 파일 크기 제한 설정
  - 파일이 저장되기 전 임시로 잠시 저장됨
  - 임시파일의 저장 경로, 파일크기 제한 등 설정 필요
  - 설정 방법
    - web.xml 에 설정
    - @MultipartConfig어노테이션을 이용한 방법

- @MultipartConfig
  - fileSizeThreshold: 파일이 디스크에 쓰여지는 사이즈를 지정할 수 있다. 크기 값은 바이트 단위(1024 * 1024 * 10 = 10MB)
  - location: 파일이 기본적으로 저장되는 디렉토리, 기본값은 ""
  - maxFileSize: 파일을 업로드할 수 있는 최대 크기, 값은 바이트 단위
  - maxRequestSize: multipart/form-data 요청에 허용되는 최대 크기, 기본값은 무제한을 의미하는 -1L

5. 업로드 처리 서블릿 작성

- web.xml 설정

```
<servlet>
    <servlet-name>FileUploadServlet</servlet-name>
    <servlet-class>servlet.FileUploadServlet</servlet-class>
    <multipart-config>
        <location>C:₩attaches</location>
        <max-file-size>-1</max-file-size>
        <max-request-size>-1</max-request-size>
        <file-size-threshold>1024</file-size-threshold>
    </multipart-config>
</servlet>
<servlet-mapping>
    <servlet-name>FileUploadServlet</servlet-name>
    <url-pattern>/fileUpload</url-pattern>
</servlet-mapping>
```

태그	설명
<multipart-config>	multipart/form-data 로 전송된 데이터에 대해 Part API로 처리할 수 있도록 하는 설정
<location>	업로드한 파일의 임시 저장 경로 지정
<max-file-size>	업로드 가능한 최대 size를 byte단위로 지정 -1인 경우 제한 없음 request.getParts()호출시 파일 크기가 이 값을 넘는 경우 IllegalStateException가 발생
<max-request-size>	전체적인 multipart 데이터 최대 size를 byte단위로 지정 -1인 경우 제한 없음
<file-size-threshold>	임시파일로 저장 여부를 결정할 데이터 크기를 byte로 지정 이값을 넘는 경우 <location>으로 지정한 경로에 임시파일로 저장, 아니면 메모리상에 가지고 있음 메모리상에 저장된 파일 데이터는 언젠가 제거되나 크기가 큰 파일을 메모리상에 적재하게 되면 서버에 부하를 줄 수 있으므로 적당한 크기를 지정해 곧바로 임시파일로 저장하는것이 좋음

5. 업로드 처리 서블릿 작성

- @MultipartConfig() 으로 설정

```
@MultipartConfig(
        location = "C:₩₩attaches",
        maxFileSize = -1,
        maxRequestSize = -1,
        fileSizeThreshold = 1024)
@WebServlet("/fileUpload")
```

- maxFileSize 설정을 통해 업로드 파일 크기 제한이 있는 경우제한을 넘기게 되면 request.getParts()호출시 다음과 같이 IllegalStateException예외가 발생

```
Collection<Part> parts = null;
try {
    parts = request.getParts();
}catch (IllegalStateException e) {
    //업로드 크기 제한을 넘겼을 경우의 처리
}
```

5. 업로드 처리 서블릿 작성

```
 1 package cs.dit;
 2 import java.io.File;
19
20 /**=========================================================================
21  * 패키지명 : cs.dit
22  * 파일명  : FileUploadServlet.java
23  * 작성자 : 김진숙
24  * 변경이력 :
25  *    2022.04.24/ 최초작성/ 김진숙
26  * 프로그램 설명 :
27  *    - 폼에서 전달된 파일ㅇ르 지정 폴더에 넣고 그 파일 정보를 DB 테이블에 insert
28 *=========================================================================*/
29 @WebServlet("/fileUpload")
30 @MultipartConfig(
31         location = "C:\files-tmp",
32         maxFileSize = 1024 * 1024 * 5,
33         maxRequestSize = 1024 * 1024 * 50
34 )
35 public class FileUploadServlet extends HttpServlet {
36     private static final long serialVersionUID = 1L;
37
38     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
   IOException {
39         System.out.println("hi!!!!");
40         response.setContentType("text/html; charset=utf-8"); //response객체에 결과값 인코딩 방법 설정
41         request.setCharacterEncoding("utf-8");       //서버로 전달되는 데이터의 인코딩 방법 설정
42         PrintWriter out = response.getWriter(); //화면출력을 위한 객체 얻기
43
44         String filename = "";
```

5. 업로드 처리 서블릿 작성

```
46          //HTTP 요청객체인 HttpServletRequest 객체로부터 Content-Type 헤더값을 꺼내어 multipart/form-data인지 확인합니다.
47          String contentType = request.getContentType();
48          if(contentType != null && contentType.toLowerCase().startsWith("multipart/")) {
49              String dir = request.getServletContext().getRealPath("/uploadFiles"); //실제 경로
50
51              File f = new File(dir);//File(String pathname) : 지정된 경로 문자열을 추상적인 경로로 변환하여 새로운 File 객체 생성
52              if(!f.exists()){//생성된 객체가 존재하지 않으면 폴더가 없는 것이므로
53                  f.mkdirs();  //해당경로에 디렉토리 생성
54              }
55              //request.getParts()메서드를 통해 여러개의 Part를 Collection에 담아 리턴
56              Collection<Part> parts = request.getParts();
57
58              for(Part p : parts) {
59                  //part의 Content-Disposition 헤더가 filename=을 포함하면 파일로 구분됨
60                  if(p.getHeader("Content-Disposition").contains("filename=")) {
61
62                      if(p.getSize()>0) {
63                          filename = p.getSubmittedFileName(); //업로드한 파일명을 리턴
64                          String filePath = dir + File.separator + filename;// File.separator : \
65                          p.write(filePath);//디스크에 파일을 쓰기
66                          p.delete(); //임시저장된 파일 데이터를 수동으로 제거. 일반적으로 HTTP 요청이 처리되고 응답을 출력하는 시점에 자동으로 제거됨
67                      }
68                  }
69              }
70          }
```

5. 업로드 처리 서블릿 작성

```
72          //DB 처리
73          String author = request.getParameter("author");
74          String title = request.getParameter("title");
75
76          String sql = "INSERT INTO UPLOADFILES VALUES(?,?,?)";
77
78          try(Connection con = getConnection();
79              PreparedStatement pstmt = con.prepareStatement(sql);
80          ) {
81              pstmt.setString(1,author);
82              pstmt.setString(2, title);
83              pstmt.setString(3, filename);
84
85              pstmt.executeUpdate();
86          } catch (Exception e) {
87              e.printStackTrace();
88          }
89      }
90
```

```
91      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Se
92          doPost(request, response);
93      }
94
95      public Connection getConnection() throws Exception {
96          //데이터베이스에 file이름 등록
97          Context initCtx = new InitialContext();
98          Context envCtx = (Context) initCtx.lookup("java:comp/env");
99          DataSource ds = (DataSource) envCtx.lookup("jdbc/jskim");
00          Connection con = ds.getConnection();
01
02          return con;
03      }
04
05  }
```

6. 파일 다운로드

- <a> 태그에 download 옵션을 사용하여 파일을 다운로드 할 수 있음

```
<body>
<br>
  <div class="container">
    <h3>파일 업로드</h3>
    <table class="table">
      <tr>
        <th>작성자</th>
        <th>제목</th>
        <th>파일명</th>
      </tr>
      <c:forEach var="dto" items="${dtos}">
        <tr>
          <td>${dto.author}</td>
          <td>${dto.title}</td>
          <td><a download href="/fileupload/uploadFiles/${dto.filename}">${dto.filename}</a></td>
        </tr>
      </c:forEach>

    </table>
  </div>
</body>
```