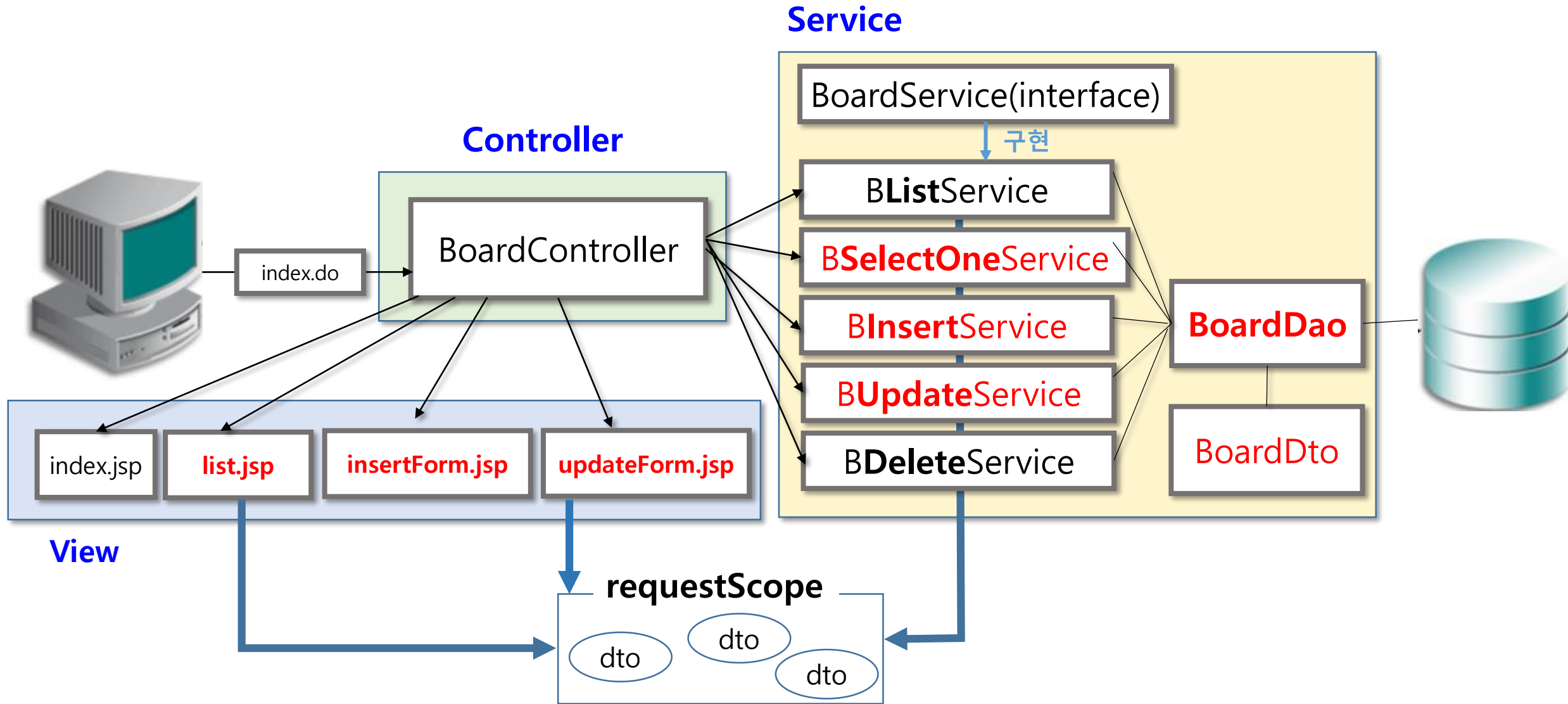


4. 파일 업로드 다운로드

동의과학대학교
컴퓨터정보과
김진숙

파일업로드-다운로드 추가하기

- 자바 패키지와 파일 변경



파일 업로드 실습 순서

1. 데이터베이스에 테이블 변경(alter table...)
2. BoardDto.java 변경(filename 추가)
3. 업로드 파일 저장 폴더 생성(/uploadfiles)
4. 이클립스의 퍼블리싱 기능 해제
5. insertForm.jsp 폼 변경
6. 업로드 처리 서블릿 작성
7. 데이터베이스 처리
8. 파일 다운로드(list.jsp)
9. 기타 변경 필요 파일

1. 데이터베이스에 테이블 변경

- 데이터베이스 테이블(board)에 하나의 컬럼(FILENAME) 추가

```
ALTER TABLE board  
ADD FILENAME VARCHAR(50)  
AFTER REGDATE;
```

- 또는 다른 테이블 board1 생성(이전 프로젝트의 호환성을 위해)
 - 이전 프로젝트를 복사하여 사용(board-upload)

```
CREATE TABLE BOARD1(  
    BCODE INT AUTO_INCREMENT PRIMARY KEY,  
    SUBJECT VARCHAR(100),  
    CONTENT TEXT,  
    WRITER VARCHAR(50),  
    REGDATE DATE,  
    FILENAME VARCHAR(50)  
);
```

2. BoardDto.java 변경

- 데이터베이스 테이블(board)에 하나의 컬럼(FILENAME) 추가

```
package cs.dit.board;

import java.sql.Date;

public class BoardDto {
    private int bcode;
    private String subject;
    private String content;
    private String writer;
    private Date regDate;
    private String filename;
```

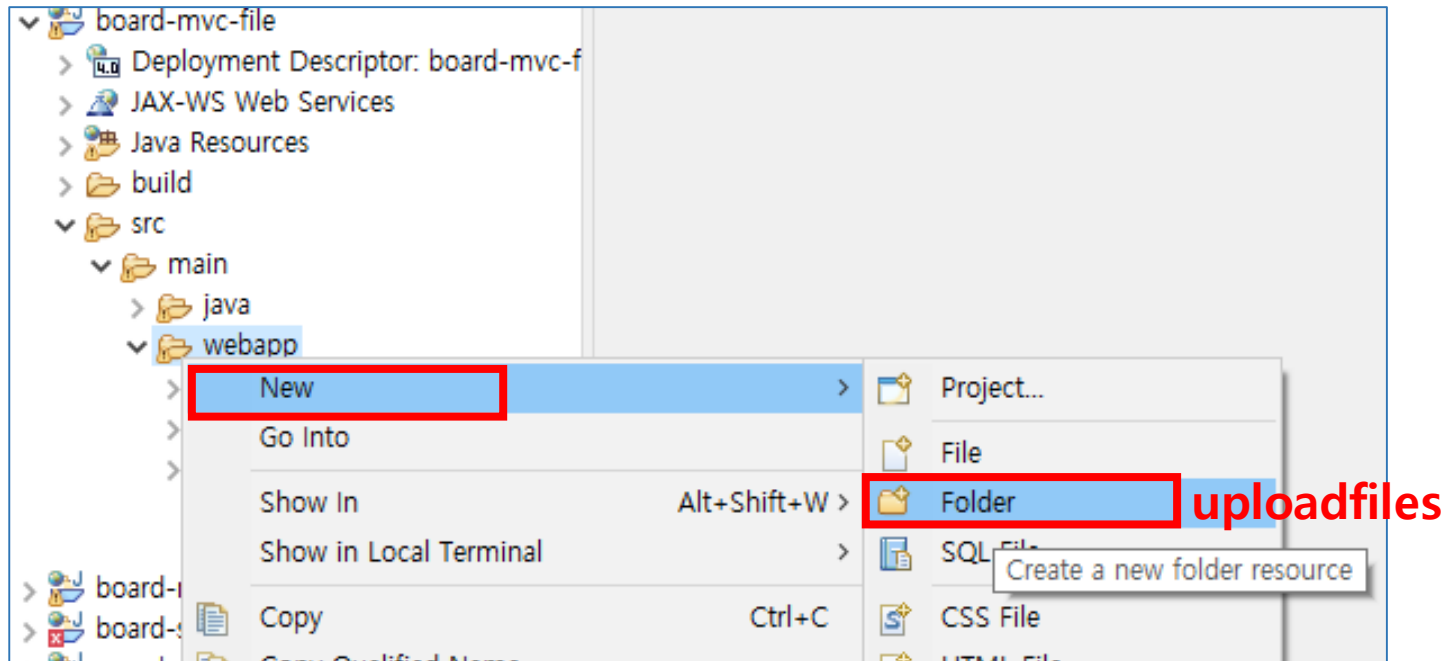
```
    public String getFilename() {
        return filename;
    }
    public void setFilename(String filename) {
        this.filename = filename;
    }
    public int getBcode() {
        return bcode;
    }
}
```

```
public BoardDto(int bcode, String subject, String content, String writer,
                Date regDate, String filename) {
    super();
    this.bcode = bcode;
    this.subject = subject;
    this.content = content;
    this.writer = writer;
    this.regDate = regDate;
    this.filename = filename;
}
```

추가

3. 업로드 파일 저장 폴더 생성

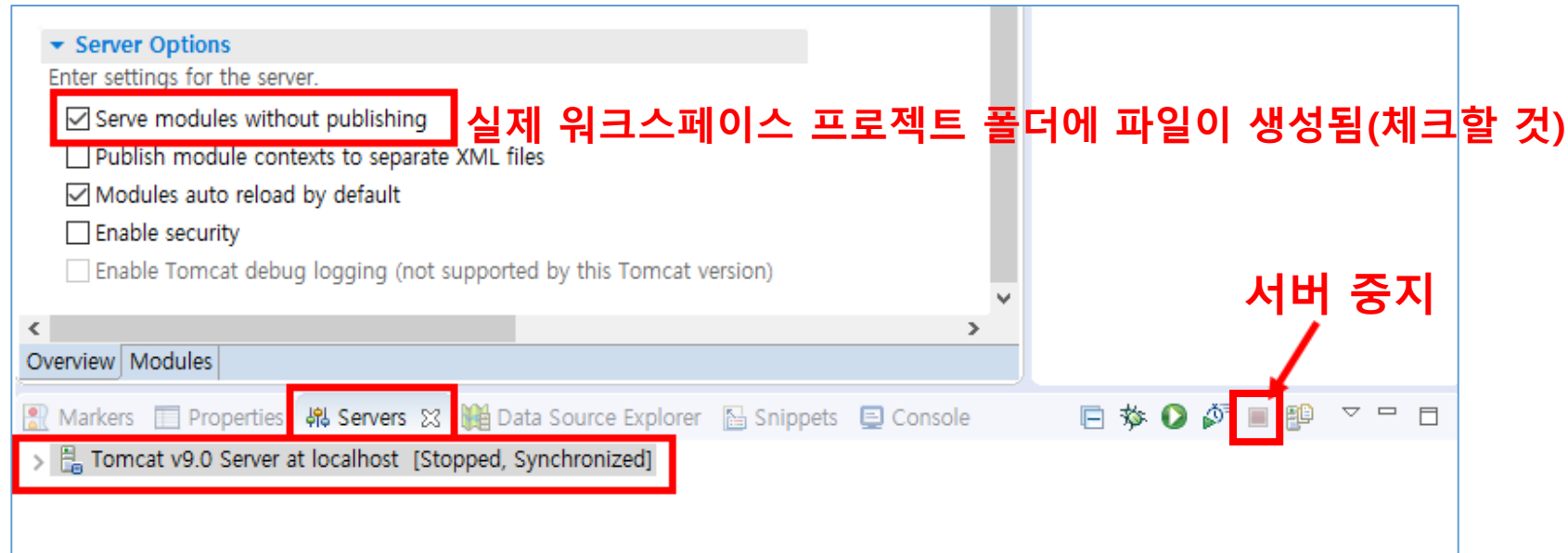
- webapp 하위 폴더로 업로드 파일 저장 폴더 **uploadfiles** 생성



4. 이클립스 퍼블리싱 기능 해제

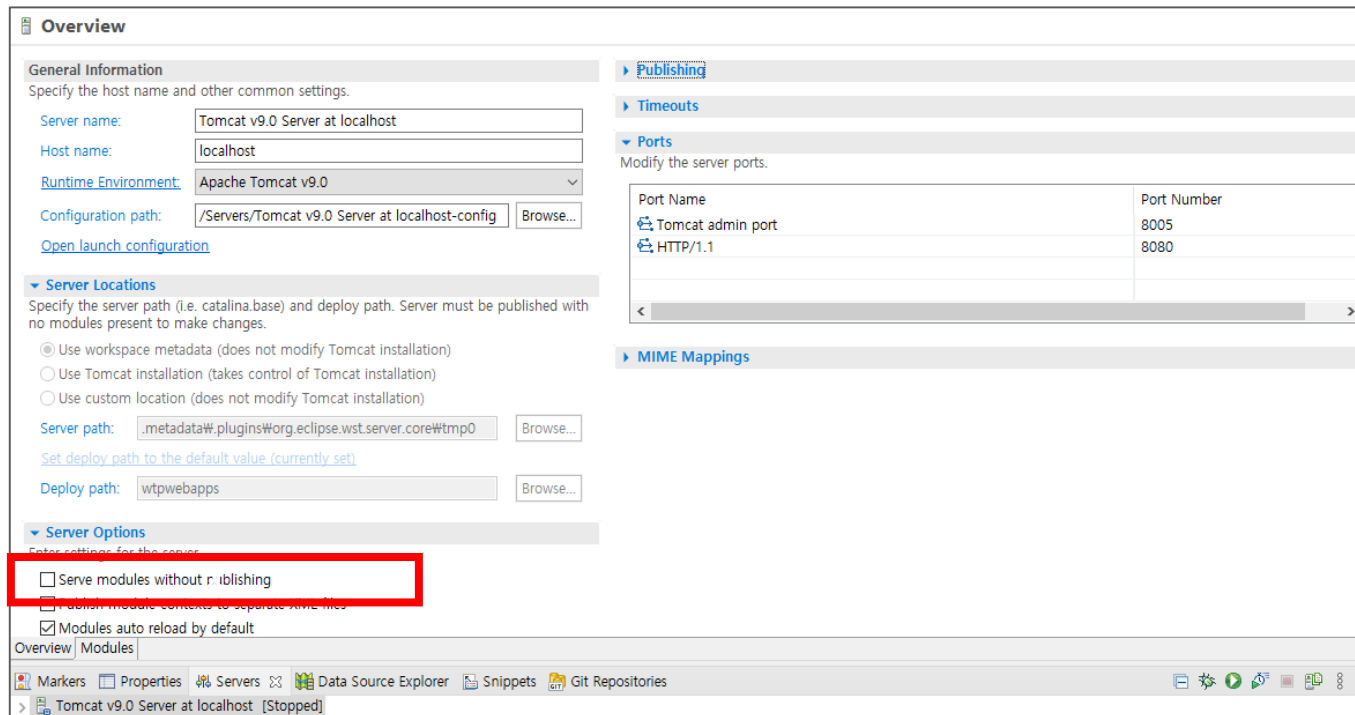
• 퍼블리싱 기능

- 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



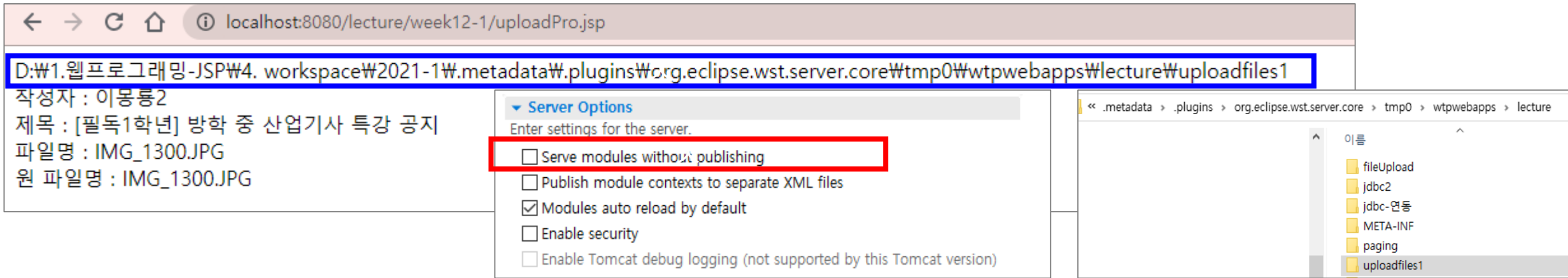
서버 설정

- 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)



[참고]

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



localhost:8080/lecture/week12-1/uploadPro.jsp

D:\1.웹프로그래밍-JSP\4. workspace\2021-1\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\lecture\uploadfiles1

작성자 : 이몽룡2
제목 : [필독1학년] 방학 중 산업기사 특강 공지
파일명 : IMG_1300.JPG
원 파일명 : IMG_1300.JPG

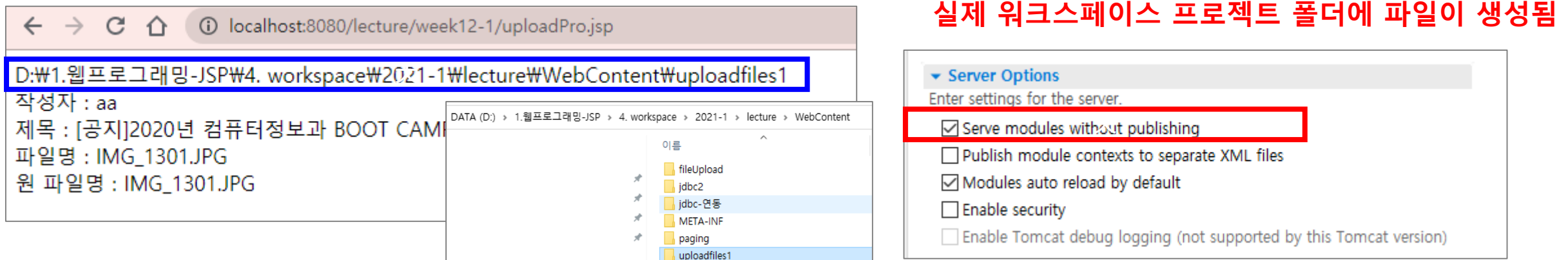
Server Options
Enter settings for the server.

- ☐ Serve modules without publishing
- ☐ Publish module contexts to separate XML files
- ☒ Modules auto reload by default
- ☐ Enable security
- ☐ Enable Tomcat debug logging (not supported by this Tomcat version)

이름
fileUpload
jdbc2
jdbc-연동
META-INF
paging
uploadfiles1

2. Server modules without publishing 체크(사용할 것)

- 메인 프로젝트는 의존하는 모듈(프로젝트 등)을 퍼블리싱 하지 않고 직접 참조



localhost:8080/lecture/week12-1/uploadPro.jsp

D:\1.웹프로그래밍-JSP\4. workspace\2021-1\lecture\WebContent\uploadfiles1

작성자 : aa
제목 : [공지]2020년 컴퓨터정보과 BOOT CAMP
파일명 : IMG_1301.JPG
원 파일명 : IMG_1301.JPG

DATA (D:) > 1.웹프로그래밍-JSP > 4. workspace > 2021-1 > lecture > WebContent

이름
fileUpload
jdbc2
jdbc-연동
META-INF
paging
uploadfiles1

Server Options
Enter settings for the server.

- ☒ Serve modules without publishing
- ☐ Publish module contexts to separate XML files
- ☒ Modules auto reload by default
- ☐ Enable security
- ☐ Enable Tomcat debug logging (not supported by this Tomcat version)

실제 워크스페이스 프로젝트 폴더에 파일이 생성됨

5. insertForm.jsp 폼 변경

[reference] :

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

- 폼의 **enctype** 주요 속성값

- application/x-www-form-urlencoded(기본값)
 - 요청 Http의 기본 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 이미지**를 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

- insertForm.jsp

```
<form action="insert.do" method="post" enctype="multipart/form-data">
  <div class="form-group">
    <label for="subject">subject :</label>
    <input type="text" class="form-control" id="subject" name="subject">
  </div>
  <div class="form-group">
    <label for="content">content :</label><br>
    <textarea rows="10" cols="80" name="content" id="content"></textarea>
  </div>
  <div class="form-group">
    <label for="writer">writer :</label>
    <input type="text" class="form-control" id="writer" name="writer">
  </div>
  <div class="form-group">
    <label for="filename">filename :</label>
    <input type="file" class="form-control" id="filename" name="filename">
  </div>
  <div class="text-center">
    <button type="submit" class="btn btn-secondary">등록</button>
  </div>
</form>
```

[참조]

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

request.getContentType()

```
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=----WebKitFormBoundaryg3lbmadDo87Bmh2R
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
```

```
-----WebKitFormBoundaryg3lbmadDo87Bmh2R
```

Part.getHeader("Content-Disposition");

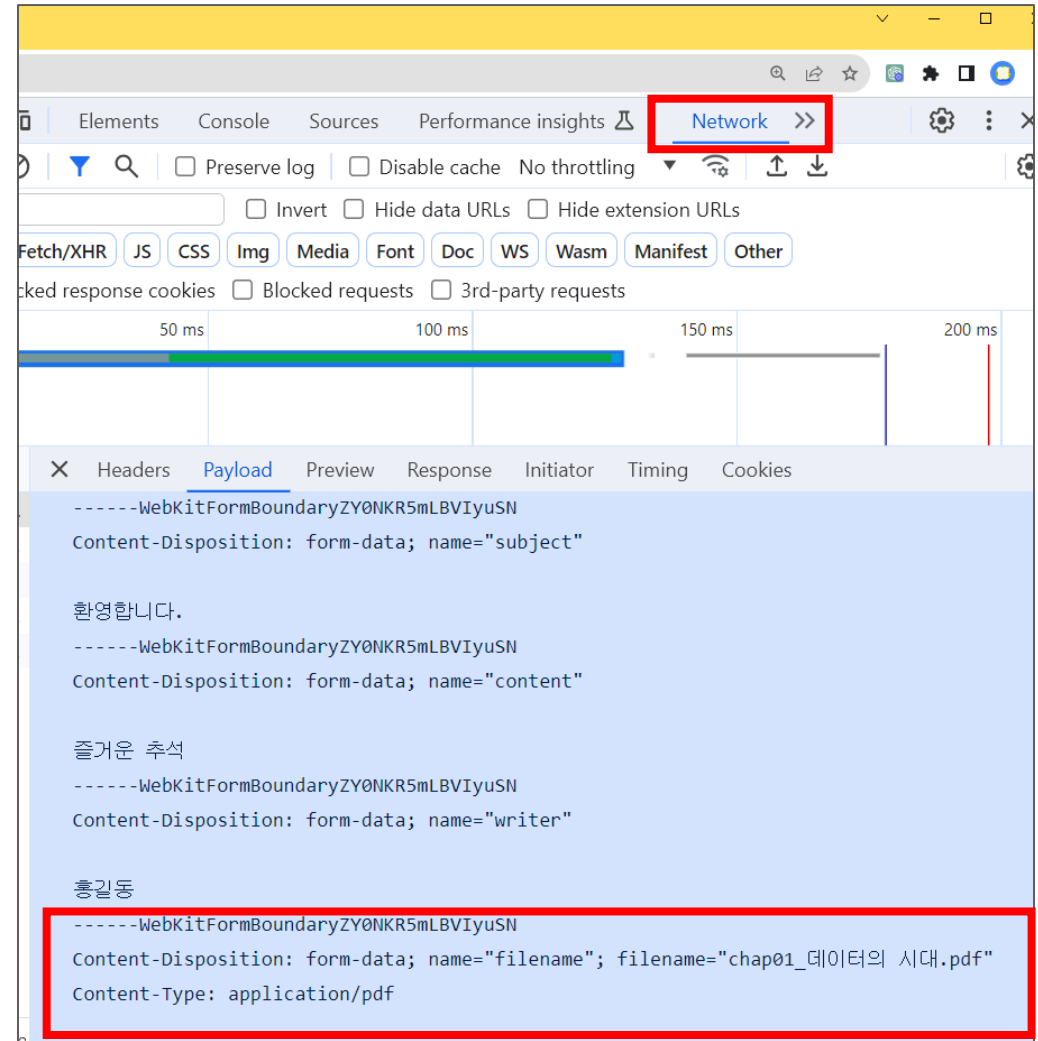
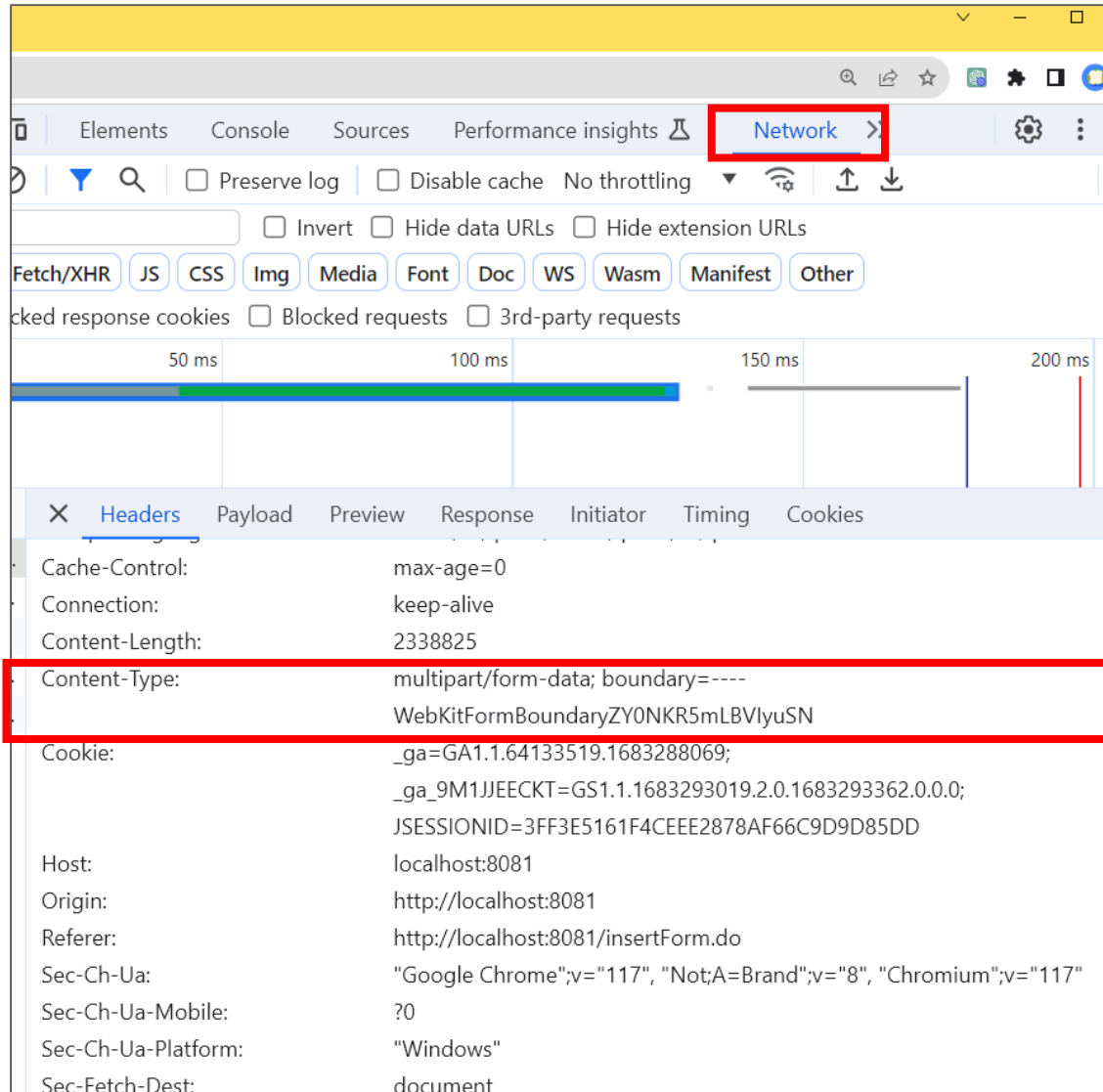
```
Content-Disposition: form-data; name="file1"; filename="메이븐.pptx"
Content-Type: application/vnd.openxmlformats-officedocument.presentationml.presentation
```

Part.getSubmittedFileName()

```
ppt 파일에 대한바이너리 데이터...
```

request body

[참조]크롬의 개발자도구 보기(F12)



6. 업로드 처리 서비스 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>

```
▼ Request Headers View parsed
POST /board-mvc/insert.do HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 158196
Content-Type: multipart/form-data; boundary=---WebKitFormBoundaryF7Qeu7EVrtm6uoAF
Cookie: JSESSIONID=43A783B2E3A02E0E32ABE142058457F2
Host: localhost:8080
Origin: http://localhost:8080
Referer: http://localhost:8080/board-mvc/insertForm.do
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
sec-ch-ua: "Chromium";v="106", "Google Chrome";v="106", "Not;A=Brand";v="99"
```

6. 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 헤더명 name에 대한 헤더값을 리턴 Ex: Part.getHeader("Content-Disposition");

6. 업로드 처리 서블릿 작성

- Part 인터페이스와 관련된 HttpServletRequest 객체의 메소드

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

6. 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
 - 두 가지 설정 방법
 1. @MultipartConfig 어노테이션을 이용한 방법
 2. web.xml 에 설정

1. @MultipartConfig 어노테이션

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

6. 업로드 처리 서블릿 작성

2. web.xml 설정

```
<servlet>
  <servlet-name>FileUploadServlet</servlet-name>
  <servlet-class>servlet.FileUploadServlet</servlet-class>
  <multipart-config>
    <location>C:\Wattaches</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

6. 업로드 처리 서블릿 작성

- **@MultipartConfig()** 어노테이션으로 설정

```
@MultipartConfig(  
    location = "C:\\\\attaches",  
    maxFileSize = -1,  
    maxRequestSize = -1,  
    fileSizeThreshold = 1024)  
@WebServlet(*.do")
```

- maxFileSize 설정을 통해 업로드 파일 크기 제한이 있는 경우 제한을 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  
}
```

6. 업로드 처리 서블릿 작성

- BoardController.java

```
BoardController.java X
1 package cs.dit.board;
2
3 import java.io.IOException;
4
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.MultipartConfig;
8 import javax.servlet.annotation.WebServlet;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 @MultipartConfig(
14     maxFileSize = 1024 * 1024 * 5,
15     maxRequestSize = 1024 * 1024 * 50
16 )
17 @WebServlet("*.do")
18 public class BoardController extends HttpServlet {
```

6. 업로드 처리 서블릿 작성

- BInsertService.java

```
BInsertService.java x
1 package cs.dit.board;
2
3 import java.io.File;
4
11
12 public class BInsertService implements BoardService {
13
14     @Override
15     public void execute(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         request.setCharacterEncoding("utf-8");
17
18         String subject = request.getParameter("subject");
19         String content = request.getParameter("content");
20         String writer = request.getParameter("writer");
21         String filename = null;
22         String dir = null;
23
24         //HTTP 요청객체인 HttpServletRequest 객체로부터 Content-Type의 헤더값이 multi-part/form-data 인지 확인
25         String contentType = request.getContentType();
26         if(contentType != null && contentType.toLowerCase().startsWith("multipart/")) {
27             dir = request.getServletContext().getRealPath("/uploadfiles");//실제경로
28         }
29
30         File f = new File(dir);//File(String pathname):지정된 경로 문자열을 추상 경로로 변환하여 새 File 객체 생성
31         if(!f.exists()) { //생성된 객체가 존재하지 않으면 폴더가 없는 것이므로
32             f.mkdir(); //해당 경로에 폴더 생성
33         }
34     }
35 }
```

6. 업로드 처리 서블릿 작성

- BlnsertService.java

```
30     File f = new File(dir); //File(String pathname):지정된 경로 문자열을 추상 경로로 변환하여 새 File 객체 생성
31     if(!f.exists()) { //생성된 객체가 존재하지 않으면 폴더가 없는 것이므로
32         f.mkdir(); //해당 경로에 폴더 생성
33     }
34
35     //request.getParts() 메서드를 통해 여러 개의 Part를 Collection 에 담아 리턴
36     Collection<Part> parts = request.getParts();
37
38     for(Part p :parts) {
39         //part의 Content-Disposition 헤더가 filename=을 포함하면 파일로 구분
40         if(p.getHeader("Content-Disposition").contains("filename=")) {
41             if(p.getSize()>0) {
42                 filename = p.getSubmittedFileName(); //업로드한 파일명 리턴
43                 String filePath = dir + File.separator + filename; //File.separator : \
44
45                 p.write(filePath); //디스크에 파일 쓰기
46
47                 p.delete(); //임시저장된 파일 데이터를 수동으로 제거, 일반적으로 http 요청이 처리되고 응답을 출력하는 시점에 자동으로 제거됨
48             }
49         }
50     }
51
52     BoardDto dto = new BoardDto(0, subject, content, writer, null, filename);
53     BoardDao dao = new BoardDao();
54
55     dao.insert(dto);
56
57 }
58
59 }
```

7. 데이터베이스 처리

- BoardDao.java

```
public void insert(BoardDto dto) {  
    String sql = "INSERT INTO board1(SUBJECT, CONTENT, WRITER, REGDATE, FILENAME) "  
        + "VALUES(?, ?, ?, SYSDATE(), ?)";  
  
    try (  
        Connection con = getConnection();  
        PreparedStatement pstmt = con.prepareStatement(sql);  
    )  
    {  
        pstmt.setString(1, dto.getSubject());  
        pstmt.setString(2, dto.getContent());  
        pstmt.setString(3, dto.getWriter());  
        pstmt.setString(4, dto.getFilename());  
  
        pstmt.executeUpdate();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

8. 파일 다운로드(updateForm.jsp)

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

```
<form action="update.do" method="post" enctype="multipart/form-data">
  <input type="hidden" name="bcode" value="${dto.bcode}">
  <table class="table table-striped table-hover">
    <tr>
      <th>bcode</th><td>${dto.bcode}</td>
      <th>subject</th><td><input type="text" value="${dto.subject}" name="subject"></td>
    </tr>
    <tr>
      <th>content</th>
      <td colspan="3"><textarea rows="10" cols="80" name="content">${dto.content}</textarea></td>
    </tr>
    <tr>
      <th>writer</th><td><input type="text" value="${dto.writer}" name="writer"></td>
      <th>regDate</th><td><input type="text" value="${dto.regDate}" name="regDate"></td>
    </tr>
    <tr>
      <th>기존파일</th><td><a download href="/uploadfiles/${dto.filename }">${dto.filename }</a></td>
      <th>변경파일</th><td><input type="file" class="form-control" id="filename" name="filename"></td>
    </tr>
    <tr>
      <td colspan="4">
```


9. 기타 변경 필요 파일 – 잘 찾아서 완성시킬 것

- BoardDao.java – selectOne() 등 다양한 곳에서 변경사항 발생

```
public BoardDto selectOne(int bcode) {  
  
    String sql = "SELECT SUBJECT, CONTENT, WRITER, REGDATE, FILENAME FROM board1 WHERE bcode =?";  
    BoardDto dto = new BoardDto();  
  
    try (    Connection con = getConnection();  
           PreparedStatement pstmt = con.prepareStatement(sql);  
        )  
    {    pstmt.setInt(1, bcode);  
      try(ResultSet rs = pstmt.executeQuery();)  
      {  
          rs.next();  
          dto.setBcode(bcode);  
          dto.setSubject(rs.getString("subject"));  
          dto.setContent(rs.getString("content"));  
          dto.setWriter(rs.getString("writer"));  
          dto.setRegDate(rs.getDate("regDate"));  
          dto.setFilename(rs.getString("filename"));  
  
      }catch (Exception e) {  
          e.printStackTrace();  
      }  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    }  
    return dto;  
}
```


<과제>

추석 끝나고 전체 파일이 잘 돌아갈 수 있도록 할 것



1. github에 업로드
2. 수업시간에 실행 내용 확인

[문제 발생 시]

- 파일 다운로드 시 자꾸 .htm 문서로 될 때 확인해야 할 것
 - **실행 경로 확인할 것!!!**
 - 서버를 더블클릭하여 [Modules] 탭을 선택
 - Path, Document-Base, Module 의 값 확인 할 것

 **Web Modules**

Web Modules
Configure the Web Modules on this server.

Path	Document Base	Module	Auto Reload
 /board-fileupload	board-fileupload	 board-fileupload	Enabled

Add Web Module...

Add External Web Module...

Edit...

Remove