

KB금융그룹



2025년 상반기 K-디지털 트레이닝

File Upload

저장하고 DB등록

[KB] IT's Your Life

파일을 저장할때 2가지 방식

파일시스템에 저장해놓고 파일에 대한 경로를 디비로 관리

blob타입으로 테이블에 직접 넣어놓는 방법.

우리는 1번째 방법으로

stream 작업이 필요함. binary데이터니

생성하고 업데이트에서 업로드가 일어나고 상세보기에서 다운로드가 일어날것

 KB 국민은행

✓ File upload

- 첨부파일을 위한 tbl_board_attachment 테이블 생성
- board의 bno에 대한 FK 설정

```
DROP TABLE IF EXISTS tbl_board_attachment;
```

```
CREATE TABLE tbl_board_attachment (  
  no INTEGER AUTO_INCREMENT PRIMARY KEY, ✓  
  filename VARCHAR(256) NOT NULL, -- 원본 파일 명  
  path VARCHAR(256) NOT NULL, -- 서버에서의 파일 경로  
  content_type VARCHAR(56), -- content-type  
  size INTEGER, -- 파일의 크기  
  bno INTEGER NOT NULL, ✓  
  reg_date DATETIME DEFAULT now(),  
  CONSTRAINT FOREIGN KEY(bno) REFERENCES tbl_board(no)  
);
```

다운로드시 저장명을 위해 원본명을 저장할 필드

-- 서버에서의 파일 경로

-- content-type

-- 파일의 크기

-- 게시글 번호, FK

서버에 업로드할때 이름이 같으면 덮어씌워진다.

서버에서 저장할때 사용할 유일한 이름이 되도록해야한다

✓ 주의 사항

- 업로드 된 파일을 저장할 때 파일 명 중복 발생 가능

→ 유일한 파일명이 되도록 해야함. U

- 방법

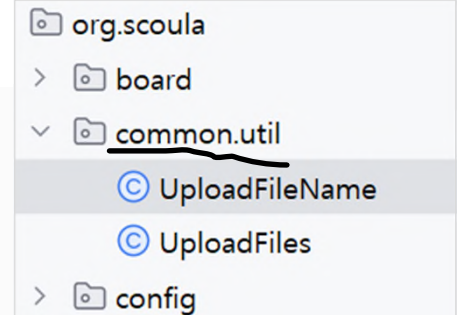
- UUID를 사용 ✓
- 기존 파일명에 timestamp를 추가 ✓ 000 저장될 당시에 시간을 추가

- 다운로드를 대비해서 원본 파일명을 저장 해야함

UploadFileName.java

```
package org.scoula.common.util;  
  
public class UploadFileName {  
    public static String getUniqueName(String filename) {  
        // 기존 파일명에서 유일한 이름을 뽑게끔  
        // 서버에 저장하기 위해서  
  
        int ix = filename.lastIndexOf(".");  
        String name = filename.substring(0, ix);  
        String ext = filename.substring(ix+1);  
        // 파일명 추출  
        // 확장명 추출  
  
        return String.format("%s-%d.%s", name, System.currentTimeMillis(), ext);  
    }  
    // 타임스탬프  
}
```

a.txt ==> a-2025???.txt



UploadFiles.java

```
package org.scoula.common.util;
```

```
import org.springframework.web.multipart.MultipartFile;
```

```
import java.io.File;
```

```
import java.io.IOException;
```

```
public class UploadFiles {
```

```
    public static String upload(String baseDir, MultipartFile part) throws IOException {
```

```
        // 기본 디렉토리가 있는지 확인, 없으면 새로 생성
```

```
        File base = new File(baseDir);
```

```
        if(!base.exists()) {
```

```
            base.mkdirs();
```

```
        }
```

// 중간에 존재하지 않는 디렉토리까지 모두 생성

```
        String fileName = part.getOriginalFilename();
```

```
        File dest = new File(baseDir, UploadFileName.getUniqueName(fileName));
```

```
        part.transferTo(dest);
```

```
        return dest.getPath();
```

```
    }
```

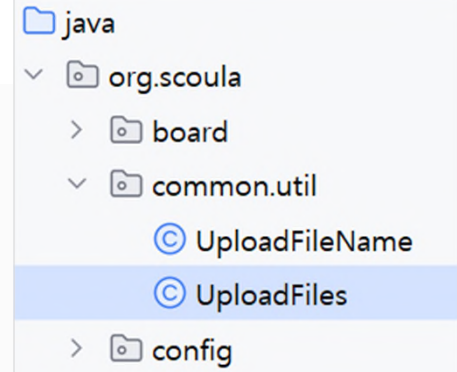
```
}
```

✓ 클라이언트 단에서 사용한 파일이름

// 지정된 경로로 업로드 파일 이동 파일명은 서버 유일이름

// 저장된 파일 경로 리턴

dest로 옮겨라. 우리가 정한 기준 크기에 따라 옮길 데이터가 톰캐의 임시파일로 존재하거나 메모리에 존재한다. 존재하는 것을 FS로 옮겨라. 라는 명령 즉 실질적인 업로드는 이미 끝나 있음. 영속성을 위해서 서버에 제대로 저장(옮기는)하는 일.



BoardAttachmentVO.java 첨부파일 표현

```
package org.scoula.board.domain;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.web.multipart.MultipartFile;
```

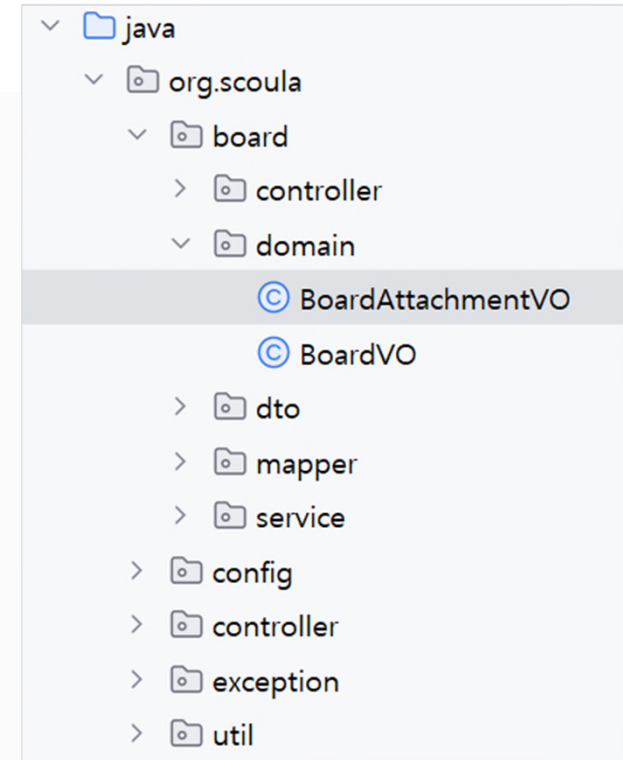
```
import java.util.Date;
```

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
```

```
public class BoardAttachmentVO {
    private Long no; // pk
    private Long bno; // FK: Board의 no
    private String filename; // 원본 파일명
    private String path; // 서버에 저장된 파일 경로
    private String contentType; // 파일 mime-type
    private Long size; // 파일의 크기
    private Date regDate; // 등록일
```

테이블에 준하는 VO

mime type == content type ex)image/jpeg



BoardAttachmentVO.java

// 팩토리 메서드

public static BoardAttachmentVO of(MultipartFile part, Long bno, String path) { // path: 업로드된 파일 경로

FK

실제 경로

첨부파일 관한 VO
생성기

return builder()

.bno(bno)

.filename(part.getOriginalFilename()) ✓

.path(path)

.contentType(part.getContentType()) ✓

.size(part.getSize()) ✓

.build();

}

}

빌더를 이용한 팩토리 패턴

Board 하고 BoardAttachment하고 1:N 관계죠?

board를 통해서 boardAttachment가 핸들링 됨.

boardAttachment매핑을 따로 만들어도 되지만
이번엔 board매핑에 통합하여 만들자

File upload

BoardMapper.java

```
package org.scoula.board.mapper;  
...
```

```
public interface BoardMapper {  
    ...
```

인터페이스 추가

```
    public void createAttachment(BoardAttachmentVO attach);
```

```
    public List<BoardAttachmentVO> getAttachmentList(Long bno);
```

```
    public BoardAttachmentVO getAttachment(Long no);
```

```
    public int deleteAttachment(Long no);
```

```
}
```

binary 데이터이므로 update는 없다

resources::BoardMapper.xml

```
<insert id="createAttachment">
    insert into tbl_board_attachment(filename, path, content_type, size, bno)
    values("#{filename}", #{path}, #{contentType}, #{size}, #{bno})
</insert>

<select id="getAttachmentList" resultType="org.scoula.board.domain.BoardAttachmentVO">
    select * from tbl_board_attachment
    where bno = #{bno} PK가 아닌 FK로 where절 구성
    order by filename
</select>

<select id="getAttachment" resultType="org.scoula.board.domain.BoardAttachmentVO">
    select * from tbl_board_attachment
    where no = #{no}
</select>

<delete id="deleteAttachment">
    delete from tbl_board_attachment
    where no = #{no}
</delete>
```

추가한 인터페이스 메서드에 대한
sql 매핑 xml 설정

게시글
상세보기

where bno = #{bno} PK가 아닌 FK로 where절 구성

✓ BoardVO에 BoardAttachmentVO 연결

○ BoardVO.java

```
@Data
public class BoardVO {
...

```

private List<BoardAttachmentVO> attaches; 1대N 관계니까 이렇게 표현하는거 기억하죠?

```
private Date regDate;
private Date updateDate;
}
```

- 1) join 처리 → resultMap 구성
- 2) setAttaches() 호출로 처리

게시판안에는

한개이상의 첨부파일이 들어가니까

해당 필드를 어떻게 채울까?

1. join mybatis는 join처리하기 쉬움
2. 매퍼의 메서드 호출하여 얻기

일반적으로 2번째는 jpa에서 취하는 방법

우리는 1로 해보자.

resultMap구성방법 배워야함.

BoardDTO.java

```
...  
public class BoardDTO {  
    private Long no;  
    private String title;  
    private String content;  
    private String writer;  
    private Date regDate;  
    private Date updateDate;
```

DTO에도
BoardAttachmentVO에 대한 리스트.
실제 업로드된 파일들 리스트.

// 첨부 파일

private List<BoardAttachmentVO> attaches;

애는 매퍼를 통해 쿼리의 결과가 들어감.

List<MultipartFile> files = new ArrayList<>(); // 실제 업로드된 파일(Multipart) 목록

폼 인풋들을 처리하기 위한 녀석

<input type="file" name="files">

애는 이제 폼 입력될때
매핑된 컨트롤러의 메소드를 통해서
DTO의 해당 멤버에
바이너리데이터가 들어감.

BoardDTO.java

```
// VO --> DTO 변환
public static BoardDTO of(BoardVO vo) {
    return BoardDTO.builder()
        ...
        .attaches(vo.getAttaches())
        .regDate(vo.getRegDate())
        .updateDate(vo.getUpdateDate())
        .build();
}

// DTO --> VO 변환
public BoardVO toVo() {
    return BoardVO.builder()
        ...
        .attaches(attaches)
        .regDate(regDate)
        .updateDate(updateDate)
        .build();
}
}
```

변환과정에 attaches 변수도 변환하는 거 추가

mybatis가 join을 어떻게 처리하는지

지금까지는 쿼리의 결과를 그대로 VO에 매핑했다.
칼럼명 -> 필드명 값으로.

File upload

resultmap 방식은 어떤 칼럼 정보가 어떤 필드정보에 들어가는지 일일이 다 지정해야 함.
그게 resultMap이다. * KB 국민은행

org/scoula/board/mapper/BoardMapper.xml

join

```
<resultMap id="attachmentMap" type="org.scoula.board.domain.BoardAttachmentVO">
  <id column="ano" property="no"/>
  <result column="bno" property="bno"/>
  <result column="filename" property="filename"/>
  <result column="path" property="path"/>
  <result column="contentType" property="contentType"/>
  <result column="size" property="size"/>
  <result column="a_reg_date" property="regDate"/>
</resultMap>
```

```
<resultMap id="boardMap" type="org.scoula.board.domain.BoardVO">
  <id column="no" property="no"/>
  <result column="title" property="title"/>
  <result column="content" property="content"/>
  <result column="writer" property="writer"/>
  <result column="reg_date" property="regDate"/>
  <result column="update_date" property="updateDate"/>
  <collection property="attaches" resultMap="attachmentMap"/>
</resultMap>
```

boardvo의 추가한 필드명

1:N처리
1:1처리는
association
태그 사용

일일이 어느것에 매핑할지 지정
그래서
테이블 컬럼명과 객체 프로퍼티명이랑
다를 때도 많이 사용함

resulttype 말고

```
<select id="get" resultMap="boardMap">
  select b.*, a.no as ano, a.bno, a.filename, a.path,
    a.content_type, a.size, a.reg_date as a_reg_date
  from tbl_board b
  left outer join tbl_board_attachment a
    on b.no = a.bno
  where b.no = #{no}
  order by filename
</select>
```

join된 형태를 담을때 각 정보를 매핑해주는
것을 지정하는 형태태의 resultMap

BoardService.java

```
public interface BoardService {
```

```
    ...
```

```
    public BoardAttachmentVO getAttachment(Long no);
```

```
    public boolean deleteAttachment(Long no);
```

```
}
```

) 추가 삭제 기능

✓ Transaction 처리

○ RootConfig 설정에 @EnableTransactionManagement 추가 ✓

- 트랜잭션 처리 활성화

...

@EnableTransactionManagement ✓

public class RootConfig {

...

○ @Transactional

- 클래스 레벨
 - 클래스의 모든 메서드에 트랜잭션 처리
- 메서드 레벨
 - 지정한 메서드에만 트랜잭션 처리
- 동작
 - 에러가 없으면 commit ✓
 - 예외(RuntimeException) 발생 시 rollback Y

트랜잭션 안에서는 예외발생시
런타임 예외로 바꿔줘야 한다.
@Transactional이 인지하게끔

그리고 mybatis는
sql예외를 runtime예외로
자동으로 바꿔준다
하지만 지금처럼 sql예외가 아닐경우
runtime으로 바꿔줘야 함

BoardServiceImpl.java

```
public BoardServiceImpl implements BoardService {  
    private final static String BASE_DIR = "c:/upload/board";  
    ...  
}
```

// 2개 이상의 insert 문이 실행될 수 있으므로 트랜잭션 처리 필요
// RuntimeException인 경우만 자동 rollback.
@Transactional 스프링의 기가막힌 기능

@Override

```
public void create(BoardDTO board) {  
    log.info("create....." + board);  
  
    BoardVO boardVO = board.toVo();  
    mapper.create(boardVO);  
}
```

// 파일 업로드 처리

```
List<MultipartFile> files = board.GetFiles();  
if(files != null && !files.isEmpty()) {  
    upload(boardVO.getNo(), files);  
}
```

// 첨부 파일이 있는 경우

insert에 대한 실행이
board에 대한 insert와
boardAttachment에 대한 insert가
동시에 일어날 수 있다. 하나가 실패하면
전체가 복원되어야한다. 롤백.
트랜잭션 처리해야한다.

File upload

BoardServiceImpl.java

```
private void upload(Long bno, List<MultipartFile> files) {
    for(MultipartFile part: files) {
        if(part.isEmpty()) continue;
        try {
            String uploadPath = UploadFiles.upload(BASE_DIR, part);
            BoardAttachmentVO attach = BoardAttachmentVO.of(part, bno, uploadPath);
            mapper.createAttachment(attach);
        } catch (IOException e) {
            throw new RuntimeException(e); // @Transactional에서 감지, 자동 rollback
        }
    }
}
```

예외 던질 수 있음

전에 만든 유틸리티 사용

예외 발생시
IO예외를 런타임 예외로 바꿈
왜?

호출하는 쪽에서 따로
예외처리할 필요없어짐

그리고 Transactional 어노테이션은
런타임예외 대해서만 동작.

transactional 에 대해서 자동으로 롤백

BoardServiceImpl.java

```
...  
  
// 첨부파일 한 개 얻기  
@Override  
    public BoardAttachmentVO getAttachment(Long no) {  
        return mapper.getAttachment(no);  
    }  
  
// 첨부파일 삭제  
@Override  
    public boolean deleteAttachment(Long no) {  
        return mapper.deleteAttachment(no) == 1;  
    }  
}
```

✓ 파일 업로드

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

...

<input type="file" multiple />

...

인코딩 타입드러가야하고

인풋타입이 파일

form 태그에서 action속성을 지정하지 않으면
폼 데이터는 현재 페이지의 url을 기준으로 전송됨

File upload

views/board/create.jsp

중요!!!!

```
...
<div>
  <form method="post" enctype="multipart/form-data">
    <div>
      <label>제목</label>
      <input name="title" class="form-control">
    </div>

    <div>
      <label>작성자</label>
      <input name="writer" class="form-control">
    </div>

    <div>
      <label>첨부파일</label>
      <input type="file" class="form-control-file border" multiple name="files"/>
    </div>

    <div>
      <label>내용</label>
      <textarea class="form-control" name="content" rows="10"></textarea>
    </div>
  </form>
</div>
...
```

다중선택 가능하게

여는 이제 폼 입력될때
매핑된 컨트롤러의 메소드를 통해서
DTO의 해당 멤버에
바이너리데이터가 들어감.

dto

아까 DTO에

List<MultipartFile> files 필드에 자동으로
매핑됨!

```
public class BoardDTO {
    private Long no;
    private String title;
    private String content;
    private String writer;
    private Date regDate;
    private Date updateDate;
```

DTO에도
BoardAttachmentVO에 대한 리스트.
실제 업로드된 파일들 리스트.

```
// 첨부 파일
private List<BoardAttachmentVO> attaches;
```

```
List<MultipartFile> files = new ArrayList<>(); // 실제 업로드된 파일(Multipart) 목록
// 폼 인풋들을 처리하기 위한 녀석
```

```
<input type="file" name="files">
```


2025년 상반기 K-디지털 트레이닝

File Download


[KB] IT's Your Life


- ✓ 상세 보기에서 첨부파일 목록 보여주기
 - 다운로드 링크 전시

📄 파일 업로드 테스트

 admin

🕒 2023-12-07

 백엔드.mm (631 Bytes)

 컨트롤러에서 직접 인증하기.pdf (1.2 MB)

파일 업로드 테스트

☰ 목록

✎ 수정

🗑 삭제

다운로드시
경우의 수 2가지

저장 대화 상자 표시
(어디다 저장할지)

바로보기로 넘어갈때

헤더 ○에서
저장용으로
지정하면
대화상자가 뜨지만
아니면
바로보기

✓ 첨부파일 목록 보여주기

```
@Data
public class BoardDTO {
    ...

    private List<BoardAttachmentVO> attaches;
    ...
}
```

📄 파일 업로드 테스트

👤 admin

🕒 2023-12-07

📄 백엔드.mm (631 Bytes)

📄 컨트롤러에서 직접 인증하기.pdf (1.2 MB)

파일 업로드 테스트

☰ 목록

✎ 수정

🗑 삭제

✓ Java로 파일 크기 포맷 출력 메서드


○ 1225957 → 1.225MB

```
public class UploadFiles {  
    ...  
  
    public static String getFormatSize(Long size) {  
        if (size <= 0)  
            return "0";  
        final String[] units = new String[] { "Bytes", "KB", "MB", "GB", "TB" };  
        int digitGroups = (int) (Math.log10(size) / Math.log10(1024));  
        return new DecimalFormat("#,##0.#").format(size / Math.pow(1024, digitGroups)) + " " + units[digitGroups];  
    }  
}
```

나중에 뷰 적용할 때는 js가 해줘야하는 일

BoardAttachmentVO.java

```
public class BoardAttachmentVO {  
    private Long no;        // PK  
    private Long bno;       // FK: Board.no  
    private String filename; // 원본 파일명  
    private String path;    // 서버에 저장된 파일 경로  
    private String contentType; // 파일 mime-type  
    private Long size;      // 파일의 크기  
    private Date regDate;   // 등록일  
  
    public static BoardAttachmentVO of(MultipartFile part, Long bno, String path) {  
        return builder()  
            .bno(bno)  
            .filename(part.getOriginalFilename())  
            .path(path)  
            .contentType(part.getContentType())  
            .size(part.getSize())  
            .build();  
    }  
  
    public String getFileSize() {  
        return UploadFiles.getFormatSize(size);  
    }  
}
```



getter fileSize프로퍼티.

그러면 jsp에서 `${ attatch.fileSize }`로 표현할 수 있따.

꼭 필드 멤버가 아니어도
저런 프로퍼티 게터 형식만 있으면
jsp에서 변수 표현이 가능하다

views/board/get.jsp

```
...
<h1 class="page-header my-4"><i class="far fa-file-alt"></i> ${board.title}</h1>

<div class="d-flex justify-content-between">
  <div><i class="fas fa-user"></i> ${board.writer}</div>
  <div>
    <i class="fas fa-clock"></i>
    <fmt:formatDate pattern="yyyy-MM-dd" value="${board.regDate}"/>
  </div>
</div>

<div class="text-end">
  <c:forEach var="file" items="${board.attaches}">
    <div class="attach-file-item">
      <a href="/board/download/${file.no}" class="file-link">
        <i class="fa-solid fa-floppy-disk"></i>
        ${file.filename} (${file.fileSize})<br>
      </a>
    </div>
  </c:forEach>
</div>

<hr>
...
```

@GetMapping("download/변수처리")
동적 패스 변수.

파일 다운로드 링크

아까 만든 getFileSize 메서드 동작됨

✓ File download

- c:/upload/board 경로는 웹 url로 표현할 수 없음
 - 직접 링크를 걸 수 없음
- Controller에서 download 요청을 받았을 때 직접 파일을 읽어서 처리
 - url: /board/download/{no}
 - 경로변수: url상에 있는 변수 ✓
 - @PathVariable("no")로 추출 ✓ — di 요청하여 적용된 변수에 경로변수 값 넣어줌
- 다운로드
 - 응답 헤더로 파일 첨부 정보를 설정 아까 말한 헤더의 설정에 따라 다운로드 2가지 경우의 수 중 하나 택 가능
 - HttpResponse에 binary stream을 얻어 직접 write

outputstream => BufferedOutputStream
빠른 작업을 위해서

File download

UploadFiles.java

다운로드 기능은 범용 기능이다.
따로 유틸리티로 구성해서 활용하자

```
public class UploadFiles {
```

```
...
```

컨트롤러에서 받으면 돼 서버에 있는 파일 객체

```
public static void download(HttpServletResponse response, File file, String orgName) throws Exception {
```

원본파일명

헤더
수정

```
response.setContentType("application/download");
```

```
response.setContentLength((int)file.length());
```

```
String filename = URLEncoder.encode(orgName, "UTF-8"); // 한글 파일명인 경우 인코딩 필수
```

```
response.setHeader("Content-disposition", "attachment;filename=\"" + filename + "\"");
```

```
try(OutputStream os = response.getOutputStream();
```

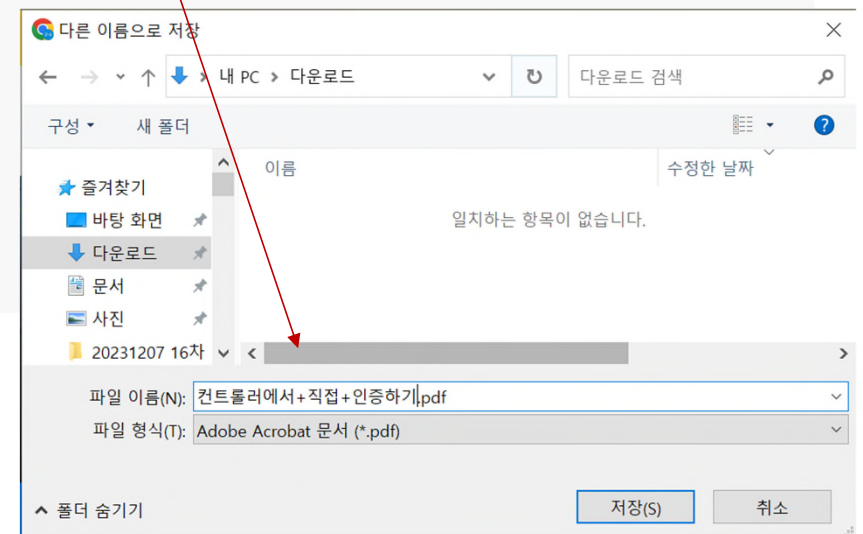
```
    BufferedOutputStream bos = new BufferedOutputStream(os)) {
```

```
    Files.copy(Paths.get(file.getPath()), bos);
```

Path객체

텍스트가 아닌 바이너리 데이터이므로
writer가 아닌 stream을 갖해서
출력

```
}  
}  
}
```



File download

BoardController.java

```

@GetMapping("/download/{no}")
@ResponseBody // view를 사용하지 않고, 직접 내보냄
public void download(@PathVariable("no") Long no, HttpServletResponse response) throws Exception {
    BoardAttachmentVO attach = service.getAttachment(no);

    File file = new File(attach.getPath()); 서버 경로

    UploadFiles.download(response, file, attach.getFilename());
}

```

전 페이지에서 만든 static method
파일 데이터 정보와 원본이름 가지고
response객체를 통해 헤더 구성을하고
outputstream을 얻어 바이너리 데이터 옮기고

ControllerAdvice에서
처리

여기서 void는 스프링의 뷰리솔버를 거치지 않고
컨트롤러에서 직접 응답을 처리하겠다는 뜻.

ResponseBody는
뷰를 사용하지 않고 테스트/파일 응답 등을 위한 설정.

여기서 @ResponseBody는
반환형이 없기에 JSON으로 변환하진 않고
뷰를 사용하지 않는다고 간주하게끔만 명시하는 역할만함