**파일 처리**

|  |  |  |
| --- | --- | --- |
| 1 | 요구 분석 | 기능, 파일 수, 보안, UI 흐름 |

|  |  |  |
| --- | --- | --- |
| 2 | 저장 전략 | 파일시스템 저장 채택 |

|  |  |  |
| --- | --- | --- |
| 3 | DB 설계 | 첨부파일 테이블, FK 구성 |

|  |  |  |
| --- | --- | --- |
| 4 | 도메인 설계 | BoardVO, BoardAttachmentVO, DTO |

|  |  |  |
| --- | --- | --- |
| 5 | 파일명 처리 | 유일성 확보: 타임스탬프 기반 |

|  |  |  |
| --- | --- | --- |
| 6 | 유틸 구현 | upload, download, size format |

|  |  |  |
| --- | --- | --- |
| 7 | MyBatis 설정 | resultMap, join, collection |

|  |  |  |
| --- | --- | --- |
| 8 | Service 구성 | 트랜잭션 포함 insert + upload |

|  |  |  |
| --- | --- | --- |
| 9 | JSP 연동 | form enctype, file input |

|  |  |  |
| --- | --- | --- |
| 10 | 다운로드 처리 | Controller + 응답 스트림 |

**✅ 1단계: 요구사항 분석 및 저장 전략 수립**

**📌 핵심 사용자 요구**

* 사용자는 게시글 작성 시 파일을 첨부할 수 있어야 함
* 업로드 가능한 파일은 **1개 이상 (다중 업로드)**
* 업로드된 파일은 서버에 안전하게 저장되어야 하며
* 게시글 상세 화면에서 **파일명과 용량이 표시**되고
* 다운로드 클릭 시 **브라우저가 파일 다운로드 수행** (저장/미리보기 가능)

**📌 고려할 기술적 요소**

* 파일 유효성 검사 (확장자 제한, 크기 제한)
* 업로드 시 중복 파일명 충돌 방지
* 보안: 파일 이름 노출 여부, 경로 탈취 방지, 디렉토리 traversal 공격 차단
* 저장 위치 관리 (예: 날짜별 디렉토리 구조로 저장 가능)

**📌 저장 전략 비교**

| **항목** | **파일시스템 저장** | **BLOB 방식 (DB 저장)** |
| --- | --- | --- |
| 속도 | 빠름 | 느림 |
| 확장성 | 유리 (외부 스토리지 연동 가능) | 낮음 (DB 크기 폭증) |
| 백업 | 개별 파일/폴더 백업 가능 | DB 백업 일원화 |
| 실무 채택률 | ★★★★★ (대부분) | ★☆☆☆☆ (거의 안 씀) |

* **📌 결론**: 서버 디스크에 저장 + DB에 메타데이터 저장 (경로, 파일명, 크기 등)
*  업로드는 "생성/수정" 시, 다운로드는 "상세보기" 시 발생.
*  Binary 데이터를 다루기 때문에 **Stream 처리** 필요. Inputstream/outputstream

실행 타이밍 : 업로드 (글작성(board create요청), 글수정(board update요청)) 다운로드(상세보기 요청시 서버에 저장된 바이너리데이터가 응답으로 오게끔)

**✅ 2단계: DB 테이블 설계 및 도메인 클래스 정의**

**📁 첨부파일 테이블 구조 설계**

**테이블명**: tbl\_board\_attachment  
**연관 테이블**: tbl\_board (게시글)

✔ 첨부파일 테이블: tbl\_board\_attachment 📎 bno는 게시글의 PK를 참조 → 1:N 관계 형성

DROP TABLE IF EXISTS tbl\_board\_attachment;

CREATE TABLE tbl\_board\_attachment (

no INT AUTO\_INCREMENT PRIMARY KEY,

filename VARCHAR(256), -- 사용자 업로드한 원본 파일명

path VARCHAR(256), -- 서버에 저장된 실제 경로

content\_type VARCHAR(56), -- MIME 타입 (image/jpeg, application/pdf 등)

size INT, -- 파일 크기 (byte)

bno INT NOT NULL, -- 게시글 번호 (FK)

reg\_date DATETIME DEFAULT NOW(),

FOREIGN KEY (bno) REFERENCES tbl\_board(no)

);

파일명 중복 방지를 위해 서버에 저장될 때는 유일한 이름 저장 필요.

✔ VO 객체: main>java>org.scoula>ex>domain>BoardAttachmentVO

@Data @Builder @NoArgsConstructor @AllArgsConstructor

public class BoardAttachmentVO { //DB테이블에 대응하는 VO

private Long no; //pk

private Long bno; //FK Board의 no

private String filename; // 원본명

private String path; // 서버에 저장된 파일 경로

private String contentType; // MIME. 파일 타입 ex)image/jpeg

private Long size; // byte

private Date regDate;

//multipartfile 🡺 VO객체 생성

public static BoardAttachmentVO of(MultipartFile file, Long bno, String path) { //팩토리 메서드

return BoardAttachmentVO.builder() //첨부파일에 관한 VO 생성기

.bno(bno)

.filename(file.getOriginalFilename())

.path(path)

.contentType(file.getContentType())

.size(file.getSize())

.build();

}

public String getFileSize() {

return UploadFiles.getFormatSize(size);

}

}

📌 getFileSize()는 뷰에서 ${file.fileSize}로 접근 가능  
📌 of() 메서드는 VO 생성 편의 제공 (팩토리 패턴)

**✅ 3단계: 파일명 충돌 방지 설계**

**📌 문제**

* 서로 다른 사용자가 같은 이름의 파일(a.txt) 업로드 가능
* 서버는 파일명을 기준으로 저장 → 덮어씌움

**📌 해결 전략**

Main>java>org.scoula>common.util>UploadFileName.java//파일 업로드시 이름 지정 관련 유틸 클래스

public static String getUniqueName(String filename) {

int ix = filename.lastIndexOf(".");

String name = filename.substring(0, ix); //파일명 추출

String ext = filename.substring(ix + 1); //확장명 추출

return String.format("%s-%d.%s", name, System.currentTimeMillis(), ext);/타임스탬프

}

**예시**

* resume.pdf → resume-1720271001123.pdf

**✅ 실무 팁**

* UUID도 가능하지만, 타임스탬프가 디버깅 시 더 직관적임
* 필요 시 사용자의 ID 또는 게시글 번호도 포함해 더 고유하게 만들 수 있음

**✅ 4단계: 파일 업로드/다운로드 유틸리티 구현**

**📁 main>java>org.scoula>common.util>UploadFiles.java**

**📍 upload() //인자로 저장 위치, 실제 데이터 넘김**

public static String upload(String baseDir, MultipartFile part) throws IOException { //예외 던짐으로 대응

File base = new File(baseDir); //기본 디렉토리 있는지 확인하고 없으면 새로 생성

if (!base.exists()) base.mkdirs(); // 중간 폴더까지 생성

String originalName = part.getOriginalFilename();//클라에서 사용한 파일 이름

String savedName = UploadFileName.getUniqueName(originalName);//서버 저장용 uuid이름.

File dest = new File(baseDir, savedName);//저장용 파일 객체 생성. 생성시 파일명 지정.

part.transferTo(dest); // 서버 파일 시스템으로 이동. 지정한 경로로.  
 //톰캣의 임시파일로 존재하거나 메모리에 존재하는 파일을 서버의 FS로 옮겨라.   
 //즉 이미 실질적인 업로드는 이미 된 상태. 영속성을 위해서 서버에 제대로 저장하는 일.

return dest.getPath();//저장한 파일 경로 리턴

}

**📍 getFormatSize()**

public static String getFormatSize(Long size) { //나중에 뷰에서 사용할 함수.

final String[] units = {"Bytes", "KB", "MB", "GB"};

int unit = (int) (Math.log10(size) / Math.log10(1024));

return new DecimalFormat("#,##0.#").format(size / Math.pow(1024, unit)) + " " + units[unit];

}

**📍 download() 정적 메서드 //컨트롤러에서 받은 response객체 이용, 서버에 있는 파일 객체 인자로.**//컨트롤러에서 첨부파일 다운url 요청 담당 메서드가 호출할 예정

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

response.setContentType("application/download");

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

String encodedName = URLEncoder.encode(orgName, "UTF-8"); //원래 이름

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

//”attatchment;filename=…” Content-disposition 설정 : 저장 대화상자 유도.  
 // Content-disposition 설정 안하면 바로보기로 넘어감 혹은 inline으로 지정

try (OutputStream os = response.getOutputStream();

BufferedOutputStream bos = new BufferedOutputStream(os)) {//성능 최적화. binary데이터잖아.

//서버의 파일을 스트림으로 읽어서 response에 출력  
Files.copy(file.toPath(), bos);

}

}

**✅ 5단계: MyBatis Mapper 구성**

**확장**

**📄 Mapper java>org.scoula>ex>mapper>BoardMapper.java //인터페이스 영속성 🡺 VO하기 위한.**

void createAttachment(BoardAttachmentVO vo);

List<BoardAttachmentVO> getAttachmentList(Long bno);

BoardAttachmentVO getAttachment(Long no);

int deleteAttachment(Long no);

**📄 XML 매핑 파일 resources>org>scoual>board>mapper> BoardMapper.xml**

<selectid="get" resultMap="boardMap">   
selectb.\*, a.noasano, a.bno,a.filename,a.path, a.content\_type,a.size, a.reg\_dateasa\_reg\_date fromtbl\_boardb leftouterjointbl\_board\_attachmenta onb.no=a.bno whereb.no=#{no} orderbyfilename   
</select>  
  
<insertid="createAttachment">  
 insertintotbl\_board\_attachment(filename,path,content\_type,size,bno) values(#{filename},#{path},#{contentType},#{size},#{bno})   
</insert>   
<selectid="getAttachmentList"resultType="org.scoula.board.domain.BoardAttachmentVO"> select\*fromtbl\_board\_attachment wherebno=#{bno} orderbyfilename //게시글 상세보기 FK기준으로.  
</select>  
<selectid="getAttachment"resultType="org.scoula.board.domain.BoardAttachmentVO"> select\*fromtbl\_board\_attachment whereno=#{no}   
</select>  
<deleteid="deleteAttachment">  
deletefromtbl\_board\_attachment whereno=#{no}  
</delete>

<resultMapid="attachmentMap"type="org.scoula.board.domain.BoardAttachmentVO"> //조인처리위한<idcolumn="ano" property="no"/> <resultcolumn="bno" property="bno"/> <resultcolumn="filename" property="filename"/> <resultcolumn="path" property="path"/> <resultcolumn="contentType"property="contentType"/> <resultcolumn="size"property="size"/> <resultcolumn="a\_reg\_date" property="regDate"/> </resultMap>   
  
<resultMapid="boardMap"type="org.scoula.board.domain.BoardVO"> <idcolumn="no"property="no"/> <resultcolumn="title"property="title"/> <resultcolumn="content"property="content"/> <resultcolumn="writer"property="writer"/> <resultcolumn="reg\_date"property="regDate"/> <resultcolumn="update\_date"property="updateDate"/> <collectionproperty="attaches"resultMap="attachmentMap"/> </resultMap>

* 첨부파일은 게시글의 attaches 필드에 자동으로 들어감
* 첨부가 없어도 게시글은 조회돼야 하므로 **LEFT JOIN 필수**

**✅ 6단계: BoardVO, DTO 설계**

**📦 BoardVO 확장**

public class BoardVO {

private Long no;

...

private List<BoardAttachmentVO> attaches; 1:N관계 추가.

…

}

**📦 BoardDTO 확장**

public class BoardDTO {

…

private List<MultipartFile> files=new ArrayList<>( ); //사용자가 실제 업로드한 파일(multipart) 목록

private List<BoardAttachmentVO> attaches //첨부파일 매퍼를 통해 쿼리 결과가 들어갈 애.  
 //DB에 저장된 파일 정보

//VO-->DTO 변환

publicstaticBoardDTOof(BoardVOvo){

returnBoardDTO.builder()

…

.attaches(vo.getAttaches())

.regDate(vo.getRegDate())

.updateDate(vo.getUpdateDate())

.build();

}

//DTO-->VO 변환

publicBoardVOtoVo(){

returnBoardVO.builder()

…

.attaches(attaches)

.regDate(regDate)

.updateDate(updateDate)

.build();

}

}

📌 폼 전송된 <input type="file" name="files">는 files 필드에 바인딩됨

폼 입려겨시 매핑된 컨트롤러의 메소드를 통해서 files멤버에 binary데이터 들어감.

**✅ 7단계: Service 계층 구현 + 트랜잭션 처리**

**인터페이스 확장 BoardService.java**

publicinterfaceBoardService{

…

Public BoardAttachmentVO getAttachment(Longno);//추가

Public boolean deleteAttachment(Longno);//삭제

}

RootConfig에 @EnableTransactionManagement추가.

서비스 계층에서 @Transactional 클래스 레벨, 메서드 레벨 적용가능. 메서드 RuntimeException시 롤백

**📄 게시글 생성**

**BoardServiceImpl.java//확장 및 수정**

@Override @Transactional //2개이상의 insert문이 실행될 수 있으므로 트랜잭션 처리 필요

public void create(BoardDTO dto) {

BoardVO board = dto.toVo();

mapper.create(board);

List<MultipartFile> files = dto.getFiles();

if (files != null && !files.isEmpty()) { //첨부파일이 있는 경우

upload(board.getNo(), files); //서비스 레이어의 업로드 메서드 실시. 업로드 및 DB 등록 수행

}

}

//업로드는 게시물이 create될 때 될 수도 있음.//즉 컨트롤러에 따로 업로드용 메서드를 만들필요 없이  
//원래 게시물 만드는 메서드에서 해당 서비스의 메서드를 호출할 때 업로드가 진행됨.   
//하지만 다운로드는 따로 요청 url있고 따로 다운로드 요청 url을 담당하는 컨트롤러 메서드를 정의해서 처리해야함

**📄 업로드 처리**

private void upload(Long bno, List<MultipartFile> files) {

for (MultipartFile part : files) {

if (part.isEmpty()) continue;

try {

String path = UploadFiles.upload(BASE\_DIR, part); //전에 만든 유틸리티 사용. 예외 던질수도.

BoardAttachmentVO vo = BoardAttachmentVO.of(part, bno, path);

mapper.createAttachment(vo);

} catch (IOException e) {

throw new RuntimeException(e); // 트랜잭션 롤백 유도

}

}  
 // 첨부파일 한 개 얻기 //컨트롤러에서 첨부파일 다운url 요청 담당 메서드가 호출할 예정  
 @Override public BoardAttachmentVO getAttachment(Long no) { return mapper.getAttachment(no); }   
 // 첨부파일 삭제  
 @Override public boolean deleteAttachment(Long no) { return mapper.deleteAttachment(no) == 1; }

}

📌 IOException은 체크 예외이므로 RuntimeException으로 감싸야 @Transactional에서 rollback 됨

**✅ 8단계: JSP 업로드 폼 구성**

<form method="post" enctype="multipart/form-data">   
//action속성 미지정시 현재페이지의 URL로 요청

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

</form>

* multiple → 다중 업로드 허용
* name="files" → DTO 필드명과 일치 // BoardDTO의 files 필드에 자동으로 매핑
* enctype="multipart/form-data"는 필수 설정

**✅ 9단계: 첨부파일 목록 출력 (상세 조회)**

<c:forEach var="file" items="${board.attaches}">

<a href="/board/download/${file.no}"> //컨트롤러에서 @GetMapping(“download/동적패스변수”)예정

${file.filename} (${file.fileSize})

</a>

</c:forEach>

* fileSize는 getFileSize()로 계산된 포맷 문자열

**✅ 10단계: 다운로드 요청 처리**

서버에 저장된 경로를 웹 url로 표현할 수 없음 == 직접 링크를 걸 수 없음

controller에서 download요청을 받았을 때 직접 파일을 읽어서 처리.

파일 조회 후, 유틸 메서드로 응답 처리.

요청 url은 /board/download/변수 형식.

@GetMapping("/download/{no}")

@ResponseBody //뷰 리솔버X. 다른 방법으로 응답. 파일로 응답

public void download(@PathVariable Long no//파일번호 추출, HttpServletResponse response) {

BoardAttachmentVO attach = service.getAttachment(no);

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

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

}

* @ResponseBody → 뷰 사용 안 함 == 뷰 리솔버 안 거치고
* UploadFiles의 download메소드를 통해 바이너리 데이터를 응답으로 전송. JSON변환X.
* UploadFiles의 download메소드안에서 파일명 URLEncoder.encode()로 한글 파일명 인코딩
* 응답 헤더 설정 후 직접 파일 스트림 출력