

2025년 상반기 K-디지털 트레이닝

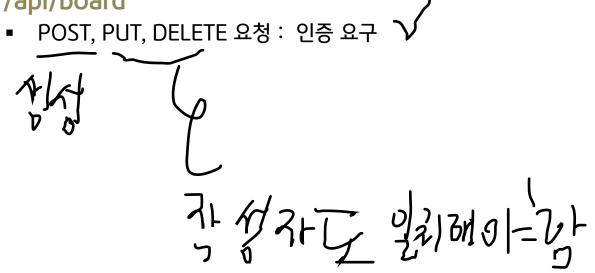
게시판 - 백엔드

[KB] IT's Your Life



💟 접근 권한 설정

o /api/board



```
.authorizeRequests() // 경로별 접근 권한 설정
                                                                     .antMatchers(HttpMethod.OPTIONS).permitAll()
                                                                       .antMatchers(HttpMethod.POST, "/api/member").authenticated()
SecurityConfig.java
                                                                     .ahtMatchers(HttpMethod.PUT, ⊘ "/api/member", ⊘ "/api/member/*/chan
                                                                     .antMatchers(HttpMethod.POST, ∅ "/api/board/**").authenticated()
public class SecurityConfig extends WebSecurityConfigurerAdapter {
                                                                     .antMatchers(HttpMethod.PUT, ∅ "/api/board/**").authenticated()
                                                                     .antMatchers(HttpMethod.DELETE, ⊘ "/api/board/**").authenticated()
   @Override
                                                                     .anyRequest().permitAll();
    public void configure(HttpSecurity http) throws Exception {
                                                                  다 모드 정근 원육
                                                                                       최종적으로
       http
                                  *하나는 해당 레벨 모든것 **2개는 하위 쭉 다
            .authorizeRequests()
            .antMatchers(HttpMethod.POST, "/api/board/**").authenticated()
            .antMatchers(HttpMethod.PUT, "/api/board/**").authenticated()
            .antMatchers(HttpMethod.DELETE, "/api/board/**").authenticated()
            .anyRequest().permitAll();
        http.httpBasic().disable() // 기본 HTTP 인증 비활성화
            .csrf().disable() // CSRF 비활성화
            .formLogin().disable() // formLogin 비활성화 관련 필터 해제
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS); // 세션 생성 모드 설정
```

lc class SecurityConfig extends WebSecurityConfigur€ △ 11 △ 1 ✓ 4 △ ∨

rlic void configure(HttpSecurity http) throws Exception {

$^{ec{}}$ 첨부 파일 테이블 제약조건 다시 설정 $^{ extstyle \sqrt{}}$

게시글 삭제 시 연관 첨부 파일 자동 삭제

```
DROP TABLE IF EXISTS tbl_board_attachment;
                                                        1대N관계인
                                                        테이블까지 같이 삭제되게
CREATE TABLE tbl_board_attachment
           INTEGER AUTO_INCREMENT PRIMARY KEY, -- PK
   no
   filename VARCHAR(256) NOT NULL, -- 원본 파일 명
           VARCHAR(256) NOT NULL, -- 서버에서의 파일 경로
   path
   content_type VARCHAR(56),
                          -- content-type
   size
           INTEGER,
                              -- 파일의 크기
           INTEGER NOT NULL, -- 게시글 번호, FK
   bno
   reg date DATETIME DEFAULT now(),
   CONSTRAINT FOREIGN KEY (bno) REFERENCES tbl_board (no) ON DELETE CASCADE
);
```

Mapper

- 게시글 1<u>개 추</u>출 시 조인을 통해 첨부파일 목록을 같이 구성
- LEFT OUTER JOIN 서 첨부파일 없는 게시판이 존재할 수 있잖아!
 - 기준 테이블: tbl_board

SELECT b.*, a.no as ano, a.bno, a.filename, a.path, a.content_type, a.size, a.reg_date 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

BoardMapper.java

```
public interface BoardMapper {
    List<BoardVO> getList();
    BoardVO get(Long no);
    void create(BoardVO board);
    int update(BoardV0 board);
    int delete(Long no);
    void createAttachment(BoardAttachmentVO attach);
    List<BoardAttachmentV0> getAttachmentList(Long bno);
    BoardAttachmentVO getAttachment(Long no);
    int deleteAttachment(Long no);
```

BoardMapper.xml

```
<resultMap id="attachmentMap" type="org.scoula.board.domain.BoardAttachmentV0">
    <id column="ano"</pre>
                                  property="no"/>
    <result column="bno"</pre>
                                   property="bno"/>
    <result column="filename"</pre>
                                   property="filename"/>
    <result column="path"</pre>
                                   property="path"/>
    <result column="contentType" property="contentType"/>
    <result column="size"</pre>
                                   property="size"/>
    <result column="a_reg_date"</pre>
                                     property="regDate"/>
</resultMap>
<resultMap id="boardMap" type="org.scoula.board.domain.BoardV0">
    <id column="no"
                                   property="no"/>
    <result column="title"</pre>
                                   property="title"/>
    <result column="content"</pre>
                                   property="content"/>
    <result column="writer"</pre>
                                   property="writer"/>
    <result column="reg date"</pre>
                                   property="regDate"/>
    <result column="update_date" property="updateDate"/>
    <collection property="attaches" resultMap="attachmentMap"/>
</resultMap>
```

List<BoardAttatchmentVO> 필드에 매핑되게찌

```
<select id="get" resultMap="boardMap">
   select b.*, a.no <u>as an</u>o, a.bno, a.filename, a.path, a.content_type, a.size, a.reg_date 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>
<insert id="create">
    insert into tbl_board (title, content, writer)
    values (#{title}, #{content}, #{writer})
    <selectKey resultType="Long" keyProperty="no" keyColumn="no" order="AFTER">
        SELECT LAST_INSERT_ID()
   </selectKey>
</insert>
```

```
<update id="update">
    update tbl_board set
    title = #{title},
    content = #{content},
    writer = #{writer},
    update_date = now()
    where no = #{no}
    </update>

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

```
<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="BoardAttachmentV0">
        select * from tbl board attachment where bno = #{bno}
       order by filename
    </select>
    <select id="getAttachment" resultType="BoardAttachmentV0">
        select * from tbl_board_attachment
       where no = \#\{no\}
   </select>
    <delete id="deleteAttachment">
       delete from tbl_board_attachment
       where no = #{no}
   </delete>
</mapper>
```

게시판 - 백에드 디비처리 마무리 했으니 서비스 레이어로

🕜 서비스

- 추가, 수정 시 첨부파일 업로드 처리
- o **첨부파일 하나 추출 기능** 다운로드
- 첨부파일 하나 삭제 기능

글 수정시 첨부파일도 수정할 수 있지. 그럴때 삭제후 추가

BoardService.java

```
package org.scoula.board.service;
public interface BoardService {
    List<BoardDTO> getList();
    BoardDTO get(Long no);
    BoardDTO create(BoardDTO board);
    BoardDTO update(BoardDTO board);
    BoardDTO delete(Long no);
    BoardAttachmentVO getAttachment(Long no);
    boolean deleteAttachment(Long no);
```

```
package org.scoula.board.service;
@Log4j2
@Service
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService {
    private final static String BASE_DIR = "c:/upload/board"; 
    private final BoardMapper mapper;
    @Override
    public List<BoardDTO> getList() {
        log.info("getList....");
        return mapper.getList().stream()
                .map(BoardDT0::of)
                .toList();
```

```
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.from(part, bno, uploadPath);
            mapper.createAttachment(attach);
        } catch (IOException e) {
            throw new RuntimeException(e);
            //log.error(e.getMessage());
        }
    }
}
```

```
@Transactional // 2개 이상의 insert 문이 실행될 수 있으므로 트랜잭션 처리 필요
@Override
public BoardDTO create(BoardDTO board) {
   log.info("create....." + board);
   BoardV0 boardV0= board.toVo();
   mapper.create(boardV0); 
   // 파일 업로드 처리
   List<MultipartFile> files = board.getFiles();
   if(files != null && !files.isEmpty()) {
       upload(boardV0.getNo(), files);
   return get(boardV0.getNo());
```

```
@Override
public BoardDTO update(BoardDTO board) {
    log.info("update..... " + board);
   BoardV0 boardV0 = board.toVo();
   log.info("update..... " + boardV0);
   mapper.update(boardV0);
   // 파일 업로드 처리
   List<MultipartFile> files = board.getFiles();
   if(files != null && !files.isEmpty()) {
       upload(board.getNo(), files);
    return get(board.getNo());
```

```
@Override
public BoardDTO delete(Long no) {
   log.info("delete...." + no);
   BoardDTO board = get(no);
   mapper.delete(no);근데 DB상에서 경로 정보가 삭제되는 거지. 파일 자체가 삭제되진 않는다.
   return board;
@Override
public BoardAttachmentVO getAttachment(Long no) {
   return mapper.getAttachment(no);
@Override
public boolean deleteAttachment(Long no) {
   return mapper.deleteAttachment(no) == 1;
```

🗸 컨트롤러

- o RestController 구성
- 첨부파일 다운로드 및 삭제 처리

HTTP 메서드	URL	메서드명	리턴
GET		getList()	ResponseEntity < List < Board DTO > >
GET	/{no}	get()	ResponseEntity < BoardDTO >
POST		create()	ResponseEntity < BoardDTO >
PUT	/{no}	update()	ResponseEntity < BoardDTO >
DELETE	/{no}	delete()	ResponseEntity < BoardDTO >
GET	/download/{no}	download	void
DELETE	/deleteAttachment/{no}	deleteAttachment	ResponseEntity < Boolean >

☑ BoardController.java (Swagger 코드 생략)

```
package org.scoula.board.controller;
@RestController
@RequestMapping("/api/board")
@RequiredArgsConstructor
@Log4j2
public class BoardController {
    private final BoardService service;
    @GetMapping("")
    public ResponseEntity<List<BoardDTO>> getList() {
        return ResponseEntity.ok(service.getList());
    @GetMapping("/{no}")
    public ResponseEntity<BoardDTO> getById(@PathVariable Long no) {
        return ResponseEntity.ok(service.get(no));
```

☑ BoardController.java (Swagger 코드 생략)

```
@PostMapping("")
                                                                ** 주의 **
public ResponseEntity<BoardDTO> create(BoardDTO board) {
                                                                Multipart 인코딩이므로
@RequestBody 붙이지 않음
    return ResponseEntity.ok(service.create(board));
@PutMapping("/{no}")
public ResponseEntity<BoardDTO> update(@PathVariable Long no, BoardDTO board) {
    return ResponseEntity.ok(service.update(board));
@DeleteMapping("/{no}")
public ResponseEntity<BoardDTO> delete(@PathVariable Long no) {
    return ResponseEntity.ok(service.delete(no));
```

☑ BoardController.java (Swagger 코드 생략)

```
@GetMapping("/download/{no}")
public void download(@PathVariable Long no, HttpServletResponse response) throws Exception {
    BoardAttachmentV0 attachment = service.getAttachment(no);
    File file = new File(attachment.getPath()); V
    UploadFiles.download(response, file, attachment.getFilename());
@DeleteMapping("/deleteAttachment/{no}")
public ResponseEntity<Boolean> deleteAttachment(@PathVariable Long no) throws Exception {
    return ResponseEntity.ok(service.deleteAttachment(no));
```

번외: 지금은 백엔드 완성하고 프론트를 만드는 순서다. 실제로는 맡은 파트와 구성에 따라 달라질수있다.