

게시판 프로젝트 개발 문서

0. 목차

1. 프로젝트 개요
 2. DB 설계
 3. 서비스 URL
 4. 접근 제어(권한 처리)
 5. 예외 처리
 6. 보안 설정
 7. 기능 상세
-

1. 프로젝트 개요

1.1 프로젝트명

- 게시판 프로젝트: Spring Boot 기반 웹 애플리케이션

1.2 주요 기능

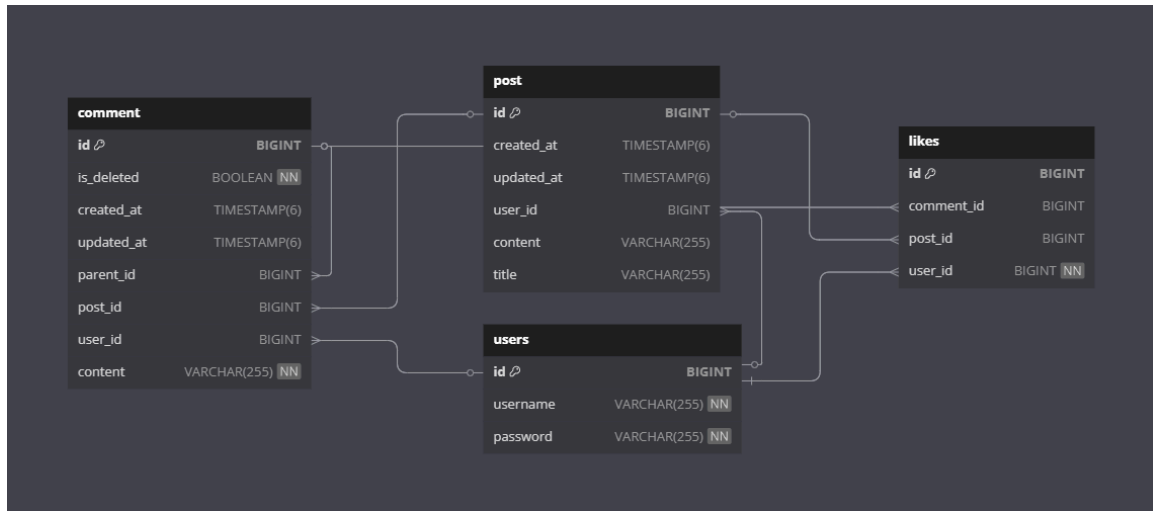
- 회원 CRUD
- 게시판 CRUD
- 댓글(대댓글) CRUD
- 게시글 및 댓글 좋아요 기능

1.3 기술 스택

- **백엔드**: Spring Boot 3.4.2, Spring Security, JPA, H2 database, MyBatis, swagger-ui, Bean Validation
 - **프론트엔드**: Thymeleaf, Bootstrap
 - **빌드 도구**: Gradle
-

2. DB 설계

2.1 ERD



2.2 DB 테이블 설명

- **USERS**

- Bcrypt 방식으로 인코딩된 비밀번호 저장됨

- **COMMENT**

- `parent_id` 값을 통해 댓글 또는 대댓글 구분

- **LIKES**

- 좋아요 등록 시 (`post_id`, `user_id`) 또는 (`comment_id`, `user_id`) 값으로 데이터가 생성되고,
좋아요 해제 시 데이터 삭제
- unique 제약 조건 적용: `user_id`당 같은 `post_id` 또는 `comment_id` 데이터 생성 방지
- check 제약 조건 적용: `post_id`와 `comment_id` 둘 다 NULL이 되지 않도록 함

3. 서비스 URL

3.1 swagger 문서 참고

<http://localhost:8989/swagger-ui/index.html#/>

4. 접근 제어(권한 처리)

4.1 Spring Security 설정

```
▼ @Configuration no usages Lee-Hyeonji99 +1
@EnableWebSecurity
@EnableMethodSecurity
public class SpringSecurityConfiguration {

    @Bean no usages Lee-Hyeonji99 +1
    > public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {...}

    @Bean no usages Lee-Hyeonji99
    > public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

}
```

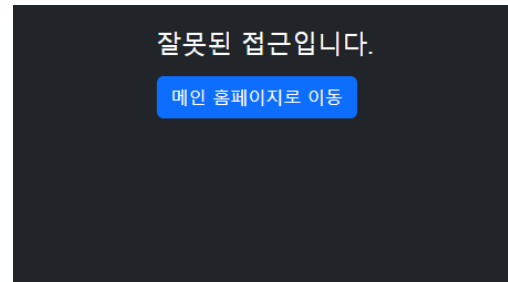
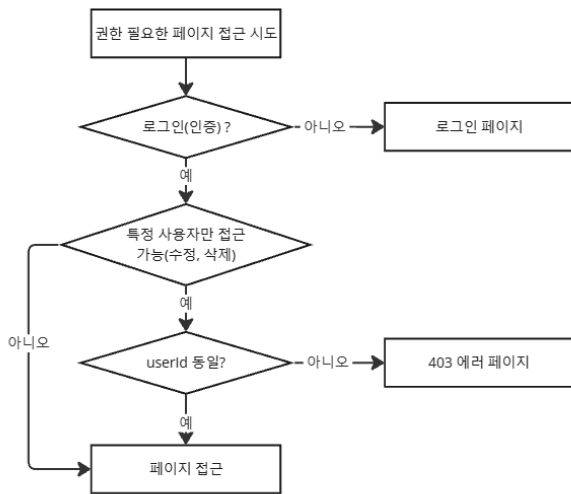
- SecurityConfig 클래스에 `@EnableMethodSecurity` 애너테이션 적용

4.2 Controller Method 설정

```
@GetMapping(value = "/posts/{postId}/edit")
@PreAuthorize("isAuthenticated()")
public String showEditPost(@PathVariable long postId, Model model,
                           @AuthenticationPrincipal User user) {...}
```

- 사용자 인증(로그인) 필요한 주소의 Controller 클래스 맵핑 메서드에 `@PreAuthorize("isAuthenticated()")` 애너테이션 적용

4.3 접근 제어 로직



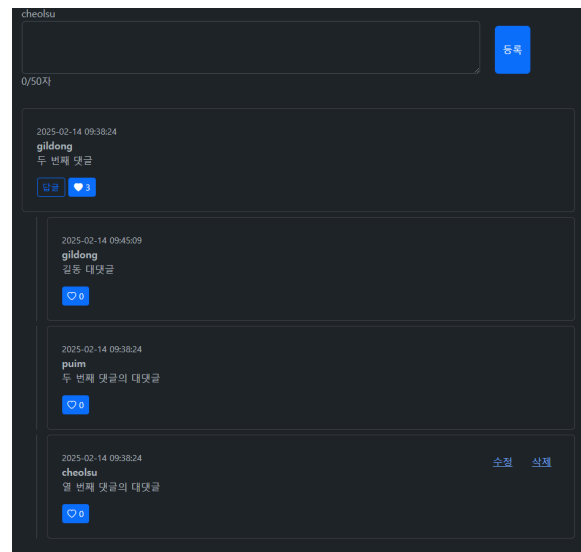
로그인 상태에서 user_id 동일하지 않은 글의 수정 페이지 url 이동시 보여지는 에러 페이지 화면

- Spring Security의 @PreAuthorize 애너테이션을 통해 접근 권한 제어
- 인증(로그인) 되지 않은 상태에서 접근시 로그인 페이지로 강제 이동

4.4 권한에 따른 가시성 제어



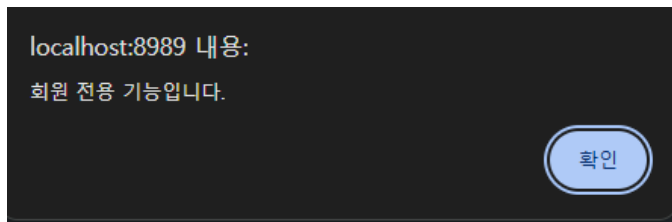
아이디가 gildong인 사용자의 게시물 상세 댓글 영역



아이디가 cheolsu인 사용자의 게시물 상세 댓글 영역

- 로그인된 유저에 따른 댓글 수정/삭제 권한 버튼 가시성 변경

4.5 권한 미충족시 alert로 제출 방지



미로그인회원 답글, 좋아요 버튼 클릭시 alert 표시

5. 예외 처리

5.1 접근 예외

- AccessDenied() 예외 발생 ⇒ 403 에러 페이지 반환

5.2 DB 리소스 접근 예외

- 사용자 정의 런타임 예외 클래스 ResourceNotFound() 예외 발생 ⇒ 404 에러 페이지 반환

6. 보안 설정

6.1 Spring Security 설정

```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

    http
        .authorizeHttpRequests(
            authorizeRequests → {
                authorizeRequests.requestMatchers("/**").permitAll();
                authorizeRequests.requestMatchers("/h2-console/**").permitAll();
            })
        .formLogin(login → login
            .loginPage("/login")
            .loginProcessingUrl("/login"))
}
```

```

        .defaultSuccessUrl("/", true) // 로그인 성공 시 리다이렉트
        .failureUrl("/login?error=true")
        .permitAll()
    )
    .sessionManagement( session →
        session.maximumSessions(1) // 사용자당 최대 세션 수 1로 제한
    )
    // h2 콘솔 확인
    .headers(
        headersConfigurer →
            headersConfigurer.frameOptions(HeadersConfigurer.FrameOptionsConfig::sameOrigin)
    ).csrf(csrf → csrf
        .ignoringRequestMatchers("/h2-console/**") // H2 콘솔에 대한 CSRF 보호를 비활성화
    );

    return http.build();
}

```

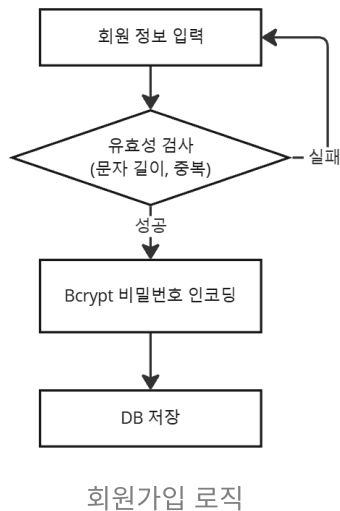
6.2 Spring Security 설정 설명

- 메서드 단위로 접근 권한 설정
 - 클래스에 @EnableMethodSecurity 적용
- global 페이지 접근 설정
 - /** 프로젝트 전체 경로에 접근 가능
- csrf
 - /h2-console/ 을 제외한 경로에 csrf 보호 설정
- 로그인 페이지
 - formLogin 페이지 지정
- 세션 설정
 - 사용자당 최대 세션 수를 1로 제한

7. 화면 및 기능 상세

7.1 회원 기능

1. 회원가입



회원가입

아이디

아이디는 영문과 숫자만 입력 가능합니다.
아이디는 2자 이상 16자 이하로 입력해주세요.

비밀번호

비밀번호는 2자 이상 20자 이하로 입력해주세요.

비밀번호 확인

비밀번호는 2자 이상 20자 이하로 입력해주세요.

회원가입

회원가입 폼 화면

- 아이디 정규표현식 유효성 검증 적용
- 유효성 검증에 실패할 경우
BindingResult 통해 에러 메시지 표시

SELECT * FROM USERS;		
ID	PASSWORD	USERNAME
1	\$2a\$10\$jHDY3ArQrZA9FoRQDOdXJeMdYSPYBC32cPQMwp6QFadUjrmVA3zrC	puim
2	\$2a\$10\$17fSXsO76rsw5czPrDlnWu2H1Xp3aPrND4lCBj18JJ6Bo.tijVz5.	gildong
3	\$2a\$10\$mj99zjrPhawK6dLJFb/GGe1m0DIHfVj4w0Hg/dxkwmNUIVmQUYyp	cheolsu
4	\$2a\$10\$J/vYQOphoPptEtADVXdFurV6wF6GEIgeRX25xcqacO6HnZm2JXIS	yeonghee

(4 rows, 0 ms)

H2 DB의 USERS 테이블 데이터 조회 화면

- 비밀번호는 BcryptPasswordEncoder 로 암호화 되어 저장됨

2. 로그인 및 로그아웃

- 로그인
 - securityconfig filterchain에 설정한 formLogin을 통해 로그인
 - 로그인에 성공할 경우 session의 SPRING_SECURITY_CONTEXT에 회원 정보가 저장됨
 - 세션에 저장된 SPRING_SECURITY_CONTEXT 정보를 기반으로 권한 처리/사용자 정보 등에 접근하여 사용
=> @AuthenticationPrincipal 애너테이션으로 로그인 상태의 user 객체 접근 가능
- 로그아웃
 - 세션에 저장된 SPRING_SECURITY_CONTEXT 값을 null로 설정

3. 회원 정보 변경

비밀번호 변경 폼 화면:
아무 값도 입력 x

비밀번호 변경 폼 화면:
현재와 동일한 비밀번호로 변경 요청시

- bean validation 유효성 검사

```
public boolean equalsCurrent(long userId, String password) { 2 usages
    User findUser = userRepository.findById(userId).orElseThrow();
    return passwordEncoder.matches(password, findUser.getPassword());
}
```

DB에 저장된 비밀번호와 일치하는지 비교하는 코드

- 현재 비밀번호와 비교시 BcryptPasswordEncoder.matches() 메서드로 판별

4. 회원 탈퇴

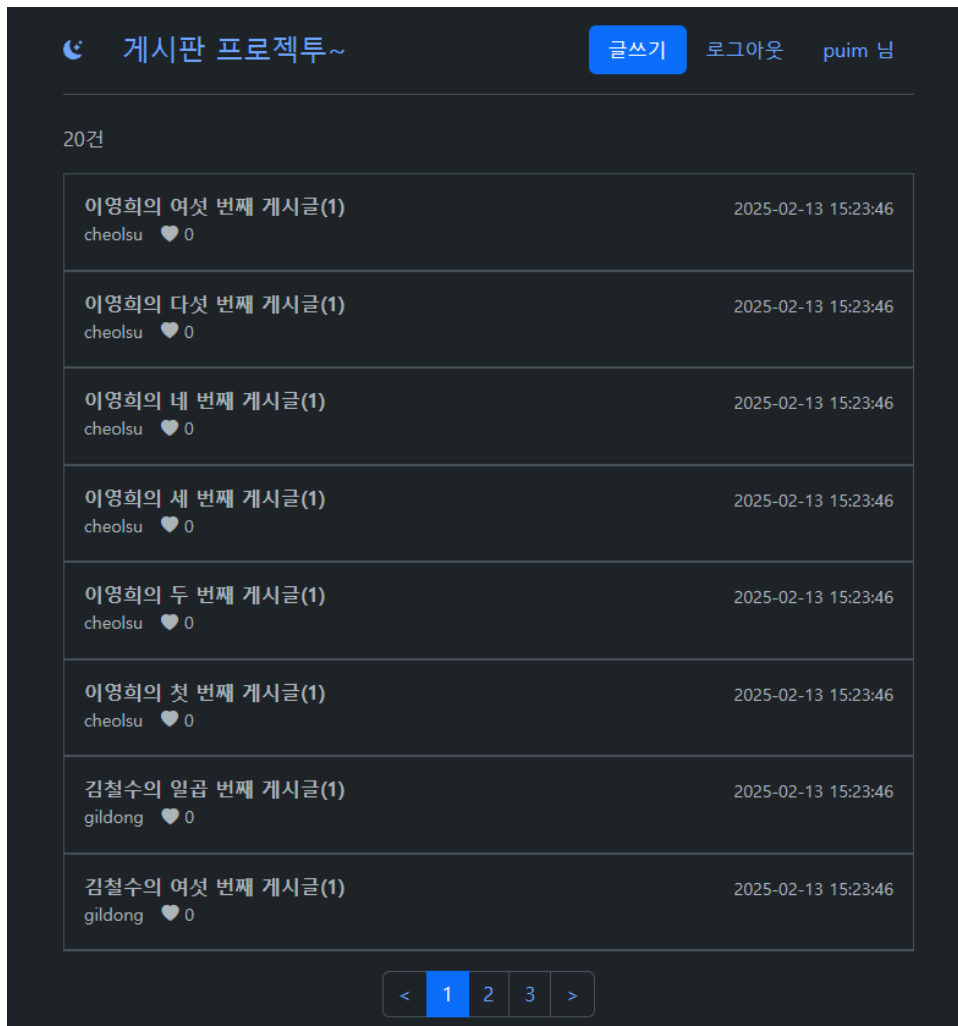
```
@Transactional 1 usage  👤 PUIM-PC#이현지
public void deleteUserById(long id) {
    likesService.deleteByUserId(id);
    postService.deletePostsByUserId(id);
    commentService.deleteByUserId(id);
    userRepository.deleteById(id);
}
```

회원 탈퇴 서비스 코드

- 회원 탈퇴시 회원이 등록한 게시글, 좋아요, 댓글, 회원 데이터 삭제
- UserService의 삭제 메서드에 @Transactional 을 적용하여 일부 데이터만 삭제되는 일이 발생하지 않도록 함

7.2 게시글 기능

1. 게시글 조회



게시글 목록 페이지

- 게시글 목록
 - MyBatis 활용하여 postId 컬럼으로 likes, comment 조인 쿼리 반환된 DTO (총 게시글 개수 + 게시글 정보 + 좋아요 개수 + 댓글 개수) 리스트 렌더링
- 페이지네이션
 - 게시판 목록 페이지네이션(페이지당 8건씩)
 - javascript로 페이지네이션 생성 및 클릭 이벤트 바인딩, 현재 페이지 태그 class 속성에 active 값 적용



게시글 상세 조회 페이지

- 게시글: 컨트롤러에서 반환된 PostViewDTO
- 좋아요: likes.js 에서 관리(DOM 로드시 데이터 조회)
- 댓글: comment.js 에서 관리(DOM 로드시 데이터 조회)

2. 게시글 등록

게시판 프로젝트~ 글쓰기 로그인 puim 님

제목

내용

0/80자

등록 취소

© 2024 Company, Inc Home

게시글 등록 페이지

- 게시글 관련 기능은 api가 아닌 form 방식으로 요청하기 때문에 csrf 보호를 활성화 했음에도 POST 요청시 header에 따로 토큰 정보 포함 불필요
- 게시글 작성 및 수정시 글자 수 표시 및 최대 글자 수 제한 적용

3. 게시글 수정

게시판 프로젝트~
글쓰기
로그아웃
puim 님

글 수정

제목

홍길동의 첫 번째 게시물

내용

이것은 홍길동의 첫 번째 게시물입니다.

21/80자

수정
취소

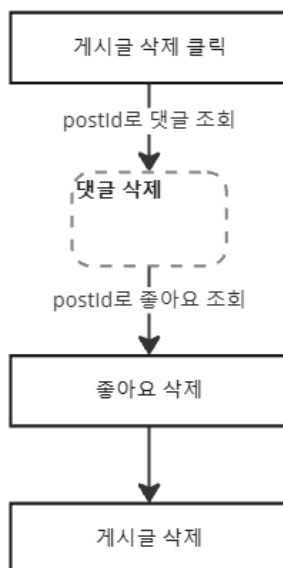
© 2024 Company, Inc

Home

게시글 수정 페이지

- `@AuthenticationPrincipal` 에 저장된 `user`의 `id`값과 `@RequestBody`를 통해 전달된 `postUpdateDTO.userId` 동일한 경우에만 성공

4. 게시물 삭제



게시글 삭제시 서비스 로직

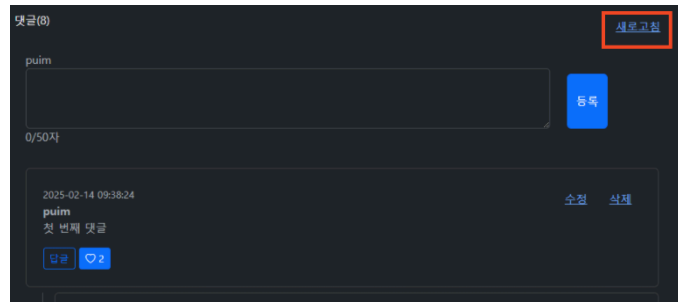
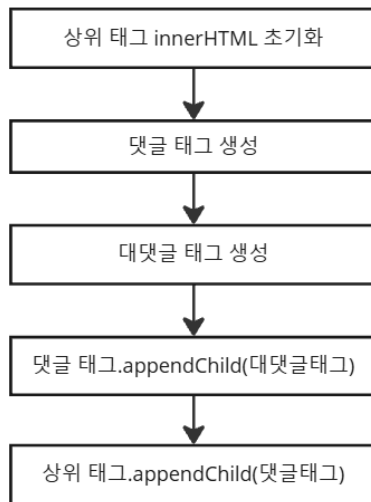
```

@Transactional 2 usages
public void deletePostById(long postId) {
    likesService.deleteByPostId(postId);
    commentService.deleteByPostId(postId);
    postRepository.deleteByPostId(postId);
}
  
```

서비스 코드:
게시글 및 게시글의 좋아요, 댓글 삭제

7.3 댓글(대댓글) 기능

1. 댓글 조회(새로고침)

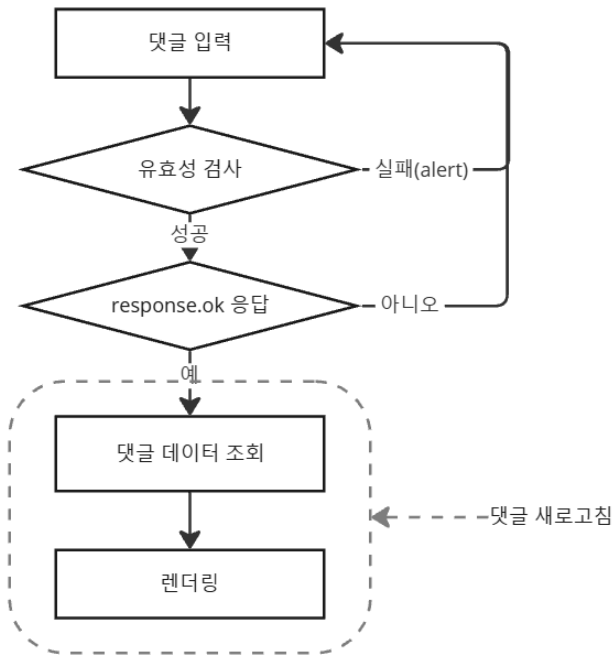


게시글 상세 페이지 댓글 영역의 새로고침 버튼

댓글 조회 렌더링

- 댓글 부분의 데이터만 새로 렌더링함
- 페이지 로드시, 댓글 작성, 댓글 수정, 댓글 좋아요 요청이 완료되면 실행됨
- 댓글에는 답글 버튼이 있고, 대댓글에는 없음
- 수정 이력이 있는 경우 수정일시가 함께 출력

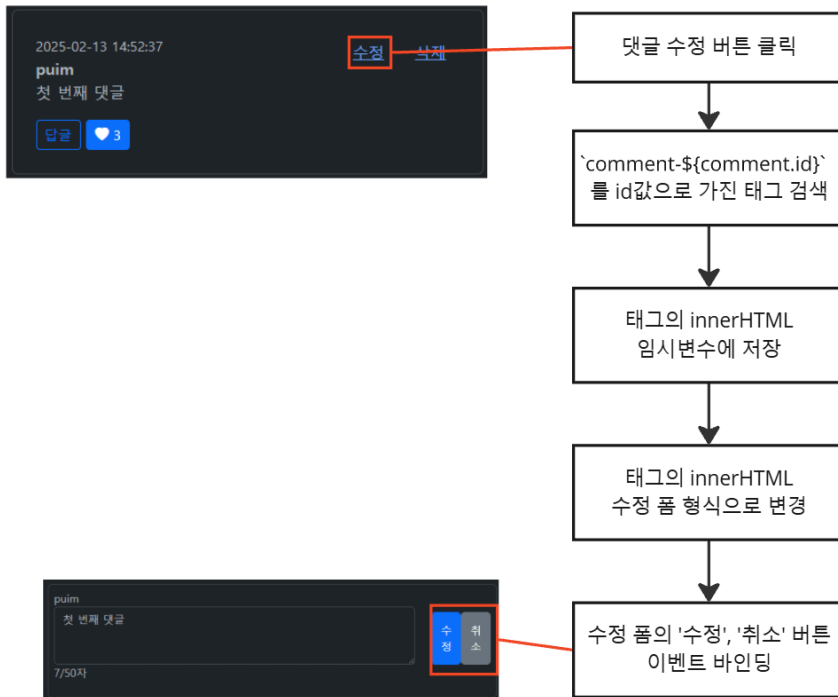
2. 댓글 등록



댓글 등록 버튼 클릭 이벤트

- csrf 토큰을 포함한 header 와 함께 POST 요청을 보냄
- 요청에 성공하여 response.ok 응답을 받으면 댓글 정보를 다시 조회하여 화면에 나타냄
- 댓글(대댓글) 작성 및 수정시 글자 수 제한 적용

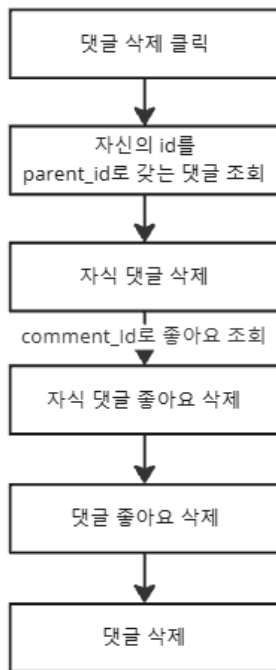
3. 댓글 수정



댓글(대댓글) 수정 폼 렌더링

1. 댓글 div 태그의 innerHTML을 임시 변수 값에 저장하고 innerHTML을 수정용 html 코드로 변경
2. 수정, 취소 버튼에 이벤트 바인딩
3. 수정 버튼 이벤트 리스너: patch 메서드 호출
4. 취소 버튼 이벤트 리스너: 임시 변수에 저장된 innerHTML 로 변경 후 답글(대댓글은 미포함), 수정, 삭제 버튼에 이벤트 다시 바인딩

4. 댓글 삭제



```

@Transactional 3 usages
public void deleteCommentById(long id) {
    List<CommentQueryDTO> childComments = commentMapper.findRepliesByParentId(id);

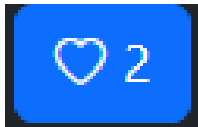
    for (CommentQueryDTO childComment : childComments) {
        likesRepository.deleteByCommentId(childComment.getId());
        commentRepository.deleteBy(commentRepository.getId());
    }

    likesRepository.deleteByCommentId(id);
    commentRepository.deleteBy(id);
}
  
```

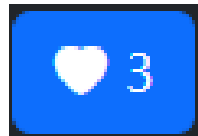
댓글 삭제 클릭시 서비스 코드:
자신(comment)의 id를 parent_id로 갖는
댓글과 댓글의 좋아요 데이터 모두 삭제

댓글 삭제 클릭시 서비스 로직

7.4 게시물 및 댓글 좋아요 기능



비로그인상태
or 좋아요
누르지 않은 경우



좋아요 누른 경우

- 게시물, 댓글(대댓글) 을 대상으로 함
- 좋아요 버튼을 누른 (user_id, post_id) 또는 (user_id, comment_id)가 일치하는 데이터가 있는 경우 칠해진 하트로 나타냄
- 로그인 되어있지 않거나 누르지 않은 경우는 빈 하트로 나타냄
- 좋아요 개수는 로그인 상태에 상관 없이 출력
- 등록 및 해제는 ajax의 POST 요청으로 실행
- response.ok를 응답으로 받으면 다시 조회한 결과를 화면에 렌더링