2024-2 오픈소스 개발 프로젝트01

SW중심대학사업단 강재구

(학연산 742호, 010-2278-8192, kangjk@cbnu.ac.kr)

06

충북대학교
SW중심대학사업단

03. SBB 서비스 개발하기 (2)

03

SBB 서비스 개발하기

3-07 로그인과 로그아웃 기능 구현하기

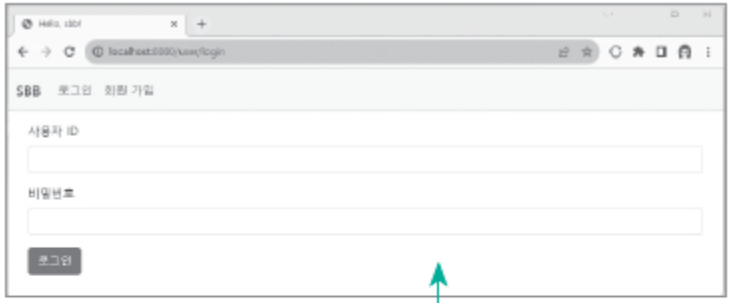
3-08 글쓴이 항목 추가하기

3-09 수정과 삭제 기능 추가하기

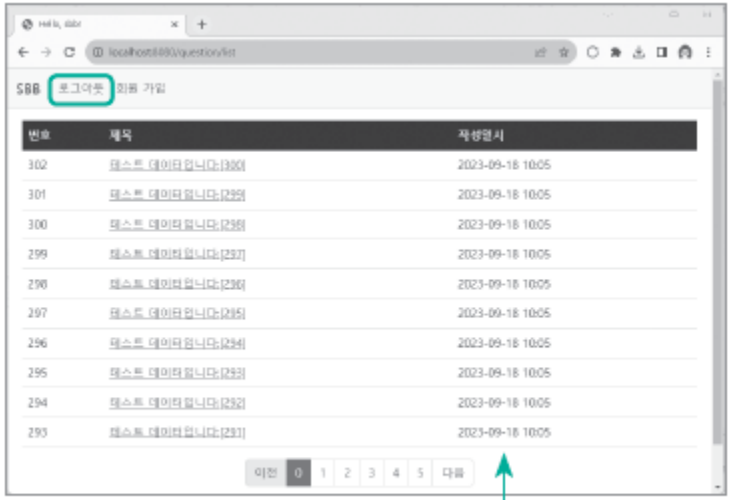
3-07 로그인과 로그아웃 기능 구현하기

- 회원 가입 기능을 완성했으니 이번에는 로그인과 로그아웃 기능을 구현해 보자

- SBB는 여러 사람이 사용하는 게시판 서비스이므로
  질문한 사람, 답변한 사람을 구별하는 목적으로 로그인은 필수 기능임



로그인 화면



로그인이 완료된 화면

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

- 회원 가입 단계에서 SITE_USER 테이블에 회원 정보를 저장함
- SITE_USER 테이블에 저장된 사용자명(사용자 ID)과 비밀번호로 로그인을 하려면 복잡한 단계를 거쳐야 함
- 하지만 스프링 시큐리티를 사용하면 이 단계를 보다 쉽게 진행할 수 있음

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 URL 등록하기

- 먼저 스프링 시큐리티에 로그인을 하기 위한 URL을 다음과 같이 설정하자

• SecurityConfig.java

```
(... 생략 ...)

@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((authorizeHttpRequests) -> authorizeHttpRequests
                .requestMatchers(new AntPathRequestMatcher("/**")).permitAll())
            .csrf((csrf) -> csrf
                .ignoringRequestMatchers(new AntPathRequestMatcher("/h2-
console/**")))
            .headers((headers) -> headers
                .addHeaderWriter(new XFrameOptionsHeaderWriter(
                    XFrameOptionsHeaderWriter.XFrameOptionsMode.SAMEORIGIN)))
            .formLogin((formLogin) -> formLogin
                .loginPage("/user/login")
                .defaultSuccessUrl("/"))
            ;
        return http.build();
    }

    @Bean
    PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

3-07 로그인과 로그아웃 기능 구현하기

- **로그인 기능 구현하기**
  - **로그인 URL 등록하기**
    - 여기서 추가한 .formLogin 메서드는 스프링 시큐리티의 로그인 설정을 담당하는 부분으로,
      설정 내용은 로그인 페이지의 URL은 /user/login이고
      로그인 성공 시에 이동할 페이지는 루트 URL(/)임을 의미함

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ User 컨트롤러에 URL 매핑 추가하기

- 스프링 시큐리티에 로그인 URL을 /user/login으로 설정했으므로 UserController에 해당 URL을 매핑해야 함
- 다음과 같이 코드를 추가해 보자

• UserController.java

```
(... 생략 ...)
public class UserController {

    private final UserService userService;

    (... 생략 ...)

    @GetMapping("/login")
    public String login() {
        return "login_form";
    }
}
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ User 컨트롤러에 URL 매핑 추가하기

- @GetMapping("/login")을 통해 /user/login URL로 들어오는 GET 요청을 이 메서드가 처리함. 즉, /user/login URL을 매핑함

- 매핑한 login 메서드는 login_form.html 템플릿을 출력하도록 만듦.
  실제 로그인을 진행하는 @PostMapping 방식의 메서드는
  스프링 시큐리티가 대신 처리하므로 직접 코드를 작성하여 구현할 필요가 없음

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 템플릿 작성하기

1. 로그인 화면을 구성하는 템플릿을 만들어 보자

• login_form.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <form th:action="@{/user/login}" method="post">
        <div th:if="${param.error}">
            <div class="alert alert-danger">
                사용자 ID 또는 비밀번호를 확인해 주세요.
            </div>
        </div>
        <div class="mb-3">
            <label for="username" class="form-label">사용자 ID</label>
            <input type="text" name="username" id="username" class="form-control">
        </div>
        <div class="mb-3">
            <label for="password" class="form-label">비밀번호</label>
            <input type="password" name="password" id="password" class="form-control">
        </div>
        <button type="submit" class="btn btn-primary">로그인</button>
    </form>
</div>
</html>
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 템플릿 작성하기

1. 사용자 ID와 비밀번호로 로그인할 수 있는 템플릿을 작성함.
   스프링 시큐리티의 로그인이 실패할 경우에는
   시큐리티의 기능으로 인해 로그인 페이지로 리다이렉트됨.

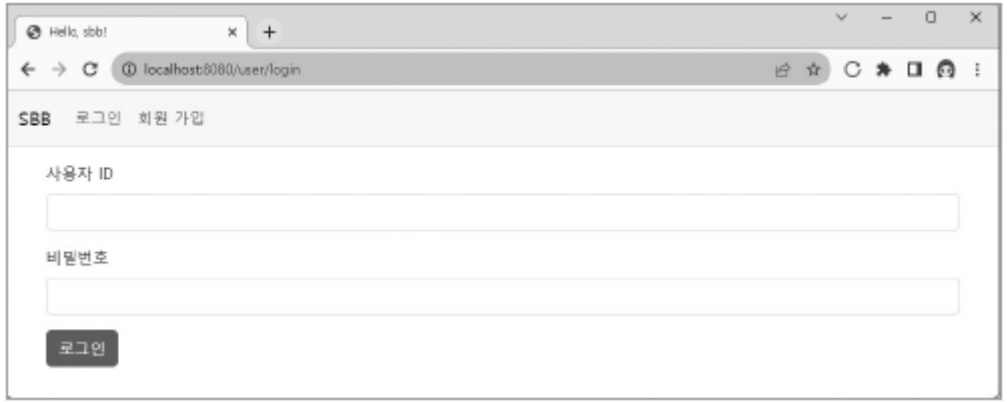
   이때 페이지 매개변수로 error가 함께 전달됨.
   따라서 로그인 페이지의 매개변수로 error가 전달될 경우
   '사용자 ID 또는 비밀번호를 확인해 주세요.'라는 오류 메시지를 출력하도록 함

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 템플릿 작성하기

2. 여기까지 수정하고 브라우저에서 http://localhost:8080/user/login을 호출해 보자.
   그러면 다음과 같은 화면이 나타남



3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 템플릿 작성하기

2. 하지만 아직 로그인을 수행할 수는 없음.
   왜냐하면 스프링 시큐리티에 무엇을 기준으로 로그인해야 하는지
   아직 설정하지 않았기 때문임.

   스프링 시큐리티를 통해 로그인을 수행하는 방법에는 여러 가지가 있음.
   가장 간단한 방법으로 SecurityConfig.java와 같은 시큐리티 설정 파일에
   사용자 ID와 비밀번호를 직접 등록하여 인증을 처리하는 메모리 방식이 있음.

   하지만 3-06절에서 회원 가입을 통해 회원 정보를 DB에 저장했으므로
   DB에서 회원 정보를 조회하여 로그인하는 방법을 사용할 것임

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 템플릿 작성하기

2. 이제 DB에서 사용자를 조회하는 서비스(UserSecurityService.java)를 만들고, 그 서비스를 스프링 시큐리티에 등록하는 방법을 알아보자.

   하지만 UserSecurityService 서비스를 만들기 전에 UserRepository를 수정하고 UserRole 클래스를 생성하는 등 준비를 해야 함.

   서비스를 활용하기 위한 밑 작업을 진행해 보자.

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ User 리포지터리 수정하기

- 뒤에서 생성할 UserSecurityService는 사용자 ID를 조회하는 기능이 필요하므로 다음과 같이 사용자 ID로 SiteUser 엔티티를 조회하는 findByUsername 메서드를 User 리포지터리에 추가하자

• /user/UserRepository.java

```
package com.mysite.sbb.user;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<SiteUser, Long> {
    Optional<SiteUser> findByUsername(String username);
}
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserRole 파일 생성하기

- 스프링 시큐리티는 인증뿐만 아니라 권한도 관리함
- 스프링 시큐리티는 사용자 인증 후에 사용자에게 부여할 권한과 관련된 내용이 필요함
- 그러므로 사용자가 로그인한 후, ADMIN 또는 USER와 같은 권한을 부여해야 함
- 다음과 같이 com.mysite.sbb.user 패키지에 UserRole.java 파일을 만들어 보자

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserRole 파일 생성하기

• /user/UserRole.java

```
package com.mysite.sbb.user;

import lombok.Getter;

@Getter
public enum UserRole {
    ADMIN("ROLE_ADMIN"),
    USER("ROLE_USER");

    UserRole(String value) {
        this.value = value;
    }

    private String value;
}
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserRole 파일 생성하기

- UserRole은 enum 자료형(열거 자료형)으로 작성함
- 관리자를 의미하는 ADMIN과 사용자를 의미하는 USER라는 상수를 만듦
- ADMIN은 'ROLE_ADMIN', USER는 'ROLE_USER'라는 값을 부여함
- UserRole의 ADMIN과 USER 상수는 값을 변경할 필요가 없으므로 @Setter 없이 @Getter만 사용할 수 있도록 함

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserSecurityService 서비스 생성하기

- com.mysite.sbb.user 패키지에 스프링 시큐리티가 로그인 시 사용할 서비스인 UserSecurityService.java를 다음과 같이 작성해 보자

• /user/UserSecurityService.java

```
package com.mysite.sbb.user;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import lombok.RequiredArgsConstructor;
@RequiredArgsConstructor
@Service
public class UserSecurityService implements UserDetailsService {

    private final UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        Optional<SiteUser> _siteUser = this.userRepository.findByUsername(use
rname);
        if (_siteUser.isEmpty()) {
            throw new UsernameNotFoundException("사용자를 찾을 수 없습니다.");
        }
        SiteUser siteUser = _siteUser.get();
        List<GrantedAuthority> authorities = new ArrayList<>();
        if ("admin".equals(username)) {
            authorities.add(new SimpleGrantedAuthority(UserRole.ADMIN.getValue()));
        } else {
            authorities.add(new SimpleGrantedAuthority(UserRole.USER.getValue()));
        }
        return new User(siteUser.getUsername(), siteUser.getPassword(), authorities);
    }
}
```

사용자의 권한 정보를 나타내는 GrantedAuthority 객체를 생성하는 데 사용할 리스트를 생성한다.

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserSecurityService 서비스 생성하기

- 스프링 시큐리티가 로그인 시 사용할 UserSecurityService는
  스프링 시큐리티가 제공하는 UserDetailsService 인터페이스를 구현(implements)해야 함

- 스프링 시큐리티의 UserDetailsService는 loadUserByUsername 메서드를
  구현하도록 강제하는 인터페이스임

- loadUserByUsername 메서드는 사용자명(username)으로 스프링 시큐리티의
  사용자(User) 객체를 조회하여 리턴하는 메서드임

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ UserSecurityService 서비스 생성하기

- loadUserByUsername 메서드는 사용자명으로 SiteUser 객체를 조회하고, 만약 사용자명에 해당하는 데이터가 없을 경우에는 UsernameNotFoundException을 발생시킴

- 사용자명이 'admin'인 경우에는 ADMIN 권한(ROLE_ADMIN)을 부여하고 그 이외의 경우에는 USER 권한(ROLE_USER)을 부여함

- 마지막으로 User 객체를 생성해 반환하는데, 이 객체는 스프링 시큐리티에서 사용하며 User 생성자에는 사용자명, 비밀번호, 권한 리스트가 전달됨

- 스프링 시큐리티는 loadUserByUsername 메서드에 의해 리턴된 User 객체의 비밀번호가 사용자로부터 입력받은 비밀번호와 일치하는지를 검사하는 기능을 내부에 가지고 있음

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 스프링 시큐리트 설정 수정하기

- 로그인 기능을 완성하기 위해 SecurityConfig.java 파일을 열어 다음과 같이 AuthenticationManager 빈을 생성하자

• SecurityConfig.java

```
(... 생략 ...)
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.configura
tion.AuthenticationConfiguration;

@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    (... 생략 ...)
    }
    @Bean
    PasswordEncoder passwordEncoder(){
    (... 생략 ...)
    }

    @Bean
    AuthenticationManager authenticationManager(AuthenticationConfiguration
authenticationConfiguration) throws Exception {
        return authenticationConfiguration.getAuthenticationManager();
    }
}
```

3-07 로그인과 로그아웃 기능 구현하기

- **로그인 기능 구현하기**
  - **스프링 시큐리트 설정 수정하기**
    - 이와 같이 AuthenticationManager 빈을 생성함
    - AuthenticationManager는 스프링 시큐리티의 인증을 처리함
    - AuthenticationManager는 사용자 인증 시 앞에서 작성한 UserSecurity Service와 PasswordEncoder를 내부적으로 사용하여 인증과 권한 부여 프로세스를 처리함

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

1. 로그인 기능을 구현하는 마지막 단계로 로그인 페이지에 곧바로 진입할 수 있도록 로그인 링크(/user/login)를 내비게이션 바에 추가해 보자.
templates의 navbar.html을 다음과 같이 수정하면 됨

• /templates/navbar.html

```
<nav th:fragment="navbarFragment" class="navbar navbar-expand-lg navbar-light bg-
light border-bottom">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">SBB</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false"
        aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/user/login}">로그인</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/user/signup}">회원 가입</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

2. 로그인 기능을 구현하는 모든 작업을 마쳤으니 로컬 서버를 재시작한 후, 브라우저를 통해 http://local:8080에 접속하고 내비게이션 바의 '로그인' 링크를 클릭해 보자.
   그러면 다음과 같은 로그인 화면이 등장함

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

3. 만약 DB에 없는 사용자 ID 또는 잘못된 비밀번호를 입력하면 다음과 같이 오류 메시지가 나타남.

이미 가입되어 있는 사용자 ID와 비밀번호를 입력하면
로그인이 정상 수행되고 메인 화면인 질문 목록 페이지로 이동할 것임

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

4. 하지만 로그인한 후에도 내비게이션 바에는 여전히 '로그인'이란 이름으로 링크가 표시됨. 일반적으로 로그인한 상태라면 이 링크는 로그아웃을 위해 '로그아웃' 링크로 바뀌어야 함 (반대로 로그아웃 상태에서는 '로그인' 링크로 바뀌어야 한다).

   그러기 위해 다음과 같은 스프링 시큐리티의 타임리프 확장 기능을 사용하여 사용자의 로그인 상태를 확인해야 함

   - sec:authorize="isAnonymous()": 로그인되지 않은 경우에 해당 요소(로그인 링크)가 표시된다.
   - sec:authorize="isAuthenticated()": 로그인된 경우에 해당 요소(로그아웃 링크)가 표시된다.

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

4. 여기서 sec:authorize 속성은 사용자의 로그인 여부에 따라
요소를 출력하거나 출력하지 않게 함.
이 내용을 활용하여 앞에서 로그인 링크를 수정한 부분을 다시 다음과 같이 수정해 보자

• /templates/navbar.html

```
<nav th:fragment="navbarFragment" class="navbar navbar-expand-lg navbar-light
bg-light border-bottom">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">SBB</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false"
        aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link" sec:authorize="isAnonymous()"
                    th:href="@{/user/login}">로그인</a>
                    <a class="nav-link" sec:authorize="isAuthenticated()"
                    th:href="@{/user/logout}">로그아웃</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/user/signup}">회원 가입</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

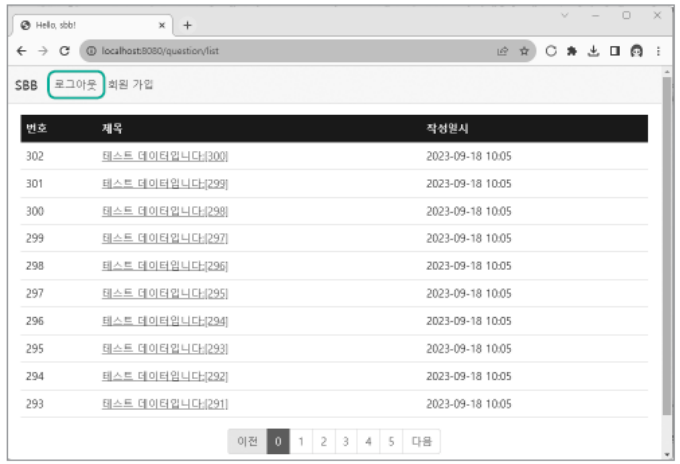
3-07 로그인과 로그아웃 기능 구현하기

▪ 로그인 기능 구현하기

▪ 로그인 화면 수정하기

4. 로그인하지 않은 상태라면 sec:authorize="isAnonymous( )"가 '참'이 되어 '로그인' 링크가 표시되고, 로그인한 상태라면 sec:authorize="isAuthenticated( )"가 '참'이 되어 '로그아웃' 링크가 표시될 것임

4. 다시 브라우저에 접속하여 로그인을 수행하면 '로그아웃' 링크가 표시되는 것을 확인할 수 있음



3-07 로그인과 로그아웃 기능 구현하기

▪ 로그아웃 기능 구현하기

1. navbar.html에서 우리는 '로그아웃' 링크를 /user/logout으로 설정함.
   하지만 아직 로그아웃 기능은 구현하지 않은 상태임.

   앞선 실습에서 SBB에 로그인을 했다면
   내비게이션바에 '로그아웃' 링크가 나타난 것을 확인할 수 있음.
   그런데 '로그아웃' 링크를 누르면 다음과 같이 404 오류 페이지가 표시됨.

   아직 로그아웃 기능을 구현하지 않아서 이와 같이 오류 페이지가 표시됨.
   로그아웃 역시 스프링 시큐리티를 사용하여 쉽게 구현할 수 있음

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그아웃 기능 구현하기

2. 로그아웃 설정을 추가하기 위해 다음과 같이 SecurityConfig 파일을 수정해 보자

• SecurityConfig.java

```
(... 생략 ...)

@Configuration
@EnableWebSecurity
public class SecurityConfig  {

    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((authorizeHttpRequests) -> authorizeHttp
Requests
                .requestMatchers(new AntPathRequestMatcher("/**")).permitAll())
            .csrf((csrf) -> csrf
                .ignoringRequestMatchers(new AntPathRequestMatcher("/h2-
console/**")))
            .headers((headers) -> headers
                .addHeaderWriter(new XFrameOptionsHeaderWriter(
                    XFrameOptionsHeaderWriter.XFrameOptionsMode.SAMEORIGIN)))
            .formLogin((formLogin) -> formLogin
                .loginPage("/user/login")
                .defaultSuccessUrl("/"))
            .logout((logout) -> logout
                .logoutRequestMatcher(new AntPathRequestMatcher("/user/logout"))
                .logoutSuccessUrl("/")
                .invalidateHttpSession(true))
            ;
        return http.build();
    }

    (... 생략 ...)
}
```

3-07 로그인과 로그아웃 기능 구현하기

▪ 로그아웃 기능 구현하기

2. 로그아웃 기능을 구현하기 위한 설정을 추가함.
   로그아웃 URL을 /user/logout으로 설정하고 로그아웃이 성공하면
   루트(/) 페이지로 이동하도록 함.

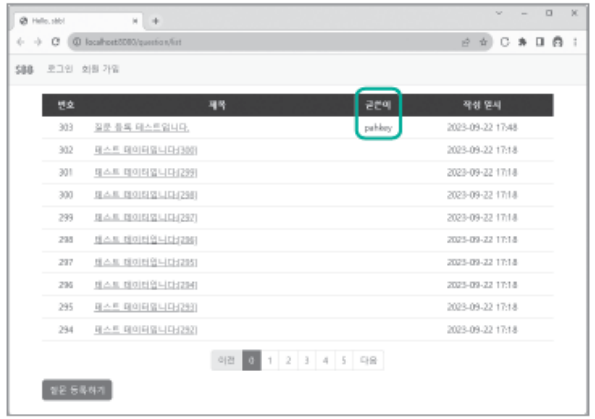
   그리고 .invalidateHttpSession(true)를 통해
   로그아웃 시 생성된 사용자 세션도 삭제하도록 처리함

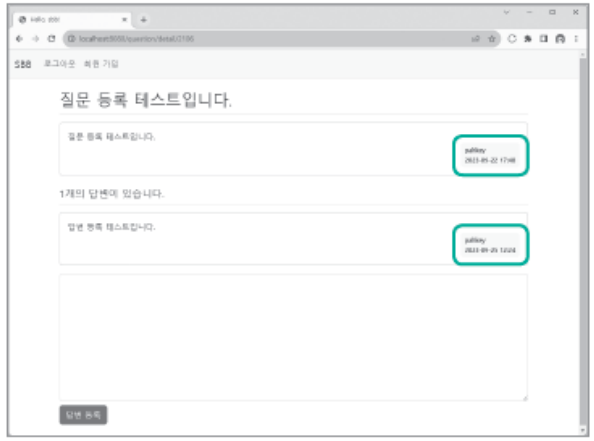
2. 수정을 완료한 후 다시 http://localhost:8080에서 로그인한 후,
   '로그아웃' 링크를 클릭하여 다시 '로그인' 링크가 등장하는지 확인해 보자

3-08 글쓴이 항목 추가하기

- 질문 또는 답변을 작성할 때 사용자 정보도 DB에 함께 저장해 보자.
  이때 질문 또는 답변을 작성한 사용자는 반드시 로그인되어 있어야 함
- 그래야 누가 작성한 글인지 알 수 있고, 수정 및 삭제도 가능하기 때문

- 게시판의 질문 목록과 답변 상세 페이지에는
  누가 글을 작성했는지 알려 주는 '글쓴이' 항목도 추가해 보자





3-08 글쓴이 항목 추가하기

▪ 엔티티에 속성 추가하기

먼저 기존에 만든 Question(질문)과 Answer(답변) 엔티티에
글쓴이에 해당하는 author 속성을 추가해 보자

▪ 질문 엔티티에 속성 추가하기

질문 테이블에 글쓴이를 저장하려면 먼저 Question 엔티티에 author 속성을 추가해야 함.
다음과 같이 Question.java를 수정해 보자

3-08 글쓴이 항목 추가하기

▪ 엔티티에 속성 추가하기

▪ 질문 엔티티에 속성 추가하기

• /question/Question.java

```
(... 생략 ...)
import com.mysite.sbb.answer.Answer;
import com.mysite.sbb.user.SiteUser;

(... 생략 ...)
import jakarta.persistence.OneToMany;
import jakarta.persistence.ManyToone;

public class Question {
    (... 생략 ...)

    @ManyToOne
    private SiteUser author;
}
```

author 속성에는 @ManyToOne 애너테이션을 적용했는데,
이는 사용자 한 명이 질문을 여러 개 작성할 수 있기 때문임

3-08 글쓴이 항목 추가하기

▪ 엔티티에 속성 추가하기

▪ 답변 엔티티에 속성 추가하기

Question 엔티티와 같은 방법으로 Answer 엔티티에도 author 속성을 추가해 보자

• /answer/Answer.java

```
(... 생략 ...)
import com.mysite.sbb.question.Question;
import com.mysite.sbb.user.SiteUser;
(... 생략 ...)
public class Answer {
    (... 생략 ...)

    @ManyToOne
    private SiteUser author;
}
```

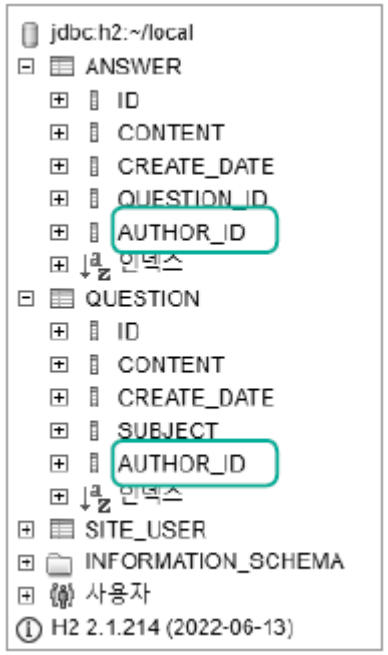
3-08 글쓴이 항목 추가하기

▪ 엔티티에 속성 추가하기

▪ 테이블 확인하기

앞선 실습에서 Question, Answer 엔티티를 변경했으므로,
H2 콘솔에 접속하여 Question, Answer 테이블을 확인해 보자.

question, answer 테이블에 author_id 열이 생성된 것을 확인할 수 있음
이 열에는 글쓴이의 ID 값이 저장됨



3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

- 이제 Question, Answer 엔티티에 author 속성이 추가되었으므로
  질문과 답변 데이터를 저장할 때 author(글쓴이)도 함께 저장할 수 있음

- 새로운 데이터를 저장하려면 서버와 DB를 관리하는 컨트롤러와
  서비스(또는 리포지터리)에도 관련 내용을 업데이트해야 함

- 기존 파일들을 수정하면서 이번에는 데이터를 어떻게 저장하는지 알아보자

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

1. 답변을 저장할 때, 사용자 정보도 저장할 수 있도록 먼저 AnswerController를 수정해 보자

• /answer/AnswerController.java

```
(... 생략 ...)

import java.security.Principal;

(... 생략 ...)
public class AnswerController {

    @PostMapping("/create/{id}")
    public String createAnswer(Model model, @PathVariable("id") Integer id, @Valid
AnswerForm answerForm, BindingResult bindingResult, Principal principal) {
        (... 생략 ...)
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

1. 현재 로그인한 사용자의 정보를 알려면
   스프링 시큐리티가 제공하는 Principal 객체를 사용해야 함.

   여기서는 일단 이와 같이 createAnswer 메서드에 Principal 객체를
   매개변수로 지정하는 작업까지만 해두고 답변 서비스를 수정해 보자

3-08 글쓴이 항목 추가하기

- **글쓴이 저장하기**
  - **답변 컨트롤러와 서비스 업데이트하기**

2. principal 객체를 사용하면 이제 로그인한 사용자명을 알 수 있으므로 사용자명으로 SiteUser 객체를 조회할 수 있음.

   먼저 SiteUser를 조회할 수 있는 getUser 메서드를 UserService에 추가하자

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

2.

• UserService.java

```
package com.mysite.sbb.user;

import java.util.Optional;

import com.mysite.sbb.DataNotFoundException;

(... 생략 ...)

@Service
public class UserService {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    public SiteUser create(String username, String email, String password) {
    (... 생략 ...)
    }

    public SiteUser getUser(String username) {
        Optional<SiteUser> siteUser = this.userRepository.findByUsername(user
name);
        if (siteUser.isPresent()) {
            return siteUser.get();
        } else {
            throw new DataNotFoundException("siteuser not found");
        }
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

2. getUser 메서드는 userRepository의 findByUsername 메서드를 사용하여 쉽게 만들 수 있음.

   사용자명에 해당하는 데이터가 없을 경우에는 DataNoFoundException이 발생하도록 함

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

3. 답변 내용을 저장할 때 글쓴이 데이터도 저장할 수 있도록
다음과 같이 AnswerService를 수정해 보자

• /answer/AnswerService.java

```
package com.mysite.sbb.answer;

import java.time.LocalDateTime;

import com.mysite.sbb.question.Question;
import com.mysite.sbb.user.SiteUser;

import org.springframework.stereotype.Service;

(... 생략 ...)
@Service
public class AnswerService {
    private final AnswerRepository answerRepository;

    public void create(Question question, String content, SiteUser author) {
        Answer answer = new Answer();
        answer.setContent(content);
        answer.setCreateDate(LocalDateTime.now());
        answer.setQuestion(question);
        answer.setAuthor(author);
        this.answerRepository.save(answer);
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 답변 컨트롤러와 서비스 업데이트하기

4. 다시 AnswerController.java로 돌아가 다음과 같이 수정하여 createAnswer 메서드를 완성해 보자

• /answer/AnswerController.java

```
(... 생략 ...)
import com.mysite.sbb.user.SiteUser;
import com.mysite.sbb.user.UserService;
(... 생략 ...)
public class AnswerController {

    private final QuestionService questionService;
    private final AnswerService answerService;
    private final UserService userService;

    @PostMapping("/create/{id}")
    public String createAnswer(Model model, @PathVariable("id") Integer id,
            @Valid AnswerForm answerForm, BindingResult bindingResult, Principal
principal) {
        Question question = this.questionService.getQuestion(id);
        SiteUser siteUser = this.userService.getUser(principal.getName());
        if (bindingResult.hasErrors()) {
            model.addAttribute("question", question);
            return "question_detail";
        }
        this.answerService.create(question, answerForm.getContent(), siteUser);
        return String.format("redirect:/question/detail/%s", id);
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 질문 컨트롤러와 서비스 업데이트하기

1. 먼저 글쓴이 데이터를 저장하기 위해 QuestionService를 다음과 같이 수정해 보자

• /question/QuestionService.java

```
(... 생략 ...)
import com.mysite.sbb.DataNotFoundException;
import com.mysite.sbb.user.SiteUser;
(... 생략 ...)
public class QuestionService {

    (... 생략 ...)

    public void  create(String subject, String content, SiteUser user) {
        Question q = new Question();
        q.setSubject(subject);
        q.setContent(content);
        q.setCreateDate(LocalDateTime.now());
        q.setAuthor(user);
        this.questionRepository.save(q);
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 질문 컨트롤러와 서비스 업데이트하기

2. 이어서 QuestionController도 다음과 같이 수정해 보자

• /question/QuestionController.java

```
(... 생략 ...)
import java.security.Principal;
import com.mysite.sbb.user.SiteUser;
import com.mysite.sbb.user.UserService;
(... 생략 ...)

public class QuestionController {

    private final QuestionService questionService;
    private final UserService userService;

    (... 생략 ...)

    @PostMapping("/create")
    public String questionCreate(@Valid QuestionForm questionForm,
            BindingResult bindingResult, Principal principal) {
        if (bindingResult.hasErrors()) {
            return "question_form";
        }
        SiteUser siteUser = this.userService.getUser(principal.getName());
        this.questionService.create(questionForm.getSubject(), questionForm.
getContent(), siteUser);
        return "redirect:/question/list";
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 글쓴이 저장하기

▪ 질문 컨트롤러와 서비스 업데이트하기

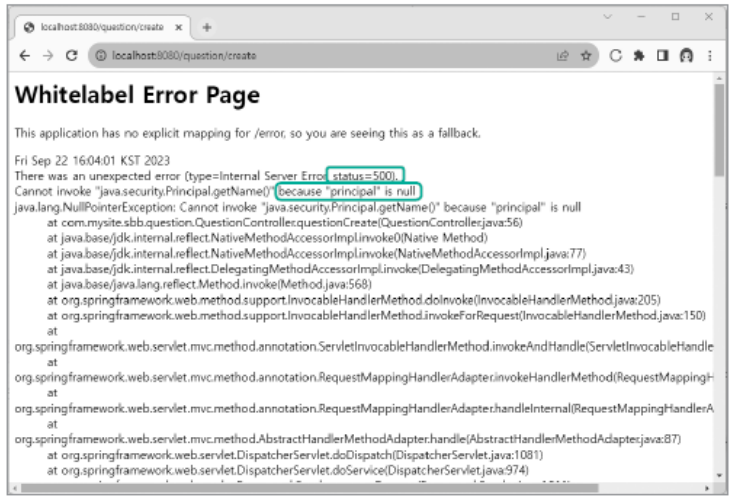
3. 다시 로컬 서버를 시작하고 로그인한 다음, 질문과 답변 등록을 테스트해 보자

3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

1. 로그아웃 상태에서 질문 또는 답변을 등록해 보자.
그럼 다음과 같은 500 오류(서버 오류)가 발생함.

   이는 principal 객체가 널(null)이라서 발생한 오류임.
principal 객체는 로그인을 해야만 생성되는 객체인데
현재는 로그아웃 상태이므로 principal 객체에
값이 없어 오류가 발생하는 것임



3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

2. 이 문제를 해결하려면 principal 객체를 사용하는 메서드에 @PreAuthorize("isAuthenti cated( )") 애너테이션을 사용해야 함.

   @PreAuthorize("isAuthenticated( )") 애너테이션이 붙은 메서드는 로그인한 경우에만 실행됨.
   즉, 이 애너테이션을 메서드에 붙이면 해당 메서드는 로그인한 사용자만 호출할 수 있음.

   @PreAuthorize("isAuthenticated( )") 애너테이션이 적용된 메서드가 로그아웃 상태에서 호출되면 로그인 페이지로 강제 이동됨.

   먼저, QuestionController부터 다음과 같이 수정해 보자

3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

2. • /question/QuestionController.java

```
package com.mysite.sbb.question;

(... 생략 ...)
import org.springframework.data.domain.Page;
import org.springframework.security.access.prepost.PreAuthorize;
(... 생략 ...)
public class QuestionController {
    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @GetMapping("/create")
    public String questionCreate(QuestionForm questionForm) {
        return "question_form";
    }
    @PreAuthorize("isAuthenticated()")
    @PostMapping("/create")
    public String questionCreate(@Valid QuestionForm questionForm,
            BindingResult bindingResult, Principal principal) {
        (... 생략 ...)
    }
}
```

로그인이 필요한 메서드(질문 등록과 관련된 메서드)들에
@PreAuthorize("isAuthenticated( )") 애너테이션을 적용함

3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

3. 마찬가지로 AnswerController도 다음과 같이 수정하자

• /answer/AnswerController.java

```
(... 생략 ...)
import org.springframework.validation.BindingResult;
import org.springframework.security.access.prepost.PreAuthorize;
(... 생략 ...)

public class AnswerController {

    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @PostMapping("/create/{id}")
    public String createAnswer(Model model, @PathVariable("id") Integer id, @Valid
AnswerForm answerForm,
            BindingResult bindingResult, Principal principal) {
        (... 생략 ...)
    }
}
```

3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

4. 마지막으로 @PreAuthorize 애너테이션이 동작할 수 있도록 스프링 시큐리티의 설정도 수정해야 함.
SecurityConfig를 다음과 같이 수정해 보자

• SecurityConfig.java

```
package com.mysite.sbb;

(... 생략 ...)
import org.springframework.security.config.annotation.authentication.configura
tion.AuthenticationConfiguration;
import org.springframework.security.config.annotation.method.configuration.
EnableMethodSecurity;

@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
    (... 생략 ...)
}
```

3-08 글쓴이 항목 추가하기

▪ 로그인 페이지로 이동시키기

4. SecurityConfig에 적용한@EnableMethodSecurity 애너테이션의 prePostEnabled = true는 QuestionController와 AnswerController에서 로그인 여부를 판별할 때 사용한 @PreAuthorize 애너테이션을 사용하기 위해 반드시 필요한 설정임

4. 이렇게 수정한 후 로그아웃 상태에서 질문 또는 답변을 등록하면 자동으로 로그인 화면으로 이동하는 것을 확인할 수 있을 것임

3-08 글쓴이 항목 추가하기

- **답변 작성 막아 두기**
  - 현재 질문 등록 페이지에서는 사용자가 로그아웃 상태라면 아예 글을 작성할 수 없음
  - 하지만 답변 등록 페이지에서는 로그아웃 상태에서도 글은 작성할 수 있어서 답변을 작성한 후 [답변 등록] 버튼을 눌러야만 로그인 화면으로 이동됨

  - 이렇게 되면 애써 사용자가 작성한 답변이 사라지는 문제가 있음
  - 이 문제를 해결하려면 사용자가 로그아웃 상태인 경우 아예 답변 작성을 못하게 막는 것이 좋은 방법임

3-08 글쓴이 항목 추가하기

▪ 답변 작성 막아 두기

1. 로그아웃 상태에서 답변을 작성하지 못하도록 question_detail.html 파일을 다음과 같이 수정해 보자

• /templates/question_detail.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
   (... 생략 ...)
   <!-- 답변 작성 -->
   <form th:action="@{|/answer/create/${question.id}|}" th:object="${answerForm}"
method="post" class="my-3">
      <div th:replace="~{form_errors :: formErrorsFragment}"></div>
      <textarea sec:authorize="isAnonymous()" disabled
      th:field="*{content}" class="form-control" rows="10"></textarea>
      <textarea sec:authorize="isAuthenticated()"
      th:field="*{content}" class="form-control" rows="10"></textarea>
      <input type="submit" value="답변 등록" class="btn btn-primary my-2">
   </form>
</div>
</html>
```

3-08 글쓴이 항목 추가하기

▪ 답변 작성 막아 두기

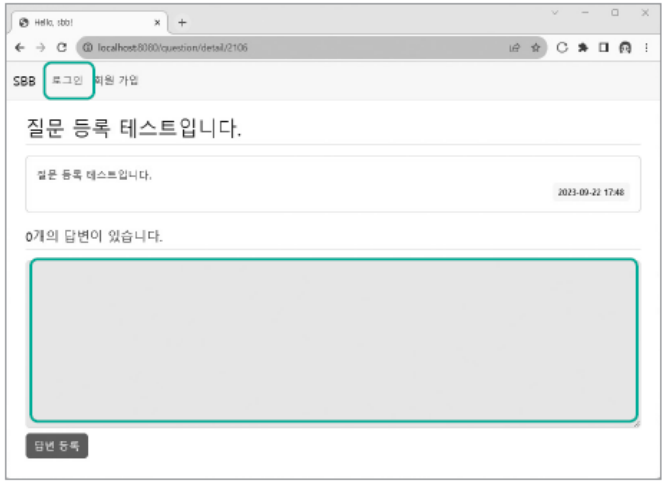
1. 로그인 상태가 아닌 경우 textarea 태그에 disabled 속성을 적용하여
   사용자가 화면에서 아예 입력하지 못하게 만들었음.

   여기서 sec:authorize="isAnonymous( )", sec:authorize="isAuthenticated( )"는
   현재 사용자의 로그인 상태를 체크하는 속성으로,
   sec:authorize="isAnonymous( )"는 현재 로그아웃 상태임을 의미하고,
   sec:authorize="isAuthenticated( )"는 현재 로그인 상태임을 의미함

3-08 글쓴이 항목 추가하기

▪ 답변 작성 막아 두기

2. 다음은 로그아웃 상태에서 disabled가 적용된 화면임.
이와 같이 로그아웃 상태에서는 사용자가 답변을 등록할 수 없도록
답변 등록 칸이 회색으로 표시되고,
키보드를 눌러도 아무런 내용이 입력되지 않음



3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

이제 질문 목록과 질문 상세 화면에 글쓴이를 표시해 보자.
앞서 Question 엔티티와 Answer 엔티티에 auther 속성을 추가함.
이를 이용하여 질문 목록, 질문 상세 화면에 글쓴이를 표시해 보자

▪ 질문 목록에 글쓴이 표시하기

1. 질문 목록 템플릿인 question_html에 글쓴이를 추가해 보자.
   그 전에 다음과 같이 테이블 헤더를 수정해 보자

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 목록에 글쓴이 표시하기

1.

• /templates/question_list.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <table class="table">
        <thead class="table-dark">
            <tr class="text-center">
                <th>번호</th>
                <th style="width:50%">제목</th>
                <th>글쓴이</th>
                <th>작성 일시</th>
            </tr>
        </thead>
```

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 목록에 글쓴이 표시하기

1. <th>글쓴이</th>를 추가함.

   그리고 각 th 요소들(번호, 제목, 글쓴이, 작성 일시)을 가운데 정렬하도록
   tr 태그에 text-center 클래스를 추가하고,

   <th>제목</th>에서는 너비가 전체에서 50%를 차지하도록
   style="width:50%"로 작성하여 너비를 지정함

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 목록에 글쓴이 표시하기

2. '글쓴이'가 화면에 보이도록 틀을 마련했으니
   글쓴이가 표시되도록 이어서 for 문에도 다음과 같이 추가해 보자

• /templates/question_list.html

```
(... 생략 ...)
        <tbody>
            <tr class="text-center" th:each="question, loop : ${paging}">
                <td th:text="${paging.getTotalElements - (paging.number * paging.
size) - loop.index}"></td>
                <td class="text-start">
                    <a th:href="@{|/question/detail/${question.id}|}"
                    th:text="${question.subject}"></a>
                    <span class="text-danger small ms-2"
                        th:if="${#lists.size(question.answerList) > 0}"
                        th:text="${#lists.size(question.answerList)}">
                    </span>
                </td>
                <td><span th:if="${question.author != null}"
                th:text="${question.author.username}"></span></td>
                <td th:text="${#temporals.format(question.createDate, 'yyyy-MM-dd
HH:mm')}"></td>
            </tr>
        </tbody>
(... 생략 ...)
```

글쓴이를 표시하기 위해 작성한다.

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 목록에 글쓴이 표시하기

2. <td> ... </td> 요소를 삽입하여 질문의 글쓴이를 표시함.

   글쓴이 정보 없이 저장된 기존의 질문들은
   author 속성에 해당하는 데이터가 없으므로
   (author 속성의 값으로 null을 가지고 있으므로)
   author 속성의 값이 null이 아닌 경우만 글쓴이를 표시하도록 함.

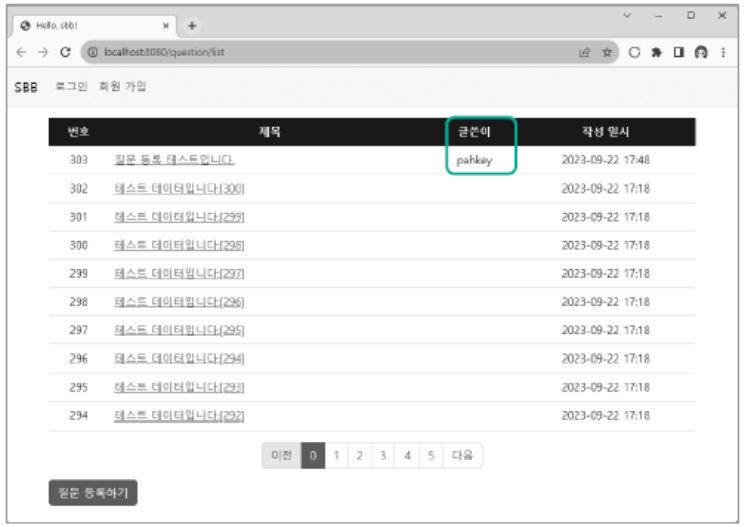
   그리고 여기서도 표시되는 항목을 모두 가운데 정렬하도록
   tr 요소에 text-center 클래스를 추가하고,
   제목 항목의 값들만 왼쪽 정렬하도록 text-start 클래스를 추가함

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 목록에 글쓴이 표시하기

3. 다시 질문 목록 화면으로 돌아가면 글쓴이 항목이 추가된 것을 확인할 수가 있음



3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 상세에 글쓴이 표시하기

1. 질문 상세 템플릿인 question_detail.html을 수정하여 질문 상세 화면에서도 글쓴이 항목이 노출되도록 만들어 보자

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 질문 -->
<h2 class="border-bottom py-2" th:text="${question.subject}"></h2>
<div class="card my-3">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;" th:text="${question.
content}"></div>
        <div class="d-flex justify-content-end">
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${question.author != null}"
                    th:text="${question.author.username}"></span>
                </div>
                <div th:text="${#temporals.format(question.createDate, 'yyyy-MM-
dd HH:mm')}"></div>
            </div>
        </div>
    </div>
</div>
(... 생략 ...)
```

글쓴이를 표시하기 위해 작성한다.

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 상세에 글쓴이 표시하기

1. 그다음 답변 부분에도 글쓴이 항목이 노출되도록 다음과 같이 내용을 추가하자

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 답변 반복 시작 -->
<div class="card my-3" th:each="answer : ${question.answerList}">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;"
        th:text="${answer.content}"></div>
        <div class="d-flex justify-content-end">
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${answer.author != null}"
                    th:text="${answer.author.username}"></span>
                </div>
                <div th:text="${#temporals.format(answer.createDate, 'yyyy-MM-dd
HH:mm')}"></div>
            </div>
        </div>
    </div>
</div>
<!-- 답변 반복 끝  -->
(... 생략 ...)
```

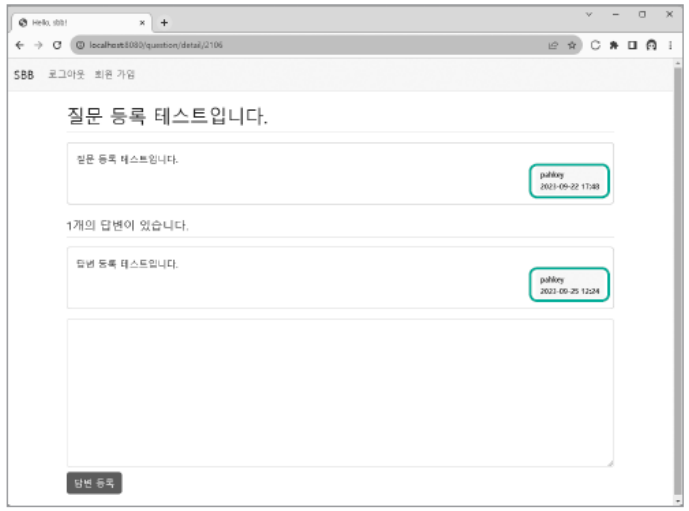
글쓴이를 표시하기 위해 작성한다.

3-08 글쓴이 항목 추가하기

▪ 화면에 글쓴이 나타내기

▪ 질문 상세에 글쓴이 표시하기

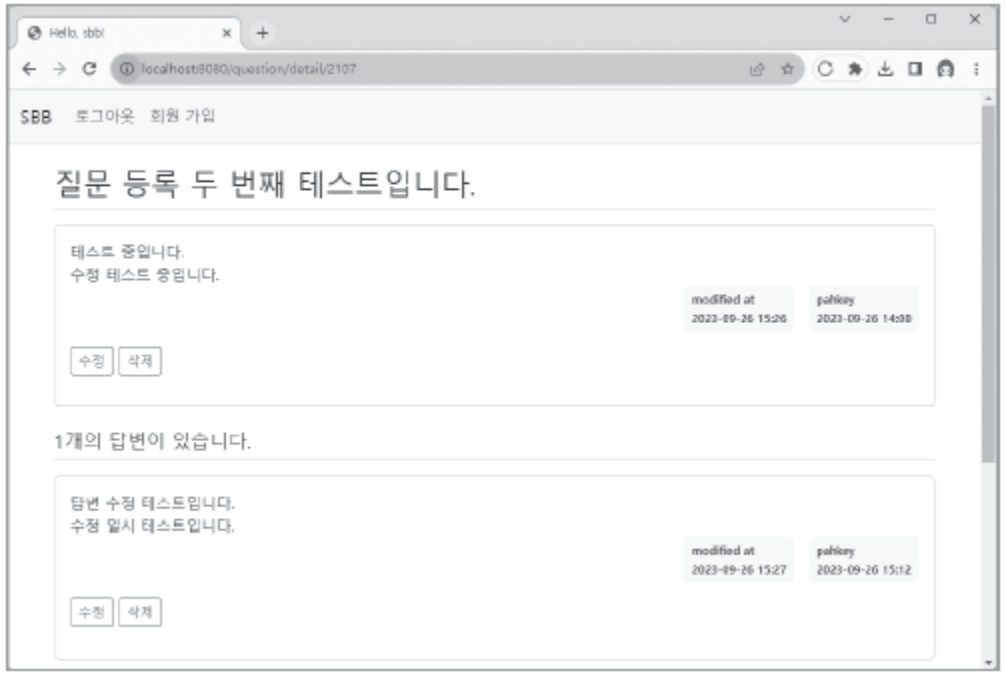
3. 로컬 서버를 재시작하여 오른쪽과 같이 질문 상세 화면에서 답변을 입력한 후, 답변의 글쓴이가 노출되는지 확인해 보자.



이와 같이 질문을 등록하는 글쓴이와 답변을 등록한 글쓴이 모두 작성 일시와 함께 노출된 것을 확인할 수 있음

3-09 수정과 삭제 기능 추가하기

- 질문 또는 답변을 작성한 후 이 글들을 수정하거나 삭제할 수 있어야 함
- 이번에는 앞서 작성한 질문 또는 답변을 수정하거나 삭제하는 기능을 추가해 보자



3-09 수정과 삭제 기능 추가하기

▪ 수정 일시 추가하기

SBB에 질문 또는 답변을 수정하거나 삭제하는 기능을 추가하기 전에
질문이나 답변이 언제 수정되었는지 확인할 수 있도록
Question 엔티티와 Answer 엔티티에 수정 일시를 의미하는 modifyDate 속성을 추가해 보자

1. Question.java와 Answer.java에 각각 다음과 같이 한 줄의 코드를 추가하면 간단히 해결됨

• /question/Question.java

```
(... 생략 ...)
public class Question {
    (... 생략 ...)

    @ManyToOne
    private SiteUser author;

    private LocalDateTime modifyDate;
}
```

• /answer/Answer.java

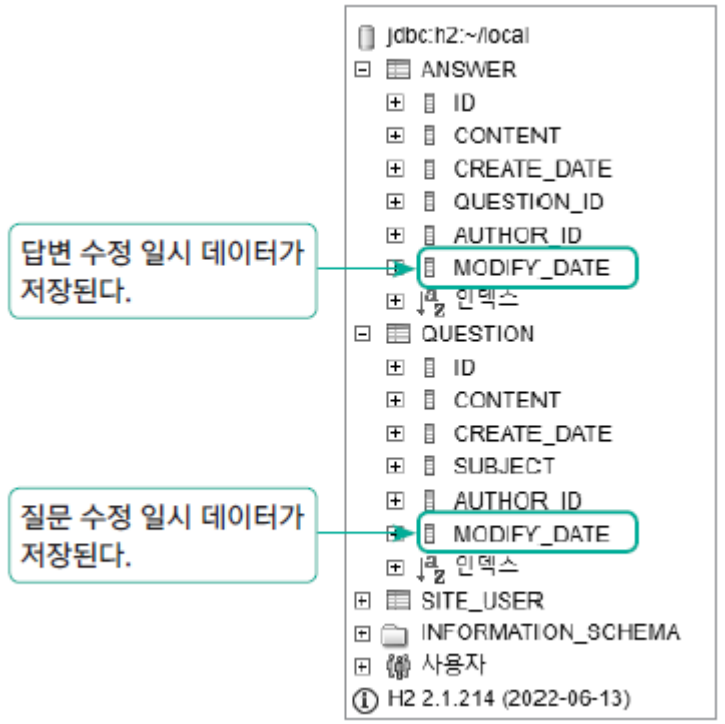
```
(... 생략 ...)
public class Answer {
    (... 생략 ...)

    @ManyToOne
    private SiteUser author;

    private LocalDateTime modifyDate;
}
```

3-09 수정과 삭제 기능 추가하기

▪ 수정 일시 추가하기

2. 이와 같이 수정한 뒤, 다시 H2 콘솔에 접속해 보

   다음과 같이 Answer와 Question 테이블에 각각 modify_date 열이 추가된 것을 확인할 수 있음



3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 수정 버튼 만들기

- 사용자가 질문 상세 화면에서 [수정] 버튼을 클릭하면
  수정할 수 있는 화면으로 진입할 수 있도록
  다음과 같이 질문 상세 화면에 질문 수정 버튼을 추가해 보자

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 수정 버튼 만들기

1.

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 질문 -->
<h2 class="border-bottom py-2" th:text="${question.subject}"></h2>
<div class="card my-3">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;" th:text="${question.
content}"></div>
        <div class="d-flex justify-content-end">
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${question.author != null}" th:text="${question.
author.username}"></span>
                </div>
                <div th:text="${#temporals.format(question.createDate, 'yyyy-MM-
dd HH:mm')}"></div>
            </div>
        </div>
        <div class="my-3">
            <a th:href="@{|/question/modify/${question.id}|}" class="btn btn-sm
btn-outline-secondary"
               sec:authorize="isAuthenticated()"
               th:if="${question.author != null and #authentication.
getPrincipal().getUsername() == question.author.username}"
               th:text="수정"></a>
        </div>
    </div>
</div>
(... 생략 ...)
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 수정 버튼 만들기

1. [수정] 버튼이 로그인한 사용자와 글쓴이가 동일할 경우에만 노출되도록 #authentication.getPrincipal( ).getUsername( ) == question.author.username을 적용함.

   #authentication.getPrincipal( )은 타임리프에서 스프링 시큐리티와 함께 사용하는 표현식으로,
   이를 통해 현재 사용자가 인증되었다면 사용자 이름(사용자 ID)을 알 수 있음.

   만약 로그인한 사용자와 글쓴이가 다르다면 이 [수정] 버튼은 보이지 않을 것임

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 컨트롤러 수정하기 1

- 앞서 작성한 [수정] 버튼에 GET 방식의 @{|/question/modify/${question.id}|} 링크가 추가되었으므로 이 링크가 동작할 수 있도록 질문 컨트롤러를 다음과 같이 수정해 보자

• /question/QuestionController.java

```
(... 생략 ...)
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.http.HttpStatus;
import org.springframework.web.server.ResponseStatusException;
(... 생략 ...)
public class QuestionController {
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 컨트롤러 수정하기 1

```
    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @GetMapping("/modify/{id}")
    public String questionModify(QuestionForm questionForm, @PathVariable("id")
Integer id, Principal principal) {
        Question question = this.questionService.getQuestion(id);
        if(!question.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "수정 권한이
없습니다.");
        }
        questionForm.setSubject(question.getSubject());
        questionForm.setContent(question.getContent());
        return "question_form";
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 컨트롤러 수정하기 1

- 이와 같이 questionModify 메서드를 추가함
- 만약 현재 로그인한 사용자와 질문의 작성자가 동일하지 않을 경우에는 '수정 권한이 없습니다.'라는 오류가 발생하도록 함

- 그리고 수정할 질문의 제목과 내용을 화면에 보여 주기 위해 questionForm 객체에 id값으로 조회한 질문의 제목(subject)과 내용(object)의 값을 담아서 템플릿으로 전달함

- 이 과정이 없다면 질문 수정 화면에 '제목', '내용'의 값이 채워지지 않아 비워져 보일 것임
- 다만, 질문을 수정할 수 있는 새로운 템플릿을 만들지 않고 질문을 등록했을 때 사용한 question_form.html 템플릿을 사용함

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 등록 템플릿 수정하기

- 질문을 수정하기 위한 템플릿을 새로 작성해도 문제는 없지만
  제목과 내용을 기입하는 화면의 모양이 동일하므로
  여기서는 굳이 새로 만들지 않고 같은 템플릿을 사용하려고 함

- 그런데 question_form.html은 질문 등록을 위해 만든 템플릿이어서
  조금 수정해야 질문 등록과 수정 기능을 함께 사용할 수 있음

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 등록 템플릿 수정하기

- 다음과 같이 질문 등록 템플릿인 question_form.html을 수정해 보자

• /templates/question_form.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container">
    <h5 class="my-3 border-bottom pb-2">질문 등록 </h5>
    <form th:object="${questionForm}" method="post">
        <input type="hidden" th:name="${_csrf.parameterName}" th:value="${_csrf.
token}" />
        <div th:replace="~{form_errors :: formErrorsFragment}"></div>
        <div class="mb-3">
            <label for="subject" class="form-label">제목</label>
            <input type="text" th:field="*{subject}" class="form-control">
        </div>
        <div class="mb-3">
            <label for="content" class="form-label">내용</label>
            <textarea th:field="*{content}" class="form-control" rows="10"></textarea>
        </div>
        <input type="submit" value="저장하기" class="btn btn-primary my-2">
    </form>
</div>
</html>
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 등록 템플릿 수정하기

- 먼저 기존에 있던 <form> 태그의 th:action 속성을 삭제해야 함
- 단, th:action 속성을 삭제하면 CSRF값이 자동으로 생성되지 않아서
  CSRF값을 설정하기 위해 hidden 형태로 input 요소를 이와 같이 작성하여 추가해야 함

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 등록 템플릿 수정하기

- <form> 태그의 action 속성 없이 폼을 전송(submit)하면
  action 속성이 없더라도 자동으로 현재 URL
  (여기서는 웹 브라우저에 표시되는 URL 주소)을 기준으로 전송되는 규칙이 있음

- 즉, 질문 등록 시에 브라우저에 표시되는 URL은
  /question/create 이어서 action 속성이 지정되지 않더라도
  POST로 폼 전송할 때 action 속성으로 /question/create가 자동 설정됨

- 질문 수정 시에 브라우저에 표시되는 URL은 /question/modify/2와 같은 URL이기 때문에
  POST로 폼 전송할 때 action 속성에 /question/modify/2와 같은 URL이 설정되는 것임

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 서비스 수정하기

- 수정된 질문이 서비스를 통해 처리될 수 있도록 QuestionService를 다음과 같이 수정해 보자

• /question/QuestionService.java

```
(... 생략 ...)
public class QuestionService {

    (... 생략 ...)

    public void modify(Question question, String subject, String content) {
        question.setSubject(subject);
        question.setContent(content);
        question.setModifyDate(LocalDateTime.now());
        this.questionRepository.save(question);
    }
}
```

3-09 수정과 삭제 기능 추가하기

- **질문 수정 기능 생성하기**
  - **질문 컨트롤러 수정하기 2**
    - 다시 질문 컨트롤러로 돌아와 질문을 수정하는 화면에서
      질문 제목이나 내용을 변경하고 [저장하기] 버튼을 누르면
      호출되는 POST 요청을 처리하기 위해 다음과 같은 메서드를 추가해 보자

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 컨트롤러 수정하기 2

• /question/QuestionController.java

```
(... 생략 ...)
public class QuestionController {

    (... 생략 ...)
    @PreAuthorize("isAuthenticated()")
    @GetMapping("/modify/{id}")
    public String questionModify(@Valid QuestionForm questionForm, @PathVari
able("id") Integer id, Principal principal) {
        (... 생략 ...)
    }

    @PreAuthorize("isAuthenticated()")
    @PostMapping("/modify/{id}")
    public String questionModify(@Valid QuestionForm questionForm, BindingResult
bindingResult, Principal principal, @PathVariable("id") Integer id) {
        if (bindingResult.hasErrors()) {
            return "question_form";
        }
        Question question = this.questionService.getQuestion(id);
        if (!question.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "수정 권한이
없습니다.");
        }
        this.questionService.modify(question, questionForm.getSubject(), ques-
tionForm.getContent());
        return String.format("redirect:/question/detail/%s", id);
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 질문 컨트롤러 수정하기 2

- POST 형식의 /question/modify/{id} 요청을 처리하기 위해 이와 같이 questionModify 메서드를 추가함
- questionModify 메서드는 questionForm의 데이터를 검증하고 로그인한 사용자와 수정하려는 질문의 작성자가 동일한지도 검증함
- 검증이 통과되면 QuestionService에서 작성한 modify 메서드를 호출하여 질문 데이터를 수정함
- 그리고 수정이 완료되면 질문 상세 화면(/question/detail/(숫자))으로 리다이렉트함

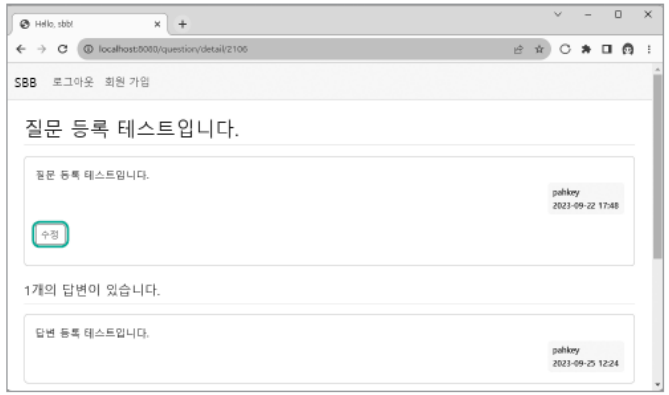
3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 수정 기능 확인하기

로컬 서버를 재시작한 뒤, 브라우저에서 질문 상세 페이지를 확인해 보자

1. 로그인한 사용자와 글쓴이가 같으면
   질문 상세 화면에 [수정] 버튼이 보일 것임



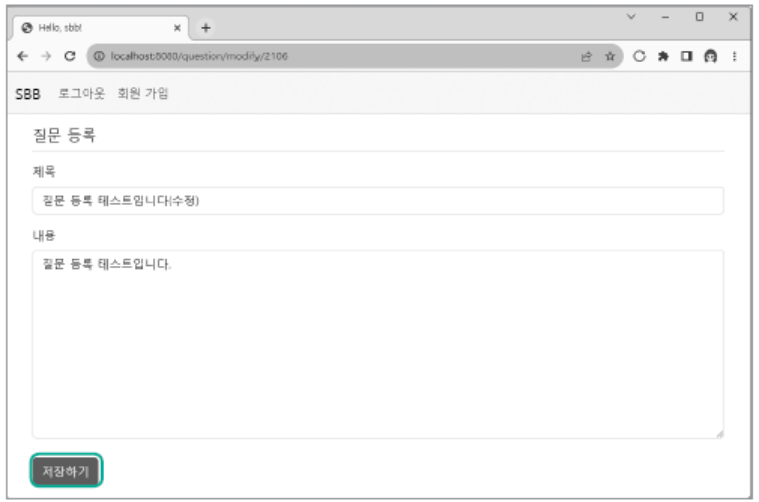
3-09 수정과 삭제 기능 추가하기

▪ 질문 수정 기능 생성하기

▪ 수정 기능 확인하기

2. [수정] 버튼을 클릭하여 수정 페이지로 이동하면 /question/modifiy/(질문 ID) URL로 넘어가고, 제목과 내용을 수정할 수 있음.

   제목 또는 내용을 수정한 후,
   [저장하기] 버튼을 클릭해
   기능이 잘 동작하는지 확인해 보자



3-09 수정과 삭제 기능 추가하기

- **질문 삭제 기능 생성하기**
  - **질문 삭제 버튼 만들기**
    - 질문을 삭제하는 기능을 추가해 보자
    - 질문 수정과 마찬가지로 질문 상세 화면에 [삭제] 버튼을 추가하여 삭제할 수 있게 하려고 함

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 삭제 버튼 만들기

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 질문 -->
<h2 class="border-bottom py-2" th:text="${question.subject}"></h2>
<div class="card my-3">
    <div class="card-body">
        (... 생략 ...)
        <div class="my-3">
            <a th:href="@{|/question/modify/${question.id}|}" class="btn
btn-sm btn-outline-secondary"
               sec:authorize="isAuthenticated()"
               th:if="${question.author != null and #authentication.
getPrincipal().getUsername() == question.author.username}"
               th:text="수정"></a>
            <a href="javascript:void(0);"
               th:data-uri="@{|/question/delete/${question.id}|}"
               class="delete btn btn-sm btn-outline-secondary"
sec:authorize="isAuthenticated()"
               th:if="${question.author != null and
#authentication.getPrincipal().getUsername() == question.author.username}"
               th:text="삭제"></a>
        </div>
    </div>
</div>
(... 생략 ...)
```

[수정] 버튼을 생성한다.

삭제 이벤트를 감지할 때 사용하는 클래스

[삭제] 버튼을 생성한다.

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 삭제 버튼 만들기

- 로그인한 사용자가 자신이 작성한 질문을 삭제할 수 있도록
  [삭제] 버튼을 클릭하면 자바스크립트 코드가 실행되도록 구현함

- [삭제] 버튼은 [수정] 버튼과는 달리 href 속성값을 javascript:void(0)로 설정하고
  삭제를 실행할 URL을 얻기 위해 th:data-uri 속성을 추가한 뒤,

  [삭제] 버튼을 클릭하는 이벤트를 확인하기 위해
  class 속성에 delete 항목을 추가함

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 삭제 버튼 만들기

- href에 삭제를 위한 URL을 직접 사용하지 않고 이러한 방식을 사용한 이유는 [삭제] 버튼을 클릭했을 때 '정말로 삭제하시겠습니까?'와 같은 메시지와 함께 별도의 확인 절차를 중간에 끼워 넣기 위해서임

- 만약 href에 삭제를 위한 URL을 직접 사용한다면 삭제를 확인하는 과정을 거치지 않고 질문이 삭제되어 버릴 것임

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

- 자바스크립트는 HTML, CSS와 함께 사용하며
  웹 페이지에 동적인 기능을 추가할 때 사용하는 스크립트 언어임

- 여기서는 이러한 자바스크립트를 활용해 [삭제] 버튼을 클릭했을 때
  '정말로 삭제하시겠습니까?'와 같은 메시지를 담은 확인 창을 호출하려고 함

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

1. 그러기 위해 다음과 같은 자바스크립트 코드가 필요함

```
<script type='text/javascript'>
const delete_elements = document.getElementsByClassName("delete");
Array.from(delete_elements).forEach(function(element) {
    element.addEventListener('click', function() {
        if(confirm("정말로 삭제하시겠습니까?")) {
            location.href = this.dataset.uri;
        };
    });
});
</script>
```

[확인]을 클릭했을 경우 삭제를 위한 URL을 호출하기 위해 작성한다.

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

1. 이 자바스크립트 코드의 의미는 delete라는 클래스를 포함하는 컴포넌트(예를 들어 버튼이나 링크 등)를 클릭하면 '정말로 삭제하시겠습니까?'라고 질문하고

   [확인]을 클릭했을 때 해당 컴포넌트에 속성으로 지정된 data-uri값으로 URL을 호출하라는 의미임.
   [확인] 대신 [취소]를 선택하면 아무런 일도 발생하지 않을 것임.

   따라서 이와 같은 스크립트를 추가하면 [삭제] 버튼을 클릭하고 [확인]을 선택하면 data-uri 속성에 해당하는 @{|/question/delete/${question.id}|} URL이 호출될 것임

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

2. 앞서 살펴본 자바스크립트 코드는 질문 상세 템플릿에 추가하면 됨.
그 전에 템플릿에 자바스크립트를 포함하는 방법을 먼저 알아보자.
자바스크립트는 HTML 구조에서 다음과 같이 </body> 태그 바로 위에 삽입하는 것을 추천함

```
<html>
<head>
(... 생략 ...)
</head>
<body>
(... 생략 ...)
<!-- 이곳에 추가 -->
</body>
</html>
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

2. 왜냐하면 화면 출력이 완료된 후에 자바스크립트가 실행되는 것이 좋기 때문.
   화면 출력이 완료되지 않은 상태에서 자바스크립트를 실행하면
   오류가 발생할 수도 있고 화면 로딩이 지연될 수도 있음.

   따라서 각 템플릿에서 자바스크립트를 </body> 태그 바로 위에 삽입하고,
   상속할 수 있도록 다음과 같이 layout.html을 수정해 보자

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

2.

• /templates/layout.html

```
<!doctype html>
<html lang="ko">
<head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
   <!-- Bootstrap CSS -->
   <link rel="stylesheet" type="text/css" th:href="@{/bootstrap.min.css}">
   <!-- sbb CSS -->
   <link rel="stylesheet" type="text/css" th:href="@{/style.css}">
   <title>Hello, sbb!</title>
</head>
<body>
<!-- 내비게이션 바 -->
<nav th:replace="~{navbar :: navbarFragment}"></nav>
```

```
<!-- 기본 템플릿 안에 삽입될 내용 Start -->
<th:block layout:fragment="content"></th:block>
<!-- 기본 템플릿 안에 삽입될 내용 End -->
<!-- Bootstrap JS -->
<script th:src="@{/bootstrap.min.js}"></script>
<!-- 자바스크립트 Start -->
<th:block layout:fragment="script"></th:block>
<!-- 자바스크립트 End -->
</body>
</html>
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

2. layout.html을 상속하는 템플릿들에서 content 블록을 구현하게 했던 것과
마찬가지 방법으로 script 블록을 구현할 수 있도록
</body> 태그 바로 위에 <th:block layout:fragment="script"></th:block> 블록을 추가함.

   이렇게 하면 이제 layout.html을 상속하는 템플릿은
자바스크립트의 삽입 위치를 신경 쓰지 않아도 되고,
필요할 경우에 스크립트 블록을 구현하여 자바스크립트를 작성할 수 있음

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 삭제를 위한 자바스크립트 작성하기

3. 이제 question_detail.html 하단에 스크립트 블록을 다음과 같이 추가하여 자바스크립트가 실행될 수 있도록 해보자

• /templates/question_detail.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    (... 생략 ...)
    <!-- 답변 작성 -->
    (... 생략 ...)
    </form>
</div>
<script layout:fragment="script" type='text/javascript'>
const delete_elements = document.getElementsByClassName("delete");
Array.from(delete_elements).forEach(function(element) {
    element.addEventListener('click', function() {
        if(confirm("정말로 삭제하시겠습니까?")) {
            location.href = this.dataset.uri;
        };

    });
});
</script>
</html>
```

앞서 눈으로만 확인했던 코드를 이렇게 작성한다.

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

1. 먼저 질문 삭제 기능을 QuestionService에 추가해 보자

• /question/QuestionService.java

```
(... 생략 ...)
public class QuestionService {

    (... 생략 ...)

    public void modify(Question question, String subject, String content) {
        (... 생략 ...)
    }

    public void delete(Question question) {
        this.questionRepository.delete(question);
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

2. 질문 컨트롤러에서는 [삭제] 버튼을 클릭했을 때 @{|/question/delete/${question.id}|} URL을 처리할 수 있도록 QuestionController에 다음과 같은 메서드를 추가하자

• /question/QuestionController.java

```
(... 생략 ...)
public class QuestionController {

    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @PostMapping("/modify/{id}")
    (... 생략 ...)
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

2.

```
    @PreAuthorize("isAuthenticated()")
    @GetMapping("/delete/{id}")
    public String questionDelete(Principal principal, @PathVariable("id") Integer
id) {
        Question question = this.questionService.getQuestion(id);
        if (!question.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "삭제 권한이
없습니다.");
        }
        this.questionService.delete(question);
        return "redirect:/";
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

2. 사용자가 [삭제] 버튼을 클릭했다면
   URL로 전달받은 id값을 사용하여 Question 데이터를 조회한 후,

   로그인한 사용자와 질문 작성자가 동일할 경우
   앞서 작성한 서비스를 이용하여 질문을 삭제하게 함.

   그리고 질문을 삭제한 후에는 질문 목록 화면(/)으로 돌아갈 수 있도록 함

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

3. 로컬 서버를 재시작한 후, 질문 상세 페이지를 확인해보자.
   질문을 작성한 사용자와 로그인한 사용자가 동일하다면
   다음과 같이 질문 상세 화면에 [삭제] 버튼이 노출될 것임

3-09 수정과 삭제 기능 추가하기

▪ 질문 삭제 기능 생성하기

▪ 질문 서비스와 컨트롤러 수정하기

4. [삭제] 버튼을 클릭하면 다음과 같이 메시지가 등장함.
   [확인] 버튼을 클릭하면 다시 질문 목록 페이지로 돌아오고
   해당 질문이 삭제된 것을 확인할 수 있음.
   삭제 기능이 정상적으로 동작한 것임

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

- 이번에는 답변 수정 기능을 구현해 보자.
  질문 수정 기능과 비슷한 과정으로 진행할 것임

- 다만, 답변 수정 기능을 구현하기 위한 템플릿이 따로 없으므로
  답변 수정 시 사용할 템플릿이 추가로 필요함

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

1. 질문 상세 템플릿에서 답변 목록이 출력되는 부분에 답변 수정 버튼을 추가해 보자

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 답변 반복 시작 -->
<div class="card my-3" th:each="answer : ${question.answerList}">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;"
        th:text="${answer.content}"></div>
        <div class="d-flex justify-content-end">
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${answer.author != null}"
                    th:text="${answer.author.username}"></span>
                </div>
                <div th:text="${#temporals.format(answer.createDate, 'yyyy-MM-dd
HH:mm')}"></div>
            </div>
        </div>
        <div class="my-3">
            <a th:href="@{|/answer/modify/${answer.id}|}" class="btn btn-sm
btn-outline-secondary"
                sec:authorize="isAuthenticated()"
                th:if="${answer.author != null and #authentication.getPrincipal().
getUsername() == answer.author.username}"
                th:text="수정"></a>
        </div>
    </div>
</div>
<!-- 답변 반복 끝  -->
(... 생략 ...)
```

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

1. 로그인한 사용자와 답변 작성자가 동일한 경우 답변의 [수정] 버튼이 노출되도록 함.
   [답변] 버튼을 누르면 '/answer/modify/ 답변 ID' 형태의 URL이 GET 방식으로 요청될 것임

1. 답변을 수정하려면 답변을 먼저 조회해야 하므로
   AnswerService에 답변을 조회하는 기능을 추가하고,
   답변을 수정할 수 있는 기능도 다음과 같이 추가해 보자

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

2. • /answer/AnswerService.java

```
(... 생략 ...)
import java.util.Optional;
import com.mysite.sbb.DataNotFoundException;
(... 생략 ...)

public class AnswerService {

    (... 생략 ...)


    public void create(Question question, String content, SiteUser author) {
        (... 생략 ...)
    }

    public Answer getAnswer(Integer id) {
        Optional<Answer> answer = this.answerRepository.findById(id);
        if (answer.isPresent()) {
            return answer.get();
        } else {
            throw new DataNotFoundException("answer not found");
        }
    }

    public void modify(Answer answer, String content) {
        answer.setContent(content);
        answer.setModifyDate(LocalDateTime.now());
        this.answerRepository.save(answer);
    }
}
```

이와 같이 해당 답변을 조회하는 getAnswer 메서드와
답변 내용을 수정하는 modify 메서드를 추가함

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

3. 답변 영역의 [수정] 버튼 클릭 시 GET 방식으로 요청되는 '/answer/modify/답변ID' URL을 처리하기 위해 다음과 같이 AnswerController를 수정해 보자

• /answer/AnswerController.java

```
(... 생략 ...)
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.server.ResponseStatusException;
(... 생략 ...)
```

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

3.

```
public class AnswerController {

    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @PostMapping("/create/{id}")
    (... 생략 ...)
    @PreAuthorize("isAuthenticated()")
    @GetMapping("/modify/{id}")
    public String answerModify(AnswerForm answerForm, @PathVariable("id") Integer
id, Principal principal) {
        Answer answer = this.answerService.getAnswer(id);
        if (!answer.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "수정 권한
이 없습니다.");
        }
        answerForm.setContent(answer.getContent());
        return "answer_form";
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 버튼 추가하고 서비스와 컨트롤러 수정하기

3. 이와 같이 answerModify 메서드를 추가함.

   DB에서 답변 ID를 통해 조회한 답변 데이터의 내용(content)을
   AnswerForm 객체에 대입하여 answer_form.html 템플릿에서 사용할 수 있도록 함.

   아직 answer_form.html이 존재하지 않으므로
   다음 실습에서 해당 템플릿을 만들어 보자

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 답변 수정 템플릿 생성하기

- 답변을 수정하기 위해 템플릿을 만들어 보자.
  templates에 answer_form.html 파일을 생성한 뒤, 다음과 같은 내용을 입력해 보자

• /templates/answer_form.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container">
    <h5 class="my-3 border-bottom pb-2">답변 수정</h5>
    <form th:object="${answerForm}" method="post">
        <input type="hidden" th:name="${_csrf.parameterName}" th:value="${_csrf.
token}"/>
        <div th:replace="~{form_errors :: formErrorsFragment}"></div>
        <div class="mb-3">
            <label for="content" class="form-label">내용</label>
            <textarea th:field="*{content}" class="form-control" rows="10"></textarea>
        </div>
        <input type="submit" value="저장하기" class="btn btn-primary my-2">
    </form>
</div>
</html>
```

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 답변 수정 템플릿 생성하기

- 답변 작성 시 사용하는 <form> 태그에도 역시 action 속성을 사용하지 않음
- 앞서 설명했듯이 action 속성을 생략하면 현재 호출된 URL로 폼이 전송됨
- th:action 속성이 없으므로 csrf 항목을 직접 추가함

▪ 답변 컨트롤러 재수정하기

1. 이제 폼을 통해 POST 방식으로 요청되는
   /answer/modify/ 답변 ID URL을 처리하기 위해
   다음과 같이 AnswerController로 돌아가 코드를 추가해 보자

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 답변 컨트롤러 재수정하기

1.

• /answer/AnswerController.java

```
(... 생략 ...)
public class AnswerController {

    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @GetMapping("/modify/{id}")
    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @PostMapping("/modify/{id}")
    public String answerModify(@Valid AnswerForm answerForm, BindingResult bind
ingResult,
            @PathVariable("id") Integer id, Principal principal) {
        if (bindingResult.hasErrors()) {
            return "answer_form";
        }
        Answer answer = this.answerService.getAnswer(id);
        if (!answer.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "수정 권한이
없습니다.");
        }
        this.answerService.modify(answer, answerForm.getContent());
        return String.format("redirect:/question/detail/%s", answer.getQuestion().
getId());
    }
}
```

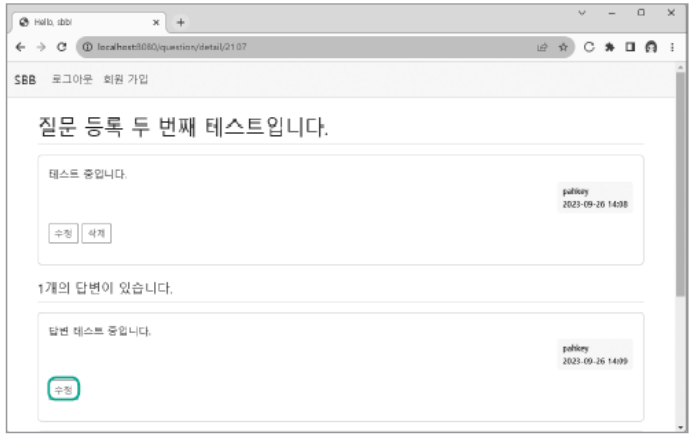
POST 방식의 답변 수정을 처리하기 위해 answerModify 메서드를 추가함.
그리고 답변 수정을 완료한 후에는 질문 상세 페이지로 리다이렉트하도록 함

3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 답변 컨트롤러 재수정하기

2. 답변 수정도 질문 수정과 마찬가지로 답변 등록 사용자와 로그인 사용자가 동일할 때만 [수정] 버튼이 나타남.
로컬 서버를 재시작한 후, 질문 상세 페이지에서 확인해 보자



3-09 수정과 삭제 기능 추가하기

▪ 답변 수정 기능 추가하기

▪ 답변 컨트롤러 재수정하기

3. [수정] 버튼을 클릭해 답변 내용을 수정한 후,
   [저장하기] 버튼을 클릭해 답변 수정 기능이 잘 동작하는지 확인해 보자

3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

1. 질문 상세 템플릿에 답변을 삭제할 수 있는 버튼을 다음과 같이 추가하자

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 답변 반복 시작 -->
<div class="card my-3" th:each="answer : ${question.answerList}">
    <div class="card-body">
        (... 생략 ...)
        <div class="my-3">
            <a th:href="@{|/answer/modify/${answer.id}|}" class="btn
btn-sm btn-outline-secondary"
               sec:authorize="isAuthenticated()"
               th:if="${answer.author != null and
#authentication.getPrincipal().getUsername() == answer.author.username}"
               th:text="수정"></a>
            <a href="javascript:void(0);" th:data-uri="@{|/answer/delete/$
{answer.id}|}"
               class="delete btn btn-sm btn-outline-secondary"
sec:authorize="isAuthenticated()"
               th:if="${answer.author != null and #authentication.
getPrincipal().getUsername() == answer.author.username}"
               th:text="삭제"></a>
        </div>
    </div>
</div>
<!-- 답변 반복 끝  -->
(... 생략 ...)
```

[수정] 버튼을 생성한다.

[삭제] 버튼을 생성한다.

3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

1. [수정] 버튼 옆에 [삭제] 버튼이 노출되도록 [삭제] 버튼을 생성하는 코드를 추가함.

   질문의 [삭제] 버튼과 마찬가지로
   답변의 [삭제] 버튼에 delete 클래스를 적용했으므로

   [삭제] 버튼을 누르면 앞서 작성한 자바스크립트에 의해
   data-uri 속성에 설정한 url이 실행됨

3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

2. 답변을 삭제하기 위해 AnswerService에 다음과 같이 코드를 추가해 보자

• /answer/AnswerService.java

```
(... 생략 ...)
public class AnswerService {

    (... 생략 ...)

    public void delete(Answer answer) {
        this.answerRepository.delete(answer);
    }
}
```

3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

3. 이제 답변의 [삭제] 버튼을 누르면 GET 방식으로 요청되는 /answer/delete/답변 ID URL을 처리하기 위해 다음과 같이 AnswerController를 수정해 보자

• /answer/AnswerController.java

```
(... 생략 ...)
public class AnswerController {

    (... 생략 ...)

    @PreAuthorize("isAuthenticated()")
    @GetMapping("/delete/{id}")
    public String answerDelete(Principal principal, @PathVariable("id") Integer
id) {
        Answer answer = this.answerService.getAnswer(id);
        if (!answer.getAuthor().getUsername().equals(principal.getName())) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "삭제 권한이
없습니다.");
        }
        this.answerService.delete(answer);
        return String.format("redirect:/question/detail/%s", answer.getQuestion().
getId());
    }
}
```

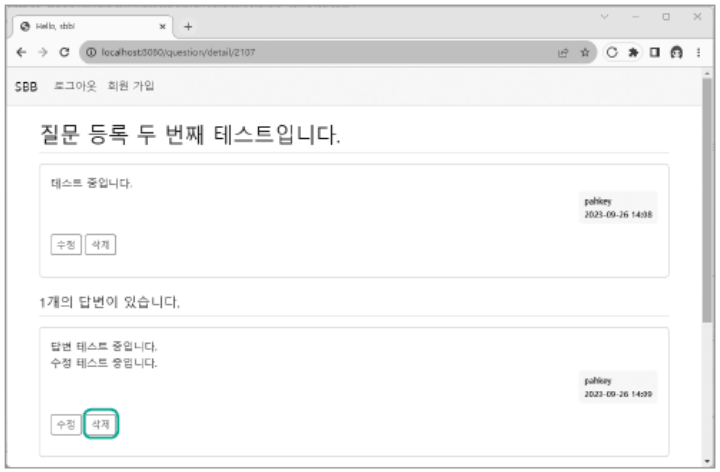
3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

3. 답변을 삭제하는 answerDelete 메서드를 추가함.
   답변을 삭제한 후에는 해당 답변이 있던 질문 상세 화면으로 이동할 수 있도록 만듦

3. 로컬 서버를 재시작한 후,
   질문 상세 페이지를 확인해 보자

   다음과 같이 질문 상세 화면에서
   답변을 작성한 사용자와
   로그인한 사용자가 같으면
   [삭제] 버튼이 나타날 것임

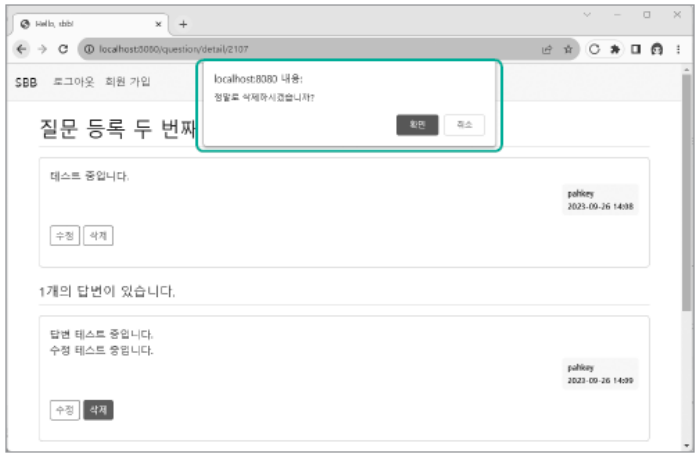


3-09 수정과 삭제 기능 추가하기

▪ 답변 삭제 기능 추가하기

5. [삭제] 버튼을 클릭하면 질문을 삭제할 때와 마찬가지로
다음과 같이 메시지가 등장함.

   여기서는 [확인] 버튼을 클릭하면 다시 질문 상세 페이지로 돌아오고
해당 답변이 삭제된 것을 확인할 수 있음



3-09 수정과 삭제 기능 추가하기

▪ 수정 일시 표시하기

1. 이미 표시된 질문과 답변의 작성 일시 바로 왼쪽에 수정 일시를 추가해 보자

• /templates/question_detail.html

```
(... 생략 ...)
<!-- 질문 -->
<h2 class="border-bottom py-2" th:text="${question.subject}"></h2>
<div class="card my-3">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;" th:text="${question.
content}"></div>
        <div class="d-flex justify-content-end">
            <div th:if="${question.modifyDate != null}" class="badge bg-light
text-dark p-2 text-start mx-3"
                <div class="mb-2">modified at</div>
                <div th:text="${#temporals.format(question.modifyDate,
'yyyy-MM-dd HH:mm')}"></div>
            </div>
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${question.author != null}"
th:text="${question.author.username}"></span>
                </div>
                <div th:text="${#temporals.format(question.createDate,
'yyyy-MM-dd HH:mm')}"></div>
            </div>
        </div>
```

질문의 수정 일시를 추가한다.

질문의 작성 일시를 위해 작성한다.

```
        (... 생략 ...)
(... 생략 ...)
<!-- 답변 반복 시작 -->
<div class="card my-3" th:each="answer : ${question.answerList}">
    <div class="card-body">
        <div class="card-text" style="white-space: pre-line;" th:text="${answer.
content}"></div>
        <div class="d-flex justify-content-end">
            <div th:if="${answer.modifyDate != null}" class="badge bg-light
text-dark p-2 text-start mx-3">
                <div class="mb-2">modified at</div>
                <div th:text="${#temporals.format(answer.modifyDate,
'yyyy-MM-dd HH:mm')}"></div>
            </div>
            <div class="badge bg-light text-dark p-2 text-start">
                <div class="mb-2">
                    <span th:if="${answer.author != null}" th:text="${answer.
author.username}"></span>
                </div>
                <div th:text="${#temporals.format(answer.createDate,
'yyyy-MM-dd HH:mm')}"></div>
            </div>
        </div>
        (... 생략 ...)
```

답변의 수정 일시를 추가한다.

질문의 작성 일시를 위해 작성한다.

3-09 수정과 삭제 기능 추가하기

▪ 수정 일시 표시하기

2. 로컬 서버를 재시작하고 로그인한 뒤, 질문 내용과 답변 내용을 모두 수정해 보자.
질문이나 답변에 수정 일시가 존재하면(즉, null이 아니면)
수정 일시가 작성 일시 바로 왼쪽에 표시됨