2024-2 오픈소스 개발 프로젝트01

SW중심대학사업단 강재구

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

05

충북대학교
SW중심대학사업단

03. SBB 서비스 개발하기 (1)

03

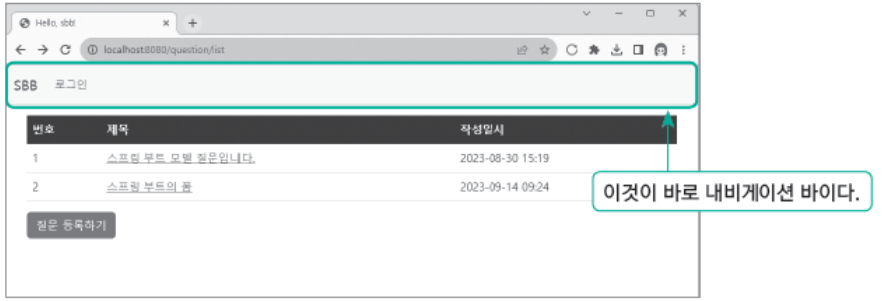
SBB 서비스 개발하기

3-01 내비게이션 바 추가하기

- 웹 서비스 개발자는 서비스 이용자가 편하게 사용할 수 있도록
  작은 기능 하나에도 공을 들임

- 지금까지 우리는 질문 목록, 질문 상세, 질문 등록, 답변 등록 등
  굵직한 기능을 중심으로 SBB 서비스를 구현했지만
  이제부터는 사용자가 이 서비스를 좀 더 편리하게 이용할 수 있도록
  다양한 기능을 구현해 보려고 함

- 먼저, 어떠한 화면에 있더라도 항상 메인 화면으로 돌아갈 수 있도록
  내비게이션 바를 만들어 화면 상단에 고정해 보자



3-01 내비게이션 바 추가하기

▪ 내비게이션 바 만들기

1. 내비게이션 바는 모든 화면 위쪽에 고정되어 있는 부트스트랩의 컴포넌트임.
   내비게이션 바는 모든 페이지에서 공통으로 보여야 하므로
   다음과 같이 layout.html 템플릿에 내용을 추가하자

• /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>
```

3-01 내비게이션 바 추가하기

▪ 내비게이션 바 만들기

1.

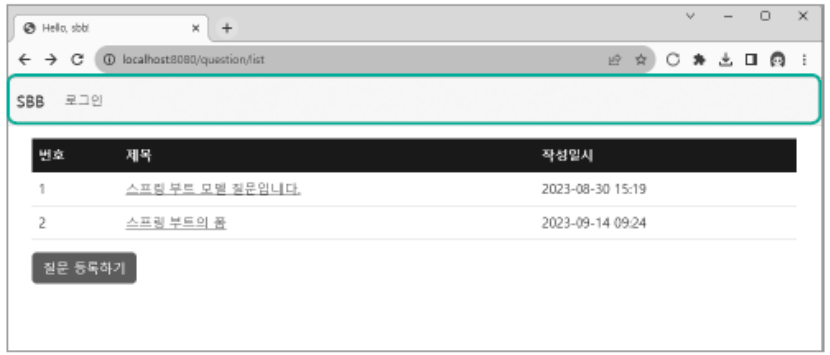

```
<nav class="navbar navbar-expand-lg navbar-light bg-light border-bottom">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">SBB</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" 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" href="#">로그인</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
<!-- 기본 템플릿 안에 삽입될 내용 Start -->
<th:block layout:fragment="content"></th:block>
<!-- 기본 템플릿 안에 삽입될 내용 End -->
</body>
</html>
```

3-01 내비게이션 바 추가하기

▪ 내비게이션 바 만들기

1. 이 코드는 부트스트랩을 활용하여 내비게이션 바를 생성하는 내용을 작성한 것임.
   이와 같이 <li> 태그를 활용하여 내비게이션 바에 메뉴를 추가할 수 있음

1. 이제 브라우저에서 질문 목록 페이지를 요청하면
   화면 상단에 다음과 같은 내비게이션 바가 보일 것임



3-01 내비게이션 바 추가하기

▪ 내비게이션 바 만들기

2. 질문 목록 페이지 외에 질문 상세나 질문 등록 페이지에서
내비게이션 바의 SBB 로고를 클릭하면
바로 메인 페이지인 질문 목록 페이지로 돌아갈 수 있음.

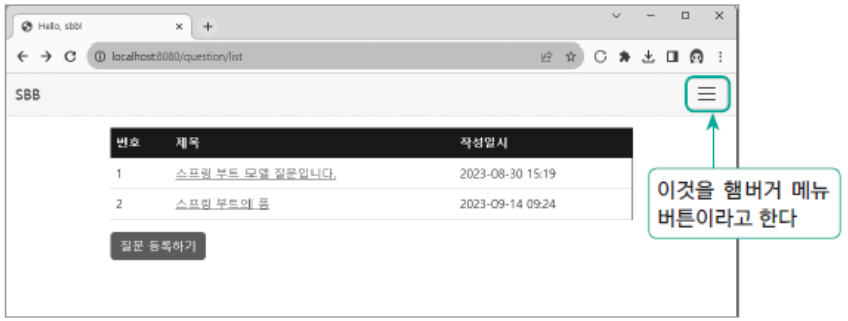
SBB 로고를 클릭해 제대로 작동하는지 확인해 보자

3-01 내비게이션 바 추가하기

▪ 내비게이션 바의 숨은 기능 알기

1. 이 내비게이션 바에는 재미있는 기능이 하나 숨어 있음.
   한번 아무 페이지나 접속해서 브라우저의 가로 사이즈를
   마우스를 이용하여 점점 줄여 보자.

   그러면 어느 순간 햄버거 메뉴버튼이 생김.
   이와 동시에 로그인 링크는 사라짐.



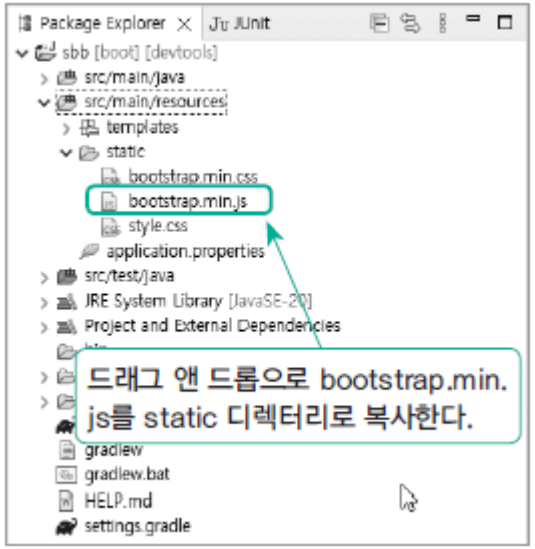
   이와 같이 부트스트랩은 브라우저의 크기가 작아지면
   자동으로 내비게이션 바에 있는 링크들을 햄버거 메뉴 버튼으로 숨김

3-01 내비게이션 바 추가하기

▪ 내비게이션 바의 숨은 기능 알기

2. 햄버거 메뉴 버튼을 클릭하면 숨어 있는 로그인 링크가 보여야 함.
하지만 현 상태에서는 햄버거 메뉴 버튼을 클릭해도 아무런 변화가 없음.

햄버거 메뉴 버튼을 활용할 수 있도록
부트스트랩 자바스크립트 파일(bootstrap.min.js)을
static 디렉터리로 복사해 보자



3-01 내비게이션 바 추가하기

▪ 내비게이션 바의 숨은 기능 알기

3. 이제 추가한 자바스크립트(Javascript, JS) 파일을 사용할 수 있도록 layout.html의 </body> 태그 바로 위에 다음과 같이 추가하자.

• /templates/layout.html

```
<!doctype html>
<html lang="ko">
(... 생략 ...)
<!-- 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>
</body>
</html>
```

부트스트랩의 JS 파일을 사용하겠다는 주석과 함께 이와 같이 JS 파일을 추가함

3-01 내비게이션 바 추가하기

▪ 내비게이션 바의 숨은 기능 알기

4. 이렇게 수정하면 햄버거 메뉴 버튼 클릭 시 숨어 있는 링크가 다음과 같이 표시되는 것을 확인할 수 있음

3-01 내비게이션 바 추가하기

▪ 내비게이션 바 분리하기

1. 다음과 같이 내비게이션 바를 활용하기 위한 공통 템플릿으로 navbar.html를 작성해 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" href="#">로그인</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

'내비게이션 바 만들기'에서 추가한 코드를 복사해 붙여 넣은 후, 이 부분만 수정해 보자.

'내비게이션 바 만들기'에서 layout.html에 추가한 내용을 복사해 만들면 돼!

3-01 내비게이션 바 추가하기

▪ 내비게이션 바 분리하기

2. 다시 layout.html로 돌아가 navbar.html에 작성한 내용을 모두 삭제한 후, 다음과 같이 작성해 보자

• /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>
</body>
</html>
```

20줄에 가까웠던 코드가 단 한 줄로 해결됐다!

3-01 내비게이션 바 추가하기

▪ 내비게이션 바 분리하기

2. 기존의 내비게이션 바 HTML 코드들을 삭제하고 navbar.html 템플릿을 타임리프의 th:replace 속성으로 layout.html 템플릿에 포함시킴.

   사실, navbar.html 파일은 form_errors.html처럼 다른 템플릿들에서 중복해 사용하지는 않지만 독립된 하나의 템플릿으로 관리하는 것이 유지 보수에 유리하므로 이와 같이 분리함.

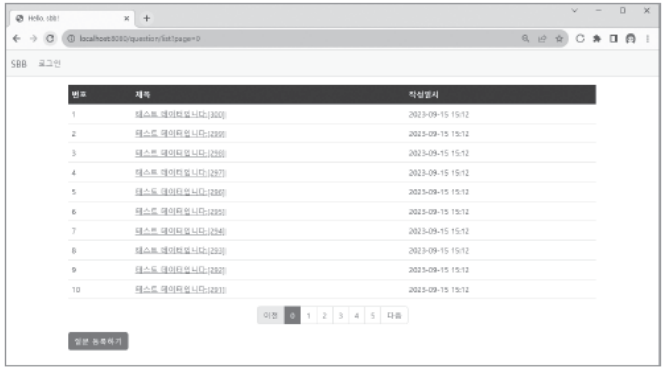
   공통 템플릿을 따로 관리하는 것은 코드의 재사용성을 높여 줄 뿐만 아니라 코드의 유지 보수에도 도움이 됨

3-02 페이징 기능 추가하기

- SBB의 질문 목록은 현재 페이징 기능이 없어 게시물을 300개 작성하면
  한 페이지에 300개의 게시물이 모두 조회됨

- 이 경우 한 화면에 표시할 게시물이 많아져서 스크롤바를 내려야하는 불편함이 생김.
  이를 해결하기 위해 질문 목록 화면에 페이징 기능을 적용해 보자

- 여기서 페이징(paging)이란 입력된 정보나 데이터를 여러 페이지에 나눠 표시하고,
  사용자가 페이지를 이동할 수 있게 하는 기능을 말함



3-02 페이징 기능 추가하기

- **대량 테스트 데이터 만들기**
  - 페이징을 구현하기 전에 페이징을 테스트할 수 있을 정도로 충분한 테스트 데이터를 만들어 보자
  - 대량의 테스트 데이터를 만드는 가장 간단한 방법은 2-05절에서 살펴본 스프링 부트의 테스트 프레임워크를 이용하는 것임

3-02 페이징 기능 추가하기

▪ 대량 테스트 데이터 만들기

1. 테스트 케이스를 작성하기 위해 SbbApplicationTests.java 파일을 수정해 보자.

• SbbApplicationTests.java

```
package com.mysite.sbb;

import com.mysite.sbb.question.QuestionService;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class SbbApplicationTests {

    @Autowired
    private QuestionService questionService;

    @Test
    void testJpa() {
        for (int i = 1; i <= 300; i++) {
            String subject = String.format("테스트 데이터입니다:[%03d]" , i);
            String content = "내용 없음 ";
            this.questionService.create(subject, content);
        }
    }
}
```

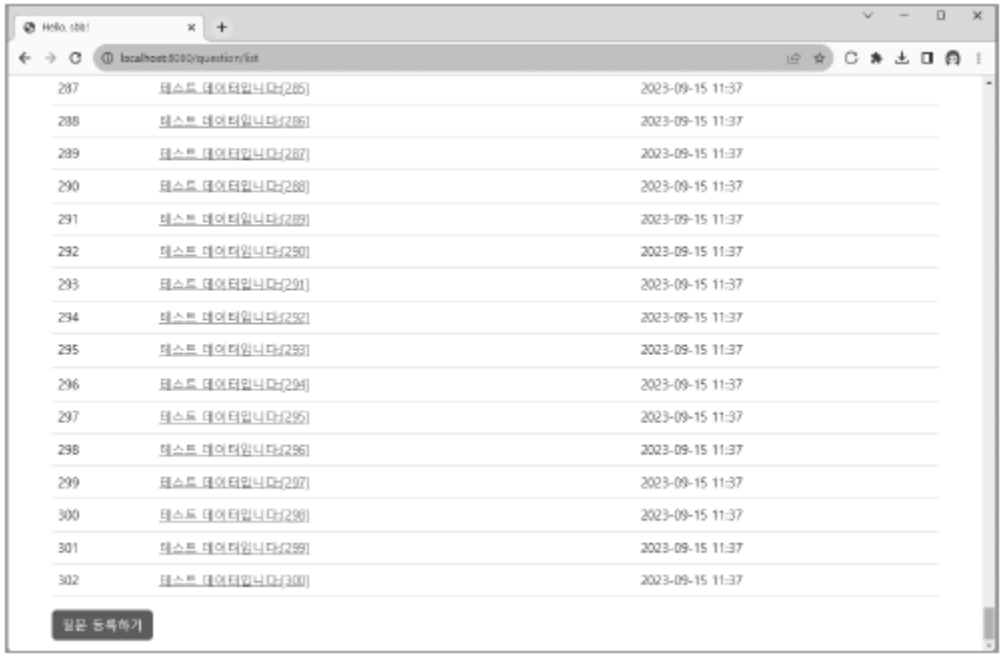
제목에 번호를 부여하는 코드이다.

이와 같이 총 300개의 테스트 데이터를 생성하는 테스트 케이스를 작성함

3-02 페이징 기능 추가하기

▪ 대량 테스트 데이터 만들기

2. 로컬 서버를 중지하고 [Run → Run As → Junit Test]로 testJpa 메서드를 실행하자. 그리고 다시 로컬 서버를 실행한 후, 브라우저에서 질문 목록 페이지를 요청해 보자



3-02 페이징 기능 추가하기

▪ 대량 테스트 데이터 만들기

2. 이와 같이 테스트 케이스로 등록한 데이터가 보일 것임.

   그리고 300개 이상의 데이터가 한 페이지 보여지는 것을 확인할 수 있음.
   300개가 넘는 데이터를 확인하려면 계속 스크롤을 내려야 함.

   이러한 불편함을 해결하기 위해 이어서 페이징 기능을 구현하고,
   등록한 게시물이 최신순으로 보여지는 기능까지 추가해 보자

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

- 페이징을 구현하기 위해 추가로 설치해야 하는 라이브러리는 없음
- JPA 환경 구축 시 설치했던 JPA 관련 라이브러리에
  이미 페이징을 위한 패키지들이 들어 있기 때문
- 그러므로 다음 클래스들을 이용하면 페이징을 쉽게 구현할 수 있음

> • org.springframework.data.domain.Page: 페이징을 위한 클래스이다.
> • org.springframework.data.domain.PageRequest: 현재 페이지와 한 페이지에 보여 줄 게시물 개수 등을 설정하여 페이징 요청을 하는 클래스이다.
> • org.springframework.data.domain.Pageable: 페이징을 처리하는 인터페이스이다.

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

1. 위에 소개한 3가지 클래스를 사용하여 페이징을 구현해 보자.

   먼저 QuestionRepository에 다음과 같이 페이징을 구현하기 위한 클래스들을 import 한 후, findAll 메서드를 추가해 보자.

   Pageable 객체를 입력받아 Page<Question> 타입 객체를 리턴하는 findAll 메서드를 생성함

• /question/QuestionRepository.java

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

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

public interface QuestionRepository extends JpaRepository<Question, Integer> {
    Question findBySubject(String subject);
    Question findBySubjectAndContent(String subject, String content);
    List<Question> findBySubjectLike(String subject);
    Page<Question> findAll(Pageable pageable);
}
```

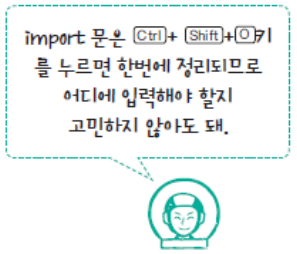
3-02 페이징 기능 추가하기

▪ 페이징 구현하기

2. 이번에는 QuestionService도 다음과 같이 수정해 보자

• /question/QuestionService.java

```
(... 생략 ...)

import org.springframework.stereotype.Service;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;

import lombok.RequiredArgsConstructor;

@RequiredArgsConstructor
@Service
public class QuestionService {
    private final QuestionRepository questionRepository;

    public Page<Question> getList(int page) {
        Pageable pageable = PageRequest.of(page, 10);
        return this.questionRepository.findAll(pageable);
    }

    public Question getQuestion(Integer id) {
       (... 생략 ...)
    }

    public void create(String subject, String content) {
        (... 생략 ...)
    }
}
```



3-02 페이징 기능 추가하기

▪ 페이징 구현하기

2. 질문 목록을 조회하는 getList 메서드를 이와 같이 변경함.

   getList 메서드는 정수 타입의 페이지 번호를 입력받아
   해당 페이지의 Page 객체를 리턴하도록 변경함.

   Pageable 객체를 생성할 때 사용한 PageRequest.of(page, 10)에서
   page는 조회할 페이지의 번호이고
   10은 한 페이지에 보여 줄 게시물의 개수를 의미함.

   이렇게 하면 데이터 전체를 조회하지 않고
   해당 페이지의 데이터만 조회하도록 쿼리가 변경됨

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

3. QuestionService의 getList 메서드 입출력 구조가 변경되었으므로 QuestionController도 다음과 같이 수정해야 함

• /question/QuestionController.java

```
(... 생략 ...)
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.data.domain.Page;

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

    (... 생략 ...)

    @GetMapping("/list")
    public String list(Model model, @RequestParam(value="page", defaultValue="0")
int page) {
        Page<Question> paging = this.questionService.getList(page);
        model.addAttribute("paging", paging);
        return "question_list";
    }
    @GetMapping(value = "/detail/{id}")
    (... 생략 ...)
    @GetMapping("/create")
    (... 생략 ...)
    @GetMapping("/create")
    (... 생략 ...)
}
```

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

3. http://localhost:8080/question/list?page=0와 같이
GET 방식으로 요청된 URL에서 page값을 가져오기 위해 list 메서드의 매개변수로
@RequestParam(value="page",defaultValue="0") int page가 추가됨.

   URL에 매개변수로 page가 전달되지 않은 경우 기본값은 0이 되도록 설정함

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

3. 템플릿에 Page 클래스의 객체인 paging을 model에 설정하여 전달함.
paging 객체에는 다음과 같은 속성들이 있는데,
이 속성들은 템플릿에서 페이징을 처리할 때 필요하므로 미리 알아 두자

속성	설명
paging.isEmpty	페이지 존재 여부를 의미한다(게시물이 있으면 false, 없으면 true).
paging.totalElements	전체 게시물 개수를 의미한다.
paging.totalPages	전체 페이지 개수를 의미한다.
paging.size	페이지당 보여 줄 게시물 개수를 의미한다.
paging.number	현재 페이지 번호를 의미한다.
paging.hasPrevious	이전 페이지의 존재 여부를 의미한다.
paging.hasNext	다음 페이지의 존재 여부를 의미한다.

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

4. 컨트롤러에서 model 객체에 기존에 전달했던 이름인 'questionList' 대신 'paging'으로 전달하기 때문에 질문 목록 템플릿(question_list.html)을 다음과 같이 변경해야 함

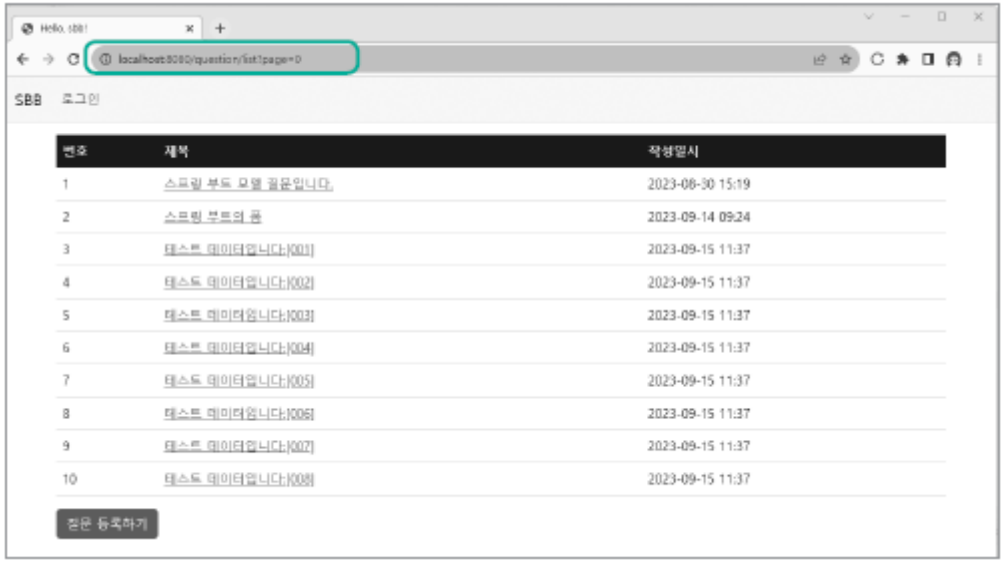
• /templates/question_list.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <table class="table">
        (... 생략 ...)
        <tbody>
            <tr th:each="question, loop : ${paging}">
                (... 생략 ...)
            </tr>
        </tbody>
    </table>
    <a th:href="@{/question/create}" class="btn btn-primary">질문 등록하기</a>
</div>
</html>
```

3-02 페이징 기능 추가하기

▪ 페이징 구현하기

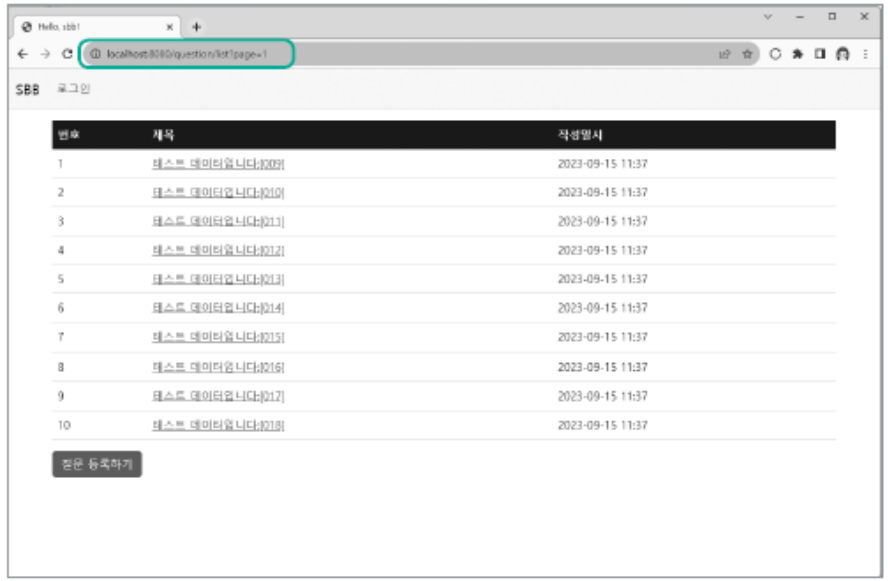
5. 수정한 후, 브라우저에서 http://localhost:8080/question/list?page=0이라는 URL을 요청해 보자. 다음과 같이 첫 페이지에 해당하는 게시물 10개만 조회되는 것을 확인할 수 있음



3-02 페이징 기능 추가하기

▪ 페이징 구현하기

6. 이번에는 http://localhost:8080/question/list?page=1과 같이 URL을 요청하면 다음과 같이 두 번째 페이지에 해당하는 게시물들이 조회됨



3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 추가하기

1. question_list.html의 </table> 태그 바로 밑에 다음 코드를 작성해 보자

• /templates/question_list.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <table class="table">
        (... 생략 ...)
    </table>
    <!-- 페이징 처리 시작 -->
    <div th:if="${!paging.isEmpty()}">
        <ul class="pagination justify-content-center">
            <li class="page-item" th:classappend="${!paging.hasPrevious} ? 'disabled'">
                <a class="page-link" th:href="@{|?page=${paging.number-1}|}">
                    <span>이전</span>
                </a>
            </li>
            <li th:each="page: ${#numbers.sequence(0, paging.totalPages-1)}"
th:classappend="${page == paging.number} ? 'active'"class="page-item">
                <a th:text="${page}" class="page-link"
                th:href="@{|?page=${page}|}"></a>
            </li>
            <li class="page-item" th:classappend="${!paging.hasNext} ? 'disabled'">
                <a class="page-link" th:href="@{|?page=${paging.number+1}|}">
                    <span>다음</span>
                </a>
            </li>
        </ul>
    </div>
    <!-- 페이징 처리 끝 -->
    <a th:href="@{/question/create}" class="btn btn-primary">질문 등록하기</a>
</div>
</html>
```

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 추가하기

1. 상당히 많은 양의 HTML 코드가 추가되었지만 어렵지 않으니 찬찬히 살펴보자.

   페이지 리스트를 보기 좋게 표시하기 위해
   부트스트랩의 pagination 컴포넌트를 이용함.

   이 템플릿에 사용한 pagination, page-item, page-link 등이
   pagination 컴포넌트의 클래스로,

   pagination은 ul 요소 안에 있는 내용을 꾸밀 수 있고,
   page-item은 각 페이지 번호나 '이전', '다음' 버튼을 나타내도록 하고,
   page-link는 '이전', '다음' 버튼에 링크를 나타냄

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 추가하기

1. 이전 페이지가 없는 경우에는 '이전' 링크가 비활성화(disabled)되도록 함.
'다음' 링크의 경우도 마찬가지 방법으로 적용함.

   그리고 th:each 속성을 사용해 전체 페이지 수만큼 반복하면서
   해당 페이지로 이동할 수 있는 '이전', '다음' 링크를 생성함.

   이때 반복하던 도중 요청 페이지가 현재 페이지와 같을 경우에는
   active 클래스를 적용하여 페이지 링크에 파란색 배경이 나타나도록 함

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 추가하기

1. 위 템플릿에 사용한 주요 페이징 기능을 표로 정리해 봄

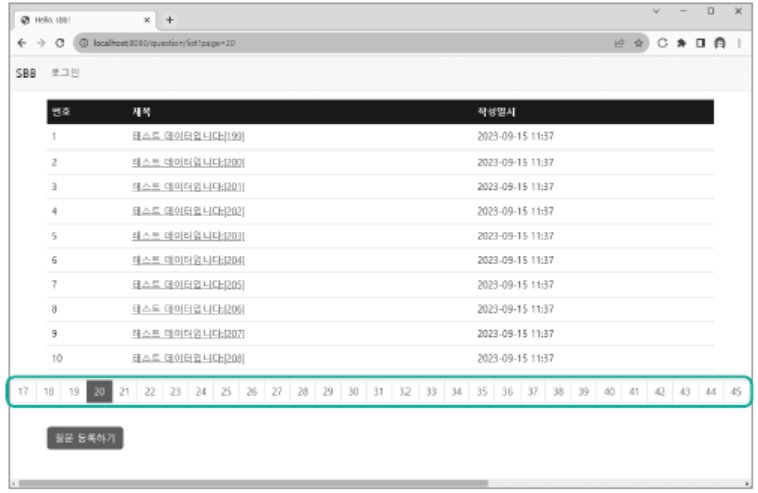
페이징 기능 관련 주요 코드	설명
th:classappend="${!paging.hasPrevious} ? 'disabled'"	이전 페이지가 없으면 '이전' 링크를 비활성화한다.
th:classappend="${!paging.hasNext} ? 'disabled'"	다음 페이지가 없으면 '다음' 링크를 비활성화한다.
th:href="@{|?page=${paging.number-1}|}"	이전 페이지 링크를 생성한다. .
th:href="@{|?page=${paging.number+1}|}"	다음 페이지 링크를 생성한다.
th:each="page: ${#numbers.sequence(0, paging.totalPages-1)}"	0부터 전체 페이지 수 만큼 이 요소를 반복하여 생성한다. 이때 현재 순번을 page 변수에 대입한다.
th:classappend="${page == paging.number} ? 'active'"	반복 구간 내에서 해당 페이지가 현재 페이지와 같은 경우 active 클래스를 적용한다.

한 가지 더 설명하면, #numbers.sequence(시작 번호, 끝 번호)는
시작 번호부터 끝 번호까지 정해진 범위만큼 반복을 만들어 내는 타임리프의 기능임

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 추가하기

2. 여기까지 수정한 후, 다시 질문 목록 URL을 조회해 보자.
   페이지 이동 기능은 구현했지만 화면에서 보듯이
   이동할 수 있는 페이지가 모두 표시되는 문제가 발생함



3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 완성하기

1. 앞서 발생한 문제를 해결하기 위해 다음과 같이 질문 목록 템플릿에 코드를 추가해 보자

• /templates/question_list.html

```
(... 생략 ...)
    <!-- 페이징 처리 시작 -->
    <div th:if="${!paging.isEmpty()}">
        <ul class="pagination justify-content-center">
            <li class="page-item" th:classappend="${!paging.hasPrevious} ? 'disabled'">
                <a class="page-link" th:href="@{|?page=${paging.number-1}|}">
                    <span>이전</span>
                </a>
            </li>
            <li th:each="page: ${#numbers.sequence(0, paging.totalPages-1)}"
            th:if="${page >= paging.number-5 and page <= paging.number+5}"
            th:classappend="${page == paging.number} ? 'active'" class="page-item">
                <a th:text="${page}" class="page-link"
                th:href="@{|?page=${page}|}"></a>
            </li>
            <li class="page-item" th:classappend="${!paging.hasNext} ? 'disabled'">
                <a class="page-link" th:href="@{|?page=${paging.number+1}|}">
                    <span>다음</span>
                </a>
            </li>
        </ul>
    </div>
    <!-- 페이징 처리 끝 -->
    <a th:href="@{/question/create}" class="btn btn-primary">질문 등록하기</a>
</div>
</html>
```

페이지 표시 제한 기능을 구현한다.

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 완성하기

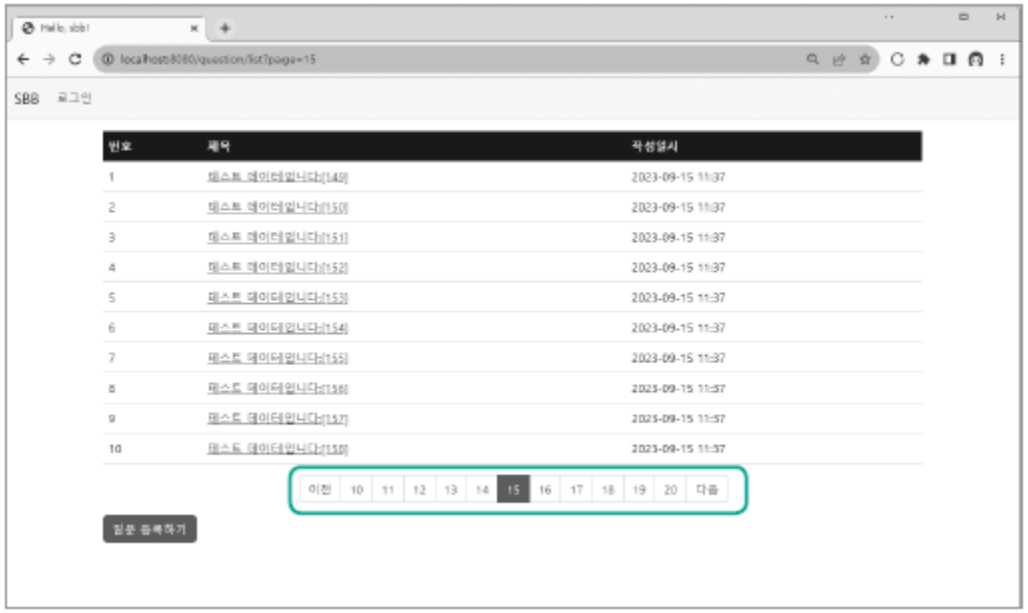
1. 이와 같이 한 줄의 코드를 삽입하여 페이지 표시 제한 기능을 구현함.
   이 코드는 현재 페이지 기준으로 좌우 5개씩 페이지 번호가 표시되도록 만듦.

   즉, 반복문 내에서 표시되는 페이지가 현재 페이지를 의미하는
   paging.number보다 5만큼 작거나 큰 경우에만 표시되도록 한 것

3-02 페이징 기능 추가하기

▪ 페이지 이동 기능 완성하기

2. 만약 현재 페이지가 15페이지라면 다음과 같이 페이지 번호가 표시될 것임



이와 같이 15페이지보다 5만큼 작은 10페이지부터 5만큼 큰 20페이지까지만 표시됨

3-02 페이징 기능 추가하기

▪ 최신순으로 데이터 조회하기

1. 현재 질문 목록은 등록한 순서대로 데이터가 표시됨.
   하지만 대부분의 게시판 서비스는 최근에 작성한 게시물이
   가장 위에 보이는 것이 일반적임.
   이를 구현하기 위해 QuestionService를 다음과 같이 수정해 보자

• /question/QuestionService.java

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

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

(... 생략 ...)
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
```

```
(... 생략 ...)
public class QuestionService {
    private final QuestionRepository questionRepository;
    public Page<Question> getList(int page) {
        List<Sort.Order> sorts = new ArrayList<>();
        sorts.add(Sort.Order.desc("createDate"));
        Pageable pageable = PageRequest.of(page, 10, Sort.by(sorts));
        return this.questionRepository.findAll(pageable);
    }

    (... 생략 ...)
}
```

3-02 페이징 기능 추가하기

▪ 최신순으로 데이터 조회하기

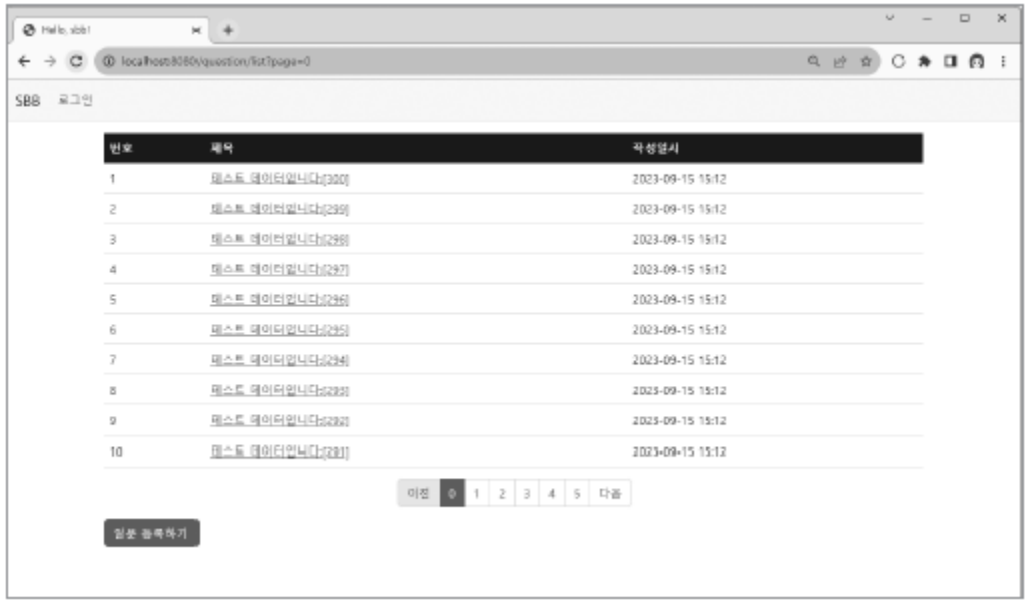
1. 게시물을 역순(최신순)으로 조회하려면 이와 같이 PageRequest.of 메서드의 세 번째 매개변수에 Sort 객체를 전달해야 함.

   작성 일시(createDate)를 역순(Desc)으로 조회하려면
   Sort.Order.desc("createDate")와 같이 작성함

3-02 페이징 기능 추가하기

▪ 최신순으로 데이터 조회하기

2. 수정한 뒤, 첫 번째 페이지를 조회하면 가장 최근에 등록한 순서대로 게시물이 출력되는 것을 확인할 수 있음



3-03 게시물에 번호 지정하기

- 현재 질문 목록 화면을 보면 어느 페이지에서나 게시물 번호가 1부터 시작해 10까지만 표시됨
- 각 게시물에 맞게 번호가 제대로 표시되도록 문제를 해결해 보자

3-03 게시물에 번호 지정하기

▪ 게시물 번호 공식 만들기

- 만약 질문 게시물이 12개라면 1페이지에는 가장 최근 게시물인 12번째~3번째 게시물이,
  2페이지에는 2번째~1번째 게시물이 역순으로 표시되어야 함
- 질문 게시물의 번호를 역순으로 정렬하려면 다음 공식을 적용해야 함

```
게시물 번호 = 전체 게시물 개수 - (현재 페이지 * 페이지당 게시물 개수) - 나열 인덱스
```

항목	설명
게시물 번호	최종 표시될 게시물의 번호
전체 게시물 개수	데이터베이스에 저장된 게시물 전체 개수
현재 페이지	페이징에서 현재 선택한 페이지
페이지당 게시물 개수	한 페이지당 보여 줄 게시물의 개수
나열 인덱스	for 문 안의 게시물 순서(나열 인덱스는 현재 페이지에서 표시할 수 있는 게시물의 인덱스이므로, 예를 들어 10개를 표시하는 페이지에서는 0~9, 2개를 표시하는 페이지에서는 0~1로 반복된다.)

3-03 게시물에 번호 지정하기

▪ 게시물 번호 공식 만들기

- 공식이 조금 복잡하니 질문 게시물이 12개인 상황을 예로 들어 설명해 보자
- 현재 페이지가 0이면 게시물의 번호는 전체 게시물 개수 12에서
  나열 인덱스 0~9를 뺀 12~3이 됨
- 현재 페이지가 1이면 페이지당 노출되는 게시물 개수는 10이므로
  12에서 10을 뺀 값인 2에 나열 인덱스 0~1을 다시 빼므로 게시물 번호는 2~1이 됨

3-03 게시물에 번호 지정하기

▪ 게시물 번호 공식 적용하기

1. 이제 게시물 번호 공식을 다음과 같이 질문 목록 템플릿에 적용해 보자.
   다음 코드의 1번째 td 요소에 이 공식을 그대로 적용함

• /templates/question_list.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <table class="table">
        <thead class="table-dark">
            (... 생략 ...)
        </thead>
        <tbody>
            <tr th:each="question, loop : ${paging}">
                <td th:text="${paging.getTotalElements - (paging.number * paging.
size) - loop.index}"></td>
                <td>
                    <a th:href="@{|/question/detail/${question.id}|}"
                    th:text="${question.subject}"></a>
                </td>
                <td th:text="${#temporals.format(question.createDate, 'yyyy-MM-dd
HH:mm')}"></td>
            </tr>
        </tbody>
    </table>
    <!-- 페이징 처리 시작 -->
    (... 생략 ...)
<!-- 페이징 처리 끝-->
<a th:href="@{/question/create}" class="btn btn-primary">질문 등록하기</a>
</div>
</html>
```

3-03 게시물에 번호 지정하기

▪ 게시물 번호 공식 적용하기

1. paging.getTotalElement는 전체 게시물 개수를 말함.
   paging.number는 현재 페이지 번호로 페이지를 변경할 때마다 달라짐.
   paging.size는 페이지당 게시물 개수로 여기서는 10으로 정해져 있음.
   loop.index는 나열 인덱스로 0부터 시작함.

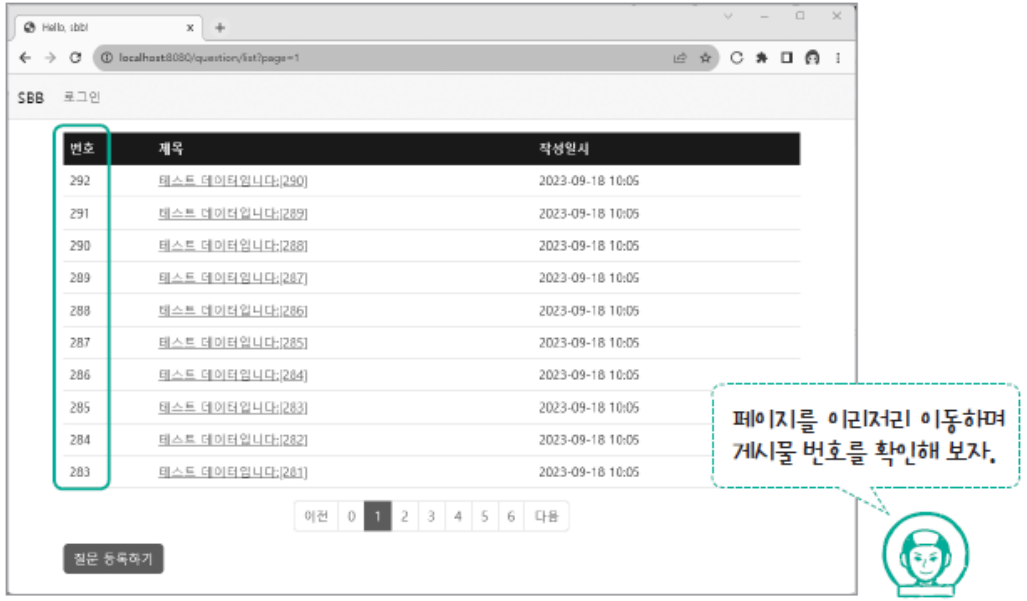
   다음 표는 템플릿에 사용한 공식의 상세 정보를 정리한 것임

항목	설명
paging.getTotalElements	전체 게시물 개수를 의미한다.
paging.number	현재 페이지 번호를 의미한다.
paging.size	페이지당 게시물 개수를 의미한다.
loop.index	나열 인덱스를 의미한다(0부터 시작).

3-03 게시물에 번호 지정하기

▪ 게시물 번호 공식 적용하기

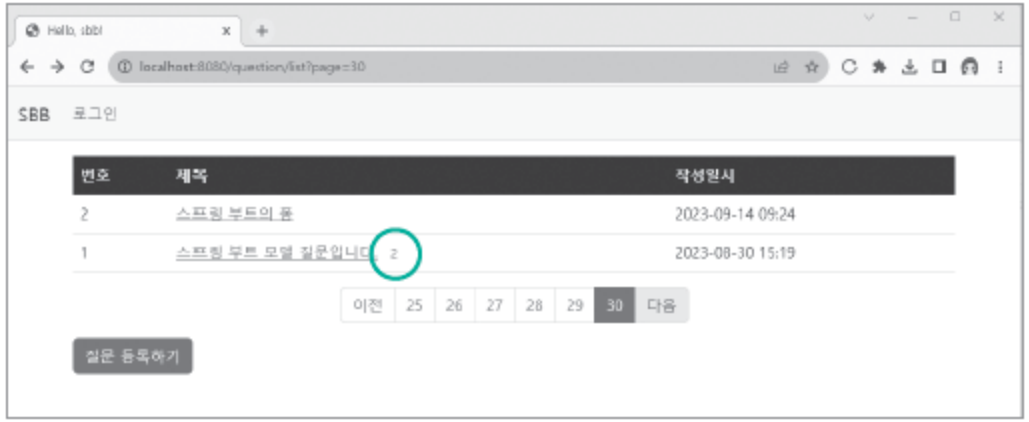
2. 이제 게시물 번호가 우리가 의도한 대로 출력됨



3-04 답변 개수 표시하기

- 이번에는 질문 목록 화면에서 해당 질문에 달린 답변 개수를 표시할 수 있는 기능을 추가해 보자

- 코드의 분량은 많지 않지만, 게시판 서비스를 사용자 입장에서 더욱 편리하게 만들어 주는 기능임



3-04 답변 개수 표시하기

1. 답변 개수는 앞서 본 화면과 같이
게시물 제목 바로 오른쪽에 표시하도록 만들어 보자

• /templates/question_list.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <table class="table">
    (... 생략 ...)
        <tbody>
            <tr th:each="question, loop : ${paging}">
                <td th:text="${paging.getTotalElements - (paging.number * paging.
size) - loop.index}"></td>
                <td>
                    <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 th:text="${#temporals.format(question.createDate, 'yyyy-MM-dd
HH:mm')}"></d>
            </tr>
        </tbody>
    </table>
    (... 생략 ...)
```

th:if="${#lists.size(question.answerList) > 0}"로 답변이 있는지 조사하고,
th:text="${#lists.size(question.answerList)}"로 답변 개수를 표시함

3-04 답변 개수 표시하기

2. 이제 답변이 있는 질문은 다음과 같이 제목 오른쪽에
빨간색(text-danger) 숫자가 작게(small) 왼쪽 여백(ms-2)이 추가되어 표시됨

3-05 스프링 시큐리티란?

- 스프링 부트는 회원 가입과 로그인을 도와주는
  스프링 시큐리티(Spring Security)를 사용할 수 있음

- SBB도 스프링 시큐리티를 사용하여 회원 가입과 로그인 기능을 만들 것임.
  SBB에 회원 가입과 로그인 기능을 추가하기 전에
  먼저 스프링 시큐리티를 간단하게 알아보고 설치와 설정도 진행해 보자

- 스프링 시큐리티는 스프링 기반 웹 애플리케이션의 인증과 권한을 담당하는
  스프링의 하위프레임워크임

- 여기서 인증(authenticate)은 로그인과 같은 사용자의 신원을 확인하는 프로세스를,
  권한(authorize)은 인증된 사용자가 어떤 일을 할 수 있는지(어떤 접근 권한이 있는지)
  관리하는 것을 의미함

3-05 스프링 시큐리티란?

▪ 스프링 시큐리티 설치하기

- 스프링 시큐리티를 사용하기 위해 다음과 같이 build.gradle 파일을 수정해 보자

• build.gradle

```
(... 생략 ...)

dependencies {
    (... 생략 ...)
    implementation 'org.springframework.boot:spring-boot-starter-validation'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity6'
}

(... 생략 ...)
```

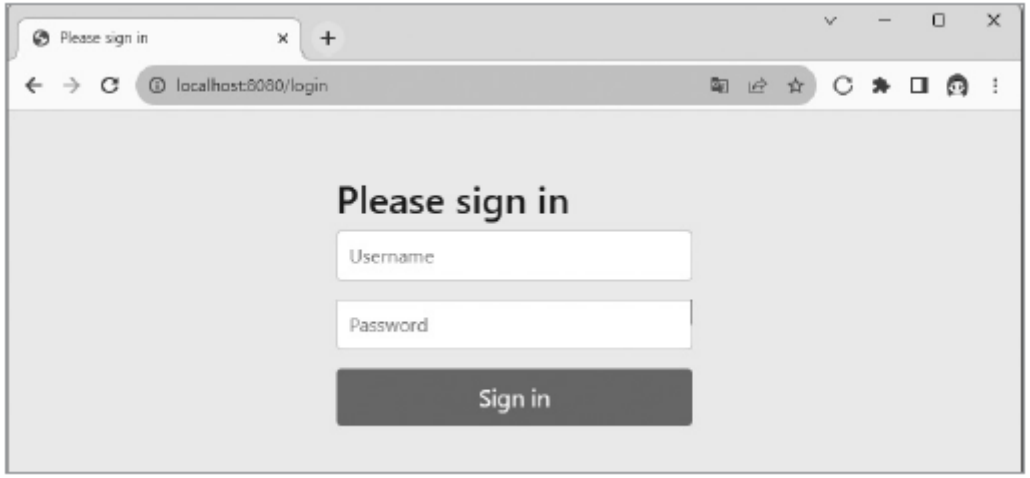
3-05 스프링 시큐리티란?

- **스프링 시큐리티 설치하기**
  - 스프링 시큐리티와 이와 관련된 타임리프 라이브러리를 사용하도록 설정함
  - build.gradle 파일을 선택한 후 마우스 오른쪽 버튼을 눌러 [Gradle → Refresh Gradle Project]를 클릭하여 변경 사항을 적용하면 해당 라이브러리가 설치됨
  - 로컬 서버도 한번 재시작하자

3-05 스프링 시큐리티란?

▪ 스프링 시큐리티 설정하기

1. 스프링 시큐리티를 설치하고 로컬 서버를 재시작한 후에 SBB의 질문 목록 화면에 접속해보자



3-05 스프링 시큐리티란?

▪ 스프링 시큐리티 설정하기

2. 스프링 시큐리티는 기본적으로 인증되지 않은 사용자가
SBB와 같은 웹 서비스를 사용할 수 없게끔 만듦.

   따라서 이와 같이 인증을 위한 로그인 화면이 나타나는 것임.

   이러한 스프링 시큐리티의 기본 기능을 SBB에 그대로 적용되면 곤란하므로
설정을 통해 바로잡아야 함.

   SBB는 로그인하지 않아도 게시물을 조회할 수 있어야 하기 때문임

3-05 스프링 시큐리티란?

▪ 스프링 시큐리티 설정하기

2. 다음과 같이 com.mysite.sbb 패키지에 스프링 시큐리티의 설정을 담당할 SecurityConfig.java 파일을 작성해 보자

• SecurityConfig.java

```
package com.mysite.sbb;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.
EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
```

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((authorizeHttpRequests) -> authorizeHttpRequests
                .requestMatchers(new AntPathRequestMatcher("/**")).permitAll())
            ;
        return http.build();
    }
}
```

인증되지 않은 모든 페이지의 요청을 허락한다는 의미이다. 따라서 로그인하지 않더라도 모든 페이지에 접근할 수 있도록 한다.

3-05 스프링 시큐리티란?

- **스프링 시큐리티 설정하기**

2. @Configuration은 이 파일이 스프링의 환경 설정 파일임을 의미하는 애너테이션.
   여기서는 스프링 시큐리티를 설정하기 위해 사용함.

   @EnableWebSecurity는 모든 요청 URL이 스프링 시큐리티의
   제어를 받도록 만드는 애너테이션.

   이 애너테이션을 사용하면 스프링 시큐리티를 활성화하는 역할을 함.

3-05 스프링 시큐리티란?

▪ 스프링 시큐리티 설정하기

2. 내부적으로 SecurityFilterChain 클래스가 동작하여
모든 요청 URL에 이 클래스가 필터로 적용되어
URL별로 특별한 설정을 할 수 있게 됨.

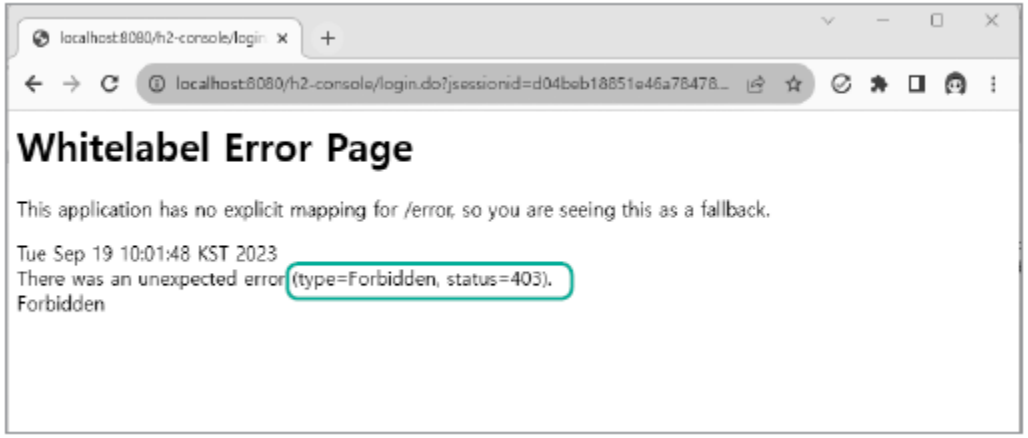
   스프링 시큐리티의 세부 설정은 @Bean 애너테이션을 통해
SecurityFilterChain 빈을 생성하여 설정할 수 있음.

   이렇게 스프링 시큐리티 설정 파일을 구성하면
이제 질문 목록, 질문 답변 등의 기능을 이전과 동일하게 사용할 수 있음

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

- 그런데 스프링 시큐리티를 적용하면 H2 콘솔 로그인 시 다음과 같은 403 Forbidden 오류가 발생함
- 403 Forbidden은 작동 중인 서버에 클라이언트의 요청이 들어왔으나, 서버가 클라이언트의 접근을 거부했을 때 반환하는 HTTP 오류 코드임
- 이 오류는 서버 또는 서버에 있는 파일 등에 접근 권한이 없을 경우에 발생함



3-05 스프링 시큐리티란?

H2 콘솔 오류 수정하기

- 403 Forbidden 오류가 발생하는 이유를 좀 더 구체적으로 설명하면,
  스프링 시큐리티의 CSRF 방어 기능에 의해 H2 콘솔 접근이 거부되기 때문

- CSRF는 웹 보안 공격 중 하나로, 조작된 정보로 웹 사이트가 실행되도록 속이는 공격 기술

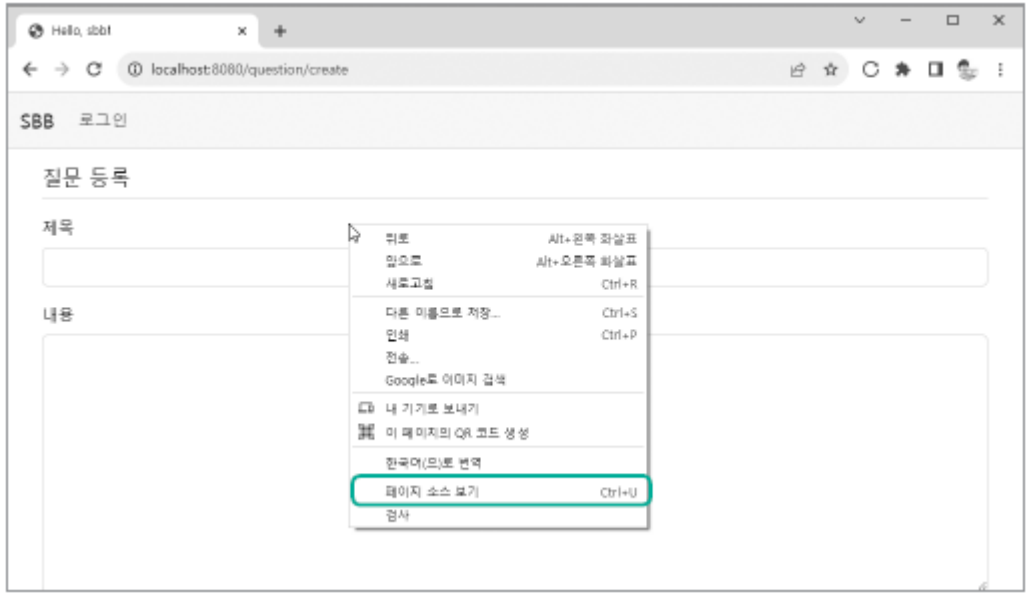
- 스프링 시큐리티는 이러한 공격을 방지하기 위해
  CSRF 토큰을 세션을 통해 발행하고,

  웹 페이지에서는 폼 전송 시에 해당 토큰을 함께 전송하여
  실제 웹 페이지에서 작성한 데이터가 전달되는지를 검증함

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

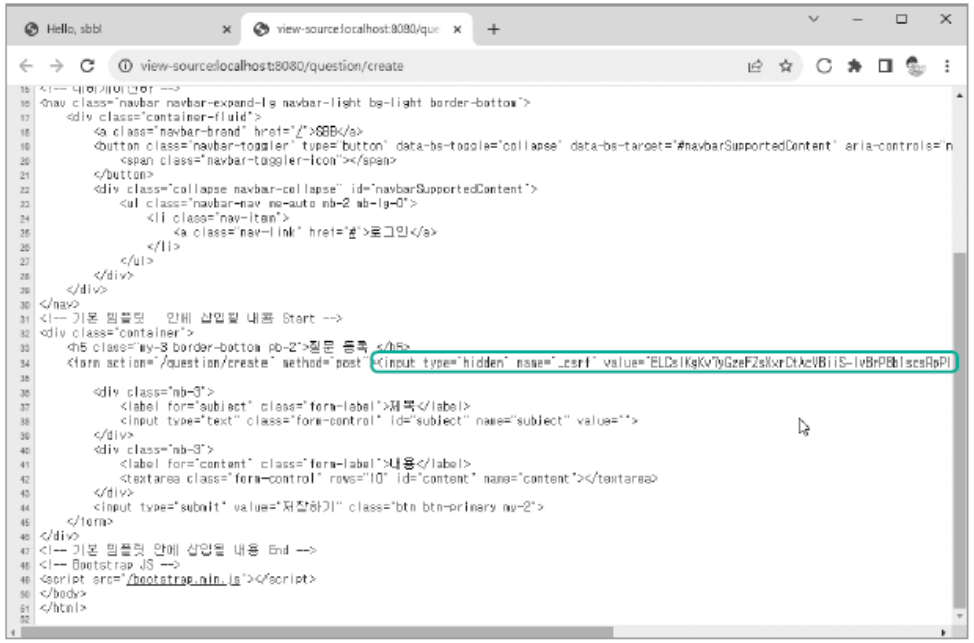
1. 이 오류를 해결하기 전에 다음과 같이 질문 등록 화면을 열고
   브라우저의 '페이지 소스 보기' 기능을 이용하여
   질문 등록 화면의 소스를 잠시 확인해 보자



3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

2. 그러면 다음과 같이 질문 등록 화면의 소스를 볼 수 있음



3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

2. 다음과 같은 input 요소가 <form> 태그 안에 자동으로 생성된 것을 확인할 수 있음.

```
<input type="hidden" name="_csrf" value="ELCsIKgKv7yGzeFZsXxrCtAcVBiiS-lvBrP8b-
1scsRpPlrWHJoKfFsw4ioyr-thtgFFfbLF6eSCUctFCYICYW2l4gC57o4W1"/>
```

스프링 시큐리티에 의해 이와 같은 CSRF 토큰이 자동으로 생성됨

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

2. 스프링 시큐리티는 이런 식으로 페이지에 CSRF 토큰을 발행하여
이 값이 다시 서버로 정확하게 들어오는지를 확인하는 과정을 거침.

   만약 CSRF 토큰이 없거나 해커가 임의의 CSRF 토큰을 강제로 만들어 전송한다면
스프링 시큐리티에 의해 차단될 것임.

   정리하자면, H2 콘솔은 스프링 프레임워크가 아니므로
CSRF 토큰을 발행하는 기능이 없어 이와 같은 403 오류가 발생한 것

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

3. 스프링 시큐리티가 CSRF 처리 시 H2 콘솔은 예외로 처리할 수 있도록 다음과 같이 설정 파일을 수정하자.

• SecurityConfig.java

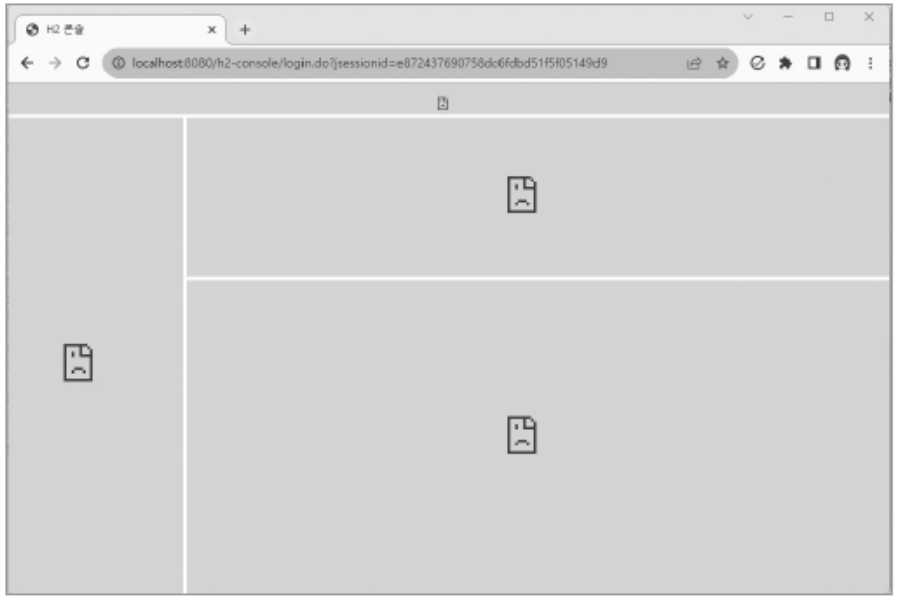
```
(... 생략 ...)

@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/**")))
            ;
        return http.build();
    }
}
```

/h2-console/로 시작하는 모든 URL은 CSRF 검증을 하지 않는다는 설정을 추가함.
이렇게 수정하고 로컬 서버를 재시작한 후 다시 H2 콘솔에 접속해 보자

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

4. 이제 CSRF 검증에서 예외 처리되어 로그인은 잘 수행됨.
   하지만 다음과 같이 화면이 깨져 보임



3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

4. 이와 같은 오류가 발생하는 원인은 H2 콘솔의 화면이
프레임(frame) 구조로 작성되었기 때문.
즉, H2 콘솔 UI(user interface) 레이아웃이 이 화면처럼 작업 영역이 나눠져 있음을
의미함.

   스프링 시큐리티는 웹 사이트의 콘텐츠가 다른 사이트에 포함되지 않도록 하기 위해
X-Frame-Options 헤더의 기본값을 DENY로 사용하는데,
프레임 구조의 웹 사이트는 이 헤더의 값이 DENY인 경우 이와 같이 오류가 발생함

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

5. 이 문제를 해결하기 위해 다음과 같이 설정 파일을 수정하자

• SecurityConfig.java

```
(... 생략 ...)
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.security.web.header.writers.frameoptions.XFrameOption
sHeaderWriter;

@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)))
            ;
        return http.build();
    }
}
```

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

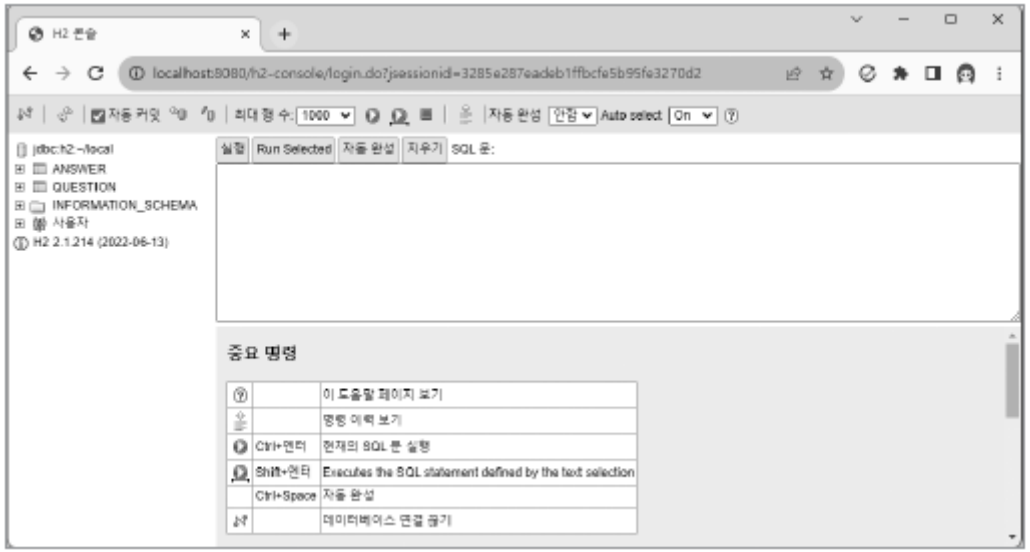
5. 이와 같이 URL 요청 시 X-Frame-Options 헤더를 DENY 대신 SAMEORIGIN으로 설정하여 오류가 발생하지 않도록 함.

   X-Frame-Options 헤더의 값으로 SAMEORIGIN을 설정하면
   프레임에 포함된 웹 페이지가 동일한 사이트에서 제공할 때에만 사용이 허락됨

3-05 스프링 시큐리티란?

▪ H2 콘솔 오류 수정하기

6. 이제 다시 H2 콘솔로 로그인하면 우리에게 익숙한 화면이 등장함.
   즉, 정상으로 동작하는 것을 확인할 수 있음



3-05 스프링 시큐리티란?

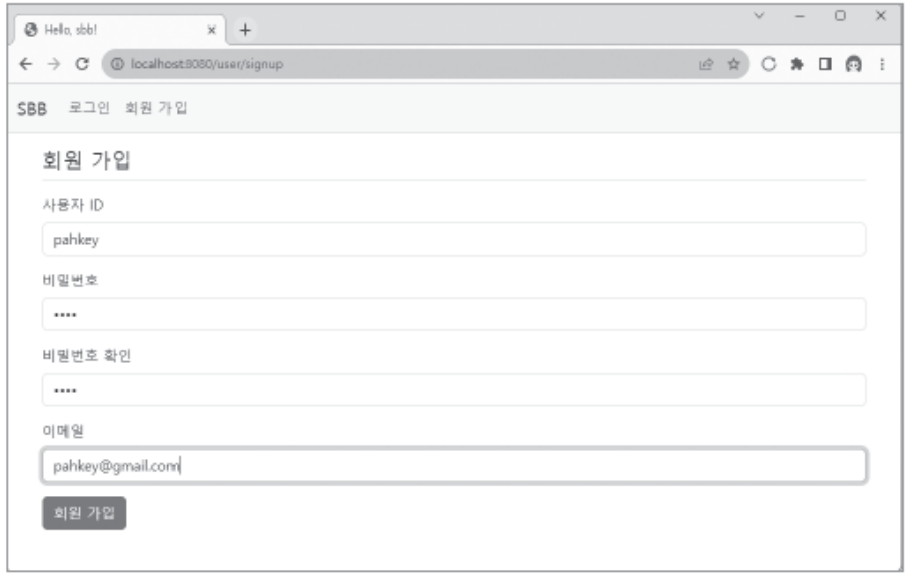
▪ H2 콘솔 오류 수정하기

6. 스프링 시큐리티를 사용하면 웹 프로그램(애플리케이션)의 보안을 강화하고
사용자 인증 및 권한 부여를 효과적으로 관리할 수 있으며,
외부 공격으로부터 시스템을 보호하는 데 도움을 얻을 수 있음.

이러한 스프링 시큐리티의 설치와 설정을 마쳤으니
SBB에 회원 가입과 로그인 기능을 추가해 보자

3-06 회원 가입 기능 구현하기

- 먼저 SBB에 사용자가 회원 가입할 수 있는 화면을 만들고 회원 가입 기능을 완성해 보자
- 회원 가입 기능은 웹 사이트의 중요 기능이고 이를 구현하는 것은 웹 프로그래밍의 핵심임



3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

회원 가입 기능을 구현하려면 회원 정보와 관련된 데이터를 저장하고
이를 관리하는 엔티티와 리포지터리 등을 만들어야 하고,
폼과 컨트롤러와 같은 요소를 생성해 사용자로부터 입력받은 데이터를
웹 프로그램에서 사용할 수 있도록 만들어야 함

▪ 회원 엔티티 생성하기

지금까지는 질문, 답변 엔티티만 사용했다면
이제 회원 정보와 관련된 데이터를 저장하는 엔티티가 필요함.
즉, 회원 엔티티를 구상해야 함

3-06 회원 가입 기능 구현하기

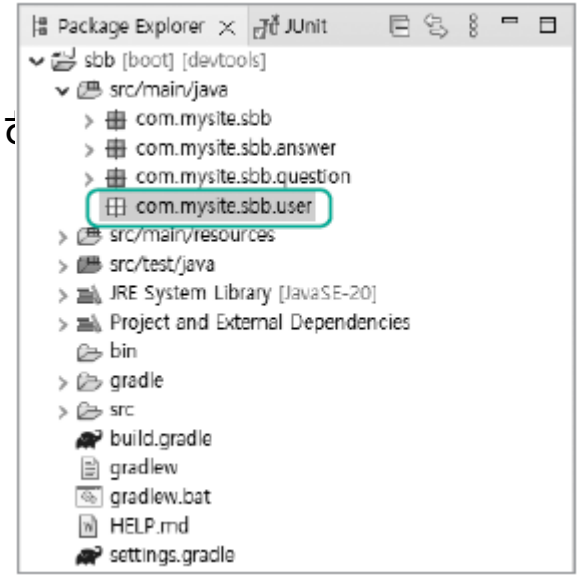
▪ 회원 가입 기능 구성하기

▪ 회원 엔티티 생성하기

1. 회원 엔티티에는 최소한 다음 속성이 필요함

속성 이름	설명
username	사용자 이름(또는 사용자 ID)
password	비밀번호
email	이메일

1. 회원 관련 자바 파일은 질문, 답변 도메인에 포함하기는
어색하므로 다음과 같이 com.mysite.sbb.user 패키지를 생성
사용자(User) 도메인을 새로 만들어 보자



3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 엔티티 생성하기

3. 새로 생성한 com.mysite.sbb.user 패키지에 SiteUser.java 파일을 만들어 회원 정보 데이터를 저장할 회원 엔티티를 다음과 같이 작성해 보자

• /user/SiteUser.java

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

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
public class SiteUser {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true)
    private String username;

    private String password;

    @Column(unique = true)
    private String email;
}
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 엔티티 생성하기

3. 질문(Question)과 답변(Answer) 엔티티를 만든 것과 동일한 방법으로
회원 엔티티(SiteUser 엔티티)를 만들었음

그리고 username, email 속성에는 @Column(unique = true)으로 지정함.
여기서 unique = true는 유일한 값만 저장할 수 있음을 의미함.

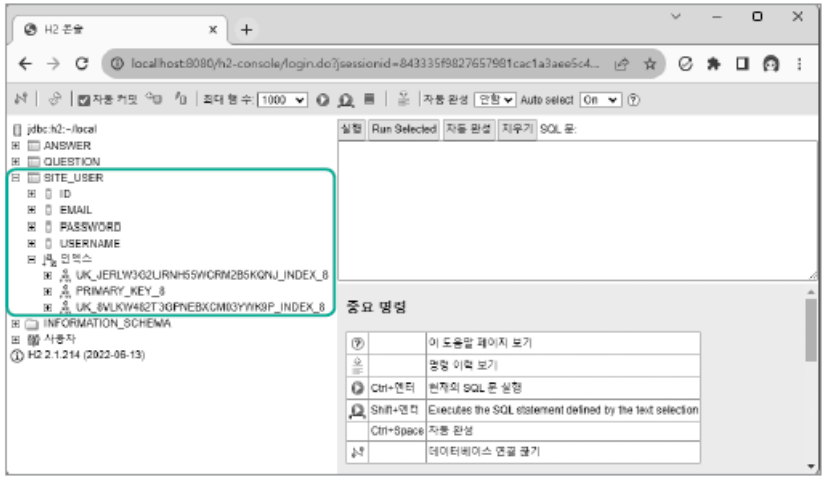
즉, 값을 중복되게 저장할 수 없음을 말함.
이렇게 해야 username과 email에 동일한 값이 저장되는 것을 막을 수 있음

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 엔티티 생성하기

4. SiteUser 엔티티를 생성하였으므로 로컬 서버를 재시작한 후, H2 콘솔에 접속하여 테이블이 잘 만들어졌는지 확인해 보자



3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 엔티티 생성하기

4. SITE_USER 테이블과 데이터 열들 그리고 unique로 설정한 속성들로 인해 생긴 UK_로 시작하는 인덱스들을 확인할 수 있음

▪ User 리포지터리와 서비스 생성하기

SiteUser 엔티티가 준비되었으니
이제 User 리포지터리와 User 서비스를 만들어 보자

1. 다음과 같이 UserRepository를 만들어 보자
   리포지터리는 인터페이스임.

   SiteUser의 기본키 타입은 Long이므로
   JpaRepository<SiteUser, Long>으로 사용한

• /user/UserRepository.java

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

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

public interface UserRepository extends JpaRepository<SiteUser, Long> {
}
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ User 리포지터리와 서비스 생성하기

2. 이번에는 UserService.java 파일을 생성하여 서비스를 활용하기 위해 다음과 같은 내용을 작성하자

• UserService.java

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import lombok.RequiredArgsConstructor;

@RequiredArgsConstructor
@Service
public class UserService {
    private final UserRepository userRepository;

    public SiteUser create(String username, String email, String password) {
        SiteUser user = new SiteUser();
        user.setUsername(username);
        user.setEmail(email);
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        user.setPassword(passwordEncoder.encode(password));
        this.userRepository.save(user);
        return user;
    }
}
```

비밀번호를 암호화하기 위해 필요하다.

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ User 리포지터리와 서비스 생성하기

2. User 서비스에는 User 리포지터리를 사용하여
   회원(User) 데이터를 생성하는 create 메서드를 추가함.

   이때 User의 비밀번호는 보안을 위해 반드시 암호화하여 저장해야 함.

   그러므로 스프링 시큐리티의 BCryptPasswordEncoder 클래스를 사용하여
   암호화하여 비밀번호를 저장함

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ User 리포지터리와 서비스 생성하기

3. 하지만 이렇게 BCryptPasswordEncoder 객체를 직접 new로 생성하는 방식보다는 PasswordEncoder 객체를 빈으로 등록해서 사용하는 것이 좋음.

   왜냐하면 암호화 방식을 변경하면 BCryptPasswordEncoder를 사용한 모든 프로그램을 일일이 찾아다니며 수정해야 하기 때문.

   PasswordEncoder 빈을 만드는 가장 쉬운 방법은 @Configuration이 적용된 Security Config.java 파일에 @Bean 메서드를 새로 추가하는 것임.

   다음과 같이 SecurityConfig.java 파일을 수정하자

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ User 리포지터리와 서비스 생성하기

3.

• /sbb/SecurityConfig.java

```
(... 생략 ...)
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@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)))
        ;
        return http.build();
    }

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

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ User 리포지터리와 서비스 생성하기

4. PasswordEncoder를 @Bean으로 등록하면 UserService.java도 다음과 같이 수정할 수 있음.

• /user/UserService.java

```
package com.mysite.sbb.user;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import lombok.RequiredArgsConstructor;
@RequiredArgsConstructor
@Service public class UserService {

    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
```

```
    public SiteUser create(String username, String email, String password) {
        SiteUser user = new SiteUser();
        user.setUsername(username);
        user.setEmail(email);
        user.setPassword(passwordEncoder.encode(password));
        this.userRepository.save(user);
        return user;
    }
}
```

BcryptPasswordEncoder 객체를 직접 생성하여 사용하지 않고
빈으로 등록한 Password Encoder 객체를 주입받아 사용할 수 있도록 수정함

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 가입 폼 생성하기

- 이번에는 com.mysite.sbb.user 패키지에 회원 가입을 위한 폼 클래스인 UserCreateForm을 다음과 같이 만들어 보자

• /user/UserCreateForm.java

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

import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.Size;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class UserCreateForm {
    @Size(min = 3, max = 25)
    @NotEmpty(message = "사용자 ID는 필수 항목입니다.")
    private String username;

    @NotEmpty(message = "비밀번호는 필수 항목입니다.")
    private String password1;

    @NotEmpty(message = "비밀번호 확인은 필수 항목입니다.")
    private String password2;

    @NotEmpty(message = "이메일은 필수 항목입니다.")
    @Email
    private String email;
}
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 가입 폼 생성하기

- username은 입력받는 데이터의 길이가 3~25 사이여야 한다는 검증 조건을 설정함
- @Size는 문자열의 길이가 최소 길이(min)와 최대 길이(max) 사이에 해당하는지를 검증함
- password1과 password2는 '비밀번호'와 '비밀번호 확인'에 대한 속성임

- 로그인할 때는 비밀번호가 한 번만 필요하지만
  회원 가입 시에는 입력한 비밀번호가 정확한지 확인하기 위해
  2개의 필드가 필요하므로 이와 같이 작성함

- email 속성에는 @Email 애너테이션이 적용됨.
  @Email은 해당 속성의 값이 이메일 형식과 일치하는지를 검증함

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 가입 컨트롤러 생성하기

- 회원 가입을 위한 엔티티/서비스/폼이 준비되었으니 URL 매핑을 하기 위한 User 컨트롤러를 만들어 보자

• /user/UserController.java

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

import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;


import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
@RequiredArgsConstructor
@Controller
@RequestMapping("/user")
public class UserController {

    private final UserService userService;

    @GetMapping("/signup")
    public String signup(UserCreateForm userCreateForm) {
        return "signup_form";
    }

    @PostMapping("/signup")
    public String signup(@Valid UserCreateForm userCreateForm, BindingResult
bindingResult) {
        if (bindingResult.hasErrors()) {
            return "signup_form";
        }

        if (!userCreateForm.getPassword1().equals(userCreateForm.getPassword2()))
{
            bindingResult.rejectValue("password2", "passwordInCorrect", "2개의 비
밀번호가 일치하지 않습니다.");
            return "signup_form";
        }
        userService.create(userCreateForm.getUsername(), userCreateForm.
getEmail(), userCreateForm.getPassword1());

        return "redirect:/";
    }
}
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 구성하기

▪ 회원 가입 컨트롤러 생성하기

- /user/signup URL이 GET으로 요청되면 회원 가입을 위한 템플릿을 렌더링하고, POST로 요청되면 회원 가입을 진행하도록 함
- 그리고 회원 가입 시 password1과 password2가 동일한지를 검증하는 조건문을 추가함

- 만약 2개의 값이 서로 일치하지 않을 경우에는 bindingResult.rejectValue를 사용하여 입력받은 2개의 비밀번호가 일치하지 않는다는 오류가 발생하게 함
- bindingResult.rejectValue의 매개변수는 순서대로 각각 bindingResult.rejectValue(필드명, 오류 코드, 오류 메시지)를 의미함

- userService.create 메서드를 사용하여 사용자로부터 전달받은 데이터를 저장함

3-06 회원 가입 기능 구현하기

▪ 회원 가입 화면 구성하기

이제 회원 가입 화면을 만들어 보자.
템플릿을 생성하여 사용자가 화면을 통해 회원 가입을 할 수 있도록 해보자

▪ 회원 가입 템플릿 생성하기

- 다음과 같이 signup_form.html 파일을 작성해
  회원 가입 화면을 구성하는 템플릿을 만들어 보자
- templates에 sign_form.html 파일을 만든 후 다음 내용을 작성함

3-06 회원 가입 기능 구현하기

▪ 회원 가입 화면 구성하기

▪ 회원 가입 템플릿 생성하기

• /templates/signup_form.html

```
<html layout:decorate="~{layout}">
<div layout:fragment="content" class="container my-3">
    <div class="my-3 border-bottom">
        <div>
            <h4>회원 가입</h4>
        </div>
    </div>
    <form th:action="@{/user/signup}" th:object="${userCreateForm}" method="post">
        <div th:replace="~{form_errors :: formErrorsFragment}"></div>
        <div class="mb-3">
            <label for="username" class="form-label">사용자 ID</label>
            <input type="text" th:field="*{username}" class="form-control">
        </div>
        <div class="mb-3">
            <label for="password1" class="form-label">비밀번호</label>
            <input type="password" th:field="*{password1}" class="form-control">
        </div>
        <div class="mb-3">
            <label for="password2" class="form-label">비밀번호 확인</label>
            <input type="password" th:field="*{password2}" class="form-control">
        </div>
        <div class="mb-3">
            <label for="email" class="form-label">이메일</label>
            <input type="email" th:field="*{email}" class="form-control">
        </div>
        <button type="submit" class="btn btn-primary">회원 가입</button>
    </form>
</div>
</html>
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 화면 구성하기

▪ 회원 가입 템플릿 생성하기

- ‘사용자 ID’, ‘비밀번호’, ‘비밀번호 확인, ‘이메일’에 해당하는 input 요소들을 추가하여 회원 가입 화면에 각각의 필드가 나타나도록 함
- [회원 가입] 버튼을 누르면 <form> 데이터가 POST 방식으로 /user/signup/URL에 전송됨

▪ 내비게이션 바에 회원 가입 링크 추가하기

- 사용자가 회원 가입을 쉽게 할 수 있도록 회원 가입 화면으로 이동할 수 있는 링크를 내비게이션 바에 추가해 보자
- 그러기 위해 navbar.html에서 다음과 같은 내용을 추가해 보자

3-06 회원 가입 기능 구현하기

▪ 회원 가입 화면 구성하기

▪ 내비게이션 바에 회원 가입 링크 추가하기

• /templates/navbar.html

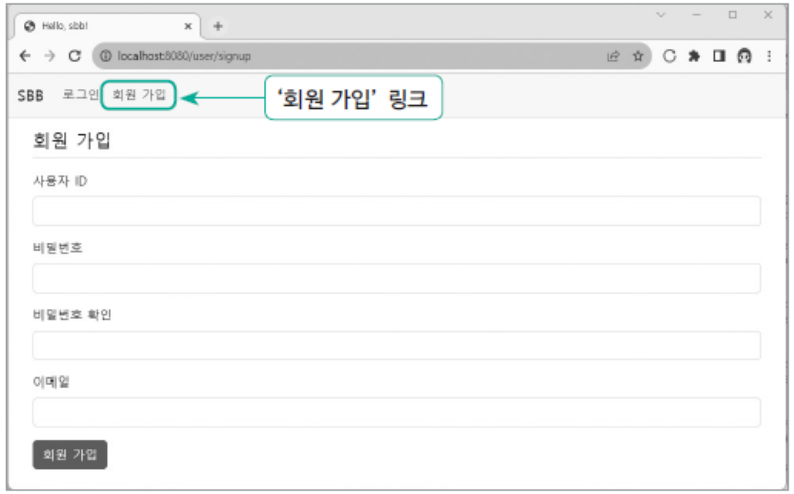
```
<nav th:fragment="navbarFragment" class="navbar navbar-expand-lg navbar-light bg-
light border-bottom">
    <div class="container-fluid">
        <a class="navbar-brand" href="/">SBB</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
        data-bs-target="#navbarSupportedContent"
        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" href="#">로그인</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" th:href="@{/user/signup}">회원 가입</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 확인하기

1. 회원 가입 기능을 확인하기 위해 로컬 서버를 재시작한 후, 브라우저에서 http://localhost:8080 URL을 요청해 보자.

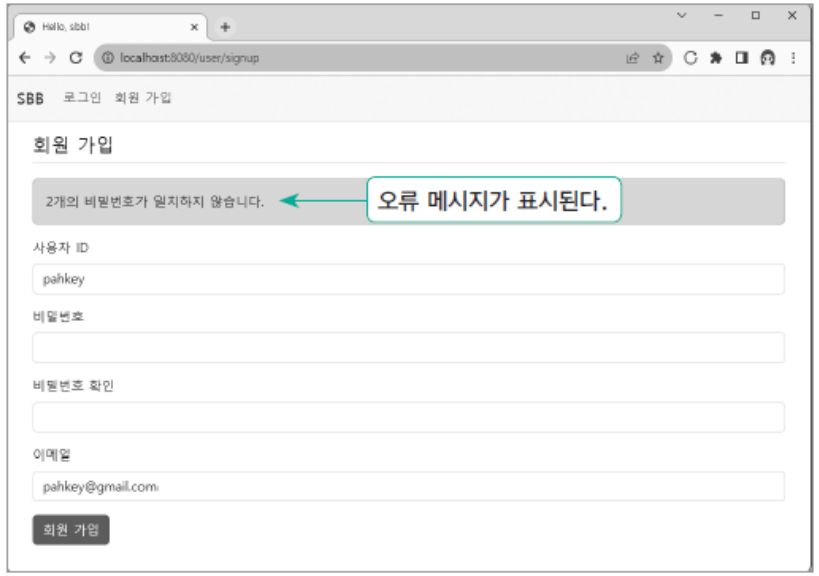
   이어서 내비게이션 바에 있는 '회원 가입' 링크를 누르면 다음과 같은 회원 가입 화면이 등장함



3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 확인하기

2. 비밀번호, 비밀번호 확인 항목을 다르게 입력하고 [회원 가입] 버튼을 누르면 검증 오류가 발생하여 화면에 다음과 같은 오류 메시지가 표시될 것임



3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 확인하기

2. 이처럼 우리가 만든 회원 가입 기능에는
각 항목에 값이 제대로 입력되었는지를 확인하는 필수 값 검증, 양식에 따라
이메일이 제대로 입력되었는지를 확인하는 이메일 규칙 검증 등이 적용되어 있음.

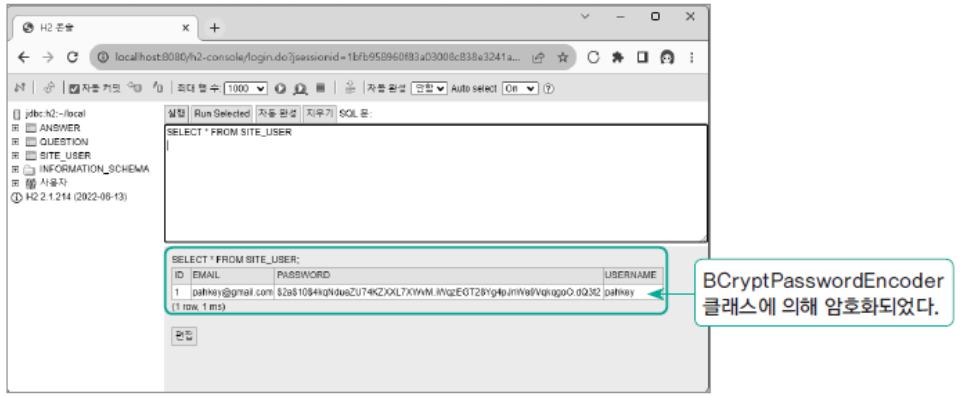
   올바른 입력 값으로 회원 가입을 완료하면 메인 페이지로 리다이렉트될 것임

3-06 회원 가입 기능 구현하기

▪ 회원 가입 기능 확인하기

3. 이번에는 브라우저에서 http://localhost:8080/h2-console URL을 요청해 보자.
H2 콘솔에서 다음과 같이 SQL을 실행하여 앞에서 만든 회원 정보를 확인해 보자.

```
SELECT * FROM SITE_USER
```

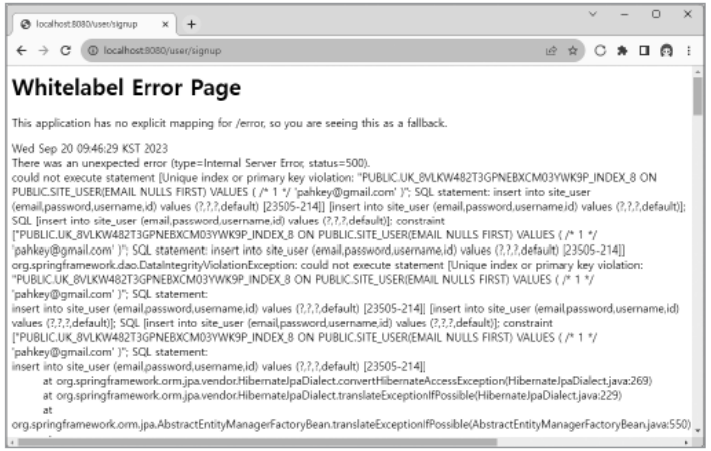


SBB에서 회원 가입을 완료하면 DB에 회원 정보가 저장되는 것을 확인할 수 있음.
즉, 이제 우리가 만든 SBB 서비스에 회원 가입 기능이 추가됨

3-06 회원 가입 기능 구현하기

▪ 중복 회원 가입 방지하기

1. 회원 가입할 때, 이미 등록된 사용자 ID 또는 이메일 주소로 회원 가입을 시도해 보자.
   아마도 다음과 같은 오류가 발생할 것임.



이미 등록된 사용자 ID 또는 이메일 주소를 DB에 저장하는 것은
회원 엔티티의 unique=true 설정으로 허용되지 않으므로 이와 같은 오류가 발생하는 것임

3-06 회원 가입 기능 구현하기

▪ 중복 회원 가입 방지하기

2. 화면에 500 오류 메시지를 그대로 보여 주는 것은 좋지 않음.

   회원 가입 화면에서 비밀번호가 일치하지 않는다는 오류 메시지를 등장시킨 것처럼,
   회원 가입 시 이미 동일한 ID와 이메일 주소가 있다는 것을 알리는
   메시지가 나타나도록 다음과 같이 UserController를 수정해 보자

3-06 회원 가입 기능 구현하기

▪ 중복 회원 가입 방지하기

2.

• /users/UserController.java

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

(... 생략 ...)
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.dao.DataIntegrityViolationException;
(... 생략 ...)

public class UserController {

    (... 생략 ...)
    @PostMapping("/signup")
    public String signup(@Valid UserCreateForm userCreateForm, BindingResult
bindingResult) {
        if (bindingResult.hasErrors()) {
            return "signup_form";
        }

        if (!userCreateForm.getPassword1().equals(userCreateForm.getPassword2()))
{
            bindingResult.rejectValue("password2", "passwordInCorrect",
                    "2개의 비밀번호가 일치하지 않습니다.");
            return "signup_form";
        }

        try {
            userService.create(userCreateForm.getUsername(), userCreateForm.
getEmail(), userCreateForm.getPassword1());
        }catch(DataIntegrityViolationException e) {
            e.printStackTrace();
            bindingResult.reject("signupFailed", "이미 등록된 사용자입니다.");
            return "signup_form";
        }catch(Exception e) {
            e.printStackTrace();
            bindingResult.reject("signupFailed", e.getMessage());
            return "signup_form";
        }

        return "redirect:/";
    }
}
```

중복된 데이터에 대한 예외 처리를 하는 코드이다.

DateIntegrityViolationException 외에 다른 예외 처리를 하는 코드이다.

3-06 회원 가입 기능 구현하기

▪ 중복 회원 가입 방지하기

2. 사용자 ID 또는 이메일 주소가 이미 존재할 경우에는
DataIntegrityViolationException라는 예외가 발생하므로
'이미 등록된 사용자입니다.'라는 오류 메시지가 화면에 표시하도록 함.

그리고 그 밖에 다른 예외들은 해당 예외에 관한
구체적인 오류 메시지를 출력하도록 e.getMessage()를 사용함.

여기서 bindingResult.reject(오류 코드, 오류 메시지)는
UserCreateForm의 검증에 의한 오류 외에 일반적인 오류를 발생시킬 때 사용함

3-06 회원 가입 기능 구현하기

▪ 중복 회원 가입 방지하기

3. 이렇게 코드를 수정하고 다시 이미 등록된 사용자 ID로 회원 가입을 시도하면 다음과 같은 오류 메시지를 표시하는 화면을 볼 수 있음