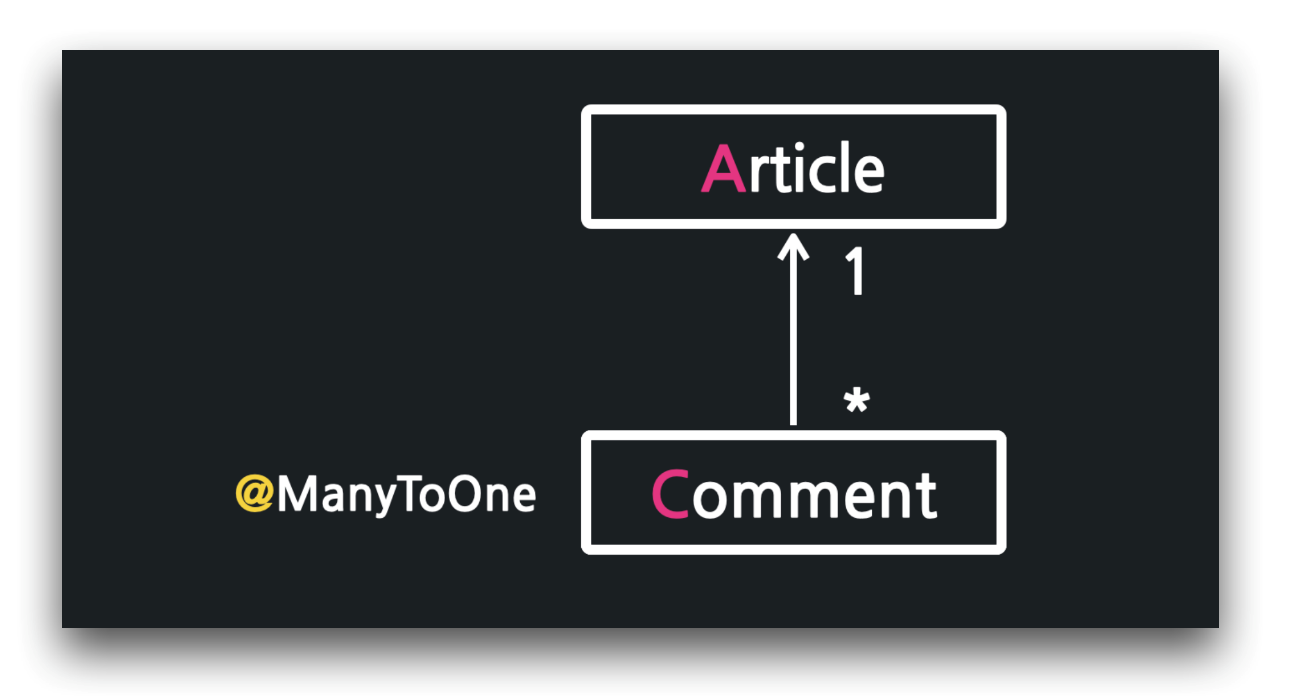
22 댓글 엔티티와 리파지터리(feat. 데이터의 관계)

**댓글 엔티티와 리파지터리(feat. 데이터의 관계)**

**미션**

댓글(Comment)을 위한 엔티티와 리파지터리를 만들고, 이를 테스트하시오.  


**02:46 댓글 엔티티 구현 - @ManyToOne, @JoinColumn**

../entity/Comment

1. package com.example.firstproject.entity;
2. import lombok.AllArgsConstructor;
3. import lombok.Getter;
4. import lombok.NoArgsConstructor;
5. import lombok.ToString;
6. import javax.persistence.\*;
7. @Entity
8. @Getter
9. @ToString
10. @AllArgsConstructor
11. @NoArgsConstructor
12. public class Comment {
13. @Id
14. @GeneratedValue(strategy = GenerationType.IDENTITY)
15. private Long id;
16. @ManyToOne
17. @JoinColumn(name = "article\_id")
18. private Article article;
19. @Column
20. private String nickname;
21. @Column
22. private String body;
23. }

**07:08 더미 데이터 추가**

../resources/data.sql

1. -- 15강: article 더미 데이터
2. INSERT INTO article(id, title, content) VALUES(1, '가가가가', '1111');
3. INSERT INTO article(id, title, content) VALUES(2, '나나나나', '2222');
4. INSERT INTO article(id, title, content) VALUES(3, '다다다다', '3333');
5. -- 22강: article 더미 데이터
6. INSERT INTO article(id, title, content) VALUES(4, '당신의 인생 영화는?', '댓글 ㄱ');
7. INSERT INTO article(id, title, content) VALUES(5, '당신의 소울 푸드는?', '댓글 ㄱㄱ');
8. INSERT INTO article(id, title, content) VALUES(6, '당신의 취미는?', '댓글 ㄱㄱㄱ');
9. -- 22강: comment 더미 데이터
10. ---- 4번 게시글의 댓글들
11. INSERT INTO comment(id, article\_id, nickname, body) VALUES(1, 4, 'Park', '굳 윌 헌팅');
12. INSERT INTO comment(id, article\_id, nickname, body) VALUES(2, 4, 'Kim', '아이 엠 샘');
13. INSERT INTO comment(id, article\_id, nickname, body) VALUES(3, 4, 'Choi', '쇼생크의 탈출');
14. ---- 5번 게시글의 댓글들
15. INSERT INTO comment(id, article\_id, nickname, body) VALUES(4, 5, 'Park', '치킨');
16. INSERT INTO comment(id, article\_id, nickname, body) VALUES(5, 5, 'Kim', '샤브샤브');
17. INSERT INTO comment(id, article\_id, nickname, body) VALUES(6, 5, 'Choi', '초밥');
18. ---- 6번 게시글의 댓글들
19. INSERT INTO comment(id, article\_id, nickname, body) VALUES(7, 6, 'Park', '조깅');
20. INSERT INTO comment(id, article\_id, nickname, body) VALUES(8, 6, 'Kim', '유튜브');
21. INSERT INTO comment(id, article\_id, nickname, body) VALUES(9, 6, 'Choi', '독서');

**12:09 SQL 연습**

h2 콘솔에서 SQL 연습

1. -- 4번 게시글의 모든 댓글
2. SELECT
3. \*
4. FROM
5. comment
6. WHERE
7. article\_id = 4
8. ;
9. -- 닉네임이 Park인 모든 댓글
10. SELECT
11. \*
12. FROM
13. comment
14. WHERE
15. nickname = 'Park'
16. ;

**13:41 댓글 리파지터리 구현 - @Query**

../repository/CommentRepository

1. package com.example.firstproject.repository;
2. import com.example.firstproject.entity.Comment;
3. import org.springframework.data.jpa.repository.JpaRepository;
4. import org.springframework.data.jpa.repository.Query;
5. import java.util.List;
6. public interface CommentRepository extends JpaRepository<Comment, Long> {
7. @Query(value =
8. "SELECT \* " +
9. "FROM comment " +
10. "WHERE article\_id = :articleId",
11. nativeQuery = true)
12. List<Comment> findByArticleId(Long articleId);
13. List<Comment> findByNickname(String nickname);
14. }

⚠️ articleId를 찾지 못해 에러 발생 시, @Param 어노테이션으로 파라미터 정보 추가

1. @Query(value =
2. "SELECT \* " +
3. "FROM comment " +
4. "WHERE article\_id = :articleId",
5. nativeQuery = true)
6. List<Comment> findByArticleId(@Param("articleId") Long articleId);

**17:28 네이티브 쿼리 - named-native-query, orm.xml**

../resources/META-INF/orm.xml

1. <?xml version="1.0" encoding="utf-8" ?>
2. <entity-mappings xmlns="https://jakarta.ee/xml/ns/persistence/orm"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence/orm
5. https://jakarta.ee/xml/ns/persistence/orm/orm\_3\_0.xsd"
6. version="3.0">
7. <named-native-query
8. name="Comment.findByNickname"
9. result-class="com.example.firstproject.entity.Comment">
10. <query>
11. <![CDATA[
12. SELECT
13. \*
14. FROM
15. comment
16. WHERE
17. nickname = :nickname
18. ]]>
19. </query>
20. </named-native-query>
21. </entity-mappings>

**20:24 리파지터리 테스트**

../test/…/repository/CommentRepositoryTest

1. package com.example.firstproject.repository;
2. import com.example.firstproject.entity.Article;
3. import com.example.firstproject.entity.Comment;
4. import org.junit.jupiter.api.DisplayName;
5. import org.junit.jupiter.api.Test;
6. import org.springframework.beans.factory.annotation.Autowired;
7. import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
8. import java.util.List;
9. import static org.junit.jupiter.api.Assertions.\*;
10. @DataJpaTest
11. class CommentRepositoryTest {
12. @Autowired
13. CommentRepository commentRepository;
14. @Test
15. void findByArticleId() {
16. }
17. @Test
18. void findByNickname() {
19. }
20. }

../test/…/repository/CommentRepositoryTest

1. package com.example.firstproject.repository;
2. import com.example.firstproject.entity.Article;
3. import com.example.firstproject.entity.Comment;
4. import org.junit.jupiter.api.DisplayName;
5. import org.junit.jupiter.api.Test;
6. import org.springframework.beans.factory.annotation.Autowired;
7. import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
8. import java.util.Arrays;
9. import java.util.List;
10. import static org.junit.jupiter.api.Assertions.\*;
11. @DataJpaTest
12. class CommentRepositoryTest {
13. @Autowired
14. CommentRepository commentRepository;
15. @Test
16. @DisplayName("특정 게스글의 모든 댓글 조회")
17. void findByArticleId() {
18. /\* Case 1: 4번 게시글의 모든 댓글 조회 \*/
19. {
20. // 준비
21. Long articleId = 4L;
22. // 수행
23. List<Comment> comments = commentRepository.findByArticleId(articleId);
24. // 예상
25. Article article = new Article(4L, "당신의 인생 영화는?", "댓글 ㄱ");
26. Comment a = new Comment(1L, article, "Park", "굳 윌 헌팅");
27. Comment b = new Comment(2L, article, "Kim", "아이 엠 샘");
28. Comment c = new Comment(3L, article, "Choi", "쇼생크의 탈출");
29. List<Comment> expected = Arrays.asList(a, b, c);
30. // 검증
31. assertEquals(expected.toString(), comments.toString(), "4번 글의 모든 댓글을 출력!");
32. }
33. /\* Case 2: 1번 게시글의 모든 댓글 조회 \*/
34. {
35. // 준비
36. Long articleId = 1L;
37. // 수행
38. List<Comment> comments = commentRepository.findByArticleId(articleId);
39. // 예상
40. Article article = new Article(1L, "가가가가", "1111");
41. List<Comment> expected = Arrays.asList();
42. // 검증
43. assertEquals(expected.toString(), comments.toString(), "1번 글은 댓글이 없음");
44. }
45. }
46. @Test
47. @DisplayName("특정 닉네임의 모든 댓글 조회")
48. void findByNickname() {
49. /\* Case 1: "Park"의 모든 댓글 조회 \*/
50. {
51. // 준비
52. String nickname = "Park";
53. // 수행
54. List<Comment> comments = commentRepository.findByNickname(nickname);
55. // 예상
56. Comment a = new Comment(1L, new Article(4L, "당신의 인생 영화는?", "댓글 ㄱ"), nickname, "굳 윌 헌팅");
57. Comment b = new Comment(4L, new Article(5L, "당신의 소울 푸드는?", "댓글 ㄱㄱ"), nickname, "치킨");
58. Comment c = new Comment(7L, new Article(6L, "당신의 취미는?", "댓글 ㄱㄱㄱ"), nickname, "조깅");
59. List<Comment> expected = Arrays.asList(a, b, c);
60. // 검증
61. assertEquals(expected.toString(), comments.toString(), "Park의 모든 댓글을 출력!");
62. }
63. }
64. }

**🔥 구글링 훈련하기**

* 일대다 다대일 관계
* @ManyToOne
* @JoinColumn
* JpaRepository
* @Query
* JPA native query orm xml
* @DataJpaTest @SpringBootTest 비교