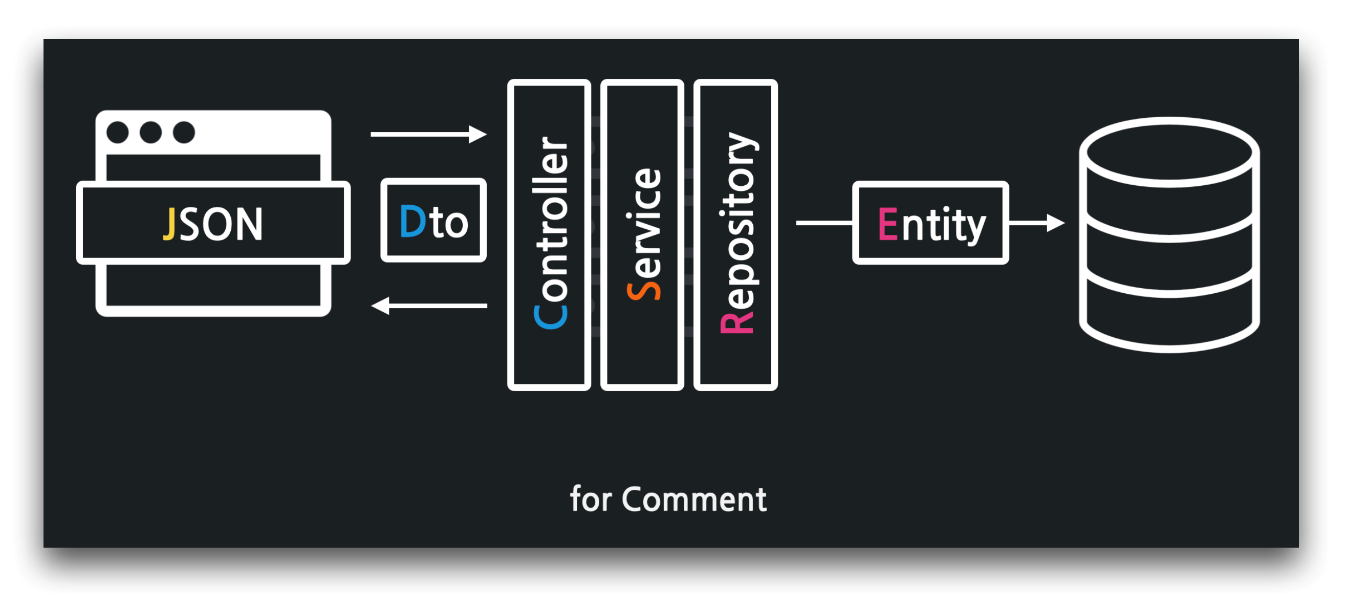
23 댓글 서비스와 컨트롤러(feat. REST API)

**댓글 서비스와 컨트롤러(feat. REST API)**

**미션**

댓글 REST API를 완성하기 위해, 컨트롤러와 서비스를 구현하시오.  


**01:07 REST 컨트롤러 생성 - @RestController**

../api/CommentApiController

1. package com.example.firstproject.api;
2. import org.springframework.beans.factory.annotation.Autowired;
3. import org.springframework.web.bind.annotation.RestController;
4. @RestController
5. public class CommentApiController {
6. @Autowired
7. private CommentService commentService;
8. // 댓글 목록 조회
9. // 댓글 생성
10. // 댓글 수정
11. // 댓글 삭제
12. }

**01:54 Comment 서비스 생성 - @Service**

../service/CommentService

1. package com.example.firstproject.service;
2. import com.example.firstproject.repository.ArticleRepository;
3. import com.example.firstproject.repository.CommentRepository;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.stereotype.Service;
6. @Service
7. public class CommentService {
8. @Autowired
9. private CommentRepository commentRepository;
10. @Autowired
11. private ArticleRepository articleRepository;
12. }

**03:40 댓글 목록 조회**

../api/CommentApiController

1. ...
2. @RestController
3. public class CommentApiController {
4. @Autowired
5. private CommentService commentService;
6. // 댓글 목록 조회
7. @GetMapping("/api/articles/{articleId}/comments")
8. public ResponseEntity<List<CommentDto>> comments(@PathVariable Long articleId) {
9. // 서비스에게 위임
10. List<CommentDto> dtos = commentService.comments(articleId);
11. // 결과 응답
12. return ResponseEntity.status(HttpStatus.OK).body(dtos);
13. }
14. ...
15. }

../dto/CommentDto

1. package com.example.firstproject.dto;
2. import lombok.\*;
3. @AllArgsConstructor
4. @NoArgsConstructor
5. @Getter
6. @ToString
7. public class CommentDto {
8. private Long id;
9. private Long articleId;
10. private String nickname;
11. private String body;
12. }

../service/CommentService

1. ...
2. @Service
3. public class CommentService {
4. @Autowired
5. private CommentRepository commentRepository;
6. @Autowired
7. private ArticleRepository articleRepository;
8. public List<CommentDto> comments(Long articleId) {
9. // 조회: 댓글 목록
10. List<Comment> comments = commentRepository.findByArticleId(articleId);
11. // 변환: 엔티티 -> DTO
12. List<CommentDto> dtos = new ArrayList<CommentDto>();
13. for (int i = 0; i < comments.size(); i++) {
14. Comment c = comments.get(i);
15. CommentDto dto = CommentDto.createCommentDto(c);
16. dtos.add(dto);
17. }
18. // 반환
19. return dtos;
20. }
21. }

../dto/CommentDto

1. ...
2. public class CommentDto {
3. ...
4. public static CommentDto createCommentDto(Comment comment) {
5. return new CommentDto(
6. comment.getId(),
7. comment.getArticle().getId(),
8. comment.getNickname(),
9. comment.getBody()
10. );
11. }
12. }

**13:44 반복을 스트림으로 - stream(), map(), collent(), Collectors.toList()**

../service/CommentService

1. ...
2. public class CommentService {
3. ...
4. public List<CommentDto> comments(Long articleId) {
5. // 반환
6. return commentRepository.findByArticleId(articleId)
7. .stream()
8. .map(comment -> CommentDto.createCommentDto(comment))
9. .collect(Collectors.toList());
10. }
11. }

**15:22 댓글 생성하기 - HTTP POST, @PostMapping, IllegalArgumentException,**

../api/CommentApiController

1. ...
2. @RestController
3. public class CommentApiController {
4. ...
5. // 댓글 생성
6. @PostMapping("/api/articles/{articleId}/comments")
7. public ResponseEntity<CommentDto> create(@PathVariable Long articleId,
8. @RequestBody CommentDto dto) {
9. // 서비스에게 위임
10. CommentDto createdDto = commentService.create(articleId, dto);
11. // 결과 응답
12. return ResponseEntity.status(HttpStatus.OK).body(createdDto);
13. }
14. ...
15. }

../service/CommentService

1. ...
2. public class CommentService {
3. ...
4. @Transactional
5. public CommentDto create(Long articleId, CommentDto dto) {
6. // 게시글 조회 및 예외 발생
7. Article article = articleRepository.findById(articleId)
8. .orElseThrow(() -> new IllegalArgumentException("댓글 생성 실패! 대상 게시글이 없습니."));
9. // 댓글 엔티티 생성
10. Comment comment = Comment.createComment(dto, article);
11. // 댓글 엔티티를 DB로 저장
12. Comment created = commentRepository.save(comment);
13. // DTO로 변경하여 반환
14. return CommentDto.createCommentDto(created);
15. }
16. }

../entity/Comment

1. ...
2. public class Comment {
3. ...
4. public static Comment createComment(CommentDto dto, Article article) {
5. // 예외 발생
6. if (dto.getId() != null)
7. throw new IllegalArgumentException("댓글 생성 실패! 댓글의 id가 없어야 합니다.");
8. if (dto.getArticleId() != article.getId())
9. throw new IllegalArgumentException("댓글 생성 실패! 게시글의 id가 잘못되었습니다.");
10. // 엔티티 생성 및 반환
11. return new Comment(
12. dto.getId(),
13. article,
14. dto.getNickname(),
15. dto.getBody()
16. );
17. }
18. }

**25:00 디버깅 - @JsonProperty**

../dto/CommentDto

1. package com.example.firstproject.dto;
2. ...
3. public class CommentDto {
4. private Long id;
5. @JsonProperty("article\_id")
6. private Long articleId;
7. private String nickname;
8. private String body;
9. ...
10. }

**27:09 댓글 수정하기 - HTTP PATCH, @PatchMapping, IllegalArgumentException**

../api/CommentApiController

1. ...
2. @RestController
3. public class CommentApiController {
4. ...
5. // 댓글 수정
6. @PatchMapping("/api/comments/{id}")
7. public ResponseEntity<CommentDto> update(@PathVariable Long id,
8. @RequestBody CommentDto dto) {
9. // 서비스에게 위임
10. CommentDto updatedDto = commentService.update(id, dto);
11. // 결과 응답
12. return ResponseEntity.status(HttpStatus.OK).body(updatedDto);
13. }
14. ...
15. }

../service/CommentService

1. ...
2. @Service
3. public class CommentService {
4. ...
5. @Transactional
6. public CommentDto update(Long id, CommentDto dto) {
7. // 댓글 조회 및 예외 발생
8. Comment target = commentRepository.findById(id)
9. .orElseThrow(() -> new IllegalArgumentException("댓글 수정 실패! 대상 댓글이 없습니다."));
10. // 댓글 수정
11. target.patch(dto);
12. // DB로 갱신
13. Comment updated = commentRepository.save(target);
14. // 댓글 엔티티를 DTO로 변환 및 반환
15. return CommentDto.createCommentDto(updated);
16. }
17. }

../entity/Comment

1. ...
2. public class Comment {
3. ...
4. public void patch(CommentDto dto) {
5. // 예외 발생
6. if (this.id != dto.getId())
7. throw new IllegalArgumentException("댓글 수정 실패! 잘못된 id가 입력되었습니다.");
8. // 객체를 갱신
9. if (dto.getNickname() != null)
10. this.nickname = dto.getNickname();
11. if (dto.getBody() != null)
12. this.body = dto.getBody();
13. }
14. }

**34:23 댓글 삭제하기 - HTTP DELETE, @DeleteMapping, IllegalArgumentException**

../api/CommentApiController

1. ...
2. @RestController
3. public class CommentApiController {
4. ...
5. // 댓글 삭제
6. @DeleteMapping("/api/comments/{id}")
7. public ResponseEntity<CommentDto> delete(@PathVariable Long id) {
8. // 서비스에게 위임
9. CommentDto deletedDto = commentService.delete(id);
10. // 결과 응답
11. return ResponseEntity.status(HttpStatus.OK).body(deletedDto);
12. }
13. }

../service/CommentService

1. ...
2. public class CommentService {
3. ...
4. @Transactional
5. public CommentDto delete(Long id) {
6. // 댓글 조회(및 예외 발생)
7. Comment target = commentRepository.findById(id)
8. .orElseThrow(() -> new IllegalArgumentException("댓글 삭제 실패! 대상이 없습니다."));
9. // 댓글 삭제
10. commentRepository.delete(target);
11. // 삭제 댓글을 DTO로 반환
12. return CommentDto.createCommentDto(target);
13. }
14. }

**🔥 구글링 훈련하기**

* 롬복 @Builder 사용법
* 자바8 스트림이란
* 자바8 스트림 map
* 자바8 스트림 collect
* @JsonProperty