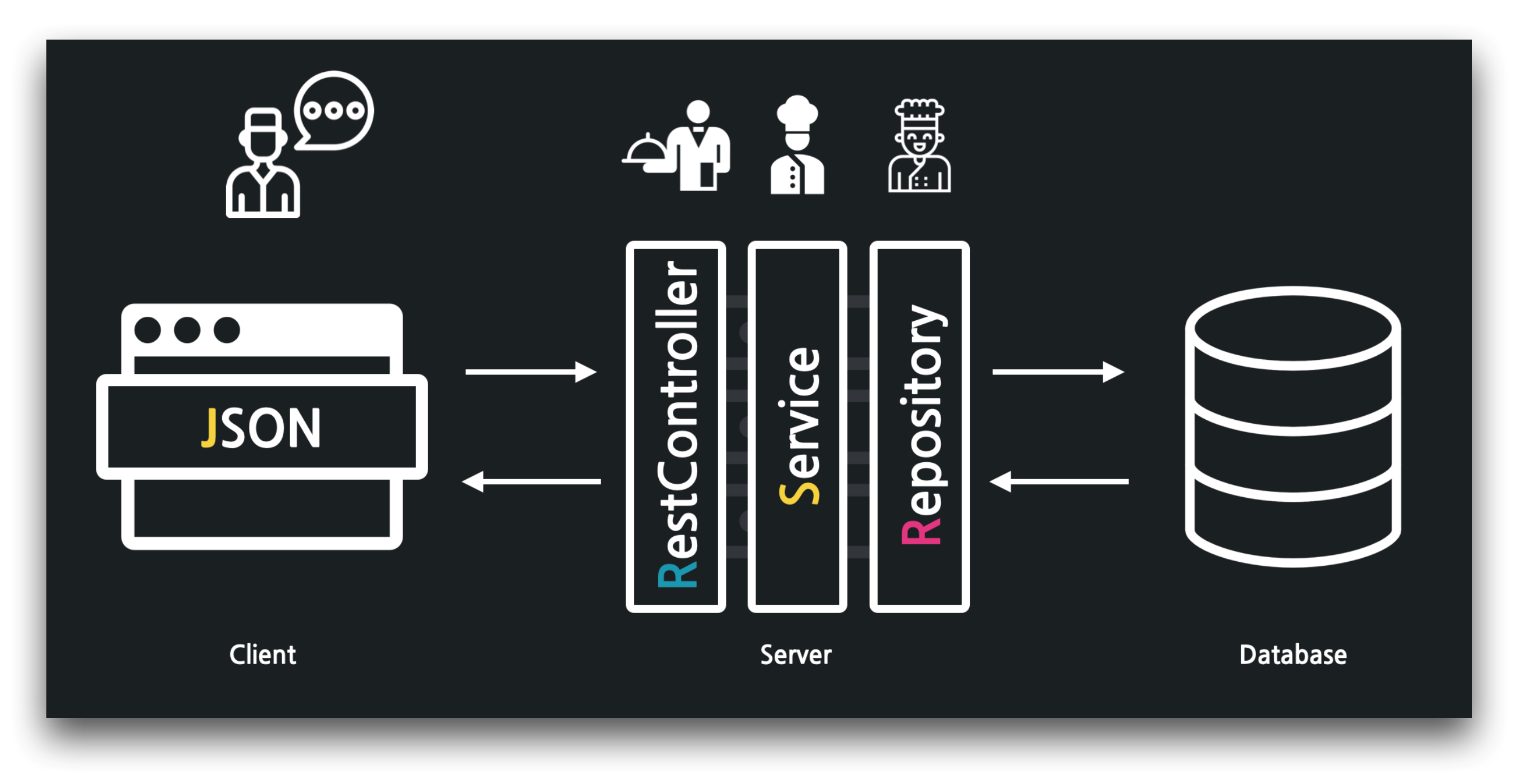
20 서비스와 트랜잭션(transaction), 그리고 롤백(rollback)

**서비스와 트랜잭션, 그리고 롤백**

**미션**

서비스 계층을 추가하여, 기존 Article REST API를 리팩터링하시오.  


**서비스 생성 - @Service, DI, @Autowired**

../api/ArticleApiController

1. package com.example.firstproject.api;
2. import lombok.extern.slf4j.Slf4j;
3. import org.springframework.beans.factory.annotation.Autowired;
4. import org.springframework.web.bind.annotation.\*;
5. @Slf4j
6. @RestController
7. public
8. class ArticleApiController {
9. @Autowired
10. private ArticleService articleService;
11. }

../serivce/ArticleService

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

**리팩토링, Article 목록 조회**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. @Autowired
6. private ArticleService articleService;
7. // GET
8. @GetMapping("/api/articles")
9. public List<Article> index() {
10. return articleService.index();
11. }
12. }

../service/ArticleService

1. package com.example.firstproject.service;
2. import com.example.firstproject.entity.Article;
3. import com.example.firstproject.repository.ArticleRepository;
4. import org.springframework.beans.factory.annotation.Autowired;
5. import org.springframework.stereotype.Service;
6. import java.util.List;
7. @Service
8. public class ArticleService {
9. @Autowired
10. private ArticleRepository articleRepository;
11. public List<Article> index() {
12. return articleRepository.findAll();
13. }
14. }

**리팩토링, Article 단건(상세) 조회**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. ...
6. @GetMapping("/api/articles/{id}")
7. public Article show(@PathVariable Long id) {
8. return articleService.show(id);
9. }
10. }

../service/ArticleService

1. ...
2. @Service
3. public class ArticleService {
4. ...
5. public Article show(Long id) {
6. return articleRepository.findById(id).orElse(null);
7. }
8. }

**리팩토링, Article 생성**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. ...
6. // POST
7. @PostMapping("/api/articles")
8. public ResponseEntity<Article> create(@RequestBody ArticleForm dto) {
9. Article created = articleService.create(dto);
10. return (created != null) ?
11. ResponseEntity.status(HttpStatus.OK).body(created) :
12. ResponseEntity.status(HttpStatus.BAD\_REQUEST).build();
13. }
14. }

../service/ArticleService

1. ...
2. @Service
3. public class ArticleService {
4. ...
5. public Article create(ArticleForm dto) {
6. Article article = dto.toEntity();
7. if (article.getId() != null) {
8. return null;
9. }
10. return articleRepository.save(article);
11. }
12. }

**12:41 리팩토링, Article 수정**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. ...
6. // PATCH
7. @PatchMapping("/api/articles/{id}")
8. public ResponseEntity<Article> update(@PathVariable Long id,
9. @RequestBody ArticleForm dto) {
10. Article updated = articleService.update(id, dto);
11. return (updated != null) ?
12. ResponseEntity.status(HttpStatus.OK).body(updated):
13. ResponseEntity.status(HttpStatus.BAD\_REQUEST).build();
14. }
15. }

../service/ArticleService

1. ...
2. @Slf4j
3. @Service
4. public class ArticleService {
5. ...
6. public Article update(Long id, ArticleForm dto) {
7. // 1: DTO -> 엔티티
8. Article article = dto.toEntity();
9. log.info("id: {}, article: {}", id, article.toString());
10. // 2: 타겟 조회
11. Article target = articleRepository.findById(id).orElse(null);
12. // 3: 잘못된 요청 처리
13. if (target == null || id != article.getId()) {
14. // 400, 잘못된 요청 응답!
15. log.info("잘못된 요청! id: {}, article: {}", id, article.toString());
16. return null;
17. }
18. // 4: 업데이트
19. target.patch(article);
20. Article updated = articleRepository.save(target);
21. return updated;
22. }
23. }

**16:22 리팩토링, Article 삭제**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. ...
6. // DELETE
7. @DeleteMapping("/api/articles/{id}")
8. public ResponseEntity<Article> delete(@PathVariable Long id) {
9. Article deleted = articleService.delete(id);
10. return (deleted != null) ?
11. ResponseEntity.status(HttpStatus.NO\_CONTENT).build() :
12. ResponseEntity.status(HttpStatus.BAD\_REQUEST).build();
13. }
14. }

../service/ArticleService

1. ...
2. @Slf4j
3. @Service
4. public class ArticleService {
5. ...
6. public Article delete(Long id) {
7. // 대상 찾기
8. Article target = articleRepository.findById(id).orElse(null);
9. // 잘못된 요청 처리
10. if (target == null) {
11. return null;
12. }
13. // 대상 삭제
14. articleRepository.delete(target);
15. return target;
16. }
17. }

**트랜잭션 맛보기, 묶음 Article 생성 - @Transactional**

../api/ArticleApiController

1. ...
2. @Slf4j
3. @RestController
4. public class ArticleApiController {
5. ...
6. // 트랜잭션 -> 실패 -> 롤백!
7. @PostMapping("/api/transaction-test")
8. public ResponseEntity<List<Article>> transactionTest(@RequestBody List<ArticleForm> dtos) {
9. List<Article> createdList = articleService.createArticles(dtos);
10. return (createdList != null) ?
11. ResponseEntity.status(HttpStatus.OK).body(createdList) :
12. ResponseEntity.status(HttpStatus.BAD\_REQUEST).build();
13. }
14. }

../service/ArticleService

1. package com.example.firstproject.service;
2. import com.example.firstproject.dto.ArticleForm;
3. import com.example.firstproject.entity.Article;
4. import com.example.firstproject.repository.ArticleRepository;
5. import lombok.extern.slf4j.Slf4j;
6. import org.springframework.beans.factory.annotation.Autowired;
7. import org.springframework.stereotype.Service;
8. import org.springframework.transaction.annotation.Transactional;
9. import java.util.List;
10. import java.util.stream.Collectors;
11. @Slf4j
12. @Service
13. public class ArticleService {
14. ...
15. @Transactional
16. public List<Article> createArticles(List<ArticleForm> dtos) {
17. // dto 묶음을 entity 묶음으로 변환
18. List<Article> articleList = dtos.stream()
19. .map(dto -> dto.toEntity())
20. .collect(Collectors.toList());
21. // entity 묶음을 DB로 저장
22. articleList.stream()
23. .forEach(article -> articleRepository.save(article));
24. // 강제 예외 발생
25. articleRepository.findById(-1L).orElseThrow(
26. () -> new IllegalArgumentException("결제 실패!")
27. );
28. // 결과값 반환
29. return articleList;
30. }
31. }

**🔥 구글링 훈련하기**

* Spring Controller Service Repository 역할
* 트랜잭션과 롤백
* @Service
* @Transactional