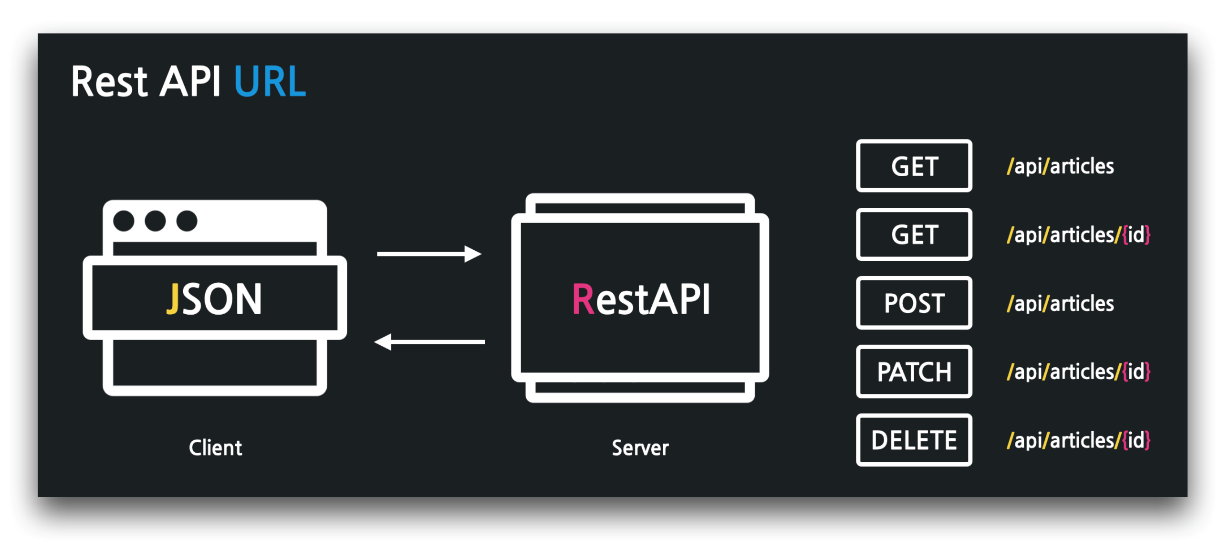
19 HTTP와 RestController

**HTTP와 RestController**

**미션**

기존 Article 데이터를 crud 하기 위한 Rest api를 구현 하시오

Article 데이터 CRUD를 위한, REST API를 만드시오.  


**03:28 헬로 RestAPI - @RestController**

../api/FirstApiController

1. package com.example.firstproject.api;
2. import org.springframework.web.bind.annotation.GetMapping;
3. import org.springframework.web.bind.annotation.RestController;
4. @RestController
5. public class FirstApiController {
6. @GetMapping("/api/hello")
7. public String hello() {
8. return "hello world!";
9. }
10. }

**06:43 RestAPI GET 구현 - @GetMapping, @Autowired, DI**

../api/ArticleApiController

1. package com.example.firstproject.api;
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.web.bind.annotation.GetMapping;
6. import org.springframework.web.bind.annotation.PathVariable;
7. import org.springframework.web.bind.annotation.RestController;
8. import java.util.List;
9. @RestController
10. public class ArticleApiController {
11. @Autowired
12. private ArticleRepository articleRepository;
13. // GET
14. @GetMapping("/api/articles")
15. public List<Article> index() {
16. return articleRepository.findAll();
17. }
18. @GetMapping("/api/articles/{id}")
19. public Article show(@PathVariable Long id) {
20. return articleRepository.findById(id).orElse(null);
21. }
22. // POST
23. // PATCH
24. // DELETE
25. }

**11:14 RestAPI POST 구현 - @PostMapping, @RequestBody**

…/api/ArticleApiController

1. ...
2. @RestController
3. public class ArticleApiController {
4. ...
5. // POST
6. @PostMapping("/api/articles")
7. public Article create(@RequestBody ArticleForm dto) {
8. Article article = dto.toEntity();
9. return articleRepository.save(article);
10. }
11. // PATCH
12. // DELETE
13. }

**14:46 RestAPI PATCH 구현**

../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. // 1: DTO -> 엔티티
11. Article article = dto.toEntity();
12. log.info("id: {}, article: {}", id, article.toString());
13. // 2: 타겟 조회
14. Article target = articleRepository.findById(id).orElse(null);
15. // 3: 잘못된 요청 처리
16. if (target == null || id != article.getId()) {
17. // 400, 잘못된 요청 응답!
18. log.info("잘못된 요청! id: {}, article: {}", id, article.toString());
19. return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);
20. }
21. // 4: 업데이트 및 정상 응답(200)
22. target.patch(article);
23. Article updated = articleRepository.save(target);
24. return ResponseEntity.status(HttpStatus.OK).body(updated);
25. }
26. // DELETE
27. }

../entity/Article

1. package com.example.firstproject.entity;
2. ...
3. public class Article {
4. ...
5. public void patch(Article article) {
6. if (article.title != null)
7. this.title = article.title;
8. if (article.content != null)
9. this.content = article.content;
10. }
11. }

**23:58 RestAPI DELETE 구현**

../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. // 대상 찾기
10. Article target = articleRepository.findById(id).orElse(null);
11. // 잘못된 요청 처리
12. if (target == null) {
13. return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(null);
14. }
15. // 대상 삭제
16. articleRepository.delete(target);
17. return ResponseEntity.status(HttpStatus.OK).build();
18. }
19. }

**🔥 구글링 훈련하기**

* @RestController
* @RequestBody
* JSON이란
* Spring ResponseEntity
* @PatchMapping
* @DeleteMapping
* 도메인 모델 패턴
* 트랜잭션 스크립트 패턴