## 12장 완성 코드

### api/ArticleApiController.java

**package** com.example.firstproject.api;  
  
**import** com.example.firstproject.dto.ArticleForm;  
**import** com.example.firstproject.entity.Article;  
**import** com.example.firstproject.service.ArticleService;  
**import** lombok.extern.slf4j.Slf4j;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.HttpStatus;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
  
**import** java.util.List;  
  
@Slf4j  
@RestController  
**public class** ArticleApiController {  
 @Autowired  
 **private** ArticleService **articleService**;  
  
 *// GET* @GetMapping(**"/api/articles"**)  
 **public** List<Article> index() {  
 **return articleService**.index();  
 }  
  
 @GetMapping(**"/api/articles/{id}"**)  
 **public** Article show(@PathVariable Long id) {  
 **return articleService**.show(id);  
 }  
  
 *// POST* @PostMapping(**"/api/articles"**)  
 **public** ResponseEntity<Article> create(@RequestBody ArticleForm dto) {  
 Article created = **articleService**.create(dto);  
 **return** (created != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***OK***).body(created) :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
  
 }  
  
 *// PATCH* @PatchMapping(**"/api/articles/{id}"**)  
 **public** ResponseEntity<Article> update(@PathVariable Long id,  
 @RequestBody ArticleForm dto) {  
 Article updated = **articleService**.update(id, dto);  
 **return** (updated != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***OK***).body(updated):  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
  
 *// DELETE* @DeleteMapping(**"/api/articles/{id}"**)  
 **public** ResponseEntity<Article> delete(@PathVariable Long id) {  
 Article deleted = **articleService**.delete(id);  
 **return** (deleted != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***NO\_CONTENT***).build() :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
  
 @PostMapping(**"/api/transaction-test"**)  
 **public** ResponseEntity<List<Article>> transactionTest(@RequestBody  
 List<ArticleForm> dtos) {  
 List<Article> createdList = **articleService**.createArticles(dtos);  
 **return** (createdList != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***OK***).body(createdList) :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
  
}

### service/ArticleService.java

**package** com.example.firstproject.service;  
  
**import** com.example.firstproject.dto.ArticleForm;  
**import** com.example.firstproject.entity.Article;  
**import** com.example.firstproject.repository.ArticleRepository;  
**import** lombok.extern.slf4j.Slf4j;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Service;  
**import** org.springframework.transaction.annotation.Transactional;  
  
**import** java.util.List;  
**import** java.util.stream.Collectors;  
  
@Slf4j  
@Service *// 서비스 객체 선언***public class** ArticleService {  
 @Autowired  
 **private** ArticleRepository **articleRepository**;  
  
 **public** List<Article> index() {  
 **return articleRepository**.findAll();  
 }  
  
 **public** Article show(Long id) {  
 **return articleRepository**.findById(id).orElse(**null**);  
 }  
  
 **public** Article create(ArticleForm dto) {  
 Article article = dto.toEntity();  
 **if** (article.getId() != **null**) {  
 **return null**;  
 }  
 **return articleRepository**.save(article);  
 }  
  
 **public** Article update(Long id, ArticleForm dto) {  
 *// 1. DTO -> 엔티티 변환하기* Article article = dto.toEntity();  
 ***log***.info(**"id: {}, article: {}"**, id, article.toString());  
  
 *// 2. 타겟 조회하기* Article target = **articleRepository**.findById(id).orElse(**null**);  
  
 *// 3. 잘못된 요청 처리하기* **if** (target == **null** || id != article.getId()) {  
 *// 400번, 잘못된 요청 응답!* ***log***.info(**"잘못된 요청! id: {}, article: {}"**, id, article.toString());  
 **return null**;  
 }  
  
 *// 4. 업데이트 및 정상 응답(200)하기* target.patch(article);  
 Article updated = **articleRepository**.save(target);  
 **return** updated;  
 }  
  
 **public** Article delete(Long id) {  
 *// 1. 대상 찾기* Article target = **articleRepository**.findById(id).orElse(**null**);  
  
 *// 2. 잘못된 요청 처리하기* **if** (target == **null**) {  
 **return null**;  
 }  
  
 *// 3. 대상 삭제하기* **articleRepository**.delete(target);  
 **return** target;  
 }  
 @Transactional  
 **public** List<Article> createArticles(List<ArticleForm> dtos) {  
 *// 1. dto 묶음을 엔티티 묶음으로 변환하기* List<Article> articleList = dtos.stream()  
 .map(dto -> dto.toEntity())  
 .collect(Collectors.*toList*());  
  
 *// 2. 엔티티 묶음을 DB에 저장하기* articleList.stream()  
 .forEach(article -> **articleRepository**.save(article));  
  
 *// 3. 강제 예외 발생시키기* **articleRepository**.findById(-1L)  
 .orElseThrow(() -> **new** IllegalArgumentException(**"결제 실패!"**));  
  
 *// 4. 결과값 반환하기* **return** articleList;  
 }  
}

### 셀프체크 정답

#### api/CoffeeApiController.java

**package** com.example.firstproject.api;  
  
**import** com.example.firstproject.dto.CoffeeDto;  
**import** com.example.firstproject.entity.Coffee;  
**import** com.example.firstproject.service.CoffeeService;  
**import** lombok.extern.slf4j.Slf4j;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.HttpStatus;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.web.bind.annotation.\*;  
  
@Slf4j  
@RestController  
**public class** CoffeeApiController {  
 @Autowired  
 **private** CoffeeService **coffeeService**;  
  
 *// GET* @GetMapping(**"/api/coffees"**)  
 **public** Iterable<Coffee> index() {  
 **return coffeeService**.index();  
 }  
  
 @GetMapping(**"/api/coffees/{id}"**)  
 **public** ResponseEntity<Coffee> show(@PathVariable Long id) {  
 Coffee coffee = **coffeeService**.show(id);  
 **return** (coffee != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***OK***).body(coffee) :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).body(**null**);  
 }  
  
 *// POST* @PostMapping(**"/api/coffees"**)  
 **public** ResponseEntity<Coffee> create(@RequestBody CoffeeDto coffeeDto) {  
 Coffee created = **coffeeService**.create(coffeeDto);  
 **return** (created != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***CREATED***).body(created) :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
  
 *// PATCH* @PatchMapping(**"/api/coffees/{id}"**)  
 **public** ResponseEntity<Coffee> update(@PathVariable Long id,  
 @RequestBody CoffeeDto coffeeDto) {  
 Coffee updated = **coffeeService**.update(id, coffeeDto);  
 **return** (updated != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***OK***).body(updated) :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
  
 *// DELETE* @DeleteMapping(**"/api/coffees/{id}"**)  
 **public** ResponseEntity<Coffee> delete(@PathVariable Long id) {  
 Coffee deleted = **coffeeService**.delete(id);  
 **return** (deleted != **null**) ?  
 ResponseEntity.*status*(HttpStatus.***NO\_CONTENT***).build() :  
 ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).build();  
 }  
}

#### service/CoffeeService.java

**package** com.example.firstproject.service;  
  
**import** com.example.firstproject.dto.CoffeeDto;  
**import** com.example.firstproject.entity.Coffee;  
**import** com.example.firstproject.repository.CoffeeRepository;  
**import** lombok.extern.slf4j.Slf4j;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Service;  
  
@Slf4j  
@Service  
**public class** CoffeeService {  
 @Autowired  
 **private** CoffeeRepository **coffeeRepository**;  
  
 **public** Iterable<Coffee> index() {  
 **return coffeeRepository**.findAll();  
 }  
  
 **public** Coffee show(Long id) {  
 **return coffeeRepository**.findById(id).orElse(**null**);  
 }  
  
 **public** Coffee create(CoffeeDto coffeeDto) {  
 Coffee coffee = coffeeDto.toEntity();  
 **if** (coffee.getId() != **null**) {  
 **return null**;  
 }  
 **return coffeeRepository**.save(coffee);  
 }  
  
 **public** Coffee update(Long id, CoffeeDto coffeeDto) {  
 Coffee coffee = coffeeDto.toEntity();  
 ***log***.info(**"id: {}, coffee: {}"**, id, coffee.toString());  
  
 Coffee target = **coffeeRepository**.findById(id).orElse(**null**);  
  
 **if** (target == **null** || id != coffee.getId()) {  
 ***log***.info(**"잘못된 요청! id: {}, coffee: {}"**, id, coffee.toString());  
 **return null**;  
 }  
 target.patch(coffee);  
 **return coffeeRepository**.save(target);  
 }  
  
 **public** Coffee delete(Long id) {  
 Coffee target = **coffeeRepository**.findById(id).orElse(**null**);  
 **if** (target == **null**) {  
 **return null**;  
 }  
 **coffeeRepository**.delete(target);  
 **return** target;  
 }  
}