4

롬복과 리팩터링

4.1 롬복이란

- **롬복**(lombok) : 코드를 간소화해 주는 라이브러리
  - 롬복을 사용하면 필수 코드를 간편하게 작성 가능
  - 로깅 기능을 통해 println() 문 개선 가능
    - **로깅**(logging) : 프로그램의 수행 과정을 기록으로 남기는 것
- **리팩터링**(refactoring) : 콩드의 기능에는 변함이 없이 코드의 구조 또는 성능을 개선하는 작업

4.2 롬복을 활용해 리팩터링하기

1) 롬복 설치하기

- firstproject > src > build.gradle 파일에 추가
- dependencies {} 블록 안에 코드 추가

```
compileOnly 'org.projectlombok:lombok'
annotationProcessor 'org.projectlombok:lombok'
```

2) DTO 리팩터링하기

- com.example.firstproject > dto > AricleForm
- ArticleForm() 생성자 간소화
  - ArticleForm() 생성자 코드 전체 삭제
  - ArticleForm 클래스 위에 `@AllArgsConstructor` 어노테이션 추가
    - 클래스 안쪽의 모든 필드를 매개변수로 하는 생성자 자동 생성
- toString() 메서드 간소화
  - toString() 메서드 코드 전체 삭제
  - `@ToString` 어노테이션 추가
    - toString() 메서드를 사용하는 것과 같은 효과가 남

3) 엔티티 리팩터링하기

- com.example.firstproject > entity > Article
- DTO와 동일한 리팩터링

4) 컨트롤러에 로그 남기기

- com.example.firstproject > controller > ArticleController

* println() 문 : 실제 서버에서 사용해 데이터를 검증하면 기록에 남지 않고, 서버의 성능에 악영향을 끼침 ⇒ 로깅 기능 사용

- println() 문 부분을 로깅으로 대체
  - ArticleController 클래스 위에 `@Slf4j` 어노테이션 추가
    - @Slf4j 어노테이션 : Simple Logging Facade for Java의 약자, 로깅 기능 사용을 위함
  - println() 문 삭제
  - log.info(); 문 작성
    - info() 괄호 안에는 출력하기 원하는 것을 넣음