4장 롬복과 리팩터링

출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팍, 길벗, 2023

롬복(Lombok)이란

- **자바 개발을 하다 보면, getter, setter, 생성자, toString() 등과 같은 필수 코드들을 만들기 위해 반복 작업을 해야 할 때가 많음**
- **롬복은 이러한 반복 잡업을 없애 주고, 코드를 간소하게 해주는 라이브러리**
- **자바 어노테이션 기반**
  - @Getter, @Setter, @ToString
  - @NoArgsConstructor, @AllArgsConstructor, @Slf4j, …
- **롬복의 기능**
  - 반복적인 코드 자동 생성
    - ✓ 코드 반복 최소화, 코드 간결화
    - ✓ 필수 코드들을 간편하게 작성
    - ✓ 가독성 증가
    - ✓ 생산성 향상
  - 로깅(logging) 기능 지원
    - ✓ 로깅: 프로그램의 수행 과정을 기록으로 남기는 것

리팩토링(Refactoring)

- **코드의 기능은 변경 없이 코드의 구조 또는 성능을 개선하는 작업**
  - 버그를 없애거나 새로운 기능의 추가하는 것이 아님
- **코드의 품질 향상과 생산성 향상이 목적**
- **리팩토링의 효과**
  - 가독성 향상
    - ✓ 코드가 이해하기 쉬워짐
  - 유지보수성 향상
    - ✓ 코드 변경시 영향 범위의 최소화
  - 테스트 용이
    - ✓ 단위 테스트 작성이 용이
  - 재사용성 향상
    - ✓ 모듈화된 구조로 재사용성 향상
  - 버그 예방
    - ✓ 구조적 개선으로 오류 감소

롬복을 활용해 리팩터링하기

- 롬복을 사용해 DTO, 엔티티를 리팩터링하기
- 로깅을 사용해 컨트롤러의 println()문을 로깅 기능으로 대체

롬복을 활용해 리팩터링하기

- **롬복 설치하기**
  - 프로젝트 탐색기에서 [firstproject – src] 디렉토리에서 build.gradle 파일을 열기
  - build.gradle 파일 수정 (교재 140페이지 수정)
    - ✓ compileOnly 'org.projectlombok:lombok:1.18.30'
      annotationProcessor 'org.projectlombok:lombok:1.18.30'
  - (Sync Gradle Changes) 클릭, 상태 표시줄에 다운로드 현황 및 완료

교제 140페이지 수정

DTO 리팩터링

- **프로젝터 탐색기에서 [com.example.firstproject – dto – ArticleForm] 열기**
- **ArticleForm() 생성자 간소화**
  - ArticleForm() 생성자 코드 전체 삭제
  - ArticleForm 클래스 선언문 윗줄에 @AllArgsConstructor 어노테이션 추가
    - ✓ 클래스에서 선언된 모든 필드들(title, content)를 매개변수로 하는 생성자가 자동으로 생성됨
- **toString() 메소드 간소화**
  - toString() 메소드 전체 삭제
  - ArticleForm 클래스 선언문 윗줄에 @ToString 어노테이션 추가
    - ✓ toString() 메서드가 자동으로 생성됨

➔ **ArticleForm 클래스의 코드가 간단해지고, 개발 시간 단축**

DTO 리팩터링

- 리팩터링된 ArticleForm 클래스

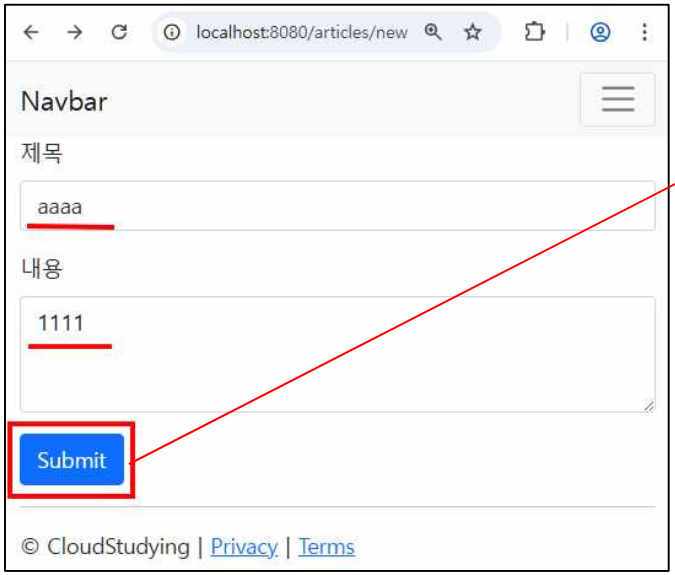
```
package com.example.firstproject.dto;

import com.example.firstproject.entity.Article;
import lombok.AllArgsConstructor;
import lombok.ToString;

@AllArgsConstructor
@ToString
public class ArticleForm {
    private String title;
    private String content;

    public Article toEntity() {
        return new Article( id: null, title, content);
    }
}
```

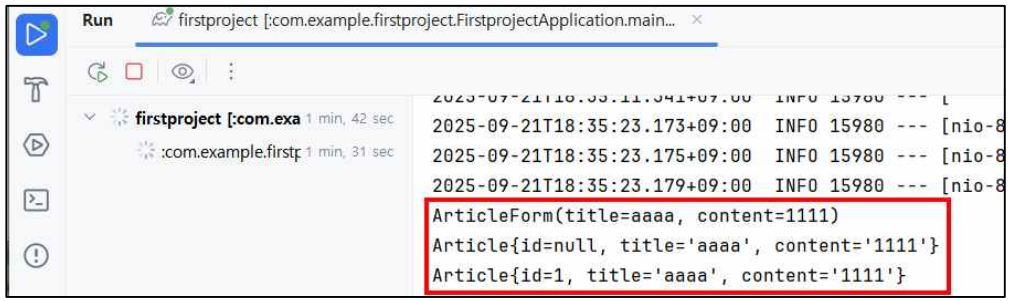
자동으로 import

새 어노테이션 추가

DTO 리팩터링

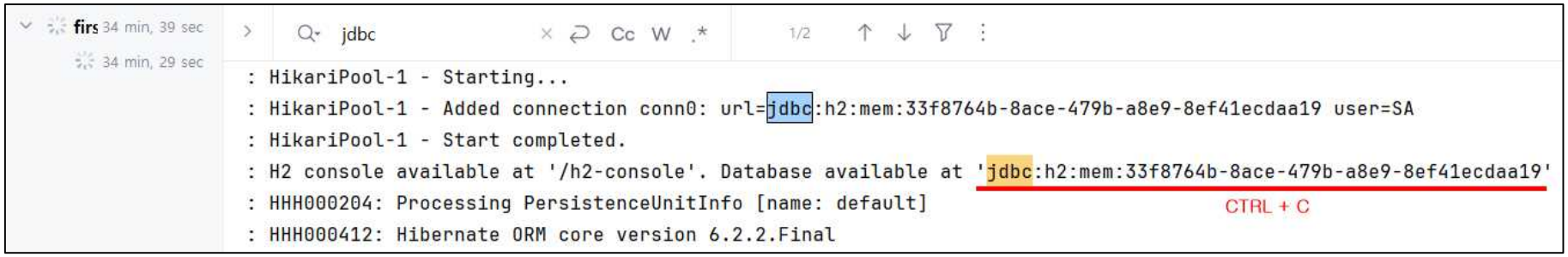
- **서버 재실행**
- **localhost:8080/articles/new 접속**
  - 새 글을 입력하고 [submit]



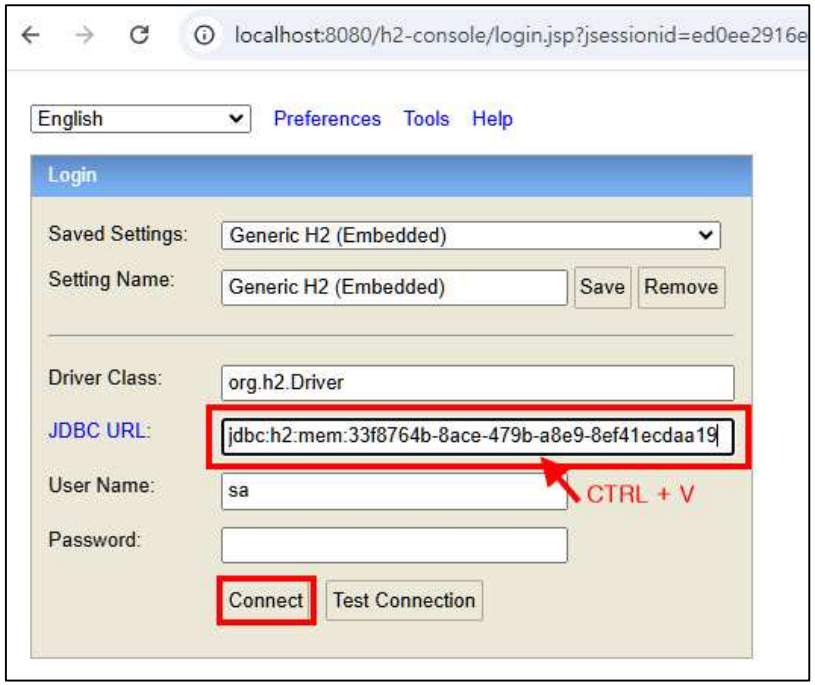


DTO 리팩터링

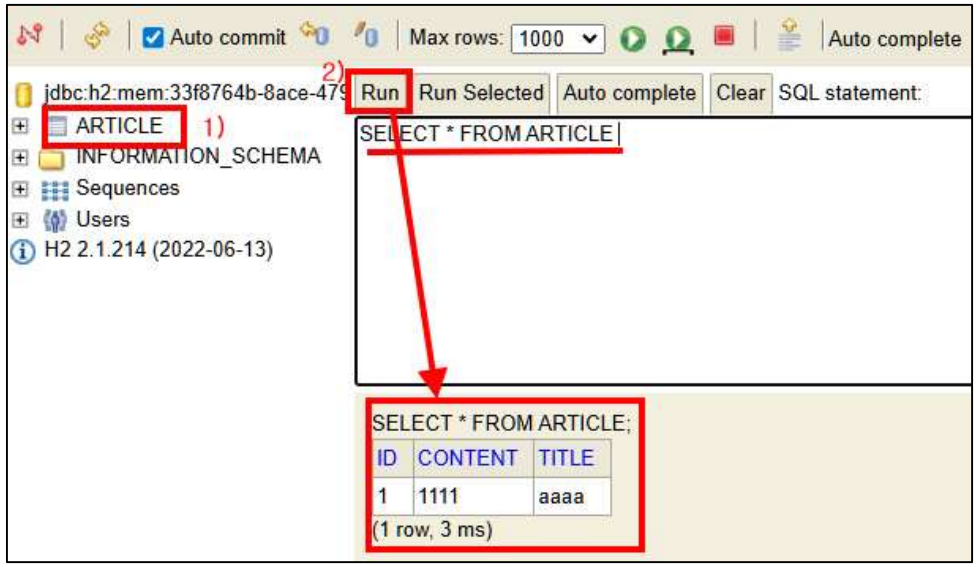
- **H2 DB 웹 콘솔 로그인**



- **localhost:8080/h2-console 접속**



Article 테이블에 데이터 삽입 확인



엔티티 리팩터링하기

- 프로젝터 탐색기에서 [com.example.firstproject – entity – Article] 열기
- Article 클래스에서 생성자와 toString()을 삭제 후, @AllArgsConstructor, @ToString 어노테이션 추가

Article.java

```
package com.example.firstproject.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.ToString;

@AllArgsConstructor
@ToString
@Entity
public class Article {
    @Id
    @GeneratedValue
    private Long id;
    @Column
    private String title;
    @Column
    private String content;

}
```

컨트롤러에 로그 남기기

- **프로젝트 탐색기에서 [com.example.firstproject-controller-ArticleController] 열기**
- **서버의 println()문으로 데이터를 검증하면 기록에 남지 않아 나중에 다시 찾아 볼 수 없고, 서버의 성능에도 악영향 ➔ 로깅 기능 사용**
  - 로깅 기능을 사용하여 서버에서 일어나는 일을 모두 기록
- **롬복의 @Slf4j 사용하여 로깅**
  - Slf4j(simple Logging Façade for java)
  - ArticleController.java에 @Slf4j 추가, log.info("문자열")추가

컨트롤러에 로그 남기기

ArticleController.java

```
import lombok.extern.slf4j.Slf4j;
        (중략)

@Slf4j
@Controller    //컨트롤러 선언
public class ArticleController {
        (중략)

    @PostMapping("/articles/create")
    public String createArticle(ArticleForm form){
        log.info(form.toString());
        //System.out.println(form.toString());
        // 1. DTO를 엔티티로 변환
        Article article = form.toEntity();
        log.info(article.toString());
        //System.out.println(article.toString());
        // 2. 리파지터리로 엔티티를 DB에 저장
        Article saved = articleRepository.save(article);
        log.info(saved.toString());
        //System.out.println(saved.toString());
        return "";
    }
}
```

컨트롤러에 로그 남기기

- **서버 재실행**
- **localhost:8080/articles/new 접속**
  - 새 글을 입력하고 [submit]

  - 로그로 찍은 결과 확인

```
2025-09-22T15:52:42.389+09:00  INFO 25744 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-09-22T15:52:42.435+09:00  INFO 25744 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 46 ms
2025-09-22T15:52:51.206+09:00  INFO 25744 --- [nio-8080-exec-2] c.e.f.controller.ArticleController       : ArticleForm(title=aaaa, content=1111)
2025-09-22T15:52:51.207+09:00  INFO 25744 --- [nio-8080-exec-2] c.e.f.controller.ArticleController       : Article(id=null, title=aaaa, content=1111)
2025-09-22T15:52:51.406+09:00  INFO 25744 --- [nio-8080-exec-2] c.e.f.controller.ArticleController       : Article(id=1, title=aaaa, content=1111)
```