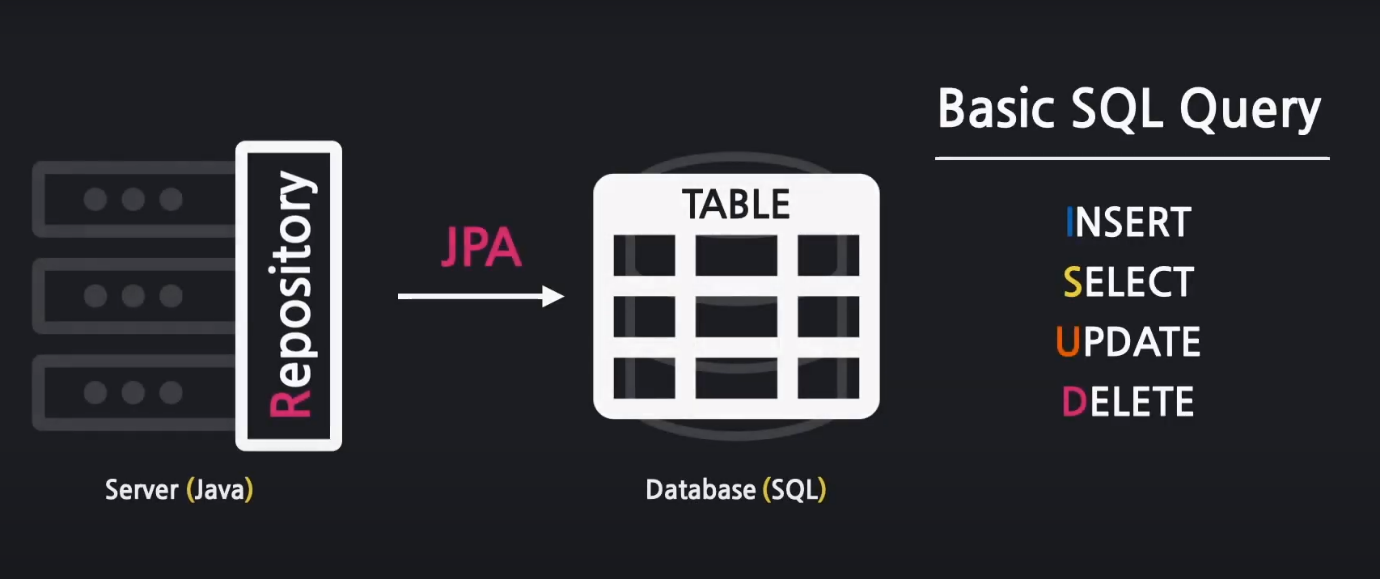
이터 CRUD와 SQL 쿼리



**JPA 로깅 설정**

application.properties 에 설정추가

# JPA 로깅 설정

# 디버그 레벨로 쿼리 출력

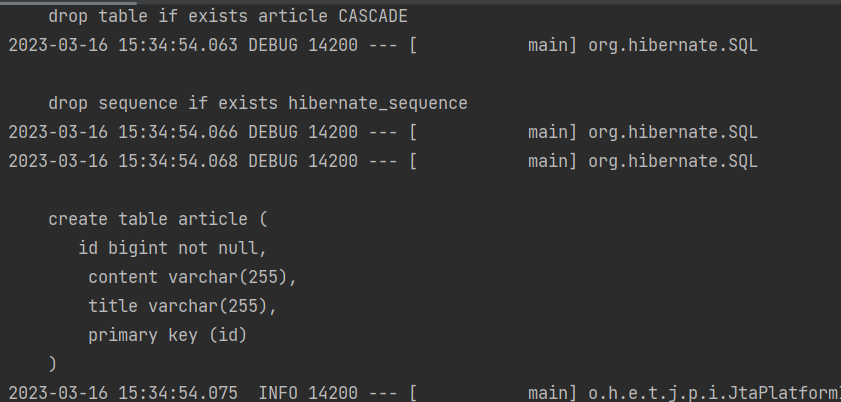
logging.level.org.hibernate.SQL=DEBUG

# 쿼리 이쁘게 보여주기

spring.jpa.properties.hibernate.format\_sql=true

# 파라미터 보여주기( ? 에 어떤 변수가 들어있는지 확인 가능)

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE



**DB URL 고정하기**

application.properties 에 설정추가

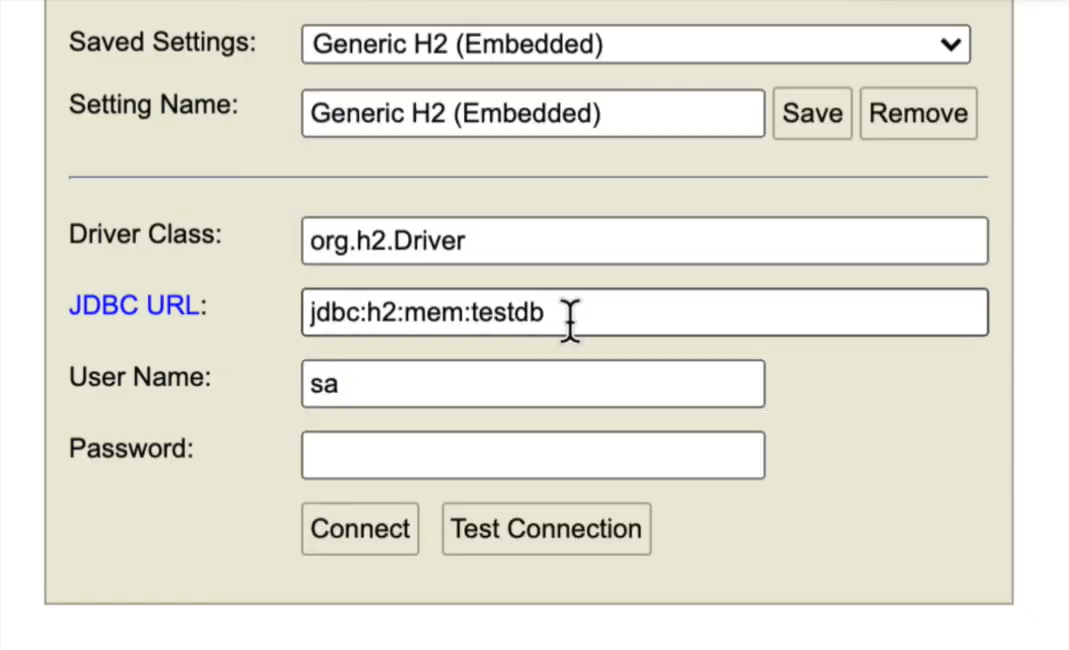
# DB URL 고정 설정

# 유니트 URL 설정 끄기

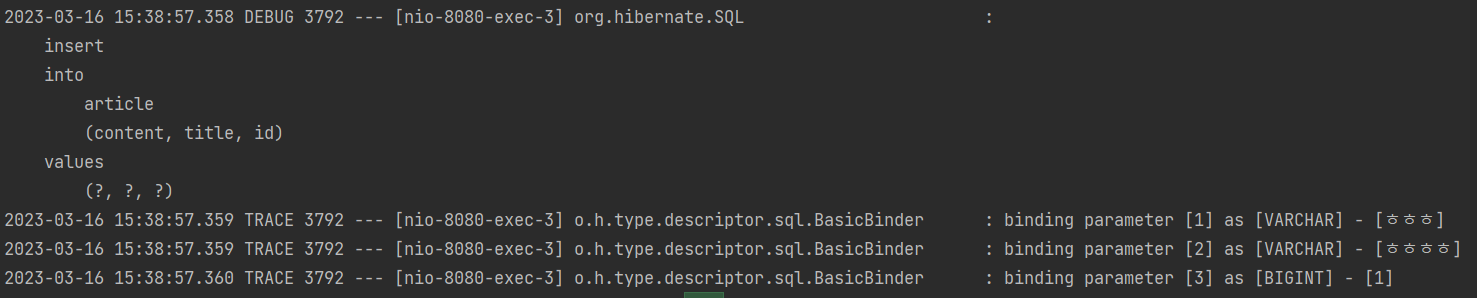
spring.datasource.generate-unique-name=false

# 고정 url 설정

spring.datasource.url=jdbc:h2:mem:testdb

이제 h2-consol에 JDBC URL에 jdbc:h2:mem:testdb 고정된 값으로 연결하면 된다.  


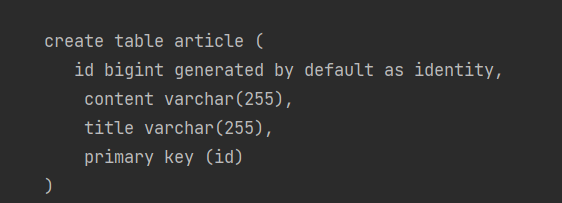
**SQL 쿼리 확인**

  
새로운 게시글을 넣었더니 id가 중복되서 에러가 발생했다.  
에러를 해결해보자.

**ID자동생성전략**

Article.java에 @GeneratedValue의 파라미터에  
@GeneratedValue(**strategy = GenerationType.IDENTITY**)를 추가하면  
DB가 id를 자동생성해준다.

**TABLE 생성**

Article.java에 @Entity 어노테이션을 추가했기때문에 테이블 자동생성.  


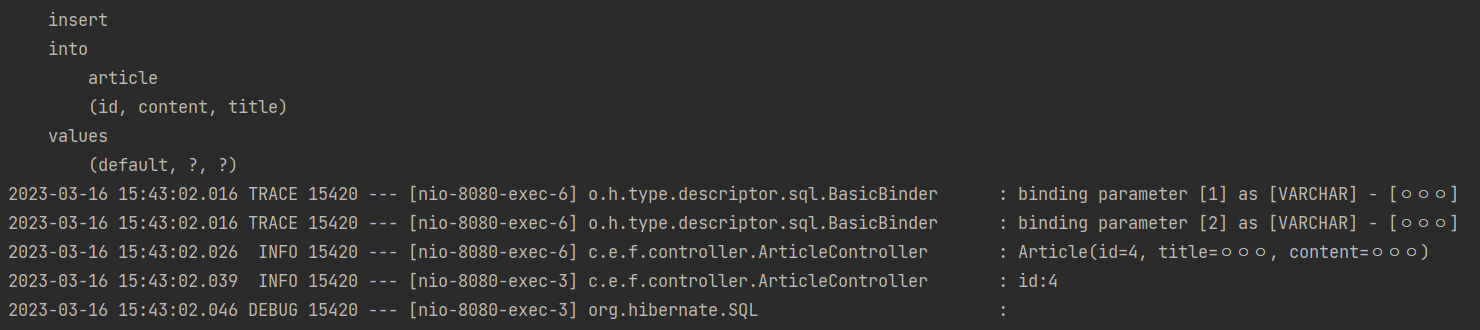
@Entity // 1.DB가 해당 객체를 인식 가능하게 됨.

public class Article {

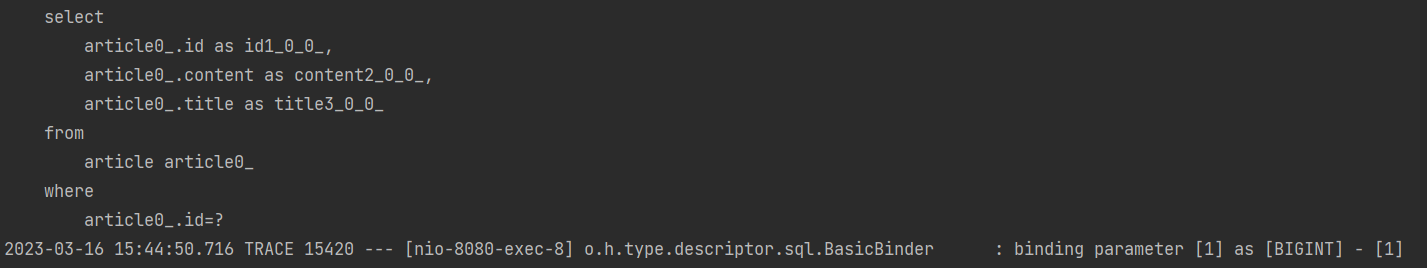
...

}

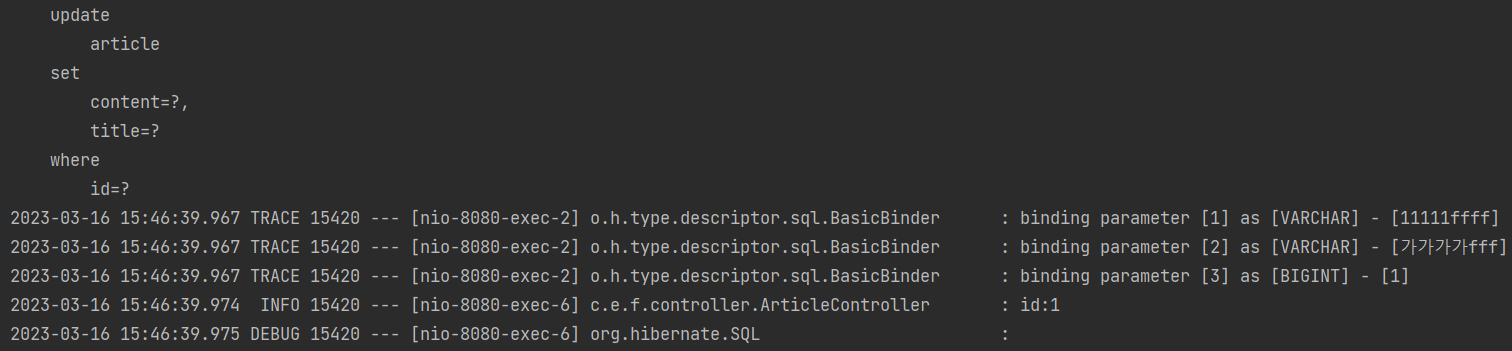
**INSERT**



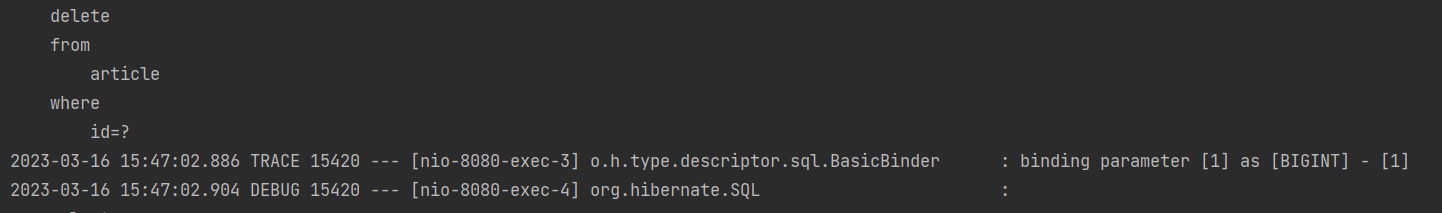
**SELECT**



**UPDATE**



**DELETE**



**ArticleController 전체코드**

package com.example.firstproject.controller;

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.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.List;

@Controller

@Slf4j // 로깅을 위한 어노테이션

public class ArticleController {

@Autowired // 스프링 부트가 미리 생성해놓은 객체를 가져다가 자동 연결!

private ArticleRepository articleRepository;

@GetMapping("articles/new")

public String newArticleForm() {

return "articles/new";

}

@PostMapping("/articles/create")

public String createArticle(ArticleForm form) {

log.info(form.toString());

// System.out.println(form.toString()); -> 로깅기능으로 대체!!

// 1. DTO -> Entity 변환

Article article = form.toEntity();

log.info(article.toString());

// System.out.println(article);

// 2. Repository 에게 Entity를 DB안에 저장하게 함

Article saved = articleRepository.save(article);

log.info(saved.toString());

// System.out.println(saved.toString());

return "redirect:/articles/" + saved.getId();

}

@GetMapping("/articles/{id}")

public String show(@PathVariable Long id, Model model) {

log.info("id:" + id);

// 1: id로 데이터를 가져옴

Article articleEntity = articleRepository.findById(id).orElse(null);

// articleRepository 리턴 타입이 Optional이기 때문에 .orElse(null) 사용

// 해당 id 값 없으면 null 반환

// 2: 가져온 데이터를 모델에 등록

model.addAttribute("article", articleEntity);

// 3: 보여줄 페이지를 설정

return "articles/show";

}

@GetMapping("/articles")

public String index(Model model) {

// 1: 모든 Article 가져오기

List<Article> articleEntityList = articleRepository.findAll();

// 2: 가져온 Article 묶음을 뷰로 전달

model.addAttribute("articleList", articleEntityList);

// 3: 뷰 페이지를 설정

return "articles/index";

}

@GetMapping("/articles/{id}/edit")

public String edit(@PathVariable Long id, Model model) {

// 수정할 데이터 가져오기

Article articleEntity = articleRepository.findById(id).orElse(null);

// 모델에 데이터 등록

model.addAttribute("article",articleEntity);

return "articles/edit";

}

// @PatchMapping

@PostMapping("/articles/update")

public String update(ArticleForm form) {

log.info(form.toString());

// 1: DTO를 Entity로 변환한다.

Article articleEntity= form.toEntity();

log.info(articleEntity.toString());

// 2: Entity를 DB로 저장한다.

// 2-1: DB에서 기존 데이터를 가져온다.

Article target = articleRepository.findById(articleEntity.getId()).orElse(null);

// 2-2: 기존 데이터에 값을 갱신한다.

if(target != null) {

articleRepository.save(articleEntity); // Entity가 DB로 갱신.

}

// 3: 수정 결과 페이지로 리다이렉트.

return "redirect:/articles/" + articleEntity.getId();

}

// @DeleteMapping("/articles/{id}/delete")

@GetMapping("/articles/{id}/delete")

public String delete(@PathVariable Long id, RedirectAttributes rttr) {

log.info("삭제 요청");

// 1: 삭제 대상을 가져온다.

Article target = articleRepository.findById(id).orElse(null);

// 2: 대상을 삭제한다.

if(target != null) {

articleRepository.delete(target);

rttr.addFlashAttribute("msg", "삭제가 완료되었습니다.");

}

// 3: 결과 페이지로 리다이렉트한다.

return "redirect:/articles";

}

}