DB 테이블과 SQL

<https://h2database.com/html/download.html>

**H2 DB 접속 설정**

**application.properties**

# ===============================  
# H2 Database ??  
# ===============================  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.url=jdbc:h2:~/test  
spring.datasource.username=sa  
spring.datasource.password=  
  
# ===============================  
# H2 Console ??  
# ===============================  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
# ===============================  
# JPA ??  
# ===============================  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true

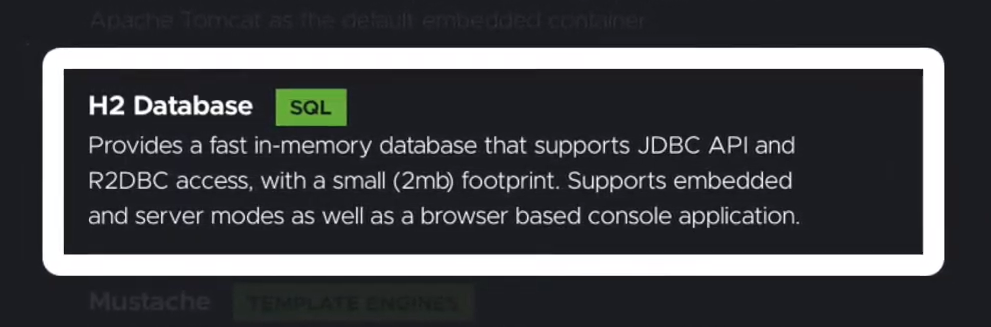
프로젝트관련 여러 설정이 들어감

# h2 DB, 웹 콘솔 접근 허용

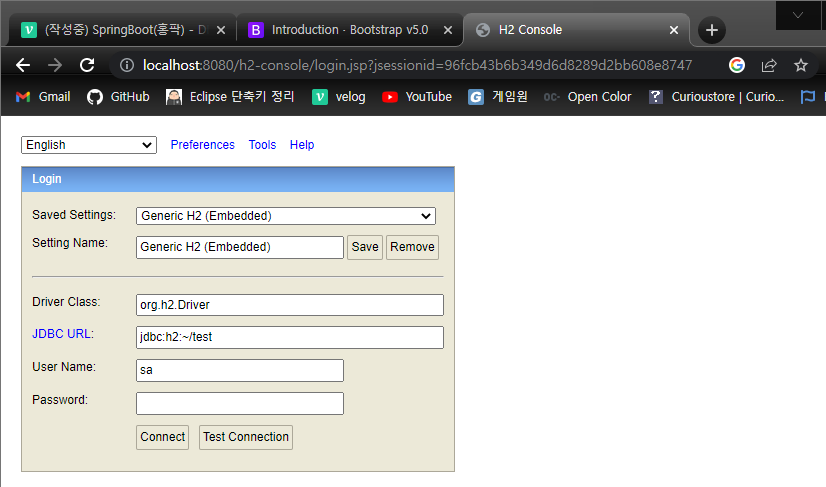
spring.h2.console.enabled = true

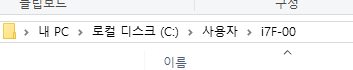
파일 저장 위치

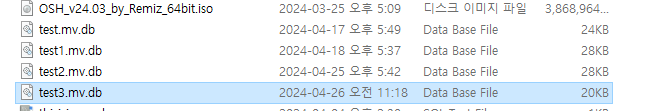
jdbc:h2:~/test는 OS 계정의 홈 디렉토리에 test.mv.db 파일을 생성합니다.

예: C:\Users\tj\test.mv.db

Spring initializr에서 설정했던 H2 Database 설정

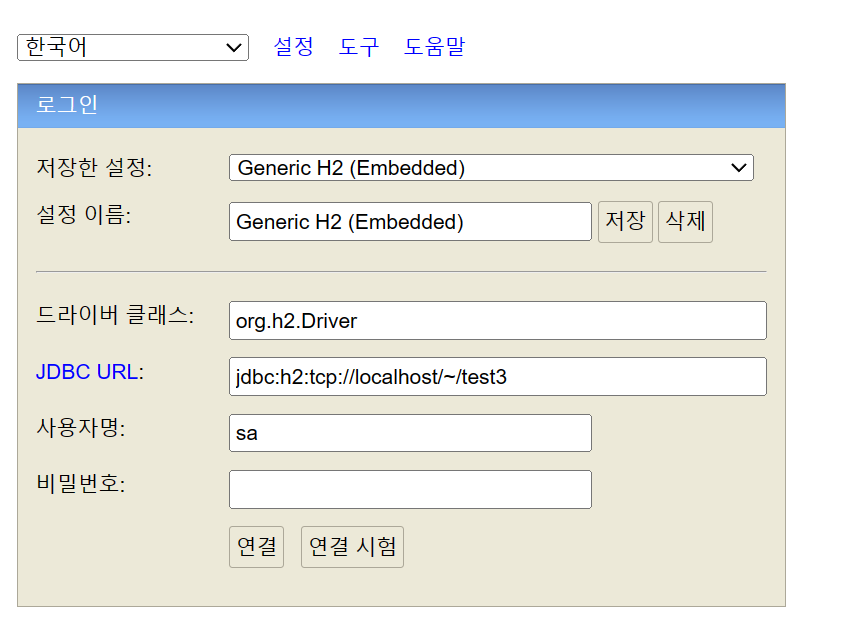
**H2 DB 접속** http://localhost:8080/h2-console





db파일이 만들어져있는지 확인

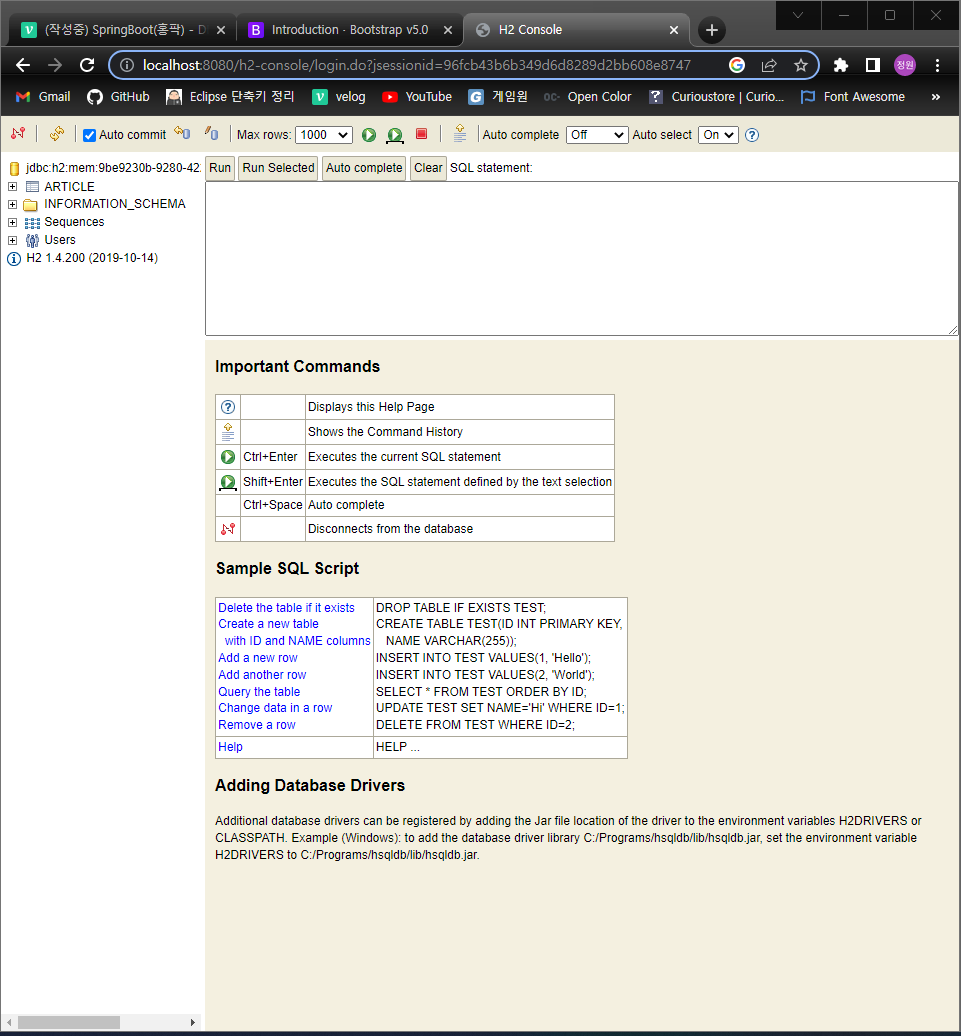
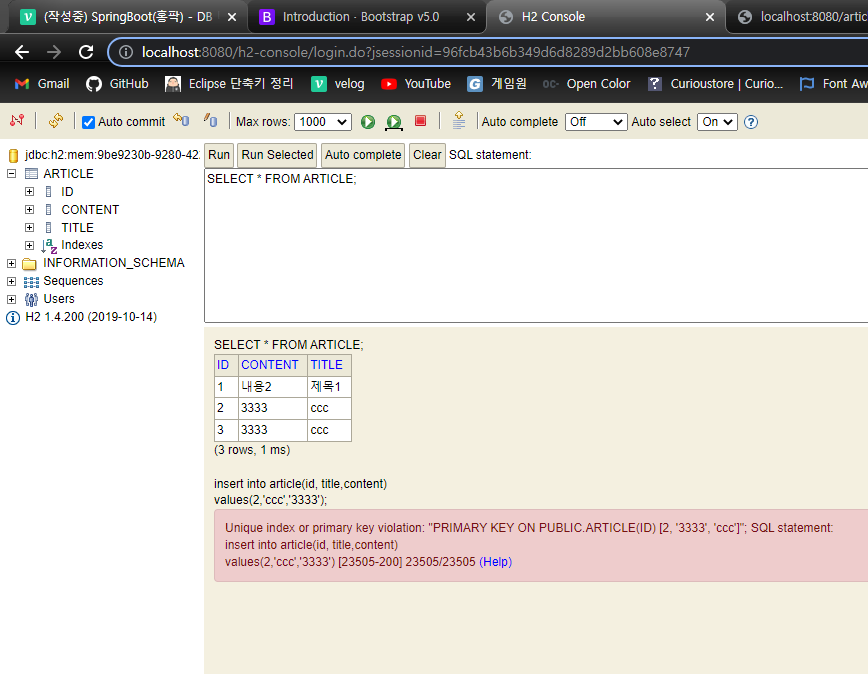
아래 url로 다시 접속



Connect 누르면 연결된다.

Application.properties추가

#h2  
spring.h2.console.enabled=true  
spring.datasource.url=jdbc:h2:tcp://localhost/~/test3  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.username=sa  
  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create

**롬복과 리팩터링**

**롬복 설치**

build.gradle안에 dependencies안에 롬복 추가

dependencies {

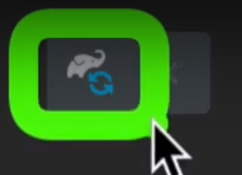
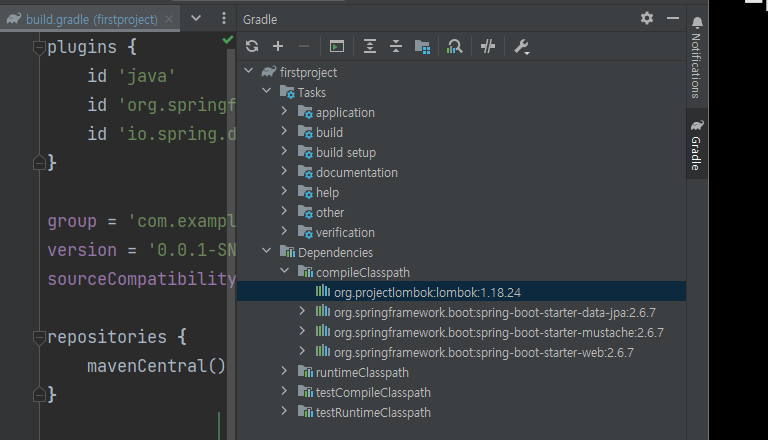
// 롬복 추가

compileOnly 'org.projectlombok:lombok'

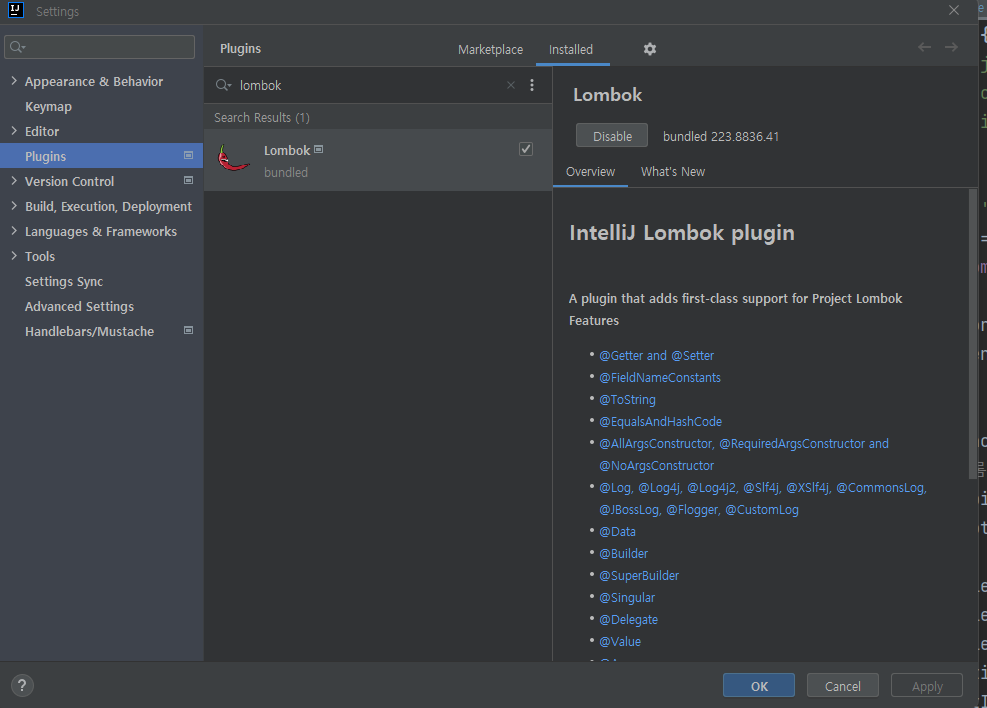
annotationProcessor 'org.projectlombok:lombok'

...

}

dependencies추가 후 코끼리 모양 눌러서 빌드하기!  
  
lombok 추가 완료 ✨  


**롬복 플러그인 설치**

최신 인텔리제이에는 롬복 플러그인이 이미 설치 되어있음 😊  


**DTO,Entity 리팩토링**

생성자와 toString()를 lobom 어노테이션으로 대체 👍  
@AllArgsConstructor  
@ToString

@AllArgsConstructor

@ToString

public class ArticleForm {

private String title;

private String content;

public Article toEntity() {

return new Article(null, title, content);

}

}

@AllArgsConstructor

@ToString

@Entity

public class Article {

@Id

@GeneratedValue

private Long id;

@Column

private String title;

@Column

private String content;

}

**로그 남기기**

현재는 컨트롤러에서 sysout으로 확인하고 있는데  
실제 서버에서는 sysout으로 사용하면 안된다. 😱

컨트롤러에 @Slf4j 어노테이션 붙이고  
sysout -> log.info()로 변환한다.

@Controller

@Slf4j // 로깅을 위한 어노테이션

public class ArticleController {

...

@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 "";

}

}

콘솔에서 확인 가능.  
