

## 3장. 스프링부트 JPA



*ORM 하이버네이트*



Spring Boot

# 스프링과 JPA

## ▪ 스프링과 JPA

데이터 베이스 연동에 사용되는 기술은 전통적인 JDBC, 스프링 MyBatis, 하이버네이트 ORM(Object Relational Mapping)에 이르기 까지 다양하다.

이 중에서 하이버네이트 ORM은 애플리케이션에서 사용하는 SQL까지도 프레임워크에서 제공하기 때문에 개발자가 처리해야 할 일들을 많이 줄여준다.

ORM이란 "객체지향 구조를 관계형 구조로 매핑"하는 기술이다.

이런 **ORM을 보다 쉽게 사용할 수 있도록 표준화 시킨 것이 JAP(Java Persistence API)** 이다  
다시 말하면 ORM 기술을 Java 언어에 맞도록 스펙으로 정리한 것이라 할 수 있다..

JAP가 제공하는 인터페이스를 이용하여 데이터베이스를 처리하면 실제로는 JPA를 구현한 구현체가 동작하는 것이다.

JPA 구현체는 하이버네이트, EclipseLink, DataNucleus 등 여러가지가 있는데 **스프링 부트**에서는 기본적으로 하이버네이트를 JPA 구현체로 이용한다.



# 스프링과 JPA

- SQL을 직접 다루는 기술

애플리케이션은 사용자가 입력한 데이터나 운용 과정에서 생성된 데이터를 재사용하기 위해 데이터베이스 같은 저장공간에 저장해야 한다. 이 때 SQL이 사용된다.

## 오라클 DB 테이블

```
CREATE TABLE board(  
    seq NUMBER(5) PRIMARY KEY,  
    title VARCHAR2(200),  
    writer VARCHAR2(20),  
    content VARCHAR2(2000),  
    regdate DATE DEFAULT SYSDATE,  
    cnt NUMBER(5) DEFAULT 0  
);
```

## VO 클래스

```
@Getter  
@Setter  
public class BoardVO {  
    private int seq;  
    private String title;  
    private String writer;  
    private String content;  
    private Date createDate;  
    private int cnt = 0;  
}
```

// 글 등록

```
INSERT INTO board(seq, title, writer, content) VALUES(seq.nextval, ?, ?, ?, ?);
```

//글 목록 조회

```
SELECT * FROM board;
```



# 스프링과 JPA

- SQL을 직접 다루지 않는 기술

BoardVO를 Map에 저장하고 관리했을때의 CRUD 코드

```
Map<String, BoardVO> boardList = new HashMap<>();

BoardVO board = new BoardVO();
board.setSeq(1);
board.setTitle("테스트 제목...");
board.setWriter("테스터");
board.setContent("테스트 내용입니다...");
board.setCreateDate(new Date());
board.setCnt(0);

//게시글 등록
boardList.put("board", board);
```

BoardVO 객체를 Map에 저장했기 때문에 소스 어디에도 BOARD 테이블(DB)과 관련된 SQL이 사용되지 않는다.



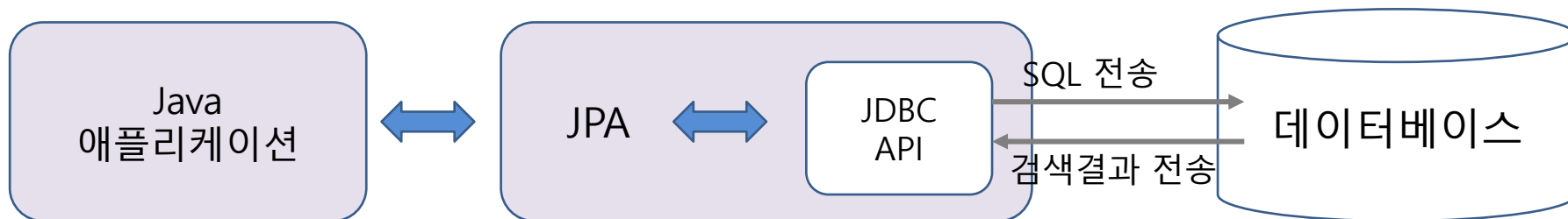
# 스프링과 JPA

## ▪ JPA 동작 원리

JPA는 자바 객체를 컬렉션에 저장하고 관리하는 것과 비슷한 개념이다.

하지만 결국 컬렉션에 저장된 객체를 테이블의 로우와 매핑하기 위해서는 누군가가 JDBC API를 이용해서 실질적인 연동 작업을 처리해야 한다.

JPA는 자바 애플리케이션과 JDBC 사이에 존재하면서 JDBC의 복잡한 절차를 대신 처리해 준다.



JPA가 데이터베이스 연동에 사용되는 코드 뿐만 아니라 SQL까지도 제공한다.

테이블과 VO 클래스 이름을 똑같이 매핑하고, 테이블의 칼럼을 VO 클래스의 멤버 변수와 매핑하여 동작한다.



# 스프링과 JPA

## ▪ H2 데이터 베이스 사용하기



### H2 Database Engine

Welcome to H2, the Java SQL database. The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes; in-memory databases
- Browser based Console application
- Small footprint: around 2.5 MB jar file size

#### Download

Version 2.1.214 (2022-06-13)





-  [Windows Installer \(6.7 MB\)](#)
-  [All Platforms \(zip, 9.5 MB\)](#)
- [All Downloads](#)

#### Support

[Stack Overflow \(tag H2\)](#)

[Google Group](#)

For non-technical issues, use:  
dbsupport at h2database.com

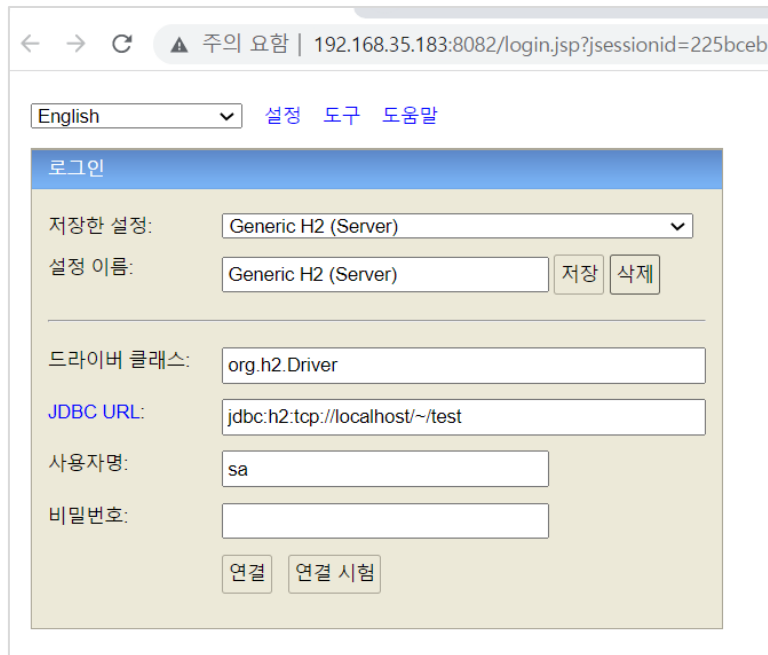
Dev > h2-2022-06-13 > h2 > bin	
이름	수정한 날짜
 h2.bat	2022-06-13 오후 9:34
 h2.sh	2022-06-13 오후 9:34
 h2-2.1.214.jar	2022-06-13 오후 9:34
 h2w.bat	2022-06-13 오후 9:34

H2 설치 폴더 -> bin -> h2w.bat 파일 실행



# 스프링과 JPA

## ▪ H2 데이터 베이스 사용하기



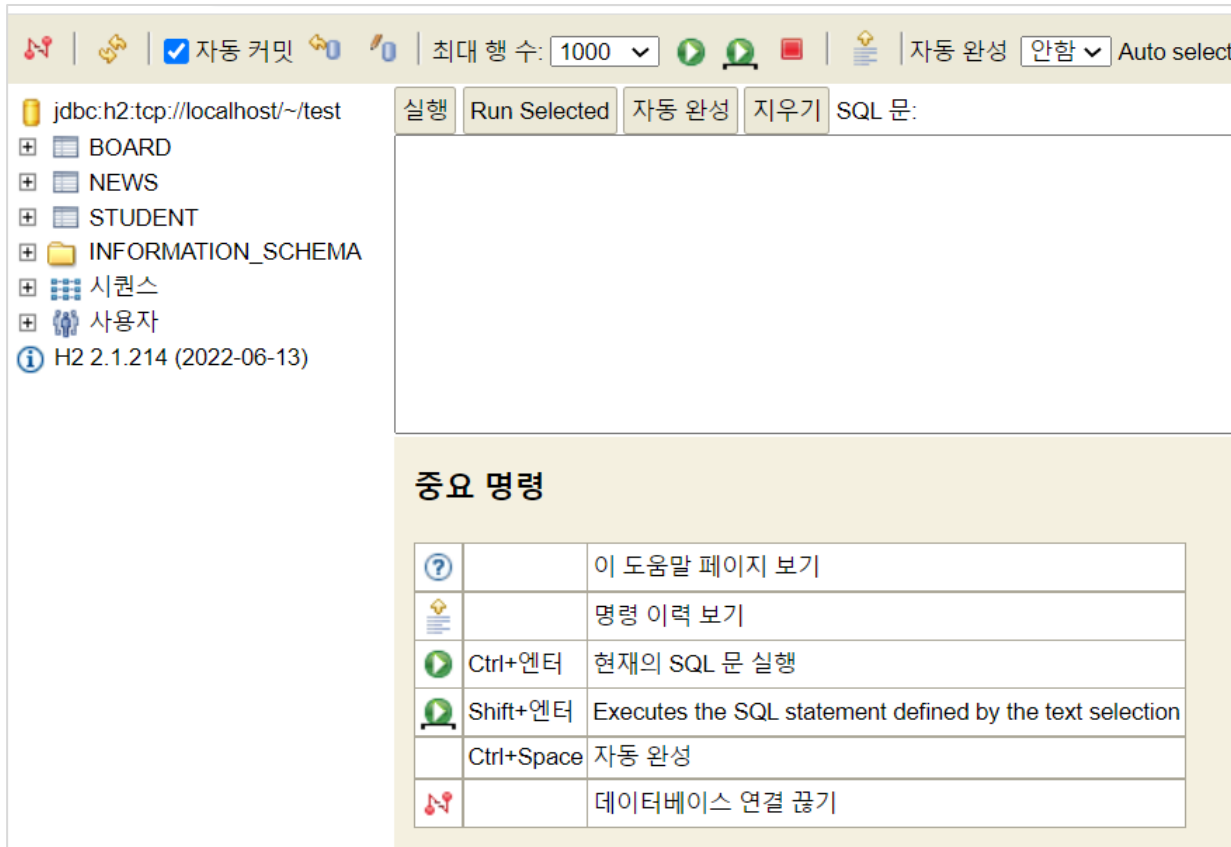
The screenshot shows the H2 database web interface. At the top, there's a navigation bar with 'English' and links for '설정' (Settings), '도구' (Tools), and '도움말' (Help). Below this is a '로그인' (Login) section. It contains several fields: '저장한 설정:' (Saved settings) with a dropdown menu showing 'Generic H2 (Server)'; '설정 이름:' (Setting name) with a text input field also containing 'Generic H2 (Server)' and buttons for '저장' (Save) and '삭제' (Delete); '드라이버 클래스:' (Driver class) with a text input field containing 'org.h2.Driver'; 'JDBC URL:' with a text input field containing 'jdbc:h2:tcp://localhost/~ /test'; '사용자명:' (Username) with a text input field containing 'sa'; and '비밀번호:' (Password) with an empty text input field. At the bottom of the login section are buttons for '연결' (Connect) and '연결 시험' (Test connection).

항목	설정 값
driverClass	org.h2.Driver
jdbc url	jdbc:h2:tcp://localhost/~ /test
사용자명	sa
비밀번호	없음



# 스프링과 JPA

## ▪ H2 데이터 베이스 사용하기



The screenshot shows the H2 Database console interface. The top toolbar includes icons for connection, schema, auto-commit, and execution, along with a dropdown for '최대 행 수' (Maximum rows) set to 1000. The left sidebar displays the database structure: jdbc:h2:tcp://localhost/~test, BOARD, NEWS, STUDENT, INFORMATION\_SCHEMA, 시퀀스 (Sequences), 사용자 (Users), and H2 2.1.214 (2022-06-13). The main area contains buttons for '실행' (Execute), 'Run Selected', '자동 완성' (Auto complete), and '지우기' (Clear), followed by a text input for 'SQL 문:' (SQL statement). Below this is a section titled '중요 명령' (Important commands) with a table of shortcuts.

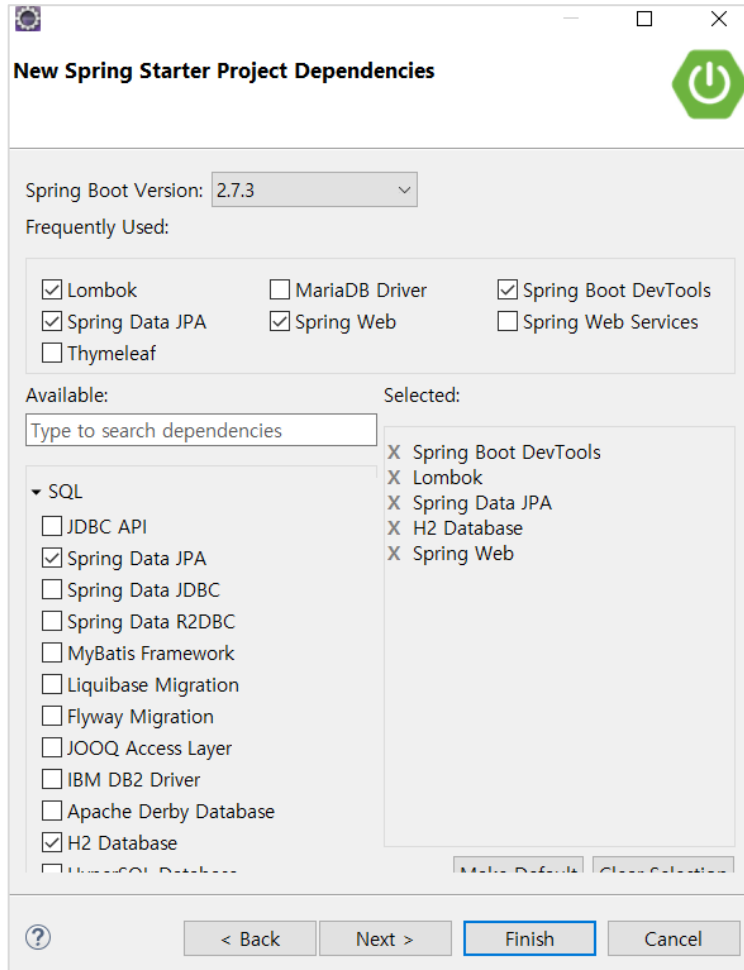
Icon	Shortcut	Description
?		이 도움말 페이지 보기
📄		명령 이력 보기
▶	Ctrl+엔터	현재의 SQL 문 실행
▶	Shift+엔터	Executes the SQL statement defined by the text selection
	Ctrl+Space	자동 완성
🔌		데이터베이스 연결 끊기





# 스프링 데이터 JPA

## ■ 프로젝트 생성 및 기본 설정



New Spring Starter Project Dependencies

Spring Boot Version: 2.7.3

Frequently Used:

- ☒ Lombok
- ☐ MariaDB Driver
- ☒ Spring Boot DevTools
- ☒ Spring Data JPA
- ☒ Spring Web
- ☐ Spring Web Services
- ☐ Thymeleaf

Available:

Type to search dependencies

- SQL
  - ☐ JDBC API
  - ☒ Spring Data JPA
  - ☐ Spring Data JDBC
  - ☐ Spring Data R2DBC
  - ☐ MyBatis Framework
  - ☐ Liquibase Migration
  - ☐ Flyway Migration
  - ☐ JOOQ Access Layer
  - ☐ IBM DB2 Driver
  - ☐ Apache Derby Database
  - ☒ H2 Database
  - ☐ Microsoft SQL Database

Selected:

- X Spring Boot DevTools
- X Lombok
- X Spring Data JPA
- X H2 Database
- X Spring Web

< Back Next > Finish Cancel

[New] -> [Spring Starter Project]

Name – SpringJPA,

Packing – Jar

Java Version – 8

Group – com.boot,

Package – com.boot

- 모듈 추가

DevTools, Lombok, Spring Web,

Spring Data JPA, H2 Database



# 스프링 데이터 JPA

## 프로젝트 생성 및 기본 설정

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

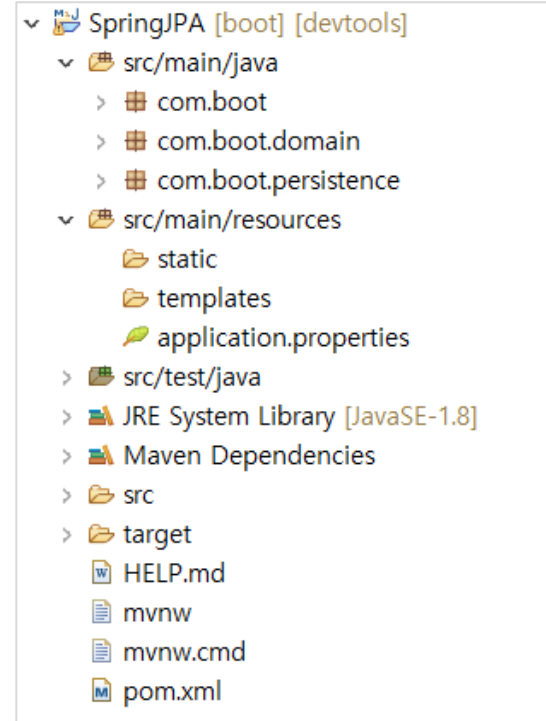
Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



# 스프링 데이터 JPA

- JPA 기본 설정

```
# DataSource Setting
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:tcp://localhost/~ /test
spring.datasource.username=sa
spring.datasource.password=

# JPA Setting
spring.jpa.hibernate.ddl-auto=create
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true
```

application.properties



# 스프링 데이터 JPA

## ■ 엔티티 매핑과 리포지터리 작성

어노테이션	의미
@Entity	@Entity가 설정된 클래스를 엔티티라 하며, 기본적으로 클래스 이름과 동일한 테이블과 매핑된다.
@Table	엔티티 이름과 매핑될 테이블 이름이 다른 경우 name 속성을 사용하여 매핑한다. 엔티티 이름과 테이블 이름이 동일하면 생략해도 됨
@Id	테이블의 기본 키를 매핑한다.
@GeneratedValue	@Id가 선언된 필드에 기본 키 값을 자동으로 할당한다.



# 스프링 데이터 JPA

- 엔티티 매핑과 리포지터리 작성

- (1) 엔티티 클래스 매핑

```
@ToString
@Setter
@Getter
@Entity
public class Board {

    @Id@GeneratedValue
    private Long seq;

    private String title;
    private String writer;
    private String content;

    private Date createDate;
    private Long cnt;
}
```



# 스프링 데이터 JPA

- 엔티티 매핑과 리포지터리 작성

- (2) 테이블 생성 확인

웹 애플리케이션이 아닌 일반 자바 애플리케이션으로 실행  
내장 톰캣을 구동하지 않고 실행됨

```
@SpringBootApplication
public class SpringJpaApplication {

    public static void main(String[] args) {
        //SpringApplication.run(SpringJpaApplication.class, args);
        SpringApplication application =
            new SpringApplication(SpringJpaApplication.class);

        application.setWebApplicationType(WebApplicationType.NONE);
        application.run(args);
    }
}
```



# 스프링 데이터 JPA

- 엔티티 매핑과 리포지터리 작성

- (2) 테이블 생성 확인

```
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
Hibernate:
```

```
create table board (
  seq bigint not null,
  cnt bigint,
  content varchar(255),
  create_date timestamp,
  title varchar(255),
  writer varchar(255),
  primary key (seq)
)
```



# 스프링 데이터 JPA

## ■ 엔티티 매핑과 리포지터리 작성

### (2) 테이블 생성 확인

The screenshot shows the H2 database console interface. The left pane displays the database structure for 'jdbc:h2:top://localhost/~test'. A red dashed box highlights the 'NEWS' table. The right pane shows the 'SQL 문:' (SQL Statement) area, which is currently empty. Below the SQL area, there is a '중요 명령' (Important Commands) section with a table of shortcuts.

Icon	Shortcut	Description
?		이 도움말 페이지 보기
📋		명령 이력 보기
▶	Ctrl+엔터	현재의 SQL 문 실행
▶	Shift+엔터	Executes the SQL statement defined by the text selection
	Ctrl+Space	자동 완성
🔌		데이터베이스 연결 끊기

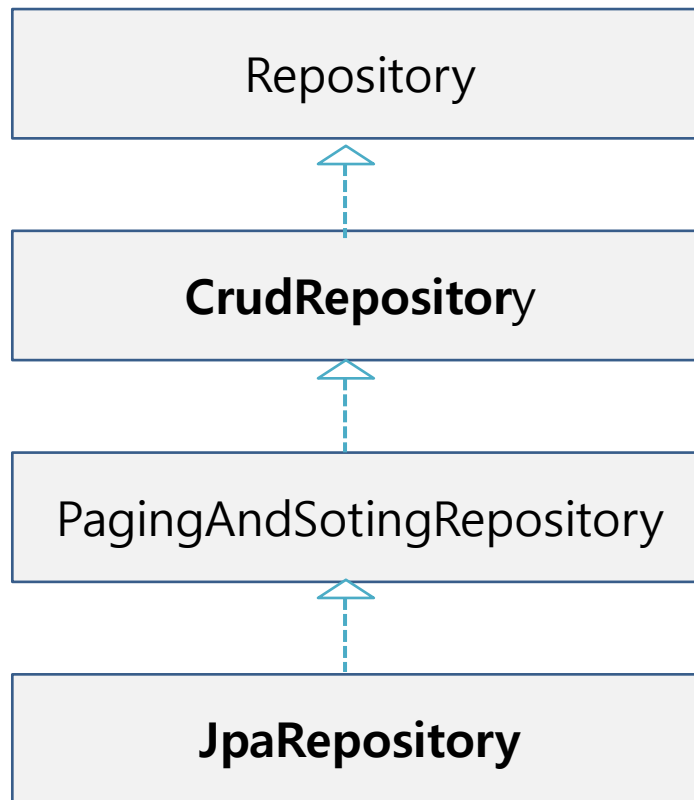




# 스프링 데이터 JPA

- 엔티티 매핑과 리포지터리 작성

(3) Repository 인터페이스 작성



# 스프링 데이터 JPA

- 엔티티 매핑과 리포지터리 작성

- (3) Repository 인터페이스 작성

CrudRepository<T, ID>

T : 엔티티의 클래스 타입

ID : 식별자(PK) 타입(@Id로 매핑한 식별자 변수의 자료형)

```
package com.boot.persistence;

import org.springframework.data.repository CrudRepository;

public interface BoardRepository extends CrudRepository<Board, Long>{

}
```

별도의 구현 클래스를 만들지 않고 인터페이스만 정의함으로써 기능을 사용할 수 있음



# 스프링 데이터 JPA

- 테스트 코드를 통한 CRUD

작업	메서드
INSERT	save(엔티티 객체)
SELECT	findById(키 타입), get()
	findAll() - 목록
UPDATE	save(엔티티 객체)
DELETE	deleteById(키 타입), delete(객체 타입)



# 스프링 데이터 JPA

- CRUD 기능 테스트

- (1) 등록 기능 테스트 – save() 메서드 사용

```
@SpringBootTest
public class BoardRepositoryTest {

    @Autowired
    private BoardRepository boardRepo;

    @Test
    public void testInsertBoard() {
        Board board = new Board();
        board.setTitle("첫 번째 게시글");
        board.setWriter("테스터");
        board.setContent("등록이 잘 되네요..");
        board.setCreateDate(new Date());
        board.setCnt(0L);

        boardRepo.save(board); //save() 메서드로 DB에 저장함
    }
}
```



# 스프링 데이터 JPA

- CRUD 기능 테스트

- (1) 등록 기능 테스트

- 실행전 테이블 자동생성 기능을 update로 변경
    - 실행전 H2 DB 접속을 반드시 해야함

```
# JPA Setting
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true
```

```
Hibernate:
    call next value for hibernate_sequence
Hibernate:
    insert
    into
        board
        (cnt, content, create_date, title, writer, seq)
    values
        (?, ?, ?, ?, ?, ?)
```



# 스프링 데이터 JPA

- CRUD 기능 테스트

- (1) 등록 기능 테스트

The screenshot shows a database management interface with a left sidebar containing a tree view of database objects: jdbc:h2:tcp://localhost/~/test, BOARD, NEWS, STUDENT, INFORMATION\_SCHEMA, 시퀀스, 사용자, and H2 2.1.214 (2022-06-13). The main area has a toolbar with buttons: 실행, Run Selected, 자동 완성, 지우기, and SQL 문:. Below the toolbar is a text input field containing the SQL query: SELECT \* FROM board;. Below the input field is a yellow banner displaying the executed query: SELECT \* FROM board;. At the bottom is a table showing the result of the query.

SEQ	CNT	CONTENT	CREATE_DATE	TITLE	WRITER
1	0	등록이 잘 되네요..	2022-09-17 11:18:55.119	첫 번째 게시물	테스터



# 스프링 데이터 JPA

- CRUD 기능 테스트

(2) 상세 조회 기능 테스트 – findById(seq) 사용

```
//상세 조회
@Test
public void testGetBoard() {
    Board board = boardRepo.findById(1L).get();
    Log.info(board.toString());
}
```

```
Board(seq=1, title=첫 번째 게시글, writer=테스터, content=등록이 잘 되네요..,
Closing JPA EntityManagerFactory for persistence unit 'default'
HikariPool-1 - Shutdown initiated...
```



# 스프링 데이터 JPA

- CRUD 기능 테스트

(3) 글 수정 기능 테스트 – findById(seq)로 조회후 save()로 재등록

```
//글 수정
@Test
public void testUpdateBoard() {
    Log.info("2번 게시글 조회");
    Board board = boardRepo.findById(2L).get();

    Log.info("2번 게시글 제목 수정");
    board.setTitle("제목을 수정합니다");
    boardRepo.save(board);
}
```

select \* from board;

SEQ	CNT	CONTENT	CREATE_DATE	TITLE	WRITER
1	0	등록이 잘 되네요..	2022-09-17 16:02:13.393	첫 번째 게시글	테스터
2	0	등록이 잘 되네요..	2022-09-17 16:02:42.378	제목을 수정합니다	테스터

(2 행, 1 ms)

```
Hibernate:
    update
        board
    set
        cnt=?,
        content=?,
        create_date=?,
        title=?,
        writer=?
    where
        seq=?
```





# 스프링 데이터 JPA

- CRUD 기능 테스트

- (4) 글 삭제 기능 테스트 – deleteById(seq) 사용

```
//글 삭제
@Test
public void testDeleteBoard() {

    Log.info("1번 게시글 삭제");

    boardRepo.deleteById(1L);
}
```

Hibernate:  
delete  
from  
board  
where  
seq=?

select \* from board;

SEQ	CNT	CONTENT	CREATE_DATE	TITLE	WRITER
2	0	등록이 잘 되네요..	2022-09-17 16:02:42.378	제목을 수정합니다	테스터

(1 row, 2 ms)



# 스프링 데이터 JPA

- 쿼리 메소드 사용하기
  - (1) 쿼리 메소드란?



# 스프링 데이터 JPA

- 쿼리 메소드 사용하기

```
public interface BoardRepository extends CrudRepository<Board, Long>{  
    List<Board> findByTitle(String searchKeyword);  
}
```



# 스프링 데이터 JPA

- 쿼리 메소드 사용하기

```
@Slf4j
@SpringBootTest
public class QueryMethodTest {

    @Autowired
    private BoardRepository boardRepo;

    //데이터 200개 저장
    @BeforeEach
    public void dataPrepare() {
        for(int i=1; i<=200; i++) {
            Board board = new Board();
            board.setTitle("테스트 제목" + i);
            board.setWriter("테스터");
            board.setContent("테스트 내용 " + i);
            board.setCreateDate(new Date());
            board.setCnt(0L);

            boardRepo.save(board);
        }
    }
}
```



# 스프링 데이터 JPA

- 쿼리 메소드 사용하기

```
@Test
public void testFindByTitle() {
    List<Board> boardList = boardRepo.findByTitle("테스트 제목10");

    Log.info("검색 결과");
    for(Board board : boardList) {
        Log.info("--->" + board.toString());
    }
}
```

검색 결과

--->Board(seq=12, title=테스트 제목10, writer=테스터, content=테스트 내용 10,

