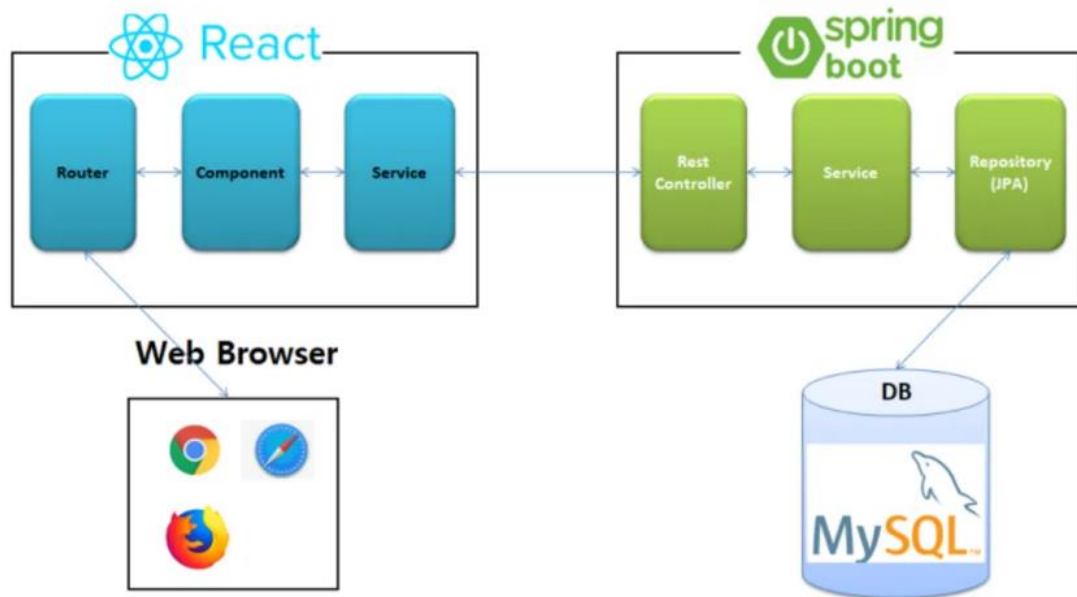


게시판 CRUD with React + Spring Boot + MySQL

Overview



목표

React 와 Spring Boot 프로젝트 진행 시 CRUD 기능과 API 주고받는 방법(with Axios)

JPA 를 활용한 MySQL 다루기

CRUD 기능

게시글 생성 (Create) → **POST | /api/create-board**

게시글 전체 보기 (Read) → **GET | /api/board-list**

게시글 상세 보기 (Read) → **GET | /api/board-detail/{ boardId }**

게시글 수정 (Update) → **PUT | /api/update-board**

게시글 삭제 (Delete) → **DELETE | /api/delete-board**

프로젝트 생성(**board_crud**)

Project

☒ Gradle - Groovy
 ☐ Gradle - Kotlin
 ☐ Maven

Language

☒ Java
 ☐ Kotlin
 ☐ Groovy

Spring Boot

☐ 3.4.2 (SNAPSHOT)
 ☐ 3.4.1
 ☐ 3.3.8 (SNAPSHOT)
 ☒ 3.3.7

Project Metadata

Group

spring_study

Artifact

board_crud

Name

board_crud

Description

Demo project for Spring Boot

Package name

spring_study.board_crud

Packaging

☒ Jar
 ☐ War

Java

☐ 23
 ☐ 21
 ☒ 17

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA

SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Lombok

DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

MySQL Driver

SQL

MySQL JDBC driver.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

...

JPA vs Hibernate vs Spring Data JPA

JPA 에서의 영속성

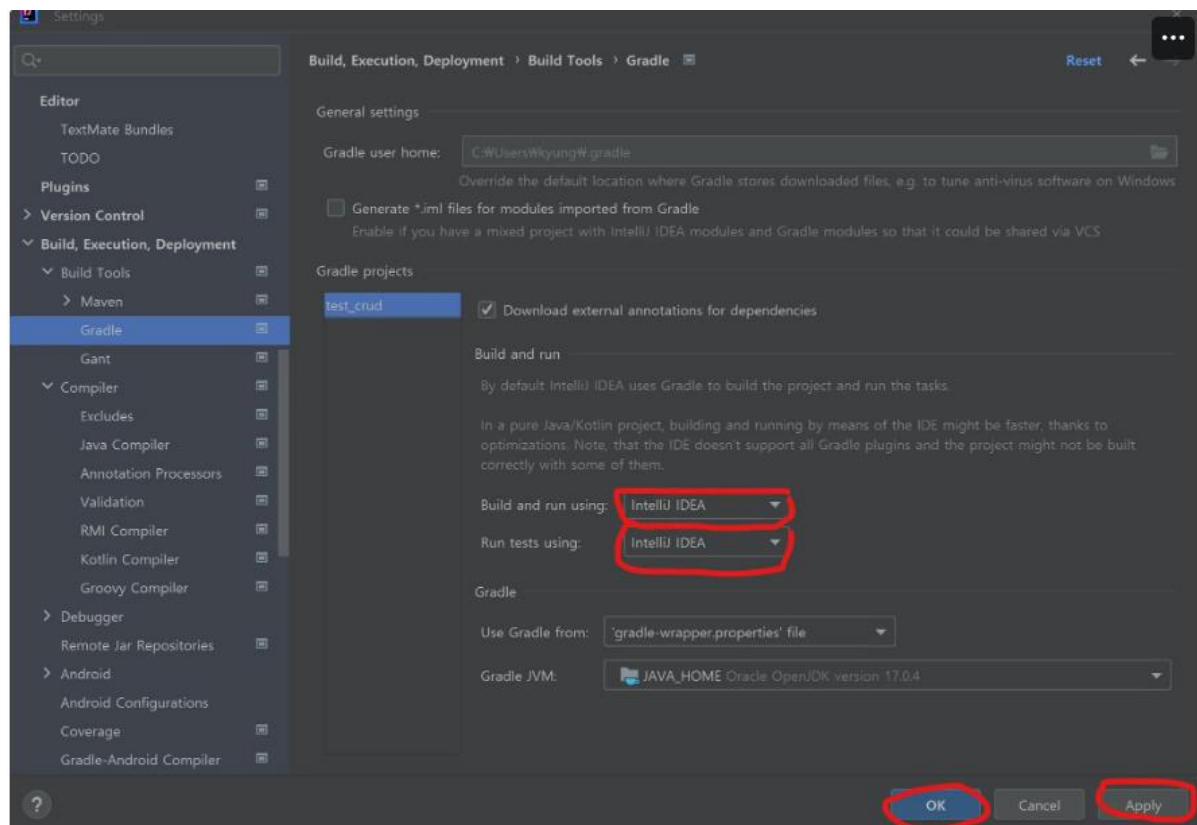
프로젝트 실행

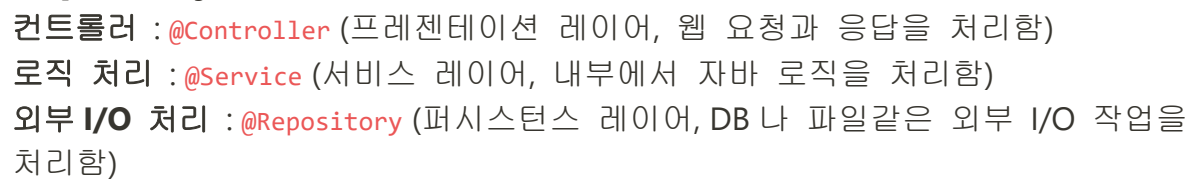
Opne with IntelliJ

Main class JDK 설정

Build Tool 설정

finished with non-zero exit value 1





Repository 란



Domain



Table

Repository



SQL

MySQL 설치

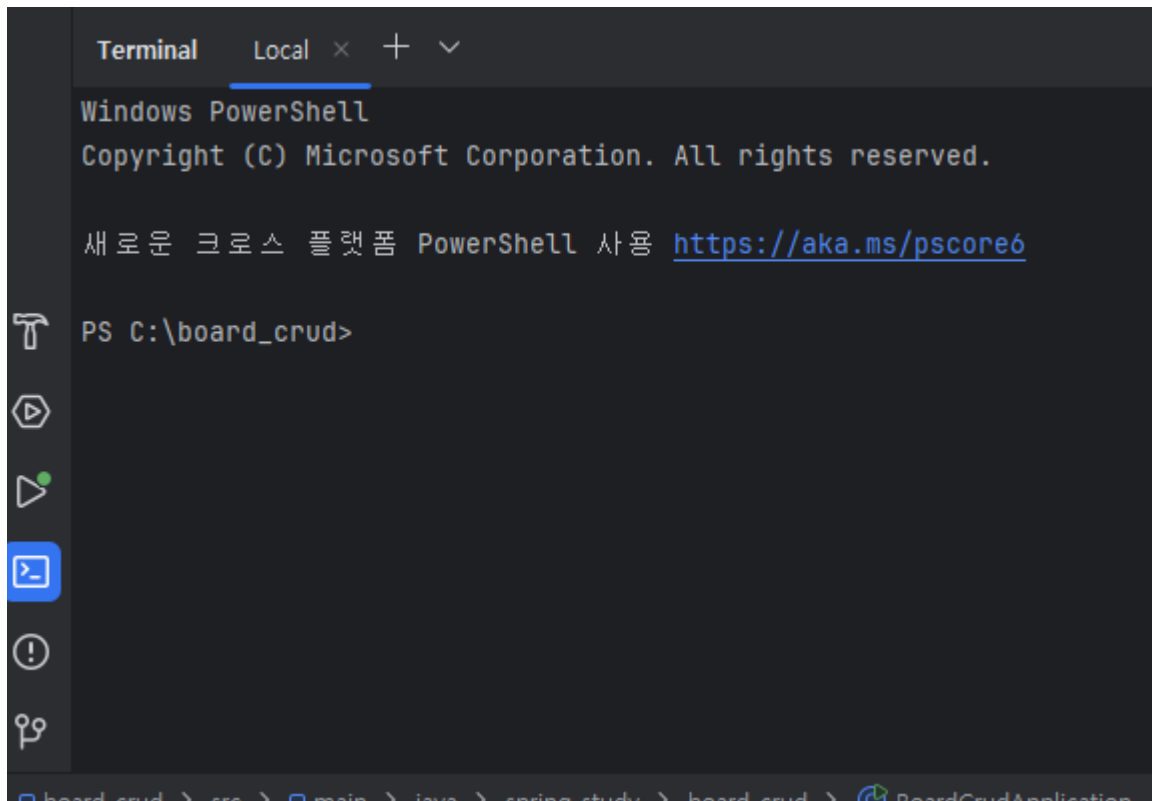
board 라는 schema 를 만들어주자!

이후 java 앱을 실행하면 위와 같이 select 쿼리를 날릴 수 있다.

Why React?



하단 왼쪽 그림같이 생긴 Terminal 클리후 아래 내용 입력



```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\board_crud>
```

```
# 프로젝트 생성
> cd src/main 해서
> npx create-react-app frontend
> cd frontend

> npm install axios --save
> npm install react-router-dom --save
> npm install http-proxy-middleware --save

> npm start
```

Proxy 설정 (BE 와 API 통신 위함)

React src 밑에 `"proxy": "http://localhost:8080"`, # 추가

package.json

```
복사
// # main/frontend/main/package.json
{
  "name": "frontend",
  "version": "0.1.0",
  "proxy": "http://localhost:8080", # 추가
  "private": true,
  ...
}
```

Src 폴더 안에 setupProxy.js
를 만든후 아래내용 복사 붙이기 한다.

setupProxy.js

```
// # main/frontend/src/setupProxy.js
const { createProxyMiddleware } = require('http-proxy-middleware');

// "/api" 경로가 시작하면 프록시 미들웨어를 실행한다.
module.exports = function(app) {
  app.use(
    '/api',
    createProxyMiddleware({
      target: 'http://localhost:8080',
      changeOrigin: true
    })
  );
};
```

개발 시작!

1. [BE] Domain 작성 및 DB 테이블 생성
2. [BE] Controller(api) / Service / Repository 개발
3. [FE] Component 생성
4. [FE] axios 요청 및 응답 데이터 렌더링

1. [BE] Domain 작성 및 DB 테이블 확인

```
Java
복사
// # src/main/java/domain/Board.java

package spring_study.board_crud.domain;

import lombok.AccessLevel;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Data // @Getter @Setter
@Table(name = "board")
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@ToString(of = {"id", "title"})
public class Board {
```

```

@Id @GeneratedValue
private Long id;

private String title;
private String content;

public Board(Long id, String title, String content) {
    this.id = id;
    this.title = title;
    this.content = content;
}
}

```

Entity 로 등록 필요함

NoArgument Contructor 필요하다

MySQL 테이블 생성 확인!

Application.properties

application.yml 로 이름변경후 아래내용 붙여넣기

```

spring:
  datasource:
    url:
jdbc:mysql://localhost:3306/board_crud?useSSL=false&characterEncoding=UTF-8&serverTimezone=UTC
    username: root # -> ?? ??? MySQL username ??
    password: 1234 # -> ?? ??? MySQL ??? ?
    driver-class-name: com.mysql.cj.jdbc.Driver
  main:
    allow-circular-references: true

jpa:
  hibernate:
    ddl-auto: update
  properties:
    hibernate:
      show_sql: false
      format_sql: true
      default_batch_fetch_size: 100 # ????? ?? ??
  database: mysql
  database-platform: org.hibernate.dialect.MySQL8Dialect

```

MySQL Schema 생성하기

앞서 application.yml 파일에 엔드포인트와 포트번호 이후 board_crud 라는 스키마이름을 명시했었다.

MySQL Workbench 를 열고 앞서 생성한 localhost:3306 들어가서 username, password 입력해서 아래와 같이 창을 열고,

board_crud 라는 스키마 (db)생성

Swagger UI - API 명세

Bash

복사

```
// # build.gradle
dependencies {
    ...
    // Swagger UI 를 위해 필요한 dep
    implementation 'org.springdoc:springdoc-openapi-ui:1.6.6'
```

Bash

복사

```
// #src/main/java/SwaggerConfig.java

package spring_study.board_crud;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import org.springdoc.core.GroupeOpenApi;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SwaggerConfig { //Swagger 에 연동하기 위해 필요한 config 파일

    //Swagger 연동
    @Bean
    public GroupeOpenApi publicApi() {
        return GroupeOpenApi.builder()
            .group("v1-definition") //GROUP 이름 지정
            .pathsToMatch("/api/**") // "http://localhost:8080/api/~~"로 URL 이
시작하는 모든 API 들에 매핑
            .build();
    }

    //Swagger API 명세를 웹 브라우저에서 확인할 때 보이는 화면 커스텀
    @Bean
    public OpenAPI springShopOpenAPI() {
        return new OpenAPI()
            .info(new Info().title("BEER_PROJECT API") //제목
                .description("맥주 커뮤니티 프로젝트 API 명세서입니다.") //설명
                .version("v0.0.1"));
    }
}
```

2. [BE] Controller(api) / Service / Repository 개발

2-1-a. @GetMapping("/api/board-list")

controller

```
Java
복사
@RestController
@RequiredArgsConstructor
public class BoardApiController {

    private final BoardService boardService; // Autowired 로 스프링 빈에 등록

    @GetMapping("/api/board-list")
    public WrapperClass board_list(){
        List<Board> boardList = boardService.findBoards();
        List<BoardDto> boardDtoList = boardList.stream().map(b -> new
BoardDto(b)).collect(Collectors.toList());
        return new WrapperClass(boardDtoList);
    }
}
```

Dto : Data Transfer Object 계층 간 데이터 전송에 사용되는 객체

```
Java
복사
package spring_study.board_crud.dto;

import lombok.Data;
import lombok.NoArgsConstructor;
import spring_study.board_crud.domain.Board;

@Data
@NoArgsConstructor
public class BoardDto {
    private Long id;
    private String title;
    private String content;

    public BoardDto(Board board) {
        this.id = board.getId();
        this.title = board.getTitle();
        this.content = board.getContent();
    }
}
```

service

```
Java
복사
@Service
@Transactional(readonly = true)
@RequiredArgsConstructor
public class BoardService {

    private final BoardRepository boardRepository; // Auto wired 로 스프링 빈에 등록

    public List<Board> findBoards() {
```

```

        return boardRepository.findAll();
    }
}

```

repository → JpaRepository 를 상속

```

Java
복사
package spring_study.board_crud.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import spring_study.board_crud.domain.Board;

public interface BoardRepository extends JpaRepository<Board, Long> {

}

```

2-1-b. @GetMapping("/api/board-detail/{boardId}") - WrapperClass controller

```

Java
복사
@GetMapping("/api/board-detail/{boardId}")
public WrapperClass board_detail(@PathVariable("boardId") Long boardId){
    Board board = boardService.findOne(boardId);
    BoardDto boardDto = new BoardDto(board);
    return new WrapperClass(boardDto);
}

```

WrapperClass (main/java/spring_study/board_crud/api)

```

Java
복사
package spring_study.board_crud.api;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class WrapperClass<E> {
    private E data;
}

```

service

```

Java
복사
public Board findOne(Long id){
    return
boardRepository.findById(id).orElseThrow(NullPointerException::new);
} // id 에 해당하는 board 가 repository 에 존재하지 않을 경우 NullPointerException 에러
핸들링
// (** 서버 죽지 않게 하기 위함 **)

```

2-1-c. @PostMapping("/api/create-board") - ResponseEntity controller

에러 핸들링 없는 버전

```
Java
복사
@PostMapping("/api/create-board")
public ResponseEntity create_board(@RequestBody BoardDto boardDto){
    System.out.println("create_board/boardDto = " + boardDto);
    HttpHeaders headers = new HttpHeaders();
    Map<String, String> body = new HashMap<>();
    HttpStatus status = HttpStatus.CREATED; // 201 잘 생성되었음을 의미
    try{
        Board board = new Board(
            boardDto.getId(),
            boardDto.getTitle(),
            boardDto.getContent()
        );
        boardService.create(board);
    } catch (Exception exception){
        status = HttpStatus.BAD_REQUEST; // 400 에러
        System.out.println("create_board/exception = " + exception);
    }
    return new ResponseEntity(body, headers, status);
}
```

service

```
Java
복사
@Transactional // DB 에 영향을 주기 때문
public void create(Board board){
    boardRepository.save(board);
}
```

2-1-d. @PutMapping("/api/update-board") - service.update() controller

에러 핸들링 없는 버전

```
Java
복사
@PutMapping("/api/update-board")
public ResponseEntity update_board(@RequestBody BoardDto boardDto){
    System.out.println("update_board/boardDto = " + boardDto);
    HttpHeaders headers = new HttpHeaders();
    Map<String, String> body = new HashMap<>();
    HttpStatus status = HttpStatus.NO_CONTENT; // 204 -> 수정이 정상적으로 완료됐음을 의미
    try{
        boardService.update(boardDto.getId(), boardDto.getTitle(),
            boardDto.getContent());
    } catch (Exception exception){

```

```

        status = HttpStatus.BAD_REQUEST; // 400 에러
        System.out.println("update_board/exception = " + exception);
    }
    return new ResponseEntity(body, headers, status);
}

```

service

```

Java
복사
@Transactional
// Dirty Checking 으로 update 수행
public void update(Long id, String title, String content){
    Board board =
boardRepository.findById(id).orElseThrow(NullPointerException::new);
    board.setTitle(title);
    board.setContent(content);
}

```

2-1-e. @DeleteMapping("/api/delete-board")

controller

에러 핸들링 없는 버전

```

Java
복사
@DeleteMapping("/api/delete-board")
public ResponseEntity delete_board(@RequestBody BoardDeleteDto boardDeleteDto){
    System.out.println("delete_board/boardDeleteDto = " + boardDeleteDto);
    HttpHeaders headers = new HttpHeaders();
    Map<String, String> body = new HashMap<>();
    HttpStatus status = HttpStatus.NO_CONTENT;
    try{
        Board board = boardService.findOne(boardDeleteDto.getId());
        boardService.delete(board);
    } catch (Exception exception){
        status = HttpStatus.BAD_REQUEST;
        System.out.println("delete_board/exception = " + exception);
    }
    return new ResponseEntity(body, headers, status);
}

```

service

```

Java
복사
@Transactional
public void delete(Board board){
    boardRepository.delete(board);
}

```