스프링 부트 + React

연동 테스트 프로젝트

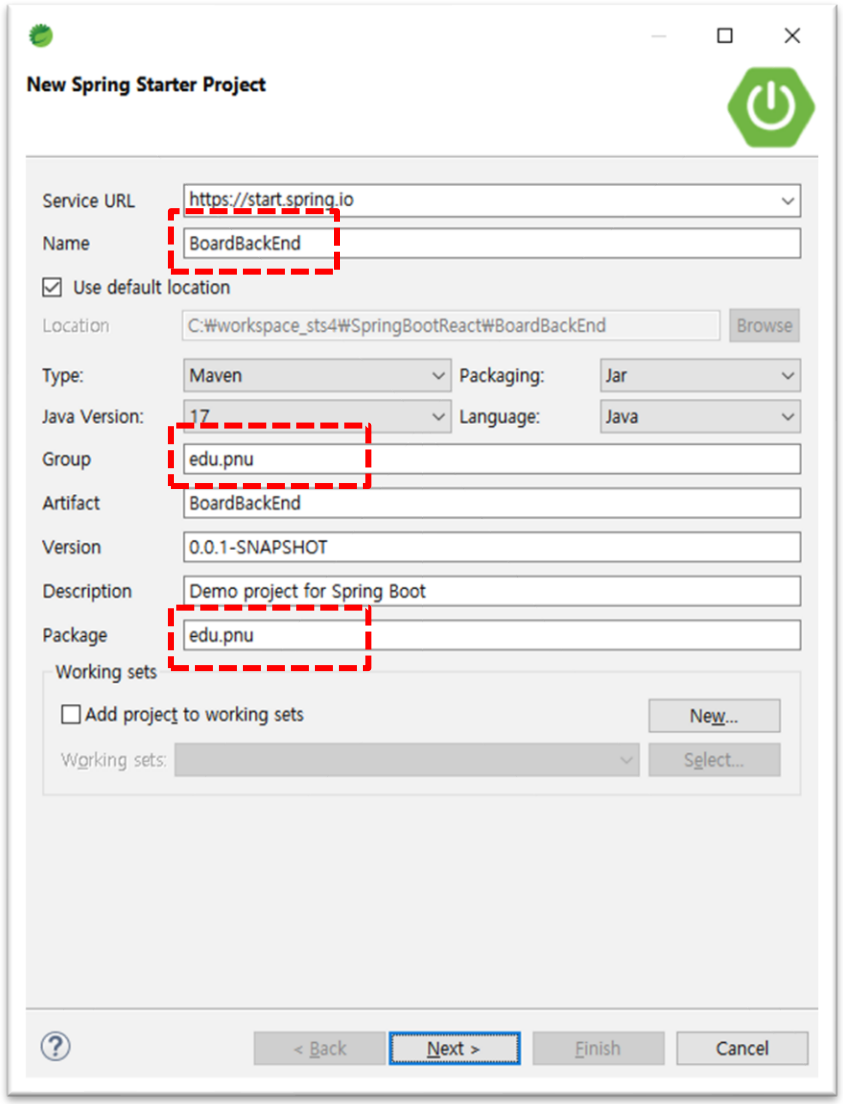
WorkSpace 설정

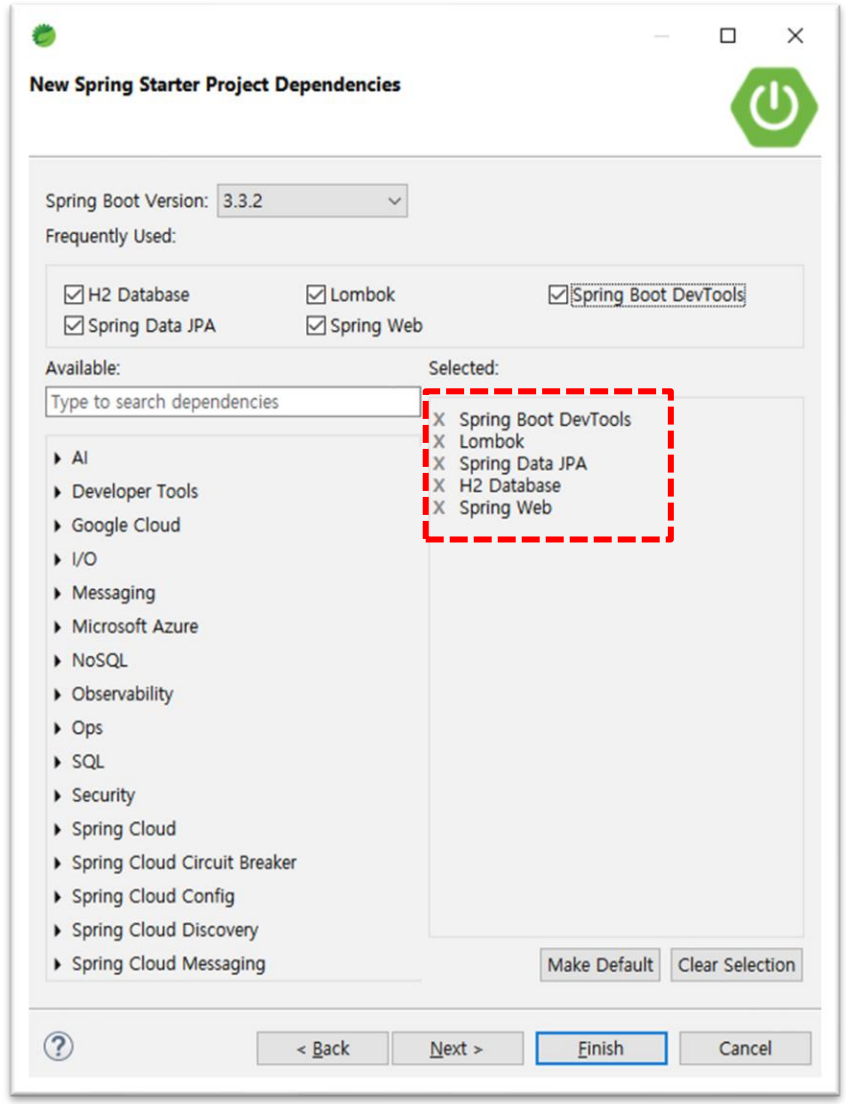
- C:\workspace-fullstack 생성해서
- STS4 실행
  - workspace 로 설정
- VS Code 실행
  - [파일] - [폴더 열기…] 로 선택

1. Back-End

STS4 for Eclipse

프로젝트 생성


New Spring Starter Project
Service URL
https://start.spring.io
Name
BoardBackEnd
Use default location
Location
C:₩workspace_sts4₩SpringBootReact₩BoardBackEnd
Browse
Type:
Maven
Packaging:
Jar
Java Version:
17
Language:
Java
Group
edu.pnu
Artifact
BoardBackEnd
Version
0.0.1-SNAPSHOT
Description
Demo project for Spring Boot
Package
edu.pnu
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel


New Spring Starter Project Dependencies
Spring Boot Version:
3.3.2
Frequently Used:
H2 Database
Lombok
Spring Boot DevTools
Spring Data JPA
Spring Web
Available:
Type to search dependencies
AI
Developer Tools
Google Cloud
I/O
Messaging
Microsoft Azure
NoSQL
Observability
Ops
SQL
Security
Spring Cloud
Spring Cloud Circuit Breaker
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Messaging
Selected:
Spring Boot DevTools
Lombok
Spring Data JPA
H2 Database
Spring Web
Make Default
Clear Selection
< Back
Next >
Finish
Cancel

Entity 클래스 작성

```
package edu.pnu.domain


import java.util.Date;
// 이하 생략
@Getter @Setter @ToString
@Entity
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Board {
    @Id @GeneratedValue(strategy
                        = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String title;
    @Column(nullable = false)
    private String content;
    private String writer;
    @Temporal(TemporalType.TIMESTAMP)
    @Builder.Default
    @Column(columnDefinition
        = "timestamp default current_timestamp")
    private Date createDate = new Date();
}
```

Repository작성, application.properties 설정

application.properties

```
spring.application.name=BoardBackEnd
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:mem:test
spring.datasource.username=sa
spring.datasource.password=

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.database-
platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

```
package edu.pnu.persistence;

import java.util.List;
// 이하 생략

public interface BoardRepository extends JpaRepository<Board, Long> {
}
```

테스트 데이터 입력 클래스 작성

```java
package edu.pnu;

import org.springframework.boot.ApplicationArguments;

@Component
@RequiredArgsConstructor
public class DataSetup implements ApplicationRunner {
    private final BoardRepository boardRepo;
    @Override
    public void run(ApplicationArguments args) throws Exception {
        String s[] = { "홍길동", "홍이동" };
        for (int i = 1 ; i <= 10 ; i++) {
            boardRepo.save(Board.builder().title("title"+i).content("content"+i).writer(s[(i%2)]).build());
        }
    }
}
```

서버 실행 후 h2 에서 데이터 확인

http://localhost:8080/h2-console

Board Controller/Service 작성

```java
@Slf4j
@RequiredArgsConstructor
@RestController
public class BoardController {
    private final BoardService boardService;

    @GetMapping("/board")
    public ResponseEntity<?> getBoard() {
        log.info("getBoard: All");
        return ResponseEntity.ok(boardService.getBoards());
    }
    @GetMapping("/board/{id}")
    public ResponseEntity<?> getBoard(@PathVariable Long id) {
        log.info("getBoard: " + id);
        return ResponseEntity.ok(boardService.getBoard(id));
    }
}
```

```java
@Service
@RequiredArgsConstructor
public class BoardService {
    private final BoardRepository boardRepo;

    public List<Board> getBoards() {
        return boardRepo.findAll();
    }
    public Board getBoard(Long id) {
        return boardRepo.findById(id).get();
    }
}
```

서버 실행 후

1. http://localhost:8080/board
2. http://localhost:8080/board/1

테스트

CORS 설정 (1) - 기본 형태

```
package edu.pnu.config;


@Configuration
public class CustomConfig implements WebMvcConfigurer {


   @Override
   public void addCorsMappings(@NonNull CorsRegistry registry) {
      registry.addMapping("/**")                          // 모든 주소에 대해서
            .allowedMethods(CorsConfiguration.ALL)        // 모든 Method에 대해서
            .allowedOrigins(CorsConfiguration.ALL);       // 모든 Origin에 대해서
   }
}
```

CORS 설정 (2) - 확장 응용 형태

```
@Override
public void addCorsMappings(@NonNull CorsRegistry registry) {
   registry.addMapping("/board/**")                    // /board 포함 하부 모든 주소에 대해서
         .allowedMethods(HttpMethod.GET.name(),
                         HttpMethod.POST.name())       // Get & Post Method에 대해서
         .allowedOrigins("http://localhost:3000", "http://127.0.0.1:3000");

   registry.addMapping("/member/**")                   // /member 포함 하부 모든 주소에 대해서
         .allowedMethods(HttpMethod.GET.name(),
                         HttpMethod.PUT.name())        // Get & Put Method에 대해서
         .allowedOrigins("http://localhost:3000");
}
```

CORS 설정 (3) - Authorization이 필요한 경우

```
@Override
public void addCorsMappings(@NonNull CorsRegistry registry) {
    registry.addMapping("/**")
        .allowCredentials(true)                    // 클라이언트가 자격증명(쿠키/인증헤더)을 포함하도록 허용
        .allowedHeaders(HttpHeaders.AUTHORIZATION)// 클라이언트가 요청 시 사용할 수 있는 헤더 지정
        .exposedHeaders(HttpHeaders.AUTHORIZATION)    // 클라이언트가 응답에 접근할 수 있는 헤더 지정
        .allowedMethods(HttpMethod.GET.name(),     // 클라이언트가 요청 시 사용할 수 있는 Method 지정
                        HttpMethod.POST.name(),
                        HttpMethod.PUT.name(),
                        HttpMethod.DELETE.name())
        .allowedOrigins("http://localhost:3000",
                        "http://127.0.0.1:3000");         // CORS 요청을 허용할 출처 지정
}
```

2. Front-End

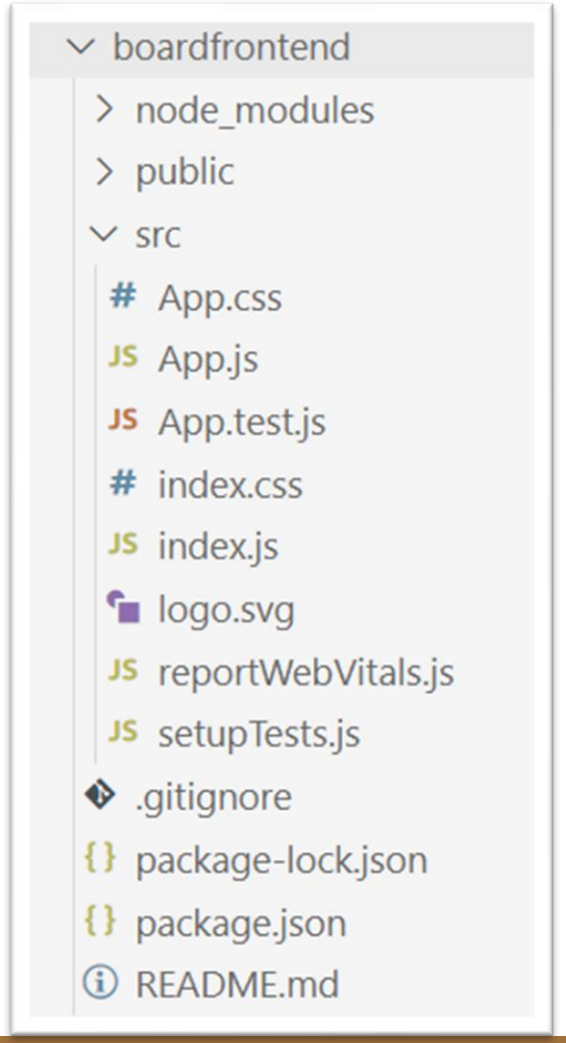
VS Code

프로젝트 생성

- ctrl + ` : Terminal 창

문제 출력 디버그 콘솔 터미널 포트

```
PS C:\workspace-fullstack> npm init react-app boardfrontend
```



- 주의 : 프로젝트명에 대문자가 있으면 에러가 발생함.

App.js 수정

```
import './App.css';
import DisplayData from './DisplayData'

function App() {
  return (
    <div className="App">
      <DisplayData />
    </div>
  );
}

export default App;
```

DisplayData.js 작성

```
import React, { useState, useEffect } from 'react';

const DataDisplay = () => {
  const [dataBoard, setDataBoard] = useState([]);

  const loadBoard = async () => {
    await fetch('http://localhost:8080/board')
    .then(resp => {
      return resp.json();
    }).then(result => {
      setDataBoard(result);
    }).catch(error => {
      console.error('Error fetching Board:', error);
    });
  };

  const loadData = () => {
    return (
      <table align="center">
        <thead>
          <tr>
            <th>ID</th><th>title</th><th>writer</th>
            <th>content</th><th>createDate</th>
          </tr>
        </thead>
        <tbody>
          {dataBoard.map(board => (
            <tr key={board.id}>
              <td>{board.id}</td>
              <td>{board.title}</td>
              <td>{board.writer}</td>
              <td>{board.content}</td>
              <td>{board.createDate}</td>
            </tr>
          ))}
        </tbody>
      </table>
    );
  };

  return (
    <div>
      <h2>Data Display</h2>
      <button onClick={() => loadBoard()}>Board</button>
      <div>{loadData()}</div>
    </div>
  );
};

export default DataDisplay;
```

Front-End 실행

- npm start

별첨 : JSON 요청 예제

```
const loadBoard = async () => {
   await fetch('http://localhost:8080/board', {
      method:'POST’,
      headers:{
         'Content-Type':'application/json',
         'Authorization':'jwt-token’
      },
      body:JSON.stringify(board)
   })
   .then(resp => {
      return resp.json();
   }).then(result => {
      setDataBoard(result);
   }).catch(error => {
      console.error('Error fetching Board:', error);
   });
};
```