

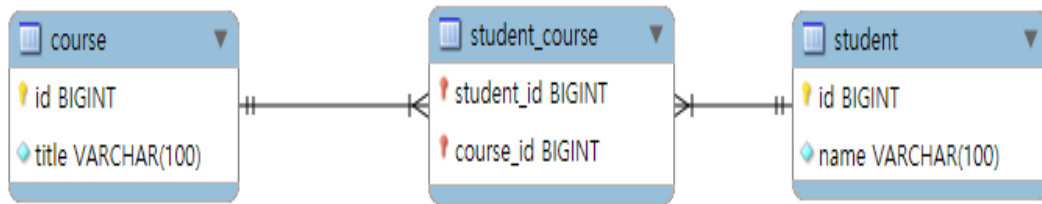
[SpringBootLab07 풀스택(Fullstack) 구현 실습]

SpringBootLab06 기반 Spring Boot[BE] + React.js[FE] 연동

학습 단계

단계	주요 내용	학습 키워드
[1]	백엔드 (Spring Boot) API 서버 만들기 ① 기존 MVC 컨트롤러 → @RestController 로 전환 ② @RequestBody 로 JSON 요청 받기 ③ @GetMapping, @PostMapping, @DeleteMapping	- RESTful API 설계 - JSON 데이터 구조 학습 - Controller ↔ Service ↔ Repository 흐름 숙지
[2]	프론트엔드 (React.js) API 연동 ① React 컴포넌트 (StudentList, CourseList) 구성 ② axios 로 API 호출 ③ 데이터 바인딩 및 상태 관리 (useState, useEffect)	- React 의 상태 관리 기본 - 비동기 통신 (axios/fetch) - API → 화면 반영 로직 이해
[3]	프론트-백 분리 학습 ① Spring Boot 는 JSON 만 제공 ② React 가 모든 UI/화면 처리 ③ CORS 허용 (서로 다른 포트에서 통신)	- 프론트/백 역할 분리 이해 - 네트워크 요청-응답 흐름 - CORS 정책과 API 보안
[4]	DB 연동과 양방향 관계 정리 ① JPA 를 통한 Student-Course 다대다 관계 ② 중간 테이블 자동 생성 및 관리	- JPA 양방향 관계 매핑 - 데이터 저장/삭제 시 동작 흐름 - DB 테이블 연동 이해
[5]	빌드 및 배포 ① React npm run build 로 정적 파일 생성 ② Spring Boot 의 resources/static 으로 복사 ③ - 최종 서비스 배포	- 정적 파일 서빙 방식 학습 - 풀스택 프로젝트 통합 - 실제 서비스 아키텍처 배포 방식 습득

[실습 01. SpringBoot 기반 [BE] 구축]



1) 디렉토리 구조

```

src/main/
├── java/com/sec01
│   ├── controller/
│   │   └── StudentCourseController.java    // 학생 & 강의 REST API 엔드포인트
│   ├── entity/
│   │   ├── Student.java                  // 학생 엔티티 (id, name, List<Course> courses)
│   │   └── Course.java                   // 강의 엔티티 (id, title, List<Student> students)
│   ├── repository/
│   │   ├── StudentRepository.java        // Student CRUD JpaRepository
│   │   └── CourseRepository.java         // Course CRUD JpaRepository
│   ├── service/
│   │   └── StudentCourseService.java     // 학생/강의 등록, 삭제 비즈니스 로직
│   ├── dto/
│   │   ├── StudentRequestDto.java       // 학생 추가 요청 JSON DTO (name, courseIds)
│   │   ├── StudentResponseDto.java      // 학생 + 수강과목 목록 응답 DTO (id, name, List<CourseDto>)
│   │   ├── CourseDto.java               // 과목 응답 DTO (id, title)
│   │   └── StudentsAndCoursesResponse.java // 학생+강의 목록 JSON 응답 DTO
│   └── SpringBootLab07Application.java   // 메인 클래스 (Spring Boot 앱 실행)
├── resources/
│   ├── application.yml                  // DB 연결, 포트, CORS 설정
│   └── static/                          // React 빌드 파일 복사 위치 (정적 리소스 제공)
  
```

2) Application.yml

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/spring_lab06
    username:
    password:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: update # 또는 create, create-drop, none
    show-sql: true # JPA 가 생성하는 SQL 을 콘솔에 출력
    properties:
      hibernate:
        format_sql: true # SQL 포매팅
  logging:
    level:
      org:
        hibernate:
          SQL: DEBUG # 실행되는 SQL 쿼리 로깅
        type:
          descriptor:
            sql: TRACE # SQL 파라미터 로깅
```

3) Entity

```
@Entity
@NoArgsConstructor
@AllArgsConstructor
public @Data class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;

    @ManyToMany(mappedBy = "courses")
    @ToString.Exclude
    private List<Student> students = new ArrayList<>();
}

@Entity
@NoArgsConstructor
@AllArgsConstructor
public @Data class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    @ManyToMany
    @JoinTable(
        name = "student_course",
        joinColumns = @JoinColumn(name = "student_id"),
        inverseJoinColumns = @JoinColumn(name = "course_id")
    )
    @ToString.Exclude
    private List<Course> courses = new ArrayList<>();
}
```

4) Repository

@Repository

public interface CourseRepository **extends** JpaRepository<Course, Long> {}

@Repository

public interface StudentRepository **extends** JpaRepository<Student, Long> {}

5) StudentCourseService

메서드명 : 반환타입	기능 설명	사용 Repository
getAllStudents(): List<Student>	모든 학생 목록을 조회한다.	StudentRepository
getAllCourses(): List<Course>	모든 과목 목록을 조회한다.	CourseRepository
addStudent(String, String) : void	학생과 과목을 동시에 추가하고, 중간테이블까지 함께 INSERT 한다.	StudentRepository (save) (JPA 가 중간테이블 insert 까지 처리)
addCourse(String) : void	과목만 단독으로 추가한다.	CourseRepository
deleteStudent(Long) : void	학생을 삭제한다.	StudentRepository
deleteCourse(Long) : void	과목을 삭제한다.	CourseRepository

6) DTO

DTO 파일명	역할 설명
CourseDto.java	과목 정보 (id, title)를 응답용으로 전달할 때 사용하는 DTO
StudentRequestDto.java	새 학생 추가할 때, 이름(name)과 과목 PK 리스트(courseIds)를 전달받는 요청 DTO
StudentResponseDto.java	프론트로 학생 정보 + 수강 과목 목록을 응답할 때 사용하는 DTO
StudentsAndCoursesResponse.java	학생 목록(StudentResponseDto 리스트)과 전체 과목 목록(Course 리스트)을 함께 응답하는 DTO
<pre>//[1] 과목 정보 (id, title)를 응답용 @Data @AllArgsConstructor public class CourseDto { private Long id; private String title; }</pre>	
<pre>//[2] 새 학생 추가할 시, 이름(name)과 과목 PK 리스트(courseIds)를 전달받는 요청 @Data public class StudentRequestDto { private String name; private List<Long> courseIds; }</pre>	
<pre>//[3] FE로 학생+과목 정보 응답용 @Data @AllArgsConstructor public class StudentResponseDto { private Long id; private String name; private List<CourseDto> courses; }</pre>	

```
// 학생 목록과 전체 과목 목록을 한번에 FE 리턴
@Data
@AllArgsConstructor
public class StudentsAndCoursesResponse {
    private List<StudentResponseDto> students;
    private List<Course> courses;
}
```

7).StudentCourseController

단계	RESTful API 메서드명	HTTP 메서드	RESTful URL	설명
[1]	getStudentsAndCourses	GET	/api/students-with-courses	모든 학생 목록과 과목 목록을 함께 조회
[2]	getCourses	GET	/api/courses	모든 과목 목록을 조회
[3]	addStudent	POST	/api/students	학생 + 과목을 동시에 추가
[4]	addCourse	POST	/api/courses	과목을 단독으로 추가
[5]	deleteStudent	DELETE	/api/students/{id}	특정 학생을 삭제
[6]	deleteCourse	DELETE	/api/courses/{id}	특정 과목을 삭제

[실습 02. React 기반 [FE] 구축]

VSCode 에서 [FE] 단계

단계	작업 내용	설 명
[1]	React 프로젝트 생성	VSCode 터미널에서 <code>npx create-react-app frontend</code> 실행 (또는 Vite 등 최신 툴 사용 가능)
[2]	필수 라이브러리 설치	<code>npm install axios</code> (백엔드 API 호출), <code>npm install bootstrap</code> (스타일링, 선택) 등
[3]	디렉토리 구조 정리	<code>src/components/</code> 폴더 생성, <code>StudentList.js</code> , <code>CourseList.js</code> 컴포넌트 파일 생성
[4]	백엔드 API 테스트	Postman/브라우저에서 Spring Boot API(<code>/api/**</code>)가 정상적으로 JSON 반환되는지 확인
[5]	컴포넌트에서 API 연결	axios 로 <code>GET /api/students-with-courses</code> , <code>POST /api/students</code> 등 API 호출
[6]	데이터 화면 출력	<code>useState</code> , <code>useEffect</code> 로 데이터 상태 관리, 화면에 학생/과목 목록 렌더링
[7]	추가/삭제 기능 연결	<code>POST</code> , <code>DELETE</code> 요청 연결 후 화면 갱신 (<code>loadStudents()</code> , <code>loadCourses()</code> 등)
[8]	화면 스타일링 및 UX 개선	Bootstrap 또는 CSS 로 테이블, 버튼 스타일링, UX 향상
[9]	백엔드 CORS 설정 확인	React 의 포트(3000) → Spring Boot API(8080) 호출할 때 CORS 허용 여부 확인 (백엔드에 CORS 설정 추가)
[10]	통합 배포 준비	<code>npm run build</code> 로 React 빌드 → Spring Boot <code>resources/static</code> 으로 복사 (또는 nginx 로 별도 배포)

1). React 프로젝트 생성 디렉토리 구조

```

SpringBootLab07/    -> 스프링부트 프로젝트 폴더 (STS 프로젝트)
├── src/             -> 백엔드 자바/리소스
├── pom.xml
├── ...
├── frontend/        -> React 프로젝트
│   ├── package.json
│   ├── src/
│   └── ...

```

1-1) Maven 설치 및 실행

<https://maven.apache.org/download.cgi> 메이븐 OS 별 선택 다운 후 압축풀고 path

등록!!! apache-maven-3.9.9-bin.zip

C:\Program Files\Java\apache-maven-3.9.9\bin 을 path 등록

```

C:\Users\Dominica>mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Program Files\Java\apache-maven-3.9.9
Java version: 21.0.7, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: ko_KR, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

```

```

● PS D:\myWork\MySpringBoot\SpringBootLab07> mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Program Files\Java\apache-maven-3.9.9
Java version: 21.0.7, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: ko_KR, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

```

```

○ PS D:\myWork\MySpringBoot\SpringBootLab07> mvn clean install

```

```

mvn -N io.takari:maven:wrapper

```

```

✓ SPRINGBOOTLAB07

```

```

  ✓ .mvn\wrapper

```

```

    ■ maven-wrapper.jar

```

```

    ≡ maven-wrapper.properties

```

```

    J MavenWrapperDownloader.java

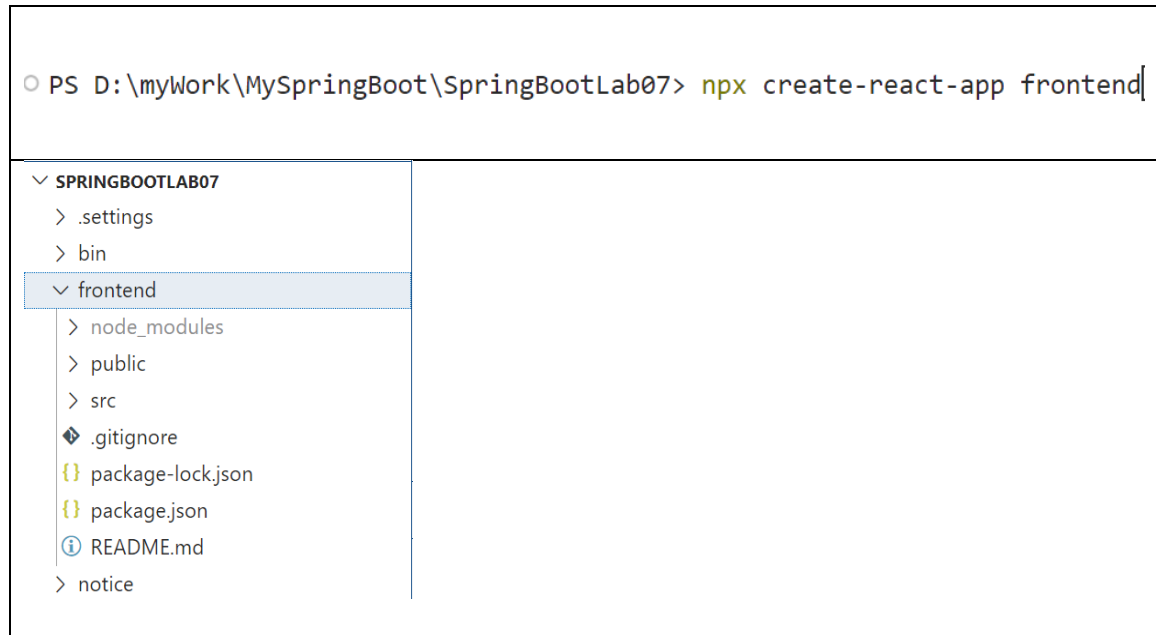
```

```

  > .settings

```

1-2) 프로젝트 생성 : npx create-react-app frontend



2). FE 필수 라이브러리 설치

npm install axios bootstrap react-router-dom

단계	라이브러리	용도 / 설명	설치 명령어
[2-1]	axios	백엔드 API 호출 (fetch 보다 사용하기 편리)	npm install axios
[2-2]	bootstrap	React UI 를 빠르게 예쁘게 꾸미기 (CDN 도 있지만 npm 으로 관리하면 더 편리)	npm install bootstrap
[2-3]	react-router-dom	여러 페이지를 URL 로 관리하는 라우팅 라이브러리	npm install react-router-dom npm install react-router-dom@6

3) 디렉토리 구조 정리 및 기본 컴포넌트 만들기

: React 프로젝트 구조를 정리하고 컴포넌트별로 파일을 나눠서 API 와 연결 준비

단계	작업 내용	상세 설명
[3-1]	src/components/ 폴더 생성	학생 목록, 강의 목록을 각자 컴포넌트로 분리!
[3-2]	StudentList.js, CourseList.js 생성	학생 목록, 강의 목록 관리하는 컴포넌트 각각 만들기
[3-3]	App.js 에서 컴포넌트 불러오기	import 로 StudentList, CourseList 를 App.js 에 연결
[3-4]	axios 기본 사용 테스트 (간단히 API 호출)	StudentList.js 에서 useEffect 로 백엔드 /api/students-with-courses 호출

디렉토리 구조 생성, 폴더 생성 , 각 파일 구현

frontend/	
— src/	
— components/	
— StudentList.js	-> 학생 목록 컴포넌트
— CourseList.js	-> 강의 목록 컴포넌트
— App.js	
— index.js	
— ...	
— package.json	
— ...	

3-2) 각 터미널에서 [BE] ,[FE]실행 확인

mvn spring-boot:run

npm start

```
PS D:\myWork\MySpringBoot\SpringBootLab07> mvn spring-boot:run
● PS D:\myWork\MySpringBoot\SpringBootLab07> cd frontend
○ PS D:\myWork\MySpringBoot\SpringBootLab07\frontend> npm start
```

3-3) 프록시 설정 (React → 8080 으로 API 전송) :frontend/package.json 에 아래 추가

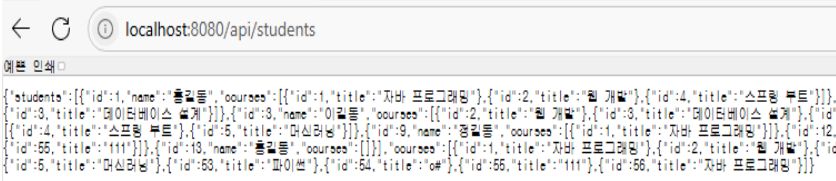
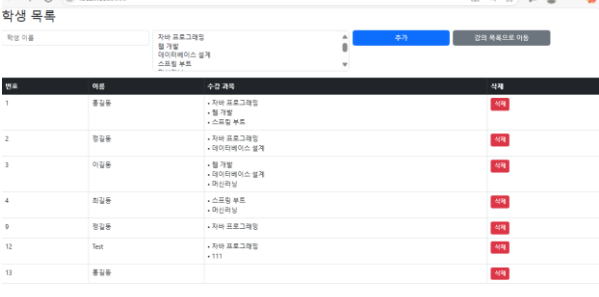
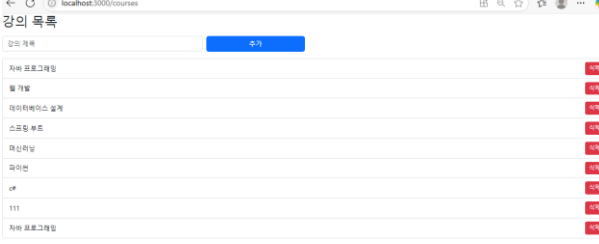
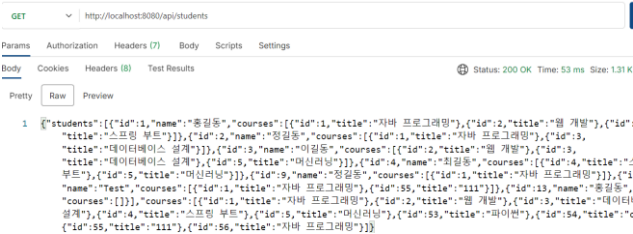
```
  },
  "proxy": "http://localhost:8080"
}
```

3-4) [BE]CORS 허용

```
@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api") // API 경로로 분리
public class StudentCourseController {
|
```

3-5) [BE]각 Entity 에 JSON 직렬화에서 제외 @JsonIgnore 선언

3-6) 요청확인

목적	URL 예시
백엔드 API	http://localhost:8080/api/students 
프론트앱 화면	http://localhost:3000  http://localhost:3000/courses 
Postman 테스트	http://localhost:8080/api/students 

4). 빌드, 배포

단계	내용	명령어/위치
1	React 정적 파일 빌드	npm run build (React 프로젝트 루트에서)
2	Spring Boot static 폴더로 복사	frontend/build → SpringBootLab07/src/main/resources/static
3	Spring Boot 빌드 & 실행	mvn clean package → java -jar target/*.jar
4	최종 확인	http://localhost:8080 에서 React 앱이 Spring Boot 로 서빙되는지 확인

4-1) React 정적 파일 빌드

React 빌드 경로:

- 빌드 결과: frontend/build
- 주요 파일: index.html, static/ (js/css 등)

Spring Boot 서빙 위치:

- 정적 파일은: src/main/resources/static
- static 폴더에 있는 정적 파일이 자동으로 / 경로로 서빙됨

결과:

- Spring Boot 로 접근 시 빌드된 React 앱이 그대로 나온다 (프록시도 필요 없음!)

•

[1] React 빌드

○ PS `D:\myWork\MySpringBoot\SpringBootLab07\frontend> npm run build`

[2] Spring Boot 로 복사 -> 수동 복사 가능!!

`xcopy /E /I /Y frontend\build src\main\resources\static\`*

```
● PS D:\myWork\MySpringBoot\SpringBootLab07\frontend> xcopy /E /I /Y build\* ..\src\main\resources\static\
>>
build\asset-manifest.json
build\favicon.ico
build\index.html
build\manifest.json
build\static\css\main.e6c13ad2.css
build\static\css\main.e6c13ad2.css.map
build\static\js\main.7873d648.js
build\static\js\main.7873d648.js.LICENSE.txt
build\static\js\main.7873d648.js.map
9 File(s) copied
```

5) 최종 배포

단계	내용	설명/명령어 예시
1	React 정적 파일 빌드	cd frontend npm run build
2	Spring Boot static 으로 복사	xcopy /E /I /Y frontend\build* src\main\resources\static\ (또는 수동으로 복사)
3	Spring Boot 빌드	mvn clean package
4	JAR 실행	java -jar target/프로젝트명-버전.jar
5	접속 확인	브라우저에서 http://localhost:8080 접속!

[배포할 때]

- AWS EC2, GCP, Azure, Naver Cloud 같은 클라우드 VM 에 위 과정대로 JAR 파일을 업로드
- 도메인 연결 (예: nginx 로 리버스 프록시 설정)
- systemctl 이나 pm2, Docker 등으로 Spring Boot 를 백그라운드에서 실행

단계	명령어	역할
JAR 생성	mvn clean package	Maven 으로 JAR 를 빌드하는 단계
JAR 실행	java -jar target/xxx.jar	만들어진 JAR 로 서버를 실행하는 단계

[1] JAR 생성

```
PS D:\myWork\MySpringBoot\SpringBootLab07> mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.sec01:SpringBootLab07 >-----
[INFO] Building SpringBootLab07 0.0.1-SNAPSHOT
[INFO]    from pom.xml
[INFO] -----[ jar ]-----
[INFO]
```

[2] JAR 실행 _ BE_FE <http://localhost:8080/> 통합

```
PS D:\myWork\MySpringBoot\SpringBootLab07> java -jar target/SpringBootLab07-0.0.1-SNAPSHOT.jar
```

[illegible]