



It's Your Life

with





어제 했어야 했는데

정신 못 차린

저라는...

알코올 trash

정말 죄송합니다 ㅠㅠ

# 마이바티스를 활용한



# 게시판 만들기 문제

[https://drive.google.com/file/d/1p-p76YWW7LXBHanARbwxCYoyH8Jv6GoU/view?usp=drive\\_link](https://drive.google.com/file/d/1p-p76YWW7LXBHanARbwxCYoyH8Jv6GoU/view?usp=drive_link)



해당 링크의 압축 파일을 다운 받고 압축을 풀어 주세요!

압축을 풀고 board2 폴더를 인텔리제이로 열어 주세요 😊

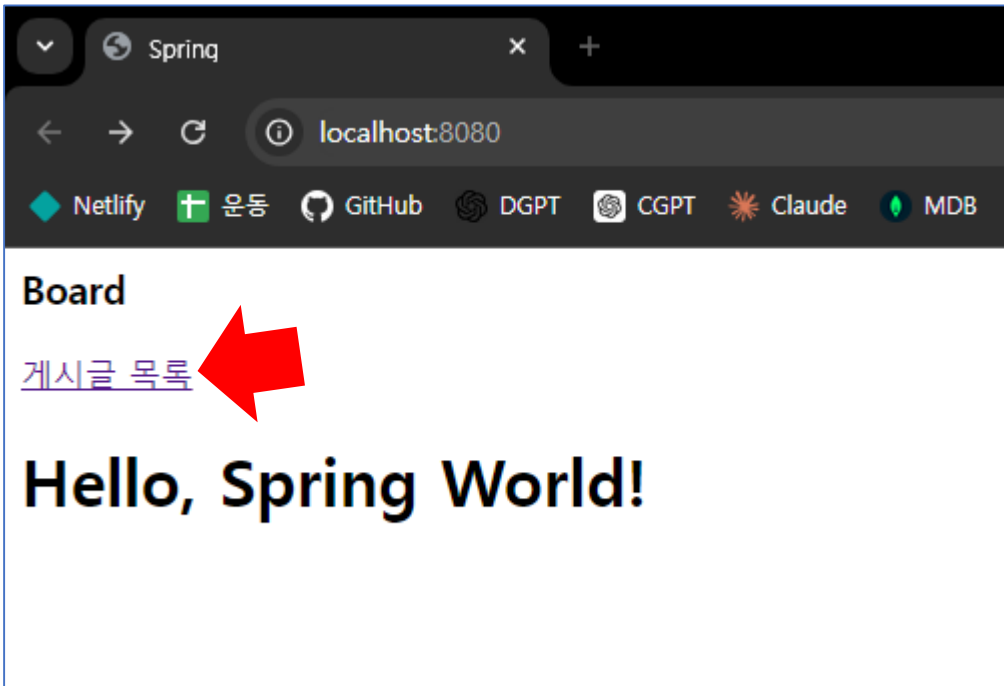


# 우리가 해야할 것!

## 요약!

# 1. 게시물 목록 페이지





스프링 프로젝트를 실행하면  
왼쪽과 같이 서비스가 실행이 됩니다!

그리고 게시글 목록을 클릭하면~!



## Board

[게시글 목록](#)

### 글 목록

ID	Title	Content
5	<a href="#">제목5</a>	내용5
4	<a href="#">제목4</a>	내용4
3	<a href="#">제목3</a>	내용3
2	<a href="#">제목2</a>	내용2
1	<a href="#">제목1</a>	내용1

글 작성하기

MyBatis 를 활용하여  
게시판 목록을 보여주는  
기능을 구현하면 됩니다!



## 2. 게시물 작성하기



## Board

[게시글 목록](#)

### 글 목록

ID	Title	Content
5	<a href="#">제목5</a>	내용5
4	<a href="#">제목4</a>	내용4
3	<a href="#">제목3</a>	내용3
2	<a href="#">제목2</a>	내용2
1	<a href="#">제목1</a>	

글 작성하기

### 글 작성

작성자

tetz

제목

글 작성 테스트

내용

글 작성 테스트

글 작성

취소

글 작성하기를 누르면  
아래의 글 작성 모드로 변환 됩니다.

새로운 글을 작성한 이후  
글 작성을 버튼을 누르면

## Board

[게시글 목록](#)

### 글 목록

ID	Title	Content
6	<a href="#">글 작성 테스트</a>	글 작성 테스트
5	<a href="#">제목5</a>	내용5
4	<a href="#">제목4</a>	내용4
3	<a href="#">제목3</a>	내용3
2	<a href="#">제목2</a>	내용2
1	<a href="#">제목1</a>	내용1

글 작성하기

새로운 글이 DB에 등록되고  
다시 글 목록 페이지로 이동하여  
작성된 게시글을  
확인할 수 있으면 됩니다!



# 3. 게시글 내용 페이지



## Board

[게시글 목록](#)

### 글 목록

ID	Title	Content
6	<a href="#">글 작성 테스트 제목</a>	글 작성 테스트 내용
5	<a href="#">제목5</a>	내용5
4	<a href="#">제목4</a>	내용4
3	<a href="#">제목3</a>	내용3
2	<a href="#">제목2</a>	내용2
1	<a href="#">제목1</a>	내용1

글 작성하기

게시글의 제목을 클릭하면  
아래와 같이 게시글의 세부 내용을  
확인할 수 있는 페이지로 이동합니다!



### 글 작성 테스트 제목

작성자: test6

작성일: Tue Sep 03 00:48:36 KST 2024

글 작성 테스트 내용

수정

삭제

목록으로

# 4. 게시물 삭제



## 글 작성 테스트 제목

작성자: test6

작성일: Tue Sep 03 00:48:36 KST 2024

글 작성 테스트 내용

수정

삭제

목록으로

게시글 세부 내용 페이지에서  
삭제 버튼을 클릭하면  
아래와 같이 게시글이  
삭제 되어야 합니다



## Board

[게시글 목록](#)

## 글 목록

ID	Title	Content
5	<a href="#">제목5</a>	내용5
4	<a href="#">제목4</a>	내용4
3	<a href="#">제목3</a>	내용3
2	<a href="#">제목2</a>	내용2
1	<a href="#">제목1</a>	내용1

글 작성하기



# 5. 게시물 수정



## 제목5

작성자: test5

작성일: Tue Sep 03 00:40:15 KST 2024

내용5

수정

삭제

목록으로

## 게시글 수정

ID

5

작성자

test5

제목

제목5 수정 테스트

내용

내용5 수정 테스트

수정

목록으로 돌아가기

게시글 세부 내용 페이지에서  
수정 버튼을 클릭하면 아래와 같이  
게시글 수정 모드로  
이동 되어야 합니다





게시글 수정 모드에서  
내용을 수정하고 수정 버튼을 누르면  
아래와 같이 게시글이  
수정 되면 됩니다!

## 게시글 수정

ID

5

작성자

test5

제목

제목5 수정 테스트

내용

내용5 수정 테스트

수정

목록으로 돌아가기

## Board

[게시글 목록](#)

## 글 목록

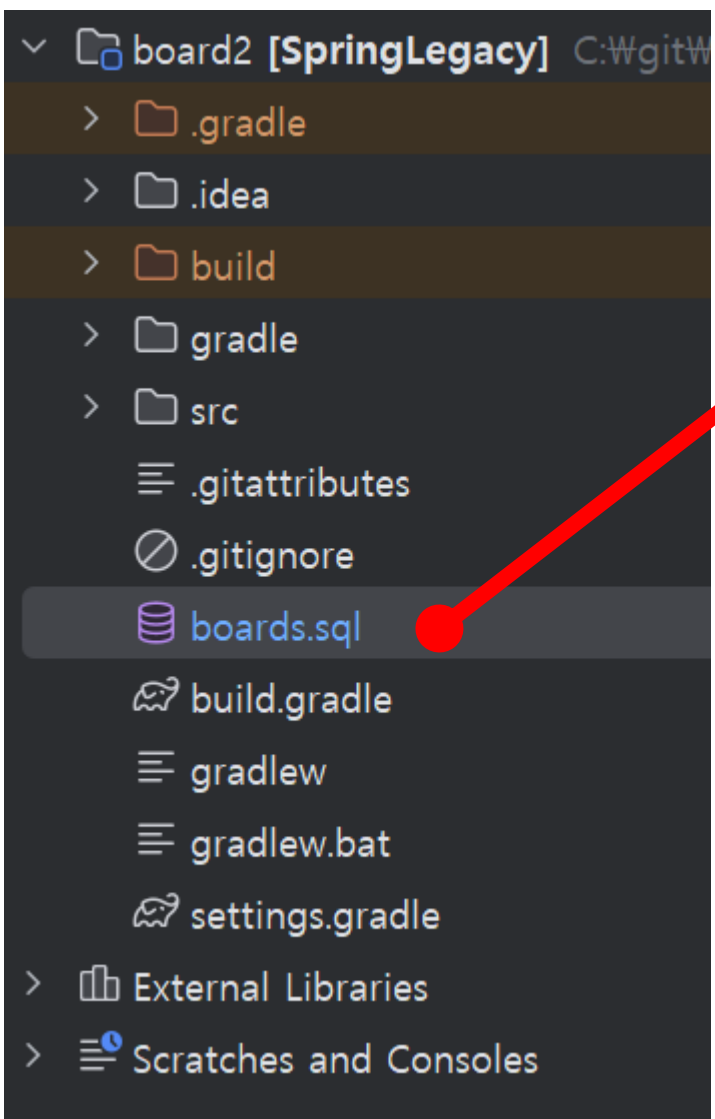
ID	Title	Content
5	<a href="#">제목5 수정 테스트</a>	내용5 수정 테스트

# DB 구축하기





폴더 제일 외부에 존재하는 boards.sql 파일을 열어 주세요




```

1 • create database board_db;
2 use board_db;
3
4 CREATE TABLE boards
5 (
6     id         INTEGER AUTO_INCREMENT PRIMARY KEY,
7     title      VARCHAR(200) NOT NULL,
8     content    VARCHAR(200) NOT NULL,
9     author     VARCHAR(50)  NOT NULL,
10    reg_date   DATETIME DEFAULT CURRENT_TIMESTAMP
11 );
12
13 • INSERT INTO boards(title, content, author)
14 VALUES ('제목1', '내용1', 'test1'),
15         ('제목2', '내용2', 'test2'),
16         ('제목3', '내용3', 'test3'),
17         ('제목4', '내용4', 'test4'),
18         ('제목5', '내용5', 'test5');
19
20
21 • SELECT * FROM boards;

```

해당 쿼리문을 사용하여  
board\_db 데이터 베이스를 만들고

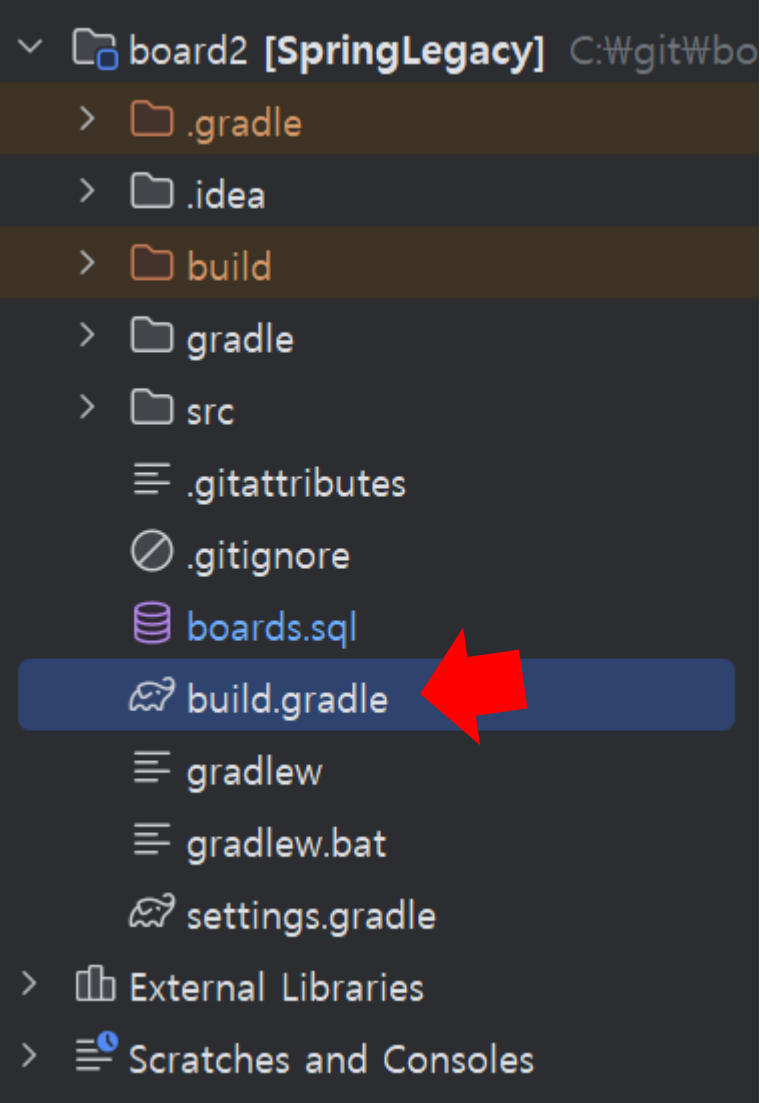
boards 테이블을 만든 다음  
테스트 데이터를 넣어 주시면 됩니다!

Result Grid					
Filter Rows: <input type="text"/>					
Edit: 					
	id	title	content	author	reg_date
▶	1	제목1	내용1	test1	2024-09-02 11:18:54
	2	제목2	내용2	test2	2024-09-02 11:18:54
	3	제목3	내용3	test3	2024-09-02 11:18:54
	4	제목4	내용4	test4	2024-09-02 11:18:54
	5	제목5	내용5	test5	2024-09-02 11:18:54
⊕	NULL	NULL	NULL	NULL	NULL



# 프로젝트 구조

## 설명

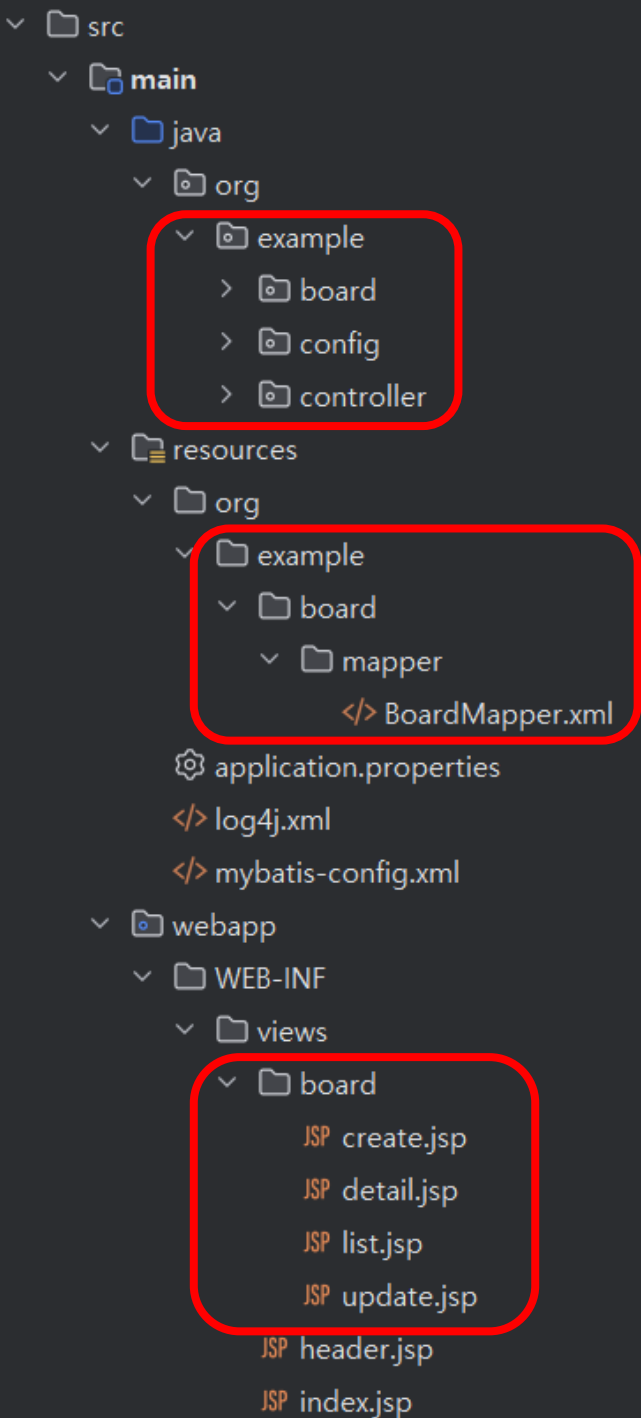


```
// 테스트
testImplementation("org.springframework:spring-test:${springVersion}")
testCompileOnly("org.projectlombok:lombok:${lombokVersion}")
testAnnotationProcessor("org.projectlombok:lombok:${lombokVersion}")
testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")

// 데이터 베이스
implementation 'com.mysql:mysql-connector-j:8.1.0'
implementation 'com.zaxxer:HikariCP:2.7.4'
implementation "org.springframework:spring-tx:${springVersion}"
implementation "org.springframework:spring-jdbc:${springVersion}"
implementation 'org.mybatis:mybatis:3.4.6'
implementation 'org.mybatis:mybatis-spring:1.3.2'
```

build.gradle 파일에는  
해당 프로젝트 진행에 필요한  
모든 라이브러리가 추가 되어있습니다!

그러니 신경 쓰지 않으셔도 됩니다!



프로젝트 폴더의 구성은  
왼쪽과 같이 구성되어 있습니다!

프로젝트를 위한 기본 설정 및 코드들은  
전부 작성되어 있으며

여러분들은 문제에서 설명하고 있는 부분만  
구현을 하시면 됩니다!!

org.example.board 패키지  
org/example/board/mapper 폴더  
views/board 폴더의  
코드만 작성하시면 됩니다!



resources  
org  
application.properties  
log4j.xml  
mybatis-config.xml



마이바티스 설정에 MapUnderscoreToCamelCase  
옵션이 true 로 설정 되어 있습니다!

또한, 메인 도메인인 Board 클래스도  
Alias 로 등록되어 있습니다

```
<configuration>
  <settings>
    <setting name="mapUnderscoreToCamelCase" value="true" />
  </settings>
  <typeAliases>
    <typeAlias type="org.example.board.domain.Board" alias="Board" />
  </typeAliases>
</configuration>
```

```
CREATE TABLE boards
```

```
(  
    id          INTEGER AUTO_INCREMENT PRIMARY KEY,  
    title       VARCHAR(200) NOT NULL,  
    content     VARCHAR(200) NOT NULL,  
    author      VARCHAR(50)  NOT NULL,  
    reg_date    DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

✓ @Data 21 usages Tetz \*

@AllArgsConstructor

@NoArgsConstructor

@Builder

public class Board {

private Long id;

private String title;

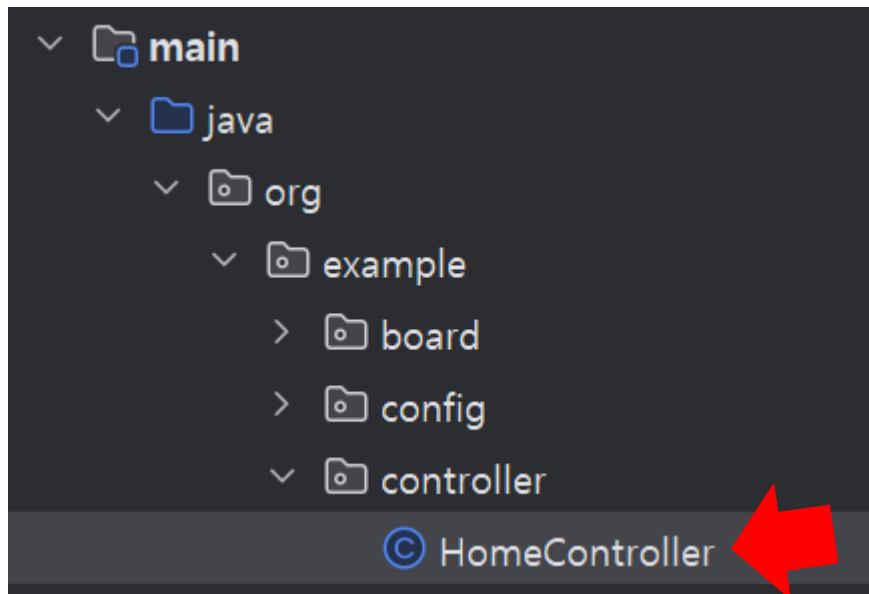
private String content;

private String author;

private Date regDate;



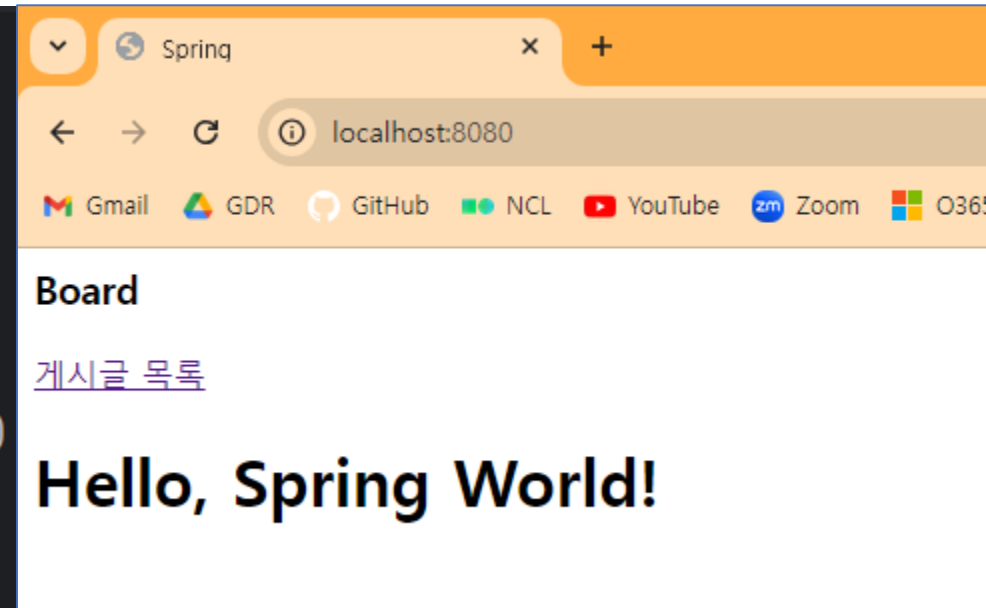
MapUnderscoreToCamelCase 설정에 의해  
reg\_date 컬럼은 자동으로 Board 클래스의  
regDate 로 매핑이 됩니다!



프로젝트를 구동하면  
HomeController 에 의해  
아래와 같이 화면이 뜹니다!



```
@Controller  
@Slf4j  
public class HomeController {  
    @GetMapping("/")  
    public String home() {  
        log.info("=====> HomeController /")  
        return "index";  
    }  
}
```



RootConfig 의 설정은 전부 되어 있으며  
프로젝트 폴더의 구조에 맞게 Scan 설정도 전부 되어 있습니다!

A screenshot of an IDE interface. On the left, the 'Project Explorer' shows a folder named 'config' containing three files: 'RootConfig', 'ServletConfig', and 'WebConfig'. A red arrow points to 'RootConfig'. Below 'config' is a folder named 'controller'. Further down, there are folders for 'resources', 'org', and 'application.properties'. The main editor area shows the code for 'RootConfig.java'. The code includes annotations for '@Configuration', '@PropertySource', '@ComponentScan', and '@MapperScan', followed by the class definition 'public class RootConfig' and a partial line for '@Value'.



## ServletConfig 와 WebConfig 역시 마찬가지 입니다!

config

RootConfig

**ServletConfig**

WebConfig

controller

10

11

12

13

14

```
@EnableWebMvc 1 usage Tetz
@ComponentScan(basePackages = {"org.example"})
public class ServletConfig implements WebMvcConfigurer {
    // Spring MVC용 컴포넌트 등록을 위한 스캔 패키지
```

config

RootConfig

ServletConfig

**WebConfig**

controller

1

2

3

6

7

8

11

12

13

14

```
package org.example.config;

import ...

public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {

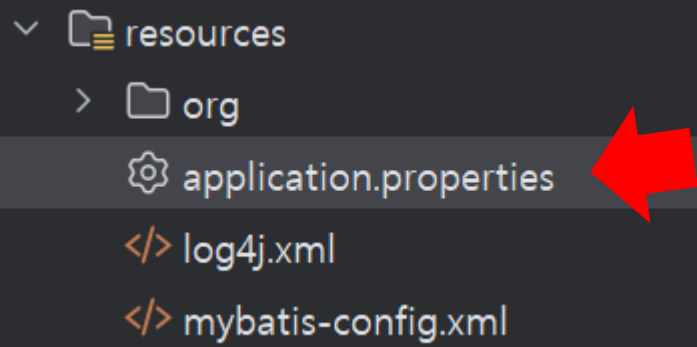
    @Override no usages Tetz
    protected Class<?>[] getRootConfigClasses() { return new Class[] { RootConfig.class; }

    @Override no usages Tetz
    protected Class<?>[] getServletConfigClasses() { return new Class[] { ServletConfig.class; }

}
```

# 프로젝트 세팅





application.properties 파일을 열어 주세요





```
1 jdbc.driver=com.mysql.cj.jdbc.Driver
2 jdbc.url=jdbc:mysql://localhost:3306/board_db
3 jdbc.username=root
4 jdbc.password=1234|
```

본인이 사용하는  
MySQL의 ID와 비밀번호로  
변경해 주세요!

데이터베이스는 방금 만든  
board\_db로 세팅이 되어 있으니  
그대로 두시면 됩니다!

혹, 문제에서 제공된 DB명과  
해당 url의 DB명이 다를 경우 반드시  
동일하게 맞춰 주셔야 합니다!





자~ 이제 시작이야(내꿈을)



# 문제 1

## 게시판 목록 기능

# 게시판 목록 기능 완성하기



- 요청

- GET 방식

- <http://localhost:8080/board/list>

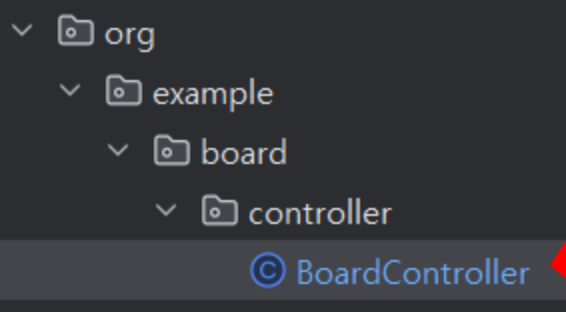
- 필요 기능

- boards 테이블의 내용을 읽어서, 게시글 목록을 출력해 주는 기능 구현

- 현재 상태

- BoardController 클래스의 listPage 메서드에서 데이터를 전달하고 있지 못한 상태

- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태



```
20 public class BoardController {
21     // @RequiredArgsConstructor 에 의해 의존성이 자동으로 주입 됩니다
22     private final BoardService boardService;
23     private final String context = "/board"; 4 usages
24
25     // 문제 1. 게시판 목록 기능 구현하기
26     @GetMapping("/list") Tetz *
27     public String listPage() {
28         // 여기 부분에 코드를 완성하여, 게시판 목록 기능을 완성시켜 주세요
29         return context + "/list";
30     }
```

BoardService 를 사용하여  
DB 의 데이터를 받아온 다음  
list.jsp 로 전달하여 게시글 목록 기능을  
완성시켜 주세요!



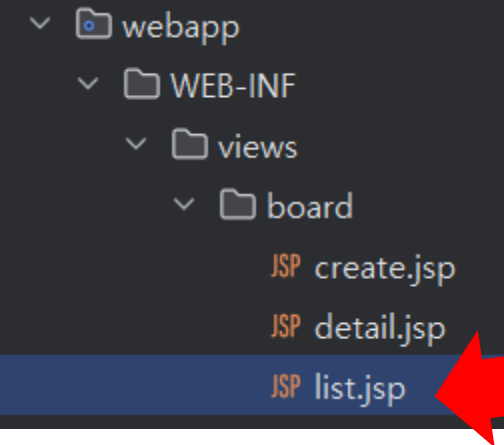
resources  
org  
example  
board  
mapper

BoardMapper.xml



```
6 <mapper namespace="org.example.board.mapper.BoardMapper">  
7   <select id="getList" resultType="Board">  
8     -- 문제 1. 게시판 목록 기능 구현하기 : 적절한 SQL 구문을 작성해 주세요  
9   </select>
```

적절한 SQL 구문을 작성해 주세요



list.jsp 페이지에서는  
게시글 데이터 배열을 boardList 로 받습니다!

본인이 원하는 경우 수정하셔도 무방합니다!  
단, 기능은 작동 해야겠죠!? 😊

```
102 <c:forEach var="board" items="${boardList}">
103     <tr>
104         <td>${board.id}</td>
105         <td><a href="/board/detail?id=${board.id}">${board.title}</a></td>
106         <td>${board.content}</td>
107     </tr>
108 </c:forEach>
```



# 문제 2

게시글 작성 기능

# 게시글 작성 기능 완성하기



- 요청 주소
  - POST 방식
  - <http://localhost:8080/board/create>
- 필요 기능
  - 새롭게 입력 받은 게시글의 내용을 DB 에 등록하고, 게시글 목록 페이지로 이동하여 작성한 글을 바로 확인할 수 있는 기능



# 게시글 작성 기능 완성하기



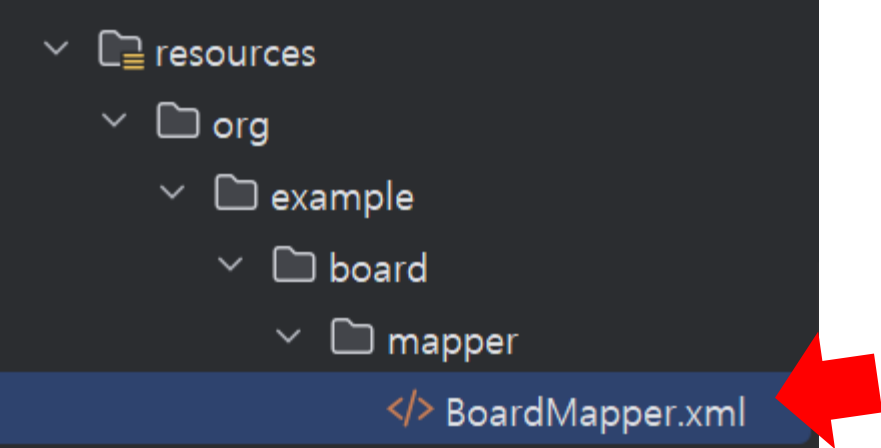
- 현재 상태

- BoardController 클래스의 createPage 메서드에서 글 작성 모드로 이동 하는 기능은 완성
- BoardController 클래스의 create 메서드에서 게시글 등록을 못하고 있는 상태
- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태



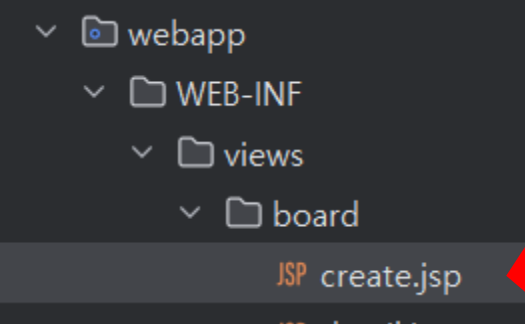
```
32 // 게시물 작성 모드 페이지로 이동하는 메서드 -> 이미 완성 된 상태
33 @GetMapping(🌐"/create") 🧑 Tetz
34 public String createPage() {
35     return context + "/create";
36 }
37
38 // 문제 2. 게시물 작성 기능 구현하기
39 @PostMapping(🌐"/create") 🧑 Tetz *
40 public String create() {
41     // 여기 부분에 코드를 완성하여, 게시물 작성 기능을 완성시켜 주세요
42     return "redirect:/board/list";
43 }
```

글 작성 페이지에서 전달하는 데이터를 받아서  
게시글을 DB 에 등록하는  
컨트롤러 코드를 완성시켜 주세요



```
11      <insert id="create">
12      -- 문제 2. 게시글 작성 기능 구현하기 : 적절한 SQL 구문을 작성해 주세요
13      </insert>
```

적절한 SQL 구문을 작성해 주세요



```
<h1>글 작성</h1>
<form action="/board/create" method="post">
  <label for="title">작성자</label>
  <input type="text" name="author" />

  <label for="title">제목</label>
  <input type="text" id="title" name="title" required>

  <label for="content">내용</label>
  <textarea id="content" name="content" required></textarea>

  <input type="submit" value="글 작성">
  <a class="cancel-button" href="/board/list">취소</a>
</form>
```

글 작성 페이지는  
작성자를 author,  
제목을 title,  
내용을 content  
라는 이름으로 전달 합니다!



# 문제 3

게시글 내용 보기 기능

# 게시글 세부 내용 보기 기능 완성



- 요청 주소
  - GET 방식
  - <http://localhost:8080/board/detail?id={id}>
- 필요 기능
  - 파라미터 방식으로 입력 받은 id 값을 가지는 게시글을 찾아 detail.jsp 로 전달하여 게시글의 세부 내용을 보여주는 기능

# 게시글 세부 내용 보기 기능 완성



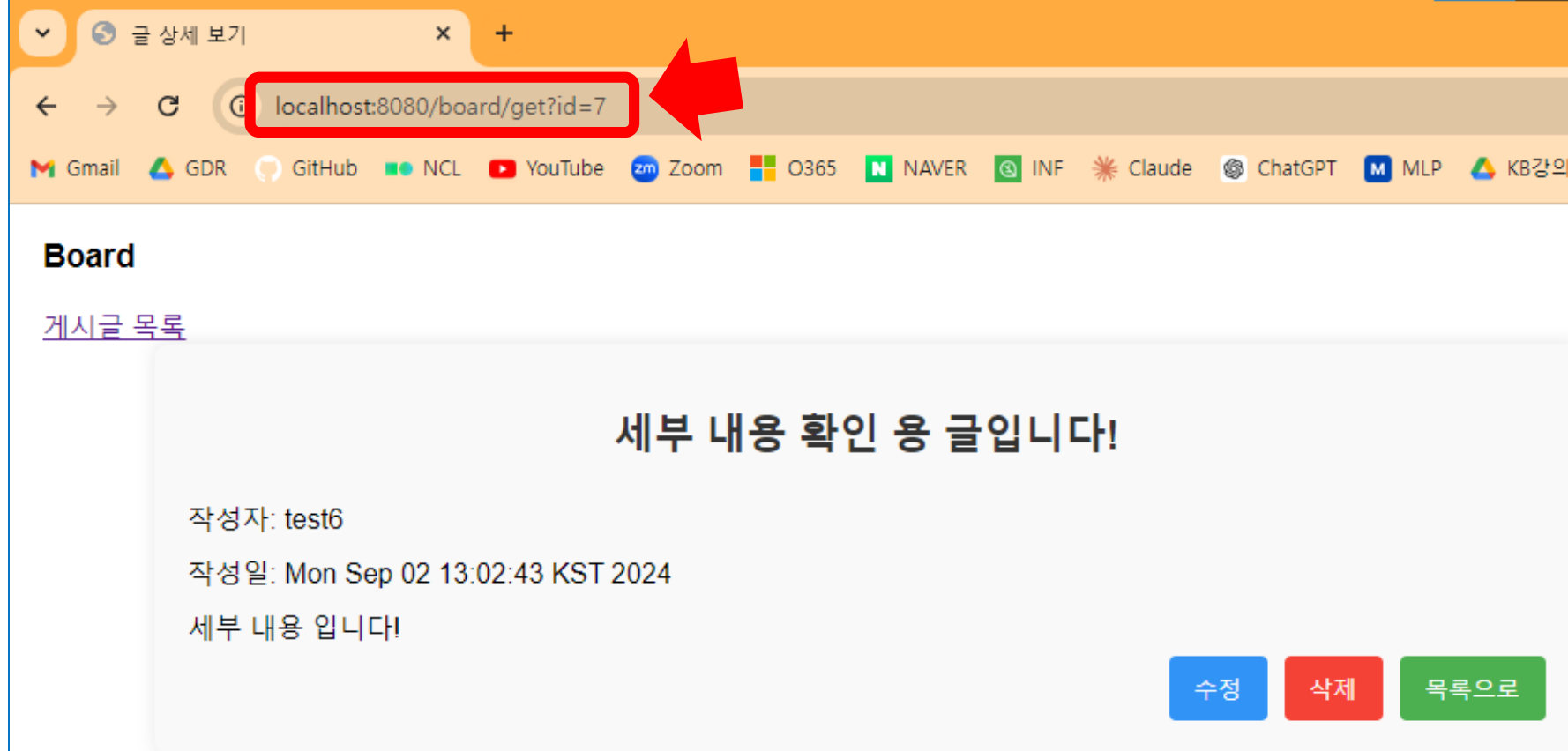
- 현재 상태

- BoardController 클래스의 detailPage 메서드에서 특정 id 를 가지는 게시글을 찾아서 전달하는 기능이 완성 안된 상태
- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태
- detail.jsp 페이지에 전달한 데이터를 출력하는 구문이 빠진 상태

## Board

[게시글 목록](#)

ID	Title
7	<a href="#">세부 내용 확인 용 글입니다!</a>



글 상세 보기

localhost:8080/board/get?id=7

Gmail GDR GitHub NCL YouTube Zoom O365 NAVER INF Claude ChatGPT MLP KB강의

### Board

[게시글 목록](#)

## 세부 내용 확인 용 글입니다!

작성자: test6  
작성일: Mon Sep 02 13:02:43 KST 2024  
세부 내용 입니다!

수정 삭제 목록으로

게시글 제목을 클릭하면 게시글의 세부 내용을 볼 수 있는 기능을 완성하시면 됩니다!

주소 창을 확인하시면 어떤 주소로 요청이 전달 되는지 확인할 수 있습니다!



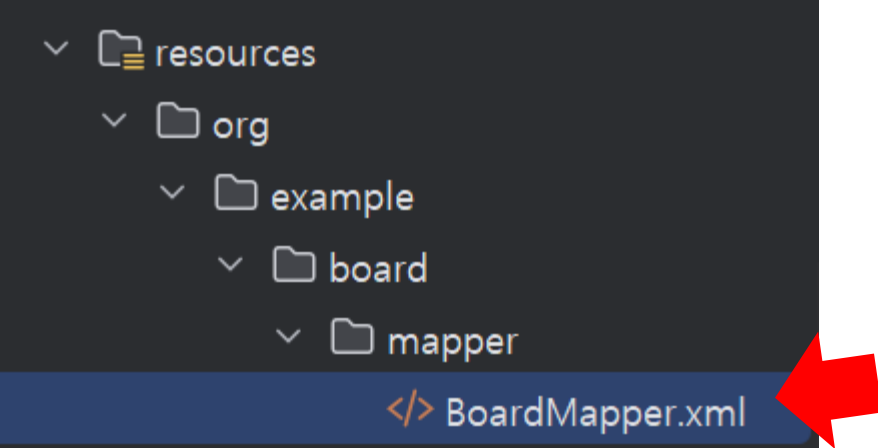


```
45 // 문제 3. 게시물 내용 보기 기능 구현하기
46 @GetMapping(🌐"/detail") new *
47 🌐 ✓ public String detailPage() {
48     // 여기 부분에 코드를 완성하여, 게시물 내용 보기 기능을 완성시켜 주세요
49     return context + "/detail";
50 }
```

detail?id={id} 로 들어오는 id 파라미터의 값을 받아서  
해당 id 를 가지는 게시글을 찾은 후,  
detail.jsp 로 전달하는 컨트롤러 코드 완성 하기!

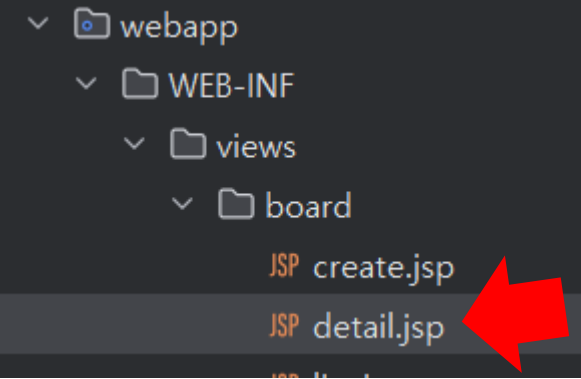
기존에 완성된 jsp 파일의 변경을 막기 위해서  
Model 을 통해 전달하는 key 의 이름은 가급적 board 로  
전달해 주세요 😊

전달 key 를 변경하는 경우  
detail.jsp 파일도 수정해 주셔야 합니다!



```
15  <select id="detail" resultType="Board">
16      -- 문제 3. 게시글 내용 보기 기능 구현하기 : 적절한 SQL 구문을 작성해 주세요
17      </select>
```

적절한 SQL 구문을 작성해 주세요



자신이 전달한 데이터를 각각의 항목에 맞게  
출력하는 코드 작성 필요!

```
76 <div class="post-container">
77   <h1 class="post-title"> <!-- 제목 넣기 --> </h1>
78   <div class="post-content">
79     <div>작성자: <!-- 작성자 넣기 --></div>
80     <div>작성일: <!-- 작성일 넣기 --></div>
81     <div>
82       <!-- 게시글 내용 넣기 -->
83     </div>
84   </div>
85   <div class="button-container">
86     <!-- 자신이 컨트롤러에서 전달한 데이터의 key 가 다른 경우 아래의 id 전달 부분 부분 수정 필요 -->
87     <a href="/board/update?id=${board.id}" class="button edit-button">수정</a>
88     <form action="/board/delete/${board.id}" method="post">
89       <input type="submit" class="button delete-button" value="삭제" />
90     </form>
91     <a href="/board/list" class="button back-button">목록으로</a>
```

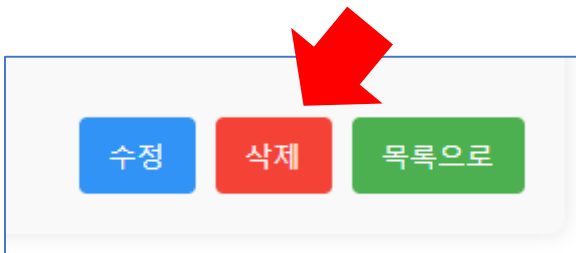
삭제, 수정 기능에  
필요한 id 전달

→ 이미 완성되어 있음



# 문제 4

## 게시글 삭제 기능



삭제 버튼을 클릭하면  
POST 방식 /board/delete/{id} 주소로  
삭제 요청을 보냅니다!



```
<div class="button-container">
  <!-- 자신이 전달한 이름이 가른 경우 아래의 부분 수정 필요 -->
  <a href="/board/update?id=${board.id}" class="button edit-button">수정</a>
  <form action="/board/delete/${board.id}" method="post">
    <input type="submit" class="button delete-button" value="삭제" />
  </form>
  <a href="/board/list" class="button back-button">목록으로</a>
</div>
```

# 게시글 삭제 기능 구현하기



- 요청 주소
  - POST 방식
  - <http://localhost:8080/board/delete/{id}>
- 필요 기능
  - PathVariable 방식으로 전달 받은 id 값을 가지는 게시글을 DB 에서 삭제하는 기능 구현

# 게시글 삭제 기능 구현하기



- 현재 상태

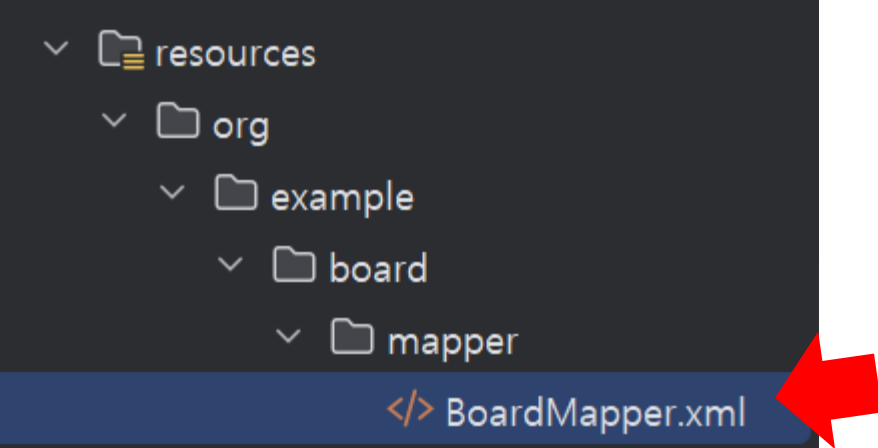
- BoardController 클래스의 delete 메서드에서 특정 id 를 가지는 게시글을 DB 에서 삭제하고 게시글 목록 페이지로 리다이렉트하는 기능이 구현이 안된 상태
- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태



```
52 // 문제 4. 게시물 삭제 기능 구현하기
53 @PostMapping(🌐"/delete") new *
54 🌐 ✓ public String delete() {
55     // 여기 부분에 코드를 완성하여, 게시물 삭제 기능을 완성시켜 주세요
56     return "redirect:/board/list";
57 }
```

PathVariable 방식(/board/delete/{id})으로 전달되는 id 값을 받아서 해당 id 를 가지는 게시글을 DB 에서 삭제하고 게시물 목록 페이지로 리다이렉트하는 컨트롤러 완성하기!





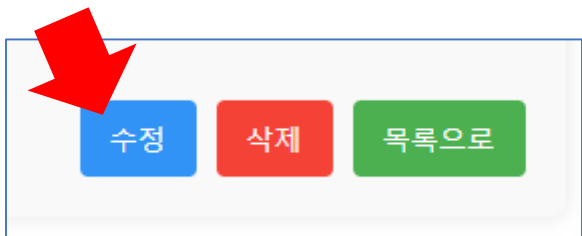
```
19      <delete id="delete">  
20      -- 문제 4. 게시물 삭제 기능 구현하기  
21      </delete>
```

적절한 SQL 구문을 작성해 주세요



# 문제 5

게시글 수정 기능 구현



수정 버튼을 클릭하면  
GET 방식 /board/update?id={id} 주소로  
요청을 보냅니다!



```
<div class="button-container">
  <!-- 자신이 전달한 이름이 가른 경우 아래의 부분 수정 필요 -->
  <a href="/board/update?id=${board.id}" class="button edit-button">수정</a>
  <form action="/board/delete/${board.id}" method="post">
    <input type="submit" class="button delete-button" value="삭제" />
  </form>
  <a href="/board/list" class="button back-button">목록으로</a>
</div>
```

# 문제 5-1, 게시글 수정 기능 구현하기 1



- 요청 주소
  - GET 방식
  - <http://localhost:8080/board/update?id={id}>
- 필요 기능
  - 파라미터 방식으로 전달 받은 id 값을 가지는 게시글을 찾아서 update.jsp 로 전달하는 기능 구현

# 문제 5-1, 게시물 수정 기능 구현하기 1



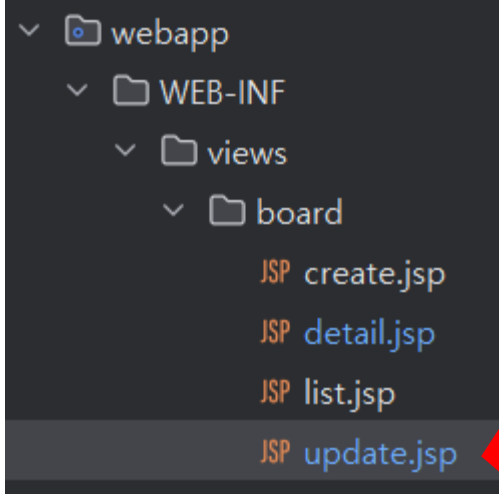
- 현재 상태

- BoardController 클래스의 updatePage 메서드에서 특정 id 를 가지는 게시글을 찾아서 update.jsp 로 전달하는 기능이 구현이 안된 상태
- 컨트롤러에서 전달한 데이터를 update.jsp 에서 출력하는 코드가 생략 된 상태



```
59 // 문제 5-1. 게시글 수정 기능 구현하기
60 @GetMapping(🌐"/update") 🧑 Tetz *
61 🌐 public String updatePage() {
62     // 여기 부분에 코드를 완성하여, 게시글 수정 페이지 이동 기능을 완성시켜 주세요
63     return context + "/update";
64 }
```

/board/update?id={id} 로 들어오는  
id 파라미터의 값을 받아서  
해당 id 를 가지는 게시글을 찾아서  
update.jsp 로 전달하는 컨트롤러 코드 완성 하기!



컨트롤러에서 전달한  
게시글의 데이터를  
update.jsp 의 각각 항목에  
적절하게 전달해야 합니다!

```
<h1>게시글 수정</h1>
<form action="/board/update" method="post">
  <!-- 문제 5-1. 게시글 수정 기능 구현하기 -->
  <!-- 게시글 ID 값 전달 필요 -->
  <label for="title">ID</label>
  <input type="text" name="id" disabled,
  <!-- 게시글 작성자 전달 필요 -->
  <label for="title">작성자</label>
  <input type="text" name="author" disabled,
  <label for="title">제목</label>
  <!-- 게시글 제목 전달 필요 -->
  <input type="text" id="title" name="title" required>
  <label for="content">내용</label>
  <!-- 게시글 내용 전달 필요 -->
  <textarea id="content" name="content" required></textarea>
  <input type="submit" value="수정" class="submit-button">
  <a href="/board/list" class="back-button">목록으로 돌아가기</a>
```

readonly 로  
수정

## 제목5 수정 테스트

작성자: test5

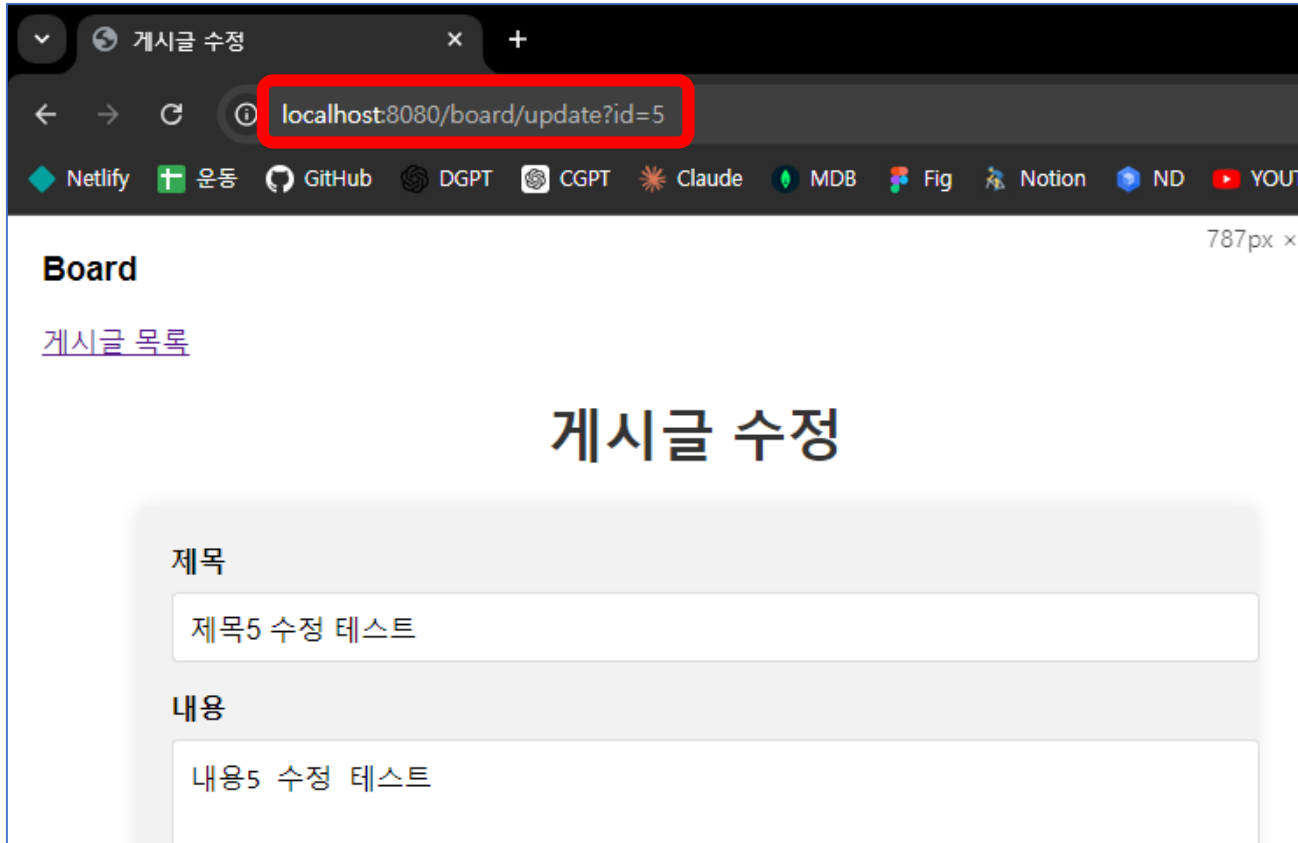
작성일: Tue Sep 03 00:40:15 KST 2024

내용5 수정 테스트

수정

삭제

목록으로



수정 버튼을 클릭하면 게시글의 수정 모드로 이동하는 기능을 완성하면 됩니다  
주소 창의 주소를 확인하시면 어떤 주소로 요청이 전달 되는지 확인할 수 있습니다!



# 게시글 수정 기능 구현하기 2



- 요청 주소
  - POST 방식
  - <http://localhost:8080/board/update>
- 필요 기능
  - update.jsp 페이지에서 전달 받은 값을 가지고 실제로 게시글을 수정하는 기능 구현
- 현재 상태
  - BoardController 클래스의 update 메서드에서 실제로 게시글을 수정하는 기능이 구현 안된 상태
  - BoardMapper.xml 에 sql 구문이 완성되지 못한 상태

# 게시글 수정 기능 구현하기 2



- 현재 상태

- BoardController 클래스의 update 메서드에서 실제로 게시글을 수정하는 기능이 구현 안된 상태
- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태
- BoardServiceImpl 클래스의 update 메서드의 내부 코드가 완성이 안된 상태

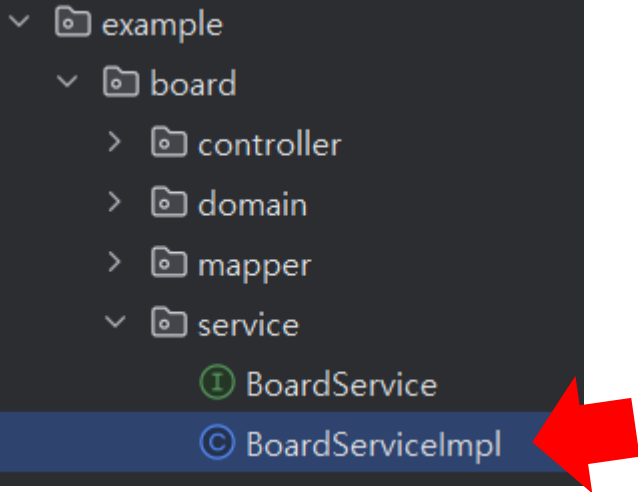


```
66 // 문제 5-2. 게시글 수정 기능 구현하기
67 @PostMapping(🌐"/update") 🧑 Tetz *
68 🌐 public String update() {
69     // 여기 부분에 코드를 완성하여, 게시글 수정 기능을 완성시켜 주세요
70     return "redirect:/board/list";
71 }
```

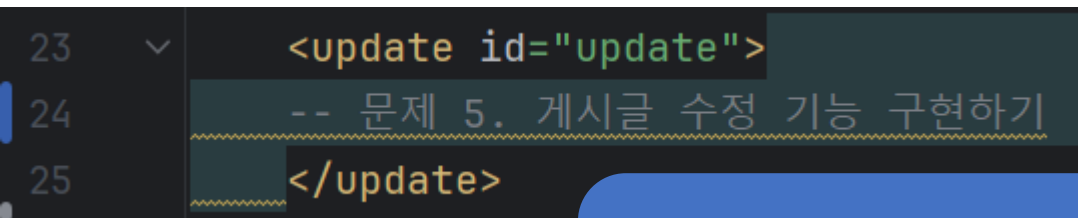
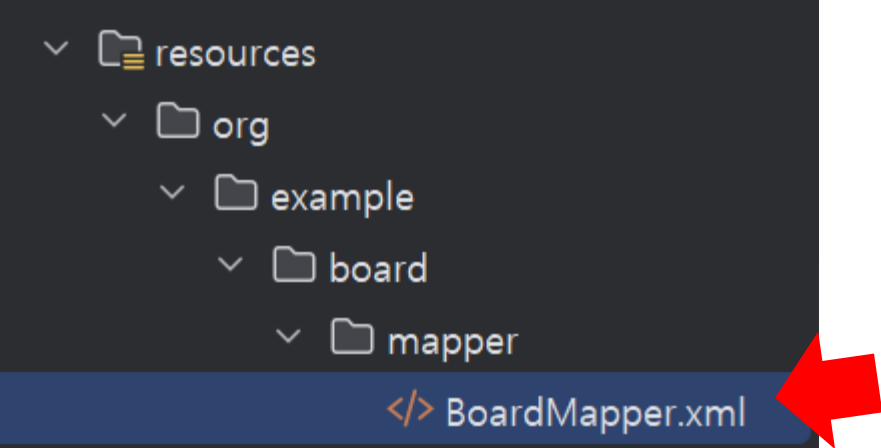
update.jsp 페이지에서 들어온 요청을 받아서  
실제 게시글을 수정한 후  
게시글 목록 페이지로 리다이렉트하는 기능 구현



BoardMapper 를 사용해서 게시글을  
실제로 업데이트하고, 업데이트 된 게시글을  
리턴하는 기능을 할 수 있도록  
메서드의 코드를 변경해 주세요



```
// 문제 5-2. 게시글 수정 기능 구현하기
@Override no usages Tetz *
public Board update(Board board) {
    // 아래의 코드는 JAVA 의 컴파일 에러를 막기 위한 코드입니다.
    // 조건에 맞는 코드로 코드를 수정하여 주세요.
    // 조건 : boardMapper 를 이용하여 실제로 게시글을 업데이트한 이후, 업데이트 된 게시글을 리턴하는 기능
    Board tempBoard = new Board();
    return tempBoard;
}
```



적절한 SQL 구문을 작성해 주세요



# 문제 6

## REST Controller

### 만들기

<https://github.com/xenosign/rest-board>



위의 코드를 클론 받아 주세요!

클론 후 → npm install → npm run dev 로 실행 시켜주시면 됩니다!

해당 코드는 Vue3 로 만들어진 게시판이며 문제 5번 까지의 내용 중,

게시글 목록 호출  
특정 id 를 가지는 게시글 데이터 전달  
게시글 작성  
게시글 수정  
게시글 삭제 기능을

REST API 형태로 만들면  
모든 기능이 정상 작동하도록 만들어져 있습니다!

# rest-vue 게시판 설명



- 요청 주소

- 기존의 모든 요청 주소 앞에 rest 만 붙이면 됩니다!

- ex) <http://localhost:8080/board/list> → <http://localhost:8080/rest/board/list>



- 모든 응답은 ResponseEntity 를 활용하여 응답을 보내시면 됩니다.
- 게시글 작성과, 수정은 Request 의 Body 를 통해서 전달 됩니다. 따라서, Spring 서버에서 받을 때에는 @RequestBody 어노테이션을 통해서 받으시면 됩니다! 물론, 프론트에서 Board 도메인에과 일치하는 형태를 가진 객체로 전달이 됩니다!



# rest-vue 게시판 설명



- rest-vue 게시판에서 요청하는 주소 모음
- 게시글 목록
  - GET / <http://localhost:8080/rest/board/list>
- 특정 ID 를 가지는 게시글
  - GET / <http://localhost:8080/rest/board/detail/{id}>
- 게시글 작성
  - POST / <http://localhost:8080/rest/board/create>
- 게시글 삭제
  - POST / <http://localhost:8080/rest/board/delete>
- 게시글 수정
  - POST / <http://localhost:8080/rest/board/update>

## 게시글 목록 페이지

### 게시판

글 작성



제목5

2024-09-03

내용5

작성자: test5

제목4

2024-09-03

내용4

작성자: test4

제목3

2024-09-03

내용3

작성자: test3

제목2

2024-09-03

내용2

작성자: test2

제목1

2024-09-03

내용1

작성자: test1

# 게시글 작성 페이지 및 기능



작성자

tetz

제목

게시글 작성 테스트

내용

게시글 작성 테스트

게시글 작성

취소

## 게시판

글 작성

게시글 작성 테스트

2024-09-03

게시글 작성 테스트

작성자: tetz

제목5

2024-09-03

내용5

작성자: test5

# 게시글 세부 내용 페이지



## 게시판

글 작성

게시글 작성 테스트

2024-09-03

게시글 작성 테스트

작성자: tetz

제목5

내용5

작성자: test5

## 게시글 작성 테스트

작성자: tetz

작성일: 2024-09-03

게시글 작성 테스트

수정

삭제

목록으로

# 게시글 삭제 기능



## 게시글 수정 테스트

작성자: tetz수정

작성일: 2024-09-03

게시글 수정 테스트

수정

삭제

목록으로

## 게시판

글 작성

제목5

2024-09-03

내용5

작성자: test5

제목4

2024-09-03

내용4

작성자: test4

# 게시글 수정 페이지 및 기능



작성자

tetz수정

제목

게시글 수정 테스트

내용

게시글 수정 테스트

수정 완료

취소

## 게시글 수정 테스트

작성자: tetz수정

게시글 수정 테스트

수정

삭제

목록으로



수고하셨습니다!