



It's Your Life

with





# Module 평가

## 대비 문제!



[https://drive.google.com/file/d/1K0h\\_kGQMV-BJfczVo\\_vg9fwsE1AiNbh8/view?usp=drive\\_link](https://drive.google.com/file/d/1K0h_kGQMV-BJfczVo_vg9fwsE1AiNbh8/view?usp=drive_link)

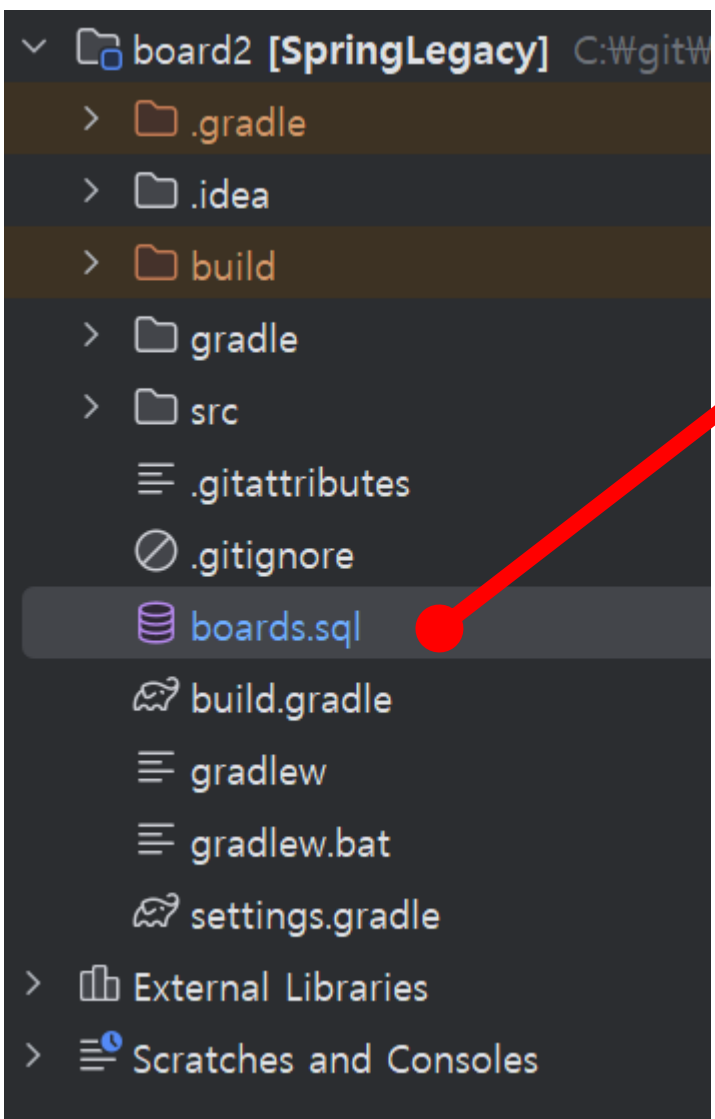
위의 주소에 들어가서 zip 파일을 다운로드 후 압축을 해제 후  
board2 폴더를 인텔리제이로 열어 주시면 됩니다!

# DB 구축하기





폴더 제일 외부에 존재하는 boards.sql 파일을 열어 주세요



```

create database module_db;
use module_db;

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
);

INSERT INTO boards(title, content, author)
VALUES ('제목1', '내용1', 'test1'),
       ('제목2', '내용2', 'test2'),
       ('제목3', '내용3', 'test3'),
       ('제목4', '내용4', 'test4'),
       ('제목5', '내용5', 'test5');

SELECT * FROM boards;

```

해당 쿼리문을 사용하여  
module\_db 데이터 베이스를 만들고

boards 테이블을 만든 다음  
테스트 데이터를 넣어 주시면 됩니다!

Result Grid

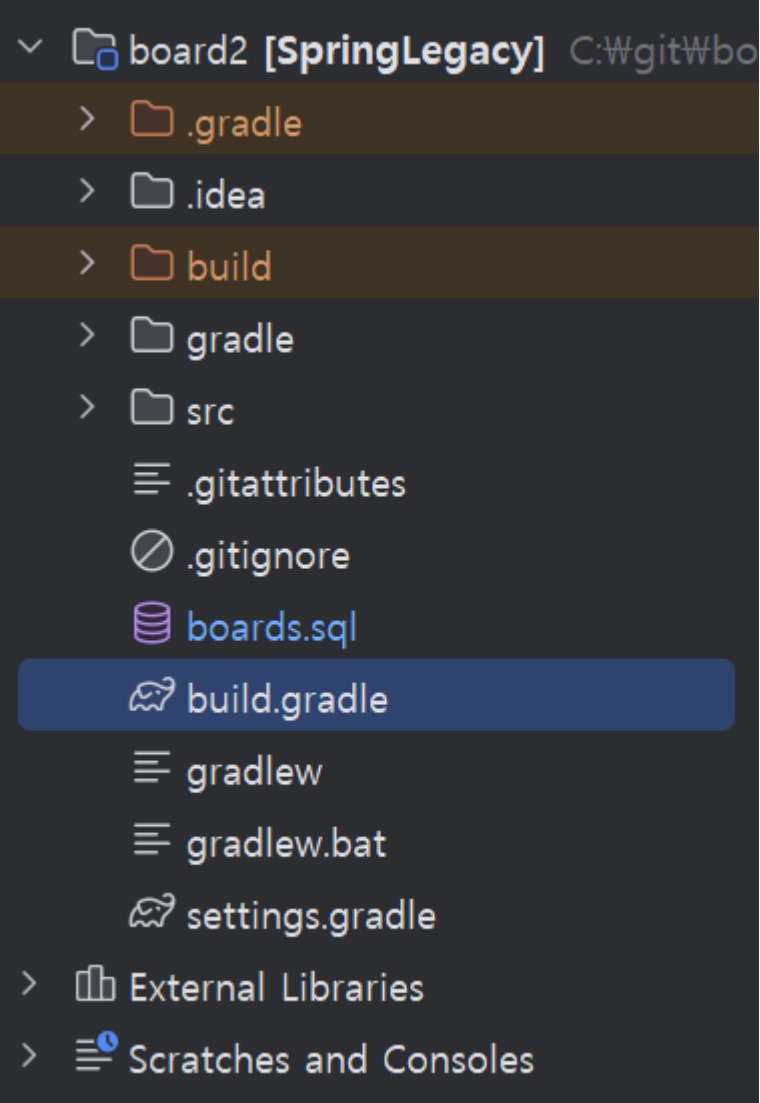
Filter Rows:

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 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}")

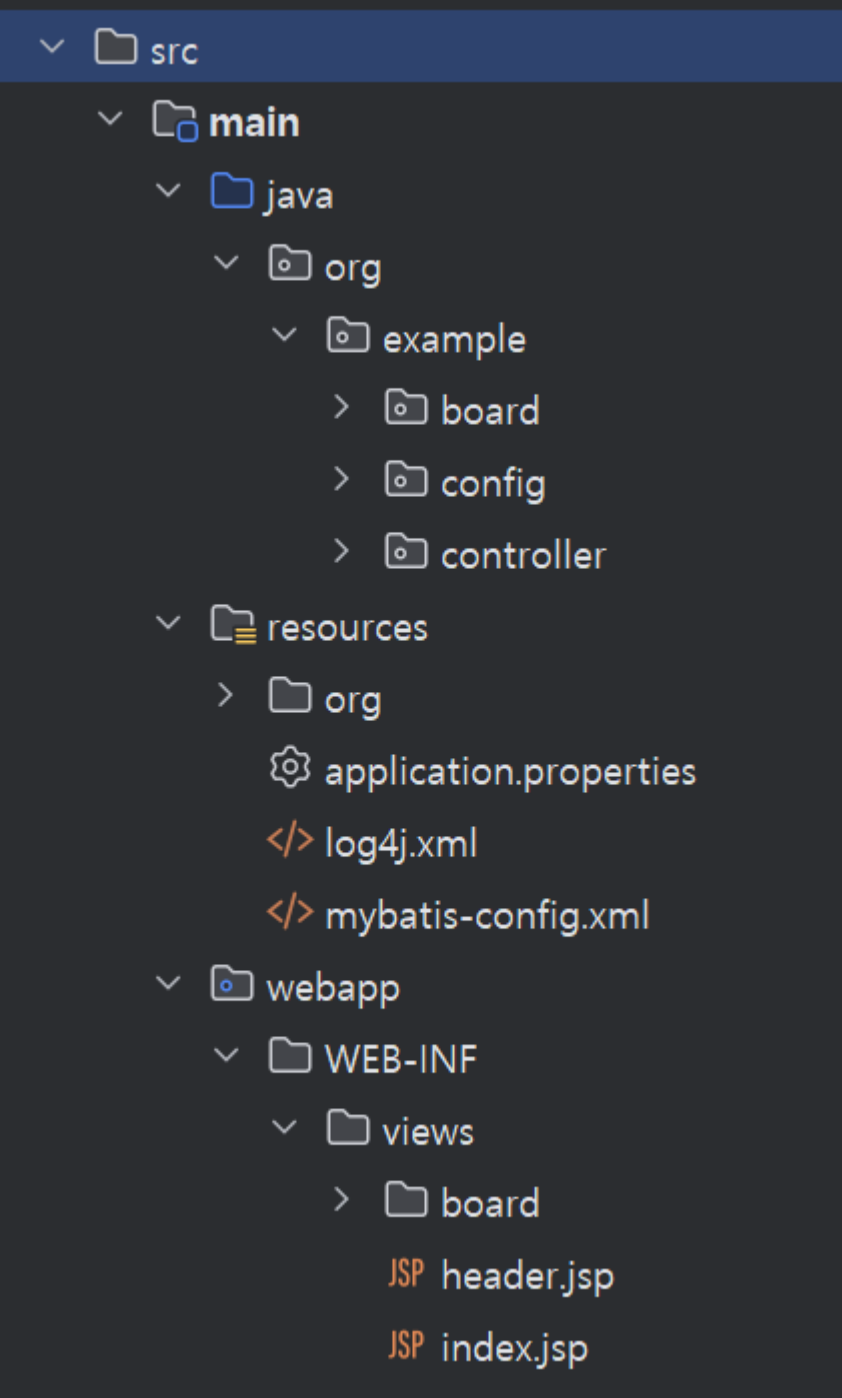
// 데이터 베이스
implementation 'com.mysql:mysql-connector-j:8.1.0'
implementation 'com.zaxxer:HikariCP:2.7.4'
implementation "org.springframework:spring-tx:${springVersion}"
implementation "org.springframework:spring-jdbc:${springVersion}"
implementation 'org.mybatis:mybatis:3.4.6'
implementation 'org.mybatis:mybatis-spring:1.3.2'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



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!!

즉, org.example.board 패키지 내부의  
코드와 views/board 폴더 내부의  
코드만 작성하시면 됩니다!

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

마이바티스 설정에 MapUnderscoreToCamelCase  
옵션이 true 로 설정 되어 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;

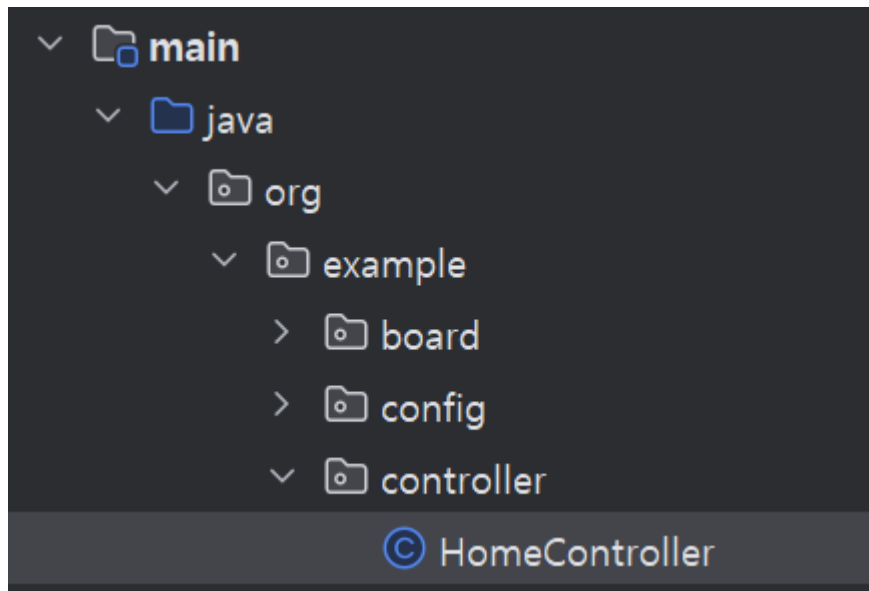
private String content;

private String author;

private Date regDate; |



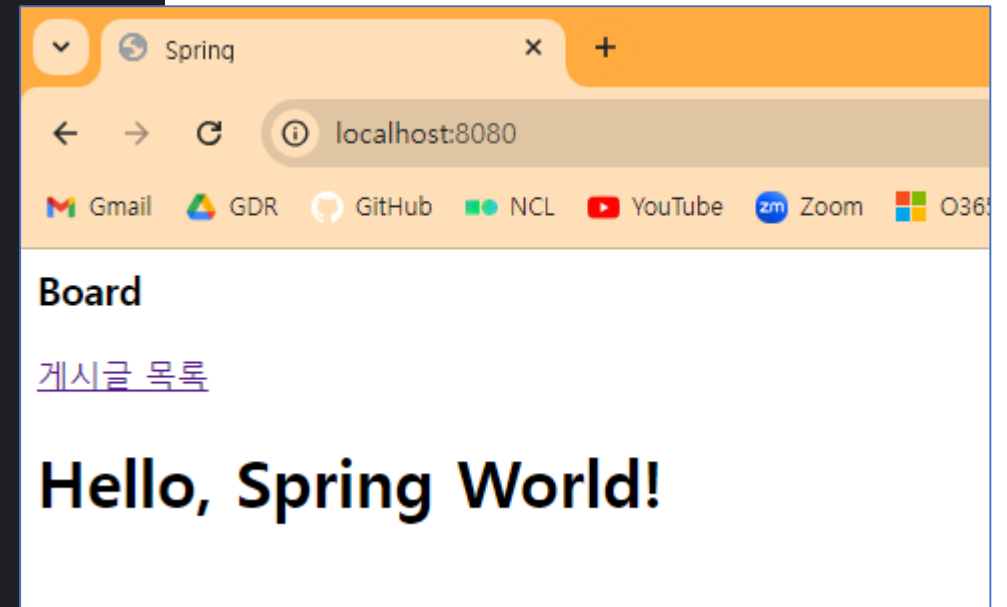
따라서 reg\_date 컬럼은 자동으로 Board 클래스의  
regDate 로 매핑이 됩니다!



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!



```

 20
 21 @Configuration   Tetz
 22 @PropertySource({"classpath:/application.properties"})
 23 @ComponentScan(basePackages = {"org.example"})
 24 @MapperScan(basePackages = {"org.example.board.mapper"})
 25 @Slf4j
 26 public class RootConfig {
 27     @Value("${com.mysql.jdbc.Driver}") String driver;
```



# ServletConfig 와 WebConfig 역시 마찬가지로입니다!

config	10	
RootConfig	11	@EnableWebMvc 1 usage Tetz
ServletConfig	12	@ComponentScan(basePackages = {"org.example"})
WebConfig	13	public class ServletConfig implements WebMvcConfigurer {
controller	14	// Spring MVC용 컴포넌트 등록을 위한 스캔 패키지

config	1	package org.example.config;
RootConfig	2	
ServletConfig	3	> import ...
WebConfig	6	public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {
controller	7	@Override no usages Tetz
resources	8	> protected Class<?>[] getRootConfigClasses() { return new Class[] { RootConfig.class; }
org	11	
application.properties	12	@Override no usages Tetz
log4j.xml	13	> protected Class<?>[] getServletConfigClasses() { return new Class[] { ServletConfig.class; }
mybatis-config.xml	14	

# 프로젝트 세팅



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

application.properties 파일을 열어 주세요







```
jdbc.driver=com.mysql.cj.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/module_db  
jdbc.username=root  
jdbc.password=1234|
```

본인이 사용하는  
MySQL 의 ID 와 비번으로  
변경해 주세요!

데이터 베이스는 방금 만든  
module\_db 로 세팅이 되어 있으니  
그대로 두시면 됩니다!

혹, 문제에서 제공 된 DB 명과  
해당 url 의 DB 명이 다를 경우 반드시  
동일하게 맞춰 주셔야 합니다!



자~ 이제 시작이야(내꿈을)



# 문제 1

게시판 목록 기능 구현

# 게시판 목록 기능 완성하기



- 요청

- GET 방식

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

- 필요 기능

- boards 테이블의 내용을 읽어서, 목록을 출력해 주는 기능 구현하기

- 현재 상태

- BoardController 클래스의 listPage 메서드에서 데이터를 전달하고 있지 못한 상태

- BoardMapper.xml 에 sql 구문이 완성되지 못한 상태

기능을 완성하여 왼쪽과 같은 화면을  
구현하시면 됩니다!

## 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

새글 작성하기



```
@Controller  Tetz *
@Log4j
@RequestMapping(🌐"/board")
@RequiredArgsConstructor
public class BoardController {
    // @RequiredArgsConstructor 에 의해 의존성이 자동으로 주입 됩니다
    private final BoardService boardService;
    private final String context = "/board"; 4 usages

    @GetMapping(🌐"/list")  Tetz *
    public String listPage() {
        return context + "/list";
    }
}
```

BoardService 를 사용하여  
게시글의 데이터를 받아온 다음  
list.jsp 로 전달하면 됩니다!



resources  
org  
example  
board  
mapper

BoardMapper.xml

```
<mapper namespace="org.example.board.mapper.BoardMapper">  
  <select id="getList" resultType="Board">  
      
  </select>
```

여기에 SQL 구문 완성 필요!

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>ID</th>
```

```
<th>Title</th>
```

```
<th>Content</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
<c:forEach var="board" items="${boardList}">
```

```
<tr>
```

```
<td>${board.id}</td>
```

```
<td><a href="/board/detail?id=${board.id}">${board.title}</a></td>
```

```
<td>${board.content}</td>
```

```
</tr>
```

```
</c:forEach>
```

```
</tbody>
```

```
</table>
```

list.jsp 페이지에서는  
게시글 데이터 배열을 boardList 로 받습니다!







# 문제 2

# 게시글 작성 기능 구현

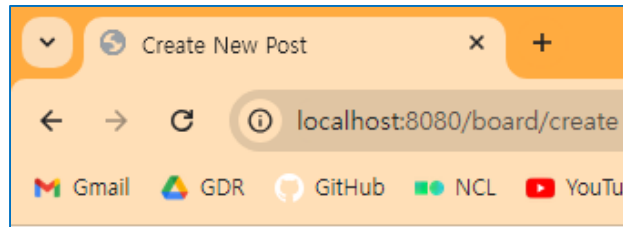
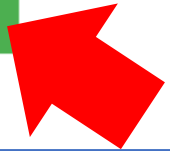
# 게시글 작성 기능 완성하기



- 요청 주소
  - POST 방식
  - <http://localhost:8080/board/create>
- 필요 기능
  - 새롭게 입력 받은 게시글의 내용을 DB 에 등록하고, 목록 출력 페이지로 이동
- 현재 상태
  - BoardController 클래스의 create 메서드에서 게시글 등록을 못하고 있는 상태
  - BoardMapper.xml 에 sql 구문이 완성되지 못한 상태

4	<a href="#">제목4</a>
3	<a href="#">제목3</a>
2	<a href="#">제목2</a>
1	<a href="#">제목1</a>

새글 작성하기



## Board

[게시글 목록](#)

## 글 작성

제목

내용

글 작성

취소

글 작성 버튼을 누르면 글이 완성 되도록  
게시글 작성 기능을 아래의 설명에 따라  
완성해 주세요!

```
<%@include file="../header.jsp"%>
<h1>글 작성</h1>
<form action="/board/create" method="post">
  <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="hidden" name="author" value="test6" />
  <input type="submit" value="글 작성">
  <a class="cancel-button" href="/board/list">취소</a>
</form>
```

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

로그인 기능이 없으므로  
글쓴이는 author 라는  
이름으로 전달 합니다  
(단, 숨겨져 있음)

```
// 게시글 작성 모드 페이지
@GetMapping("/create")
public String createPage() {
    return context + "/create";
}
```

위 컨트롤러 메서드는 게시글 작성 모드 페이지를 호출하는 기능으로 이미 완료된 상태입니다!

```
// 게시글 작성 기능
@PostMapping("/create")
public String create() {
    return "redirect:/board/list";
}
```

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

```
<insert id="create">
```

```
</insert>
```



여기에 SQL 구문 완성 필요!

글을 작성 후, 버튼을 누르면  
글이 등록되고 글 목록 페이지로 이동하여  
작성한 글이 보이면 됩니다!!

Board

[게시글 목록](#)

글 작성

제목

테스트 글 입니다

내용

테스트 글 입니다!

글 작성

취소



[게시글 목록](#)

글 목록

ID	Title	Content
6	<a href="#">테스트 글 입니다</a>	테스트 글 입니다!
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

새글 작성하기



# 문제 3

게시글 내용 보기

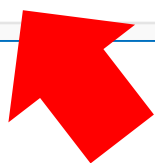
기능 구현



## 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!

수정삭제목록으로

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

# 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
- 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

Rename usages

```
@GetMapping("/detail")  
public String detailPage() {  
    return context + "/detail";  
}
```

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

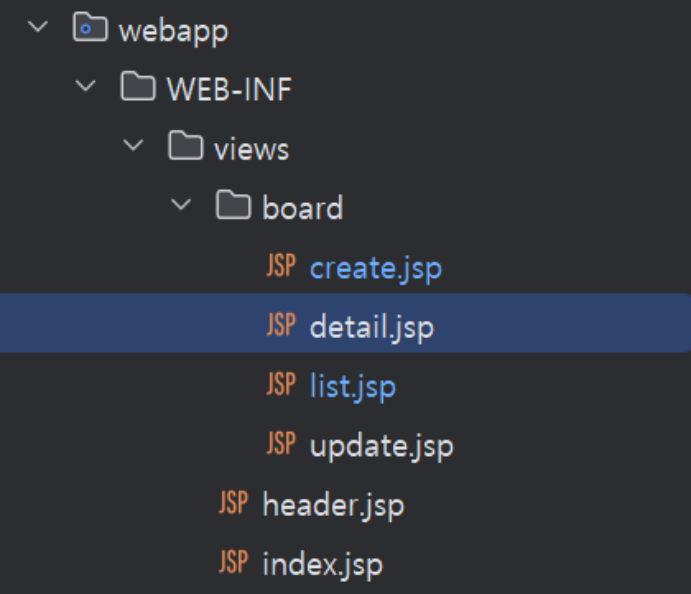
기존에 완성된 jsp 파일의 변경을 막기 위해서  
Model 에 전달하는 key 의 이름은 board 로  
통일해 주세요 😊

```
<select id="detail" resultType="Board">
```

```
</select>
```



여기에 SQL 구문 완성 필요!



자신이 전달한 데이터를 각각의 항목에 맞게  
출력하는 코드 작성 필요!

```
<div class="post-container">
  <h1 class="post-title"> <!-- 제목 넣기 --> </h1>
  <div class="post-content">
    <div>작성자: <!-- 작성자 넣기 --></div>
    <div>작성일: <!-- 작성일 넣기 --></div>
    <div>
      <!-- 게시글 내용 넣기 -->
    </div>
  </div>
  <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>
</div>
```

삭제, 수정 기능에  
필요한 id 전달



# 문제 4

게시글 삭제 기능 구현



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

수정

삭제

목록으로

```
<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
- 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



```
@PostMapping(🌐"/delete")  👤 Tetz *  
public String delete() {  
    return "redirect:/board/list";  
}
```



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



```
<delete id="delete">
```

```
</delete>
```

여기에 SQL 구문 완성 필요!



# 문제 5

게시글 수정 기능 구현



수정

삭제

목록으로

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>
```



# 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
- 현재 상태
  - BoardController 클래스의 updatePage 메서드에서 특정 id 를 가지는 게시글을 찾아서 update.jsp 로 전달하는 기능이 구현이 안된 상태
  - BoardMapper.xml 에 sql 구문이 완성되지 못한 상태

```
@GetMapping(🌐"/update") new *  
public String updatePage() {  
    return context + "/update";  
}
```

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



```
<update id="update">
```

```
</update>
```

여기에 SQL 구문 완성 필요!

세부 내용 확인 용 글입니다!

작성자: test6

작성일: Mon Sep 02 13:02:43 KST 2024

세부 내용 입니다!

수정



## 게시글 수정

제목

세부 내용 확인 용 글입니다!

내용

세부 내용 입니다!

수정

목록으로 돌아가기




# 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

```
@PostMapping(🌐✓"/update")  👤 Tetz *  
public String update() {  
    return "redirect:/board/list";  
}
```



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

```
<update id="update">
```

```
</update>
```



여기에 SQL 구문 완성 필요!

# 게시글 수정



제목

세부 내용 확인 용 글입니다! 수정!!

내용

세부 내용 입니다! 수정!!

수정

목록으로 들

## Board

[게시글 목록](#)

## 글 목록

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