

2. Spring Boot Mybatis 및 jsp 사용

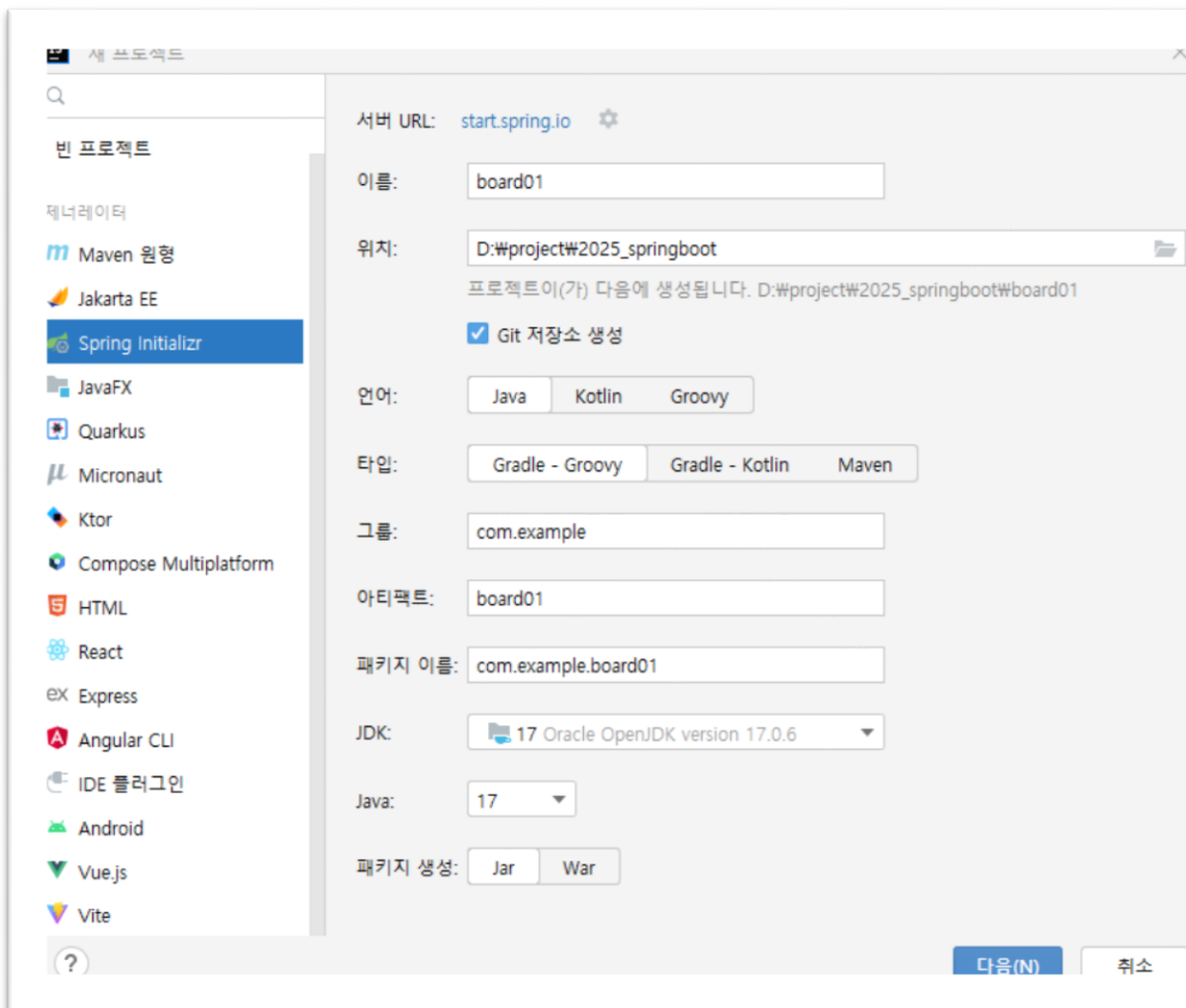
1. Mybatis 활용 프로젝트 생성
2. 사용 라이브러리 추가
3. application.properties
4. 테스트 활용
5. Mybatis 활용 DB 연결
6. Service 구현
7. Controller 구현
8. View 구현

Mybatis 실습 예제

- ❖ 데이터베이스 : springbootdb
- ❖ 테이블 : tbl_board
- ❖ MCV 패턴 작성 : mapper, dto, service, controller
- ❖ 구현내용 : list, register, read, update, delete

1. Mybatis 활용 프로젝트 생성

❖ 프로젝트 생성 및 종속성 추가



새 프로젝트

빈 프로젝트

제너레이터

- Maven 원형
- Jakarta EE
- Spring Initializr
- JavaFX
- Quarkus
- Micronaut
- Ktor
- Compose Multiplatform
- HTML
- React
- Express
- Angular CLI
- IDE 플러그인
- Android
- Vue.js
- Vite

서버 URL: start.spring.io

이름: board01

위치: D:\project\2025_springboot
프로젝트(가) 다음에 생성됩니다. D:\project\2025_springboot\board01

☒ Git 저장소 생성

언어: Java Kotlin Groovy

타입: Gradle - Groovy Gradle - Kotlin Maven

그룹: com.example

아티팩트: board01

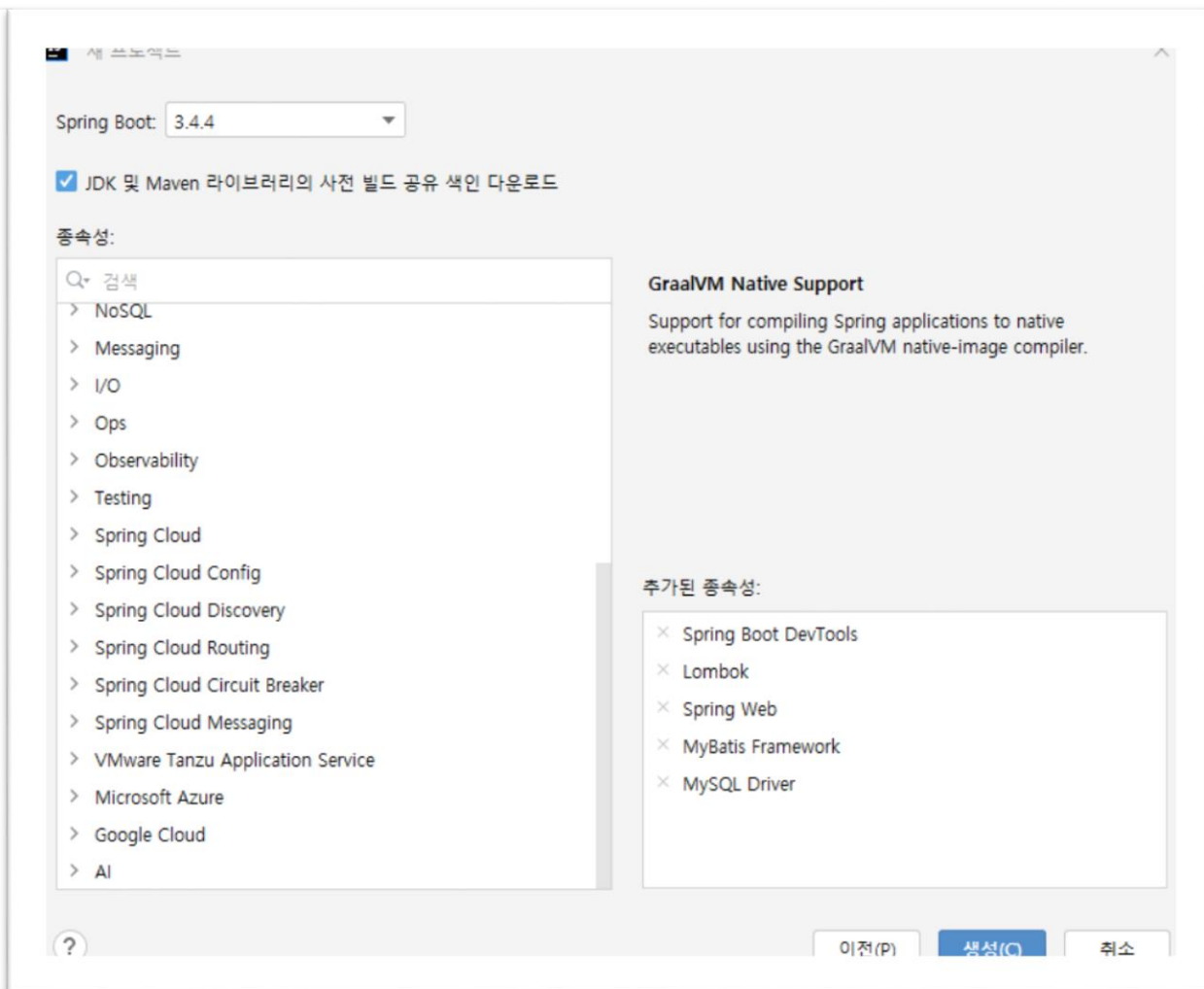
패키지 이름: com.example.board01

JDK: 17 Oracle OpenJDK version 17.0.6

Java: 17

패키지 생성: Jar War

다음(N) 취소



새 프로젝트

Spring Boot: 3.4.4

☒ JDK 및 Maven 라이브러리의 사전 빌드 공유 색인 다운로드

종속성:

검색

- NoSQL
- Messaging
- I/O
- Ops
- Observability
- Testing
- Spring Cloud
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Routing
- Spring Cloud Circuit Breaker
- Spring Cloud Messaging
- VMware Tanzu Application Service
- Microsoft Azure
- Google Cloud
- AI

GraalVM Native Support

Support for compiling Spring applications to native executables using the GraalVM native-image compiler.

추가된 종속성:

- × Spring Boot DevTools
- × Lombok
- × Spring Web
- × MyBatis Framework
- × MySQL Driver

이전(P) 생성(C) 취소

2. 사용 라이브러리 추가

❖ build.gradle에 jasper와 jstl 종속성 추가(Maven과 Gradle 비교)

```
// https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-jasper
implementation group: 'org.apache.tomcat.embed', name: 'tomcat-embed-jasper'

// https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.servlet.jsp.jstl-api
implementation 'jakarta.servlet:jakarta.servlet-api' //스프링부트 3.0 이상
implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api' //스프링부트 3.0 이상
implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl' // 스프링부트 3.0 이상
```

3. application.properties

❖ 데이터베이스 설정

Application.properties

server.port=8081

#database setting

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/springbootdb

spring.datasource.username=pgm

spring.datasource.password=1234

#view setting

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

3. application.properties

❖ 로그레벨 설정

```
server.port=8081
```

```
#database setting
```

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/springbootdb
```

```
spring.datasource.username=pgm
```

```
spring.datasource.password=1234
```

```
#log setting
```

```
logging.level.org.springframework=info
```

```
logging.level.com.example=debug
```

```
#view setting
```

```
spring.mvc.view.prefix=/WEB-INF/views/
```

```
spring.mvc.view.suffix=.jsp
```

4. 테스트 활용

❖ Datasource 테스트

- build.gradle dependencies에 종속성 추가

```
testCompileOnly 'org.projectlombok:lombok'  
testAnnotationProcessor  
'org.projectlombok:lombok'
```

DB 연결 테스트 코드

```
package com.example.board01;  
  
import lombok.extern.log4j.Log4j2;  
import org.junit.jupiter.api.Test;  
import  
org.springframework.beans.factory.annotation.Autowired;  
import  
org.springframework.boot.test.context.SpringBootTest;  
import javax.sql.DataSource;  
import java.sql.Connection;  
import java.sql.SQLException;  
  
@SpringBootTest  
@Log4j2  
public class DataSourceTest {  
    @Autowired  
    private DataSource dataSource;  
  
    @Test  
    public void testConnection() throws SQLException{  
  
        Connection conn=dataSource.getConnection();  
        log.info(conn);  
    }  
}
```

5. Mybatis 활용 DB 연결

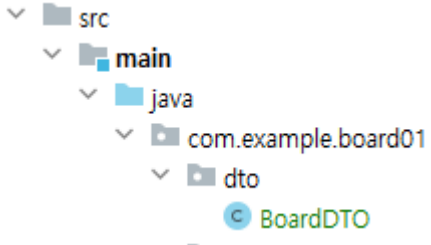
❖ mysql에 테이블 작성

- 테이블 명 :tbl_board
- 필드
 - bno : int, primary key, auto increment
 - title : varchar(250), not null
 - writer : varchar(45) not null
 - content : varchar(1000) not null
 - postdate : datetime default now()
 - readcount: int default 0

```
CREATE TABLE tbl_board(  
  bno INT NOT NULL AUTO_INCREMENT,  
  title VARCHAR(250) NOT NULL,  
  writer VARCHAR(45) NOT NULL,  
  content VARCHAR(1000) NOT NULL,  
  postdate DATETIME NULL DEFAULT now(),  
  readcount INT NULL DEFAULT 0,  
  PRIMARY KEY (`bno`));
```


5. Mybatis 활용 DB 연결

❖ DTO 클래스 만들기



```
package com.example.demo.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

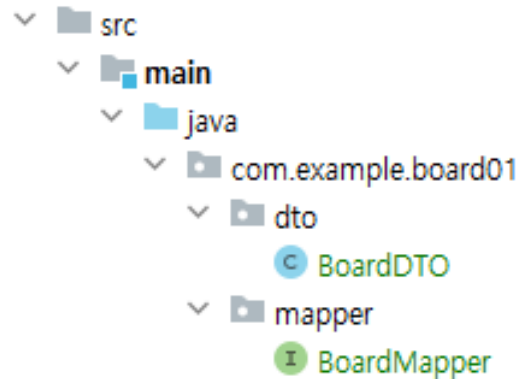
import java.util.Date;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class BoardDTO {
    private int bno;
    private String title;
    private String content;
    private String writer;
    private Date postdate;
    private int readcount;
}
```

- DTO(Data Transfer Object) vs VO(Value Object)
 - 가변성
DTO: 값이 변할 수 있는 가변 객체
VO: 값이 변하지 않는 불변 객체
 - 사용 범위
DTO: 주로 레이어 간 데이터 전송에 사용
VO: 모든 레이어에서 사용 가능하며, 값 자체를 표현
 - 동등성 비교
DTO: 내부 속성 값이 같아도 다른 객체로 식별.
VO: 내부 속성 값이 같다면 같은 객체로 간주, 이를 위해 equals()와 hashCode() 메서드를 오버라이드함.
 - 기능
DTO: 데이터 접근 이외의 기능을 가지지 않으며, 주로 getter/setter 메서드만 포함
VO: 특정 비즈니스 로직을 포함할 수 있음
 - 공통점
둘 다 레이어 간 데이터 전달에 사용될 수 있음.
데이터의 표현과 전달에 중요한 역할을 함.
 - 사용 목적
DTO: 효율적인 데이터 전송과 코드의 간결성을 위해 사용
VO: 데이터의 불변성과 동등성을 보장, 특정 값 자체 표현하는 데 사용

5. Mybatis 활용 DB 연결

❖ Mapper interface



```
package com.example.board01.mapper;
```

```
import com.example.board01.dto.BoardDTO;
import org.apache.ibatis.annotations.Mapper;
```

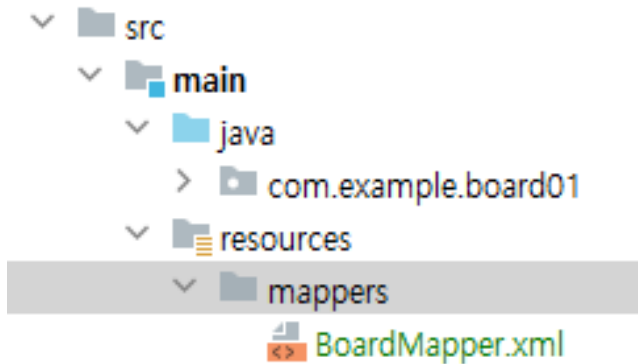
```
import java.util.List;
```

```
@Mapper
```

```
public interface BoardMapper {
    List<BoardDTO> selectAll();
    BoardDTO selectOne(int bno);
    void insert(BoardDTO board);
    void update(BoardDTO board);
    void delete(int bno);
    void readCountUpdate(int bno);
}
```

5. Mybatis 활용 DB 연결

❖ mybatis mapper 작성



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.board01.mapper.BoardMapper">
  <select id="selectAll" resultType="BoardDTO">
    select * from tbl_board
  </select>
  <select id="selectOne" resultType="BoardDTO">
    select * from tbl_board where bno=#{bno}
  </select>
  <insert id="insert">
    insert into tbl_board(title, content, writer) values("#{title}, #{content}, #{writer})
  </insert>
  <update id="update">
    update tbl_board set title=#{title}, content=#{content}, writer=#{writer} where bno=#{bno}
  </update>
  <delete id="delete">
    delete from tbl_board where bno=#{bno}
  </delete>
  <update id="readCountUpdate">
    update tbl_board set readcount=readcount+1 where bno=#{bno}
  </update>
</mapper>
```

5. Mybatis 활용 DB 연결

❖ application.properties 설정

- xml mapper와 dto class 위치 설정

```
#xml mapper
```

```
locationmybatis.mapper-locations=classpath:mappers/**/*.xml
```

```
#dto location
```

```
mybatis.type-aliases-package=com.example.demo.dto
```

5. Mybatis 활용 DB 연결

❖ CRUD 테스트-1

```
// insert test
public class DataSourceTest {
    @Autowired
    private DataSource dataSource;
    @Autowired
    private BoardMapper boardMapper;

    @Test
    public void testInsert() { //데이터 추가
        BoardDTO dto=new BoardDTO();
        dto.setTitle("title1");
        dto.setContent("content1");
        dto.setWriter("user00");
        boardMapper.insert(dto);
    }
}
```

```
@Test
public void testSelectAll(){
    List<BoardDTO> dtos=boardMapper.selectAll();
    for(int i=0; i<dtos.size(); i++){
        BoardDTO dto=dtos.get(i);
        log.info(dto);
    }
}
```

```
@Test
public void testSelectOne(){
    BoardDTO dto=boardMapper.selectOne(1);
    log.info(dto);
}
```

5. Mybatis 활용 DB 연결

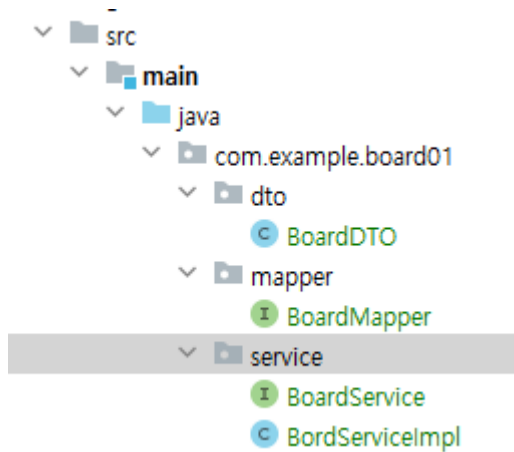
❖ CRUD 테스트-2

```
@Test
public void testUpdate(){
    BoardDTO updateDto=new BoardDTO();
    updateDto.setBno(1);
    updateDto.setTitle("수정 제목");
    updateDto.setContent("수정 내용");
    updateDto.setWriter("user11");
    boardMapper.update(updateDto);
}
```

```
@Test
public void testDelete(){
    boardMapper.delete(1);
}
```

6. Service 구현

❖ BoardService 인터페이스와 BoardServiceImpl 구현



```
package com.example.board01.service;

import com.example.board01.dto.BoardDTO;
import java.util.List;

public interface BoardService {
    List<BoardDTO> getList();
    BoardDTO getOne(int bno);
    void register(BoardDTO boardDTO);
    void modify(BoardDTO boardDTO);
    void remove(int bno);
}
```

6. Service 구현

❖ BoardService 인터페이스와 BoardServiceImpl 구현

```
package com.example.board01.service;

import com.example.board01.dto.BoardDTO;
import com.example.board01.mapper.BoardMapper;
import com.example.board01.mapper.BoardMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Service;

import java.util.List;
@Service
@Log4j2
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService{
    private final BoardMapper boardMapper;

    @Override
    public List<BoardDTO> getList() {
        return boardMapper.selectAll();
    }
}
```

```
@Override
public BoardDTO getOne(int bno) {
    boardMapper.visitCountUpdate(bno);
    return boardMapper.selectOne(bno);
}

@Override
public void register(BoardDTO boardDTO) {
    boardMapper.insert(boardDTO);
}

@Override
public void modify(BoardDTO boardDTO) {
    BoardDTO dto=boardMapper.selectOne(boardDTO.getBno());
    boardMapper.update(boardDTO);
}

@Override
public void remove(int bno) {
    boardMapper.delete(bno);
}
}
```


7. Controller 구현

❖ BoardController

```
@Controller
@Log4j2
@RequestMapping("/board")
public class BoardController {
    @Autowired
    private BoardService boardService;

    @GetMapping("/list")
    public void list(Model model) {
        log.info("list");
        model.addAttribute("boardList",boardService.getList());
    }
    @GetMapping("/register")
    public void regisertGet(){
        log.info("regisertGet");
    }
    @PostMapping("/register")
    public String registerPost(BoardDTO boardDTO) {
        log.info("registerPost");
        boardService.register(boardDTO);
        return "redirect:/board/list";
    }
}
```

```
@GetMapping({"/read","/modify"})
public void read(@RequestParam("bno") int bno, Model model) {
    log.info("read");
    BoardDTO dto=boardService.getOne(bno);
    model.addAttribute("dto",dto);
}
@PostMapping("/modify")
public String modifyPost(BoardDTO boardDTO) {
    log.info("modifyPost");
    boardService.modify(boardDTO);
    return "redirect:/board/read?bno="+boardDTO.getBno();
}
@GetMapping("/remove")
public String remove(int bno) {
    log.info("remove");
    boardService.remove(bno);
    return "redirect:/board/list";
}
}
```

8. View 구현

❖ Views 구성 요소

- board 폴더
 - 리스트 페이지 : list.jsp
 - 데이터 등록페이지 : register.jsp
 - 상세보기 페이지 : read.jsp
 - 수정페이지 : update.jsp
- includes
 - 페이지 상단 요소: footer.jsp
 - 페이지 하단요소 : header.jsp
- jsp 샘플코드 : test.jsp

