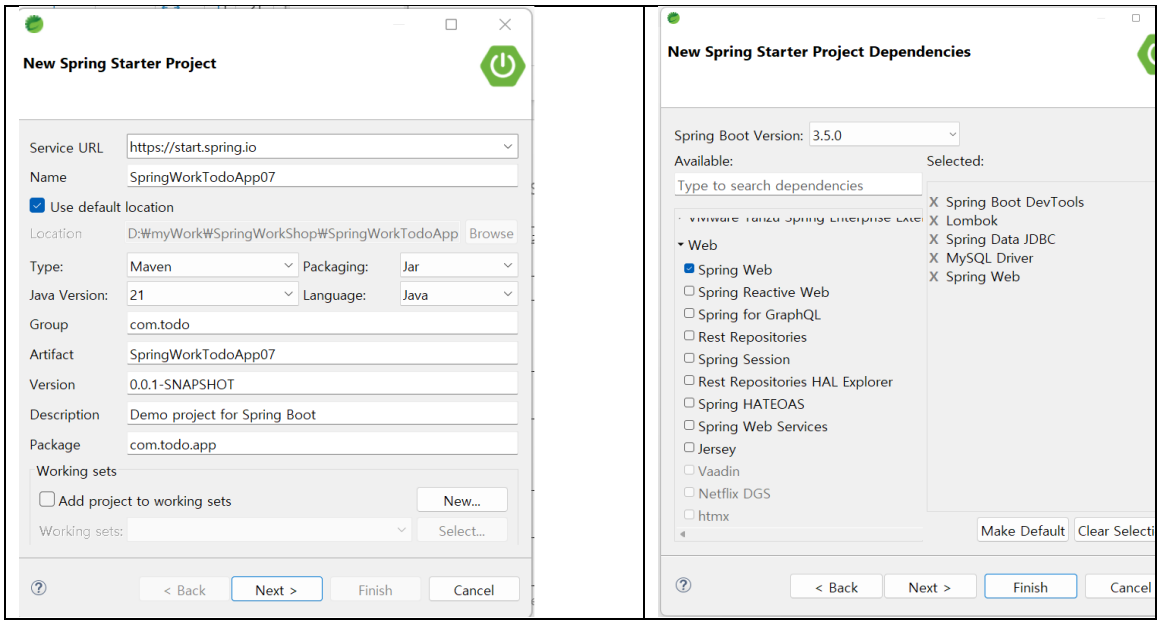
[문제 1] Spring Boot 기반 MySQL TodoAPP 을 다음과 같은 조건으로 구현합니다.

[구현 단계 요약]

1. 요구사항 분석: 등록, 조회, 수정, 삭제
2. 개념적 설계: TodoItem 엔티티 정의
3. 논리적/물리적 설계: MySQL 테이블 설계 및 인덱스
4. 구현: Spring Boot + MyBatis + MySQL 연동

프로젝트 구성 가이드

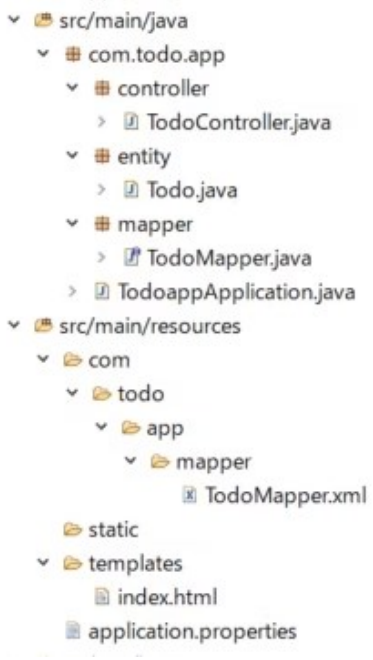


설치항목

항목	선택 여부
MySQL Driver	✅
Spring Web	✅
JDBC API or Spring Data JDBC	✅
MyBatis (수동 추가)	✅
Thymeleaf (템플릿 있을 경우)	⭕ 선택
Lombok	⭕ 선택

Dependencies

- spring-boot-starter-data-jdbc (managed:3.5.0)
- spring-boot-starter-web (managed:3.5.0)
- spring-boot-devtools [runtime] (managed:3.5.0)
- mysql-connector-j [runtime] (managed:9.2.0)
- mysql-connector-java : 8.0.33
- lombok (managed:1.18.38)
- spring-boot-starter-test [test] (managed:3.5.0)
- mybatis-spring-boot-starter : 3.0.3
- spring-boot-starter-thymeleaf (managed:3.5.0)



application.properties

```
spring.application.name=SpringWorkTodoApp07

#mysql 구현
spring.datasource.url=jdbc:mysql://localhost:3306/유명
spring.datasource.username=계정
spring.datasource.password=암호
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Mybatis Mapper XML 파일 위치 (필수!)
mybatis.mapper-locations=classpath:/com/todo/app/mapper/*.xml
```

MySQL 테이블 명세서: todo_items

컬럼명	자료형	NULL 허용	기본값	설명
id	INT	NOT NULL	AUTO_INCREMENT	기본키 (자동 증가)
title	VARCHAR(40)	YES	없음	할 일 제목
done_flg	TINYINT(1)	YES	0	완료 여부 (0: 미완료, 1: 완료)
time_limit	DATE	YES	없음	마감 기한

임의의 데이터 입력합니다

[문제 2] Entity를 만드시오

```
package com.todo.app.entity;
import java.sql.Date;
import lombok.Data;
@Data
public class Todo {
	private long id;
	private String title;
	private int done_flg;
	private String time_limit;
}
```

[문제 3] Mapper를 만드시오

```
package com.todo.app.mapper;
import java.util.List;
import org.apache.ibatis.annotations.Mapper;
import com.todo.app.entity.Todo;

@Mapper
public interface TodoMapper {
	public List<Todo> selectAll();
}
```

[문제 4] Xml 파일을 만드시오

TodoMapper.xml 파일이 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.todo.app.mapper.TodoMapper">
  <select id="selectAll" resultType="com.todo.app.entity.Todo">
    select * from todo_items
  </select>
</mapper>
```

[문제 5] Thymeleaf 구현합니다

TodoList라는 문자와 Todo 목록의 제목을 표시합니다.

```
<!DOCTYPE html>
<html  xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>TodoApp</title>
</head>
<body>
  <h1>TodoList</h1>
  <p th:each="todo : ${todos}" th:text="${todo.title}" />
</body>
</html>
```

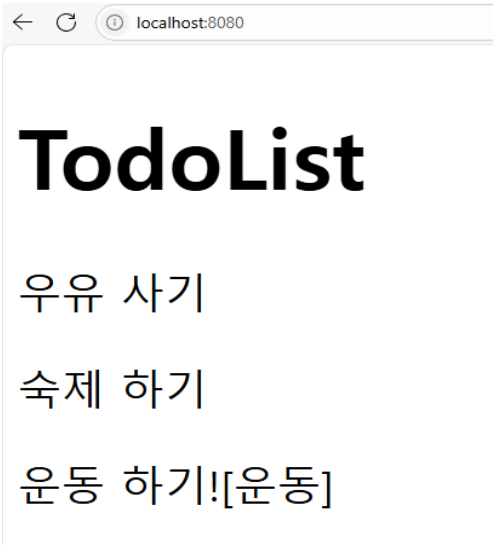
[문제 6] 컨트롤러 구현합니다

```
package com.todo.app.controller;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import com.todo.app.entity.Todo;
import com.todo.app.mapper.TodoMapper;

@Controller
public class TodoController {

	@Autowired
	TodoMapper todoMapper;

	@RequestMapping(value="/")
	public String index(Model model) {
		List<Todo> list = todoMapper.selectAll();
		model.addAttribute("todos",list);
		return "index";
	}
}
```

[문제 7] main에서 환경설정지정 후 실행 확인

```
@SpringBootApplication
@MapperScan("com.todo.app.mapper")
public class TodoappApplication {
    public static void main(String[] args) {
        SpringApplication.run(TodoappApplication.class, args);
    }
}
```

[실행결과]



[문제 8] 추가 기능 구현합니다

```
@Mapper
public interface TodoMapper {
        public List<Todo> selectAll();
        public void add(Todo todo); // 추가
}
```

```
<!-----매퍼  추가  ---- >
  <insert id="add" parameterType="com.todo.app.entity.Todo">
    insert into todo_items (title,time_limit)
    values (#{title}, STR_TO_DATE (#{time_limit},'yy-mm-dd'))
  </insert>
```

```
//Controller 추가
@RequestMapping(value="/add")
        public String add(Todo todo) {
                todoMapper.add(todo);
                return "redirect:/";
        }
```

```
//thymeleaf 추가
<h3>새로운 할 일 추가 (새로운 태스크를 추가)</h3>
  <form method="post" th:action="@{/add}">
    <input type="text" name="title" />
    <input type="date" name="time_limit"/>
    <input type="submit" value="확인" />
  </form>
```

[문제 9] 업데이트 기능 구현합니다

```
@Mapper
public interface TodoMapper {
        public void update(Todo todo);//추가}
}
```

```
<!-----매퍼  추가  ---- >
   <update id="update" parameterType="com.todo.app.entity.Todo">
    update todo_items set
      title = #{title},
      time_limit = #{time_limit},
      done_flg = #{done_flg}
      where id = #{id}
  </update>
```

```
//Controller 추가
@RequestMapping(value="/update")
        public String update(Todo todo) {
                todoMapper.update(todo);
                return "redirect:/";
        }
```

//**thymeleaf 추가**

```
<h1>할 일 목록 (TodoList)</h1>
  <h3>내 할 일 (마이 태스크)</h3>
  <form method="post" th:action="@{/update}" th:each="todo :
${todos}" >
    <input type="checkbox" name="done_flg" value="1"/>
    <input type="hidden" name="id" th:value="${todo.id}" />
    <input type="text" name="title" th:value="${todo.title}" />
    <input type="date" name="time_limit"
th:value="${todo.time_limit}" />
    <input type="submit" value="업데이트" />
  </form>
```

실행후 데이터 한줄 입력해 본다. 아직 화면에 변화가 나타나지 않으므로 확인하고 싶다면 DB 확인해 본다

[문제 10] 미완료 표시를 해본다.

```
@Mapper
public interface TodoMapper {        ....
        public List<Todo> selectIncomplete();  //추가
        }
```

```
<!----- 매퍼 추가 --- >
<!-- 미완료 항목 조회 -->
  <select id="selectIncomplete" resultType="com.todo.app.entity.Todo">
    SELECT * FROM todo_items WHERE done_flg = 0
  </select
```

//**Controller 를 편집한다.** selectAll 을 주석해 둔다.

```
@RequestMapping(value="/")
      public String index(Model model) {
//          List<Todo> list = todoMapper.selectAll();
            List<Todo> list = todoMapper.selectIncomplete();
            model.addAttribute("todos",list);
            return "index";
      }
```

브라우저를 시작하면 완료되지 않은 작업 만 표시확인

[문제 11] 완료도 표시한다.

```
@Mapper
public interface TodoMapper {
        public List<Todo> selectAll();
        public List<Todo> selectIncomplete();
        public List<Todo> selectComplete();
        public void add(Todo todo);
        public void update(Todo todo);
}
```

```
<!-----매퍼 추가 ---- >
<!-- 완료 항목 조회 -->
  <select id="selectComplete" resultType="com.todo.app.entity.Todo">
    SELECT * FROM todo_items WHERE done_flg = 1
  </select>
```

```
//Controller 편집
@RequestMapping(value="/")
        public String index(Model model) {
          //  List<Todo> list = todoMapper.selectAll();
                List<Todo> list = todoMapper.selectIncomplete();
                List<Todo> doneList = todoMapper.selectComplete();
                model.addAttribute("todos",list);
                model.addAttribute("doneTodos",doneList);
                return "index";
        }
```

//thymeleaf 편집

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>TodoApp</title>
</head>
<body>
  <h1>할 일 목록 (TodoList)</h1>
  <h3>내 할 일 (마이 태스크)</h3>
<form method="post" th:action="@{/update}" th:each="todo : ${todos}"
>
```

```
    <input type="checkbox" name="done_flg" value="1"/>
    <input type="hidden" name="id" th:value="${todo.id}" />
    <input type="text" name="title" th:value="${todo.title}" />
    <input type="date" name="time_limit"
th:value="${todo.time_limit}" />
    <input type="submit" value="업데이트" />
  </form>

  <h3>완료됨 (완료된 할 일)</h3>
  <form method="post" th:action="@{/update}" th:each="todo :
${doneTodos}" >
    <input type="checkbox" name="done_flg" value="1"/>
    <input type="hidden" name="id" th:value="${todo.id}" />
    <input type="text" name="title" th:value="${todo.title}"
style="text-decoration:line-through"/>
    <input type="date" name="time_limit"
th:value="${todo.time_limit}" />
    <input type="submit" value="업데이트" />
  </form>

  <h3>새로운 할 일 추가 (새로운 태스크를 추가)</h3>
  <form method="post" th:action="@{/add}">
    <input type="text" name="title" />
    <input type="date" name="time_limit"/>
    <input type="submit" value="확인" />
  </form>
</body>
</html>
```

완료와 미완료 나눈화면과 결과 확인_DB 에서 입력후 확인

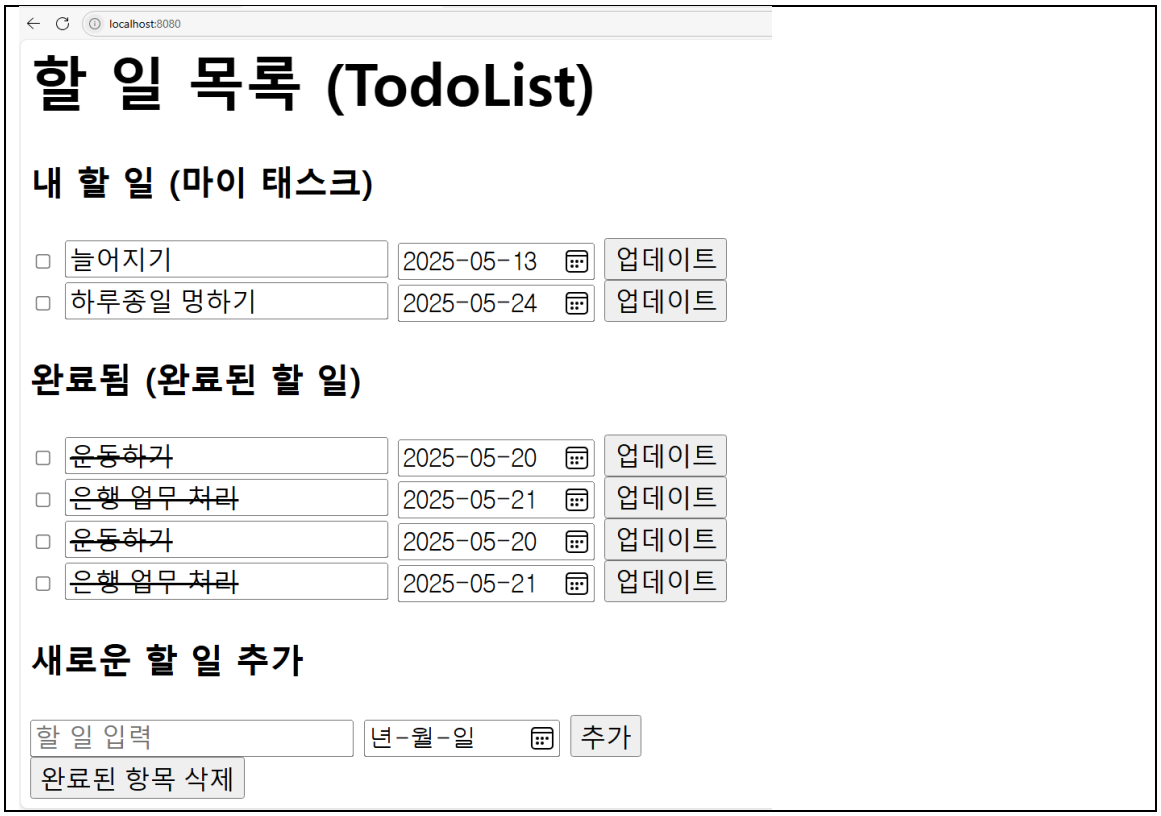
```
-- 미완료 항목
INSERT INTO todo_items (title, done_flg, time_limit) VALUES
('스터디 준비하기', 0, '2025-05-30'),
('이메일 확인하기', 0, '2025-05-24'),
('쇼핑 리스트 작성', 0, '2025-05-27');
-- 완료 항목
INSERT INTO todo_items (title, done_flg, time_limit) VALUES
('운동하기', 1, '2025-05-20'),
('은행 업무 처리', 1, '2025-05-21');
commit;
```

[문제 12] 체크하게 되면 완료 항목을 전달되게 구현한다.

```
<!-- 미완료 영역 -->
        <h3>내 할 일 (마이 태스크)</h3>
        <form method="post" th:action="@{/update}" th:each="todo : ${todos}">
          <!-- 1. 체크 안 할 경우에도 기본값 0 이 전달되도록 hidden 추가 -->
          <input type="hidden" name="done_flg" value="0"/>

          <!-- 2. 체크하면 이 값이 덮어쓰기됨 → 1 로 전송 -->
          <input type="checkbox" name="done_flg" value="1"
th:checked="${todo.doneFlg}"/>

          <input type="hidden" name="id" th:value="${todo.id}" />
          <input type="text" name="title" th:value="${todo.title}" />
          <input type="date" name="time_limit" th:value="${todo.time_limit}"
required/>
          <input type="submit" value="업데이트" />
        </form>
```

```
<!-- 완료 영역 -->
      <h3>완료됨 (완료된 할 일)</h3>
      <form method="post" th:action="@{/update}" th:each="todo :
${doneTodos}">
        <!-- 동일하게 hidden 추가 -->
        <input type="hidden" name="done_flg" value="0"/>
        <input type="checkbox" name="done_flg" value="1"
th:checked="${todo.doneFlg}"/>

        <input type="hidden" name="id" th:value="${todo.id}" />
        <input type="text" name="title" th:value="${todo.title}"
style="text-decoration:line-through"/>
        <input type="date" name="time_limit"
th:value="${todo.time_limit}" required/>
        <input type="submit" value="업데이트" />
      </form>
```

[문제 13] 삭제 버튼을 만들어 체크한 내용을 삭제하는 코드를 추가 후 실행한다.



☑ 실행 결과 확인 체크리스트 (할 일 → 완료 이동 정상 여부)

☑ 1. 초기 화면 출력

- localhost:8080 접속 시 **"할 일 목록"**이 보여야 한다
- 미완료 항목은 위쪽(todos)
- 완료된 항목은 아래쪽(doneTodos)에 "완료됨" 제목과 함께 표시된다
- 완료된 항목은 title 에 취소선(line-through)이 적용된다

☑ 2. 업데이트 기능 (미완료 → 완료로 이동)

- 미완료 항목에서 **체크박스를 클릭**한 뒤, "업데이트" 버튼을 누른다
- 해당 항목이 화면에서 **위 → 아래(완료됨 영역)** 으로 옮겨간다
- 서버 콘솔에 ✅ 업데이트 요청: Todo{...} 로그가 찍히며, doneFlg=true 로 나온다
- DB 조회 시 해당 done_flg 값이 1 로 바뀌어 있다

☑ 3. 업데이트 기능 (완료 → 다시 미완료로 이동)

- 완료 영역에서 **체크박스를 해제한 후**, "업데이트" 버튼을 누른다
- 항목이 다시 위쪽 미완료 목록으로 이동한다
- DB 의 done_flg=0 으로 변경됨을 확인한다

☑ 4. 추가 기능 확인

- 아래쪽 "새로운 할 일 추가" 폼에 값을 입력 후 "확인" 클릭 시
- 새로운 항목이 **미완료 영역(todos)** 에 표시된다

☑ 5. 완료 항목 일괄 삭제 확인

- "완료된 항목 삭제" 버튼을 누르면
- 완료 영역 항목들이 화면에서 제거되고

- DB 에서도 done_flg=1 인 항목들이 삭제된다.