프로그래밍 기초 및 응용

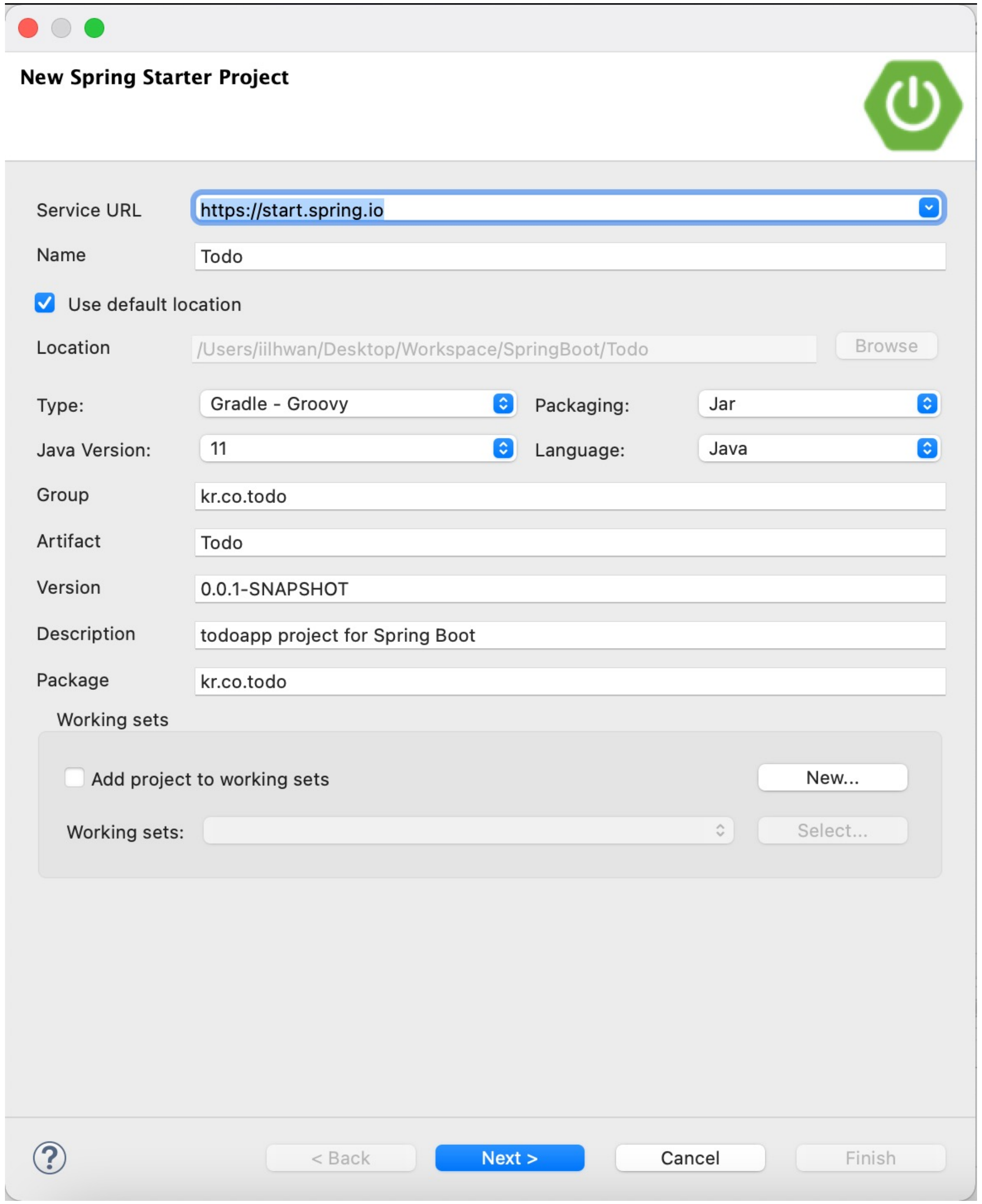
통합 구현

이원정

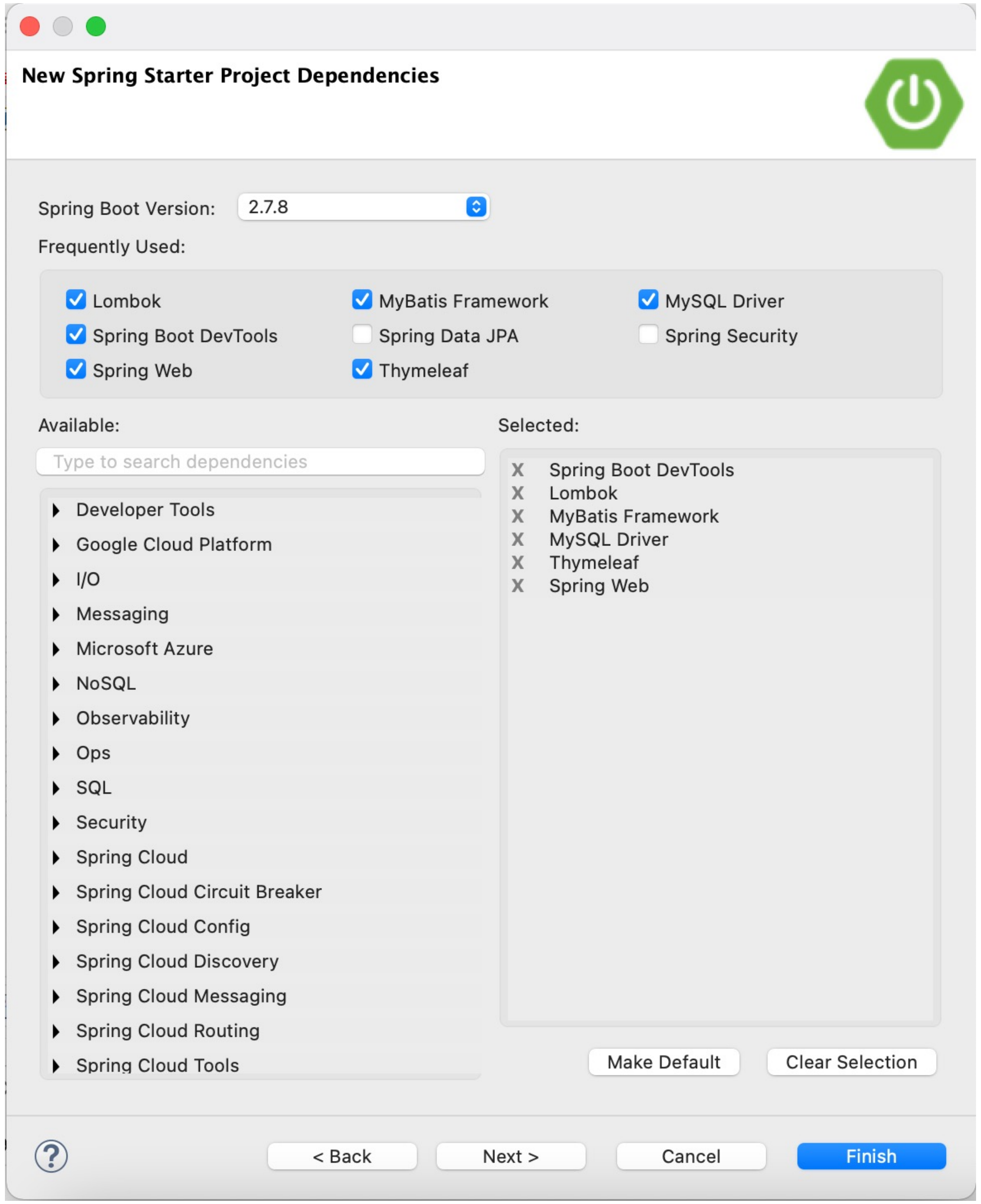
Contents

1. 프로그램 생성

프로젝트 개요	할 일 관리를 위한 애플리케이션
프로젝트 이름	Todo
Framework	Spring Boot 2.7.8
Build	Maven
Group	kr.co.todo
Artifact	Todo
Package	kr.co.todo
Packaging	Jar
Langauage	Java 11
Dependencies	Spring Boot DevTools, Lombok, Spring web, Thymeleaft, Mybatis, MySQL


New Spring Starter Project

Service URL: https://start.spring.io
Name: Todo
[✓] Use default location
Location: /Users/iilhwan/Desktop/Workspace/SpringBoot/Todo
Type: Gradle - Groovy
Packaging: Jar
Java Version: 11
Language: Java
Group: kr.co.todo
Artifact: Todo
Version: 0.0.1-SNAPSHOT
Description: todoapp project for Spring Boot
Package: kr.co.todo
Working sets
[ ] Add project to working sets
Working sets:


New Spring Starter Project Dependencies

Spring Boot Version: 2.7.8
Frequently Used:

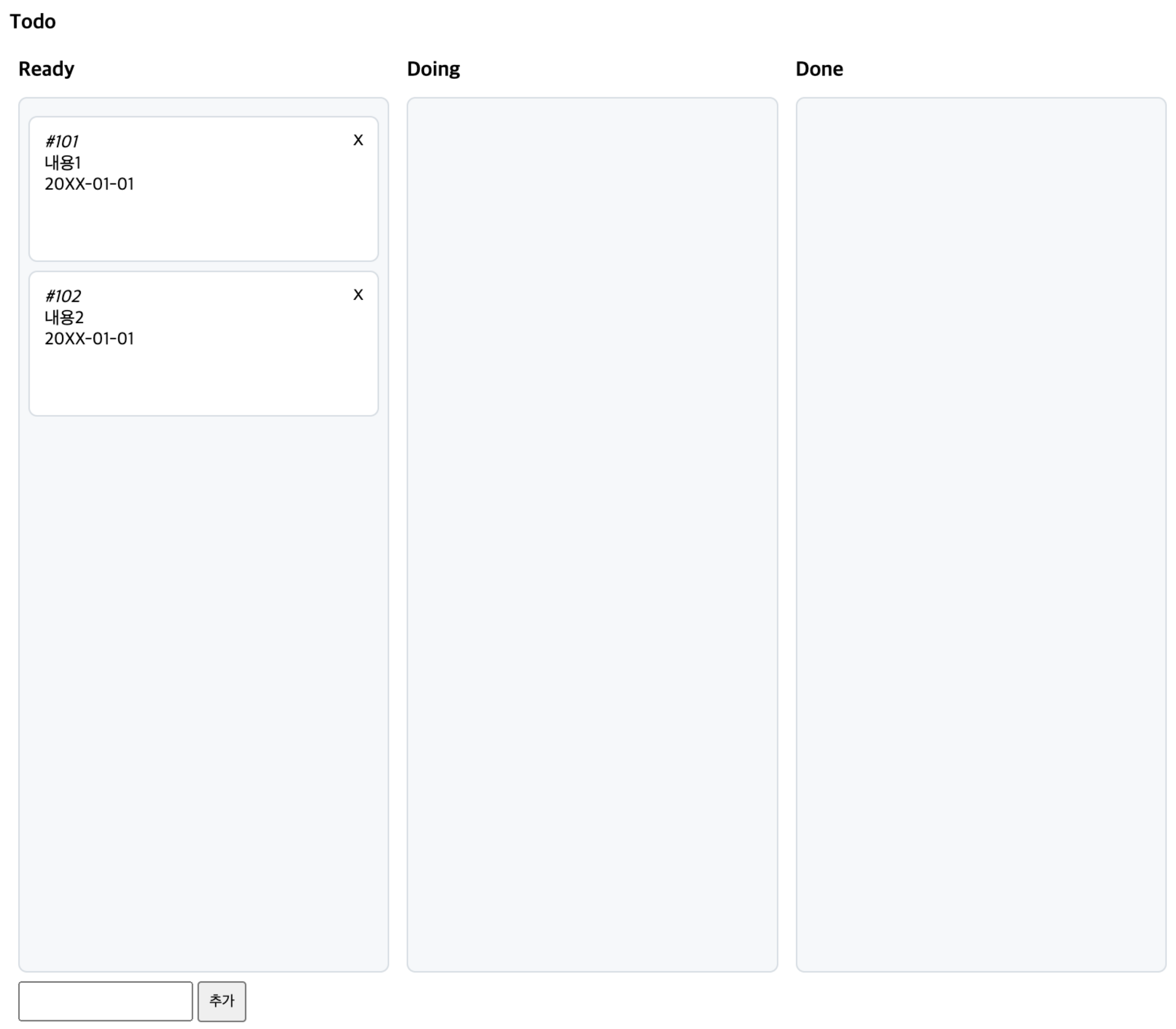
[✓] Lombok	[✓] MyBatis Framework	[✓] MySQL Driver
[✓] Spring Boot DevTools	[ ] Spring Data JPA	[ ] Spring Security
[✓] Spring Web	[✓] Thymeleaf	

Available:
Type to search dependencies
- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging
- Spring Cloud Routing
- Spring Cloud Tools

Selected:
- X Spring Boot DevTools
- X Lombok
- X MyBatis Framework
- X MySQL Driver
- X Thymeleaf
- X Spring Web

2. 화면 구현

(1) 화면 레이아웃 작업


Todo

Ready

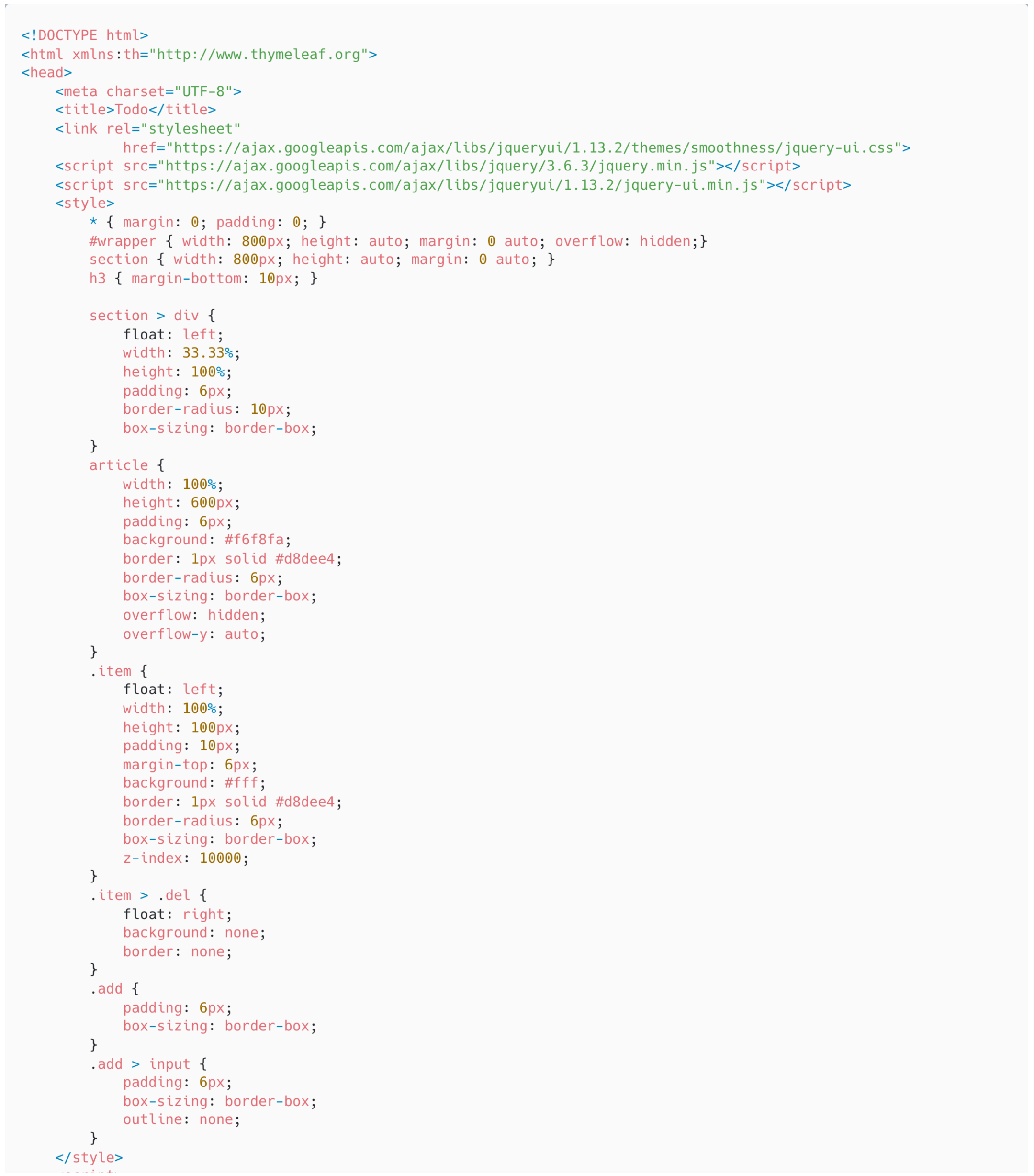
#101
내용1
20XX-01-01

#102
내용2
20XX-01-01

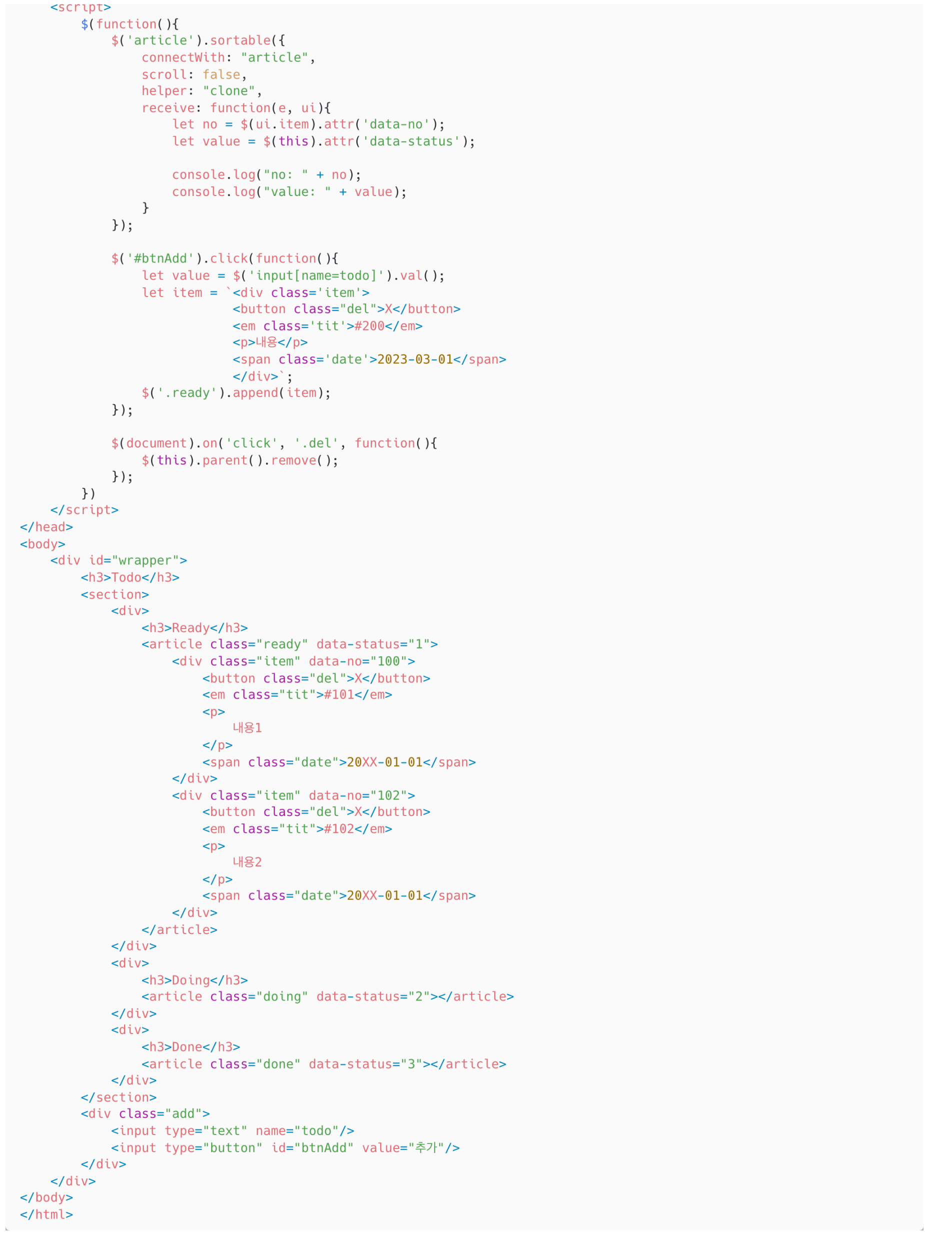
Doing

Done

2. 화면 구현



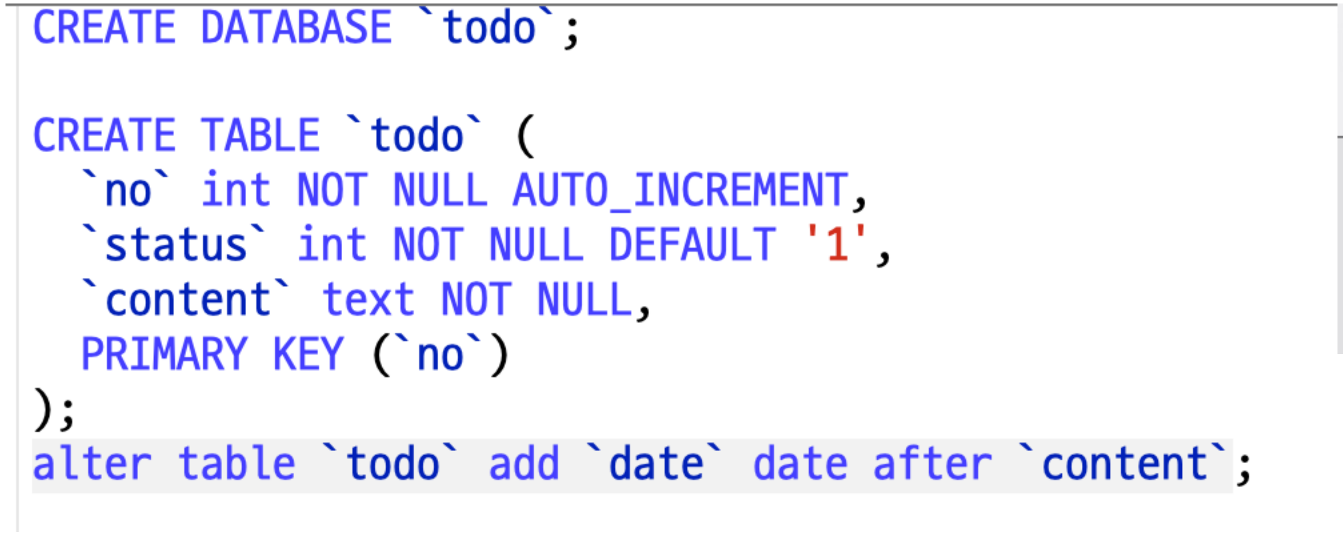
```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Todo</title>
    <link rel="stylesheet"
            href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.13.2/themes/smoothness/jquery-ui.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.13.2/jquery-ui.min.js"></script>
    <style>
        * { margin: 0; padding: 0; }
        #wrapper { width: 800px; height: auto; margin: 0 auto; overflow: hidden;}
        section { width: 800px; height: auto; margin: 0 auto; }
        h3 { margin-bottom: 10px; }

        section > div {
            float: left;
            width: 33.33%;
            height: 100%;
            padding: 6px;
            border-radius: 10px;
            box-sizing: border-box;
        }
        article {
            width: 100%;
            height: 600px;
            padding: 6px;
            background: #f6f8fa;
            border: 1px solid #d8dee4;
            border-radius: 6px;
            box-sizing: border-box;
            overflow: hidden;
            overflow-y: auto;
        }
        .item {
            float: left;
            width: 100%;
            height: 100px;
            padding: 10px;
            margin-top: 6px;
            background: #fff;
            border: 1px solid #d8dee4;
            border-radius: 6px;
            box-sizing: border-box;
            z-index: 10000;
        }
        .item > .del {
            float: right;
            background: none;
            border: none;
        }
        .add {
            padding: 6px;
            box-sizing: border-box;
        }
        .add > input {
            padding: 6px;
            box-sizing: border-box;
            outline: none;
        }
    </style>
```



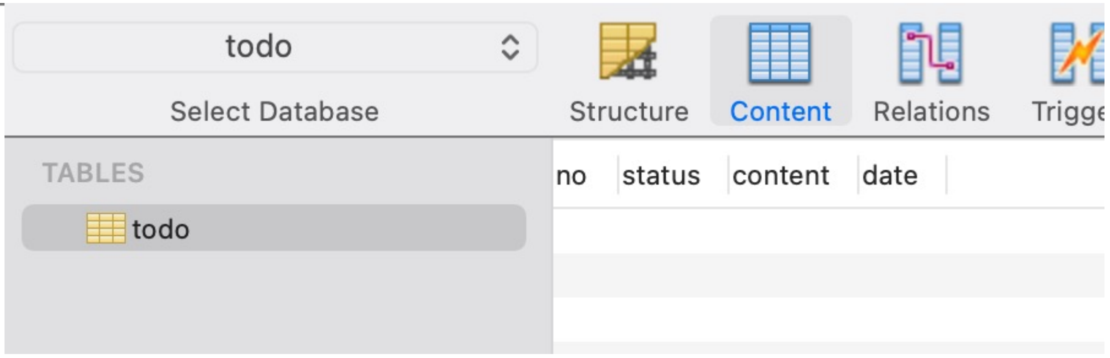
```html
    <script>
        $(function(){
            $('article').sortable({
                connectWith: "article",
                scroll: false,
                helper: "clone",
                receive: function(e, ui){
                    let no = $(ui.item).attr('data-no');
                    let value = $(this).attr('data-status');

                    console.log("no: " + no);
                    console.log("value: " + value);
                }
            });

            $('#btnAdd').click(function(){
                let value = $('input[name=todo]').val();
                let item = `<div class='item'>
                                <button class="del">X</button>
                                <em class='tit'>#200</em>
                                <p>내용</p>
                                <span class='date'>2023-03-01</span>
                                </div>`;
                $('.ready').append(item);
            });

            $(document).on('click', '.del', function(){
                $(this).parent().remove();
            });
        })
    </script>
</head>
<body>
    <div id="wrapper">
        <h3>Todo</h3>
        <section>
            <div>
                <h3>Ready</h3>
                <article class="ready" data-status="1">
                    <div class="item" data-no="100">
                        <button class="del">X</button>
                        <em class="tit">#101</em>
                        <p>
                            내용1
                        </p>
                        <span class="date">20XX-01-01</span>
                    </div>
                    <div class="item" data-no="102">
                        <button class="del">X</button>
                        <em class="tit">#102</em>
                        <p>
                            내용2
                        </p>
                        <span class="date">20XX-01-01</span>
                    </div>
                </article>
            </div>
            <div>
                <h3>Doing</h3>
                <article class="doing" data-status="2"></article>
            </div>
            <div>
                <h3>Done</h3>
                <article class="done" data-status="3"></article>
            </div>
        </section>
        <div class="add">
            <input type="text" name="todo"/>
            <input type="button" id="btnAdd" value="추가"/>
        </div>
    </div>
</body>
</html>
```

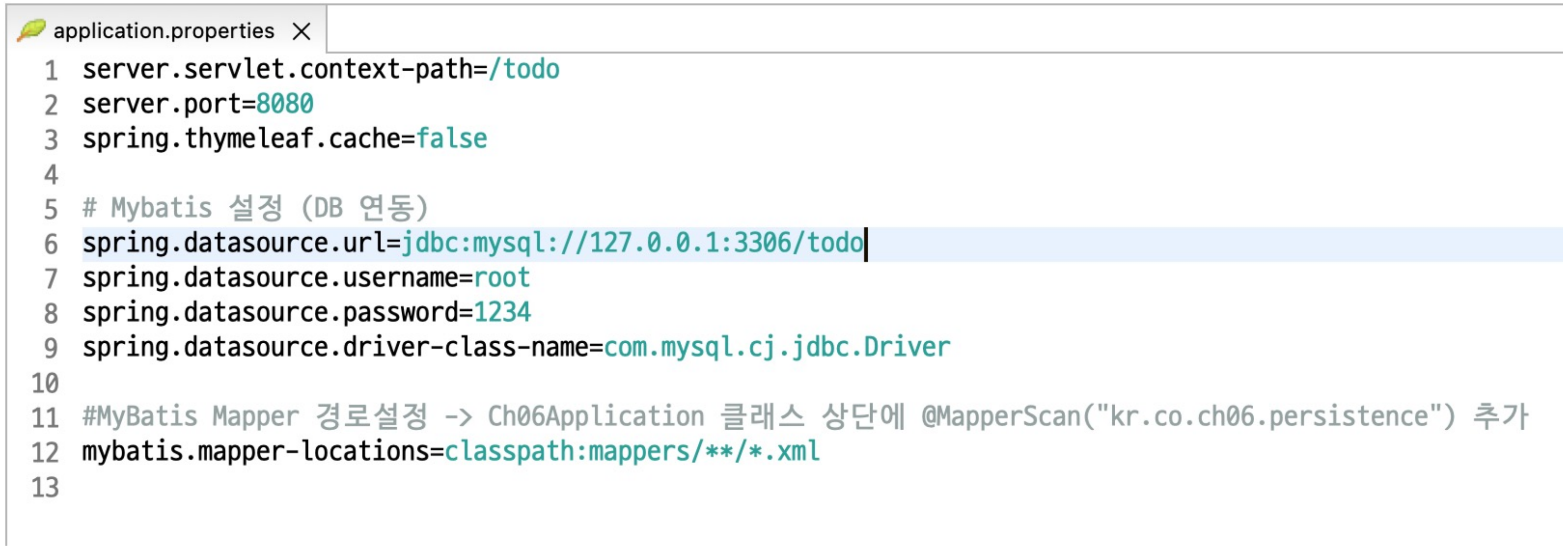
3. 테이블 설계

(1) 리스트별 데이터 저장을 위한 테이블 설계 (테이블명: todo)



```sql
CREATE DATABASE `todo`;

CREATE TABLE `todo` (
  `no` int NOT NULL AUTO_INCREMENT,
  `status` int NOT NULL DEFAULT '1',
  `content` text NOT NULL,
  PRIMARY KEY (`no`)
);
alter table `todo` add `date` date after `content`;
```

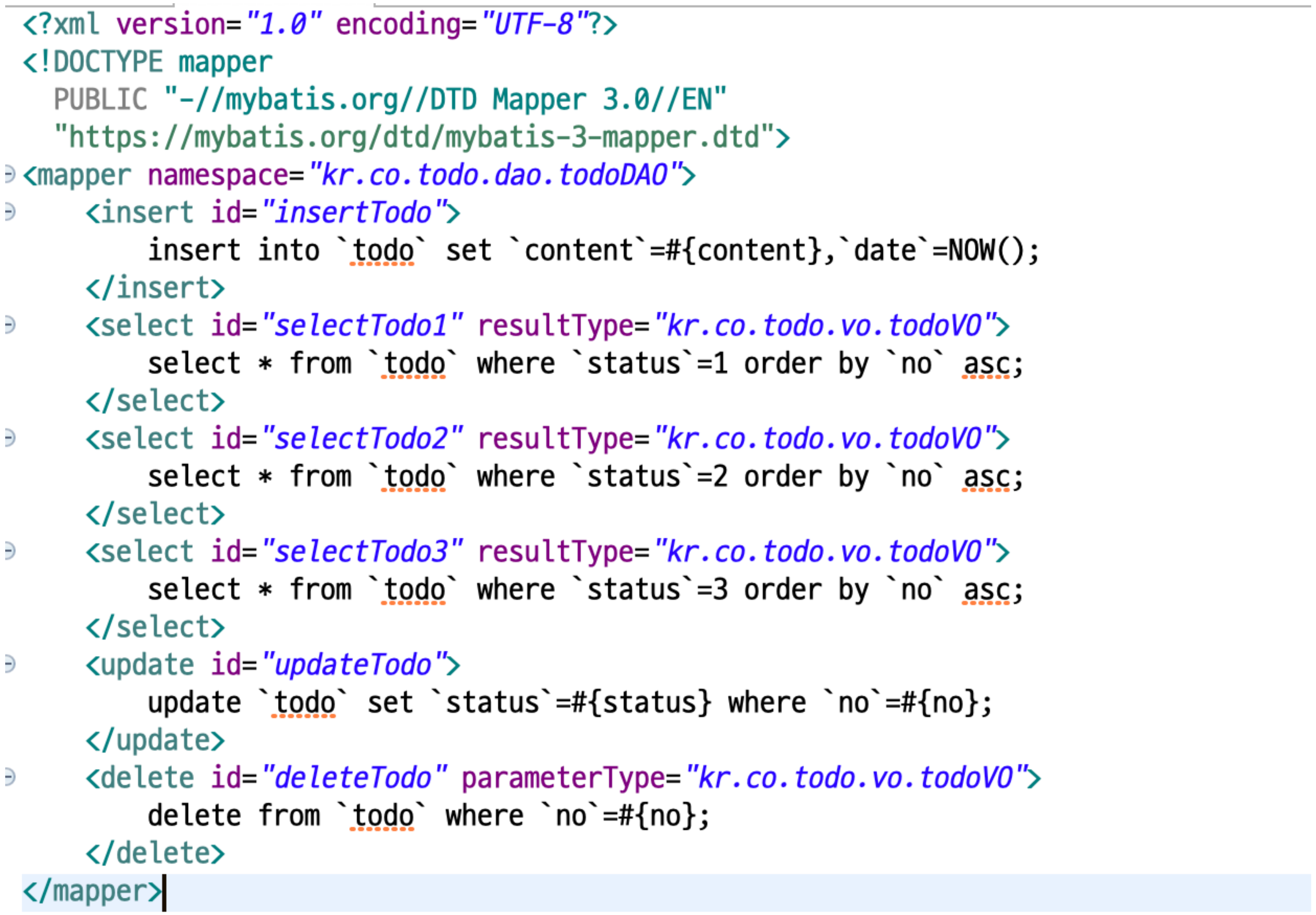




```properties
server.servlet.context-path=/todo
server.port=8080
spring.thymeleaf.cache=false

# Mybatis 설정 (DB 연동)
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/todo
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#MyBatis Mapper 경로설정 -> Ch06Application 클래스 상단에 @MapperScan("kr.co.ch06.persistence") 추가
mybatis.mapper-locations=classpath:mappers/**/*.xml

```

4. 기능 구현

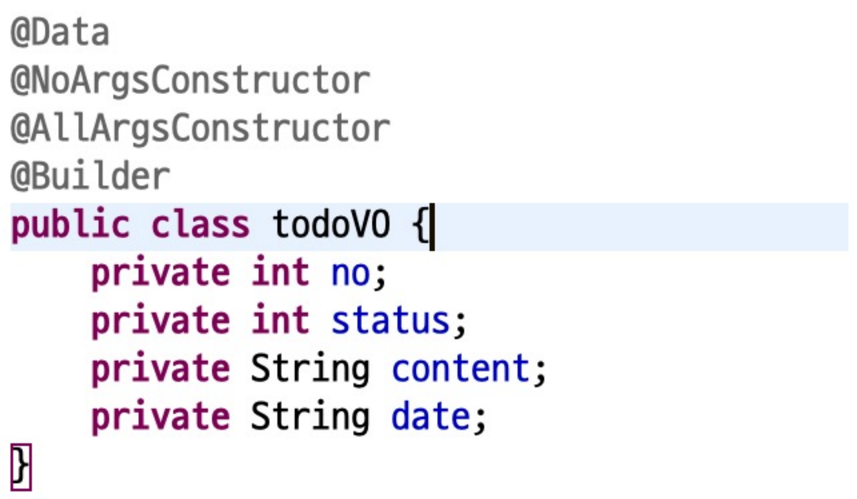
(1) Mapper



```
<?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="kr.co.todo.dao.todoDAO">
    <insert id="insertTodo">
        insert into `todo` set `content`=#{content},`date`=NOW();
    </insert>
    <select id="selectTodo1" resultType="kr.co.todo.vo.todoVO">
        select * from `todo` where `status`=1 order by `no` asc;
    </select>
    <select id="selectTodo2" resultType="kr.co.todo.vo.todoVO">
        select * from `todo` where `status`=2 order by `no` asc;
    </select>
    <select id="selectTodo3" resultType="kr.co.todo.vo.todoVO">
        select * from `todo` where `status`=3 order by `no` asc;
    </select>
    <update id="updateTodo">
        update `todo` set `status`=#{status} where `no`=#{no};
    </update>
    <delete id="deleteTodo" parameterType="kr.co.todo.vo.todoVO">
        delete from `todo` where `no`=#{no};
    </delete>
</mapper>
```

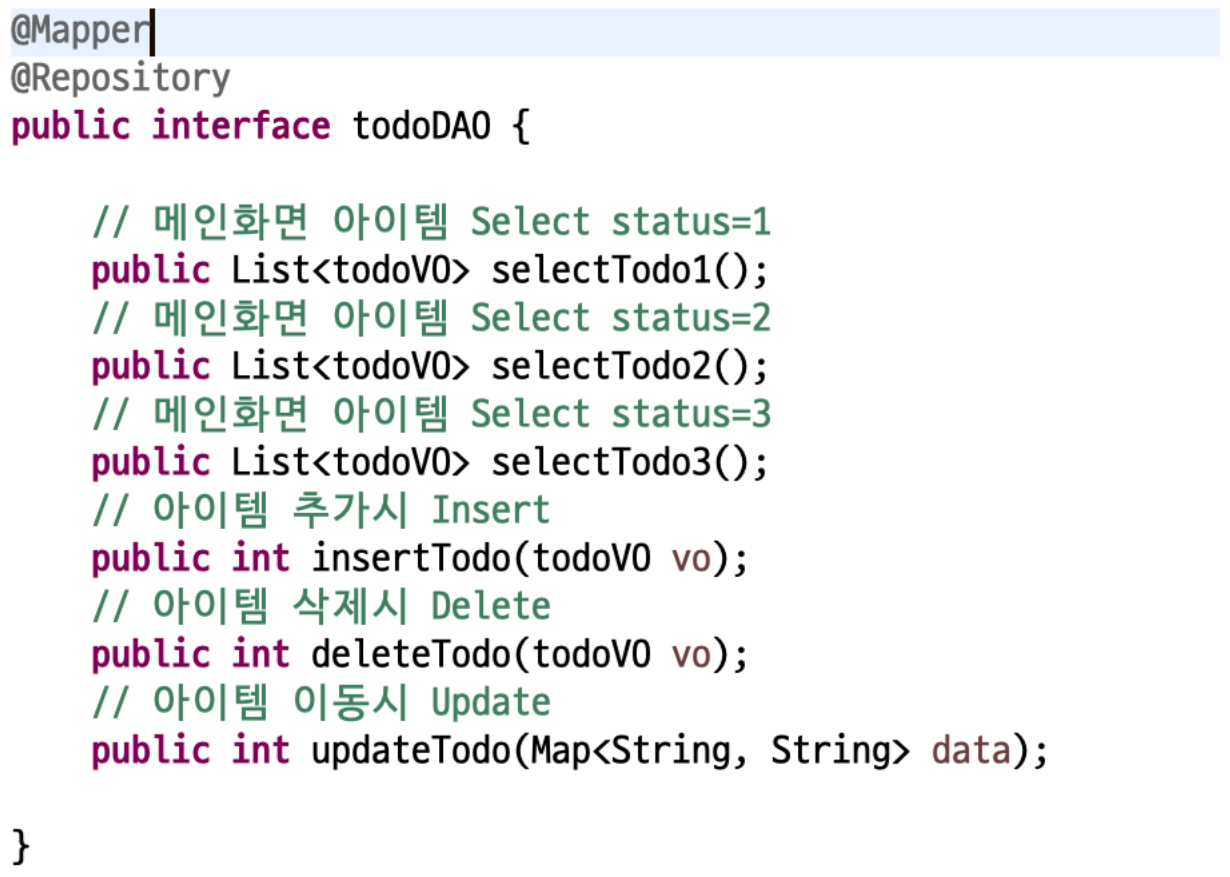
4. 기능 구현

(2) VO



```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class todoVO {
    private int no;
    private int status;
    private String content;
    private String date;
}
```

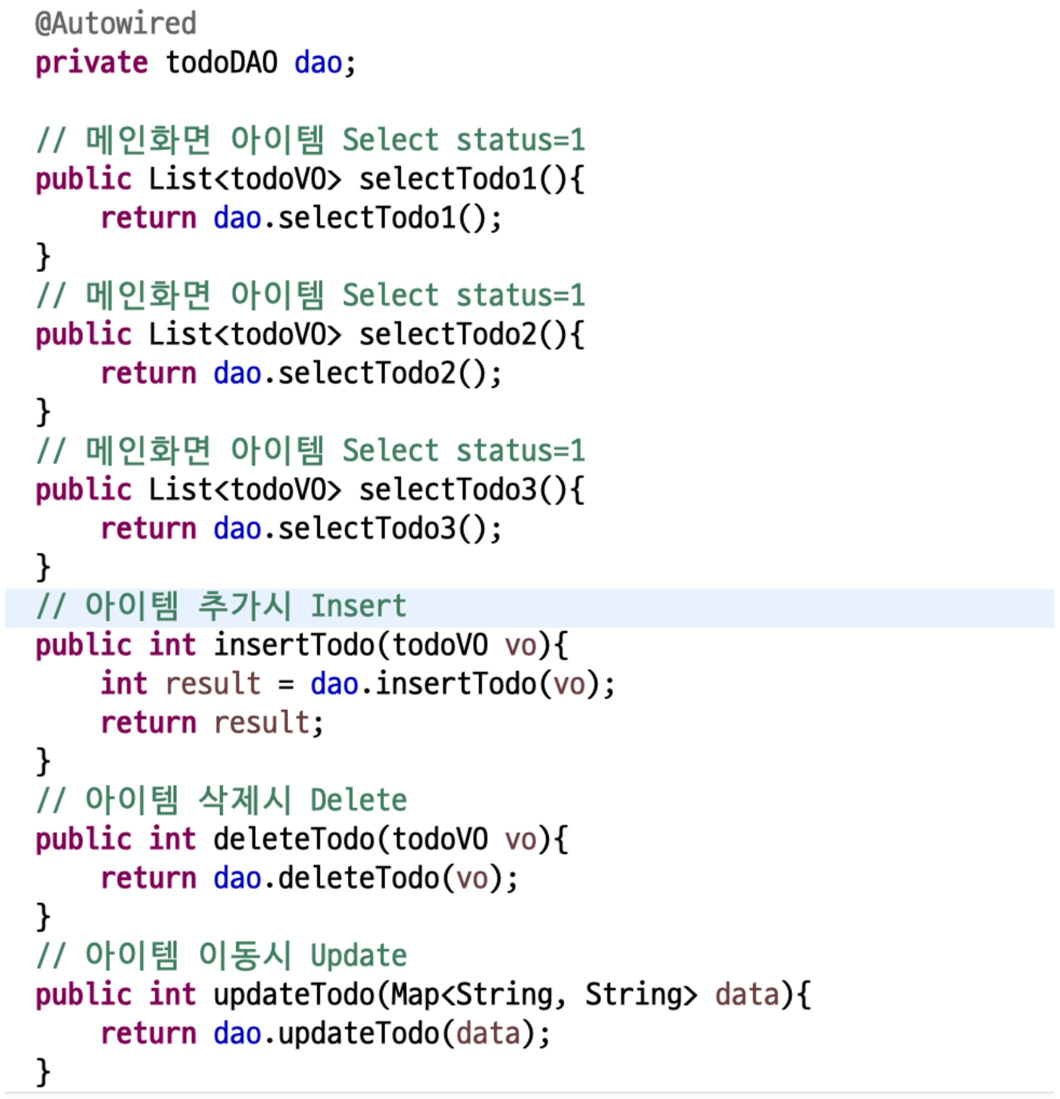
(3) DAO



```java
@Mapper
@Repository
public interface todoDAO {

    // 메인화면 아이템 Select status=1
    public List<todoVO> selectTodo1();
    // 메인화면 아이템 Select status=2
    public List<todoVO> selectTodo2();
    // 메인화면 아이템 Select status=3
    public List<todoVO> selectTodo3();
    // 아이템 추가시 Insert
    public int insertTodo(todoVO vo);
    // 아이템 삭제시 Delete
    public int deleteTodo(todoVO vo);
    // 아이템 이동시 Update
    public int updateTodo(Map<String, String> data);

}
```

4. 기능 구현

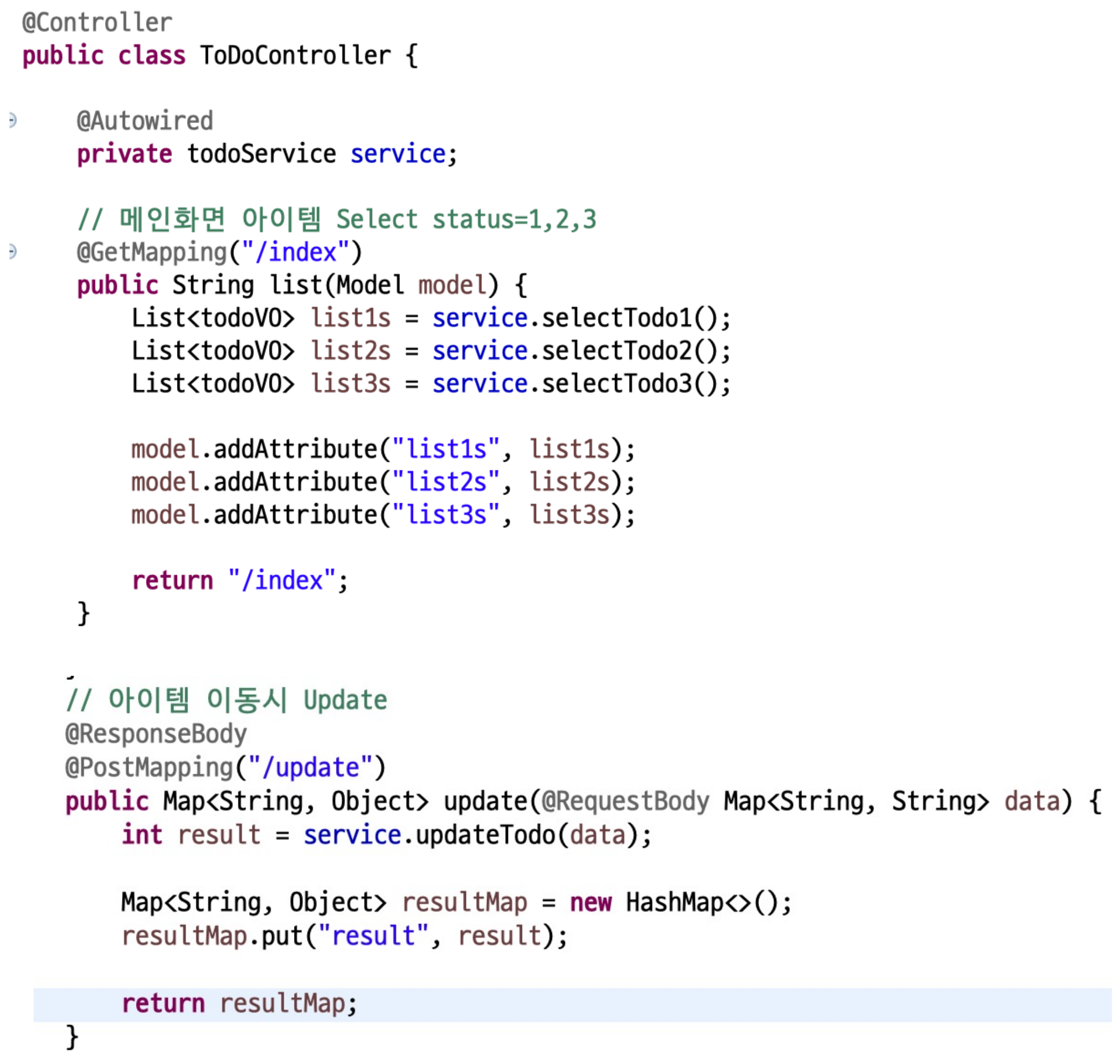
(2) Service



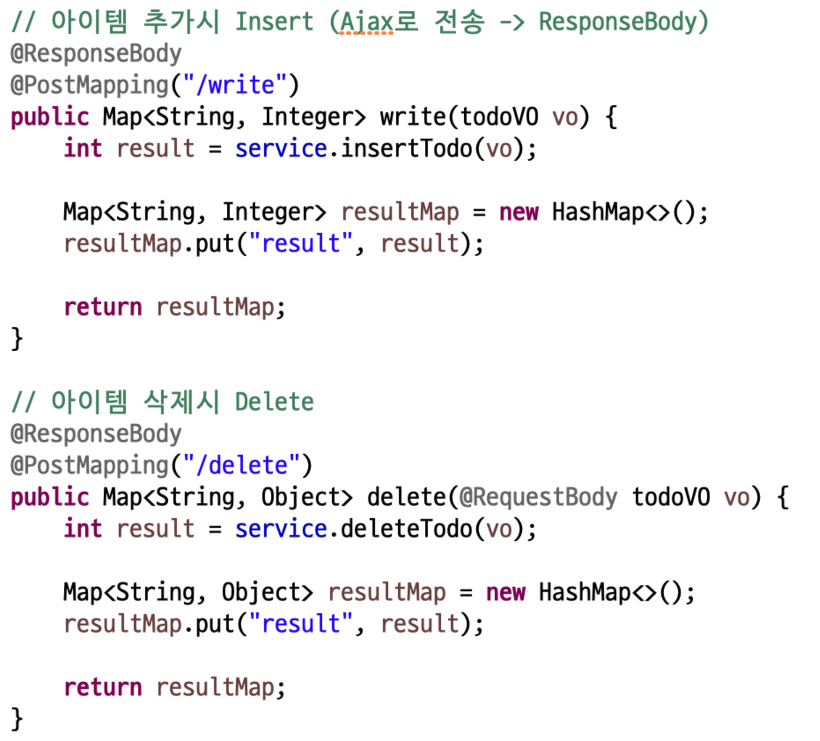
```java
@Autowired
private todoDAO dao;

// 메인화면 아이템 Select status=1
public List<todoVO> selectTodo1(){
    return dao.selectTodo1();
}
// 메인화면 아이템 Select status=1
public List<todoVO> selectTodo2(){
    return dao.selectTodo2();
}
// 메인화면 아이템 Select status=1
public List<todoVO> selectTodo3(){
    return dao.selectTodo3();
}
// 아이템 추가시 Insert
public int insertTodo(todoVO vo){
    int result = dao.insertTodo(vo);
    return result;
}
// 아이템 삭제시 Delete
public int deleteTodo(todoVO vo){
    return dao.deleteTodo(vo);
}
// 아이템 이동시 Update
public int updateTodo(Map<String, String> data){
    return dao.updateTodo(data);
}
```

4. 기능 구현

(3) Controller



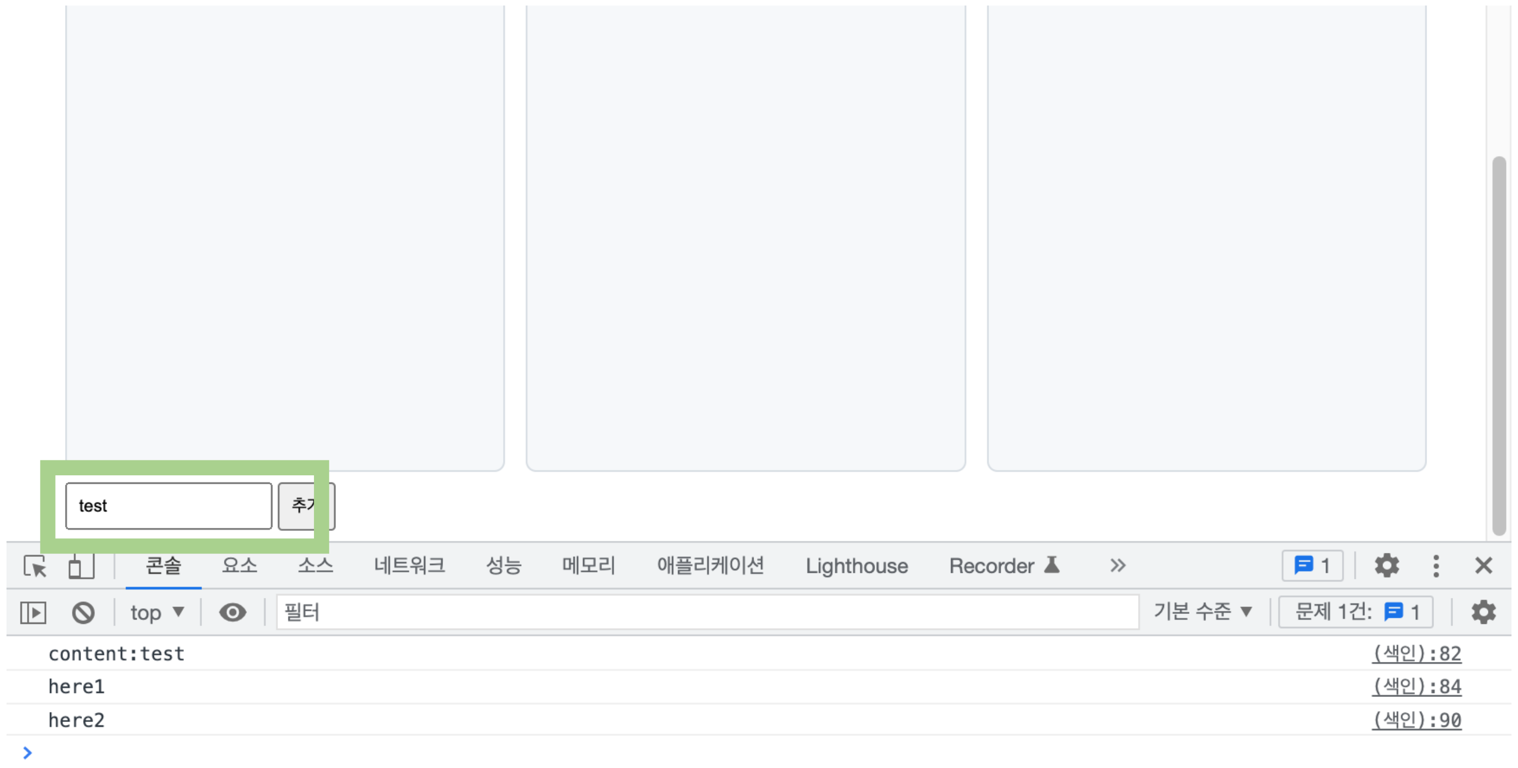
```java
@Controller
public class ToDoController {

    @Autowired
    private todoService service;

    // 메인화면 아이템 Select status=1,2,3
    @GetMapping("/index")
    public String list(Model model) {
        List<todoVO> list1s = service.selectTodo1();
        List<todoVO> list2s = service.selectTodo2();
        List<todoVO> list3s = service.selectTodo3();

        model.addAttribute("list1s", list1s);
        model.addAttribute("list2s", list2s);
        model.addAttribute("list3s", list3s);

        return "/index";
    }

   // 아이템 이동시 Update
   @ResponseBody
   @PostMapping("/update")
   public Map<String, Object> update(@RequestBody Map<String, String> data) {
       int result = service.updateTodo(data);

       Map<String, Object> resultMap = new HashMap<>();
       resultMap.put("result", result);

       return resultMap;
   }
```



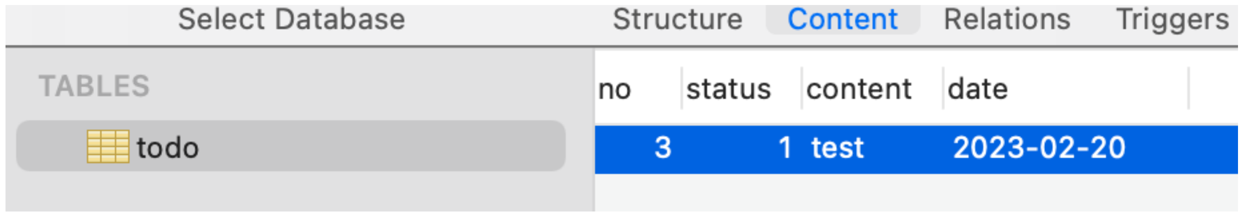
```java
// 아이템 추가시 Insert (Ajax로 전송 -> ResponseBody)
@ResponseBody
@PostMapping("/write")
public Map<String, Integer> write(todoVO vo) {
    int result = service.insertTodo(vo);

    Map<String, Integer> resultMap = new HashMap<>();
    resultMap.put("result", result);

    return resultMap;
}

// 아이템 삭제시 Delete
@ResponseBody
@PostMapping("/delete")
public Map<String, Object> delete(@RequestBody todoVO vo) {
    int result = service.deleteTodo(vo);

    Map<String, Object> resultMap = new HashMap<>();
    resultMap.put("result", result);

    return resultMap;
}
```

4. 실행

(1) Insert



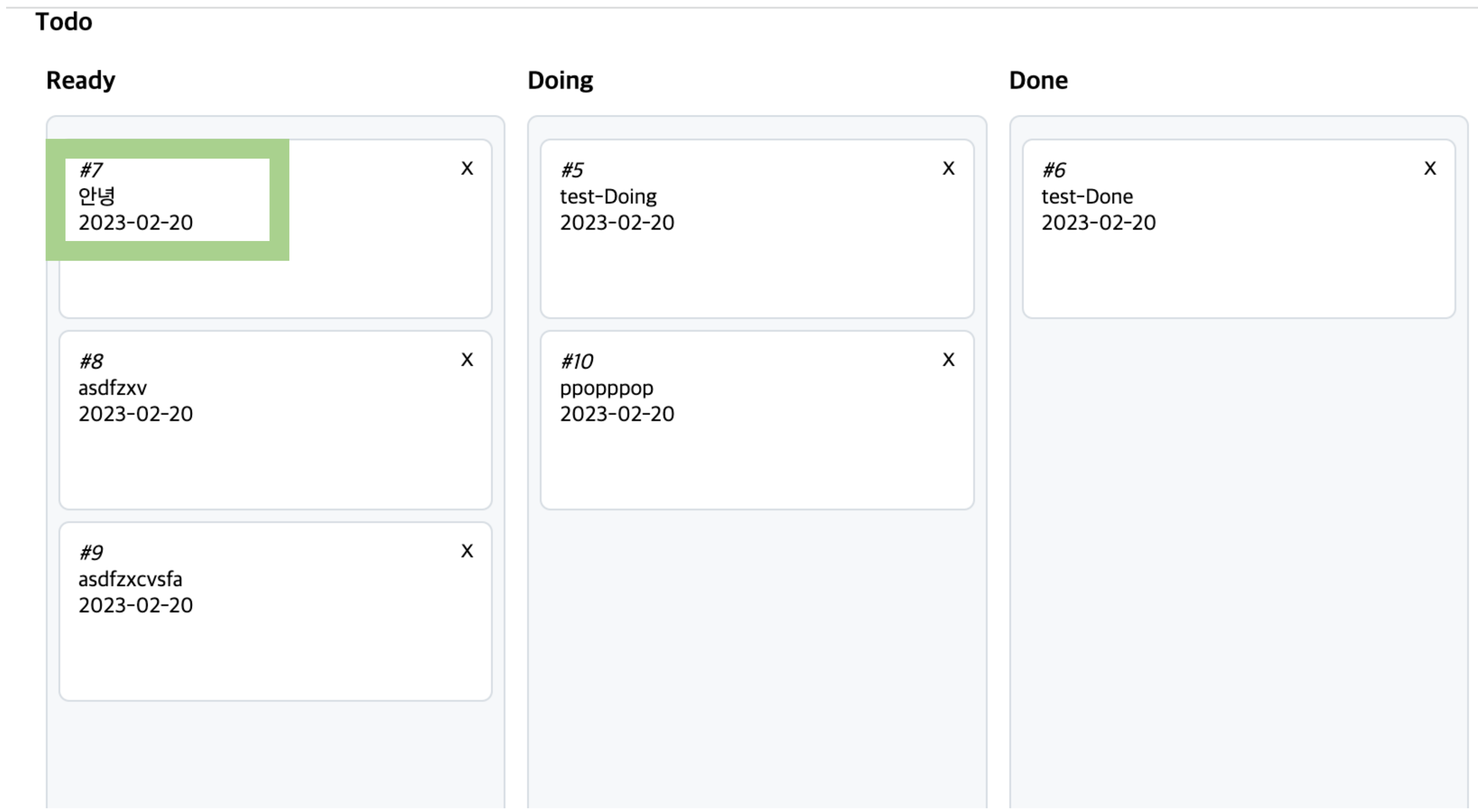
```
content:test        (색인):82
here1               (색인):84
here2               (색인):90
```



no	status	content	date
3	1	test	2023-02-20

4. 실행

(2) Delete



Todo

Ready

#7
안녕
2023-02-20

#8
asdfzxv
2023-02-20

#9
asdfzxcvsfa
2023-02-20

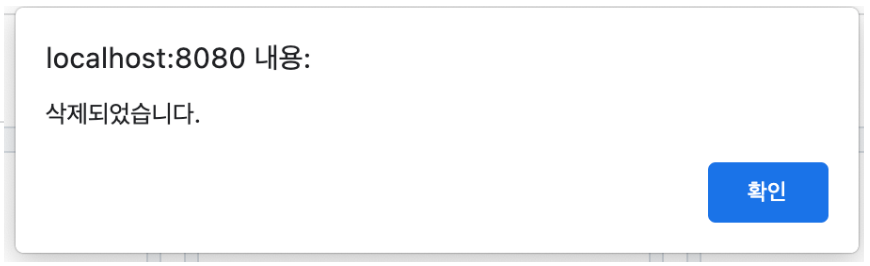
Doing

#5
test-Doing
2023-02-20

#10
ppopppop
2023-02-20

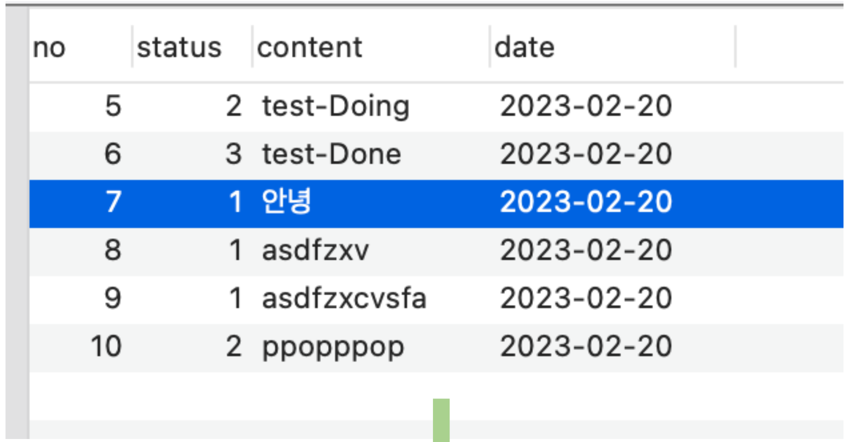
Done

#6
test-Done
2023-02-20

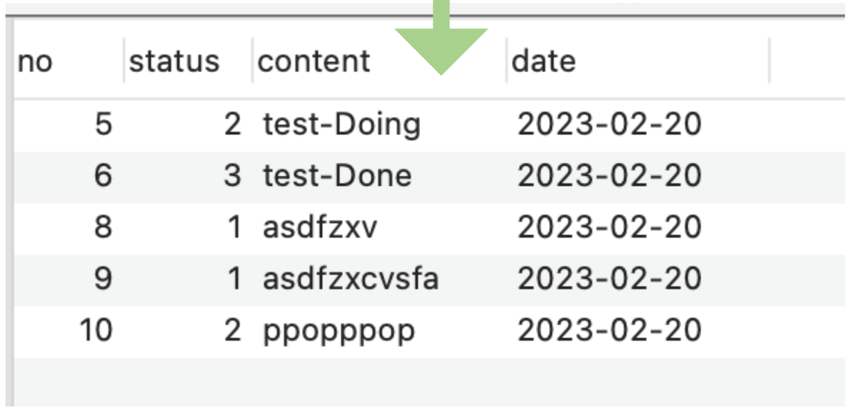


localhost:8080 내용:

삭제되었습니다.



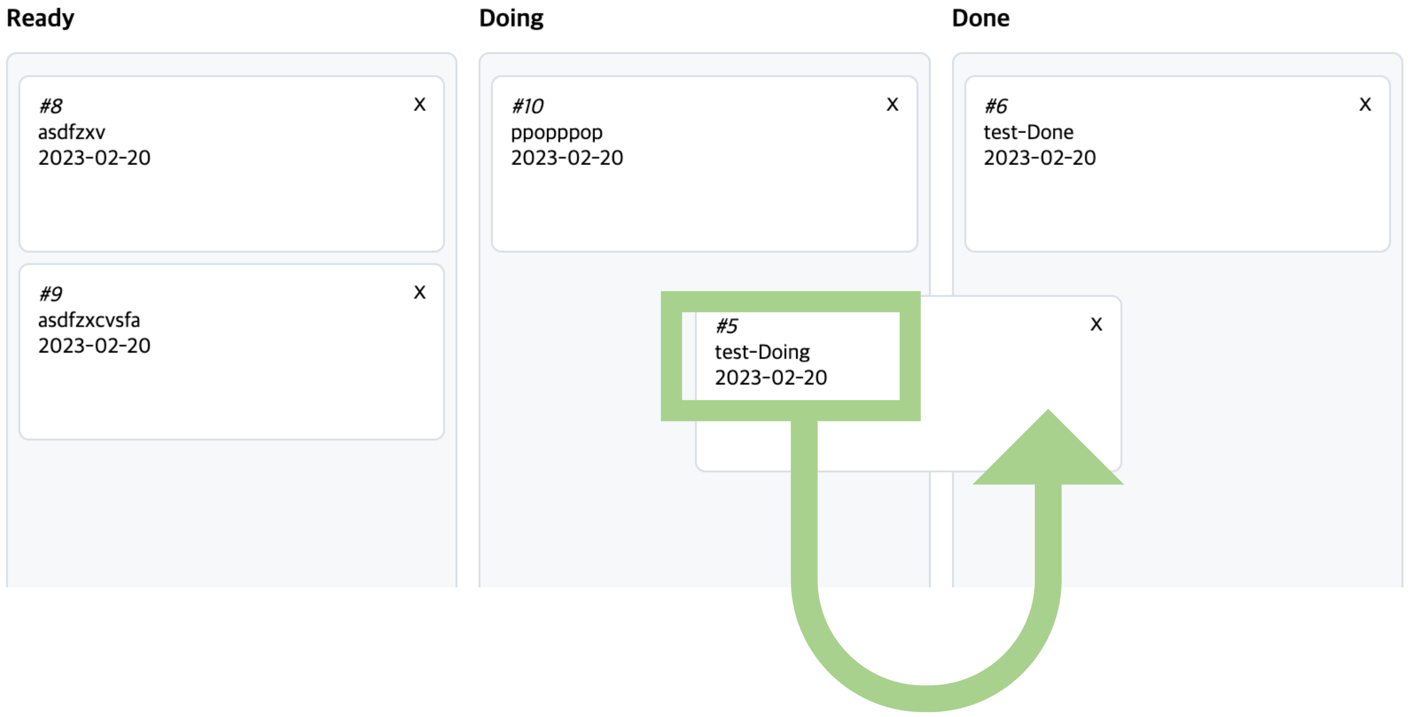
no	status	content	date
5	2	test-Doing	2023-02-20
6	3	test-Done	2023-02-20
7	1	안녕	2023-02-20
8	1	asdfzxv	2023-02-20
9	1	asdfzxcvsfa	2023-02-20
10	2	ppopppop	2023-02-20



no	status	content	date
5	2	test-Doing	2023-02-20
6	3	test-Done	2023-02-20
8	1	asdfzxv	2023-02-20
9	1	asdfzxcvsfa	2023-02-20
10	2	ppopppop	2023-02-20

4 실행

(3)Update


Ready

#8
asdfzxv
2023-02-20

#9
asdfzxcvsfa
2023-02-20

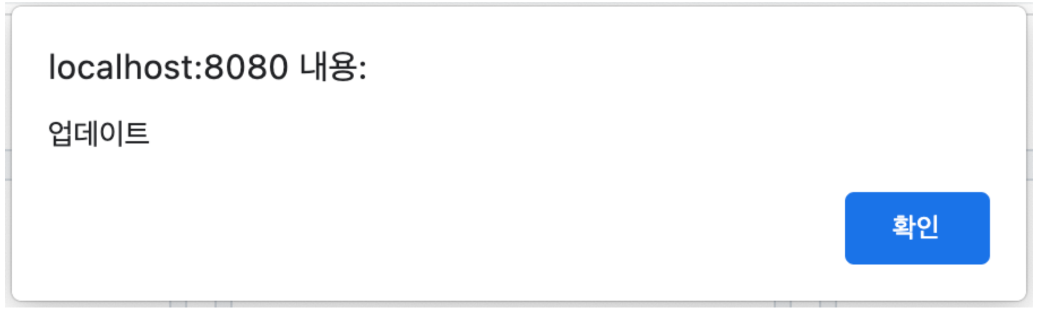
Doing

#10
ppopppop
2023-02-20

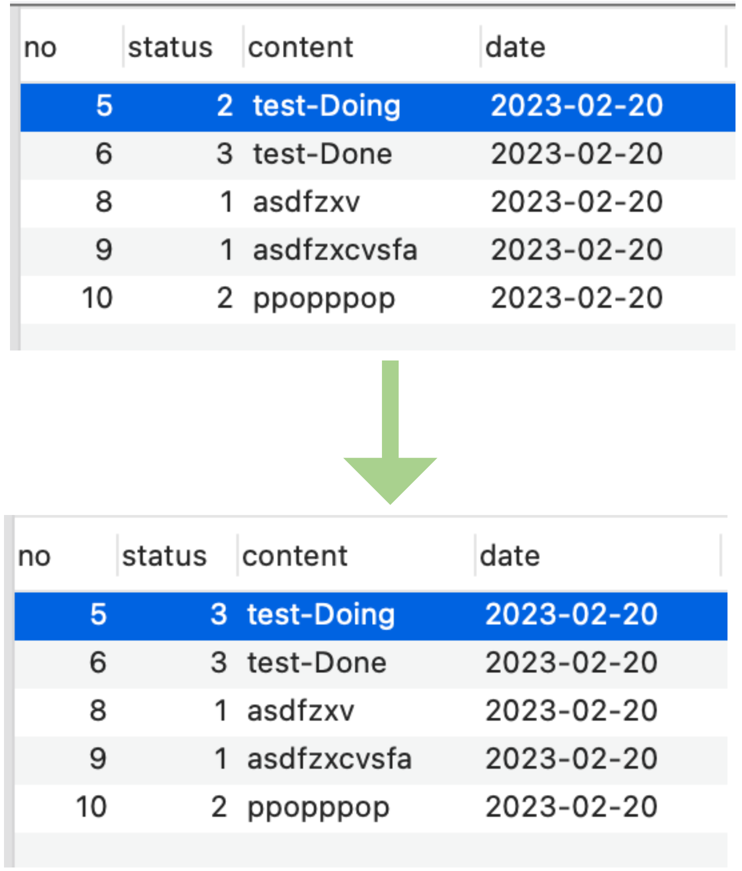
#5
test-Doing
2023-02-20

Done

#6
test-Done
2023-02-20


localhost:8080 내용:

업데이트



no	status	content	date
5	2	test-Doing	2023-02-20
6	3	test-Done	2023-02-20
8	1	asdfzxv	2023-02-20
9	1	asdfzxcvsfa	2023-02-20
10	2	ppopppop	2023-02-20

no	status	content	date
5	3	test-Doing	2023-02-20
6	3	test-Done	2023-02-20
8	1	asdfzxv	2023-02-20
9	1	asdfzxcvsfa	2023-02-20
10	2	ppopppop	2023-02-20