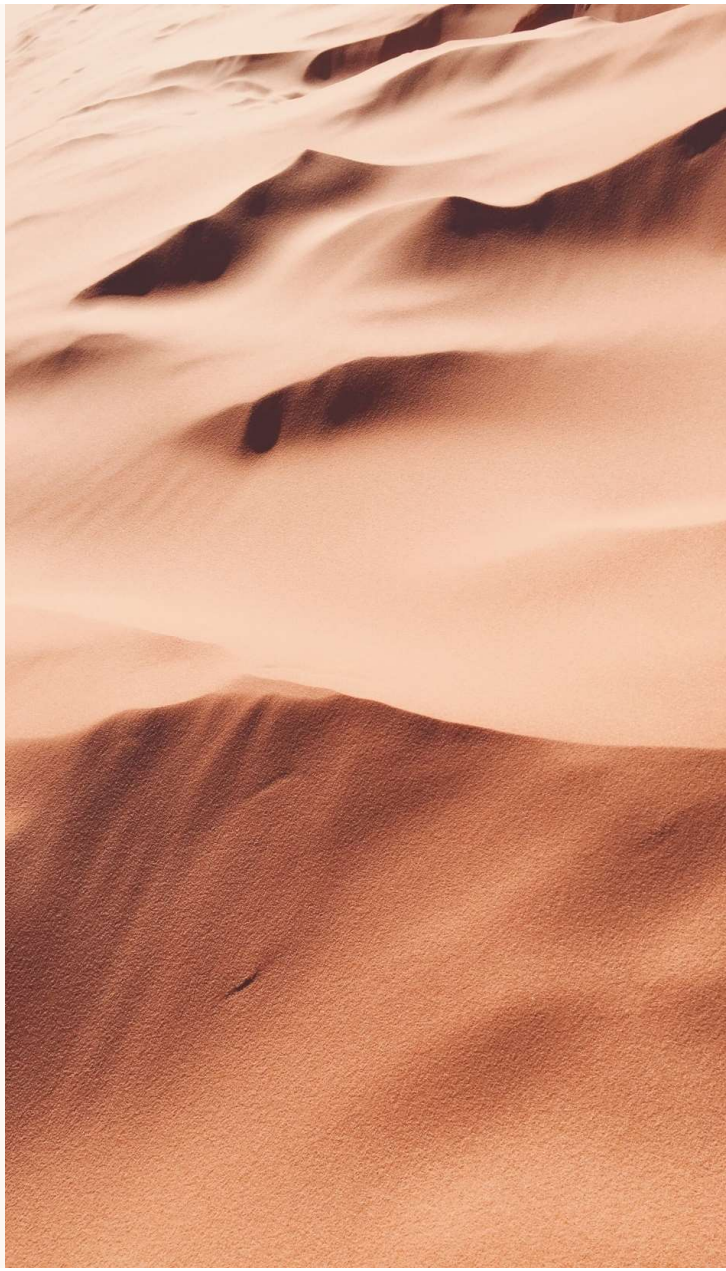
통합 구현

서정현

목차 A table of contents.

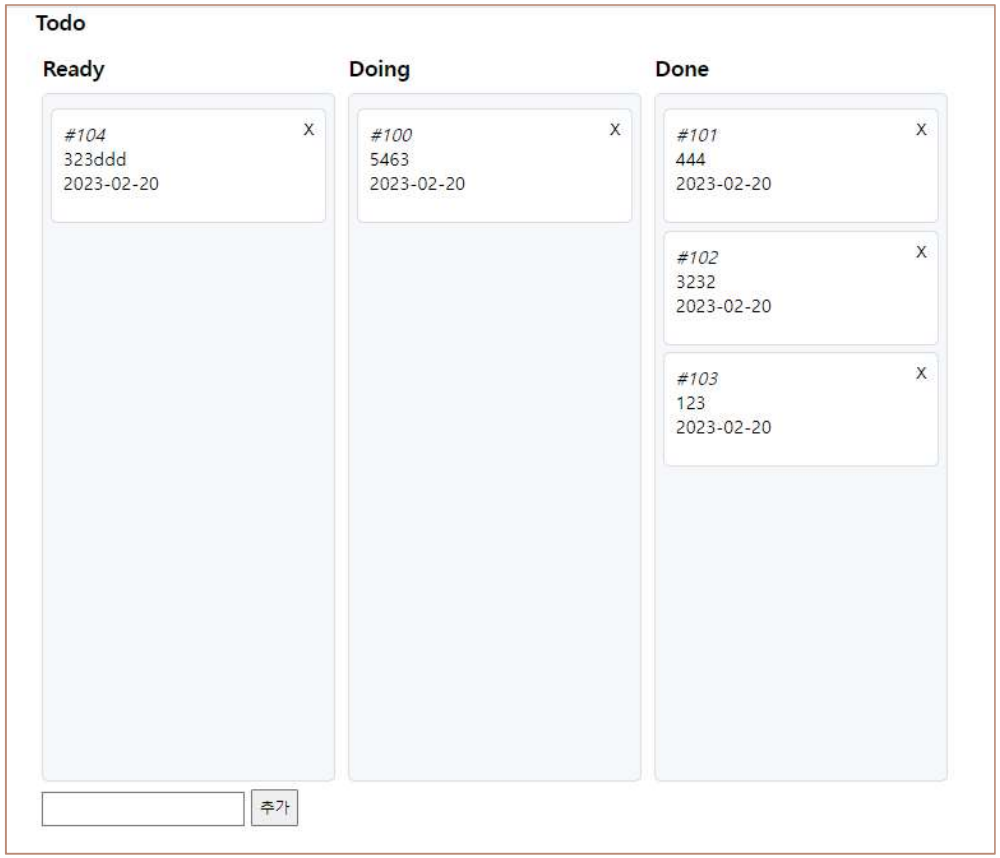


1. 문제 1 – 프로젝트 구성

1. 프로젝개요 : 할 일 관리를 위한 애플리케이션
2. 프로젝트 이름 : Todo
3. Framework : Spring Boot 2.7.8
4. Build : Gradle / Maven
5. Group : kr.co.todo
6. Aritifact :Todo
7. Package : kr.co.todo
8. Packaging : Jar
9. Language : java11
10. Dependencies : Spring Boot DevTools, Lombok, Spring Web, Thymeleaf, Mybatis, MySQL

2. 문제 2 – 화면 구현

1. 화면 레이아웃 작업
2. 아이템 드래깅 처리(jQueryUI sortable API 활용)
3. 하단 리스트 아이템 추가 기능
4. 리스트별 아이템 삭제 기능



2. 문제 2 – 화면 구현

1. index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>Insert title here</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>
        <link rel="stylesheet" th:href="@{/css/style.css}"/>
        <script th:src="@{/js/script.js}"></script>
        <script th:src="@{/js/ajaxAPI.js}"></script>
        <script>
            const contextPath = "[[@{/}]]";
        </script>
    </head>
    <body>
        <div id="wrapper">
            <h3>Todo</h3>
            <section>
                <div>
                    <h3>Ready</h3>
                    <article class="ready" data-status="1">
                        <th:block th:if="${map.get('ready') != null}" >
                            <th:block th:each="ready : ${map.ready}">
                                <div class="item" th:data-no="${ready.no}">
                                    <button class="del">X</button>
                                    <em class="tit">#[[${ready.no}]]</em>
                                    <p>
                                        [[${ready.content}]]
                                    </p>
                                    <span class="date">[[${#strings.substring(ready.rdate, 0, 10)}]]</span>
                                </div>
                            </th:block>
                        </th:block>
                    </article>
                </div>
                <div>
                    <h3>Done</h3>
                    <article class="done" data-status="3">
                        <th:block th:if="${map.get('done') != null}" >
                            <th:block th:each="done : ${map.done}">
                                <div class="item" th:data-no="${done.no}">
                                    <button class="del">X</button>
                                    <em class="tit">#[[${done.no}]]</em>
                                    <p>
                                        [[${done.content}]]
                                    </p>
                                    <span class="date">[[${#strings.substring(done.rdate, 0, 10)}]]</span>
                                </div>
                            </th:block>
                        </th:block>
                    </article>
                </div>
            </section>
            <div class="add">
                <input type="text" name="todo"/>
                <input type="button" id="btnAdd" value="추가"/>
            </div>
        </div>
    </body>
</html>
```

2. 문제 2 – 화면 구현

2. script.js

```
$(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');

                let jsonData = {
                    "no":no,
                    "status":value
                }
                console.log(jsonData);

                ajaxAPI(no, jsonData, "PATCH").then((response) => {
                    console.log(response);
                    if(response.result == 1){
                        //sorting($(this));
                    } else {
                        alert("수정에 실패했습니다. 다시 시도해주세요.");
                    }


                }).catch((errorMsg) => {
                    console.log(errorMsg)
                });
            },
            update: function(e, ui){...}
    });
```

```
$('#btnAdd').click(function(){
    let value = $('input[name=todo]').val();

    if(value.trim() == "") {alert("내용을 입력해주세요."); return;}

    let jsonData = {"content":value};
    console.log(jsonData);

    ajaxAPI("", jsonData, "POST").then((response) => {
        console.log(response);
        if(response.entity != null){


                let item = "<div class='item' data-no='" + response.entity.no + "'>"
                         + "     <button class='del'>X</button>"
                         + "     <em class='tit'>#" + response.entity.no + "</em>"
                         + "     <p>" + response.entity.content + "</p>"
                         + "     <span class='date'>" + response.entity.rdate.substr(0, 10); + "</span>"
                         + " </div>";

                $('.ready').append(item);

        } else {
            alert("업로드에 실패했습니다. 다시 시도해주세요.");
        }


    }).catch((errorMsg) => {
        console.log(errorMsg)
    });


});
```

```
    $(document).on('click', '.del', function(){

        if(!confirm("정말 삭제 하시겠습니까?")) {return;}

        let no =  $(this).parent().data("no");
        let jsonData = {no:"no"}
        ajaxAPI(no, jsonData, "DELETE").then((response) => {
            console.log(response);
            if(response.result == 1){
                $(this).parent().remove();

            } else {
                alert("삭제에 실패했습니다. 다시 시도해주세요.");
            }
        }).catch((errorMsg) => {
            console.log(errorMsg)
        });


    });
})
```

2. 문제 2 – 화면 구현

3. style.css

```css
* {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;
}
```

```css
.item {
    float:left;
    width: 100%;
    height: 100px;
    padding: 10px;
    margin-top: 6px;
    background: white;
    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;
}
```

3. 문제 3 – 테이블 설계

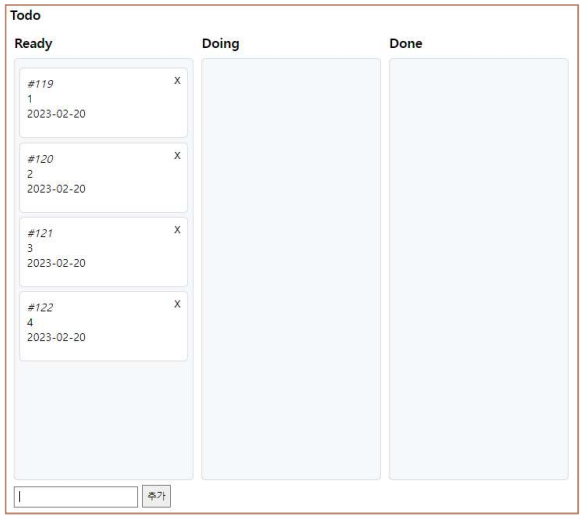
1. 리스트별 데이터 저장을 위한 테이블 설계

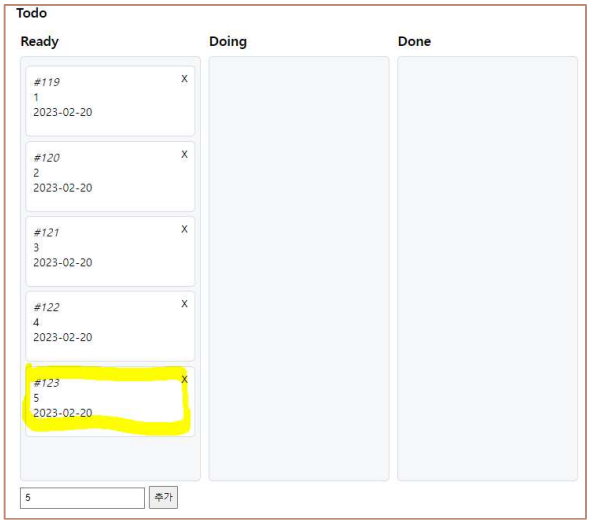
2. 테이블명 : Todo

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	no	INT	10	☐	☐	☐	AUTO_INCREMENT
2	content	VARCHAR	300	☐	☑	☐	NULL
3	status	VARCHAR	50	☐	☑	☐	NULL
4	rdate	DATETIME		☐	☑	☐	CURRENT_TIMESTAMP
5	wdate	DATETIME		☐	☐	☐	CURRENT_TIMESTAMP
6	order	INT	10	☐	☑	☐	NULL

4. 문제 4 – 기능 구현

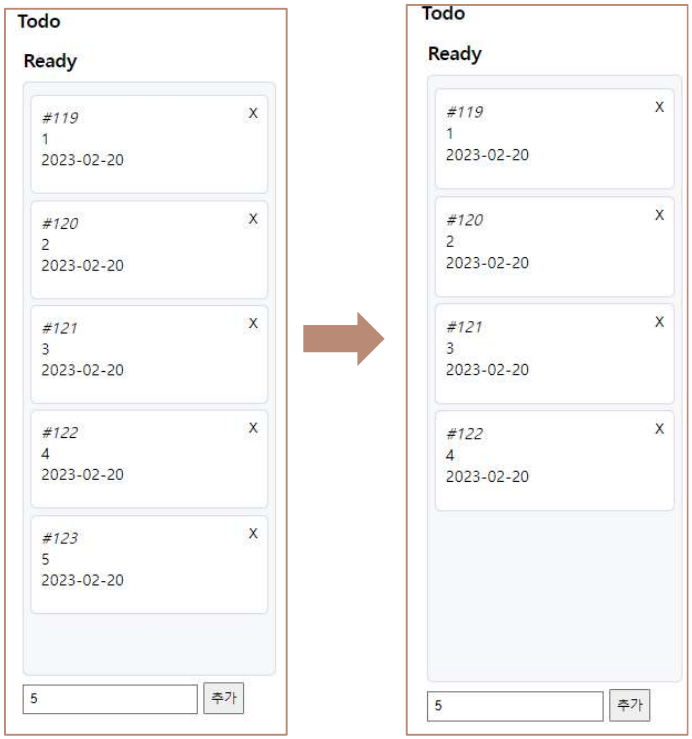
1. 메인화면 아이템 Select 처리
2. 아이템 추가시 Insert 처리



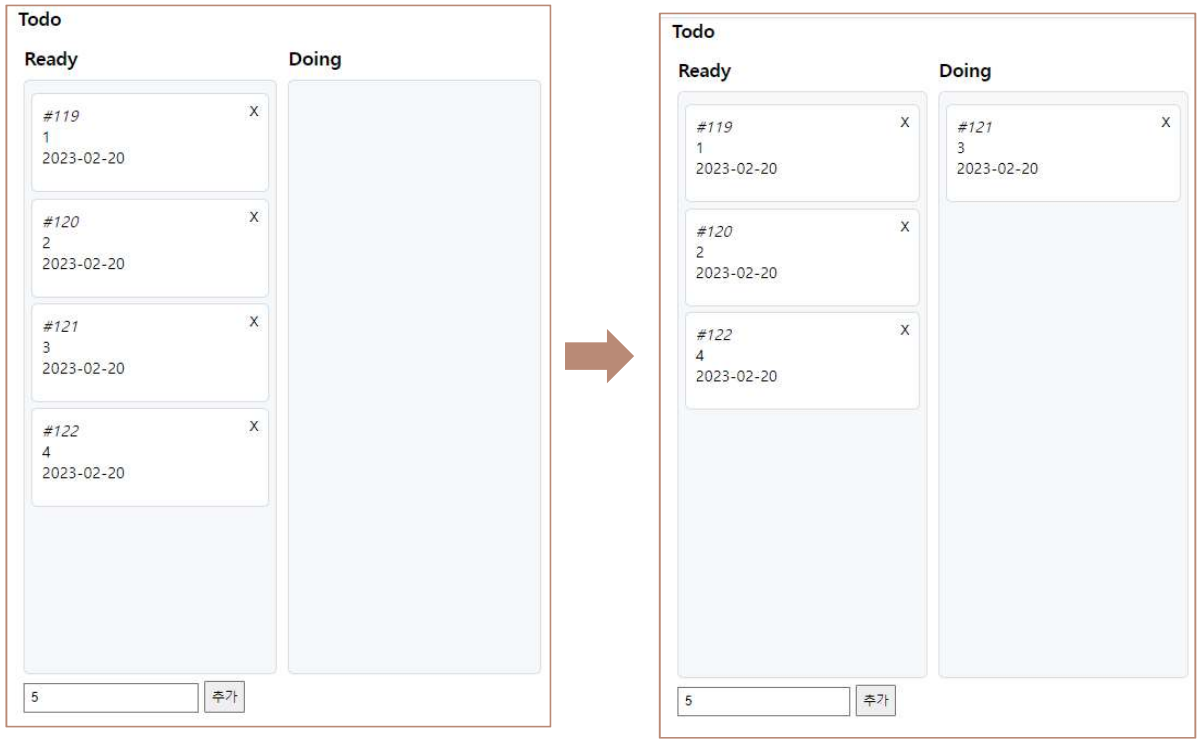


4. 문제 4 – 기능 구현

3. 아이템 삭제시 Delete 처리



4. 아이템 이동시 Update 처리



4. 문제 4 – 기능 구현(controller)

```java
@Slf4j
@Controller
public class MainController {

    @Autowired
    private ArticleService service;


    @GetMapping(value = {"/", "/index"})
    public String index(Model m){
        m.addAttribute( attributeName: "map", service.select());
        return "/index";
    }


    @ResponseBody
    @PostMapping("/")
    public Map write(@RequestBody ArticleVO vo){
        log.info("MainController Post write...");
        log.info(vo.toString());

        ArticleEntity entity = ArticleEntity.builder()
                .content(vo.getContent())
                .rdate(new Date())
                .status("ready")
                .build();

        entity = service.insert(entity);

        Map map = new HashMap();
        map.put("entity", entity);
        return map;
    }
```

```java
    @ResponseBody
    @DeleteMapping("/{no}")
    public Map delete(@PathVariable int no){
        int result = service.delete(no);
        Map map = new HashMap();
        map.put("result", result);
        return map;
    }


    @ResponseBody
    @PatchMapping("/{no}")
    public Map update(@RequestBody Map<String, Integer> map){
        int result = service.update(map.get("no"), map.get("status"));
        map.put("result", result);
        return map;
    }
}
```

4. 문제 4 – 기능 구현(service)

```java
@Service
public class ArticleService {

    @Autowired
    private ArticleRepo repo;

    @Autowired
    private ArticleDAO dao;


    public Map<String, List<ArticleVO>> select(){
        return dao.select().stream().sorted(Comparator.comparing(ArticleVO::getOrder)).collect(Collectors.groupingBy(ArticleVO::getStatus));
    };


    public ArticleEntity insert(ArticleEntity entity) { return repo.save(entity); };


    public int delete(int no) { return dao.delete(no); };

    public int update(int no, int status) { return dao.update(no, convertStatus(status)); };


    public String convertStatus(int status){
        String statusStr = null;
        switch (status){
            case 1: statusStr = "ready"; break;
            case 2: statusStr = "doing"; break;
            case 3: statusStr = "done"; break;
        }
        return statusStr;
    }
}
```

4. 문제 4 – 기능 구현(dao, vo)

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

    List<ArticleVO> select();

    int insert(ArticleVO vo);

    int delete(@Param(value = "no") int no);

    int update(@Param(value = "no") int no, @Param(value = "status") String status);
}
```

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class ArticleVO {
    int no;
    String content;
    String rdate;
    String wdate;
    String status;
}
```

4. 문제 4 – 기능 구현(mapper.xml)

```xml
<?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.ArticleDAO">
    <select id="select" resultType="kr.co.todo.vo.ArticleVO">
        SELECT * FROM `todo`
        ORDER BY `order`;
    </select>
    <insert id="insert" parameterType="kr.co.todo.vo.ArticleVO">
        INSERT INTO `todo` SET
            `content`=#{content},
            `status`='ready'
    </insert>
    <delete id="delete">
        DELETE FROM `todo` WHERE `no`=#{no}
    </delete>
    <update id="update">
        UPDATE `todo` SET `status`=#{status}, `wdate`=now() WHERE `no`=#{no}
    </update>
</mapper>
```

감사합니다