

13. 통합 구현 수행평가

Html, 테이블 설계, 기능 구현

날짜 : 2023/02/20

이름 : 조주영



INDEX

1

프로젝트 생성 및 구성

2

화면 구현

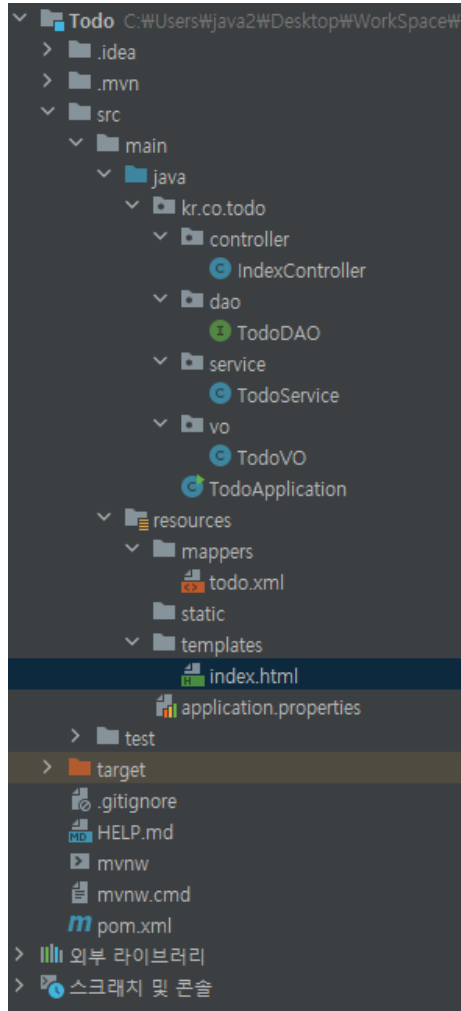
3

테이블 설계

4

기능 구현

1. 프로젝트 생성 및 구성



Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.0.3 (SNAPSHOT) ☐ 3.0.2 ☐ 2.7.9 (SNAPSHOT) ☒ 2.7.8

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 19 ☐ 17 ☒ 11 ☐ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools **DEVELOPER TOOLS**

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok **DEVELOPER TOOLS**

Java annotation library which helps to reduce boilerplate code.

Spring Web **WEB**

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf **TEMPLATE ENGINES**

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

MySQL Driver **SQL**

MySQL JDBC driver.

MyBatis Framework **SQL**

Persistence framework with support for custom SQL, stored procedures and advanced mappings. MyBatis couples objects with stored procedures or SQL statements using a XML descriptor or annotations.

2. 화면 구현

최종 화면

Todo

Ready

#107 X
모레 할 일
2023-02-20

Doing

#104 X
내일 할 일
2023-02-20

#105 X
123
2023-02-20

Done

#100 X
오늘 할 일
2023-02-20

추가

스타일

```
<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: 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;
}
</style>
```

2. 화면 구현

Html code

```
<body>
  <div id="wrapper">
    <h3>Todo</h3>
    <section>
```

```
<div>
  <h3>Ready</h3>
  <article class="ready" data-status="1">
    <th:block th:each="art : ${ready}">
      <div class="item" th:data-no="${art.id}">
        <button class="del">X</button>
        <em class="tit">#[[${art.id}]]</em>
        <p>[[${art.content}]]</p>
        <span class="date">[[${art.date}]]</span>
      </div>
    </th:block>
  </article>
</div>
```

```
<div>
  <h3>Doing</h3>
  <article class="doing" data-status="2">
    <th:block th:each="art : ${doing}">
      <div class="item" th:data-no="${art.id}">
        <button class="del">X</button>
        <em class="tit">#[[${art.id}]]</em>
        <p>[[${art.content}]]</p>
        <span class="date">[[${art.date}]]</span>
      </div>
    </th:block>
  </article>
</div>
```

```
<div>
  <h3>Done</h3>
  <article class="done" data-status="3">
    <th:block th:each="art : ${done}">
      <div class="item" th:data-no="${art.id}">
        <button class="del">X</button>
        <em class="tit">#[[${art.id}]]</em>
        <p>[[${art.content}]]</p>
        <span class="date">[[${art.date}]]</span>
      </div>
    </th:block>
  </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 • create table `Todo` (  
2     `id` int auto_increment primary key,  
3     `content` varchar(255),  
4     `status` tinyint default 1,  
5     `date` datetime  
6 );
```

The screenshot shows a MySQL database management tool interface. On the left, a tree view displays the database structure, with the 'todo' table selected under the 'java2db' database. The table's size is listed as 16.0 KIB. On the right, a table view shows the data for the 'todo' table. The table has four columns: 'id', 'content', 'status', and 'date'. The data is as follows:

| id | content | status | date |
|-----|---------|--------|---------------------|
| 100 | 오늘 할 일 | 3 | 2023-02-20 17:48:32 |
| 104 | 내일 할 일 | 2 | 2023-02-20 17:47:46 |
| 105 | 123 | 2 | 2023-02-20 17:47:49 |
| 107 | 모레 할 일 | 1 | 2023-02-20 17:48:30 |

At the bottom of the interface, two SQL queries are visible:

```
153 SELECT * FROM `java2db`.`todo` LIMIT 100;  
154 SELECT * FROM `java2db`.`todo` LIMIT 100;
```

4. 기능 구현

insert

Ready

#107
모레 할 일
2023-02-20

X

내일 할 일

추가

Doing

#104
내일 할 일
2023-02-20

X

#105
123
2023-02-20

X

Done

#100
오늘 할 일
2023-02-20

X

Ready

#107
모레 할 일
2023-02-20

X

#108
내일 할 일
2023-02-20

X

Doing

#104
내일 할 일
2023-02-20

X

#105
123
2023-02-20

X

Done

#100
오늘 할 일
2023-02-20

X

추가

4. 기능 구현

delete

Ready

#107
모레 할 일
2023-02-20

#108
내일 할 일
2023-02-20

추가

Doing

#104
내일 할 일
2023-02-20

#105
123
2023-02-20

추가

Done

#100
오늘 할 일
2023-02-20

추가

localhost:8080 내용:
삭제 완료!

확인

Todo

Ready

#107
모레 할 일
2023-02-20

#108
내일 할 일
2023-02-20

추가

4. 기능 구현

update

The image illustrates a task update process. On the left, a Kanban board shows three columns: Ready, Doing, and Done. In the 'Doing' column, task #104 '내일 할 일' (Tomorrow's task) is highlighted with a yellow box. The database screenshot shows the 'todo' table with columns id, content, status, and date. Task #104 is highlighted with a yellow box, showing its status as 2 and date as 2023-02-20 17:47:46.

| id | content | status | date |
|-----|---------|--------|---------------------|
| 100 | 오늘 할 일 | 3 | 2023-02-20 17:48:32 |
| 104 | 내일 할 일 | 2 | 2023-02-20 17:47:46 |
| 107 | 모레 할 일 | 1 | 2023-02-20 17:48:30 |
| 108 | 내일 할 일 | 1 | 2023-02-20 18:01:58 |

4. 기능 구현 - 코드 첨부

controller

```
0개의 사용위치
@Controller
public class IndexController {

    6개 사용 위치
    @Autowired
    private TodoService service;

    0개의 사용위치
    @GetMapping(value = {"/", "/index"})
    public String index(Model model) {
        List<TodoVO> ready = service.selectTodos( status: 1);
        List<TodoVO> doing = service.selectTodos( status: 2);
        List<TodoVO> done = service.selectTodos( status: 3);

        model.addAttribute( attributeName: "ready", ready);
        model.addAttribute( attributeName: "doing", doing);
        model.addAttribute( attributeName: "done", done);
        return "index";
    }
}
```

```
0개의 사용위치
@ResponseBody
@PostMapping("insert")
public Map<String, Integer> insert(@RequestParam("content") String content){
    int result = service.insertTodo(content);

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

    return resultMap;
}

0개의 사용위치
@ResponseBody
@PostMapping("update")
public Map<String, Integer> update(@RequestParam("id") int id, @RequestParam("status") int status){
    int result = service.updateTodo(id, status);

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

    return resultMap;
}

0개의 사용위치
@ResponseBody
@GetMapping("delete")
public Map<String, Integer> delete(@RequestParam("id") int id){
    int result = service.deleteTodo(id);

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

    return resultMap;
}
}
```

4. 기능 구현 - 코드 첨부

script

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

      $.ajax({
        url: 'update',
        type: 'post',
        data: {'id':id, 'status':status},
        dataType: 'json',
        success: function(data){
          if (data.result > 0) {
            location.reload();
          }
        }
      });
    }
  });
});
```

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

  $.ajax({
    url: 'insert',
    type: 'post',
    data: {'content':content},
    dataType: 'json',
    success: function(data){
      if (data.result > 0) {
        alert('등록 성공!');
        location.reload();
      }
    }
  });
});

$(document).on('click', '.del', function(){
  let id = $(this).parent().attr('data-no');

  $.ajax({
    url: 'delete',
    type: 'get',
    data: {'id':id},
    success: function(data){
      if (data.result > 0) {
        alert('삭제 완료!');
      }
    }
  });

  $(this).parent().remove();
});
});
```

> script

4. 기능 구현 - 코드 첨부

VO

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class TodoVO {

    0개의 사용위치
    private int id;
    0개의 사용위치
    private String content;
    0개의 사용위치
    private int status;
    1개의 사용 위치
    private String date;

    0개의 사용위치
    public String getDate() {
        return date.substring(0, 10);
    }
}
```

dao

```
3개의 사용 위치
@Repository
@Mapper
public interface TodoDAO {

    1개의 사용 위치
    public int insertTodo(String content);
    1개의 사용 위치
    public TodoVO selectTodo(int id);
    1개의 사용 위치
    public List<TodoVO> selectTodos(int status);
    1개의 사용 위치
    public int updateTodo(@Param("id") int id, @Param("status") int status);
    1개의 사용 위치
    public int deleteTodo(int id);
}
```

service

```
@Service
public class TodoService {

    5개의 사용 위치
    @Autowired
    private TodoDAO dao;

    1개의 사용 위치
    public int insertTodo(String content){
        return dao.insertTodo(content);
    }

    0개의 사용위치
    public TodoVO selectTodo(int id){
        return dao.selectTodo(id);
    }

    3개의 사용 위치
    public List<TodoVO> selectTodos(int status){
        return dao.selectTodos(status);
    }

    1개의 사용 위치
    public int updateTodo(int id, int status){
        return dao.updateTodo(id, status);
    }

    1개의 사용 위치
    public int deleteTodo(int id){
        return dao.deleteTodo(id);
    }
}
```

4. 기능 구현 - 코드 첨부

mappers / todo.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.TODODAO">

  <insert id="insertTodo" keyProperty="id">
    INSERT INTO `todo` set
    `content`=#{content},
    `date`= NOW();
  </insert>

  <select id="selectTodos" resultType="kr.co.todo.vo.TODOVO">
    SELECT * FROM `todo` WHERE `status` = #{status} ORDER BY `date` ASC;
  </select>

  <update id="updateTodo">
    UPDATE `todo`
    SET `status` = #{status}, `date` = NOW()
    WHERE `id` = #{id};
  </update>

  <delete id="deleteTodo">
    DELETE FROM `todo` WHERE `id`= #{id};
  </delete>

</mapper>
```

감사합니다.