Tutorial 5 Write-Up

Muhammad Alvin Abyan – 1506721844 – APAP [A]

Yang saya pelajari pada tutorial 5 ini adalah saya dapat mempelajari bagaimana menggunakan database serta relasi database dalam project Spring Boot. Juga melihat dan mengetahui bagaimana mengimplementasi MyBatis dan Lombok *framework* pada Spring Boot+Maven.

- Method yang Anda ubah pada Latihan Merubah SelectAllStudents, jelaskan
  - Yang saya ubah pada Latihan Merubah SelectAllStudents adalah saya menambahkan beberapa baris berikut pada viewall.html untuk menampilkan mata kuliah apa saja yang diambil oleh masing-masing siswa dan berapa sks

```
<div th:each="student,iterationStatus: ${students}">
    <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
    <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
    <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
    <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
    <h3>Kuliah yang diambil</h3>
    <ul th:each="course,iterationStatus: ${student.courses}">
        <li th:text="${course.name} + '-' + ${course.credits} + ' sks'">
            Nama kuliah-X SKS
        </li>
    </ul>
    <a th:href="'/student/delete/' + ${student.npm}" > Delete Data</a><br/>
    <a th:href="'/student/update/' + ${student.npm}" > Update Data</a><br/>
    <hr/>
</div>
```

  - Selain itu, saya juga mengubah query pada method selectStudent pada StudentMapper untuk mengambil hasil dari query menjadi property yang dimasukkan kedalam List

```
@Select("select npm, name, gpa from student where npm = #{npm}")
@Results(value = {
        @Result(property="npm", column="npm"),
        @Result(property="name", column="name"),
        @Result(property="gpa", column="gpa"),
        @Result(property="courses", column="npm",
        javaType = List.class,
        many=@Many(select="selectCourses"))
})
StudentModel selectStudent (@Param("npm") String npm);
```

- Method yang Anda buat pada Latihan Menambahkan View pada Course, jelaskan
  - Pada Latihan Menambahkan View pada Course saya menambahkan beberapa class, yaitu:
    - CourseController

```
public class CourseController {

    @Autowired
    CourseService courseDAO;

    @RequestMapping("/course/view/{id_course}")
    public String viewCourse (Model model,
            @PathVariable(value = "id_course") String id_course)
    {
        CourseModel course = courseDAO.selectCourse (id_course);

        if (course != null) {
            model.addAttribute ("course", course);
            return "viewcourse";
        } else {
            model.addAttribute ("id_course", id_course);
            return "not-found";
        }
```

- CourseService (interface)

```java
package com.example.service;

import com.example.model.CourseModel;

public interface CourseService {
    CourseModel selectCourse(String id_course);
}
```

- CourseServiceDatabase

```java
import com.example.dao.CourseMapper;
import com.example.model.CourseModel;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Service
public class CourseServiceDatabase implements CourseService {

    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseModel selectCourse(String id_course) {
        log.info ("select course with id course {}", id_course);
        return courseMapper.selectCourse(id_course);
    }
}
```

- CourseMapper

```java
package com.example.dao;

import java.util.List;

@Mapper
public interface CourseMapper {

    @Select("select student.npm, name, gpa " +
            "from studentcourse join student " +
            "on studentcourse.npm = student.npm "
            + "where studentcourse.id_course = #{id_course}")
     List<StudentModel> selectCourseStudent (@Param("id_course") String id_course);

    @Select("select id_course, name, credits " +
             "from course " +
              "where id_course = #{id_course}")
    @Results(value = {
            @Result(property="idCourse", column="id_course"),
            @Result(property="name", column="name"),
            @Result(property="credits", column="credits"),
            @Result(property="students", column="id_course",
                    javaType = List.class,
                    many=@Many(select="selectCourseStudent"))
        }) CourseModel selectCourse (@Param("id_course") String id_course);
}
```

- CourseModel (sudah ada)

```java
package com.example.model;

import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CourseModel {
    private String idCourse;
    private String name;
    private Integer credits;
    private List<StudentModel> students;
}
```

- Viewcourse.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course</title>
    </head>
    <body>
        <h3 th:text="'ID = ' + ${course.idCourse}">Course ID</h3>
        <h3 th:text="'Nama = ' + ${course.name}">Course Name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">Course Credits</h3>

        <h3>Mahasiswa yang mengambil</h3>
        <ul th:each="student,iterationStatus: ${course.students}">
            <li th:text="${student.npm} + '-' + ${student.name}">
                Npm-mahasiswa
            </li>
        </ul>
    </body>
</html>
```

Untuk membuat fungsi baru, kita sebaiknya membuat satu set kesatuan yang baru (controller-service-view-database-mapper-model) agar terpisah dari fungsi lain dan tidak menyebabkan konflik antar fungsi