Tutorial 5

Yang saya pelajari dari tutorial kali ini adalah bagaimana menggunakan database serta relasi database dalam project Spring Boot. Di tutorial ini, database-nya terdapat 2 entity yang saling berelasi, yaitu entity Course dan Student yang memiliki hubungan many-to many. Karena itu, dalam kodingan harus dibuat bagian-bagian untuk setiap entity yang ada. Kemudian tidak lupa untuk membuat list pada setiap kelas Model yang terhubung. Pada kelas Mapper juga saya belajar bagaimana menggunakan anotasi Results, dimana property perlu diisi dengan nama variable yang sesuai dengan kelas Model, dan column diisi dengan yang ada di queri.

- Method yang Anda ubah pada Latihan Merubah SelectAllStudents, jelaskan
  Dengan mengubah method selectAllStudents() pada StudentMapper, yang awalnya adalah:

```
@Select("select npm, name, gpa from student")
List<StudentModel> selectAllStudents ();
```

  Menjadi seperti di bawah ini, seperti penambahan yang dilakukan untuk method selectSttudent():

```
@Select("select npm, name, gpa from student")
@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"))
})
List<StudentModel> selectAllStudents ();
```

  Selanjutnya tinggal mengubah bentuk tampilan awal viewall.html dengan menambah barisan kode berikut:

```
<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>
```

- Method yang Anda buat pada Latihan Menambahkan View pada Course, jelaskan
  Untuk menambahkan view pada course, dimana telah terbuat kelas CourseModel, maka selanjutnya dibuat kelas CourseMapper, yang isinya:

```
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}")
    List<StudentModel> selectStudents(@Param("id_course") String idCourse);

    @Select("select id_course, name, credits from course where id_course = #{id}")
    @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="selectStudents"))
    })
    CourseModel selectCourse(@Param("id") String id);
}
```

Method selectStudent() memiliki logika yang sama dengan selectCourse() di StudentMapper, sedangkan method selectCourse memiliki logika seperti selectStudent(). Setelah itu dibuat kelas CourseService seperti ini:

```
package com.example.service;

import com.example.model.CourseModel;

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

Kemudian buat kelas CourseServiceDatabase yang mengimplementasi kelas CourseService tersebut, dengan isi seperti:

```
package com.example.service;

import org.springframework.beans.factory.annotation.Autowired;

@Slf4j
@Service
public class CourseServiceDatabase implements CourseService
{
    @Autowired
    private CourseMapper courseMapper;


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

}
```

Terakhir, dibuat kelas StudentController untuk menampilkan tampilan dari view ini, yang berisi seperti berikut:

```java
package com.example.controller;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
public class CourseController
{
    @Autowired
    CourseService courseDAO;

    //Latihan Tutorial 5 nomor 2. Menampilkan data course.
    @RequestMapping("/course/view/{id_course}")
    public String viewCourse (Model model, @PathVariable(value = "id_course") String id)
    {
        CourseModel course = courseDAO.selectCourse(id);

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

Bentuk view-course.html adalah seperti berikut:

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

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

Bentuk not-found-course.html adalah seperti berikut:

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Course not found</title>
    </head>
    <body>
        <h1>Course not found</h1>
        <h3 th:text="'ID = ' + ${id}">ID Course</h3>
    </body>
</html>
```