LESSON LEARNED

Hal yang saya pelajari pada tutorial kali ini adalah saya mempelajari bagaimana menggunakan database dan relasi database dalam arsitektur MVC.

LATIHAN MENGUBAH SELECTALLSTUDENTS

Langkah pertama yang perlu dilakukan adalah mengubah method selectAllStudents pada StudentMapper.

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

```
@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();
```

Pengubahan yang dilakukan dari kotak pertama adalah pemetaan hasil pengambilan data dari student itu sendiri kepada masing-masing atribut dari kelas model. Seperti yang sudah dijelaskan pada dokumen tutorial 5, anotasi @Result digunakan untuk memetakan hasil dari query select ke kelas model. Untuk variabel courses, query yang digunakan adalah seperti di kotak kedua karena hasilnya diambil dari method lain. Property diset dengan variabel courses karena kita ingin mengisi variabel courses di kelas StudentModel, sedangkan variabel column diisi dengan npm karena kolom npm pada database akan dikirim menjadi parameter di method selectCourses. Variabel javaType menentukan class yang menjadi kembalian. Kemudian variabel many diset dengan method yang akan mengisi variabel courses yaitu selectCourses.

Langkah selanjutnya adalah menambahkan tampilan daftar courses yang diambil oleh mahasiswa ybs pada viewall.html setelah informasi mengenai GPA ditampilkan.

```
<h3 th:text="'Kuliah yang diambil:'">List Courses</h3>
<ul th:each="course, iterationStatus: ${student.courses}">
  <li th:text="${course.name} + ' - ' + ${course.credits} + ' sks'">
      Course Name - X SKS</li>
</ul>
```

Baris ini persis seperti pada halaman view.html yang telah dilakukan sebelumnya.

Selanjutnya, aplikasi dapat dijalankan dan dapat menampilkan data mata kuliah yang diambil dari masing-masing student ybs.

LATIHAN MENAMBAH VIEW PADA COURSE

Untuk memulai latihan ini, saya membuat mapper, service, dan controller baru untuk model course. Oleh karena itu, saya memulai latihan ini dengan membuat mapper dengan nama CourseMapper.

```
package com.tutorial5.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Many;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Result;
import org.apache.ibatis.annotations.Results;
import org.apache.ibatis.annotations.Select;

import com.tutorial5.model.CourseModel;
import com.tutorial5.model.StudentModel;

@Mapper
public interface CourseMapper {
  @Select("SELECT id_course, name, credits FROM course WHERE id_course
    = #{idCourse}")
  @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("idCourse") String idCourse);

  @Select("SELECT student.npm, name, gpa FROM student join
    studentcourse on student.npm = studentcourse.npm WHERE
    studentcourse.id_course = #{id_course}")
  List<StudentModel> selectStudents(@Param("id_course") String
    idCourse);
}
```

Anotasi @Select yang pertama berfungsi untuk mengambil data dari suatu course berdasarkan id_course tersebut, sedangkan anotasi @Select kedua berfungsi untuk mengambil data student yang mengambil course ybs.

Selanjutnya adalah membuat service dari course dan service databasenya.

```
package com.tutorial5.service;

import com.tutorial5.model.CourseModel;
```

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

```
package com.tutorial5.service;

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

import com.tutorial5.mapper.CourseMapper;
import com.tutorial5.model.CourseModel;

import lombok.extern.slf4j.Slf4j;

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

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

Kemudian, buat controller untuk menampilkan halaman course.

```
package com.tutorial5.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.tutorial5.model.CourseModel;
import com.tutorial5.service.CourseService;

@Controller
public class CourseController {
  @Autowired
  CourseService courseDAO;

  @RequestMapping("/course/view")
  public String view(@RequestParam(value = "idCourse", required =
    false) String idCourse, Model model) {
    CourseModel course = courseDAO.selectCourse(idCourse);
```

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

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

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

Method view dibuat untuk mengatasi jika user tidak memasukkan id course pada path variable.

Langkah selanjutnya adalah membuat halaman view-course.html untuk menampilkan detail course dan halaman not-found-course.html untuk menampilkan error jika user tidak memasukkan id course atau memasukkan id course yang tidak ada di database.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

  <head>
    <title>View Course by ID 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 th:text="'Mahasiswa yang mengambil:'">List Students</h3>
    <ul th:each="student, iterationStatus: ${course.students}">
      <li th:text="${student.npm} + ' - ' + ${student.name}">Student
        NPM - Student Name</li>
    </ul>
  </body>

</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 Course = ' + ${idCourse}">Course ID</h3>
  </body>

</html>
```

Selanjutnya, aplikasi dapat dijalankan dan dapat melihat detail data dari course ybs.