Muhammad Farhan Mardadi
1506757743

Tutorial 5 APAP

- **Apa saja yang pelajari dari tutorial ini**
  Yang saja pelajari dari tutorial kali ini adalah menggunakan database dengan relasi pada databasenya. Pada tutorial sebelumnya pada database hanya terdapat tabel student, namun pada tutorial kali ini ditambah dengan tabel course. Karena tabel student dan course memiliki relasi "take" maka diperlukan untuk membuat tambahan tabel untuk menghubungkan tabel tersebut, yaitu tabel student course yang mempresentasikan bahwa student mengambil suatu matkul. Dengan dua tabel yang memiliki relasi maka diperlukan teknik yang berbeda dengan yang hanya menggunakan satu tabel saja.

  Hal paling signifikan pada tutorial kali ini dibanding tutorial sebelumnya adalah method selectStudent pada class StudentaMapper. Karena adanya anotasi baru yang dipakai yaitu anotasi "@Results" yang digunakan untuk memetakan hasil query ke dalam class model.

- **Method yang Anda ubah pada Latihan Merubah SelectAllStudents, jelaskan**
  Untuk mengubah method selectAllStudents di class StudentMapper sama halnya dengan mengubah method selectStudent sebelumnya, yaitu dengan menambahkan anotasi "@Results" dan implementasinya. Jadi saya yang ubah pada method ini sama dengan method selectStudent.

```
@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 ();
```

  Setelah mengubah method diperlukan juga untuk menambah beberapa baris kode pada view untuk menampilkan mata kuliah yang diambil oleh student. Kode yang ditambahkan juga sama seperti yang ditambahkan pada view untuk view.html.

```
<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/update/' + ${student.npm}">Update Data</a><br/>
    <a th:href="'/student/delete/' + ${student.npm}">Delete Data</a><br/>
    <hr/>
</div>
```

- **Method yang Anda buat pada Latihan Menambahkan View pada Course, jelaskan**

Untuk membuat view pada course pertama-tama yang membuat interface CourseService. Saya membuat satu method untuk dapat memilih course.

```
public interface CourseService {

    CourseModel selectCourse(String id);

}
```

Lalu saya membuat controller pada kelas CourseController. Yang berguna mengolah hasil apa yang akan ditampilkan di view.

```
@Controller
public class CourseController {

    @Autowired
    CourseService courseDAO;

    @RequestMapping("/course/view/{id}")
    public String viewCourse(Model model, @PathVariable(value = "id") String id) {

        CourseModel course = courseDAO.selectCourse(id);

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

}
```

Lalu saya membuat class StudentMapper untuk melalukan query dan memetakan hasil query ke dalam object.

```
@Mapper
public interface CourseMapper {

    @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);

    @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") String id);

}
```

Lalu saya membuat class CourseServiceDatabase dengan mengimplements interface CourseService. Class ini berguna untuk memanggil method-method di class mapper.

```
@Slf4j
@Service
public class CourseServiceDatabase implements CourseService {

    @Autowired
    CourseMapper courseMapper;

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

}
```

Dan yang terakhir saya membuat view untuk menampilkan hasilnya.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by ID</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 - name</li>
        </ul>
    </body>
</html>
```