UNIVERSITAS INDONESIA

ARSITEKTUR DAN PEMROGRAMAN APLIKASI PERUSAHAAN - KELAS A

WRITE UP – TUTORIAL 5

MENGGUNAKAN DATABASE SERTA RELASI DATABASE DALAM PROJECT SPRING BOOT

LUTHFI ABDURRAHIM

1406557535

FAKULTAS ILMU KOMPUTER

PROGRAM STUDI ILMU KOMPUTER

DEPOK

OKTOBER 2017

Daftar Isi

1. Jelaskan apa saja hal yang Anda pelajari dari tutorial ini.

Hal yang saya pelajari dari ini ialah, saya mengetahui secara spesifik dimana saya dapat melakukan pengagabungan atau join antar table pada spring boot projectt. Intinya pertama, kita buat pada file Mapper, dalam hal ini StudentMapper.java dan membuat query untuk menggabungkannya, dan beberapa hal lain untuk menggabungkannya.

Studi kasus yang digunakan untuk menggabugnkan dua table yaitu, databse-nya memiliki 2 entity yang saling berelasi, yaitu entity Course dan Student yang memiliki hubungan many-to-many. Sehingga, pada saat implementasi diharuskan untuk membuat bagian-per-bagian untuk setiap entity yang ada. Kemudian saya belajar mengenai tentang penggunaan anotasi Results pad akelas Mapper, dimana property memerlukan pengisian dengan nama variable yang sesuai dengan kelas model, dan column diisi sesuai dengan nama kolom di database.

Hal yang perlu dicatat yaitu: Pada Result, “Property diset dengan variabel courses karena kita ingin mengisi variabel courses di class StudentModel. 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”.

2. Method yang Anda ubah pada Latihan Merubah SelectAllStudents, jelaskan

Method SelectAllStudents dirubah dengan yang awalnya yaitu:

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

Menjadi seperti berikut:

```
// list all student with their list of courses
@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 ();
```

Lalu mengubah pada file view.html dengan menambah kodingan berikut ini:

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

Yang berbeda pada selectAllStudent sebelumnya yaitu menambahkan course yang diambil pada masing-masing mahasiswa. Sehingga dibutuhkan join table mahasiswa dan course pada selectAllStudents() di StudentMapper.java. Untuk menggabugnkannya, diperukan annotation Result, property=”courses” karena kita ingin memanggil courses dari studentModel dengan parameter npm, type yang kita inginkan untuk ditampilkan yaitu List maka javaType=List, lalu hubungan antar table yaitu many to many maka many=@Many .

3. Method yang Anda buat pada Latihan Menambahkan View pada Course, jelaskan

Untuk menampilkan view pada course, saya melakukan pemisahan object dengan student. Maksudnya, yaitu dimulai dari pembuatan CourseModel, CourseMapper, CourseService, CourseServiceDatabase, CourseController, lalu, view-course.html, not-found-course.html

Dibuat terpisah, agar dapat menerapkan object oriented dengan baik. Meskipun, saya telah melakukan penerapan penambahan method pada StudentController untuk menampilkan course dan berhasil serta meskpun hanya menerapkan satu fungsionalitas yaitu hanya menampilkan. Namun, menurut saya tidak best practice jika menjadikan satu dengan student, karena menampilkan course itu merupakan kepemilikan dari course, sehingga perlu dipisah dalam pengimplementasiannya.

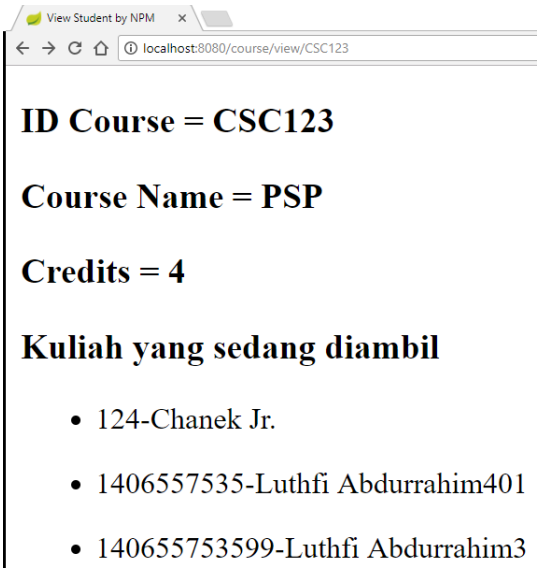
Berikut ini implementasi pada CourseMapper.java

```
@Mapper
public interface CourseMapper {
    //List a course based on id_course
    @Select("select id_course, name, "
            + "credits from course where id_course = #{id_course}")
    @Results(value= {
            @Result(property="id_course", 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_course") String id_course);


    //list of all students on a course with id_course tertentu
    @Select("select student.npm, name, gpa "+
            "from studentcourse join student " +
            "on studentcourse.npm = student.npm " +
            "where studentcourse.id_course = #{id_course}")
    List<StudentModel> selectStudents (@Param("id_course") String id_course);
}
```

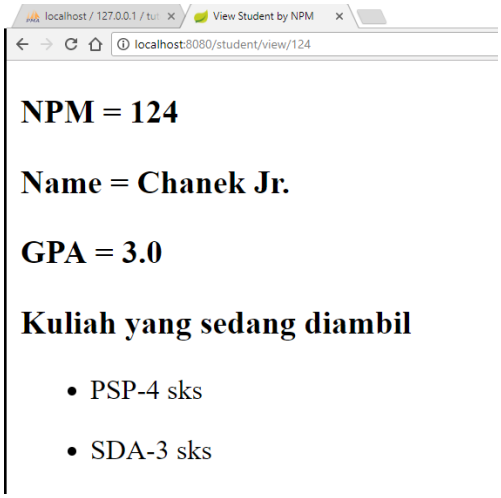
Berikut ini implementasi pada CourseController.java

```
@Controller
public class CourseController {

    @Autowired
    CourseService courseDAO;

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

        CourseModel course = courseDAO.selectCourse(id_course);

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

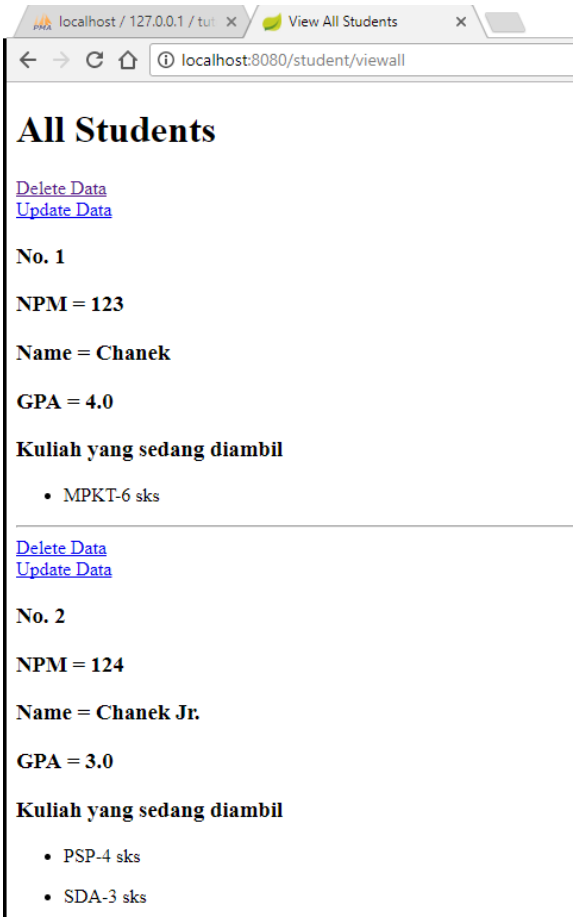
4. Screenshot Pendukung



Gambar 2 View Course by id_course sukses



Gambar 3 View Student by NPM dan memunculkan Course-nya



Gambar 1 View AllStudents dan Courses masing-masing