## 1. Merubah method selectAllStudents

Saya mempelajari hal baru terkait membuat relasi database dalam Project Spring Boot. Pada tutorial kali ini, saya mendapatkan pengetahuan baru mengenai anotasi @Result, variabel property, dan variabel column. Anotasi @Result digunakan untuk memetakan hasil dari query select ke class model. Variabel property diisi dengan nama variabel di suatu class sedangkan variabel column diisi dengan nama kolom hasil kueri di database (parameter yang dibutuhkan). Selain itu, tutorial kali in juga mengingatkan saya terhadap beberapa kueri yang telah diajarkan pada mata kuliah basis data.

### Method 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 ();
```

Method selectAllStudent mengambil data npm, nama, dan gpa dari tabel student serta nama course pada tabel course. Seperti yang telah disampaikan diatas, anotasi @Result digunakan untuk memetakan hasil dari query select ke class model. Variabel column diisi dengan npm karena kolom npm pada database akan dikirim menjadi parameter di method selectCourses. Kemudian variabel many diset dengan method selectCourses.

### viewCourse.html dan hasil

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

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

            </li>
        </ul>
    </body>
</html>
```

# All Students

Delete Data
Update Data

**No. 1**

**NPM = 123**

**Name = Bintang**

**GPA = 3.54**

**Kuliah yang diambil**

- MPKT-6 sks

---

Delete Data
Update Data

**No. 2**

**NPM = 124**

**Name = Glenn**

**GPA = 3.51**

**Kuliah yang diambil**

- PSP-4 sks
- SDA-3 sks

---

## 2. Menambahkan view pada course

Pertama, buat method selectCourse pada studentMapper yang memiliki paramater (id) yang berasal dari request Mapping. Method tersebut digunakan untuk mengakses data dari tabel course pada database dan menyimpannya ke course model. Kemudian untuk mendapatkan list dari students, method tersebut akan menggunakan bantuan dari method selectStudentCourses.

Method selectStudentCourses digunakan untuk mengambil data npm dan nama student dengan melakukan join antara tabel studentcourse dan student berdasarkan id_course suatu mata kuliah. Sehingga kedua method tersebut akan mengembalikan course model yang berisikan id_course, name, credits, dan list studentmodel yang mengikuti mata kuliah tersebut.

```
@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="selectStudentCourses"))
})
CourseModel selectCourse (@Param("id") String id);

@Select("SELECT student.npm, name " +
        "FROM studentcourse JOIN student " +
        "ON studentcourse.npm = student.npm " +
        "WHERE studentcourse.id_course = #{id}")
List<StudentModel> selectStudentCourses (@Param("id") String id);
```

Kemudian buat method viewCourse pada class student controller. Method tersebut akan membuat course model dan memanggil method selectCourse yang telah dibuat sebelumnya dan berisikan parameter id. Lalu menampilkan halaman “viewCourse.html” .

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

    model.addAttribute("course", course);
    return "viewCourse";
}
```

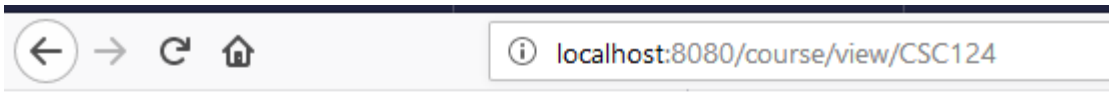
viewCourse.html tersebut akan mencetak course model dan untuk mencetak students dari course tersebut maka dibutuhkan iterasi untuk mengunjungi setiap isi dari list students.

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

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

            </li>
        </ul>
    </body>
</html>
```

**Hasil**




**ID = CSC124**

**Name = SDA**

**SKS = 3**

**Mahasiswa yang mengambil**

- 124-Glenn