Pada tutorial 5 ini saya belajar cara memetakan hasil query ke class model yang diinginkan. Selain itu juga saya belajar bahwa kita dapat memetakan hasil query ke dalam attribut pada class model menggunakan mapper juga(di kasus ini, pada saat students diisi, akan dipetakan hasil query selectCourses ke dalam students).

Pada bagian Latihan pertama, saya diminta untuk membuat viewall student dengan course yang mereka ambil. berikut beberapa method yang saya ubah:

1. Pada mapper

```
@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 ini kan melakukan query yang akan mencari seluruh student yang ada pada database lalu memasukkan nilai-nilai hasil query ke dalam studentModel. Pada method ini juga akan dimasukkan list of courses yang diambil student tersebut dengan cara selectCourses yang merupakan method yang sebelumnya(pada tutorial) sudah dibuat. Method ini akan mencari course apa saja yang diambil student dengan npm yang diminta pada database.

```
@Select("select course.id_course, name, credits " +
        "from studentcourse join course " +
        "on studentcourse.id_course = course.id_course " +
        "where studentcourse.npm = #{npm}")
List<CourseModel> selectCourses (@Param("npm") String npm);
```

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

Pada view-all.html akan ditambahkan list yang akan melakukan looping courses yang dimiliki student dan akan menampilkan course name dan juga course credits.

Pada latihan kedua, siswa diminta untuk menampilkan course yang diminta beserta peserta yang mengikutinya. Method yang saya buat adalah sebagai berikut:

1. Pada Mapper

```
@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 = "credit", column = "credit"),
        @Result(property = "students", column = "id_course", javaType = List.class, many = @Many(select="selectAllStudentInCourse"))
})
CourseModel selectCourse (@Param("id_course") String id_course);

@Select("select course.id_course, student.npm, student.name from course " +
        "join studentcourse on studentcourse.id_course = course.id_course " +
        "join student on studentcourse.npm = student.npm " +
        "where course.id_course = #{id_course}")
List<StudentModel> selectAllStudentInCourse(@Param("id_course") String id_course);
```

pada mapper class, saya membuat 2 method yang berhubungan. Method pertama selectAllStudentInCourse yang akan menjalankan query untuk mencari mahasiswa yang mengambil course yang dicari. Hasil dari method ini adalah list student yang mengambil dan akan digunakan oleh method lain. Method selanjutnya adalah selectcourse yang akan mencari course sesuai dengan id_course yang diminta dan memetakan data-data hasil querynya ke courseModel. Pada method ini juga akan dipetakan hasil query method selectAllStudentInCourse ke atribut students pada courseModel.

2. Pada studentInterface dan studentServiceDatabase

```
CourseModel selectCourse (String id);
```

```
@Override
public CourseModel selectCourse(String id){
    log.info("masuk ke select course");
    return studentMapper.selectCourse(id);
}
```

Pada interface studentInterface akan ditambahkan method selectCourse yang akan dioverride pada studentServiceDatabase. Method ini digunakan untuk memanggil mapper selectStudent yang sebelumnya sudah dijelaskan dan akan mengembalikan courseModel.

3. Pada studentController

Pada studentController, saya membuat method viewCourse yang memiliki parameter Model dan juga id yang diambil dari path. Method ini akan melakukan method selectCourse yang terdapat pada interface studentInterface dan disimpan di dalam variable bernama course. Jika course ditemukan, maka akan ditambahkan var course ke dalam model dan mereturn ke view-course.html dan jika tidak ditemukan akan mereturn ke page not-found.

```java
@RequestMapping("/course/view/{id}")
public String viewCourse(Model model, @PathVariable("id") String id){
    CourseModel course = studentDAO.selectCourse(id);
    if(course != null){
        model.addAttribute( attributeName: "course", course);
        return "view-course";
    }
    else {
        model.addAttribute( attributeName: "npm", id);
        return "not-found";
    }

}
```

4. Pada view-course.html

html ini ditambahkan untuk menampilkan data course dan juga siapa saja yang mengambil course terserbut.

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>View Course</title>
</head>
<body>
<h3 th:text="'ID = ' + ${course.id_course}">Course ID</h3>
<h3 th:text="'Name = ' + ${course.name}">Course Name</h3>
<h3 th:text="'SKS = ' + ${course.credits}">Course Credit</h3>
<h3>Mahasiswa yang mengambil</h3>
<ul th:each="student,iterationStatus: ${course.students}">
    <li th:text="${student.npm} + ' - ' + ${student.name}" >
        Nama kuliah-X SKS
    </li>
</ul>
</body>
</html>
```