Sandar Klinton – 1306404720
APAP - A
Writeup Tutorial 5

Yang didapatkan dari Tutorial 5 kali ini adalah:

- Mengetahui bila kita bisa mengubah hasil output dari database tidak dari controller, melainkan lewat DAO dengan annotations @Results dan @Result.
- Cara memodifikasi objek model sesuai dengan relasi database 1-N atau M-N.

Method yang dibuat & diubah pada mengubah Select All Students

1. Menambahkan method selectAllStudents dengan @results

```
@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 ();
```

@Results berguna untuk mengkonfigurasi hasil dari query yang didapatkan, sehingga hasil tersebut dapat sesuai dengan permintaan. Pada line terakhir, kita mengassign sebuah variabel pada studentmodel bernama courses, dan diambil berdasarkan kolom npm pada database, dan tipe dari result adalah list, dan maksud many disitu adalah kita mengambil secara one to many relationship menggunakan metode selectCourses yang telah ada di bawah. Students dan courses memang mempunyai many to many relationship, namun metode pengambilannya tetap one to many.

2. Menambahkan *code* untuk menampilkan course pada viewall.html

```
    <h1>All Students</h1>

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

Tidak berbeda dengan view.html, bila students.course = null maka dia tidak akan mencetak course yang diambil.

Method dan tahapan yang dibuat pada Latihan Menambahkan View pada Course

1. Mengcopy requestmapping /student/view/npm beserta methodnya dan mengganti apa yang kira kira akan dibutuhkan.

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

2. Karena pada studentDAO belum ada selectCourse(String id), maka kita buat method selectCourse yang akan mengembalikan course beserta student yang mengambil Course tersebut.

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

@Select("select * from course where id_course = #{id}")
@Results(value= {
        @Result(property="idCourse", 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="selectStudentsCourse"))
})
CourseModel selectCourse (@Param("id") String id);
```

   Dibuat sama saja seperti selectStudent yang baru, agar tidak memakan waktu lama. Student.course yang akan dimasukkan ke dalam course.student tidak perlu ada, sehingga dapat mempercepat waktu pemrosesan.

3. Membuat courseModel selectCourse(id) pada StudentService dan StudentService Database.

   Pada StudentService:

```
    void deleteStudent (String npm);

    void updateStudent (StudentModel student);


    CourseModel selectCourse(String id);
}
```

   Pada StudentServiceDatabase:

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

4. Menambahkan halaman not found untuk Course dan view untuk course not-foundCourse.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="'Course = ' + ${id}">Course ID</h3>
    </body>
</html>
```

viewCourse.html

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

        <h3>Mahasiswa yang Mengambil</h3>
        <ul th:each="students,iterationStatus: ${course.students}">
            <li th:text="${students.npm} + '-' + ${students.name}">
                Nama kuliah-X SKS
            </li>
        </ul>
    </body>
</html>
```

5. Bukti bila fitur bekerja