TUTORIAL 5 WRITE UP

Pada tutorial kali ini saya kurang lebih mempelajari hal yang sama seperti tutorial kemarin, dan juga saya mulai menemukan *pattern* dari pengerjaan proyek kode sumber menggunakan SpringBoot. Saya juga kembali mempelajari cara membuat *relationship* dan mengimplementasikannya sampai ke *view*. Saya juga mendapat motivasi untuk lebih semangat belajar hal lain yang saya sukai agar usai kuliah nanti, pekerjaan saya bukanlah membuat kode sumber.

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 ();
```

```
@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);
```

Pada method ini saya menggunakan kembali anotasi @Results dan memanggil method selectCourses yang akan menampilkan course mahasiswa.

Menambahkan View pada 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);
```

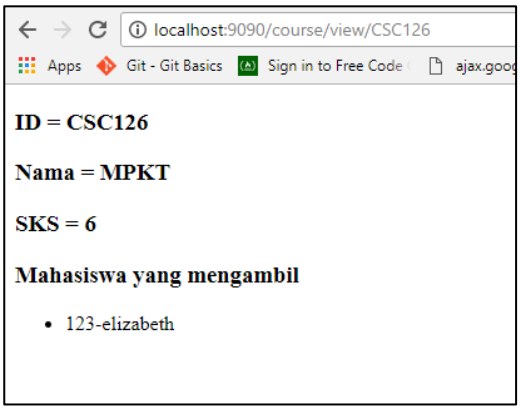
```java
@Select("select student.npm, name " +
        "from studentcourse join student " +
        "on studentcourse.npm = student.npm " +
        "where studentcourse.id_course = #{id_course}")
List<StudentModel> selectStudents (@Param("id_course") String id_course);
```

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

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

```
APAPTutorial04/src/main/java/com/example/controller/StudentController.java
<!DOCTYPE html>
<html>
    <head>
        <title>View Course</title>
    </head>
    <body>
        <h3 th:text="'ID = ' + ${course.id_course}">Course ID</h3>
        <h3 th:text="'Nama = ' + ${course.name}">Course Name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">Course Credits</h3>

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



Pertama di StudentMapper saya membuat method selectCourse dan selectStudents yang harusnya mengembalikan course yang dipilih oleh mahasiswa tertentu. Kemudian membuat interface-nya di StudentService dan tidak lupa meng-override di StudentServiceDatabase. Lalu di kelas controller menghubungkan antara model dengan view. Terakhir membuat view. Dalam pengerjaan nomor ini, jujur saya melakukan banyak kesalahan kecil seperti lupa import dan sering tertukar urutannya (tadinya saya mengerjakan method untuk kelas controller dulu, terus bingung lho kok ini error nya banyak ya).