

1. Untuk membuat selectAllStudent sesuai dengan format yang baru, pertama saya pada Controller saya tidak merubah apapun. Kurang lebih seperti tutorial sebelumnya

```
@RequestMapping("/student/viewall")
public String view (Model model)
{
    List<StudentModel> students = studentDAO.selectAllStudents ();
    model.addAttribute ( "students", students);

    return "viewall";
}
```

Begitu pula ketika service berjalan. Tidak ada syntax yang diubah.

```
@Override
public List<StudentModel> selectAllStudents ()
{
    log.info ("select all students");
    return studentMapper.selectAllStudents ();
}
```

Dan pada mapper, saya merubah dao dari student tersebut terlebih dahulu dikarenakan ditambahkannya atribut list of matkul pada setiap student. Kurang lebih seperti selectStudent pada tutorial kali ini.

```
@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 ();
```

Setelah itu kembali menuju controller dan dikirim ke viewall.html. Pada view all saya menambahkan string list matkul yang telah diambil pada mapper sebelumnya.

```
studentController.java x viewall.html x course-not-found.html x StudentMapper.java x student [eaap@
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>View All Students</title>
  </head>
  <body>
    <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>

      <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>

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

      <hr/>
    </div>
  </body>
</html>
```

2. Untuk view course, pertama-tama saya membuat controller terlebih dahulu untuk handle jika routingsnya /course/view/{id_course}. Setelah itu menggunakan pathvariable untuk mengambil data pada url dan memasukkan pada objek coursemodel menggunakan studentmapper select course

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

    if (course != null) {
        model.addAttribute ( S: "course", course);
        return "view-course";
    } else {
        model.addAttribute ( S: "npm", id);
        return "course-not-found";
    }
}
```

Kedua masuk ke service dimana ia akan direfer menuju dao dari course tersebut. Saya menambahkan service baru baik di interface maupun classnya sehingga ia akan menjalankan query untuk mencari course


```
@Override
public CourseModel selectCourse(String id) { return studentMapper.selectCourse(id); }
}
```

Ketiga pada mapper ia akan mengambil data pada course melalui data yang telah di passing dari id_course yang telah dimasukkan. Kurang lebih seperti select student dimana ia juga meloop student untuk mencari list of student yang mengambil matkul tersebut.

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

Dan yang terakhir ia akan kembali ke controller dan akan mengecek apakah ada atau tidak. jika tidak ada maka akan dilempar ke halaman not found dan jika berhasil masuk ke halaman view-course. Pada halaman course kurang lebih seperti view student namun dengan kondisi dimana halaman tersebut menampilkan course apa yang dicari, data data mengenai course tersebut berikut dengan siswa yang mendaftar pada matkul tersebut.

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