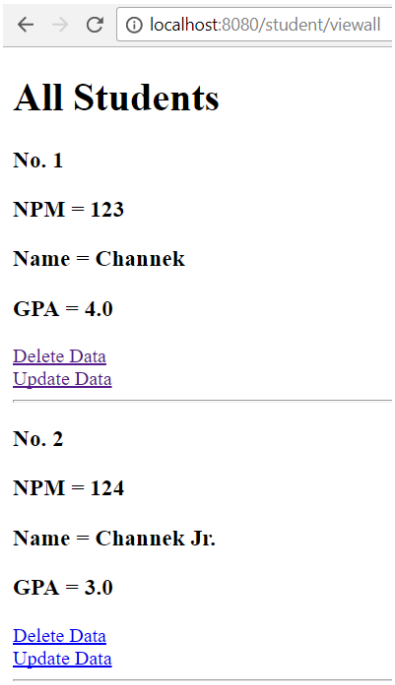
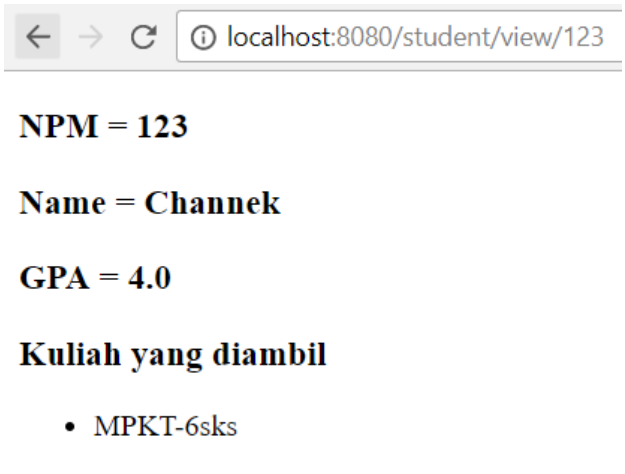
Tuturial

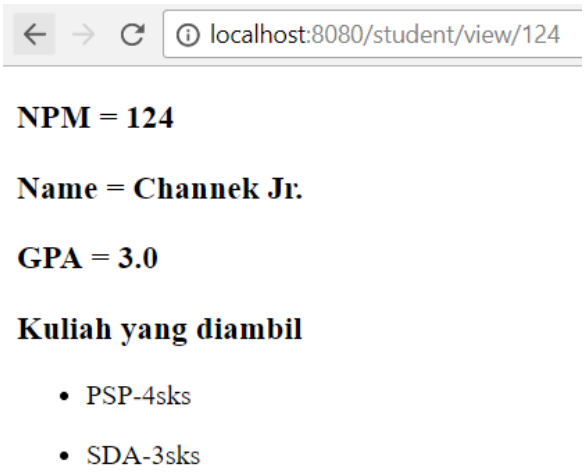
Data:



NPM: 123



NPM: 124



Latihan

1. Ubah method **selectAllStudents** pada kelas **StudentMapper** agar halaman *viewall* menampilkan semua *student* beserta daftar kuliah yang diambil.

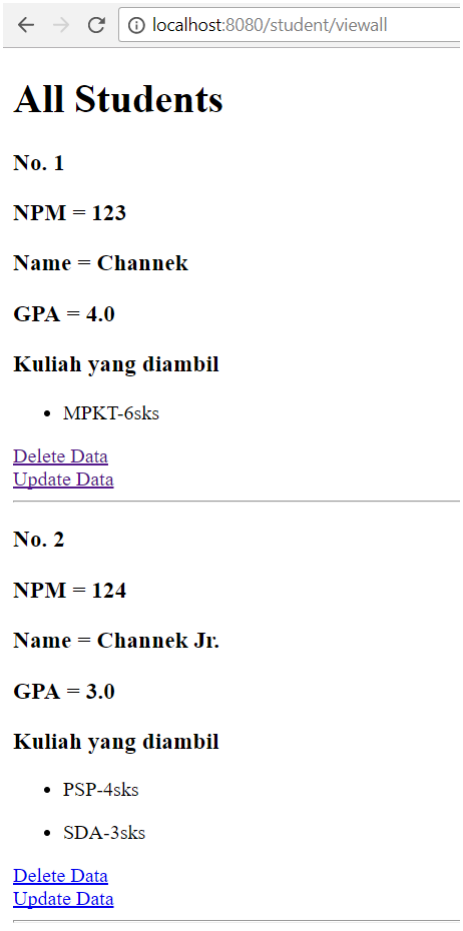
   a. Pada studentMapper menambahkan @Results sama seperti pada method selectStudent yang berfungsi menghubungkan method selectAllStudent dengan method selectCourses agar saat melakukan select All maka variabel List<Course> juga akan terisi. Jadi List Course akan berisi data-data yang diambil dari hasil @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();
```

b. Menambahkan view pada viewall.html untuk menampilkan list kuliah yang diambil oleh mahasiswa dengan cara melakukan iterasi pada course berdasarkan npm pada coursestudent dan mengambil data pada course kemudain ditampilkan, sama seperti view pada selectstudent

```
<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 />
```

c. Hasil:



2. Buatlah view pada halaman http://localhost:8080/course/view/{id} untuk Course sehingga dapat menampilkan data course beserta Student yang mengambil.

a. Menambahkan method interface selectCourse pada StudentService berfungsi mendefinisikan method-method apa saja yang dapat dilakukan untuk memanipulasi kelas Course

```
CourseModel selectCourse(String id_course);
```

b. Menambahkan method di StudentServiceDatabase yang diimplementasikan dari StudentService berfungsi untuk melakukan perintah yang ada di studentMapper yaitu memasukan data ke dalam database

```
@Override
public CourseModel selectCourse(String id_course) {
    log.info("select course with id_course {}", id_course);
    return studentMapper.selectCourse(id_course);
}
```

c. Membuat method mapper yang berfungsi mencari data course yang ada di database, method tersebut terdiri dari selectStudentCourse dimana mencari siswa-siswa yang mendaftar pada sebuah course dengan menggunakan query select join. Yang kedua method selectCourse dimana mencari course yang sudah ada terdiri dari id, name, credit

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

@Select("select id_course, name, credits from course where id_course = #{id_course}")
@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 = "selectStudentCourse"))
})
CourseModel selectCourse (@Param("id_course") String id_course);
```

d. Membuat method controller dengan path sebuah id_course dengan mencari course dalam database dari courseModel kemudian ditampilkan dengan viewcourse jika ada dan jika tidak ada not-found-course

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

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

e. Membuat view dari controller selectcourse ketentuan jika course ada, dengan menampilkan course dan student yang mengambil course tersebut

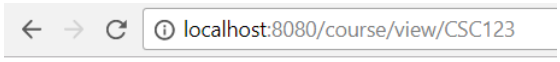
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>View Course by ID</title>
</head>
<body>
    <h3 th:text="'ID = ' + ${course.idCourse}">ID Course</h3>
    <h3 th:text="'Nama = ' + ${course.name}">Course Name</h3>
    <h3 th:text="'SKS = ' + ${course.credits}">Course Credits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

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

f. Membuat view not-found-course dari controller selectcourse ketentuan jika course tidak ada, dengan menampilkan id course

```
<!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 = ' + ${course}">ID Course</h3>
    </body>
</html>
```

g. Hasil:



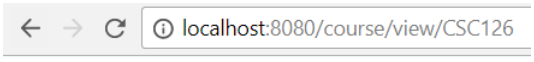
ID = CSC123

Nama = PSP

SKS = 4

Mahasiswa yang mengambil

- 124 - Channek Jr.



ID = CSC126

Nama = MPKT

SKS = 6

Mahasiswa yang mengambil

- 123 - Channek