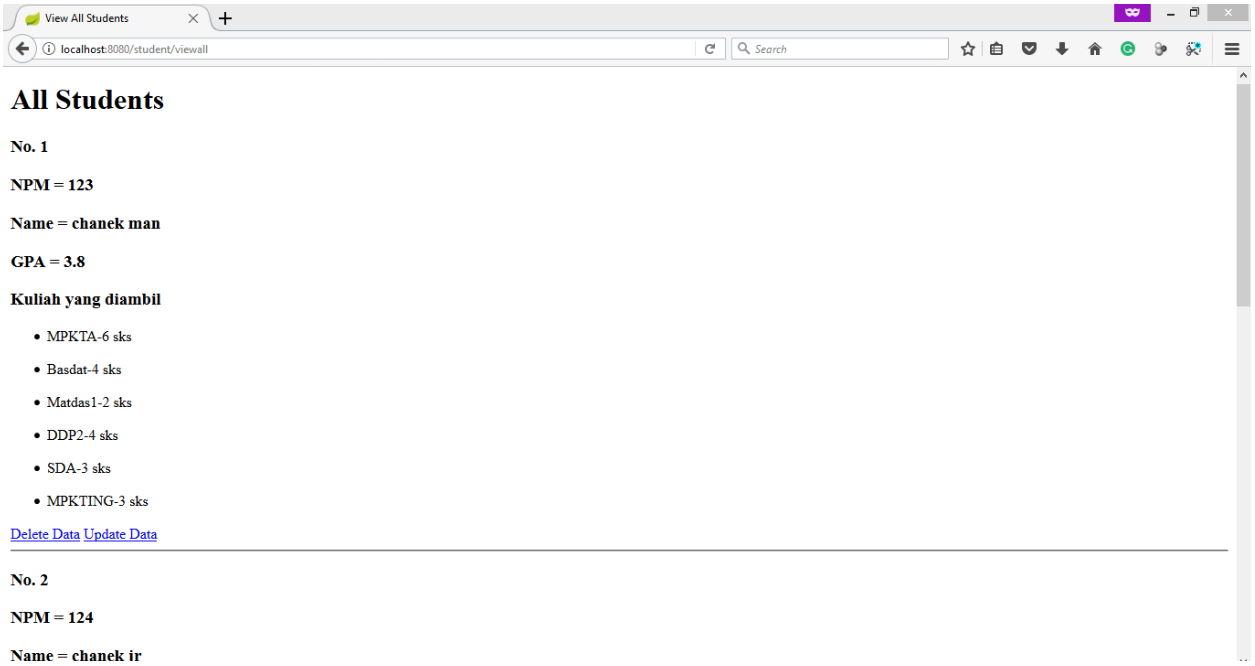
Latihan

1. Screenshot



All Students

No. 1

NPM = 123

Name = chanek man

GPA = 3.8

Kuliah yang diambil

- MPKTA-6 sks
- Basdat-4 sks
- Matdas1-2 sks
- DDP2-4 sks
- SDA-3 sks
- MPKTING-3 sks

Delete Data Update Data

No. 2

NPM = 124

Name = chanek ir

Method yang diubah

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

StudentMapper

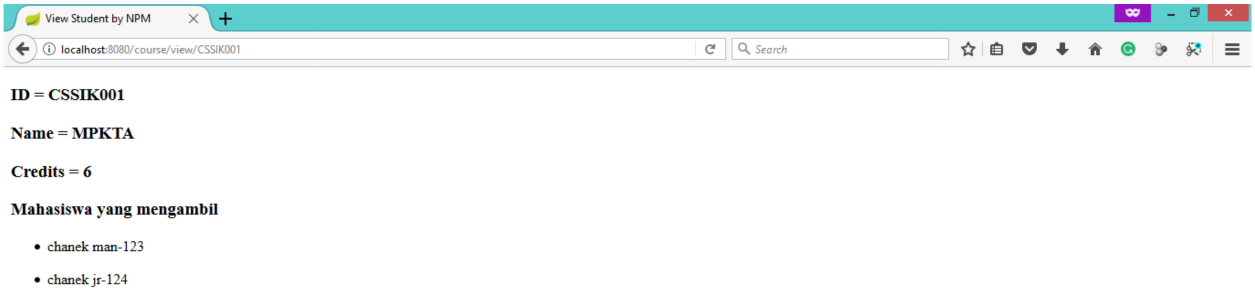
```
@Select("select npm, name, gpa from student where npm = #{npm}")
    @Results(value =
      {@Result(property="npm", column="npm"),
    @Result(property="name", column="name"),
    @Result(property="gpa", column="gpa"),
    @Result(property="courses", column="npm",
    javaType=List.class,
    many=@Many(select="selectCourses"))
```

```
      })
    StudentModel selectStudent (@Param("npm") String npm);


 @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);
```

Penjelasan

Jadi pertama itu ditambahkan ke List untuk courses yang diambil student. Caranya pakai many. Dibikin method buat ngelist per siswacoursenya apa. Jadi di select dulu dengan join antara studentcourse dan course, dicari yang sesuai dengan npm. Itu di method selectCourses. Terus dimasukin ke list dengan cara pakai Many. Jadi pas di htmlnya itu diiterasi listnya itu buat dicetak. Di kelas StudentModel juga ditambah list courseny itu sebagai attribut.

2. **Screenshot**



ID = CSSIK001

Name = MPKTA

Credits = 6

Mahasiswa yang mengambil

- chanek man-123
- chanek jr-124

Method yang diubah

View

```
<!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="'Name = ' + ${course.name}">Student Name</h3>
            <h3 th:text="'Credits = ' + ${course.credits}">Student GPA</h3>
            <h3>Mahasiswa yang mengambil</h3>
            <ul th:each="student, iterationStatus: ${course.students}">
                  <li th:text="${student.name} + '-' + ${student.npm}">
                        Nama kuliah-X SKS
                  </li>
            </ul>
    </body>
</html>
```

StudentMapper

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

    @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="selectStudentCourses"))
      })
    CourseModel selectCourse (@Param("id_course") String id_course);

    @Select("select id_course, name, credits from 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="selectStudentCourses"))
      })
    List<CourseModel> selectAllCourses ();
```

StudentController

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

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

StudentServiceInDatabase

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


    @Override
    public List<CourseModel> selectAllCourse ()
    {
        log.info ("select all course");
        return studentMapper.selectAllCourses ();
    }
```

Penjelasan

Jadi case ini mirip sama yang student bedanya dibalik sekarang mainnya itu CourseModelnya. Yang pertama dilakukan itu ditambahkan ke List untuk student yang ngambil courses. Caranya pakai many. Dibikin method buat ngelist per coursenya siswanya siapa. Jadi di select dulu dengan join antara studentcourse dan student, dicari yang sesuai dengan npm. Terus dimasukin ke list dengan cara pakai Many. Jadi pas di htmlnya itu diiterasi listnya itu buat dicetak. Di kelas CourseModel juga ditambah list studentnya itu sebagai attribut. Nah setelah itu harus ditambahin di StudentServiceInDatabase dan StudentService. Tujuannya adalah biar pas di Controller bisa dipakai methodnya. Nah setelah dibuat method di Controller yang mirip-mirip sama student/view/{npm}, bedanya yang ini course/view/{id}, nanti syntax di view htmlnya bisa jalan. Di run. Selesai.