Write Up Tutorial 5

Hal yang dipelajari dari tutorial ini adalah

Mengerti untuk menghubungkan, mengambil, dan menampilkan data dari dua tabel yang berbeda di database. Adapun anotasi Results yang digunakan untuk memetakan hasil dari query select ke class model. Variable property diisikan dengan nama variable yang ada pada class model dan variable column diisi dengan nama kolom hasil kueri di database.

SelectAllStudent

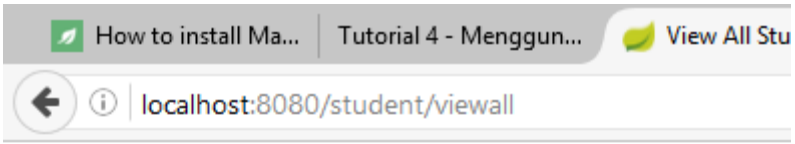
Pada latihan ini, saya mengubah method selectAllStudent dengan menambahkan method tersebut dengan anotasi @Results sebagaimana yang ditambahkan pada method selectStudent sebelumnya.

```
@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 ();
```

Kemudian saya juga menambahkan bagian yang akan menampilkan kuliah yang diambil di setiap iterasi student.

```
<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>
```

Sehingga saat daftar semua mahasiswa ditampilkan akan ditampilkan juga kuliah yang diambil mahasiswa tersebut.



All Students

Delete Data
Update Data

No. 1

NPM = 123

Name = Chanek

GPA = 3.2

Kuliah yang diambil

- MPKT-6 sks

Delete Data
Update Data

No. 2

NPM = 124

Name = Chanek Jr.

GPA = 3.4

Kuliah yang diambil

- PSP-4 sks

View pada Course

Pada saat membuat view pada course, yang pertama dilakukan adalah membuat bagaimana detail course dapat ditampilkan terlebih dahulu saat memasuki id-nya.

Membuat method selectCourse pada StudentMapper.

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

Membuat method selectCourse pada StudentService

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

Yang kemudian di override di kelas

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

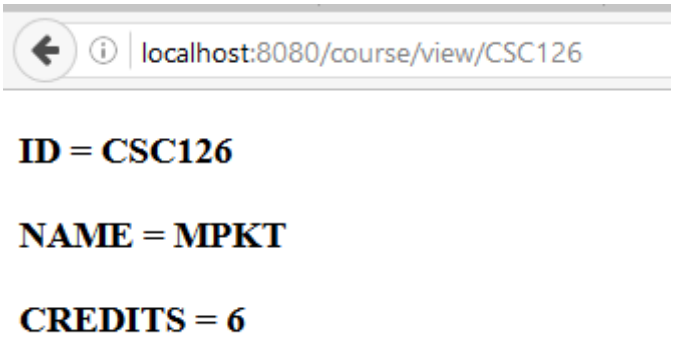
Membuat method viewCourse pada StudentController.

```
@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 "view-course";
    } else {
        model.addAttribute ("id_course", id_course);
        return "not-found-course";
    }
}
```

Dan membuat view-course.html

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

    </body>
</html>
```

Sehingga course yang ada di database dapat ditampilkan seperti ini



Setelah itu, saya membuat method untuk melihat siapa saja mahasiswa yang mengambil mata kuliah tersebut. Dengan melihat dari tabel studentcourse yang menyimpan hubungan dari tabel course dan student.

Membuat method selectStudents pada StudentMapper untuk mengembalikan List student yang mengambil course tertentu.

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

Kemudian menambahkan anotasi result dengan menyesuaikan variabel property dan column-nya.

```
@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);
```

Kemudian saya juga menambahkan bagian yang akan menampilkan mahasiswa yang mengambil di setiap iterasi course pada view-course.html.

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

Sehingga saat dijalankan http://localhost:8080/course/view/CSC124 akan tampil seperti ini.