Yang saya dapat belajar pada tutorial kali ini adalah belajar mengakses dan mengambil data-data yang terdapat pada database.

Latihan View All

```
@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 ();
```

Di method selectAllStudents pada StudentMapper diganti sehingga setiap student yang dipilih bisa ditampilkan courses yang diambilnya juga yaitu dengan menambahkan semua tersebut ke dalam List of StudentModel.

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>View All Students</title>
5      </head>
6      <body>
7          <h1>All Students</h1>
8
9          <div th:each="student,iterationStatus: ${students}">
10             <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
11             <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12             <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13             <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14
15             <h3>Kuliah yang diambil</h3>
16             <ul th:each="course,iterationStatus: ${student.courses}">
17                 <li th:text="${course.name} + '-' + ${course.credits} + ' sks'" >
18                     Nama kuliah-X SKS
19                 </li>
20             </ul>
21
22             <a th:href="'/student/delete/' + ${student.npm}" > Delete Data</a><br/>
23             <a th:href="'/student/update/' + ${student.npm}" > Update Data</a><br/>
24             <hr/>
25         </div>
26     </body>
27 </html>
```

Setelah itu ditambahkan juga iterasi untuk mengambil course yang sudah diambil pada file viewall.html dengan mengambil course name dan course creditnya sehingga bisa ditampilkan.

Latihan View Course

```
@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);
```

Pada StudentMapper pertama-tama dibuat method selectStudents yang mengambil semua student yang berdasarkan pada id course yang diminta.

```
@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, ditambahkan juga method selectCourse yang mengambil semua data-data yang ada pada course seperti id_course, name, dan credits.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <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
            </li>
        </ul>
    </body>
</html>
```

Setelah itu ditambahkan file viewCourse.html yang menampilkan course id, name, dan credits dan juga semua student yang mengambil course tersebut dengan iterasi.

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

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

Kemudian, pada StudentController ditambahkan method viewCourse yang didalamnya akan memanggil method selectCourse yang sudah dibuat kemudian jika ada course yang ditemukan maka akan ditampilkan file viewCourse.html, jika tidak maka akan ditampilkan file error yaitu course-not-found.html.