LATIHAN

1. Ubah method selectAllStudents pada kelas StudentMapper agar halaman viewall

menampilkan semua student beserta daftar kuliah yang diambil.

Contoh tampilan:

StudentMapper

```
@Mapper
public interface 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 ();
```

Halaman viewall.html

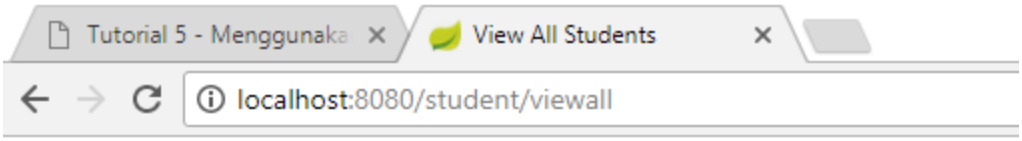
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All Students</title>
    </head>
    <body>
        <h1>All Students</h1>

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


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

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

Tampilan



Name = Nabilah

GPA = 3.9

Delete Data
Update Data

Kuliah yang diambil

- MPKT-6 sks

No. 2

NPM = 124

Name = bila

GPA = 3.99

Delete Data
Update Data

Kuliah yang diambil

- PSP-4 sks
- SDA-3 sks

---

2. Buatlah view pada halaman http://localhost:8080/course/view/{id} untuk Course

sehingga dapat menampilkan data course beserta Student yang mengambil.

1. Hal apa saja yang anda pelajari dari tutorial ini

Di tutorial kali ini, saya menjadi tahu tentang relasi pada database yang diimplementasikan pada *MVC Programming,* contohnya saat menggunakan *join*.

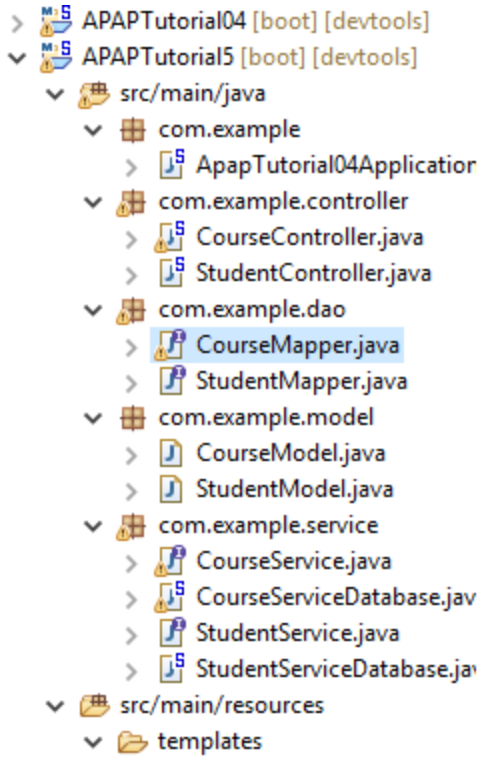
Saya juga belajar cara memetakkan hasil *query* di *class Model* yang ada pada MVC. Dimana *property* diisi dengan nama *variable* yang ingin dituju, dan *column* diisi dengan hasil *query* dari *database*.

PENJELASAN METHOD

1. Pada method selectAllStudents(), saya menambah @Results seperti pada selectStudent(String npm) sehingga courses setiap students yang ada pada List<studentModel> dapat terisi.

   Selain itu saya mengubah pada *viewall.html* sehingga terdapat iterasi untuk *courses* setiap *students* yang akan diiterasikan pada *view* tersebut.

2. Untuk soal ke dua, saya membuat Course*Mapper, CourseService, dan CourseServiceDatabase* baru.



Pada mapper yang baru, buat method baru bernama *selectCourse(String id).*
Untuk mengisi List<StudentModel> students pada course yang di-select dan mengisi bagian "Mahasiswa yang mengambil" pada view, diperlukan method yang mirip dengan selectCourses yang telah diberikan pada tutorial

```
@Mapper
public interface CourseMapper
{
    @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="selectStudents"))
    })
    CourseModel selectCourse (@Param("id_course") String id_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);
}
```

Implementasikan *interface – interface* yang dibutuhkan pada *CourseService* dan *CourseServiceDatabase*

CourseMappe... StudentMapp... CourseServic... CourseContr... viewCourse.html StudentCont...

```
1  package com.example.service;
2
3  import java.util.List;
4
5  import com.example.model.CourseModel;
6  import com.example.model.StudentModel;
7
8  public interface CourseService
9  {
10     CourseModel selectCourse (String id_course);
11 }
12
```

```
package com.example.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.dao.CourseMapper;
import com.example.dao.StudentMapper;
import com.example.model.CourseModel;
import com.example.model.StudentModel;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Service
public class CourseServiceDatabase implements CourseService
{
```

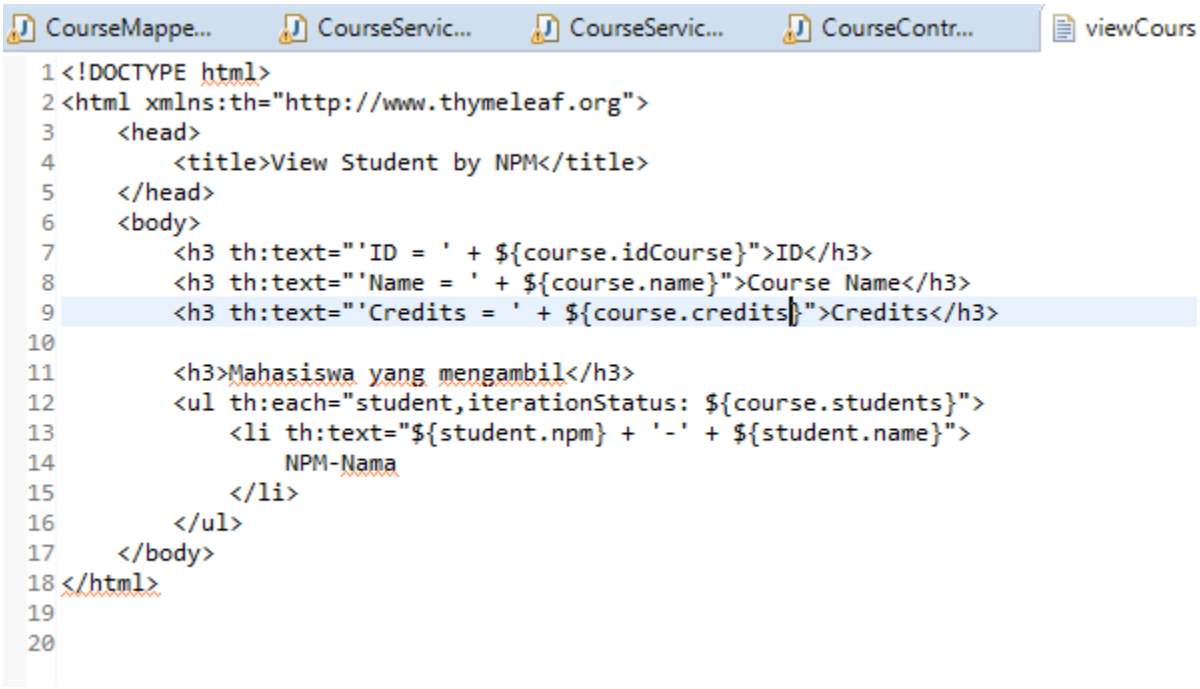
Lalu buat *controller* baru untuk CourseService, pada *controller*, mengimport *CourseService* dengan anotasi *@Autowired*

Lalu, membuat *method* baru untuk *view* yang dibutuhkan:

```java
@Controller
public class CourseController
{
    @Autowired
    CourseService courseDAO;

    @RequestMapping("/course/view/{id_course}")
    public String viewCourse (Model model,
            @PathVariable(value = "id_course") String id_course)
    {
        CourseModel course = courseDAO.selectCourse (id_course);
        if (course != null) {
            model.addAttribute ("course", course);
            return "viewCourse";
        } else {
            model.addAttribute ("id_course", id_course);
            return "not-found-course";
        }
    }
```

Terakhir, membuat *view* template untuk *viewCoursePath* dengan nama *view-course.html* dan *not-found_course.html* sesuai dengan *return value* pada *controller*:



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

```