A. Ringkasan dari materi yang Saya pelajari pada tutorial ini

Pada tutorial kali ini saya belajar lebih banyak bagaimana menggunakan database pada program spring boot. Lalu juga kembali menggunakan library eksternal yaitu mybatis dan lombok. Pada tutorial kali ini sudah menggunakan 3 tabel pada database yaitu student, course dan studentcourse yang menghubungkan kedua tabel.

Pada tutorial kali ini, juga kembali mengulangi step-step yang ada pada tutorial sebelum-sebelumnya, yaitu untuk class course. Yaitu menambah model untuk course, lalu membuat mapper untuk course, lalu membuat service untuk course dan membuat controller untuk course. Hal ini untuk bisa mengerjakan latihan no. 2 yaitu menampilkan view course.

B. Penjelasan Method-Method

1. Method yang saya ubah pada Latihan Merubah SelectAllStudents

   Yang saya rubah adalah pada StudentMapper.java yang tadinya hanya query biasa untuk select npm, nama, dan gpa biasa dirubah menjadi seperti yang dicontohkan di method selectStudent. Berikut method selectAllStudent setelah dirubah :

```
@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 ();
```

   Pada method tersebut menggunakan anotasi results untuk memetakan hasil dari query ke class courseModel. Untuk variabel courses kembali dilakukan pemanggilan ke selectCourses seperti pada method selectStudent.

   Selain merubah pada StudentMapper.java, juga dilakukan edit pada viewall.html yaitu sebagai berikut :

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

   Yaitu ditambahkan list untuk menampilkan course-course yang diambil seorang student.

2. Method yang saya buat pada Latihan Menambahkan View pada Course

Untuk membuat view pada course step-step yang dilakukan mirip seperti step-step ketika membuat view pada student, tetapi saya mengerjakan dengan urutan yang agak berbeda yaitu pertama membuat model (yang dilakukan di tutorial ini) lalu mulai membuat controller untuk course yaitu CourseController.java, dimana ia mengatur untuk routing "/course/view" dan "/course/view/{id}. Berikut gambarnya :

```
public class CourseController {

    @Autowired
    CourseService courseDAO;

    @RequestMapping("/course/view")
    public String viewCourse (Model model,
                        @RequestParam(value = "id", required = false) String id)
    {
        CourseModel course = courseDAO.selectCourse (id);

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

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

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

Selanjutnya menambahkan halaman view-course.html dan not-found-course, yang fungsinya untuk tampilan course.

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

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

Pada halaman view-course tersebut ditampilkan id, nama, dan sks course dan list mahasiswa yang mengambil course tsb.

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

Pada halaman not-found-course tsb menampilkan bahwa suatu course tidak ditemukan.

Setelah menambahkan kedua halaman tersebutmembuat file CourseService.java yang berupa interface dan CourseServiceDatabase.java yang mengimplementasi CourseService.

```
package com.example.service;

import com.example.model.CourseModel;

public interface CourseService
{
    CourseModel selectCourse(String id);
}
```

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

    @Autowired
    private CourseMapper courseMapper;

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

Pada service tersebut hanya terdapat 1 method yaitu selectCourse untuk memilih course yang ingin dilihat detailnya.

Lalu selanjutnya membuat mapper untuk course yaitu CourseMapper yang melakukan akses ke database dengan query. Yang melakukan select id_course, name, dan credits serta melakukan select student yang mengambil suatu course.

```java
@Mapper
public interface CourseMapper
{
    @Select("select id_course, name, credits from course where id_course = #{id}")
    @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") String id);

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