Mohammad Izzat Raihan
1506689446

Tutorial 5

Membuat tabel database

Table	Action	Rows	Type	Collation	Size	Overhead
course	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16 KiB	-
student	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16 KiB	-
studentcourse	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16 KiB	-
3 tables	Sum	10	InnoDB	latin1_swedish_ci	48 KiB	0 B

Membuat Model untuk Course

CourseModel.java

```java
package com.example.model;

import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CourseModel {
    private String idCourse;
    private String name;
    private Integer credits;
    private List<StudentModel> students;
}
```

Lalu pada StudentModel.java ada penambahan

```java
package com.example.model;

import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class StudentModel
{
    private String npm;
    private String name;
    private double gpa;
    private List<CourseModel> courses;

}
```

Pada StudentController juga ada perubahan inisiasi

```java
StudentModel student = new StudentModel (npm, name, gpa, null);
studentDAO.addStudent (student);
```

Buat Method pada StudentMapper

Mohammad Izzat Raihan
1506689446

```
@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);
```

Mengubah method selectStudent menjadi

```
@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);
```

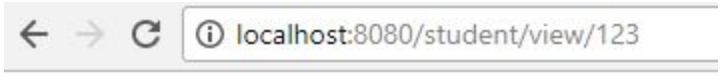
Pada view.html tambahkan

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Student by NPM</title>
    </head>
    <body>
        <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>
    </body>
</html>
```

Maka ketika dijalankan akan menghasilkan seperti berikut

Mohammad Izzat Raihan
1506689446



NPM = 123

Name = wowow

GPA = 3.0

Kuliah yang diambil

- MPKT-6 sks

LATIHAN

1. Ubah method selectAllStudents pada kelas StudentMapper agar halaman viewall menampilkan semua student beserta daftar kuliah yang diambil.
   Ubah method selectAllStudents pada StudentMapper

```
@Select("select * 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 ();
```

Method ini akan menselect database yang ada pada student. Lalu mengaitkan sesuai dengan property-property yang ada. Pada courses, karena diambil menggunakan method yang lain, maka column yang dipilih sesuai dengan kebutuhan method tersebut yaitu npm. Select tersebut akan mereturn list dengan isi semua student yang ada dalam database.

Lalu pada viewall.html diubah pula menjadi

Mohammad Izzat Raihan
1506689446

```
<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><br/>
    <hr/>
</div>
```

Ketika dijalankan akan menghasilkan

All Students

No. 1

NPM = 123

Name = wowow

GPA = 3.0

Kuliah yang diambil

- MPKT-6 sks

Delete Data Update Data

No. 2

NPM = 124

Name = A

GPA = 4.0

Kuliah yang diambil

- PSP-4 sks
- SDA-3 sks

Delete Data Update Data

Mohammad Izzat Raihan
1506689446

2. Buatlah view pada halaman http://localhost:8080/course/view/{id} untuk Course sehingga dapat menampilkan data course beserta Student yang mengambil.

   Buatlah method baru pada StudentService

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

   Lalu pada StudentServiceDatabase tambahkan

```
@Override
public CourseModel selectCourse (String id_course)
{
    return courseMapper.selectCourse(id_course);
}
```

   Buatlah sebuah interface bernama CourseMapper.java

```
@Mapper
public interface CourseMapper {
    @Select("select s.npm, s.name, s.gpa from student s, studentcourse sc where sc.npm = s.npm and id_course=#{id_course} ")
    @Results(value = {
       @Result(property="npm", column="npm"),
       @Result(property="name", column="name"),
       @Result(property="gpa", column="gpa")
       })
    StudentModel selectStudent (@Param("id_course") String id_course);

    @Select("select * 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="selectStudent"))
    })
    CourseModel selectCourse (@Param("id_course") String id_course);
}
```

CourseMapper ini akan membuat controller dapat menselect course yang dituju. Selain itu, juga membuat method selectCourse dapat menselect student yang mengambil course tersebut. Hal itu dilakukan dengan cara membuat method selectStudent yang menselect table student dan studentcourse dengan npm yang sama, lalu tampilkan id_coursenya.

Pada selectCourse, method ini akan mereturn model Course dengan id_course yang telah sesuai dengan hasil select yang dilakukan selectStudent. Karena, pada selectStudent telah disaring student yang mengambil course tersebut.

Buatlah CourseController, dan isi dengan seperti berikut

Mohammad Izzat Raihan
1506689446

```java
package com.example.controller;

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

@Controller
public class CourseController {
    @Autowired
    StudentService studentDAO;

    @RequestMapping("/course/view/{id_course}")
    public String viewPath (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";
        }
    }
}
```

Dapat dilihat bahwa, CourseController akan RequestMapping pada url /course/view/{id_course}. Setelah RequestMapping dijalankan, controller akan menselect course dengan StudentDAO dan CourseMapper yang telah dibuat sebelumnya. Lalu akan ada validasi terhadap id_course terlebih dahulu, ketika id_course ditemukan, maka akan mereturn view-course, jika tidak akan mereturn not-found-course. Untuk itu, diperlukan view-course.html dan not-found-course.html sebagai berikut

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

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

Dan

Mohammad Izzat Raihan
1506689446

```
<!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>
```

Lalu ketika dijalankan akan menghasilkan tampilan sebagai berikut

id_course = CSC123

Name = PSP

Credits = 4

Mahasiswa yang mengambil

- 124-A
- 123-wowow

Lesson Learned

Pada tutorial kali ini saya belajar bagaimana menggunakan database dan relasi pada Spring Boot. Terdapat tabel course dan relasinya dengan student (studentcourse). Hal itu memungkinkan aplikasi menampilkan course apa saja yang diambil oleh student, begitupun sebaliknya, siapa saja student yang sedang mengambil course tersebut.

Hal itu dikarenakan adanya select pada Mapper yang menggunakan @Result. @Result ini membutuhkan property yang sesuai pada variable yang ada pada Model, dan column yang sesuai dengan database. Dengan begitu, kita dapat mengaitkan satu tabel dengan tabel yang lainnya dengan query sql menggunakan springboot.