Julio Jaya Handoyo
1506757440
APAP – A

Hal yang saya pelajari dan dapatkan dari tutorial ini adalah cara menggunakan database dan relasi database dalam sebuah Project Spring Boot. Database yang digunakan pada tutorial ini terdapat penambahan entity(disbanding tutorial 4) sehingga terdapat relasi yang menghubungkan kedua entity. Ketika ada penambahan entity dalam Project Spring Boot, maka ada model baru yang harus ditambahkan. Selain itu, pada tutorial ini terdapat anotasi baru, yaitu @Result. Anotasi ini digunakan untuk memetakan hasil dari query select ke class model. Selanjutnya, ada variabel property dan column yang diisi dengan nama variabel di model dan nama kolom hasil kueri di database. Terdapat juga variabel javaType yang menentukan class yang akan dikembalikan dan variabel many yang menunjukkan method yang akan mengisi variabel courses.

LATIHAN

1. Saya menambahkan anotasi result, variabel javaType, dan many pada method selectAllStudents(). Hal ini bertujuan untuk memetakan hasil dari query select ke class model(dalam hal ini StudentModel). Selain itu, terdapat variabel many yang diisi selectCourses hal ini bertujuan untuk mengetahui courses yang diambil mahasiswa yang memiliki npm tertentu.

```
@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 ();
```

2. Saya menambahkan method selectStudentCourse dan selectCourse pada class StudentMapper. Method selectStudentCourse ini berfungsi untuk mencari semua mahasiswa yang mengambil mata kuliah tertentu. Method selectCourse berfungsi untuk memilih mata kuliah yang ingin dilihat informasinya.

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

@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);

@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="selectStudentCourse"))
})
CourseModel selectCourse(@Param("id_course") String id_course);
```

Setelah itu, saya menambahkan method selectCourse pada StudentService. Method ini mencari course berdasarkan id course yang diminta(parameter).

```
public interface StudentService
{
    StudentModel selectStudent (String npm);


    List<StudentModel> selectAllStudents ();


    void addStudent (StudentModel student);


    void deleteStudent (String npm);

    void updateStudent(StudentModel student);

    CourseModel selectCourse (String idCourse);
}
```

Selanjutnya, saya mengimplementasikan method selectCourse di StudentServiceDatabase yang sudah di override dari StudentService. Disini terdapat proses debugging menggunakan logging.

```
    @Override
    public CourseModel selectCourse(String idCourse) {
        // TODO Auto-generated method stub
        log.info("select course with id{}",idCourse);
        return studentMapper.selectCourse(idCourse);
    }

}
```

Selanjutnya, saya menambahkan 2 view baru, yaitu view-course dan course-not-found. Jika course yang ingin dilihat ada di database maka akan menampilkan informasi dari course tersebut, seperti kode mata kuliah, nama mata kuliah, jumlah sks, dan mahasiswa yang mengambil mata kuliah tersebut.

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

    </body>
</html>
```

Sebaliknya, jika mata kuliah yang ingin dilihat tidak terdapat dalam database maka akan menampikan halaman course-not-found

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Course not found</title>
    </head>
    <body>
        <h1 th:text="'Course ' + ${id} + ' not found'">Course ID</h1>
    </body>
</html>
```

Terakhir, saya menambahkan method viewCourse pada controller. Menerima parameter berupa id dari course dan melakukan pengecekan terhadap course tersebut apakah ada di database atau tidak.

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