Irfin Handiliastawan
1506689704

WRITE-UP

Pada tutorial kelima kali ini, saya mempelajari hal baru terkait membuat relasi database dalam Project Spring Boot. Tutorial kali ini merupakan pengembangan dari tutorial keempat kemarin yang mempelajari cara menghubungkan database (phpMyAdmin) dengan project Spring boot. Pada tutorial kali ini saya menambahkan tabel baru pada database yang bernama course. Tabel course merepresentasikan course/mata kuliah. Setiap mahasiswa dapat mengambil suatu mata kuliah. Tabel mata kuliah memiliki atribut id course, name, dan credits. Selain itu terdapat juga tabel baru yang bernama studentcourse. Setiap mahasiswa dapat mengambil course dan juga sebaliknya. Oleh karena itu dibutuhkan tabel studentcourse yang menjadi relasi (penguhubung) antara kedua *entity* tersebut. Tabel studentcourse memiliki atribut npm mahasiswa yang mengambil suatu mata kuliah dan atribut id mata kuliahnya. Selain melakukan penambahan pada database, pada tutorial kali ini saya juga menambahkan class Course model pada Spring Boot. Hal tersebut mengakibatkan perubahan maupun penambahan terhadap beberapa method. Kemudian saya juga diajarkan cara untuk mencetak data dari database.

Pada tutorial kali ini, saya juga mendapatkan pengetahuan baru terkait anotasi @Result, variabel property, dan variabel column. Anotasi @Result digunakan pada interface studentMapper.java akan memetakan hasil dari query select ke class model. Variabel property diisi dengan nama variabel di suatu class , sedangkan variabel column diisi dengan nama kolom hasil kueri di database (parameter yang dibutuhkan). Selain itu, tutorial kali in juga mengingatkan saya terhadap beberapa kueri yang telah diajarkan pada mata kuliah basis data.

Penjelasan perubahan method SelectAllStudents

Method SelectAllStudents pada tutorial kali ini merupakan modifikasi dari method SelectAllStudents pada tutorial sebelumnya. Perbedannya adalah terdapat daftar mata kuliah yang diambil oleh mahasiswa. Algoritma dari method ini mirip dengan select student yang telah diajarkan pada tutorial ini sebelumnya. Pada intinya, method selectAll student mengambil data npm, nama, dan gpa dari tabel student serta nama course pada tabel course.

```
@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 ();
```

Seperti yang telah sampaikan sebelumnya, anotasi @Result digunakan untuk memetakan hasil dari query select ke class model. Pada gambar diatas, syntax tersebut akan mengambil data dari database dan menyimpannya ke class student model dalam bentuk list. Kemudian pada Property courses kita ingin mengisi variabel courses di class StudentModel. Variabel column diisi dengan npm karena kolom npm pada database akan dikirim menjadi parameter di method selectCourses.

Irfin Handiliastawan
1506689704

Variabel javaType menentukan class yang menjadi kembalian. Kemudian variabel many diset dengan method selectCourses.

```
@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);
```

Method selectCourses digunakan untuk mengambil data id_course, name course, dan credits course dengan melakukan join antara tabel studentcourse dan course berdasarkan npm seorang mahasiswa. Sehingga kedua method tersebut akan mengembalikan student model yang berisikan npm, name, gpa, dan list mata kuliah yang diambil oleh mahasiswa yang bersangkutan.

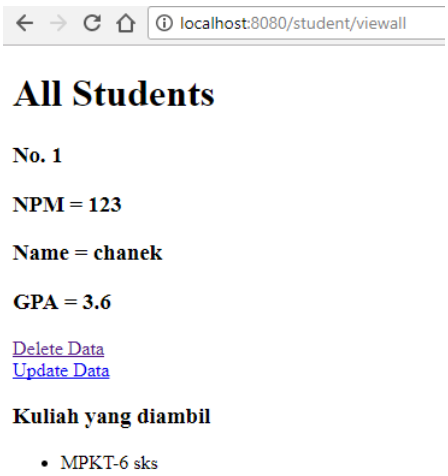
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View All Students</title>
    </head>
    <body>
        <h1>All Students</h1>

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

Untuk menampilkannya melalui html, maka dibutuhkan iterasi untuk mengunjung setiap student model. Selain itu, untuk mencetak course juga dibutuhkan iterasi untuk mengunjungi setiap isi dari isi coursemodel (courses).

Implementasi pada localhost:



Irfin Handiliastawan
1506689704

Penjelasan menambahkan View pada Course

Untuk menambahkan view pada course, kita harus mengubah perspektif dari student centric menjadi course centric. Secara garis besar caranya mirip dengan membuat view pada mahasiswa. Pada awal tutorial ini, saya membuat objek baru yang bernama courseModel yang memiliki atribut id_course, name, dan credits.

```
@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="selectStudentCourses"))
})

CourseModel selectCoursesStudent (@Param("id_course") String id_course);
```

Pertama kita harus membuat method selectCoursesStudent pada studentMapper yang memiliki paramater (id_course) yang berasal dari request Mapping. Method tersebut digunakan untuk mengakses data dari tabel course pada database dan menyimpannya ke course model. Kemudian untuk mendapatkan list dari students, method tersebut akan menggunakan bantuan dari method selectStudentCourses.

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

Method selectStudentCourses digunakan untuk mengambil data npm student dan nama student dengan melakukan join antara tabel studentcourse dan student berdasarkan id_course suatu mata kuliah. Sehingga kedua method tersebut akan mengembalikan course model yang berisikan id_course, name, credits, dan list studentmodel yang mengikuti mata kuliah tersebut.

Kemudian untuk melakukan request mapping, kita memerlukan method viewCourse pada class student controller.

```
@RequestMapping("/course/view/{id_course}")
public String viewCourse (Model model, @PathVariable(value = "id_course") String id_course)
{
    CourseModel course = studentDAO.selectCoursesStudent(id_course);
    if (course != null) {
        model.addAttribute ("course", course);
        return "viewCourse";
    } else {
        model.addAttribute ("id_course", id_course);
        return "not-found-2";
    }
}
```

Method tersebut memiliki parameter id_course yang didapat dari request Mapping. Kemudian Method tersebut akan membuat course model dan akan memanggil method selectCoursesStudent

yang telah kita buat sebelumnya dan berisikan parameter id_course. Jika objek course ada maka akan menampilkan halaman “viewCourse.html”. Jika tidak ada, maka akan menampilkan halaman “not-found-2.html”. Berikut ini isi dari halaman “viewCourse.html”.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by id course</title>
    </head>
    <body>
        <form th:object="${course}">
        <H3 th:text="'ID = ' + ${course.id_course}">Course id</H3>
        <h3 th:text="'Name = ' + ${course.name}">Course name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">Course credit</h3>
        </form>
     <h3>Mahasiswa yang mengambil</h3>
      <ul th:each="student,iterationStatus: ${course.students}">
        <li th:text="${student.npm} + '-' + ${student.name} " >

        </li>
      </ul>
    </body>
</html>
```

Html tersebut akan mencetak course model, untuk mencetak students dari course tersebut maka dibutuhkan iterasi untuk mengunjungi setiap isi dari list students.

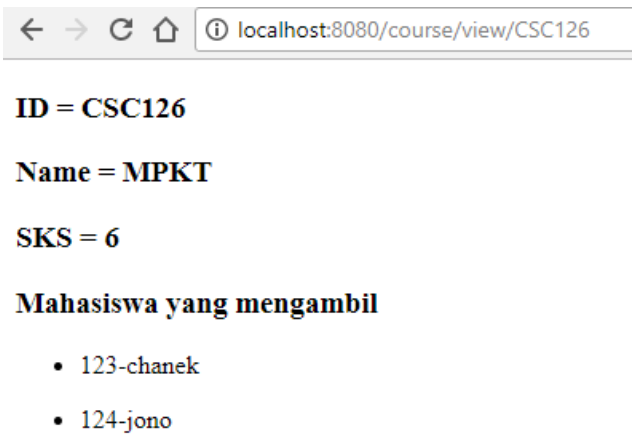
Berikut ini isi dari halaman “not-found-2.html”.

```
<!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}"></h3>
    </body>
</html>
```

Halaman tersebut akan menampilkan tulisan bahwa course dengan id course x tidak ditemukan

Implementasi pada localhost:

- Jika course tersebut ada



Irfin Handiliastawan
1506689704

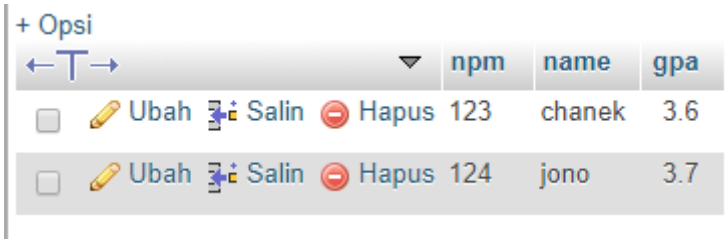
- Jika course tersebut tidak ada

Isi dari database :

- Database student course

npm	id_course
123	CSC126
124	CSC123
124	CSC124
124	CSC126

- Database student



- Database course