

Write-Up Tutorial 5: Menggunakan Database serta Relasi Database dalam Project Spring Boot

Ringkasan

Di dalam tutorial ini saya melanjutkan materi yang telah saya pelajari di tutorial 4, yaitu berhubungan dengan integrasi database pada project Spring Boot. Kali ini saya mempelajari cara mengimplementasikan relasi di dalam database.

Di dalam tutorial ini, selain membuat tabel student pada schema eaap, ditambahkan pula dua tabel baru: course dan studentcourse. Tabel course menyimpan data-data mengenai course yang diambil oleh student dan berisi id course, nama dan jumlah SKS course tersebut. Sementara itu, tabel studentcourse berfungsi sebagai relasi antara kedua tabel yang merepresentasikan mata kuliah yang diambil seorang mahasiswa dan mahasiswa yang mengikuti suatu mata kuliah.

Selanjutnya, dibuat model baru untuk course yang diikuti oleh student bernama CourseModel yang memiliki atribut id course, nama, SKS, dan daftar mahasiswa yang mengikuti mata kuliah. Selain itu, model StudentModel ditambahkan daftar course yang diikuti, sehingga memerlukan penyesuaian pada constructor yang terletak di StudentController. Pada StudentMapper ditambahkan method baru yaitu selectCourses yang mengembalikan list course yang diikuti seorang student. Terdapat perubahan pula pada method selectStudent dengan menambahkan anotasi @Results, @Results, dan @Many.

Di dalam tutorial ini saya kembali mempelajari query SQL untuk mencari data seperti mata kuliah yang diikuti oleh seorang mahasiswa dan mahasiswa yang mengikuti suatu mata kuliah. Selain itu, saya juga mempelajari cara mengimplementasikan relasi antar data dalam proyek Spring Boot.

Latihan mengubah SelectAllStudents

Dalam latihan mengubah selectAllStudents pada StudentMapper, terdapat implementasi method di beberapa class dan interface di dalam aplikasi.

Pada interface StudentMapper saya cukup menambahkan anotasi @Results pada tutorial menampilkan daftar mata kuliah yang diikuti seorang student, seperti di bawah ini.

```
@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 ();
```

Kemudian saya mengubah view pada viewall.html dengan menambahkan elemen dari view yang telah diubah ketika tutorial menampilkan daftar mata kuliah ke dalam div dengan th:each menjadi seperti berikut ini.

```
<h3 th:text="{coursemsg}">Kuliah yang diambil</h3>

<ul th:each="course,iterationStatus: {student.courses}">

    <li th:text="{course.name} + '-' + {course.credits} + ' sks'" >
```

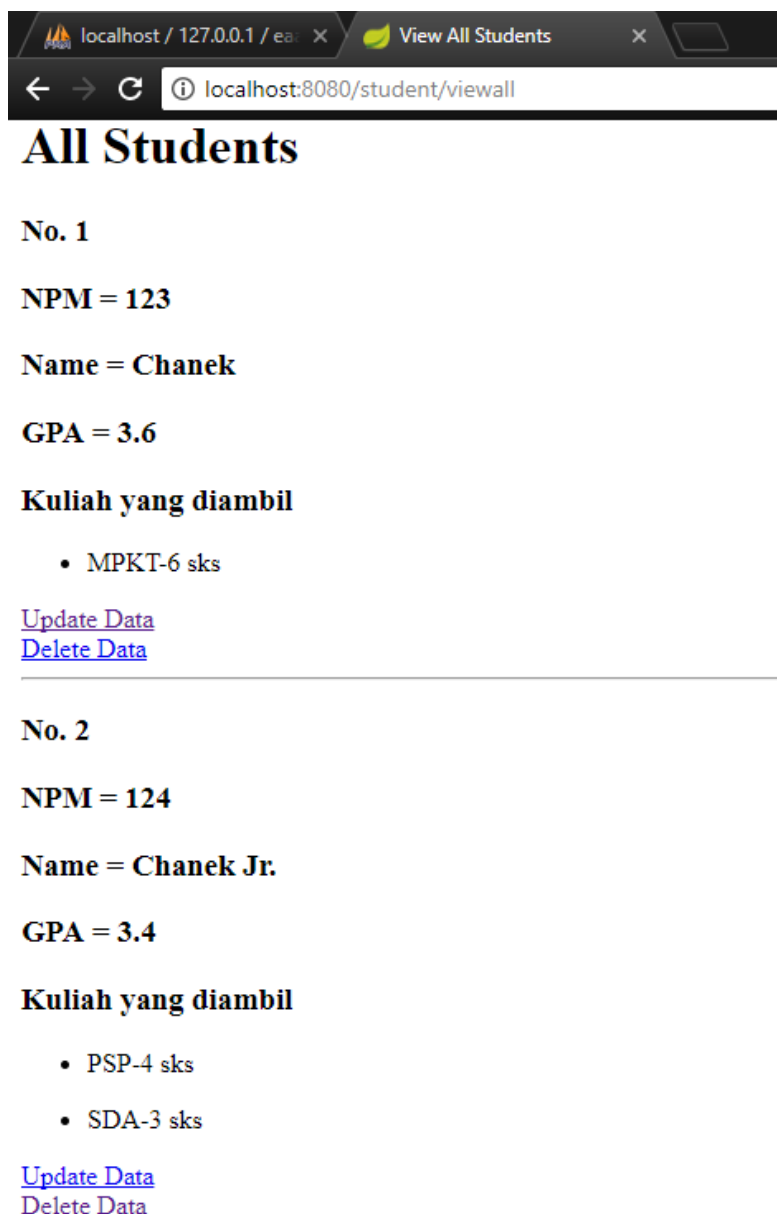
Sukrian Syahnel Putra
1306402066
ADPAP - B
Nama kuliah-X SKS

Kode di atas ditambahkan di bawah header yang menampilkan GPA para student. Kode di atas kurang lebih sama dengan kode pada view.html yang ditambahkan pada tutorial ini, akan tetapi terdapat atribut bernama coursemsg yang dimasukkan ke dalam Model melalui StudentController. Atribut ini berfungsi untuk menampilkan tulisan “Kuliah yang diambil” karena Thymeleaf menimpa html biasa yang menyebabkan tulisan tersebut tidak dapat muncul. Berikut potongan kode yang ditambahkan ke method view yang menampilkan daftar seluruh student pada kelas StudentController.

```
model.addAttribute("coursemsg", "Kuliah yang diambil");
```

Method tersebut ditambahkan sebelum method me-return “viewall”.

Hasil yang ditampilkan setelah mengimplementasi method ini adalah sebagai berikut.



localhost / 127.0.0.1 / ea View All Students X

localhost:8080/student/viewall

All Students

No. 1

NPM = 123

Name = Chanek

GPA = 3.6

Kuliah yang diambil

- MPKT-6 sks

[Update Data](#)
[Delete Data](#)

No. 2

NPM = 124

Name = Chanek Jr.

GPA = 3.4

Kuliah yang diambil

- PSP-4 sks
- SDA-3 sks

[Update Data](#)
[Delete Data](#)

Sukrian Syahnel Putra

1306402066

ADPAP - B

Latihan menambahkan View pada Course

Untuk latihan menambahkan view pada course, ada beberapa method yang ditambahkan dan diimplementasi pada beberapa class dan interface.

Pada StudentMapper saya menambahkan dua jenis method, yaitu selectCourse yang menerima parameter berupa ID course dan selectStudentsbyCourse yang menerima parameter berupa ID course. Kedua method yang ditambahkan adalah sebagai berikut.

```
@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="selectStudentsbyCourse"))
})
CourseModel selectCourse(@Param("id") String id);

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

Method pertama, yaitu selectCourse, berisi query yang mencari ID course, nama, dan SKS dari tabel mata kuliah berdasarkan ID course dari mata kuliah tersebut. Pada anotasi @Results, property untuk ID course ditulis "idCourse" karena pada CourseModel ID course bernama "idCourse", dan column untuk ID course ditulis "id_course" karena pada database nama column yang ada di tabel Course adalah "id_course". Property yang terakhir, yaitu "students", memanggil method selectStudentsbyCourse yang ada di bawahnya. Method ini me-return objek model dari course dengan course id tersebut.

Method kedua, selectStudentsbyCourse, berisi query yang mengambil npm, nama, dan GPA dari student yang mengambil mata kuliah yang bersangkutan. Method ini menerima parameter ID course yang dipanggil di method selectCourse dan mengembalikan daftar student yang mengikuti sebuah mata kuliah.

Selanjutnya, pada StudentService, saya menuliskan method selectCourse untuk kemudian diimplementasikan pada StudentServiceDatabase.

```
CourseModel selectCourse(String id);
```

Selanjutnya, pada StudentServiceDatabase saya mengimplementasikan method tersebut seperti berikut.

```
@Override
public CourseModel selectCourse(String id) {
    Log.info("course " + id + " selected");
    return studentMapper.selectCourse(id);
}
```

Method inilah yang berfungsi memanggil method selectCourse pada StudentMapper dan menjembatani antara DAO dan Controller aplikasi.

Sukrian Syahnel Putra

1306402066

ADPAP - B

Selanjutnya, pada StudentController, saya menambahkan method viewCourse dengan parameter objek Model dan string yang berupa ID dari course tersebut. Implementasi dari method ini adalah sebagai berikut.

```
@RequestMapping("/course/view/{id}")
public String viewCourse (Model model, @PathVariable(value = "id") String id)
{
    CourseModel course = studentDAO.selectCourse(id);
    if (course != null) {
        model.addAttribute("course", course);
        return "view-course";
    }

    model.addAttribute("id", id);
    return "not-found-course";
}
```

Method di atas memanggil studentDAO untuk melakukan pemilihan terhadap course. Jika course tidak ditemukan, atau course yang didapat bernilai null, maka method ini akan memerintahkan model untuk menambahkan atribut ID dari course untuk kemudian mengembalikan view di mana kode course tidak ditemukan. Sementara itu, jika berhasil ditemukan, maka course akan dimasukkan ke dalam model, lalu method tersebut mengarahkan user kepada halaman view untuk course.

Selanjutnya saya membuat halaman view untuk course dengan dua kasus, yaitu ketika ditemukan dan tidak ditemukan. Untuk kasus di mana course ditemukan (view-course) implementasinya adalah sebagai berikut.

```
<!DOCTYPE html>
```

```
<html xmlns:th="http://www.thymeleaf.org">
```

```
<head>
```

```
<title>View Course by ID</title>
```

```
</head>
```

```
<body>
```

```
<h3 th:text="'ID = ' + ${course.idCourse}">ID Course</h3>
```

```
<h3 th:text="'Nama = ' + ${course.name}">Course Name</h3>
```

```
<h3 th:text="'SKS = ' + ${course.credits}">Course Credits</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>
```

Sukrian Syahnel Putra
1306402066
ADPAP - B
</html>

Sementara itu, untuk view ketika course tidak ditemukan (not-found-course) adalah sebagai berikut.

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

    <head>

        <title>Course not found</title>

    </head>

    <body>

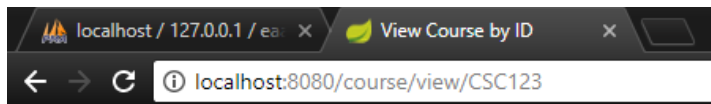
        <h1>Course not found</h1>

        <h3 th:text="'ID = ' + ${id}">Course ID</h3>

    </body>

</html>
```

Berikut adalah screenshot halaman yang muncul ketika course yang ingin dicari berhasil ditemukan.



ID = CSC123

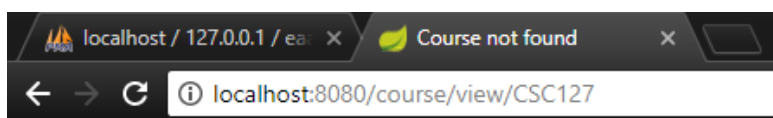
Nama = PSP

SKS = 4

Mahasiswa yang mengambil

- 124-Chanek Jr.

Sementara itu, jika course tidak ditemukan, maka tampilan halaman yang muncul adalah sebagai berikut.



Course not found

ID = CSC127