Nama : Yudistira Ramadhan
Kelas : APAP-B
NPM : 1506689635

Tutorial 5

Hal yang dipelajari

Pada tutorial kali ini saya mempelajari kembali bagaimana menghubungkan aplikasi dengan database. Pada tutorial kali ini menggunakan lebih dari satu entity. Karena menambahkan entity baru maka perlu menambahkan class model baru yang merepresentasikan entity tersebut. Lalu untuk menghubungkan antara dua entity tersebut maka diperlukan method baru yang menghubungkan relasi dua entity itu pada class StudentMapper. Saat menghubungkan kedua entity tersebut ada anotasi yang diperlukan yaitu anotasi @Result. Dimana pada anotasi tersebut terdapat beberapa variable, yaitu variable property, column, many, dan javaType. Variable property digunakan untuk mengisi nama variable yang ada di class model. Variable column digunakan untuk menerima hasil query yang ada di database atau parameter yang dibutuhkan untuk mengisi variable pada class model. Variable many digunakan untuk merefer parameter yang didapat untuk dipakai di method lain. Variable javaType digunakan untuk menentukan class return type.

Method selectAllStudents

Untuk mengubah method ini saya menambahkan beberapa baris iterasi code pada viewall.html untuk menampilkan mata kuliah apa saja yang diambil oleh mahasiswa tersebut dan jumlah sks nya. Selanjutnya saya mengubah sedikit method selectAllStudents pada class StudentMapper dengan menambahkan anotasi @Result dan beberapa variable yang diperlukan. Kemudian membuat method selectCourses pada class StudentMapper untuk mengambil data student yang berhubungan dengan course dengan query join.

Screenshoot viewall.html

```
<!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>
            <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>
            <hr/>

        <a th:href = "'/student/delete/' + ${student.npm}" > Delete Data </a><br/>
        <a th:href = "'/student/update/' + ${student.npm}" > Update Data </a><br/>
        </div>
    </body>
</html>

```

Nama : Yudistira Ramadhan
Kelas : APAP-B
NPM : 1506689635

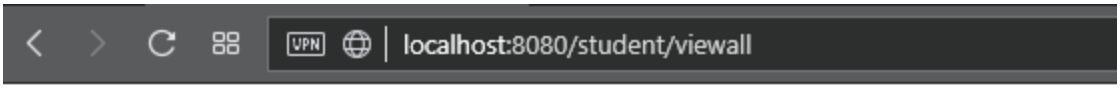
Screenshoot method selectAllStudents pada StudentMapper

```
@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 ();
```

Screenshoot method selectCourses pada StudentMapper

```
@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);
```

Nama : Yudistira Ramadhan
Kelas : APAP-B
NPM : 1506689635
Contoh dari hasil viewall



All Students

No. 1

NPM = 12345

Name = aku

GPA = 4.0

Kuliah yang diambil

- MPKT-6 sks
- PSP-4 sks

Delete Data
Update Data

No. 2

NPM = 6543

Name = mereka

GPA = 3.02

Kuliah yang diambil

- SDA-3 sks

Delete Data
Update Data

Penambahan view untuk course

Untuk membuat view course yang pertama kali saya lakukan adalah membuat class model untuk Course yang merepresentasikan idCourse, nama dari course dan credit dari course, dan list student. Lalu membuat method selectCourse dengan parameter id, method tersebut melakukan

Nama : Yudistira Ramadhan
Kelas : APAP-B
NPM : 1506689635

query untuk mengambil data dari database yang menghasilkan parameter id_course yang akan digunakan untuk method selectStudentCourse. Selanjutnya membuat method selectStudentCourse dengan menggunakan parameter yang telah diterima dari method selectCourse. Method ini melakukan pengecekan dengan melakukan join pada table studentcourse dan student dan menghasilkan npm, nama, dan gpa dari student yang berhubungan dengan course tertentu. Kemudian melakukan override method selectCourse pada class StudentServiceDatabase. Lalu memanggil method selectCourse pada class StudentService. Kemudian menambahkan @RequestMapping yang akan menerima HTTP request serta melakukan validasi terhadap id course yang di input. Jika tidak valid akan menampilkan pesan error menyatakan bahwa id course tidak ditemukan. Bila validasi berhasil akan menampilkan detail course tersebut beserta dengan para student yang mengambilnya.

Screenshoot method selectCourse dan selecStudentCourse pada StudentMapper

```
@Select("SELECT student.npm, name, gpa FROM studentcourse join student on studentcourse.npm = student.npm WHERE studentcourse.id_course = #{id}")
List<StudentModel> selectStudentCourse (@Param("id") String id);

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

Screenshoot method updateSubmit pada StudentController

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

Screenshoot method update pada StudentMapper

```
@Update("UPDATE student SET npm = #{npm}, name = #{name}, gpa = #{gpa} WHERE npm = #{npm}")
void updateStudent (StudentModel student);
```

Screenshoot method update pada StudentService

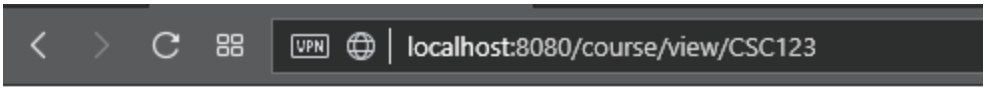
```
CourseModel selectCourse (String id);
```

Nama : Yudistira Ramadhan
Kelas : APAP-B
NPM : 1506689635

Screenshoot method selectCourse pada StudentServiceDatabase

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

Contoh hasil view course dengan id CSC123



ID = CSC123

Name = PSP

SKS = 4

Mahasiswa yang mengambil

- 12345-aku

Contoh view bila id course tidak valid

Course not found

ID = null