Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

Write Up Tutorial 5

Pembelajaran pada Tutorial 5

Pada tutorial minggu ini, saya mempelajari penghubungan antara 2 entity pada project, kedua entity yang berada dalam project ini adalah **Student** dan **Course**. Hubungan antara **Student** dan **Course** adalah Many-to-Many karena **Student** dapat mengambil beberapa **Course** dan **Course** dapat diambil oleh beberapa **Student**. Dikarenakan terdapat relasi **Student** mengambil **Course**, maka perlu dibuat satu tabel baru yang disebut dengan **studentcourse** yang menyimpan relasi tersebut.

Selain itu, pada tutorial minggu ini, saya mempelajari bagaimana memetakan hasil dari query select pada hubungan kedua tabel ke class model. Variabel property diisi dengan nama variabel di class Model yang ada, sedangkan variabel column diisi dengan nama kolom hasil query di database (parameter yang dibutuhkan).

Latihan Mengubah Method selectAllStudents

Untuk mengubah method **selectAllStudents** pada class **StudentMapper** agar halaman viewall menampilkan semua student beserta daftar kuliah yang diambil, perlu untuk memetakan hasil select query dari query yang ada untuk menampilkan seluruh student. dan menambahkan variabel courses karena 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**. Variabel javaType menentukan class yang menjadi kembalian. Kemudian variabel many diset dengan method yang akan mengisi variabel courses yaitu **selectCourses**.

```
@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 ();
```

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

Dan terakhir, perlu untuk menambahkan html code seperti dibawah ini untuk menampilkan nama mata kuliah beserta jumlah SKS dari mata kuliah yang masing-masing mahasiswa ambil.

```
<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>
```

Latihan Membuat View pada Course

Untuk membuat view pada halaman http://localhost:8080/course/view/{id} untuk Course sehingga dapat menampilkan data course beserta Student yang mengambil, perlu untuk membuat controller baru untuk menampilkan view pada halaman http://localhost:8080/course/view/{id}. Dan juga ditambahkan validasi agar jika course tidak ditemukan tampilkan view not-found dan menambahkan method yang berfungsi untuk mengubah objek course.

```
@RequestMapping("/course/view/{id_course}")
public String viewCourse (Model model, @PathVariable(value = "id_course") String id_course) {

    CourseModel course = studentDAO.selectCourse (id_course);
    if (course.getIdCourse() != null) {
        model.addAttribute ("course", course);
        return "view-course";
    } else {
        model.addAttribute ("id_course", id_course);
        return "not-found";
    }
}
```

Lalu, pada **studentMapper**, dibuat sebuah method yang berfungsi untuk mengembalikan list of student pada sebuah Course dengan id_course tertentu. Methodnya sebagai berikut:

```
@Select("SELECT student.npm, name, gpa " +
        "FROM studentcourse JOIN student " +
        "ON studentcourse.npm = student.npm " +
        "WHERE studentcourse.id_course = #{id_course} ")
List<StudentModel> selectStudents (@Param("id_course") String id_course);
```

Nama : Valian Fil Ahli
NPM : 1506724354
Kelas : B

Query tersebut akan melakukan join antara tabel **studentcourse** dengan student berdasarkan NPM dan difilter hanya pada student dengan id_course yang diberikan.

Langkah selanjutnya adalah menghubungkan method **selectCourse** dengan method **selectStudents** pada class **StudentMapper** agar saat melakukan select maka variabel List juga akan terisi.

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

Dan terakhir, perlu untuk menambahkan sebuah file html baru sesuai dengan hasil return class **StudentController** jika id_course yang dimasukkan pada url sesuai dengan salah satu id_course yang terdapat di dalam database seperti dibawah ini untuk menampilkan Course sehingga dapat menampilkan data course beserta Student yang mengambil.

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