

Tutorial 5

Menggunakan Database serta Relasi Database dalam Project Spring Boot

1. Hal apa saja yang telah Anda pelajari pada tutorial ini?

Hal yang saya pelajari dari tutorial 5 ini yaitu mengenai relasi yang ada pada *database* dan kemudian diimplementasikan pada *MVC Programming* seperti cara melakukan operasi join antara 2 *table* menggunakan MyBatis. Saya juga belajar cara menggunakan anotasi `@Results`. Anotasi `@Result` dan `@Many` untuk melakukan *mapping records* dari *database* ke Objek Model yaitu pada class Model yang ada pada *MVC property* diisikan dengan nama *variable* yang dituju serta *column* diisikan dengan hasil *query* dari *database*.

2. Berikan penjelasan terhadap hal-hal berikut:

a. Method yang diubah pada Latihan Merubah SelectAllStudents, jelaskan!

- Pada `selectAllStudents()`, anotasi `@Results` ditambahkan seperti yang ada pada `selectStudent(String npm)` sehingga *courses* setiap *students* yang ada pada `List<studentModel>` dapat terisi. `selectAllStudents` setelah diubah:

```
@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 ();
```

- Query untuk mengambil *course* yang diambil oleh *student*

```
@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);
```

- Mengubah *template* viewall.html sehingga terjadi iterasi untuk *courses* setiap *students* yang diiterasikan pada view tersebut

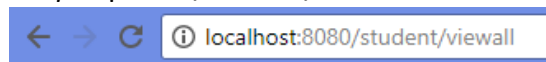
```
<div th:each="student, iterationStatus: ${students}">
  <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
  <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
  <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
  <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>

  <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>

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

  <hr/>
</div>
```

- Output pada ../student/viewall



All Students

No. 1

NPM = 123

Name = Rosalinda

GPA = 3.5

Kuliah yang diambil

- MPKT-6.0 sks

[Delete Data](#)

[Update Data](#)

No. 2

NPM = 124

Name = Gheafany Widyatika Putri

GPA = 3.4

Kuliah yang diambil

- PSP-4.0 sks
- SDA-3.0 sks

Penjelasan:

Saya mengganti proses *mapping record* dari *database* ke objek *student* dengan menggunakan anotasi `@Results` dalam proses *mapping*, saya mencocokkan nama *variable* di *StudentModel* dengan nama kolom di *table student*. Untuk *variable* `List<CourseModel> courses` yang nantinya akan menyimpan *list course* yang diambil oleh *student*, saya melakukan proses *join table course* dan *studentcourse* untuk mendapatkan *list course* tersebut dan menyimpannya dalam *variable courses* dalam *StudentModel* dengan menggunakan anotasi `@Many` karena sifat relasinya adalah *many-to-many*

b. Method yang dibuat pada Latihan Menambahkan View pada Course, jelaskan!

- Pada *StudentMapper* dibuat *method* baru bernama *selectCourse*

```
@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="selectStudentCourse"))
})
CourseModel selectCourse (@Param("id_course") String id_course);
```
- *Query* untuk mengambil *student* yang mengikuti suatu *course* yang di *select*

```
@Select("SELECT student.npm, student.name " +
    "from studentcourse " +
    "join student on student.npm = studentcourse.npm " +
    "where id_course = #{id_course}")
List<StudentModel> selectStudentCourse(@Param("id_course") String id_course);
```
- *Controller view* pada halaman `http://localhost:8080/course/view/{id}` untuk *Course* sehingga dapat menampilkan data *course* beserta *Student* yang mengambil

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

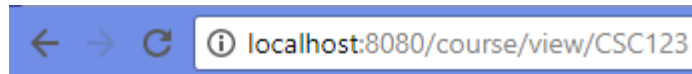
- Membuat *view template* untuk course dengan nama `course.html`:

```

1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <title>View Course</title>
5   </head>
6   <body>
7
8     <h3 th:text="'ID = ' + ${course.id_course}"></h3>
9     <h3 th:text="'Nama = ' + ${course.name}"></h3>
10    <h3 th:text="'SKS = ' + ${course.credits}"></h3>
11
12    <ul th:each="student, iterationStatus: ${course.students}">
13      <li th:text="${student.npm} + '-' + ${student.name}" >
14
15        </li>
16    </ul>
17  </body>
18 </html>

```

- *Output* yang akan ditampilkan:



ID = CSC123

Nama = PSP

SKS = 4.0

- 124-Gheafany Widyatika Putri

Penjelasan:

Variable `List<StudentModel> students` ditambahkan di dalam `CourseModel`. Untuk mengisi `List<StudentModel> students` yang ada pada `course` yang kita *select* diperlukan *query join* antara *table studentcourse* dan *student* yaitu dengan cara proses *mapping record* dari *database* ke objek `CourseModel` yang akan disimpan di dalam *variable students*. Menggunakan anotasi `@Many` karena sifat relasinya adalah *many-to-many*