Ricca Fitriani
1506689616
APAP – B

Tutorial 5

Menggunakan Database serta Relasi Database dalam Project Spring Boot

Write up :

Tutorial ini menjelaskan cara menggunakan *database* pada *project* Spring Boot secara lebih lanjut. Pada tutorial ini, dijelaskan bahwa kita bisa memetakan hasil *query* ke *class model.* Hal tersebut dapat dilakukan dengan anotasi Results dan Result. Pada anotasi Result terdapat variabel property yang diisi dengan nama atribut pada kelas model yang dituju, dan juga terdapat variabel column yang diisi dengan nama kolom pada *database* yang merupakan hasil *query* di database (parameter yang dibutuhkan oleh sebuah *method*). Pada anotasi Result, kita juga dapat memetakan hasil *query* suatu *method* lain pada *method* dengan anotasi Result tersebut. Hal tersebut dapat dilakukan dengan menambahkan variabel javaType dan many dengan anotasi Many. Variabel javaType diisi dengan kembalian *class,* sedangkan anotasi Many diisi dengan *method* yang hasil *query*nya ingin diambil.

Method selectAllStudents :

Untuk menampilkan semua *student* beserta daftar *courses* yang diambil, perlu diubah *method* selectAllStudents pada kelas StudentMapper menjadi sebagai berikut:

```
32    @Select("select npm, name, gpa from student")
33    @Results(value = {
34            @Result(property="npm", column="npm"),
35            @Result(property="name", column="name"),
36            @Result(property="gpa", column="gpa"),
37            @Result(property="courses", column="npm",
38            javaType = List.class,
39            many=@Many(select="selectCourses"))
40            })
41    List<StudentModel> selectAllStudents ();
```

Pada *method* tersebut, akan diambil hasil *query* pada *method* selectCourses yang mengembalikan daftar *courses* yang diambil *student* tersebut. Daftar *courses* tersebut diambil berdasarkan npm untuk setiap data *student* yang di-*retrieve* oleh *method* selectAllStudents tersebut. Kemudian, pada *template* viewall, perlu dilakukan iterasi pada setiap *courses* yang diambil oleh setiap *student* dengan menggunakan th:each sebagai berikut:

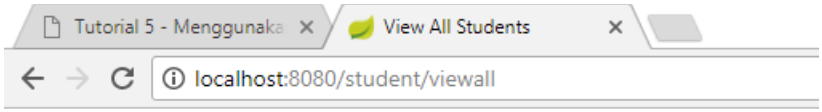
Ricca Fitriani
1506689616
APAP – B

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

```

Maka, ketika *user* memanggil /student/viewall, akan ditampilkan halaman sebagai berikut:



All Students

No. 1

NPM = 111334

Name = Jihoon

GPA = 3.23

Kuliah yang diambil

- MPKT-6 sks

Delete Data
Update Data

No. 2

NPM = 123

Name = Chanek

GPA = 3.6

Kuliah yang diambil

- MPKT-6 sks

Delete Data
Update Data

Ricca Fitriani
1506689616
APAP – B

View Course:

Untuk menampilkan data *course* beserta *student* yang mengambil *course* tersebut. Pertama, agar *logic* untuk *student* dan *course* dapat dipisah, dibuat *class controller* CourseController, sebagai berikut:

```
1  package com.example.controller;
2
3  import org.springframework.beans.factory.annotation.Autowired;
11
12 @Controller
13 public class CourseController
14 {
15     @Autowired
16     CourseService courseDAO;
17
18     @RequestMapping("/course/view/{id}")
19     public String viewCourse (Model model,
20             @PathVariable(value = "id") String id)
21     {
22         CourseModel course = courseDAO.selectCourse (id);
23
24         if (course != null) {
25             model.addAttribute ("course", course);
26             return "view-course";
27         } else {
28             model.addAttribute ("id", id);
29             return "not-found-course";
30         }
31     }
32 }
```

Pada *class* tersebut, dibuat *method* viewCourse yang menangani RequestMapping /course/view/id. *Method* tersebut akan memanggil *method* selectCourse dengan parameter id yang diinginkan pada *class* CourseService. *Method* selectCourse pada *class* CourseService tersebut adalah sebagai berikut:

```
1  package com.example.service;
2
3  import com.example.model.CourseModel;
4
5  public interface CourseService
6  {
7      CourseModel selectCourse (String id);
8  }
```

Kelas tersebut adalah *interface* yang diimplementasikan pada kelas CourseServiceDatabase sebagai berikut:

Ricca Fitriani
1506689616
APAP – B

```java
package com.example.service;


import org.springframework.beans.factory.annotation.Autowired;

@Service
public class CourseServiceDatabase implements CourseService
{
    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseModel selectCourse (String id)
    {
        return courseMapper.selectCourse (id);
    }
}
```

Pada kelas CourseServiceDatabase tersebut, *method* selectCourse memanggil *method* selectCourse pada kelas courseMapper dengan parameter id yang dikirimkan dari *controller.* Kelas courseMapper tersebut akan mengeksekusi *query* yang dibutuhkan untuk mengambil data *course* yang diinginkan beserta *student* yang mengambilnya dengan *method-method* sebagai berikut:

```java

import java.util.List;

@Mapper
public interface CourseMapper
{
    @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="selectStudents"))
            })
    CourseModel selectCourse (@Param("id") String id);


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

Method selectCourse akan mengambil *course* dengan id yang dikirimkan dari *controller.* Kemudian, untuk daftar *students* yang mengambil *course* tersebut, diambil dari hasil *query* yang dijalankan oleh *method* selectStudents. *Method* selectStudents tersebut akan mengembalikan list dari *students* yang npmnya berada di baris yang sama dengan id *course* yang diinginkan pada *database. Students* tersebut akan dipetakan menjadi atribut *students* pada kelas CourseModel dengan anotasi Result dan variabel property yang diisi dengan "*students*" yaitu atribut yang ada pada kelas CourseModel, dan variabel column yang diisi dengan "*id_course*" yaitu kolom pada *database* yang ingin dikirim menjadi paramater pada *method* selectStudents.

Ricca Fitriani
1506689616
APAP – B

Hasil data *course* yang sudah diambil dari *database* tersebut, akan dikirimkan ke *template* view-course.html oleh *controller.* Pada view-course.html perlu dilakukan iterasi untuk *list of students* yang mengambil *course* tersebut sebagai berikut:

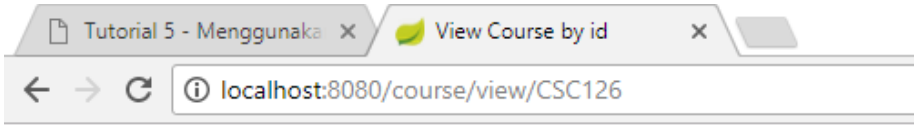
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by id</title>
    </head>
    <body>
        <h3 th:text="'ID = ' + ${course.idCourse}">Student NPM</h3>
        <h3 th:text="'Nama = ' + ${course.name}">Student Name</h3>
        <h3 th:text="'SKS =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

Jika *course* tidak ada, maka *controller* akan memanggil *template* not-found-course.html, 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}"></h3>
    </body>
</html>
```

Ketika program dijalankan dengan url /course/view/CSC126 akan ditampilkan halaman sebagai berikut:

Ricca Fitriani
1506689616
APAP – B



ID = CSC126

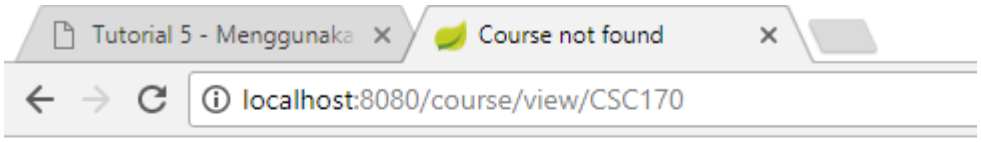
Nama = MPKT

SKS = 6

Mahasiswa yang mengambil

- 111334 - Jihoon
- 123 - Chanek

Jika id *course* yang diinginkan tidak ada, maka akan ditampilkan halaman sebagai berikut:



Course not found

ID = CSC170