

Tutorial 5

Arsitektur dan Pemrograman Aplikasi Perusahaan Semester Ganjil 2017/2018

Menggunakan Database serta Relasi Database dalam Project Spring Boot

1. Hal apa saja yang anda pelajari dari tutorial ini

Pada tutorial kali ini, saya belajar mengenai relasi pada database yang kemudian diimplement pada *MVC Programming*, seperti menggunakan *join*. Selanjutnya saya juga belajar bagaimana caranya memetakan hasil *query* pada *class Model* yang ada pada MVC, dimana *property* diisi dengan nama *variable* yang dituju, dan *column* diisi dengan hasil *query* dari *database*.

2. Penjelasan untuk method yang anda ubah pada latihan merubah `SelectAllStudents`

Pertama-tama, pada method `selectAllStudents()`, saya menambahkan anotasi `@Results` seperti yang ada pada `selectStudent(String npm)` sehingga *courses* setiap *students* yang ada pada `List<studentModel>` dapat terisikan, *method* menjadi seperti berikut:

```
@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 ();
```

Selanjutnya, mengubah pada template `viewall.html` sehingga terdapat juga iterasi untuk *courses* setiap *students* yang diiterasikan pada *view* tersebut, menjadi seperti berikut:

```
<!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="'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>
```

Sehingga keluaran pada *view* menjadi:

View All Students

← → ↻ 🏠

localhost:8080/student/viewall

All Students

NPM = 123

Name = Abdalla

GPA = 3.5

Kuliah yang diambil

- MPKT-6 sks

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

NPM = 124

Name = Kemal Abdalla Chair

GPA = 4.0

Kuliah yang diambil

- PSP-4 sks
- SDA-3 sks

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

3. Penjelasan untuk method yang anda buat pada latihan menambahkan View pada Course

Pertama-tama, saya membuat *mapper*, *service*, dan *CourseServiceDatabase* baru untuk *course*, dengan nama *CourseMapper*, *CourseService*, *CourseServiceDatabase*:

```
CourseMapper.java CourseService.java CourseServiceDatabase.java
1 package com.example.dao;
2
3 import java.util.List;
17
18 @Mapper
19 public interface CourseMapper
20 {
```

Kemudian pada mapper yang baru saja dibuat, kita buat method baru bernama *selectCourse(String id)* dengan syntax 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="selectAllStudentsWithCourse"))
})
CourseModel selectCourse (@Param("id") String id);
```

Untuk mengisi *List<StudentModel> students* yang ada pada *course* yang kita *select*, untuk mengisi bagian “Mahasiswa yang mengambil” pada *view*, diperlukan juga method yang mirip dengan *selectCourses* yang telah diberikan pada tutorial, seperti berikut:

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

Kemudian implementasikan *interface – interface* yang dibutuhkan pada *CourseService* dan *CourseServiceDatabase*:

```
CourseMapper.java CourseService.java CourseServiceDatabase.java
1 package com.example.service;
2
3 import java.util.List;
4
5
6
7 public interface CourseService
8 {
9     CourseModel selectCourse (String id);
10 }
11
```

```
CourseMapper.java CourseService.java CourseServiceDatabase.java
1 package com.example.service;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14 @Slf4j
15 @Service
16 public class CourseServiceDatabase implements CourseService
17 {
18     @Autowired
19     private CourseMapper courseMapper;
20
21     @Override
22     public CourseModel selectCourse(String id) {
23         log.info ("select course with id {}", id);
24         return courseMapper.selectCourse(id);
25     }
26 }
27
28
```

Kemudian membuat *controller* baru untuk CourseService, pada *controller*, mengimport CourseService dengan anotasi @Autowired:

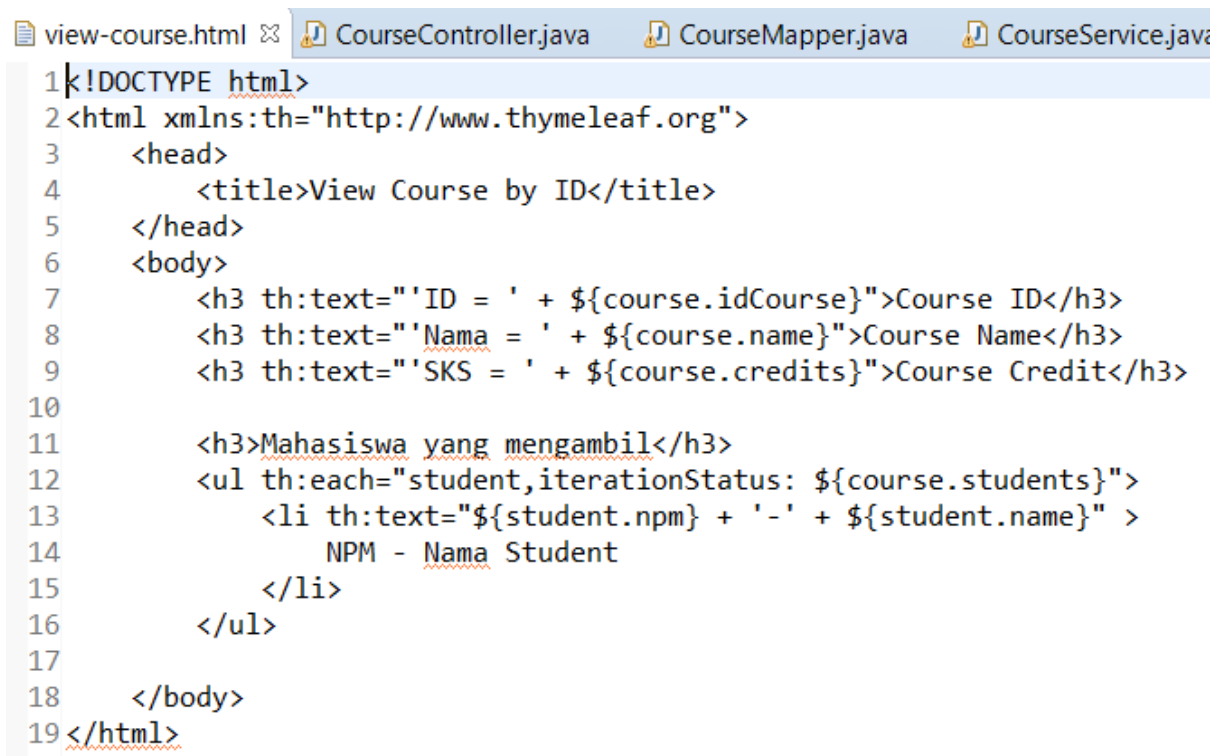
```
CourseMapper.java CourseService.java CourseServiceDatabase.java CourseController.java
1 package com.example.controller;
2
3 import java.util.List;
4
5
6
7
8 @Controller
9 public class CourseController
10 {
11
12
13
14
15
16
17
18
19
20
21
22 @Autowired
23 CourseService courseDAO;
24
```

Lalu, membuat *method* baru untuk *view* yang dibutuhkan, *method* mirip dengan *view* untuk *student*:

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

    if (course != null) {
        model.addAttribute ("course", course);
        return "view-course";
    } else {
        model.addAttribute ("id", id);
        return "not-found_course";
    }
}
```

Terakhir, membuat *view* template untuk *viewCoursePath* dengan nama *view-course.html* dan *not-found_course.html* sesuai dengan *return value* pada *controller*:



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

```
not-found_course.html view-course.html CourseController.java
1<!DOCTYPE html>
2<html xmlns:th="http://www.thymeleaf.org">
3  <head>
4    <title>Course not found</title>
5  </head>
6  <body>
7    <h1>Course not found</h1>
8    <h3 th:text="'Name = ' + ${name}">Course Name</h3>
9  </body>
10</html>
11
```

Maka, tampilan dapat memunculkan:

