WRITE-UP TUTORIAL 5 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

MENGGUNAKAN DATABASE SERTA RELASI DATABASE DALAM PROJECT SPRING BOOT

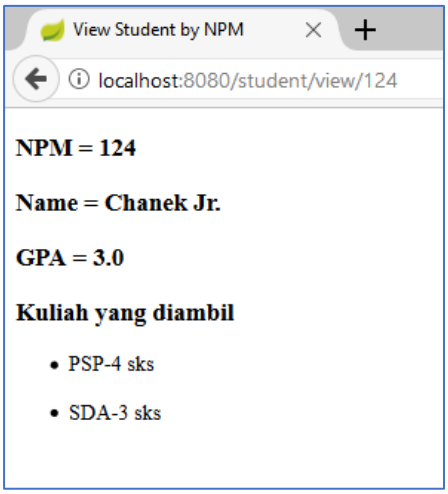
A. HAL YANG DIPELAJARI

Dalam pengerjaan tutorial kelima mata kuliah Arsitektur dan Pemrograman Aplikasi Perusahaan, saya kembali belajar mengenai penggunaan *database* dalam *project spring boot*, mempelajari mengenai hubungan antar relasi dalam *database* serta tindakan yang perlu dilakukan atau perlakuan untuk menangani kasus hubungan relasi yang sifatnya *Many-to-Many*.

Langkah pertama untuk memulai tutorial ini adalah dengan membuat tabel Course. Course berisi kolom id_course , name, dan credits. Kolom id_course adalah id dari mata kuliah, kolom name adalah nama mata kuliah, dan credits adalah jumlah sks mata kuliah yang bersangkutan. Karena penambahan tabel Course, *database* sekarang mempunyai 2 tabel yaitu Student dan Course. Data pada Student dapat mengambil banyak data pada Course dan data pada Course dapat diambil banyak oleh data pada Student. Oleh sebab itu, hubungan antara Course dan Student adalah *Many-to-Many*. Kasus hubungan relasi *Many-to-Many* antara tabel Student dan tabel Course perlu dibuat sebuah tabel / relasi lagi yaitu studentcourse untuk menyimpan relasi tersebut. Tabel studentcourse terdiri dari kolom npm dan id_course.

Tabel-tabel tersebut dimanfaatkan pada tutorial dan latihan kali ini untuk menampilkan data-data course yang diambil oleh setiap student serta menampilkan course berikut data-data student yang mengambil course tersebut.

Jadi pada tutorial ini saya mempelajari bagaimana melakukan pemetaan terhadap hubungan antar tabel / relasi pada *database* serta menampilkan hasil pemetaan tersebut pada *view*. Misalnya ketika dilakukan *view* terhadap suatu *student* berdasarkan npm tertentu dapat juga dilihat *course* yang diambil *student* tersebut dengan memanfaatkan tabel studentcourse. Berikut ini *screenshot* untuk *view student* dengan npm 124.



WRITE-UP TUTORIAL 5 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

Berikut ini kondisi *database* terkini.

Screenshot dari tabel-tabel yang ada di *database* :

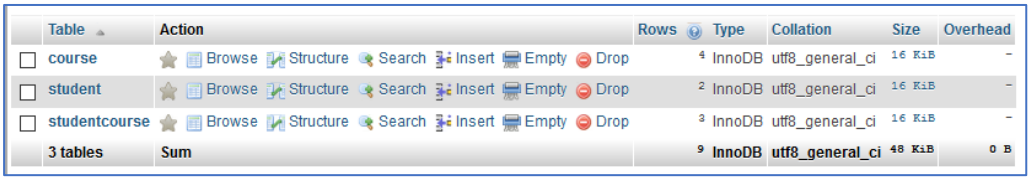
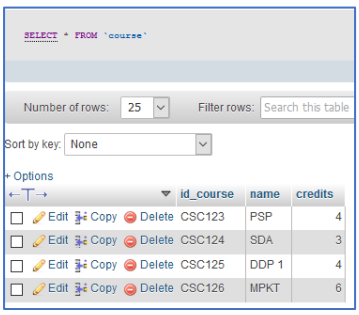


Table	Action	Rows	Type	Collation	Size	Overhead
course	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	16 KiB	-
student	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	16 KiB	-
studentcourse	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8_general_ci	16 KiB	-
3 tables	Sum	9	InnoDB	utf8_general_ci	48 KiB	0 B

Screenshot data-data yang ada pada tabel course, student, serta studentcourse :



SELECT * FROM `course`

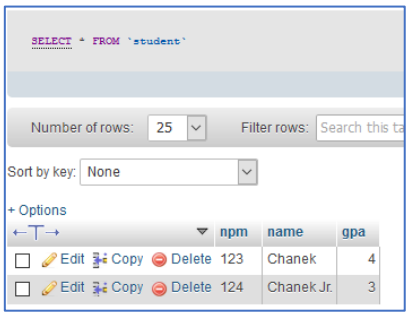
Number of rows: 25 Filter rows: Search this table

Sort by key: None

+ Options

	id_course	name	credits
Edit Copy Delete	CSC123	PSP	4
Edit Copy Delete	CSC124	SDA	3
Edit Copy Delete	CSC125	DDP 1	4
Edit Copy Delete	CSC126	MPKT	6

Course



SELECT * FROM `student`

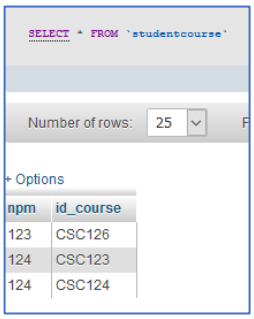
Number of rows: 25 Filter rows: Search this ta

Sort by key: None

+ Options

	npm	name	gpa
Edit Copy Delete	123	Chanek	4
Edit Copy Delete	124	Chanek Jr.	3

Student



SELECT * FROM `studentcourse`

Number of rows: 25

+ Options

npm	id_course
123	CSC126
124	CSC123
124	CSC124

Studentcourse

WRITE-UP TUTORIAL 5 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

B. PENJELASAN METHOD SELECTALLSTUDENTS

Berikut ini *screenshot* method selectAllStudents yang ada pada *class* 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 ();
```

Terdapat anotasi Results yang melakukan pemetaan dari *query* select ke *class* Model. Anotasi Result yang menjadi *value* pada anotasi Results berisi property dan column. Property diisi berdasarkan nama variabel yang ada pada *class* StudentModel, sedangkan untuk column diisi dengan nama kolom pada *database*. Yang perlu mendapat perhatian berbeda disini adalah untuk variabel courses pada StudentModel yang akan diisi. Untuk column diisi dengan npm. Hal tersebut dikarenakan column npm pada *database* akan menjadi parameter *method* selectCourses. Variabel lain yang terlibat untuk menampilkan *courses* yang diambil oleh masing-masing Student adalah javaType yang menentukan *class* yang akan di-*return* yaitu *class* List. Variabel many diset oleh *method* yang akan mengisi variabel courses pada *class* StudentModel yaitu selectCourses.

Berikut ini *screenshot* dari *method* selectCourses :

```
@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);
```

Method selectCourses ini yang akan mengembalikan *List of CourseModel* yang digunakan pada *method* selectAllStudents untuk mengisi variabel courses di *class* StudentModel. *Query* select pada *method* ini melakukan join tabel studentcourse dan course dengan kondisi sesuai pada npm student yang bersangkutan.

Pada Controller di *method* viewall tidak perlu dilakukan perubahan.

WRITE-UP TUTORIAL 5 ADPAP

Nama : Yosua Bisma Putrapratama

NPM : 1506689622

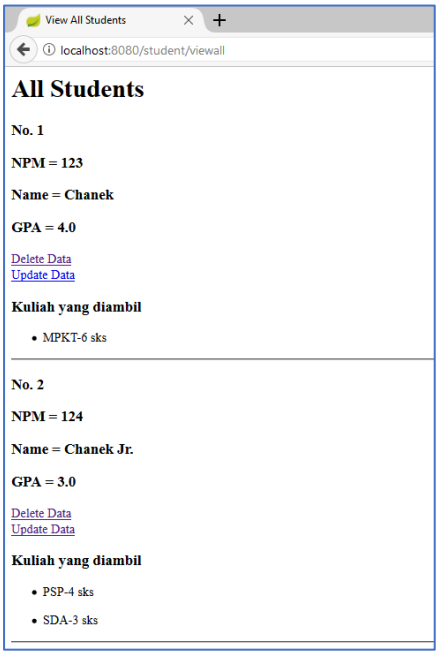
Kelas : ADPAP – B

```
84      @RequestMapping("/student/viewall")
85      public String view (Model model)
86      {
87          List<StudentModel> students = studentDAO.selectAllStudents ();
88          model.addAttribute ("students", students);
89
90          return "viewall";
91      }
```

Pada viewall.html ditambahkan *tag* ul (*unordered / bulleted list*) untuk menampilkan *course* yang diambil oleh setiap student. Disini diilakukan iterasi pada variabel courses yang ada pada *class* StudentModel untuk semua variabel student. Variabel courses yang merupakan kumpulan *list* dari *class* CourseModel akan diambil variabel name dan credits. Sehingga yang ditampilkan pada setiap student adalah *course.name* dan *course.credits.*

```
1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3      <head>
4          <title>View All Students</title>
5      </head>
6      <body>
7          <h1>All Students</h1>
8
9          <div th:each="student,iterationStatus: ${students}">
10             <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
11             <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
12             <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
13             <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
14             <a th:href="'/student/delete/' + ${student.npm}">Delete Data</a><br/>
15             <a th:href="'/student/update/' + ${student.npm}">Update Data</a><br/>
16             <h3>Kuliah yang diambil</h3>
17             <ul th:each="course,iterationStatus: ${student.courses}">
18                 <li th:text="${course.name} + '-' + ${course.credits} + ' sks'">
19                     Nama kuliah-X SKS
20                 </li>
21             </ul>
22             <hr/>
23         </div>
24     </body>
25 </html>
26
```

Berikut ini hasil yang ditampilkan pada *browser* ketika program dijalankan :



WRITE-UP TUTORIAL 5 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

C. PENJELASAN VIEW PADA COURSE

Pada latihan untuk melakukan *view* terhadap suatu *course* berdasarkan *id course* tersebut, perlu dilakukan penambahan *method* pada StudentMapper. Berikut ini *screenshot* dari *method-method* yang terlibat untuk melakukan *view course* berdasarkan *id course.*

```
59      // Membuat mapper untuk memilih course berdasarkan id course
60      @Select("select id_course, name, credits from course where id_course = #{idCourse}")
61      @Results(value = {
62              @Result(property="idCourse", column="id_course"),
63              @Result(property="name", column="name"),
64              @Result(property="credits", column="credits"),
65              @Result(property="students", column="id_course",
66                      javaType = List.class,
67                      many=@Many(select="selectStudents"))
68      })
69      CourseModel selectCourse(@Param("idCourse") String idCourse);
70
71      @Select("select student.npm, name, gpa "
72              + "from studentcourse join student "
73              + "on studentcourse.npm = student.npm "
74              + "where studentcourse.id_course = #{idCourse}")
75      List<StudentModel> selectStudents (@Param("idCourse") String idCourse);
```

Pertama-tama kita dapat membuat *method* selectCourse yang akan menerima parameter *idCourse* dan mengembalikan CourseModel. Pada bagian ini juga terdapat anotasi Results yang melakukan pemetaan dari *query* select ke *class* Model. Anotasi Result yang menjadi *value* pada anotasi Results berisi property dan column. Property diisi berdasarkan nama variabel yang ada pada *class* CourseModel, sedangkan untuk column diisi dengan nama kolom pada *database*. Sama seperti latihan sebelumnya, yang perlu mendapat perhatian berbeda disini adalah untuk variabel students pada CourseModel yang akan diisi. Untuk column diisi dengan id_course. Hal tersebut dikarenakan column id_course pada *database* akan menjadi parameter *method* selectStudents. Variabel lain yang terlibat untuk menampilkan *students* yang mengambil *course* tersebut adalah javaType yang menentukan *class* yang akan di-*return* yaitu *class* List karena pada *class* CourseModel variabel students merupakan *List of object* StudentModel. Variabel many diset oleh *method* yang akan mengisi variabel students pada *class* CourseModel yaitu selectStudents.

Setelahnya, dapat dibuat *method* selectStudents. *Method* ini yang nantinya akan menampilkan *student* yang mengambil *course* tertentu yang ingin di-*view*. Oleh karena itu, pengambilan data *student* didasarkan pada idCourse dari *course* yang ditampilkan nantinya. *Method* selectStudents ini tentunya akan mengembalikan *List of StudentModel* yang digunakan pada *method* selectCourse untuk mengisi variabel students di *class* CourseModel. *Query* select pada *method* ini melakukan join tabel studentcourse dan student dengan kondisi sesuai pada idCourse dari course yang bersangkutan.

WRITE-UP TUTORIAL 5 ADPAP
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

Selanjutnya kita tambahkan *method* selectCourse dengan parameter String idCourse pada *interface* StudentService. Berikut ini *screenshot*-nya :

```
// menambahkan method selectCourse pada interface
CourseModel selectCourse (String idCourse);
```

Dari *interface* StudentService *method* selectStudent dilengkapi pada *class* StudentServiceDatabase yang *implements* StudentService. *Method* ini akan mengembalikan CourseModel dari hasil pemanggilan *method* selectCourse pada StudentMapper dengan *parameter* String idCourse.

```
// Melakukan override method selectCourse pada interface Student Service
@Override
public CourseModel selectCourse (String idCourse) {
    log.info("select course with idCourse {}", idCourse);
    return studentMapper.selectCourse(idCourse);
}
```

Selanjutnya, pada *class* StudentController anotasi *RequestMapping* diisi dengan "/course/view/{idCourse}" , idCourse nanti akan diisi dengan id dari *course* yang ingin dilihat. Dalam viewCourse dibuat *object* CourseModel dengan nama variabel course yang berisi *object* CourseModel dari pemanggilan *method* selectCourse di StudentServiceDatabase dengan parameter idCourse. Nantinya akan dicek apakah *course* yang dicari berdasarkan id-nya ada atau tidak. Jika ada, maka akan ditambahkan *object* CourseModel dengan variabel course pada attribute course pada halaman *view*. Setelahnya akan mengembalikan *view-course* yang akan menampilkan halaman *view* yaitu view-course.html. Jika tidak ada, maka keterangan course berdasarkan idCourse tersebut tidak ada dengan menambahkan String idCourse pada atribut idCourse di halaman *view*. Lalu, akan mengembalikan *view* yaitu halaman not-found-course.html.

```
138     // Untuk Halaman view COURSE
139     @RequestMapping("/course/view/{idCourse}")
140     public String viewCourse(Model model,
141             @PathVariable(value = "idCourse") String idCourse) {
142         CourseModel course = studentDAO.selectCourse (idCourse);
143         if (course != null) {
144             model.addAttribute ("course", course);
145             return "view-course";
146         }else {
147             model.addAttribute ("idCourse", idCourse);
148             return "not-found-course";
149         }
150     }
151
```

WRITE-UP TUTORIAL 5 ADPAP

Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

Berikut ini *screenshot* dari halaman *view* untuk menampilkan *course* yang ingin dilihat berdasarkan *id course* -nya yaitu view-course.html :

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

```

Pada view-course.html akan menerima *object* CourseModel yang dikirimkan dari controller yang meliputi nama variabel dan *value*-nya yaitu idCourse, name, credits, serta students (*List of Object* StudentModel). Dari *List of Object* StudentModel tersebut nantinya akan diambil *value* dari variabel npm dan name untuk ditampilkan.

Jika *course* yang ingin dilihat tidak ada, maka akan ditampilkan halaman *view* not-found-course.html. Disini akan menerima dan menampilkan idCourse yang dicari dan menunjukkan bahwa *course* yang dicari berdasarkan id tersebut tidak ditemukan. Berikut ini *screenshot* dari halaman not-found-course.html :

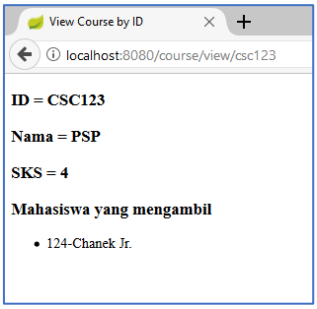
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Course not found</title>
    </head>
    <body>
        <h1>Course not found</h1>
        <h3 th:text="'ID = ' + ${idCourse}">Course ID</h3>
    </body>
</html>

```

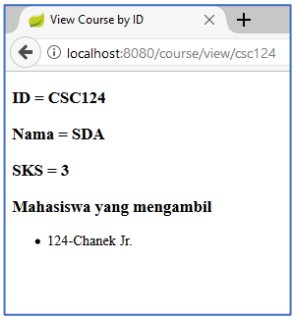
WRITE-UP TUTORIAL 5 ADPAP

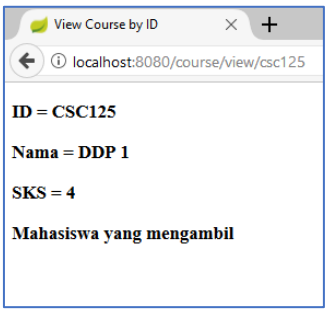
Nama : Yosua Bisma Putrapratama
NPM : 1506689622
Kelas : ADPAP – B

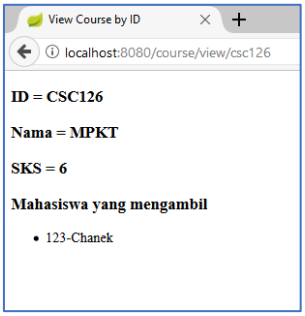
Berikut *screenshot* penerapan *view course by id* ini dengan contoh-contoh ketika program dijalankan dengan beragam kondisi :

- Ketika *course* yang dicari ditemukan









- Ketika *course* yang dicari tidak ditemukan