

MEMBUAT TABEL COURSE

Server: 127.0.0.1 » Database: eaap » Table: course

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id_course	varchar(20)	utf8_general_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext More
2	name	varchar(45)	utf8_general_ci		Yes	NULL			Change Drop Primary Unique Index Spatial Fulltext More
3	credits	double			Yes	NULL			Change Drop Primary Unique Index Spatial Fulltext More

Check all With selected: Browse Change Drop Primary Unique Index Add to central columns
Remove from central columns

Print Propose table structure Track table Move columns Improve table structure

Add 1 column(s) after credits Go

POPULASI TABEL

+ Options

			id_course	name	credits
<input type="checkbox"/>	Edit	Copy	Delete	CSC123	PSP 4
<input type="checkbox"/>	Edit	Copy	Delete	CSC124	SDA 3
<input type="checkbox"/>	Edit	Copy	Delete	CSC125	DDP 1 4
<input type="checkbox"/>	Edit	Copy	Delete	CSC126	MPKT 6

MEMBUAT TABEL STUDENTCOURSE

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	npm	varchar(20)	utf8_general_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext More
2	id_course	varchar(30)	utf8_general_ci		No	None			Change Drop Primary Unique Index Spatial Fulltext More

Check all With selected: Browse Change Drop Primary Unique Index Add to central columns
Remove from central columns







POPULASI TABEL

+ Options

	npm	id_course
	123	CSC126
	124	CSC123
	124	CSC124

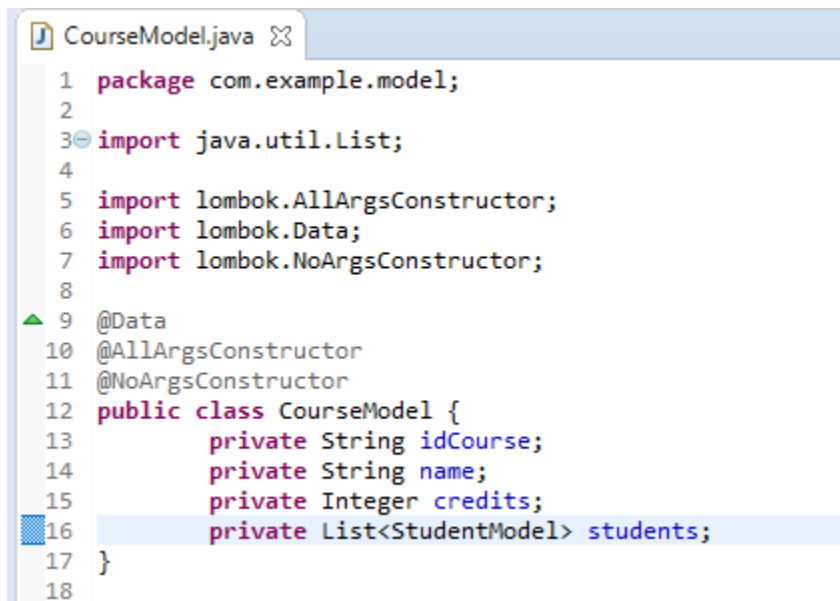
TABEL STUDENT

+ Options

				npm	name	gpa
<input type="checkbox"/>	 Edit	 Copy	 Delete	123	chanek	3.6
<input type="checkbox"/>	 Edit	 Copy	 Delete	124	chanek jr.	3.4

MENAMBAHKAN MODEL

1. Menambahkan class CourseModel

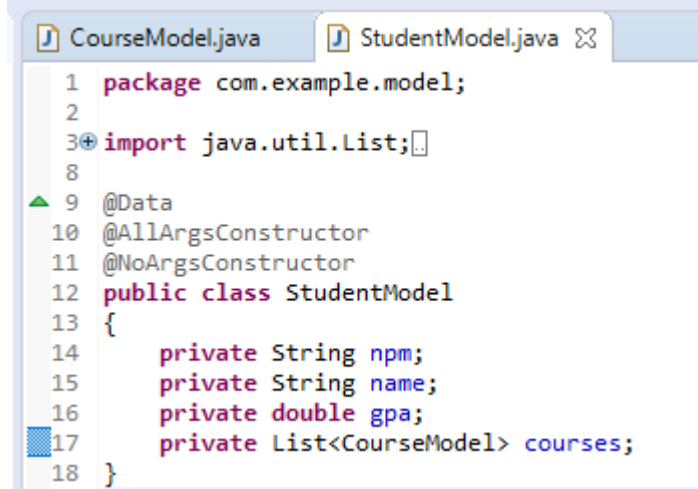


```

1 package com.example.model;
2
3 import java.util.List;
4
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8
9 @Data
10 @AllArgsConstructor
11 @NoArgsConstructor
12 public class CourseModel {
13     private String idCourse;
14     private String name;
15     private Integer credits;
16     private List<StudentModel> students;
17 }
18

```

2. Menambahkan list of CourseMode pada variabel class StudentModel



```

1 package com.example.model;
2
3 import java.util.List;
4
5 @Data
6 @AllArgsConstructor
7 @NoArgsConstructor
8
9 public class StudentModel
10 {
11     private String npm;
12     private String name;
13     private double gpa;
14     private List<CourseModel> courses;
15 }
16

```

pada StudentController akan muncul error karena menambahkan parameter constructor baru. Ubah inisialisasi constructor menjadi:

```

StudentModel student = new StudentModel (npm, name, gpa, null);
studentDAO.addStudent (student);

```

3. Selanjutnya kita akan membuat method pada StudentMapper yang mengembalikan list of course pada Student dengan NPM tertentu. Methodnya adalah sebagai berikut

```

@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);

```

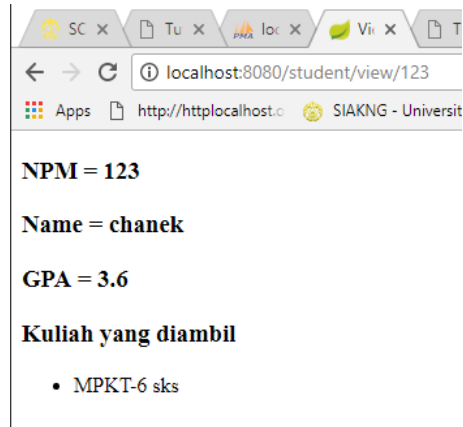
4. menghubungkan method selectStudent dengan method selectCourses agar saat melakukan select maka variabel List juga akan terisi. Pada class StudentMapper terdapat method selectStudent dan diganti menjadi

```
public interface StudentMapper
{
    @Select("select npm, name, gpa from student where npm = #{npm}")
    @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"))
    })
    StudentModel selectStudent (@Param("npm") String npm);
}
```

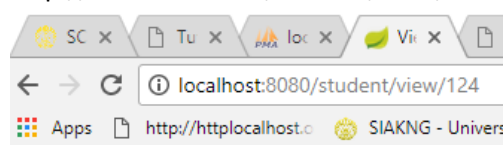
5. Selanjutnya kita akan menambahkan view pada view.html untuk menampilkan list kuliah yang diambil oleh mahasiswa seperti berikut:

```
0
1      <h3>Kuliah yang diambil</h3>
2      <ul th:each="course, iterationStatus: ${student.courses}">
3          <li th:text="${course.name} + '-' + ${course.credits} + ' sks'">
4              Nama kuliah-X SKS</li>
5      </ul>
6  </body>
7 </html>
8
```

6. Jalankan program tersebut dan buka <http://localhost:8080/student/view/123>



<http://localhost:8080/student/view/124>



LATIHAN

1. Ubah method selectAllStudents pada kelas StudentMapper agar halaman viewall menampilkan semua student beserta daftar kuliah yang diambil.

- mengubah method selectAllStudent pada StudentMapper menjadi sebagai 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 ();
```

Hal ini menggabungkan method selectAllStudent dengan method selectCourses sehingga saat melakukan selectAllStudent variable List<Course> juga terisi.

- mengubah html viewall

```
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
15 <h3>Kuliah yang diambil</h3>
16 <ul th:each="course, iterationStatus: ${student.courses}">
17 <li th:text="${course.name} + '-' + ${course.credits} + ' sks'" >
18 Nama Kuliah-X SKS
19 </li>
20 </ul>
21
22 <a th:href="'/student/delete/' + ${student.npm}" > Delete Data </a><br/>
23 <a th:href="'/student/update/' + ${student.npm}" > Update Data </a><br/>
24 <hr/>
25 </div>
26 </body>
27 </html>
```

- <http://localhost:8080/student/viewall>

All Students

No. 1

NPM = 123

Name = chaneke

GPA = 3.6

Kuliah yang diambil

- MPKT-6 sks

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

No. 2

NPM = 124

Name = chaneke jr.

GPA = 3.4

Kuliah yang diambil

- PSP-4 sks
- SDA-3 sks

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

2. Buatlah view pada halaman <http://localhost:8080/course/view/{id}> untuk Course sehingga dapat menampilkan data course beserta Student yang mengambil.

- membuat method selectCourse pada interface studentService

```
CourseModel selectCourse (String id_course);
```

- maka method tersebut diimplementasikan di studentServiceDatabase

```
@Override
public CourseModel selectCourse(String id_course) {
    log.info ("select student with id_course {}", id_course);
    return studentMapper.selectCourse (id_course);
}
```

- kemudian pada mapper StudentMapper, saya membuat method baru bernama selectCourse sebagai mana yang telah saya tulis di studentServiceDatabase

```
@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="selectAllStudentsWithCourse"))
})
```

Method ini berfungsi untuk mengambil data pada course dengan id_course tertentu dan menghubungkannya dengan method selectAllStudentWithCourse yang mana merupakan method berisi student yang mengambil mata kuliah dengan id tersebut.

Property terakhir diset students karena akan mengisi variabel students di kelas CourseModel dan variable column diisi dengan id_course karena nantinya kolom id_course pada database akan dikirimkan sebagai parameter di method selectAllStudentsWithCourse.

- membuat method untuk mengambil seluruh student yang mengambil course dengan id tertentu.

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

Method ini akan mengambil npm, nama, gpa student yang mengambil course tertentu dengan menggabungkan dua tabel student dan studentcourse yang npm dan id_coursenya sama.

- Membuat method untuk view di controller student

```
@RequestMapping("/course/view/{id_course}")  
public String viewCoursePath (Model model,  
    @PathVariable(value = "id_course") String id_course)  
{  
    CourseModel course = studentDAO.selectCourse(id_course);  
  
    if (course != null) {  
        model.addAttribute ("course", course);  
        return "viewcourse";  
    } else {  
        model.addAttribute ("id_course", id_course);  
        return "not-found-course";  
    }  
}
```

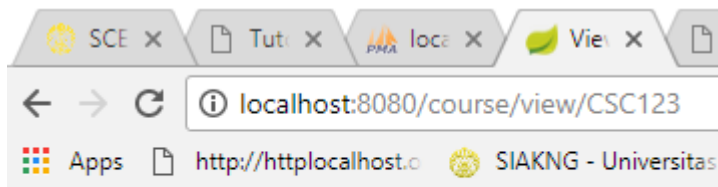
Jika course yang dipilih ada maka akan menampilkan html viewcourse dan jika tidak ada akan menampilkan html not-found-course.

- membuat html untuk viewcourse dan not-found-course

```
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="'Course = ' + ${id_course}">Course id</h3>  
9 </body>  
10 </html>  
11
```

```
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="'Name = ' + ${course.name}">Course Name</h3>
9     <h3 th:text="'Credits = ' + ${course.credits}">Course Credits</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 - Name
15       </li>
16     </ul>
17   </body>
18 </html>
19
20
```

Tampilan jika localhost:8080/course/view/CSC123



ID = CSC123

Name = PSP

Credits = 4

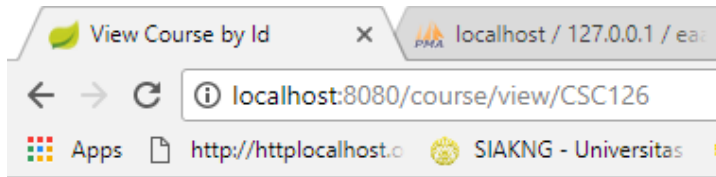
Mahasiswa yang mengambil

- 124-chanek jr.

Menambah mahasiswa yang mengambil CSC126

+ Options	
npm	id_course
123	CSC126
124	CSC123
124	CSC124
124	CSC126

Tampilan jika localhost:8080/course/view/CSC126



ID = CSC126

Name = MPKT

Credits = 6

Mahasiswa yang mengambil

- 123-chanek
- 124-chanek jr.

HAL YANG DIPELAJARI

Pada tutorial 5 ini saya mengenai database dan relasi database dengan menggunakan spring boot. Terdapat tiga database yaitu student, course, dan studentcourse. Studentcourse adalah database yang menghubungkan student dengan course dan database ini menyimpan student mengambil course apa saja. Saya juga mempelajari mengenai anotasi Result, anotasi Result ini digunakan untuk memasukkan hasil query yang di ambil ke dalam kelas tertentu. Selain itu, jika ingin mengisi suatu variabel yang bertipe List namun harus mengambil listnya dari method lain maka, column bisa diisi dengan nama kolom pada database untuk menjadi parameter dimethod yang akan dipanggil untuk mengisi list yang tadi. javaType merupakan class apa yang akan menjadi kembalian dan variabel many diisi dengan method yang nantinya akan mengisi variabel yang bertipe list tadi.