TUTORIAL 5 APAP

Pada tutorial 5 APAP membahas mengenai **Menggunakan Database serta Relasi Database dalam Project Spring Boot** dengan menggunakan dua library eksternal yaitu MyBatis dan Lombok. Jika pada tutorial sebelumnya hanya ada tabel student, maka pada tutorial kali ini akan terdapat tabel baru yaitu tabel course dan tabel studentcourse. Tabel course akan berisi id_course, name, credits. Tabel studentcourse akan menyimpan data student dari tabel student yang mengambil suatu course dari tabel course. Untuk menghubungkan tabel student dan course, maka terlebih dahulu perlu ditambahkan model untuk course dengan menambahkan kelas **CourseModel** pada package com.example.model yang berisi method constructor dengan menambahkan property yang dimiliki course serta List<StudentModel> students karena course memiliki daftar student yang mengambil suatu course. Begitu juga dengan kelas **StudentModel** perlu ditambahkan List<CourseModel> courses karena student memiliki daftar course yang diambil. Inisialisasi constructor pada **StudentController** juga perlu diubah karena menerima 4 parameter.

Pada kelas **StudentMapper** ditambahkan sebuah method yang akan mengembalikan list of course pada student dengan NPM tertentu dengan cara melakukan join antara tabel studentcourse dengan tabel student berdasarkan id_course dan dipilih hanya pada student dengan NPM yang diberikan. Selanjutnya untuk menggabungkan method selectStudent dengan selectCourses agar saat melakukan select List<Course> dapat terisi perlu ditambahkan method sebagai berikut :

```
@Select("select npm, name, gpa from student where npm = #{npm})//memilih npm, nama, gpa dari tabel student jika npm student sama dengan npm yang diberikan
@Results(value = {
@Result(property="npm", column="npm"),//ANOTASI @Result akan memetakan hasil query select ke StudentModel, property diisi dengan variable yang ada pada kelas StudentModel sedangkan variabel diisi dengan nama kolom didatabase atau parameter yang dibutuhkan
@Result(property="name", column="name"),
@Result(property="gpa", column="gpa"),
@Result(property="courses", column="npm",
    javaType = List.class,
    many=@Many(select="selectCourses"))//untuk variabel courses karena diambil hasilnya dari method lain maka variabel kolom diisi dengan npm karena kolom npm pada database akan dikirim menjadi parameter pada selectCourses yang di set menggunakan anotasi @Many
```

```
})

StudentModel selectStudent (@Param("npm") String npm);
```

Selanjutnya pada **view.html** ditambahkan list kuliah yang diambil mahasiswa tersebut

```
<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>
```

HASIL TUTORIAL

http://localhost:8080/student/viewall

All Students

No. 1

NPM = 123

Name = Mia

GPA = 3.7

Delete Data
Update Data

No. 2

NPM = 124

Name = Ririn

GPA = 3.9

Delete Data
Update Data

http://localhost:8080/student/view/123

NPM = 123

Name = Mia

GPA = 3.7

Kuliah yang diambil

- MPKT-6sks

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

NPM = 124

Name = Ririn

GPA = 3.9

Kuliah yang diambil

- PSP-4sks
- SDA-3sks

LATIHAN

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

- **StudentMapper.java**

```
    StudentModel selectStudent (@Param("npm") String npm);

@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 ();
```

Sama seperti yang telah dilakukan pada tutorial maka untuk menampilkan seluruh data student beserta dengan mata kuliah yang diambilnya, maka kita juga perlu mengubah method selectAllStudents() pada kelas **StudentMapper** dengan men-*select* npm, name, gpa dari tabel Student, lalu menggunakan anotasi @Result untuk memetakan hasil dari query select ke kelas **StudentModel**. Setiap property diatur dengan nama variabel yang terdapat pada kelas **StudentModel**, sedangkan coloumn diisi dengan nama kolom hasil query di database (parameter yang dibutuhkan). Parameter yang dibutuhkan untuk selectAllStudents adalah npm, name, gpa dan courses. Untuk mengisi variable courses maka variable coloumn diisi dengan npm karena kolom npm pada database akan dikirim menjadi sebuah parameter yang digunakan oleh method selectCourses. Variabel javaType akan menentukan class yang menjadi kembalian, dan variabel many dengan menggunakan anotasi @Many diset dengan method selectCourses yang akan mengisi variabel courses.

HASIL LATIHAN 1

http://localhost:8080/student/viewall

All Students

No. 1

NPM = 123

Name = Mia

GPA = 3.7

Kuliah yang diambil

No. 2

NPM = 124

Name = Ririn

GPA = 3.9

Kuliah yang diambil

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

- StudentMapper.java

```
@Select("select course.id_course, name, credits " + "from studentcourse join course " +
"on studentcourse.id_course = course.id_course " + "where studentcourse.npm =
#{npm}")
   @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="selectStudent"))
   })

   List<CourseModel> selectCourses (@Param("npm") String npm);
```

```
@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);
```

Untuk menampilkan data course beserta student yang mengambilnya, maka perlu menghubungkan method selectCourses dengan method selectStudent, dimana pada method selectCourses akan di-*select* id_course, name, credits dari tabel studentcourse yang di join dengan tabel course berdasarkan id_course dan dipilih hanya pada student dengan NPM yang diberikan.

```
@Select("select * 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="selectSCourse"))

   })
   CourseModel selectCourse (@Param("id_course") String id_course);
```

```
@Select("select * from student join StudentCourse on student.npm = studentcourse.npm
where studentcourse.id_course = #{id_course}")
   List<StudentModel> selectSCourse(@Param("id_course") String id_course);
```

- StudentService.java

```
CourseModel viewCourse (String idCourse);
```

Didalam kelas interface **StudentService** ditambahkan method viewCourse yang menerima parameter berupa idCourse bertipe String

- StudentServiceDatabase.java

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

Pada kelas **StudentServiceDatabase** dibuat method viewCourse yang mengembalikan method selectCourse pada kelas **StudentMapper**

- StudentController.java

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

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

Pada kelas **StudentController.java** ditambahkan anotasi @RequestMapping("/course/view/{id}") menandakan jika ada request HTTP pada path/course/view/{id} ,maka method viewCourse pada kelas tersebut akan dipanggil dengan mempassing suatu data dari URL menggunakan path variable id. Untuk menampilkan data course dan data student yang mengambil course maka akan dipanggil method studentDAO.view (id_course); yang datanya akan disimpan dalam variable course. Jika course dengan npm yang akan diubah ada/ tidak sama dengan null maka akan mengembalikan ke halaman viewcourse.html sebagai berikut :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by idCourse</title>
    </head>
    <body>
        <form th:object="${course}">
        <h3 th:text="'ID = ' + ${course.idCourse}" th:field="*{id}">ID
        Course</h3>
        <h3 th:text="'Name = ' + ${course.name}" th:field="*{name}">
        Course Name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}" th:field="*{sks}">
        Student Credits</h3>

        <h3>Mahasiswa yang mengambil</h3>
        <ul th:each="students,iterationStatus: ${course.students}">
            <li th:text="${students.npm} + '-' + ${students.name}">

            </li>
        </ul>
        </form>
    </body>
</html>
```

Pada **viewcourse.html th:object** akan menyatakan objek yang akan digunakan untuk mengumpulkan form. Tiga form field yang dinyatakan dengan **th:field** yang sesuai dengan objek course. Objek tersebut akan mencakup controller, model dan view untuk menampilkan form. Selain itu akan ditampilkan mahasiswa yang mengambil course teserbut dengan npm dan nama student.

HASIL LATIHAN 2

http://localhost:8080/course/view/CSC123

ID = CSC123

Name = PSP

SKS = 4

Mahasiswa yang mengambil

- 124-Ririn

http://localhost:8080/course/view/CSC124

ID = CSC124

Name = SDA

SKS = 3

Mahasiswa yang mengambil

- 124-Ririn

http://localhost:8080/course/view/CSC126

ID = CSC126

Name = MPKT

SKS = 6

Mahasiswa yang mengambil

- 123-Mia