Nama : Haryo Parigroho
NPM : 1506689603
Kelas : B

Write-up Tutorial 5

Pada tutorial kali ini saya mempelajari beberapa hal berikut; Dengan menggunakan MyBatis, query yang dieksekusi dapat langsung disimpan pada class model dengan menggunakan annotation **@Results** yang memetakan hasil query ke class model dan **@Resut** yang memetakan hasil query per kolom ke atribut class model. Lalu, annotation **@Many** untuk menyimpan data hasil query yang lebih dari satu.

Pada latihan nomor 1, saya mengubah method **selectAllStudents** pada kelas **StudentMapper** menjadi seperti berikut :

```
@Select("select npm, name, gpa from student")
@Results(value = {
        @Result(property="npm", column="npm", javaType=String.class),
        @Result(property="name", column="name", javaType=String.class),
        @Result(property="gpa", column="gpa", javaType=Double.class),
        @Result(property="courses", column="npm", javaType=List.class, many=@Many(select="selectCourses"))
})
List<StudentModel> selectAllStudents ();
```

Saya menambahkan annotation **@Results** dan **@Result**. **@Results** berfungsi untuk menyimpan hasil query ke class model. **@Result** berfungsi untuk menyimpan hasil query per kolom ke atribut dari class model. Walaupun return type dari method ini adalah List<StudentModel> tetapi class modelnya tetap StudentModel karena query yang dijalankan mengembalikan object StudentModel.

Pada latihan nomor 2, saya membuat;

- Class **CourseModel**

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CourseModel {
    private String idCourse;
    private String name;
    private Integer credits;
    private List<StudentModel> students;
}
```

- Interface **CourseService**

```
public interface CourseService
{
    CourseModel selectCourse(String id);

}
```

- Class **CourseServiceDatabase**

```java
@Slf4j
@Service
public class CourseServiceDatabase implements CourseService{

    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseModel selectCourse(String idCourse) {
        CourseModel courseModel = courseMapper.selectCourse(idCourse);
        log.info("student " + courseModel.getStudents());
        return courseModel;
    }

}
```

- Interface **CourseMapper**

```java
@Mapper
public interface CourseMapper {

    @Select("select id_course, name, credits from course where id_course = #{idCourse}")
    @Results(value = {
        @Result(property="idCourse", column="id_course", javaType=String.class),
        @Result(property="name", column="name", javaType=String.class),
        @Result(property="credits", column="credits", javaType=Integer.class),
        @Result(property="students", column="id_course", javaType=List.class, many=@Many(select="selectStudents"))
    })
    CourseModel selectCourse(@Param("idCourse") String idCourse);

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

- Class **CourseController**

```java
@Controller
public class CourseController {

    @Autowired
    CourseService courseDAO;

    @RequestMapping("/course/view/{id}")
    public String view(Model model,
            @PathVariable(value = "id") String id) {

        CourseModel course = courseDAO.selectCourse(id);

        if (course != null) {
            model.addAttribute ("course", course);
            return "view-course";
```

Class-class dan interface-interface diatas saya buat karena konteks aplikasi sudah berbeda yaitu “course”, dapat dilihat dari mapping yang digunakan (/course/view/{id}). Class-class dan interface-interface tersebut memiliki kegunaan yang sama seperti yang dikerjakan pada tutorial sebelumnya hanya saja menggunakan model **CourseModel.**

Method-method dan file-file berikut ini adalah yang saya buat untuk mengerjakan soal nomor 2:

- Method **selectCourse** pada Interface **CourseMapper.**

```
@Select("select id_course, name, credits from course where id_course = #{idCourse}")
@Results(value = {
    @Result(property="idCourse", column="id_course", javaType=String.class),
    @Result(property="name", column="name", javaType=String.class),
    @Result(property="credits", column="credits", javaType=Integer.class),
    @Result(property="students", column="id_course", javaType=List.class, many=@Many(select="selectStudents"))
})
CourseModel selectCourse(@Param("idCourse") String idCourse);
```

Method ini menjalankan query untuk mengambil data **course** dengan id_course tertentu. Method ini memetakan hasil query ke class model **CourseModel.** Untuk atribut students, memanggil method **selectStudents** pada Interface **CourseMapper.**

- Method **selectStudents** pada Interface **CourseMapper.**

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

Method ini menjalankan query untuk mengambil student-student yang mengambil mata kuliah dengan id_course tertentu.

- Method **view** pada class **CourseController**

```
@RequestMapping("/course/view/{id}")
public String view(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";
    }
}
```

Method ini menghandle mapping (/course/view/{id}), pertama-tama method akan memanggil method **selecCourse** pada service untuk mendapatkan course dengan id tertentu. Jika ada maka menampilkan view-course.html tetapi jika tidak ada maka menampilkan not-found-course.html.

- View-course.html dan 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 = ' + ${id}">Course ID</h3>
    </body>
</html>
```

Nama : Haryo Parigroho
NPM : 1506689603
Kelas : B

```
<!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 Credits</h3>

        <h3>Mahasiswa yang Mengambil: </h3>
        <ul th:each="student, iterationStatus: ${course.students}">
            <li th:text="${student.npm} + ' - ' + ${student.name}">
                NPM - Nama Mata Kuliah
            </li>
        </ul>

    </body>
</html>
```