Tutorial 5 Writeup

Arsitektur dan Pemrograman Aplikasi Perusahaan

Nama : Komang Adelia Nala Ratih

NPM : 1406557541

Kelas : B

Jelaskan apa saja hal yang Anda pelajari dari tutorial ini.

Pada tutorial ini, saya mempelajari penggunaan dua library eksternal, yaitu MyBatis dan Lombok dengan menggunakan anotasi @Results yang berguna untuk memetakan hasil query ke kelas model, anotasi @Result yang berguna untuk memetakan hasil query ke atribut kelas model, serta anotasi @Many untuk menyimpan data hasil query yang berjumlah lebih dari satu.

Jelaskan method yang Anda ubah pada Latihan Merubah SelectAllStudents.

1. Mengubah method **selectAllStudents** pada pada kelas **StudentMapper** yang berguna untuk mengembalikan list of course pada Student.

```
@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();
```

2. Menambahkan view pada **viewall.html** menampilkan list kuliah yang diambil oleh mahasiswa.

```
<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>
```

Jelaskan method yang Anda buat pada Latihan Menambahkan View pada Course.

1. Membuat model berupa class **CourseModel** yang berisi variabel id_course (String) , name (String), credits (Integer), dan students (List<StudentModel>).

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

2. Membuat interface **CourseService** yang mendefinisikan method-method yang dapat dilakukan untuk memanipulasi kelas Course.

```java
public interface CourseService {
    CourseModel selectCourse (String idCourse);
}
```

3. Membuat class **CourseServiceDatabase** yang mengimplementasikan interface CourseService. Method ini akan memanggil method selectCourse dengan idCourse sebagai parameternya. Kemudian, SQL untuk melakukan selectCourse akan dijalankan. Log.info akan menulis log bahwa student berhasil dipilih.

```java
@Slf4j
@Service
public class CourseServiceDatabase implements CourseService{
    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseModel selectCourse(String idCourse) {
            log.info ("select course with idCourse {}", idCourse);
            return courseMapper.selectCourse (idCourse);
    }
}
```

4. Membuat interface **CourseMapper** yang berisi method sebagai berikut:

Method selectCourse

Method selectCourse menerima parameter id_course. Query ini bekerja dengan memilih course yang memiliki id_course yang dimasukkan, lalu memetakan hasil query ke **CourseModel.** Atribut students akan didapatkan dengan memanggil method **selectStudents** pada interface CourseMapper.

```java
@Select("select id_course, name, credits from course where id_course = #{id_course}")
@Results(value = {
        @Result(property = "id_course", 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 ="selectStudents"))})
CourseModel selectCourse(@Param("id_course") String id_course);
```

Method selectStudents

Method selectStudents akan mengambil seluruh student yang mengambil mata kuliah sesuai dengan id_course yang diberikan dengan melakukan join antara tabel studentcourse dengan student berdasarkan npm dan difilter hanya pada course dengan npm yang diberikan.

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

5. Membuat class **CourseController**.

Method viewCourse

Method viewCourse menghandle mapping (/course/view/{id_course}). Method ini bekerja dengan memanggil method **selectCourse** pada service untuk mendapatkan course sesuai dengan id_course yang diberikan. Halaman view-course akan ditampilkan apabila id_course yang diberikan terdapat di database, namun apabila tidak ada, halaman not-found akan ditampilkan.

```
@Controller
public class CourseController {
    @Autowired
    CourseService courseDAO;

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

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