Bintang Glenn J

1506757535

Ringkasan Materi

Pada tutorial kali ini saya belajar lebih jauh mengenai cara berinteraksi dengan basis data beserta dengan relasinya pada program. Kini saya lebih paham untuk mengambil setiap kolom yang diinginkan dari hasil serta mengetahui bagaimana mengasosiasikannya dengan atribut dari objek.

**Latihan Merubah selectAllStudents**

Latihan ini sebenarnya tidak jauh berbeda dengan tutorial yang diajarkan sebelumnya.

Saya mengubah method selectAllStudents dari:

```
@Select("select npm, name, gpa from student")
List<StudentModel> selectAllStudents ();
```

Menjadi:

```
@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 ();
```

Karena course yang diambil setiap student akan ditampilkan, maka akan digunakan method selectCourses yang telah dibuat untuk mengambil mata kuliah yang diambil student tersebut dengan menggunakan NPM miliknya.

```
@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 sendiri akan mengembalikan sebuah list berisi objek dari class CourseModel yang didapat dengan menjalankan *query* pada anotasi @Select. *Query* tersebut akan mengambil informasi dari course yang diambil oleh student tersebut dengan melakukan join tabel course dengan studentcourse kemudian mengambil baris yang memiliki NPM yang sesuai.

Saya juga menambahkan beberapa baris pada viewall.html untuk dapat menampilkan mata kuliah yang diambil:

```
<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>
```

**Latihan Menambahkan View pada Course**

Pertama, saya menambahkan baris berikut pada interface StudentService:

```
CourseModel selectCourse (String id);
```

Kemudian saya menambahkan pemetaannya ke StudentMapper di class StudentServiceDatabase:

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

Method tersebut akan memanggil method selectCourse yang ada di StudentMapper, yang berisi:

```
@Select("select id_course, name, credits from course where id_course = #{id}")
@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="selectStudentsFromCourse"))
})
CourseModel selectCourse (@Param("id") String id);
```

Method tersebut akan mengambil informasi dari course sesuai dengan id yang diberikan dan mengembalikannya dalam bentuk objek dari CourseModel. Untuk mendapatkan semua student yang mengambil course tersebut, saya menggunakan method selectStudentsFromCourse yang saya buat dengan parameter id course tersebut. Method selectStudentsFromCourse sendiri berisi seperti berikut:

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

*Query* yang saya gunakan akan melakukan join tabel studentcourse dengan student untuk mendapatkan informasi student yang mengambil course sesuai dengan id course yang dimasukkan.

Pada controller, saya menambahkan method viewCoursePath untuk memetakan *request* yang masuk ke view. Method viewCoursePath berisi seperti berikut:

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

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

viewCourse sendiri hanya berisi sebagai berikut:

```
<!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="'Name = ' + ${course.name}">Course Name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">Course Credit</h3>

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

Sedangkan not-found-course berisi:

```
<!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 Course = ' + ${id}">Course ID</h3>
    </body>
</html>
```

Kedua file html tersebut tidak jauh berbeda dengan yang telah dibuat sebelum-sebelumnya.