Tutorial 5

Menggunakan Database serta Relasi Database dalam Project Spring Boot

Write-up:

Pada tutorial ke-5 ini, saya mempelajari bagaimana cara memetakan hasil query yang dilakukan pada database dengan atribut-atribut yang ada pada kelas model. Selain itu, saya juga mempelajari bagaimana mendapatkan hasil query hasil penggabungan lebih dari satu tabel yang dilakukan pada kelas *mapper*.

Penjelasan Method-Method pada Latihan Merubah SelectAllStudents

1. Method selectAllStudents pada *interface* **StudentMapper**.

```
@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 ();
```

   Method selectAllStudents ini berfungsi untuk mengambil data-data student yang ada pada database untuk kemudian ditampilkan pada halaman web. Anotasi result digunakan untuk memetakan hasil query yang telah dilakukan dengan kelas Studentmodel. Pada properti “courses” menggunakan nilai kolom “npm” dikarenakan pada method yang akan dipanggil nanti, yaitu method selectCourses membutuhkan parameter npm. Lalu, hasil dari pemanggilan method selectCourses tersebut nantinya akan mengembalikan daftar *course* yang diambil oleh masing-masing mahasiswa yang akan dikembalikan dalam bentuk list.

Penjelasan Method-Method pada Latihan Menambahkan View pada Course

1. Method viewPath pada kelas **CourseController**.

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

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

   Method viewPath ini berfungsi untuk mengambil id course yang telah diinput oleh *user* melalui URL. Method ini nantinya akan melakukan pengecekan pada id course tersebut. Jika *course* dengan id yang telah dimasukkan tersedia, maka akan ditampilkan detail mengenai *course* tersebut pada kelas **viewcourse.html**. Tetapi, jika id tidak tersedia maka halaman akan dialihkan pada **course-not-found.html** yang memberikan keterangan bahwa *course* dengan id yang dimaksud tidak ditemukan.

2. Method selectCourse pada kelas **CourseServiceDatabase**.

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

   Method selectCourse yang sebelumnya dipanggil oleh kelas CourseController berfungsi untuk meneruskan id course yang telah diinput oleh *user* ke kelas CourseMapper, yang nantinya akan dilakukan proses lebih lanjut untuk melakukan pengambilan data di database.

3. Method selectStudents pada *interface* **CourseMapper**.

```
@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);
```

Method selectStudents pada *interface* ini berfungsi untuk mengambil data-data student yang mengambil *course* dengan id yang telah diinput oleh *user* sebelumnya. Hasil dari method ini nantinya akan dipetakan pada kelas CourseModel yang dilakukan pada method selectCourse di *interface* yang sama.

4. Method selectCourse pada *interface* **CourseMapper**.

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

Method selectCourse berfungsi untuk memberikan data-data *course* pada database dengan id course yang dimaksud oleh *user*. Anotasi result juga digunakan pada method ini yang berfungsi untuk memetakan hasil query dengan kelas CourseModel. Pada property “students” digunakan kolom dengan nilai “id_course” dikarenakan akan dilakukan pemanggilan method selectStudents untuk mendapatkan hasil data-data mahasiswa yang mengambil *course* dimaksud, yang mana membutuhkan nilai id course sebagai parameternya.