Muhammad Umar Farisi (1506725552)

Tutorial 5

Apap – C

Hal yang dipelajari di tutorial 5

Pada tutorial ini saya belajar mengani *query* pada suatu method di kelas *mapper* yang melibatkan method lain dalam melengkapi hasil dari method tersebut. Contohnya, method **selectCourse** pada kelas **StudentMapper** yang menggunakan method **selectCourse** pada kelas yang sama untuk melengkapi hasil kembalian dari method selectCourse.

Method pada Latihan Merubah SelectAllStudent

Method **selectCourse** pada kelas **StudentMapper** akan mengambalikan daftar Course yang diambil oleh siswa dengan npm yang diberikan pada parameter method ini. Berikut ini *screenshot* dari method ini:

```
@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 **selectStudent** pada *interface* **StudentMapper** akan mengembalikan *object* **StudentModel**. *Object* tersebut memiliki atribut daftar Course yang harus diisi. Oleh karena itu, terdapat *annotation* Result yang berfungsi untuk memetakan nilainya, atribut **courses**, dengan hasil dari method **selectCourse** pada kelas yang sama dengan method selectStudent. Berikut ini *screenshot* dari method ini:

```
@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);
```

Method pada Latihan Menambahkan View pada Course

Method **selectStudents** pada *interface* **CourseMapper**. Method ini berfungsi untuk mengambil semua StudentModel dari database yang mengambil suatu Course dengan id berdasarkan parameter method ini. Berikut ini *screenshot* dari method ini:

```
@Select("SELECT * FROM student, studentcourse "
        + "WHERE student.npm = studentcourse.npm AND studentcourse.id_course = #{id_course}")
List<StudentModel> selectStudents(@Param("id_course") String id_course);
```

Method **selectCourse** pada kelas **CourseMapper**. Method ini berfungsi untuk mengambil suatu data CourseModel dengan id berdasarkan parameter method ini. CourseModel tersebut diisi dengan daftar StudentModel dengan memanfaatkan method selectStudents pada kelas yang sama. Berikut ini *screenshot* dari method ini:

```
@Select("SELECT * "
        + "FROM course "
        + "WHERE course.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(value = "id_course") String idCourse);
```

Method **selectCourse** pada *interface* **CourseService** yang akan diimplementasi pada kelas CourseServiceDatabase. Berikut ini *screenshot* dari method ini:

```
CourseModel selectCourse(String idCourse);
```

Method **selectCourse** pada kelas **CourseServiceDatabase**. Method ini berfungsi untuk mengambil data CourseModel berdasarkan idnya dari *object* courseMapper. Berikut ini *screenshot* dari method ini:

```
@Override
public CourseModel selectCourse(String idCourse) {
    return courseMapper.selectCourse(idCourse);
}
```

Method **view** pada kelas **CourseController**. Method ini berfungsi untuk mengarahkan / mengontrol tampilan ketika pengguna menuju *path* course/view/{id} dengan id dari course yang ingin dilihat pengguna. Method ini akan mengecek terlebihdahulu apakah Course yang ingin dilihat oleh pengguna ada atau tidak. Jika ada, tampilan yang akan muncul adalah informasi mengenai Course tersebut. Jika tidak ada, tampilan yang akan muncul adalah informasi mengenai tidak adanya Course yang dicari di database. Berikut ini *screenshot* dari method ini:

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

    CourseModel course = courseDAO.selectCourse(idCourse);

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