WRITE UP Tutorial 5

Menggunakan Database serta Relasi Database dalam Project Spring Boot

I. Hal yang telah dipelajari

- Dapat menambahkan model, controller, dan view dari dua objek yang berbeda (student dan course).
- Dapat mengetahui kegunaan dari anotasi @Result, variabel property, variabel column, variabel javaType dan variabel many.
- Dapat mengetahui keterhubungan antara dua objek yang berbeda melalui MVC.

II. Method yang diubah pada latihan merubah selectAllStudents

- Method selectAllStudents()

```
@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 ();
```

Pada method ini saya menambahkan beberapa dibagian anotasi @Result dimana variabel npm, name, dan gpa yang telah didapat dari query agar disimpan di instance variabel courses yang ada di class StudentModel. Property diisi dengan nama variabel yang ada di class StudentModel sedangkan column di property courses diisi npm dimana np mini nantinya akan digunakan sebagai parameter di method selectCourses.

III. Method yang diubah pada latihan menambahkan view pada Course

Untuk menambahkan view pada Course diperlukan penambahan semua komponen MVC mulai dari CourseModel, CourseMapper, CourseService, CourseServiceDatabase, dan courseController. Selain itu ada juga penambahan view dengan menambahkan viewcourse.html dan course-not-found.html. Berikut ini adalah penjelasan dari setiap method yang telah saya buat.

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

public String viewCoursePath (Model model,
      @PathVariable(value = "id") String id)
{
       CourseModel course = courseDAO.selectCourse(id);

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

```
        } else {
                model.addAttribute("course", course);
                return "course-not-found";
        }
}
```

Method ini berfungsi sebagai controller untuk menampilkan view dari setiap course yang ada. Mirip seperti method view yang ada di StudentController hanya berbeda parameter dan Model yang digunakan.

- Method selectStudents()

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

Method ini berfungsi untuk mengambil data dari database menggunakan query yang telah didefinisikan di dalam anotasi @Select.

- Method selectCourse()

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

Method ini berfungsi untuk mengambil informasi tentang mata kuliah tersebut serta data-data student yang mengambil mata kuliah tersebut. Untuk anotasi @Result dengan property student dan column id_courses digunakan untuk menyimpan hasil method selectStudents kedalam variabel students yang ada di class CourseModel.