Bintang Glenn J

1506757535

Hal yang dipelajari

Pada tutorial kali ini saya belajar mengenai *web service* dan bagaimana menggunakannya pada aplikasi. Saya kini lebih memahami bagaimana cara membuat *web service* menggunakan Spring Boot. Dengan menggunakan *web service,* akan lebih mudah untuk memisahkan antara *layer backend* dan *frontend*.

Latihan 1

Service untuk mengembalikan seluruh student tidak jauh berbeda dengan *method* viewall sebelumnya. Karena file REST controller telah dibuat sebelumnya, maka untuk membuat *service* ini hanya tinggal mengimplementasikan kodenya pada file tersebut. Adapun implementasinya cukup singkat:

```
@RequestMapping("/student/viewall")
public List<StudentModel> view() {
    List<StudentModel> students = studentService.selectAllStudents();
    return students;
}
```

Hal yang dilakukan hanya menambahkan *request mapping* ke /student/viewall kemudian membuat *method* yang akan memanggil *method* selectAllStudents pada StudentService kemudian mengembalikannya.

Latihan 2

Pada latihan kedua diminta untuk membuat *service* untuk *class* Course. Langkah yang dilakukan tidak jauh berbeda dengan pembuatan *service* untuk *class* Student. Pertama-tama dibuat REST controller untuk *class* Course pada *package* yang sama dengan REST controller untuk Student kemudian masukkan implementasinya pada *controller* tersebut, seperti berikut:

```
@RestController
@RequestMapping("/rest")
public class CourseRestController {
    @Autowired
    StudentService studentService;

    @RequestMapping("/course/view/{id}")
    public CourseModel view(@PathVariable(value = "id") String id) {
        CourseModel course = studentService.selectCourse(id);
        return course;
    }

    @RequestMapping("/course/viewall")
    public List<CourseModel> viewAll() {
        List<CourseModel> courses = studentService.selectAllCourse();
        return courses;
    }
}
```

Implementasinya tidak jauh berbeda dengan *service* untuk student. Untuk mendapatkan course dengan id tertentu, *method* yang dibuat akan ditambahkan *request mapping* ke course/view/{id} sesuai dengan idnya. Kemudian akan dikembalikan hasil dari pemanggilan *method* selectCourse dengan parameter id pada StudentService. Untuk mendapatkan semua course, *method* yang dibuat akan ditambahkan *request mapping* course/viewall. *Method* tersebut akan mengembalikan hasil dari pemanggilan *method* selectAllCourse pada StudentService.

Latihan 3

Pengimplementasian *consumer* untuk melihat semua student hanya tinggal melengkapi *method* selectAllStudents yang ada pada *class* StudentServiceRest. Adapun *method* tersebut hanya tinggal mengembalikan hasil dari *method* selectAllStudents pada StudentDAO:

```
@Override
public List<StudentModel> selectAllStudents() {
    log.info("REST - select all students");
    return studentDAO.selectAllStudents();
}
```

Pada *class* StudentDAOImpl diimplementasikan *method* selectAllStudents tersebut yang akan mendapatkan list berisi student yang dihasilkan oleh *producer*. Tidak jauh berbeda dari *method* selectStudent, digunakan RestTemplate yang akan mendapatkan objek dari url yang diberikan:

```
@SuppressWarnings("unchecked")
@Override
public List<StudentModel> selectAllStudents() {
    List<StudentModel> students = restTemplate.getForObject("http://localhost:8080/rest/student/viewall", List.class);
    return students;
}
```

Digunakan @SuppressWarnings untuk menekan *warning* yang muncul karena konversi dari List ke List<StudentModel> yang tidak dicek.

Latihan 4

Pada latihan kali ini diminta untuk implementasi *service consumer* untuk *class* CourseModel dengan membuat *class* DAO sendiri. Langkah pertama adalah membuat *interface* CourseDAO yang berisi 2 *method* seperti berikut:

```
public interface CourseDAO {
    CourseModel selectCourse(String id);
    List<CourseModel> selectAllCourse();
}
```

Kemudian dibuat *class* CourseDAOImpl yang mengimplementasikan *interface* tersebut. Implementasi 2 *method* pada *class* tersebut sebenarnya tidak jauh berbeda dengan yang ada pada *class* StudentDAOImpl yang telah dibuat sebelumnya. Baik *method* selectCourse dan

selectAllCourse akan mengambil hasil yang disediakan oleh *producer* melalui url yang sesuai. Implementasinya adalah sebagai berikut:

```java
@Service
public class CourseDAOImpl implements CourseDAO {

    @Autowired
    private RestTemplate restTemplate;

    @Override
    public CourseModel selectCourse(String id) {
        CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/" + id, CourseModel.class);
        return course;
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<CourseModel> selectAllCourse() {
        List<CourseModel> courses = restTemplate.getForObject("http://localhost:8080/rest/course/viewall", List.class);
        return courses;
    }
}
```

Implementasi ini akan dipanggil dari StudentServiceRest. Karena *service* tersebut mengimplementasikan *interface* StudentService yang juga berisi *method* untuk course, maka menurut saya tidak perlu dibuat *service* baru. Implementasinya adalah sebagai berikut:

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

@Override
public List<CourseModel> selectAllCourse() {
    log.info("REST - select all course");
    return courseDAO.selectAllCourse();
}
```