Tutorial 7 – Membuat *Web Service* Menggunakan *Spring Boot Framework*

1. Latihan 1

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

Method ini akan mengembalikan sebuah *object List* berisi *StudentModel* hasil kembalian dari *method selectAllStudents* pada *StudentService*. *Object* hasil kembalian dari *method viewAll* ini akan diubah ke dalam bentuk JSON, karena *class* dari *method* ini merupakan sebuah *Rest Controller*.

2. Latihan 2

Pertama, pada *CourseMapper* saya membaut sebuah *method* untuk mengembalikan sebuah *List of CourseModel* yang berisi semua *course* yang ada. Adapun *method*-nya adalah sebagai berikut:

```
@Select("SELECT course.id_course, course.name, course.credits "
        + "FROM 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"))
        })
List<CourseModel> selectAllCourses ();
```

Kemudian, pada *CourseService* dan *CourseServiceDatabase* saya juga membuat *method-method* bersangkutan.

```
List<CourseModel> selectAllCourses ();
```

```
@Override
public List<CourseModel> selectAllCourses () {
    log.info("DB - Select all courses");
    return courseMapper.selectAllCourses();
}
```

Kemudian, saya membuat sebuah *class RestController* dengan nama *CourseRestController* sebagai berikut:

```
@RestController
@RequestMapping("/rest")
public class CourseRestController
{
    @Autowired
    CourseService courseService;

    @RequestMapping("/course/view/{idCourse}")
    public CourseModel viewCourse (Model model,
            @PathVariable(value = "idCourse") String idCourse) {
        return courseService.selectCourse (idCourse);
    }

    @RequestMapping("/course/viewall")
    public List<CourseModel> viewAllCourses() {
        return courseService.selectAllCourses();
    }
}
```

Method viewCourse akan mengembalikan sebuah *object* berupa *CourseModel* ketika *user* mengakses (/rest/course/view/{idCourse}). *Method viewAllCourses* akan emngembalikan sebuah *object* berupa *List of CourseModel* yang berisi semua *course* yang ada ketika *user* mengakses (/rest/course/viewall). Kembalian dari kedua *method* ini akan diubah kedalam bentuk JSON untuk digunakan dalam REST API.

3. Latihan 3

Pada *StudentDAOImpl*, saya mengimplemetnasikan *method selectAllStudents* sebagai berikut:

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

Method ini menggunakan *method getForObject* dari *object restTemplate* ke URL "http://localhost:8080/rest/student/viewall" yang akan mengembalikan sebuah *object* berupa *List of StudentModel* yang berisi keseluruhan *student* yang ada.

4. Latihan 4

Pertama, saya membuat sebuah *class interface DAO* dengan nama *CourseDAO* seperti berikut:

```
public interface CourseDAO
{
    CourseModel selectCourse (String idCourse);
    List<CourseModel> selectAllCourses ();

}
```

Kemudian, saya membuat implementasi dari *interface* tersebut pada sebuah *class* bernama *CourseDAOImpl*.

```
@Service
public class CourseDAOImpl implements CourseDAO
{
@Autowired
private RestTemplate restTemplate;

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

    @Override
    @SuppressWarnings("unchecked")
    public List<CourseModel> selectAllCourses ()
    {

        List<CourseModel> courseList =
                Collections.checkedList(restTemplate.getForObject("http://localhost:8080/rest/course/viewall",
                List.class), CourseModel.class);
        return courseList;
    }
}
```

Cara kerja *class* ini hampir sama dengan *class StudentDAOImpl*, namun *class* ini merupakan *class* untuk *course*. Prinsip kerjanya adalah kedua *method* ini akan mengembalikan *object* yang didapatkan dengan mengakses URL yang mengembalikan *object* dari *Producer*.

Lalu, saya membuat sebuah *class service* untuk REST.

```
@Slf4j
@Service
@Primary
public class CourseServiceRest implements CourseService
{
    @Autowired
    private CourseDAO courseDAO;

    @Override
    public List<CourseModel> selectAllCourses () {
        log.info ("REST - select students by course");
        return courseDAO.selectAllCourses();
    }

    @Override
    public CourseModel selectCourse (String idCourse)
    {
        log.info ("REST - Select course with idCourse {}", idCourse);
        return courseDAO.selectCourse(idCourse);
    }

}
```

Method selectAllCourses akan mengembalikan *List of CourseModel* dari semua *course* yang ada. Sedangkan *method selectCourse* akan mengembalikan sebuah *object* berupa *CourseModel* berdasarkan *idCourse*.

Karena *class* ini merupakan *Primary*, maka pada *controller* mengutamakan penggunaan *class* ini.

5. Hal yang dipelajari

Pada *tutorial* ini, saya mempelejari mengenai REST API dan cara menggunakannya untuk membuat sebuah RESTful *Web Service* pada *Spring Boot*. Saya mempelajari struktur, cara kerja, implementasi, dan penggunaan dari *web service*. Selain itu, saya juga memplejari cara membuat suatu *Producer* dan *Consumer* untuk *web service* dan cara untuk menangkap REST API dan menggunakannya untuk dijadikan suatu *object* pada Java.