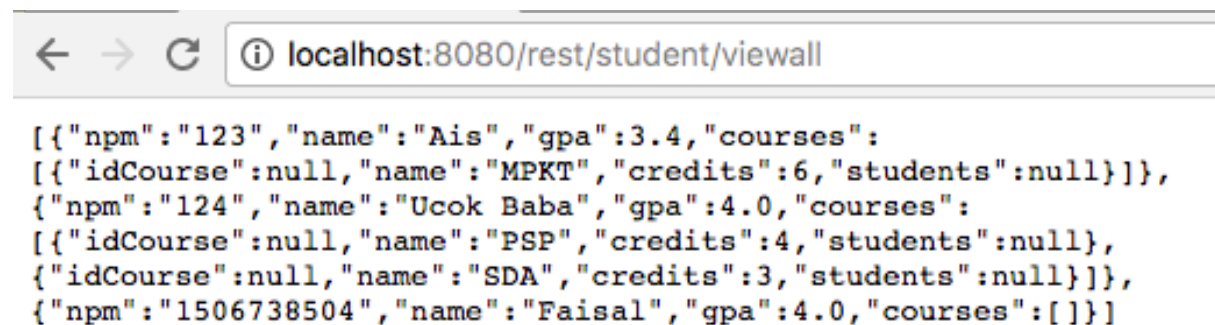


## Write Up Tutorial 7

### Latihan 1:

Buatlah service untuk mengembalikan seluruh student yang ada di basis data. Service ini mirip seperti method `viewAll` di Web Controller. Service tersebut di-mapping ke `"/rest/student/viewall"`

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



Tidak jauh berbeda dengan view sebelumnya, kali ini dengan menggunakan method `selectAllStudents` yang mengembalikan list dari semua siswa.

### Latihan 2:

Buatlah service untuk class `Course`. Buatlah controller baru yang terdapat service untuk melihat suatu course dengan masukan ID Course (view by ID) dan service untuk elihat semua course (view all)

Caranya tidak jauh berbeda dengan yang dilakukan untuk student. Pada saat ingin menampilkan semua course, ditambahkan query di mapper seperti yang tercantum di bawah.

```
CourseDAOImpl.java StudentRestController.java StudentMapper.java CourseRestController.java
1 package com.example.rest;
2
3 import java.util.List;
12
13 @RestController
14 @RequestMapping("/rest")
15 public class CourseRestController
16 {
17
18     @Autowired
19     CourseService courseService;
20
21     @RequestMapping("/course/view/{id}")
22     public CourseModel view (@PathVariable(value = "id") String id) {
23         CourseModel course = courseService.selectCourse(id);
24         return course;
25     }
26
27     @Autowired
28     @RequestMapping("/course/viewall")
29     public List<CourseModel> viewAll () {
30         List<CourseModel> courses = courseService.selectAllCourse();
31         return courses;
32     }
33
34
35 }
```

```
localhost:8080/rest/course/view/CSC125

{"idCourse": "CSC125", "name": "DDP", "credits": 4, "students": []}

@Select("select * 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="selectAllStudentsWithCourse"))
})
List<CourseModel> selectAllCourse();
```

```
localhost:8080/rest/course/viewall

[{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [{"npm": "124", "name": "Ucok Baba", "gpa": 4.0, "courses": null}]}, {"idCourse": "CSC124", "name": "SDA", "credits": 3, "students": [{"npm": "124", "name": "Ucok Baba", "gpa": 4.0, "courses": null}]}, {"idCourse": "CSC125", "name": "DDP", "credits": 4, "students": []}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": [{"npm": "123", "name": "Ais", "gpa": 3.4, "courses": null}]}]
```

Implementasikan service consumer untuk view all Students dengan melengkapi method selectAllStudents yang ada di kelas StudentServiceRest.

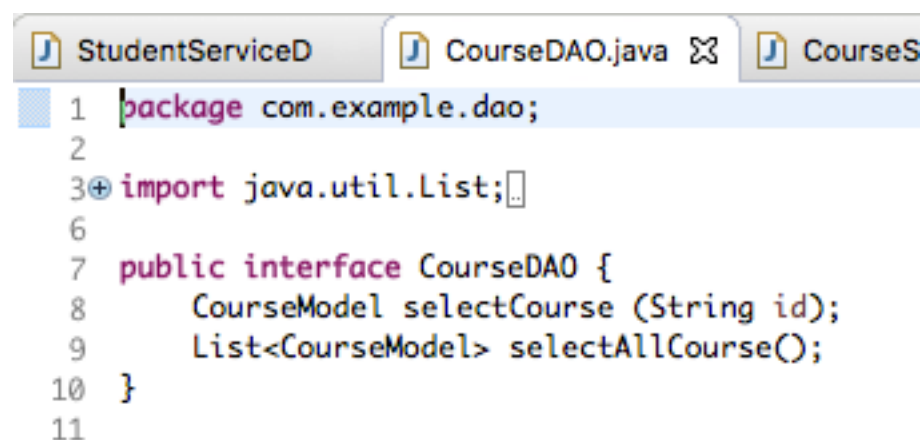
```
@Override
public List<StudentModel> selectAllStudents ()
{
    Log.info ("REST - select all students");
    return studentDAO.selectAllStudents ();
}
```

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

#### Latihan 4:

Implementasikan service consumer untuk class CourseModel dengan membuat class-class DAO dan service baru.

Sebelum membuat CourseDAOImpl, kita membuat interface terlebih dahulu. Kemudian CourseDaoImpl mengimplementasikan CourseDAO yang menjadi tempat untuk mengambil object dari operasi REST menuju localhost:8080. Kemudian di kelas CourseServiceRest mengimplement CourseService agar bisa digunakan pada Controller. Terakhir, menambahkan method viewAll pada CourseController.



```
1 package com.example.dao;
2
3 import java.util.List;
4
5
6
7 public interface CourseDAO {
8     CourseModel selectCourse (String id);
9     List<CourseModel> selectAllCourse();
10 }
11
```

```
CourseDAOImpl.j StudentServiceD CourseDAO.java CourseServiceRe StudentServiceR StudentDAOImpl.
1 package com.example.service;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import org.springframework.web.client.RestTemplate;
9
10 import com.example.dao.CourseDAO;
11 import com.example.model.CourseModel;
12
13 @Service
14 public class CourseDAOImpl implements CourseDAO
15 {
16     @Autowired
17     private RestTemplate restTemplate;
18
19     @Override
20     public CourseModel selectCourse (String id_course)
21     {
22         CourseModel course =
23             restTemplate.getForObject("http://localhost:8080/rest/course/view/"+id_course, CourseModel.class);
24         return course;
25     }
26
27     @Override
28     public List<CourseModel> selectAllCourse()
29     {
30         List<CourseModel> courses = restTemplate.getForObject("http://localhost:8080/rest/course/viewall", List.class);
31         return courses;
32     }
33 }
```

```
CourseDAOImpl.j StudentServiceD CourseDAO.java CourseServiceRe
1 package com.example.service;
2
3 import java.util.List;
4
5 @Slf4j
6 @Service
7 @Primary
8 public class CourseServiceRest implements CourseService
9 {
10     @Autowired
11     private CourseDAO courseDAO;
12
13     @Override
14     public CourseModel selectCourse(String id_course) {
15         log.info("REST - select course with id_course {}", id_course);
16         return courseDAO.selectCourse(id_course);
17     }
18
19     @Override
20     public List<CourseModel> selectAllCourse() {
21         log.info("REST - select all course");
22         return courseDAO.selectAllCourse();
23     }
24
25 }
26
```

Faisal Satrio Priatmadji  
1506738504  
ADPAP-C

### **LESSON LEARNED:**

Pada tutorial ini saya belajar mengenai cara bertukar data antar project dengan Springboot, yaitu berguna untuk membedakan antara interface *consumer* dan interface *producer*. Saya juga belajar menggunakan REST pada Springboot serta cara kerjanya.