

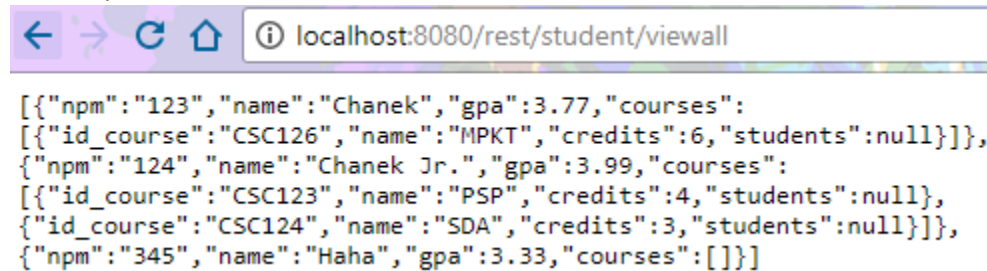
Pada tutorial 7, saya belajar menggunakan web service. Saya belajar memisah layer backend (service producer), aplikasi yang memberi data pada service consumer, dan frontend (service consumer), aplikasi yang berinteraksi dengan pengguna.

## LATIHAN

1. Pertama tambahkan viewall pada class StudentRestController. Dengan isi

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

Hasilnya adalah object StudentModel yang merupakan hasil return dari method dalam format JSON seperti berikut:



The screenshot shows a web browser address bar with the URL `localhost:8080/rest/student/viewall`. Below the address bar, the JSON response is displayed:

```
[{"npm":"123","name":"Chanek","gpa":3.77,"courses":
[{"id_course":"CSC126","name":"MPKT","credits":6,"students":null}]},
{"npm":"124","name":"Chanek Jr.","gpa":3.99,"courses":
[{"id_course":"CSC123","name":"PSP","credits":4,"students":null},
{"id_course":"CSC124","name":"SDA","credits":3,"students":null}]},
{"npm":"345","name":"Haha","gpa":3.33,"courses":[]}]
```

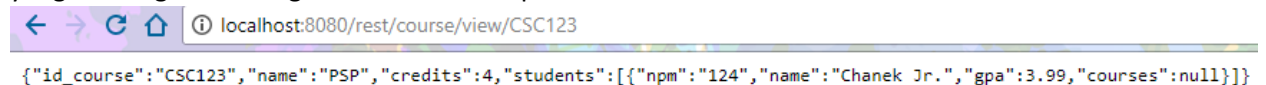
2. Membuat class CourseRestController yang berisi viewCourse dengan parameter id\_course dan viewallCourse.

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

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

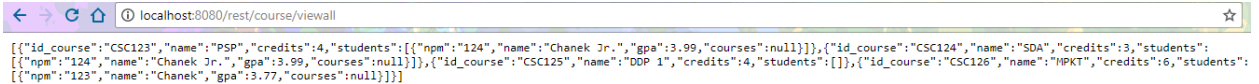
    @RequestMapping("/course/viewall")
    public List<CourseModel> viewallCourse () {
        List<CourseModel> courses = courseService.selectAllCourses();
        return courses;
    }
}
```

Dengan adanya class tersebut, jika membuka link `localhost:8080/rest/course/view/{id_course}` atau `localhost:8080/rest/course/viewall`, yang dikembalikan merupakan object CourseModel yang bersangkutan dengan format JSON seperti di bawah



The screenshot shows a web browser address bar with the URL `localhost:8080/rest/course/view/CSC123`. Below the address bar, the JSON response is displayed:

```
{"id_course":"CSC123","name":"PSP","credits":4,"students":[{"npm":"124","name":"Chanek Jr.","gpa":3.99,"courses":null}]}
```



```

[{"id_course":"CSC123","name":"PSP","credits":4,"students":[{"npm":"124","name":"Chanek Jr.","gpa":3.99,"courses":null}],{"id_course":"CSC124","name":"SDA","credits":3,"students":[{"npm":"124","name":"Chanek Jr.","gpa":3.99,"courses":null}],{"id_course":"CSC125","name":"DDP 1","credits":4,"students":[]}, {"id_course":"CSC126","name":"MPKT","credits":6,"students":[{"npm":"123","name":"Chanek","gpa":3.77,"courses":null}]}

```

- Melengkapi method `selectAllStudents` dengan memanggil method `selectAllStudents` yang terdapat pada class `StudentDAO`. Class ini mengembalikan list berisi semua student.

```

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

```

Isi dari method `selectAllStudents` pada `StudentDAOImpl` yang mengimplementasi `StudentDAO` adalah

```

public List<StudentModel> selectAllStudents() {
    ResponseEntity<List<StudentModel>> response =
    restTemplate.exchange("http://localhost:8080/rest/student/viewall",
    HttpMethod.GET, null, new
    ParameterizedTypeReference<List<StudentModel>>() {});
    List<StudentModel> students = response.getBody();

    return students;
}

```

`ResponseEntity` akan menyimpan hasil dari rest dengan url yang diinginkan dalam format JSON. `ParameterizedTypeReference` adalah untuk passing data bertipe generic. `getBody()` berguna untuk mengambil data dalam format JSON yang dihasilkan oleh response.

- Membuat interface `CourseDAO` yang berisi 2 method yaitu `selectCourse` dengan parameter `id_course` dan `selectAllCourses`. Kedua, membuat class `CourseDAOImpl` yang mengimplementasi interface `CourseDAO`.

```

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

```

`GetForObject` adalah untuk mendapat data object yang terdapat di url yang menjadi parameter bersama dengan tipe objectnya.

Tahap berikutnya adalah membuat interface `CourseService` dan class `CourseServiceRest` yang mengimplementasi `CourseService` pada package `Service`. Method yang terdapat di `CourseServiceRest` akan memanggil method yang terdapat pada class `CourseDAO` yang dibuat di awal.