

Nama : Wita Aristawidya (1506689300)

Kelas : APAP – A

## Write Up Tutorial 7

### Hal-hal yang telah dipelajari di tutorial 7

Antar aplikasi dapat saling berinteraksi. Salah satu caranya adalah *web service*. *Web service* ini dapat dikatakan mirip seperti proses komunikasi pada *client – server*. Dalam *web service* ini, terdapat dua bagian, yaitu ada yang bertindak sebagai *producer* dan juga ada yang bertindak sebagai *consumer*. *Producer* menyediakan layanan *web service*, sedangkan *consumer* yang mengirim *request* atau menggunakan layanan *web service* tersebut. Dengan adanya *web service*, *consumer* lebih memudahkan untuk mengambil data dari database yang dimiliki oleh *producer*. Sehingga *consumer* dapat lebih fokus terhadap bisnis proses dan *view* yang aplikasi tersebut kembangkan, dan *producer* juga dapat lebih fokus terhadap data dimana banyak aplikasi lain yang dapat mengakses data tersebut dengan mudah.

### Latihan 1

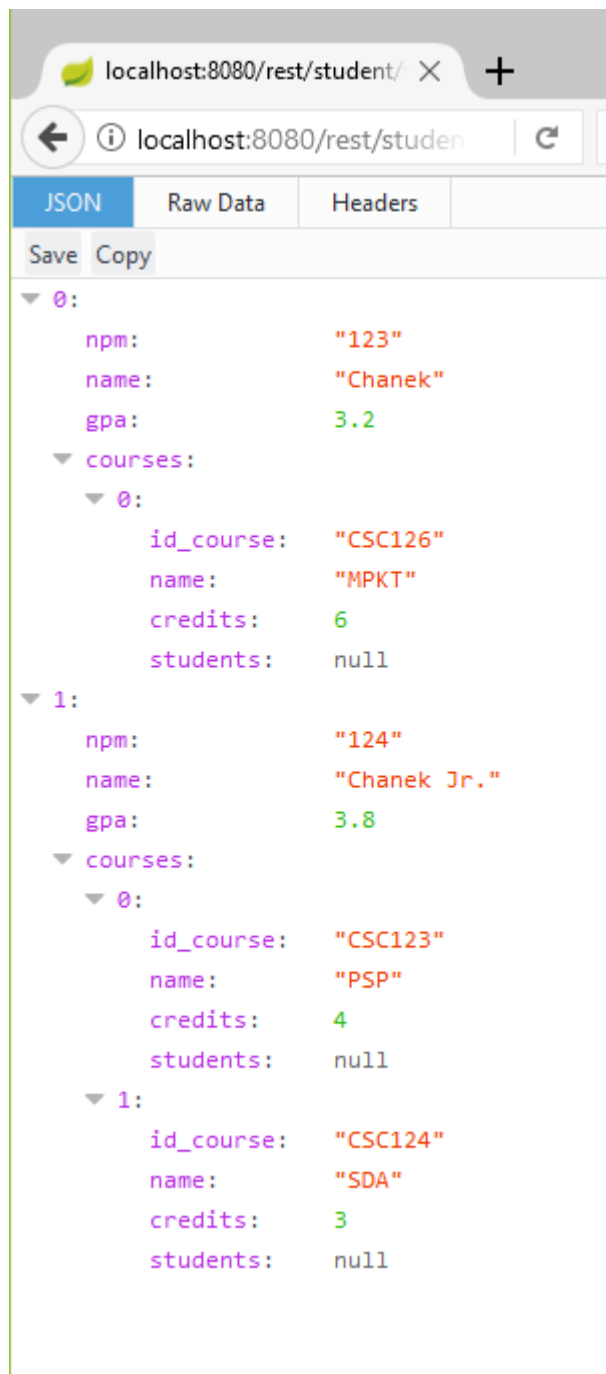
✓ *Service* viewAll untuk student

Untuk mengembalikan List StudentModel yang diminta pada *web service*, maka pada *producer* dibuat method untuk request mapping */rest/student/viewall*. Method tersebut akan mengembalikan List students yang diambil dari database.

-

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

Jika */rest/student/viewall* dijalankan, maka akan tampil hasil List berbentuk JSON seperti di bawah ini.



## Latihan 2

### ✓ Service untuk Course

Sama seperti view student dan view all student, maka di dalam controller dibuat method untuk mengembalikan course dan courses yang diambil dari database melalui method selectCourse dan selectAllCourse.

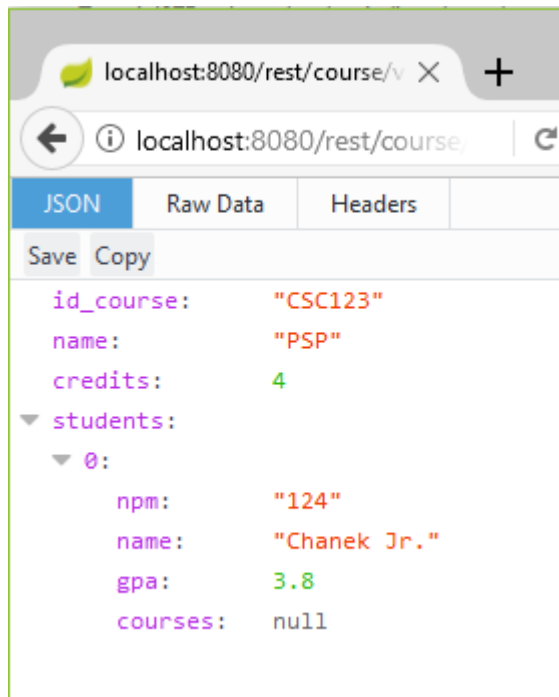
```

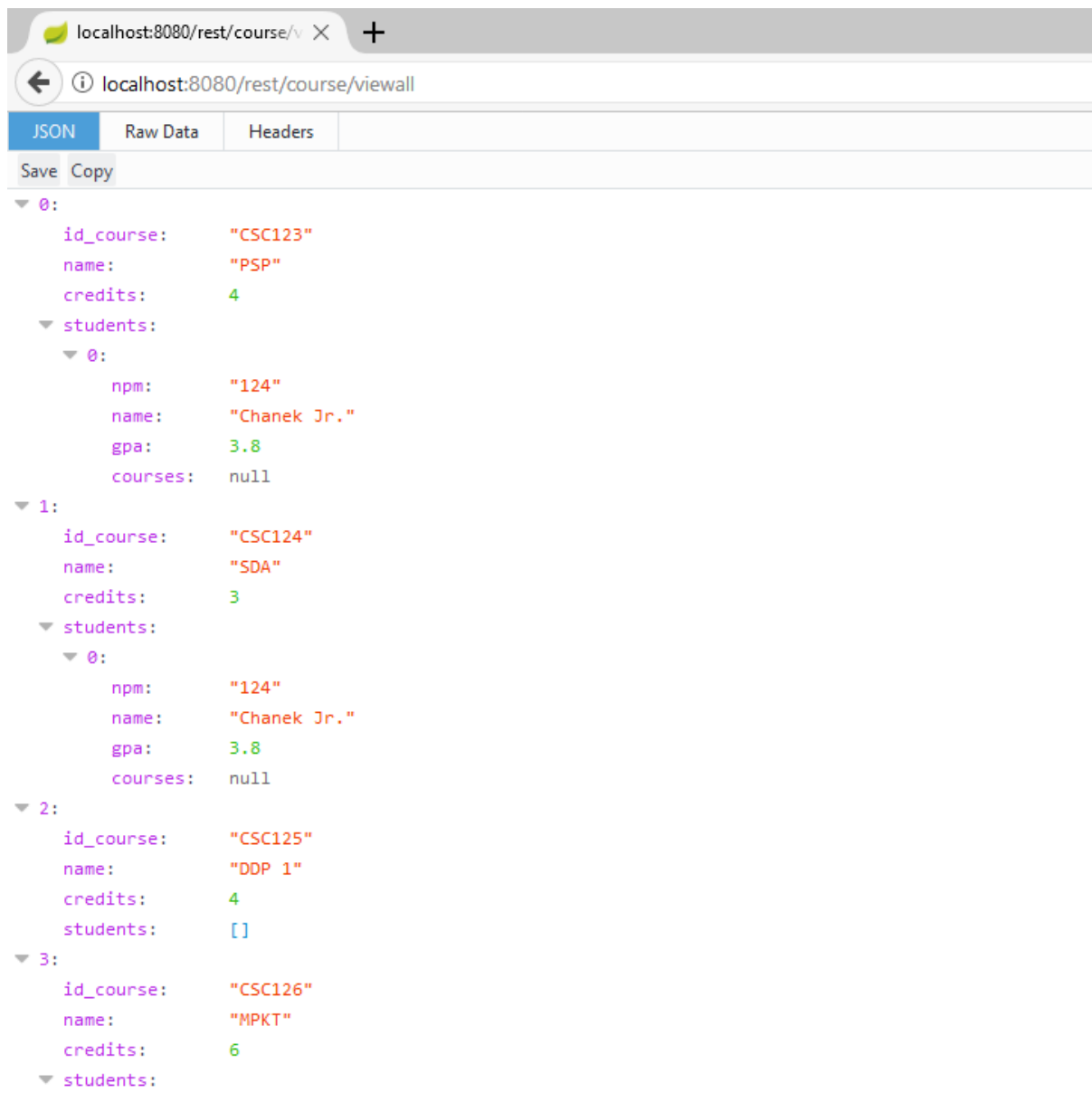
@RequestMapping("/course/view/{id}")
public CourseModel viewCourse (@PathVariable(value = "id") String id) {
    CourseModel course = studentService.selectCourse(id);
    return course;
}

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

```

Jika dijalankan, maka akan ditampilkan seperti ini.





### Latihan 3

- ✓ *Service consumer* untuk viewAll Students

Untuk dapat menampilkan data yang diambil dari *service*, maka pada *consumer* dibuat method-method seperti ini di class `StudentDAO`, `StudentDAOImpl`, dan `StudentServiceRest`.

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

Method di bawah ini untuk mengambil objek yang dikembalikan oleh *producer* dalam bentuk JSON.

```

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

```

Jika student view all dari *consumer* dijalankan pada port 9090, maka akan tampil seperti di bawah ini.

Showing 1 to 2 of 2 entries

No	NPM	Name	GPA	Status	Update	Delete
1	123	Chanek	3.2	Sangat Memuaskan!	<a href="#">Update Data</a>	<a href="#">Delete Data</a>
2	124	Chanek Jr.	3.8	Cum Laude!	<a href="#">Update Data</a>	<a href="#">Delete Data</a>

Showing 1 to 2 of 2 entries

Previous 1 Next

Mata Kuliah APAP

#### Latihan 4

✓ *service consumer* untuk Course

Sama seperti saat membuat *service consumer* untuk student, maka dibuat juga method-method seperti di bawah ini di class-class CourseDAO, CourseDAOImpl, dan CourseServiceRest.

```

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

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

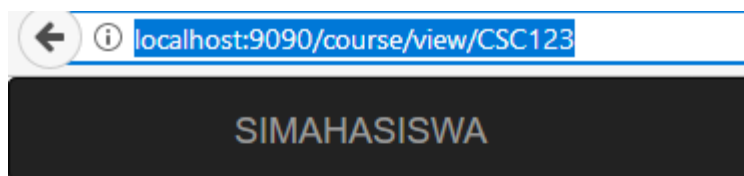
```

Method di atas mengambil object dari JSON yang dikembalikan oleh sisi *producer*.

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

@Override
public List<CourseModel> selectAllCourses ()
{
    log.info("REST - select all courses");
    return studentDAO.selectAllCourses();
}
```

Jika view course dan view all course dijalankan oleh *consumer* di port 9090, maka akan tampil seperti ini.



ID = CSC123

NAME = PSP

CREDITS = 4

Mahasiswa yang mengambil

- 124 - Chanek Jr.

A screenshot of a web browser window showing the URL 'localhost:9090/course/viewall'. The page has a dark header with 'SIMAHASISWA' and navigation links: 'Home', 'Daftar Mahasiswa', and 'Tambah Mahasiswa'. Below the header is a section titled 'All Courseess'. It includes a 'Show 10 entries' dropdown, a search bar, and a table with 4 columns: 'No', 'Id Course', 'Name', and 'Credits'. The table contains 4 rows of data. At the bottom, it says 'Showing 1 to 4 of 4 entries' and has 'Previous', '1', and 'Next' navigation buttons.

No	Id Course	Name	Credits
1	CSC123	PSP	4
2	CSC124	SDA	3
3	CSC125	DDP 1	4
4	CSC126	MPKT	6