Anindito Izdihardian Wibisono
1506757466
APAP – C

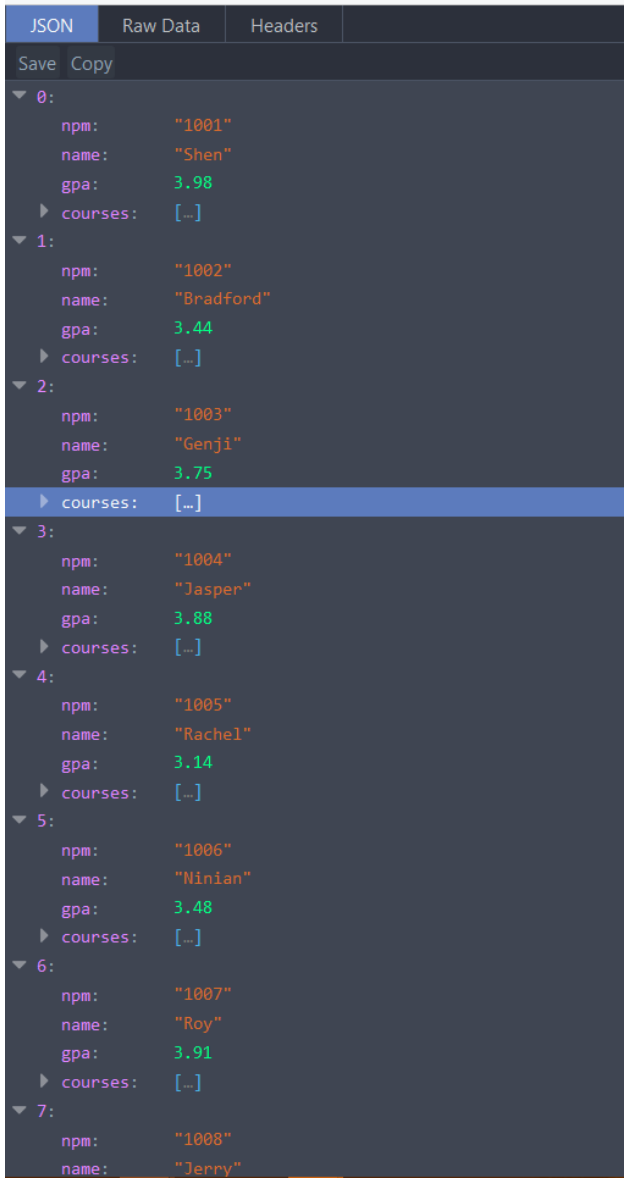
Dalam tutorial kali ini, saya belajar mengenai penggunaan metode REST untuk mengembangkan web service. Dengan metode REST, suatu web service dipisah menjadi dua komponen: “producer”, yaitu bagian ‘back-end’ yang bertugas mengolah permintaan client serta menyajikan hasil kueri basis data, dan “consumer” yang merupakan sisi ‘front-end’ yang berinteraksi dengan pengguna. Hal ini memudahkan pengembangan antara tipe-tipe client yang berbeda (misalnya beda OS atau beda jenis perangkat), karena cukup diperlukan satu producer yang bisa menanggapi permintaan dari beberapa client/consumer.

LATIHAN 1: VIEW ALL STUDENTS UNTUK PRODUCER

Berikut potongan kode untuk metode view all pada producer:

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

Potongan kode ini ditempatkan di StudentRestController.java pada package com.example.rest. Cara kerjanya mirip dengan method pada controller ‘biasa’ di tutorial 5; bedanya, method ini mengembalikan object secara langsung, bukan view yang menampilkan isi object tersebut.



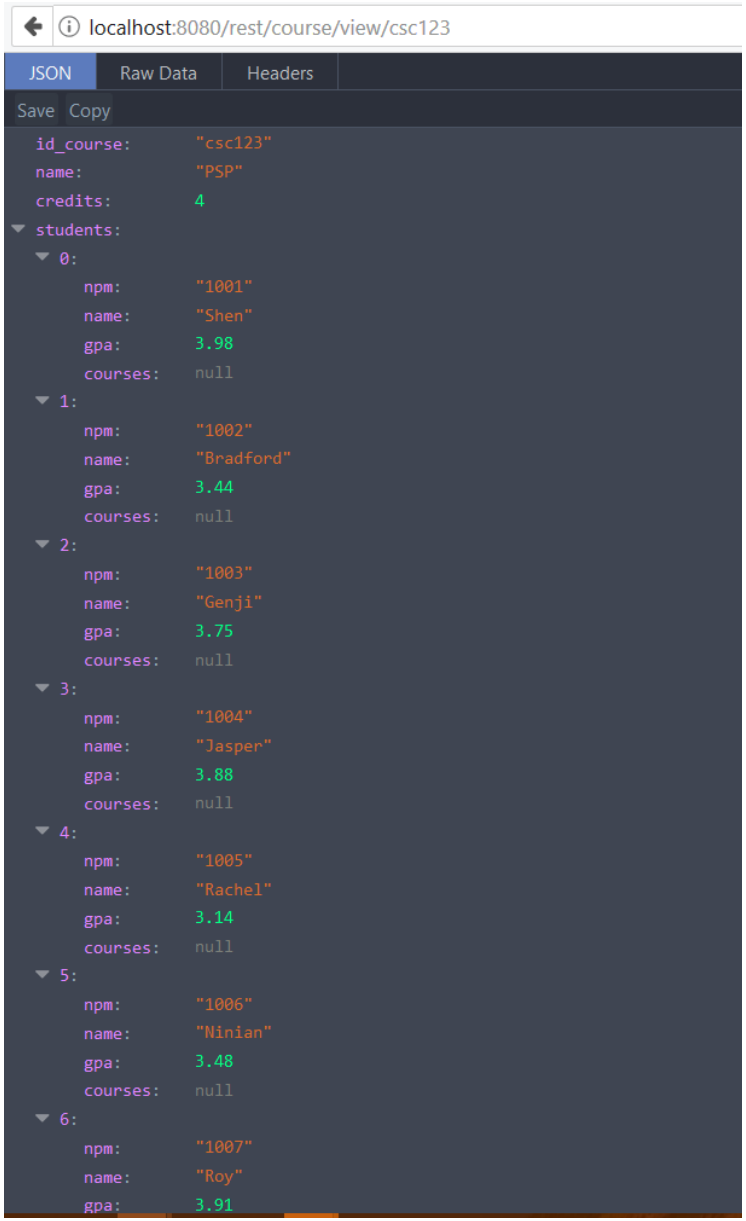
Hasil tampilan untuk halaman localhost:8080/rest/student/viewall. Kumpulan object JSON ini sudah di-parse oleh Firefox.

LATIHAN 2: COURSE SERVICE UNTUK PRODUCER

Berikut potongan kode untuk metode view course pada producer:

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

Potongan kode ini ditempatkan di StudentRestController.java pada package com.example.rest. Sama seperti method view all diatas (latihan 1), method ini dibedakan dari method yang ada di controller ‘biasa’ berdasarkan tipe returnnya. Dalam kasus ini, viewCourse mengembalikan hasil kueri ke basis data oleh mapper sebagai satu object CourseModel.



Hasil tampilan untuk halaman localhost:8080/rest/course/view/csc123. Kumpulan object JSON ini sudah di-parse oleh Firefox.

LATIHAN 3: VIEW ALL STUDENTS UNTUK CONSUMER

Berikut potongan kode untuk method view all pada consumer:

Untuk StudentServiceRest.java pada package com.example.service:

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

Untuk StudentDAOImpl.java pada package com.example.dao:

```
public List<StudentModel> selectAllStudents() {
          ResponseEntity<StudentModel[]> sm =
restTemplate.getForEntity("http://localhost:8080/rest/student/viewall/", StudentModel[].class);
          StudentModel[] smBody = sm.getBody();
          List<StudentModel> retSm = new ArrayList<>();
          for (StudentModel studentModel : smBody) {
               retSm.add(studentModel);
          }
          return retSm;
     }
```

Method utama untuk fungsionalitas ini, yaitu SelectAllStudents pada StudentDAOImpl, tidak sesederhana dua method sebelumnya. Pertama, diperlukan ResponseEntity berisi array StudentModel, dimana ResponseEntity tersebut dibuat lewat mengambil hasil view all pada producer. Setelah itu, isi array dari ResponseEntity diambil, dan dipindahkan ke object List baru. Setelah itu, barulah List yang sudah berisi banyak StudentModel tersebut dikembalikan oleh method.

LATIHAN 4: COURSE SERVICE UNTUK CONSUMER

Berikut potongan kode untuk method view course pada producer:

Untuk StudentServiceRest.java pada package com.example.service:

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

Untuk StudentDAOImpl.java pada package com.example.dao:

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

Method ini lebih sederhana dari method view all students, dan lebih mirip dengan method aslinya pada controller biasa (bukan REST). Method ini cukup mengambil hasil object dari method view yang sesuai pada producer, kemudian megembalikan object tersebut.