Ringkasan

Pada tutorial kali ini, dipelajari bagaimana cara membuat suatu *web services* dengan menggunakan RestTemplate pada *framework* SpringBoot dan cara implementasi pada sebuah web. Adapun hal-hal yang dapat dipelajari yaitu pertama, responseType dari penggunaan method getForObject pada RestTemplate belum bisa menerima semua jenis tipe data sehingga pada latihan membuat *service* viewAll harus menggunakan tipe data *array of objects* terlebih dahulu sebelum dikonversi ke dalam *list of objects*. Kedua, agar sebuah Web Controller benar-benar memanggil *class* yang ditujukan untuk *web service* pastikan menggunakan anotasi @Primary pada *class* tersebut.

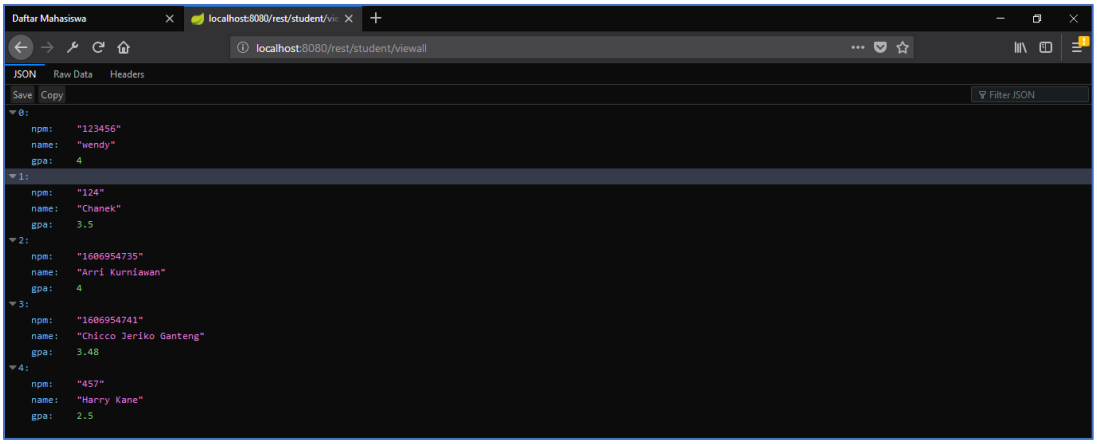
Latihan 1 – *Service* viewAll *(producer)*

Pembuatan service ini, hampir sama dengan *method* viewAll pada Web Controller yang menjadi pembeda adalah pada Web Controller method dibuat dengan kembalian berupa String, sedangkan pada service dibuat dengan kembalian berupa *list of* StudentModel seperti pada Gambar 1.

```java
package com.example.rest;

import java.util.List;

@RestController
@RequestMapping("/rest")
public class StudentRestController {
    @Autowired
    StudentService studentService;

    @RequestMapping("/student/view/{npm}")
    public StudentModel view (@PathVariable(value = "npm") String npm) {
        StudentModel student = studentService.selectStudent(npm);
        return student;
    }

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

Gambar 1. Service Method viewAll

Adapun hasilnya dapat dilihat pada Gambar 2 berikut.



Gambar 2. Result Service Method viewAll

Latihan 2 – Implementasi *Service* viewAll *(consumer)*

Dalam mengimplementasikan service, tidak ada perubahan yang harus dilakukan pada Web Controller, perubahan yang dilakukan hanya terjadi ketika pengambilan data atau pada bagian Data Access Object (DAO). Pertama-tama ditambahkan method selectAllStudents pada *interface* StudentDAO yang telah dibuat pada fase tutorial seperti Gambar 3.

```
package com.example.dao;

import java.util.List;

public interface StudentDAO {
    StudentModel selectStudent (String npm);
    List<StudentModel> selectAllStudents ();
}
```

Gambar 3. Interface StudentDAO

Setelah itu, melengkapi *method* selectAllStudents yang sebelumnya pada fase tutorial hanya mengembalikan nilai *null* sekarang dirubah dengan memanggil *service* viewAll yang telah dibuat pada latihan 1 di atas. Dikarenakan method getForObject pada RestTemplate tidak dapat menggunakan responseType berupa list maka responseType yang digunakan merupakan *array of objects,* dalam kasus ini adalah *array of* StudentModel. Kemudian, *array* tersebut selanjutnya dikonversi menjadi sebuah *list* agar dapat digunakan oleh method pada Web Controller seperti pada Gambar 4 berikut.

```
package com.example.dao;

import java.util.Arrays;

@Service
public class StudentDAOImpl implements StudentDAO {
    @Autowired
    @Lazy
    private RestTemplate restTemplate;

    @Override
    public StudentModel selectStudent (String npm) {
        StudentModel student = restTemplate.getForObject(
                "http://localhost:8080/rest/student/view/" + npm,
                StudentModel.class);
        return student;
    }

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

    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        // Do any additional configuration here
        return builder.build();
    }
}
```

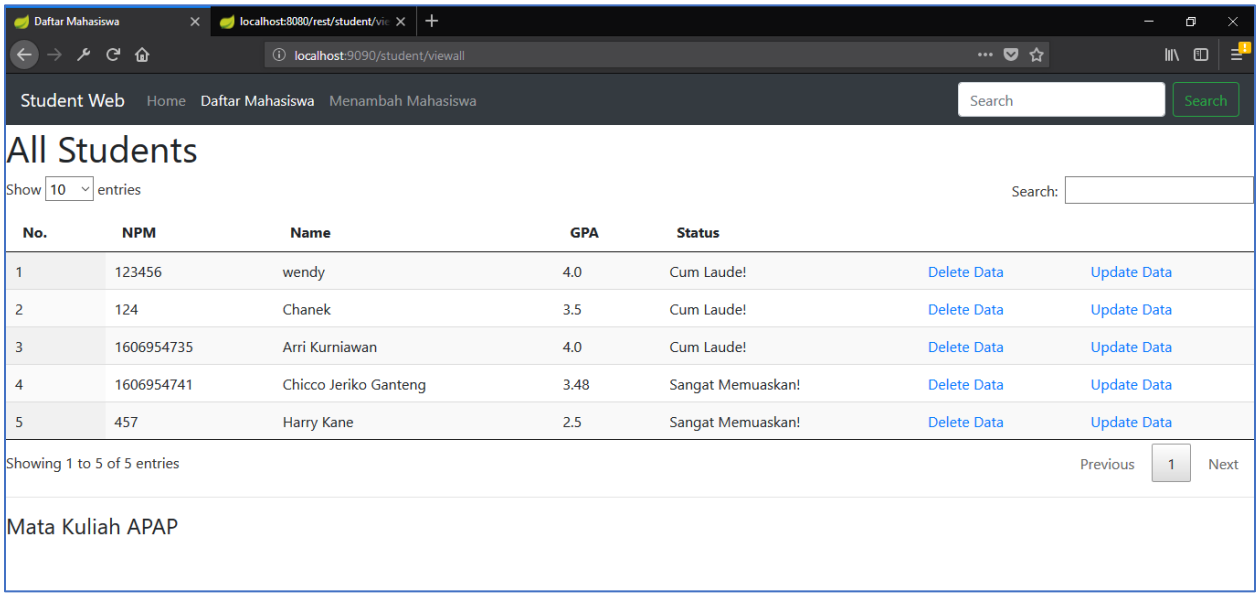
Gambar 4. Class StudentDAOImpl

Terakhir, agar studentService yang dipanggil pada Web Controller benar-benar mengakses *web services* yang telah dibuat pastikan pada *Class* StudentServiceRest menggunakan anotasi @Primary sehingga tidak lagi menggunakan *Class* StudentServiceDatabase seperti pada Gambar 5.

```
package com.example.service;

import java.util.List;

@Slf4j
@Service
@Primary
public class StudentServiceRest implements StudentService {
    @Autowired
    private StudentDAO studentDAO;

    @Override
    public StudentModel selectStudent (String npm) {
        log.info("REST - select student with npm {}", npm);
        return studentDAO.selectStudent(npm);
    }

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

    @Override
    public void addStudent(StudentModel student) {}

    @Override
    public void deleteStudent(StudentModel student) {}

    @Override
    public void updateStudent(StudentModel student) {}
}
```

Gambar 5. Class StudentServiceRest

Adapun, hasil dari implementasi yang dilakukan dapat dilihat pada Gambar 6 berikut.



Gambar 6. Result Hasil Implementasi Service viewAll