

1706106614

Alex Fransiscus Manihuruk

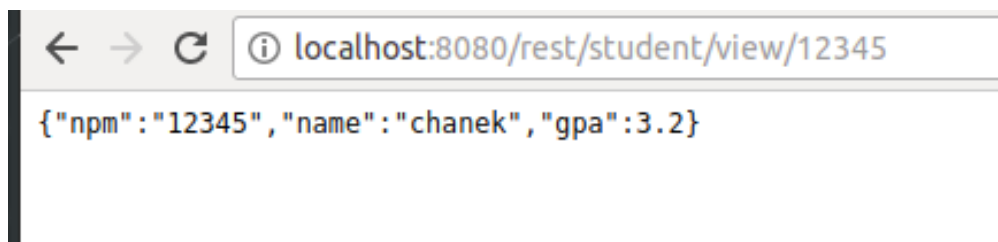
Tutorial 07-Membuat Web Service Menggunakan Spring Boot Framework

A)Membuat Service producer

Membuat class baru yaitu StudentRestController

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

Untuk mencobanya saya jalankan program dan membuka halaman "localhost:8080/rest/student/view/12345" pada browser. Database yang saya gunakan adalah database loka saya.



Untuk memudahkan pembacaan saya menggunakan JSON View dari <http://json.parser.online.fr/>. Hasilnya adalah sebagai berikut:



B)Latihan 1

Memuat 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".

Menambahi code pada StudentRestController seperti berikut:

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

Object `studentService` memanggil method `selectAllStudents()` dan ditampilkan oleh `students` yang bertipe data `List<StudentModel>`. Kemudian objek tersebut dikembalikan. Contoh tampilan ketika mengakses <http://localhost:8080/rest/student/viewall>:

Untuk memudahkan pembacaan saya berikut menggunakan JSON View.

C) Membuat Service Consumer

Tujuan dari service Consumer adalah menampilkan data pada user namun mengambil data dari Producer bukan dari database langsung. Caranya adalah menambahkan class di package dao yaitu `StudentDAO` kemudian di-implement di class `StudentDAOImpl`. Untuk percobaan saya akan menampilkan `selectstudent` dengan parameter NPM seperti berikut.

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

    return student;
}
```

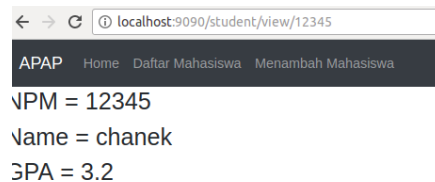
`Resttemplate` adalah sebuah *class* untuk akses HTTP sisi klien. `GetForObject` adalah salah satu *method* di `Resttemplate`. *Method* ini akan mengambil representasi dengan melakukan GET di URL, kemudian tanggapannya diubah dan dikembalikan. Parameter untuk *method* ini ada 2 yaitu url dan respon type. Url yang dikirim kali ini adalah <http://localhost:8080/rest/student/view/npm>. Dan respon typenya adalah `StudentModel`. Jadi kembalian dari *method* ini nantinya adalah objek bertipe data `StudentModel` yang nantinya akan dipakai di *Controler*. Kemudian *override method* `selectStudent` dari `StudentService` di `StudentServiceRest`

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

Untuk memastikan service yang dipakai kali ini adalah StudentServiceRest maka ditambahkan anotasi @Primary di atas deklarasi kelas StudentServiceRest. Sehingga Autowired akan secara otomatis menginstansiasi StudentService menggunakan StudentServiceRest.

```
@Primary
public class StudentServiceRest implements StudentService{
    .....
}
```

Berikut contohnya diakses dari *consumer*.



Pada percobaan kali ini *Produser* berada di port 8080 sedangkan *Consumer* di port 9090. Sehingga data yang ditampilkan tersebut adalah data yang diambil dari Produsen dikirim ke Consumen dalam bentuk JSON.

D)Latihan 2

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

Pertama menambahkan *method* `selectAllStudents` yang kemudian di-*override* di StudentDAOImpl.

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

    List<StudentModel> selectAllStudents = res.getBody();

    return selectAllStudents;
}
```

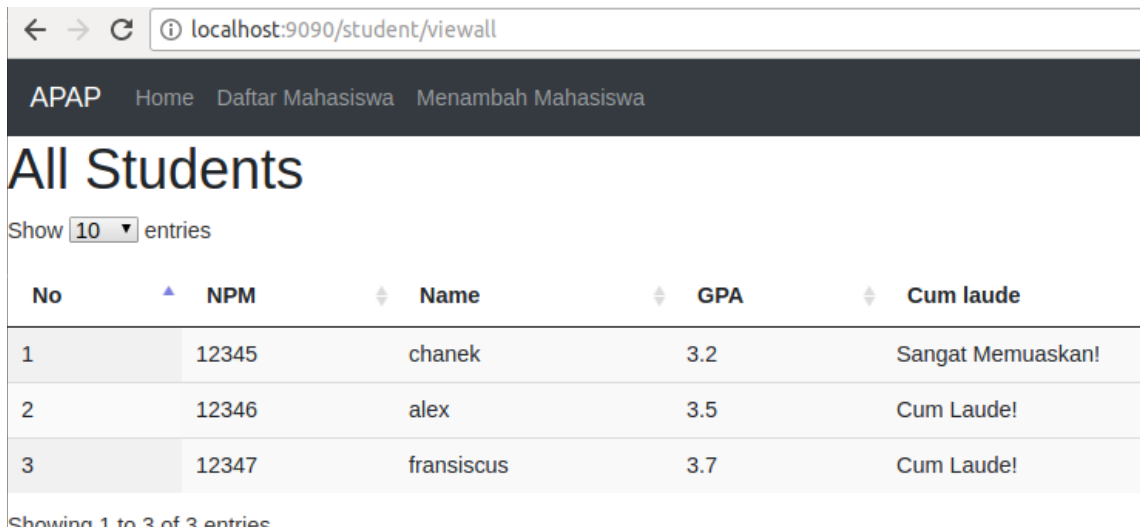
ResponseEntity adalah extension dari HttpEntity yang menambahkan HttpStatus kode. Untuk mendapatkan objek yang dikembalikan oleh web service maka nanti dipanggil method `getBody()`. Dalam RestTemplate, class ResponseEntity dikembalikan dengan method `getForEntity()` dan `exchange()`. Dan kali ini kita pakai dengan *method exchange*. Parameter untuk method ini ada 4 yaitu:

- url – alamat link yang dituju yaitu <http://localhost:8080/rest/student/view/npm>.
- method - HTTP method yang dipakai(GET, POST, PUT,etc) kali ini GET
- requestEntity - entity (*headers* dan/atau *body*) yang dikirim ke request, bisa aja *null*.
- responseType – tipe dari kembaliannya yaitu list StudentModel

Kemudian *override method* selectAllStudents dari StudentService di StudentServiceRest

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

Berikut contohnya diakses dari *consumer*.



No	NPM	Name	GPA	Cum laude
1	12345	chanek	3.2	Sangat Memuaskan!
2	12346	alex	3.5	Cum Laude!
3	12347	fransiscus	3.7	Cum Laude!

Pada percobaan kali ini *Produser* berada di port 8080 sedangkan *Consumer* di port 9090. Sehingga data yang ditampilkan tersebut adalah data yang diambil dari Produsen dikirim ke Consume dalam bentuk JSON.

Ringkasan dan Kesimpulan

- ✓ Web service merupakan sebuah web aplikasi yang akan mengirimkan atau menampilkan data dalam representasi JSON.
- ✓ Pada sisi penyedia web service (produser), untuk membuat controller digunakan anotasi `@RestController`
- ✓ Pada sisi pengguna service (consumer) bisa memanfaatkan implementasi REST Template untuk mendapatkan data dari *produser*. RestTemplate mempunyai beberapa method tergantung kebutuhan seperti `getForObject()` untuk mendapatkan Object dan `exchange()` untuk mendapatkan `responseEntity`, dll.
- ✓ Jika ada 2 service yang sama, maka tambahkan anotasi `@Primary` di service yang diinginkan untuk memastikan service yang dipakai adalah service tersebut.