APAP-C
Tutorial 7 - Write up
Wisnu Pramadhitya Ramadhan / 1606918055

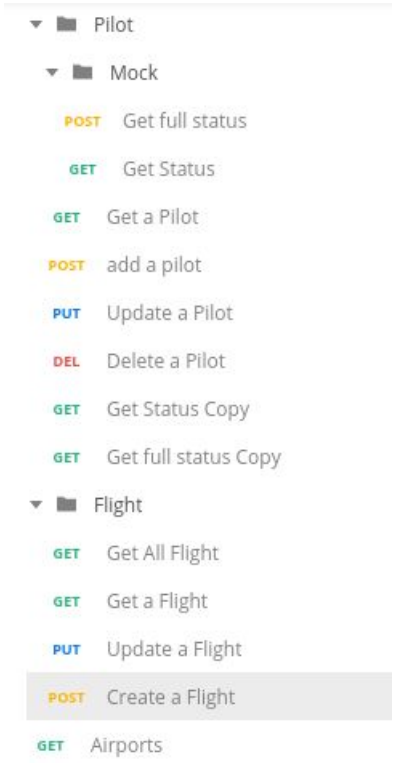
Penjelasan Tutorial

Service Producer

Web service adalah sebuah software yang didesain untuk komunikasi antara mesin ke mesin melalui network. Dalam komunikasinya menggunakan protokol HTTP dengan membawa data menggunakan XML atau JSON. Sedangkan API atau *Application Programming Interface* adalah sebuah set metode sebagai perantara komunikasi untuk antar komponen-komponen. Dalam tutorial ini menggunakan arsitektur RESTful.

Jika kita menyediakan servis untuk memberikan data ke client, maka disebut Service Producer. Dalam Spring kita dapat menggunakan RestController.

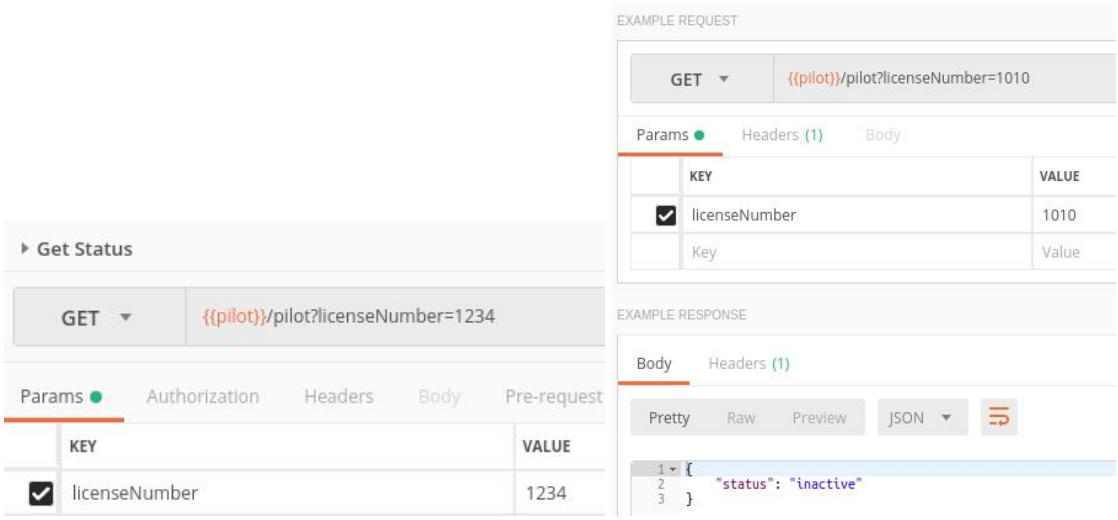
List request pada Postman:



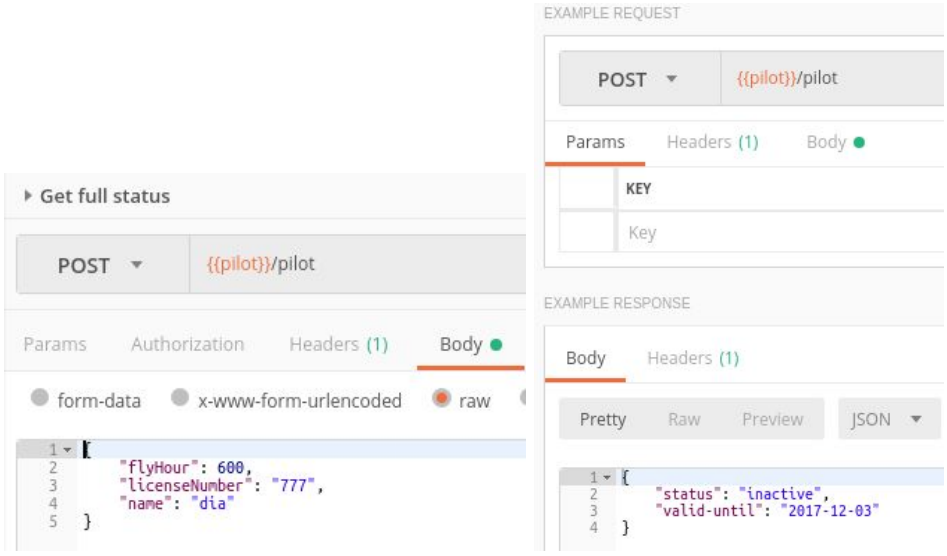
Service Consumer

Service consumer adalah client yang memerlukan service lain untuk menyediakan datanya. Dalam tutorial ini kita membuat mock up untuk service producer yang digunakan di service kita (tutorial 7) yang menjadi service consumer. Untuk membuat mock up service producer tutorial ini menggunakan postman.

Get Status



Get Full Status



Untuk menerima data dari producer kita dapat menggunakan RestTemplate. Untuk data bind ketika kita post ke service lain kita dapat menggunakan library com.fasterxml.jackson.core dan membuat class untuk data feedback.

Penjelasan Latihan

Nomor 1

Pada latihan nomor 1 tutorial meminta untuk dibuatkan 5 API baru untuk transaksi flight. Method yang diminta post untuk membuat flight, update flight, meminta flight berdasarkan flight number, meminta semua flight yang terdaftar, dan menghapus flight.

Untuk membuat flight baru, agar mudah hanya menggunakan class FlightModel saja, kita perlu mengubah class FlightModel. Pada attribute pilot, dirubah menjadi pilotLicenseNumber dan menghilangkan anotasi ManyToOne dan lainnya, dan hanya menyisakan JoinColumn. Tujuannya untuk dapat membawa license number dari pilot pada json.

```
@JoinColumn(name = "pilot_licenseNumber", referencedColumnName =
"license_number", nullable = false)
private String pilotLicenseNumber;
```

Sehingga pada payload request berisi

```
{
     "flightNumber": "10101001",
     "origin": "Surabaya",
     "destination": "Jakarta",
     "time": "2018-01-01",
     "pilotLicenseNumber": "1234"
}
```

Untuk API lainnya implementasi hampir sama dengan yang ada di tutorial.

Nomor 2

Pada latihan nomor 2, tutorial meminta service yang telah dibuat untuk menjadi consumer dan producer, maksudnya adalah bisa dikatakan service yang dibuat digunakan sebagai wrapper dari sebuah service lain. Service diminta untuk memberikan data airport-airport berdasarkan kata kunci city. Untuk mendukung ini, kita menggunakan service amadeus.

```
private static final String AMADUES_API_KEY =
"uiaunKvF5vSajCeGkAPvEJkNggsgR4A8";

public static final String AMADUES_API_URL =
"https://api.sandbox.amadeus.com/v1.2";

public static final String AMADEUS_API_AIRPORT = AMADUES_API_URL
+ "/airports/autocomplete?apikey="+AMADUES_API_KEY;

public static final String getAirportIDUrlByCity(String city) {
    return AMADEUS_API_AIRPORT +
"&all_airports=true&country=ID&term=" + city;
}
```

Lalu setelah memiliki api key dan sudah mengetahui URL API untuk meminta list airport maka kita implementasikan controller baru yaitu AirportController

```
@RestController
public class AirportController {

    @Autowired
    private RestTemplate restTemp; // var tidak boleh sama

    @Bean
    public RestTemplate restTemp() {
        return new RestTemplate();
    }

    @GetMapping("/airports")
```

```
    public String
getAllIndoensianAirportById(@RequestParam(value = "city",
required = false) String city) throws Exception {
        return restTemp
               .getForEntity(Setting.getAirportIDUrlByCity(city)
          , String.class)
          .getBody();
    }
}
```

Karena kita hanya sebagai wrapper dan tidak mengubah data maka kita langsung kembalikan body dari request yang dibuat oleh RestTemplate ke url yang kita berikan. Kita menggunakan getForEntity agar request get yang kita minta di convert menjadi ResponseEntity.