RAFIQA AMINI MULIA
1606917834
APAP – C

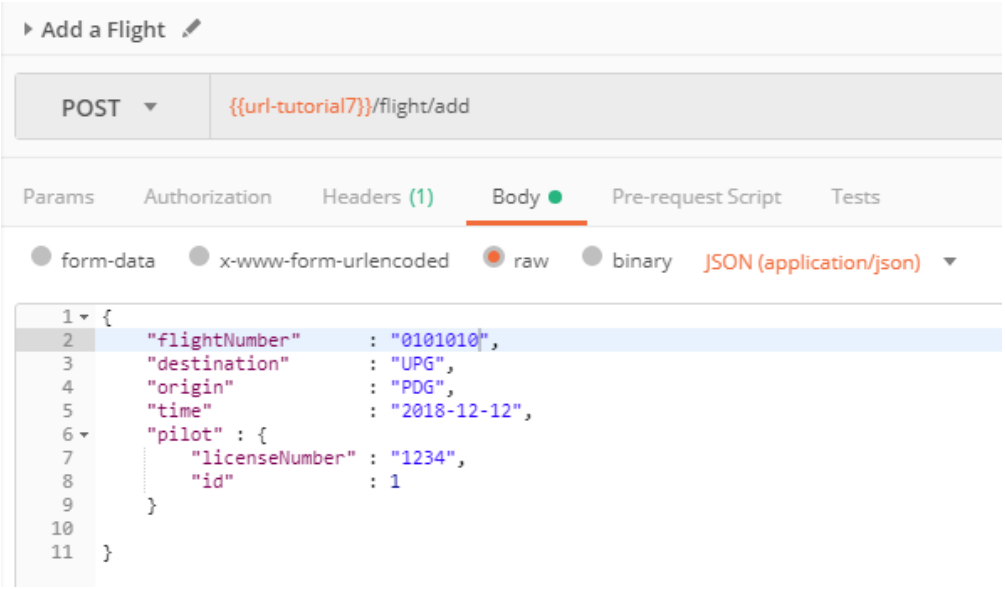
Tutorial 7 : Membuat Web Service Menggunakan Spring Boot Framework

1. **Membuat FlightController menjadi RestController**
   a. **POST add flight**

```
@PostMapping(value = "/add")
public FlightModel addFlightSubmit(@RequestBody FlightModel flight) {
    return flightService.addFlight(flight);
}
```

Screenshot method add di FlightController

Pada saat pembuatan method addFlightSubmit, anotasi mapping yang dipakai adalah @PostMapping karena ketika melakukan penambahan sebuah flight, method yang digunakan adalah POST.
Parameter yang dipakai dalam method ini adalah Objek/entity FlightModel karena ketika dilakukan penambahan flight, parameter yang dibutuhkan pada flightService adalah objeknya.
Pada isi method, akan dilakukan pemanggilan method addFlight yang sebelumnya telah di implementasikan di FlightServiceImpl memakai flightDb.save(entity flight).
Method ini akan mengembalikan entity/objek FlightModel yang baru dibuat.



Screenshot request POST add flight di Postman

Pada Postman, folder Flight, ditambahkan sebuah request post dengan nama “Add a Flight” yang akan memanggil url localhost:2016/flight/add. Request ini tidak memerlukan parameter.
Pada body, dibuatkan data yang ingin dimasukkan sesuai kebutuhan atribut Flight yakni flightNumber, destination, origin, time, dan entity Pilot. Pada entity Pilot yang harus dimasukkan adalah id dan licenseNumber nya. Id harus dimasukkan karena kalau tidak entity pilot yang diinginkan tidak bisa di retrieve. licenseNumber dimasukkan karena pada awalnya yang dijadikan joinColumn pada tabel flight adalah licenseNumber dari Pilot.

RAFIQA AMINI MULIA
1606917834
APAP – C

```
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "pilot_license_number", referencedColumnName = "license_number", nullable = false)
@OnDelete(action = OnDeleteAction.NO_ACTION)
@JsonIgnoreProperties(ignoreUnknown = true)
private PilotModel pilot;
```

Screenshot FlightModel bagian atribut pilot

```
@OneToMany(mappedBy = "pilot", fetch = FetchType.EAGER, cascade = CascadeType.ALL)
@JsonIgnore
private List<FlightModel> pilotFlight;
```

Screenshot PilotModel bagian atribut pilotFlight

Pada step ini, tipe fetct yang dipakai diganti menjadi eager pada kedua model, yakni PilotModel dan FlightModel. Hal lain yang ditambahkan pada step ini adalah penggantian anotasi JsonIgnore. Pada awalnya yang dipakai hanyalah JsonIgnore biasa (pada FlightModel). Sedangkan pada PilotModel, awalnya tidak ada anotasi JsonIgnore. Namun jika tidak ditambahkan JsonIgnore, value pilot yang keluar ketika view Flight akan berulang.
Hasil response dari request post yang diberikan adalah sebagai berikut :


```
{
    "id": 564,
    "flightNumber": "0101010",
    "origin": "PDG",
    "destination": "UPG",
    "time": "2018-12-12",
    "pilot": {
        "id": 1,
        "licenseNumber": "1234",
        "name": null,
        "flyHour": 0
    }
}
```


Screenshot respons dari request Post add flight di Postman

Memang pada response yang diberikan nama dan fliyHour dari pilot bernilai null, hal tersebut karena tidak dimasukkan secara eksplisit pada saat request. Namun ketika di cek view flight, data pilot sudah keluar dengan benar yang berarti asosiasi mereka terbaca (tidak ada nilai atribut yang null lagi).


```
{
    "id": 564,
    "flightNumber": "0101010",
    "origin": "PDG",
    "destination": "UPG",
    "time": "2018-12-12",
    "pilot": {
        "id": 1,
        "licenseNumber": "1234",
        "name": "Nana",
        "flyHour": 139
    }
}
```


Screenshot hasil view flight dengan flightNumber 0101010

RAFIQA AMINI MULIA
1606917834
APAP – C

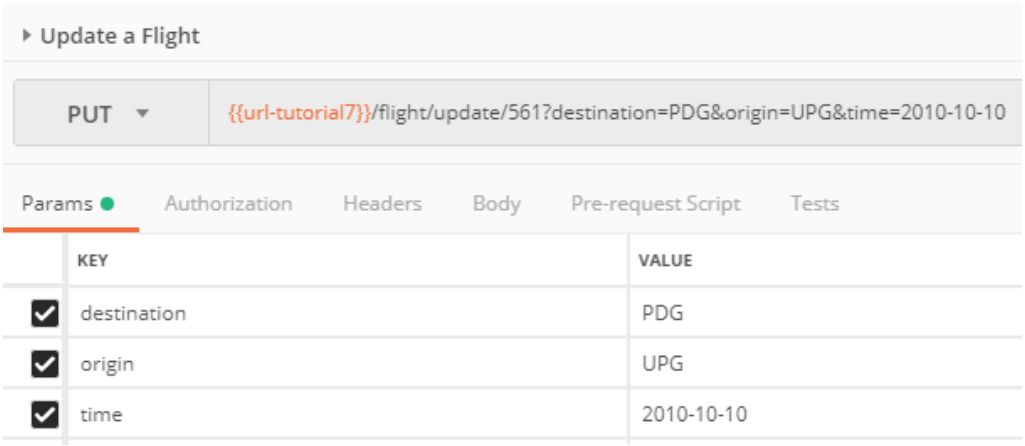
b. PUT update flight

```
@PutMapping(value = "/update/{flightId}")
public String updateFlightSubmit(@PathVariable("flightId") long id,
        @RequestParam("destination") String destination,
        @RequestParam("origin") String origin,
        @RequestParam("time") Date time) {
    FlightModel flight = flightService.getFlightDetailById(id);
    if (flight.equals(null)) {
        return "couldn'n update flight";
    }
    flight.setDestination(destination);
    flight.setOrigin(origin);
    flight.setTime(time);
    flightService.updateFlight(flight, id);
    return "flight updated success";
}
```

Screenshot method updateFlightSubmit di FlightController

Pada saat membuat method updateFlightSubmit, anotasi yang dipakai adalah PutMapping karena yang dilakukan adalah mengubah data yang sudah ada (update).
Pada method ini dibutuhkan path variable berupa id dari flight yang ingin di update. Bersamaan dengan parameter destinasi, origin, dan time yang baru.

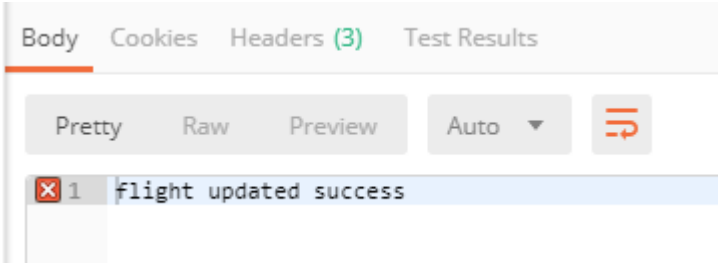
Isi dari method updateFlightSubmit ini adalah memanggil method getFlightDetailById yang akan mengembalikan semuah objek Flight sesuai dengan id yang diberikan di pathvariabel. Jika ternyata entity flight dengan id tersebut null, maka akan langsung dikembalikan tulisan couldn't update. Nilai null disebabkan karena flight dengan id tersebut tidak ditemukan (tidak ada di database). Jika nilainya tidak null (flight dengan id tersebut exist), maka atribut destination, origin, dan time akan di set sesuai dengan nilai yang diberikan di parameter. Pada step ini, diasumsikan data pilot tidak bisa diubah, karena pada lab sebelumnya ketika dilakukan update juga tidak dilakukan update Pilot di entity flight. Setelah semua di set, akan dipanggil method updateFlight yang telah diimplemetasikan sebelumya.
Method ini akan mengembalikan string "flight updated success"



Screenshot request PUT update flight di Postman

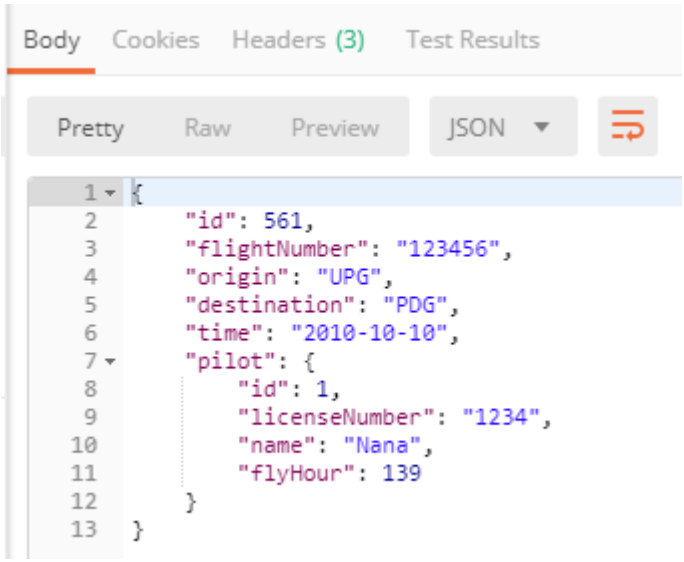
RAFIQA AMINI MULIA
1606917834
APAP – C

Pada Postman, folder Flight, ditambahkan sebuah request put dengan nama “Update a Flight” yang akan memanggil url localhost:2016/flight/update/{flightId} dan memiliki parameter destination, origin dan time.
Pengisian nilai yang akan diubah dilakukan langsung pada field param. Pada pengiriman request, akan mengembalikan response seperti berikut :



Screenshot response dari request update a flight

Yang mana response ini berarti nilai flight telah berhasil di update.



Screenshot hasil view flight yang sudah diubah

Dapat dilihat dari screenshot diatas bahwa data flight dengan id 561 telah berhasil diubah sesuai dengan parameter yang diberikan ketika request update a flight.

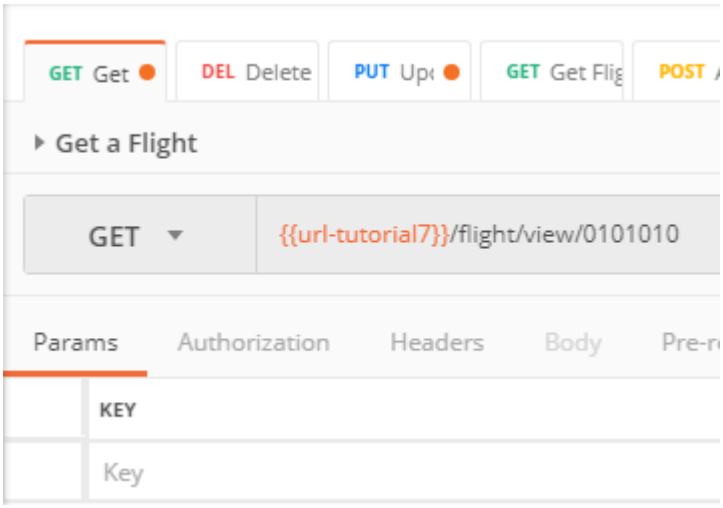
c. GET flight

```
@GetMapping(value = "/view/{flightNumber}")
public FlightModel flightView(@PathVariable("flightNumber") String flightNumber) {
    FlightModel flight = flightService.findFilghtByFlightNumber(flightNumber);
    return flight;
}
```

Screenshot method flightView di FlightController

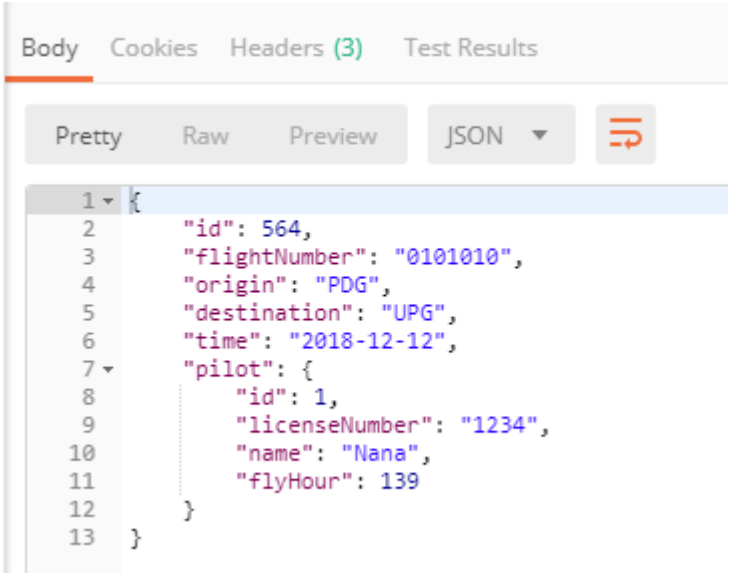
Pada saat membuat method flightView, anotasi yang dipakai adalah GetMapping karena yang dilakukan adalah mengambil data flight yang sudah ada berdasarkan flightNumber nya. Pada method ini dibutuhkan path variable berupa flightNumber yang nantinya akan dijadikan parameter pencarian data flight bersangkutan di database. Isi dari method ini

adalah pemanggilan method findFlightByFlightNumber yang diimplementasikan di service. Dimana method ini akan mengembalikan objek/entity FlightModel dengan nomor flightNumber yang dicari.



Screenshot request get untuk get flight di postman

Pada postman folder flight, akan dibuat sebuah request get yang diberi nama get a flight yang akan memanggil url localhost:2016/flight/view/{flightNumber}. Dimana pada contoh diatas yang ingin dilihat adalah flight dengan flightNumber 0101010. Dimana setelah request dikirim, akan memberikan response sebagai berikut :



Screenshot response dari request get a flight

Dimana response akan memperlihatkan semua atribut yang dimiliki oleh flight yang bersangkutan, dar i id, flightNumber, origin, destination, time, dan detail pilot yang dimiliki.

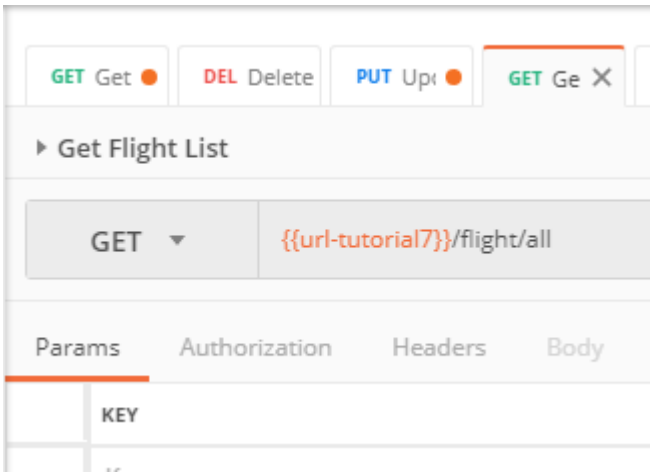
RAFIQA AMINI MULIA
1606917834
APAP – C

d. GET all flight

```
@GetMapping(value = "/all")
public List <FlightModel> flightViewAll() {
    List <FlightModel> flight = flightService.getFlightList();
    return flight;
}
```

Screenshot method flightViewAll di flightController

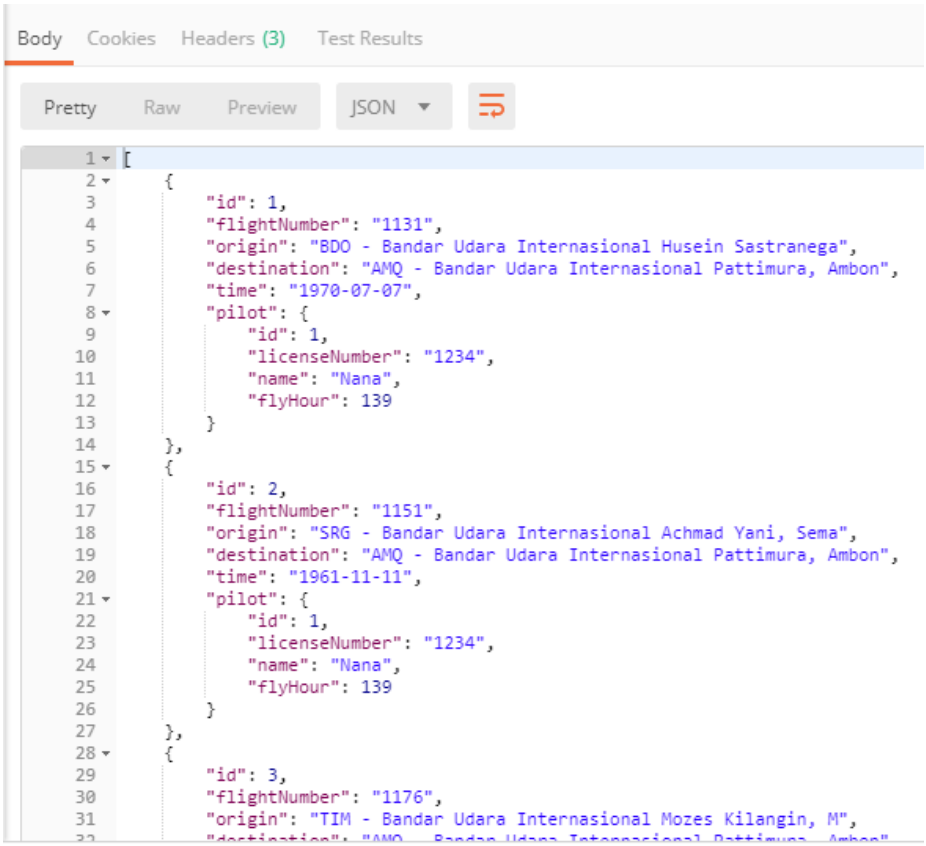
Pada flightController, dibuat senbuah method bernama flightViewAll yang mana method ini akan mengembalikan sebuah List yang berisi objek-objek FlightModel. Pemanggilan method ini akan mengembalikan seluruh data flight yang ada di database.



Screenshot request get flight list di postman

Pada postman folder flight, akan dibuat sebuah request get yang diberi nama get flight list, yang akan memanggil url localhost:2016/flight/all. Response dari request ini adalah sebagai berikut :

RAFIQA AMINI MULIA
1606917834
APAP – C



Screenshot response dari request get flight list

Seperti yang dapat dilihat, response akan memberikan lilst dari flight yang ada di database (menskipun terpotong pada gambar karena daya yang banyak tidak memungkinkan untuk di screenshot semuanya). List akan ditampilkan berurutan sesuai dengan urutan data di database.

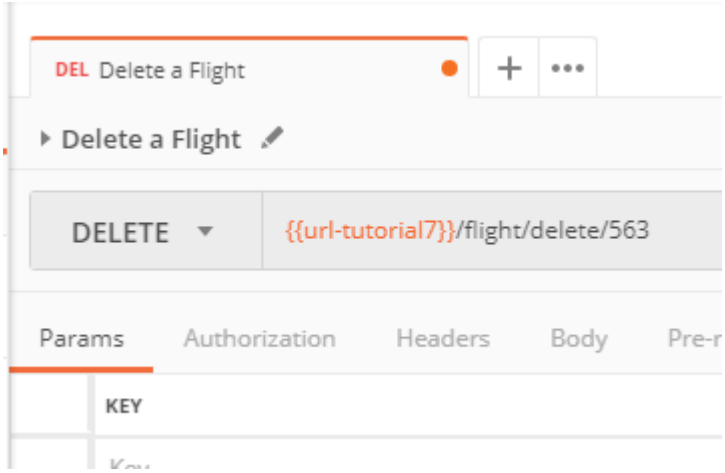
e. DELETE flight

```
@DeleteMapping(value="/delete/{flightId}")
public String deleteFlight(@PathVariable("flightId") Long id) {
    FlightModel flight = flightService.getFlightDetailById(id);
    flightService.deleteFlight(flight);
    return "flight has been deleted";
}
```

Screenshot method deleteFlight di FlightController

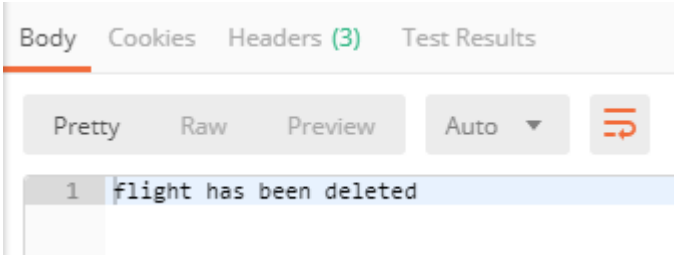
Pada flightController, dibuat sebuah method deleteFlight yang akan mengembalikan sebuah string “flight has been deleted” jika method berhasil dijalankan. Pada method ini, akan dilakukan pengambilan obej flight yang ingin dihapus dengan melakukan pemanggilan method getFlightDetailById dengan parameter id yang telah di inisiasi sebagai pathvariable. Ketika objek yang ingin dihapus sudah teridentifikasi, barulah dilakukan pemanggilan method delete Flight dengan parameter objek yang ingin dihapus, dimana method ini telah diimplementasikan di service sebelumnya.

RAFIQA AMINI MULIA
1606917834
APAP – C



Screenshot request delete flight di postman

Pada postman folder flight, akan dibuat sebuah request delete dengan nama delete a flight yang akan memanggil url localhost:2016/flight/delete/{flightId}. Response dari request ini adalah sebagai berikut :



Screenshot response dari request delete a flight

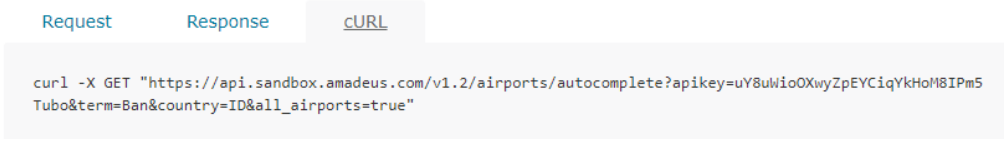
Disana dapat terlihat terdapat tulisan “flight has been deleted” yang berarti proses delete berhasil dilakukan dan data flight yang ingin dihapus sudah terhapus.

2. **Mencari daftar airport disuatu kota**

Langkah yang dilakukan pada saat membuat fitur mencari daftar airport disuatu kota terbagi kepada tiga langkah besar, yakni :

a. Pada Amadeus

- Url



Screenshot url API dari amadeus

Pada saat akses amadeus, setelah dilakukan login dan mendapatkan apikey, dibuka url airport yang diberikan pada soal

Oleh amadeus telah dilakukan setting dibutuhkan 4 parameter ketika pencarian, yakni API-key, country, boolean all airport yang berarti apakah semua airport akan ditunjukkan atau tidak (disini diisikan true karena memang seluruh airport yang

dibutuhkan), serta term yang mana disini dapat diisikan sebagai nama kota dari lokasi airport yang ingin dicari.
Url diatas akan di salin pada setting.java.

b. Pada STS

- FlightController

```
@Autowired
RestTemplate restTemplateFlight;

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

@GetMapping(value = "/airport/{term}")
public String getAirport(@PathVariable("term") String term) throws Exception {
    String path = Setting.airportUrl + term;
    return restTemplateFlight.getForEntity(path, String.class).getBody();
}
```

Screenshot method getAirport di FlightController

Pada flightController akan dibuat sebuah method dengan nama getAiport yang mempunyai pathvariable term. Term disini akan diisi dengan nama kota yang diinginkan pada saat memanggil method.
url yang ada di variabel path adalah url gabungan dari url airport yang diambil dari amadeus dan term yang merupakan parameter yang dapat diubah-ubah.
Nantinya method ini akan menggunakan service consumer untuk mendapatkan data dari third party (amadeus) lalu mengirimkannya lagi sebagai service producer.

- Setting

```
public class Setting {
    final public static String pilotUrl = "https://00ef7186-14bd-47a5-b958-709abac80188.mock.pstmn.io";
    final public static String airportUrl = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=uY8uWioOXwyZpEYCiqYkH
}
```

Screenshot url yang ditulis di setting.java

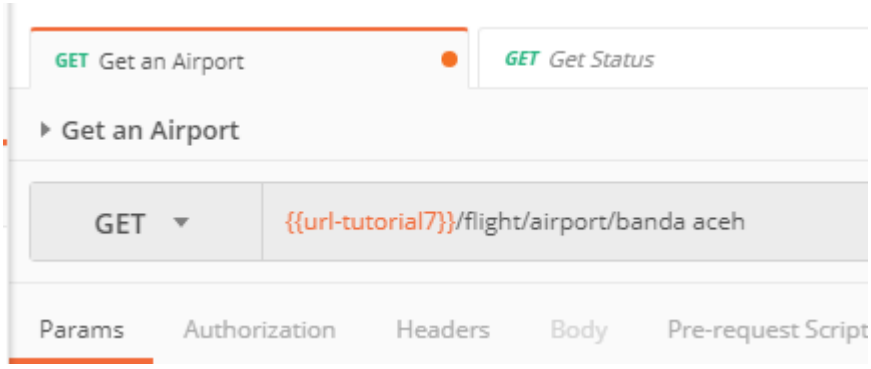
Disini dilakukan pengaturan url yang telah didapatkan dari amadeus. Default dari url yang diberi memiliki 4 parameter, namun disini nilai parameter term akan dikosongkan karena diisi kemudian (dapat berubah)

```
?apikey=uY8uWioOXwyZpEYCiqYkHoM8IPm5Tubo&country=ID&all_airports=true&term="
```

RAFIQA AMINI MULIA
1606917834
APAP – C

c. Pada PostMan

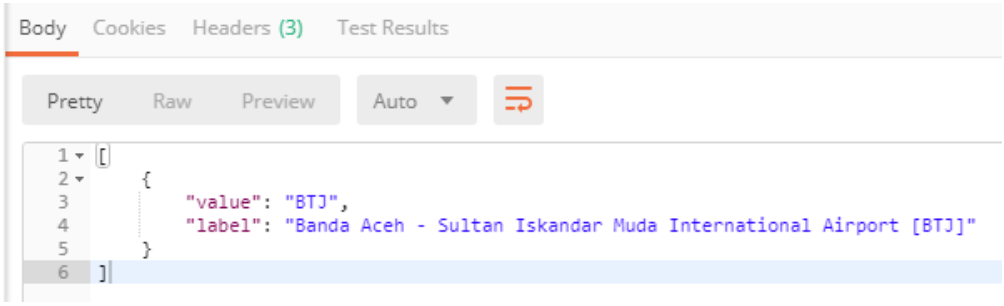
- Request GET airport



Screenshot request get airport di postman

Pada postman folder flight, akan dibuat sebuah request get yang bernama get an airport yang akan memanggil url localhost:2016/flight/airport/{term} dimana term akan diisi dengan nama kota yang airport nya ingin dicari.

- Response



Screenshot response dari request get an airport

Diatas dapat dilihat response dari request get airport. Disini akan ditampilkan data airport yang berada di kota yang dijadikan parameter sebelumnya.
Data ini didapatkan dari third-party (amadeus) yang telah diinisiasi API-key nya dalam url.

--Terima kasih bg jik, semoga hari bg jik menyenangkan--