Stanley Sebastian
APAP C
1606893494

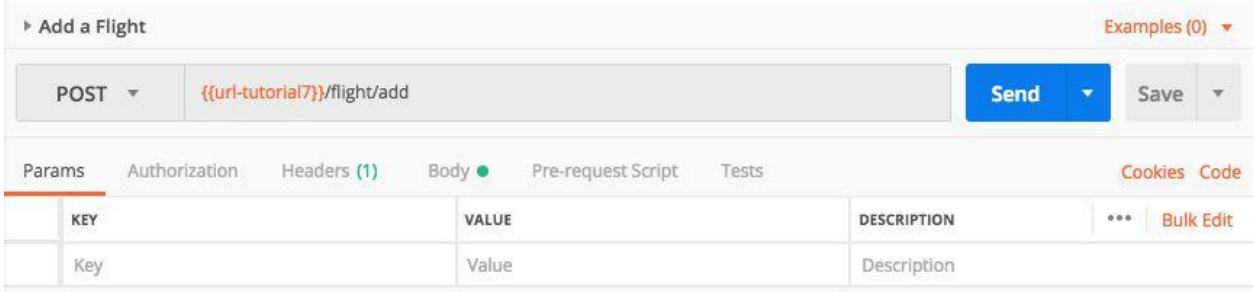
Nomor 1

1. Menambahkan flight

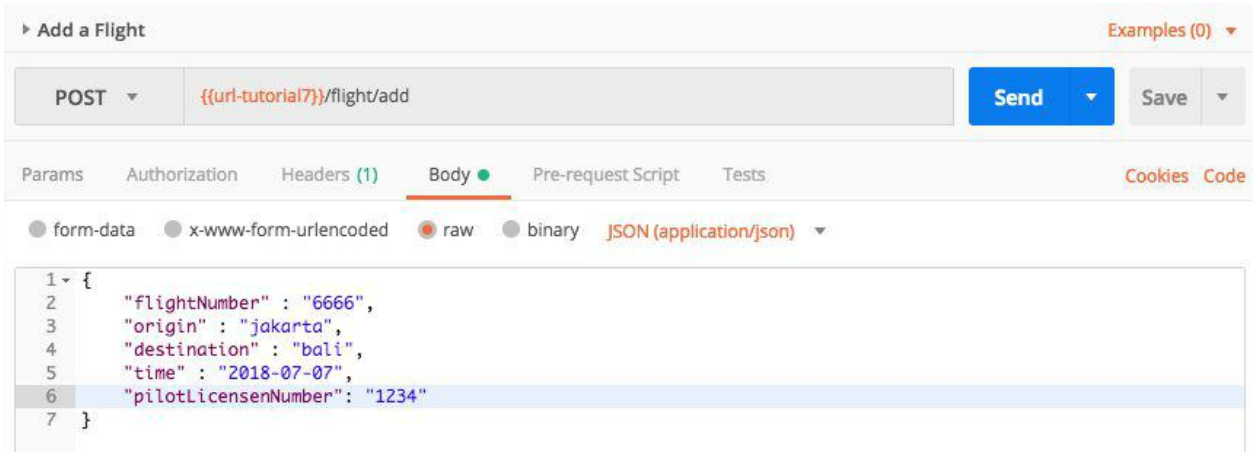
Pada controller implementasikan method addFlight seperti berikut:
Gunakan anotasi @PostMapping, anotasi ini merupakan versi simplifikasi dari @RequestMapping(method = RequestMethod.POST).

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

Lalu pada postman kirim request POST:



Kemudian tambahkan requestbody pada tab body dengan tipe JSON



2. Update flight

Implementasikan method updateFlight pada flight service:

```
@Override
public void updateFlight(FlightModel flight) {
    flightDb.save(flight);
}
```

Implementasikan method updateFlight pada controllersebagai berikut:
Gunakan anotasi @PutMapping, anotasi ini merupakan versi simplifikasi dari @RequestMapping(method = RequestMethod.PUT).

```
@PutMapping(value = "/update/{flightId}")
public String updateFlight(@PathVariable("flightId") long flightId,
        @RequestParam(value = "destination") String destination,
        @RequestParam(value = "origin") String origin,
        @RequestParam(value = "time") Date time) {
    FlightModel flight = flightService.getFlighttDetailById(flightId).get();
    if (flight.equals(null)) {
        return "couldn't find the flight";
    }
    flight.setDestination(destination);
    flight.setOrigin(origin);
    flight.setTime(time);
    flightService.updateFlight(flight);
    return "flight update success";

}
```

Setelah itu kirim request PUT pada postman

3. View flight

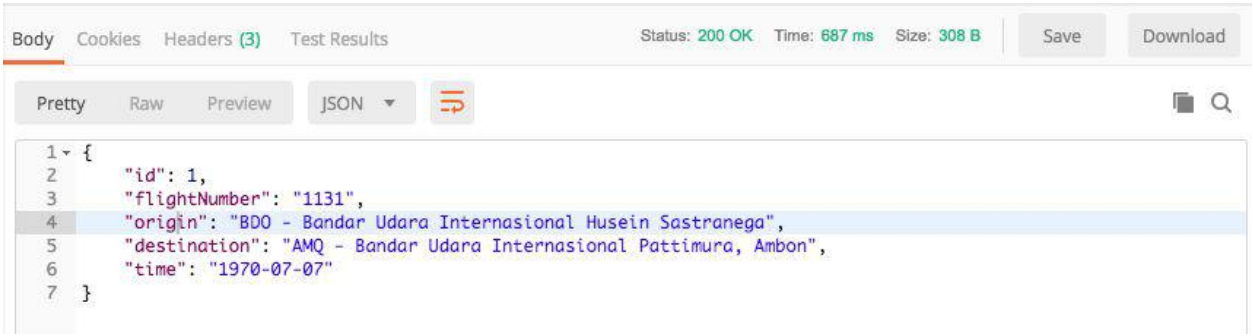
Implementasikan method flightView pada controller sebagai berikut:

Gunakan anotasi @GetMapping, anotasi ini merupakan versi simplifikasi dari @RequestMapping(method = RequestMethod.GET)

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

Kirim request GET pada postman dengan menginput flightNumber yang ada pada database

Send request maka akan dikembalikan response sebagai berikut



4. View all flight

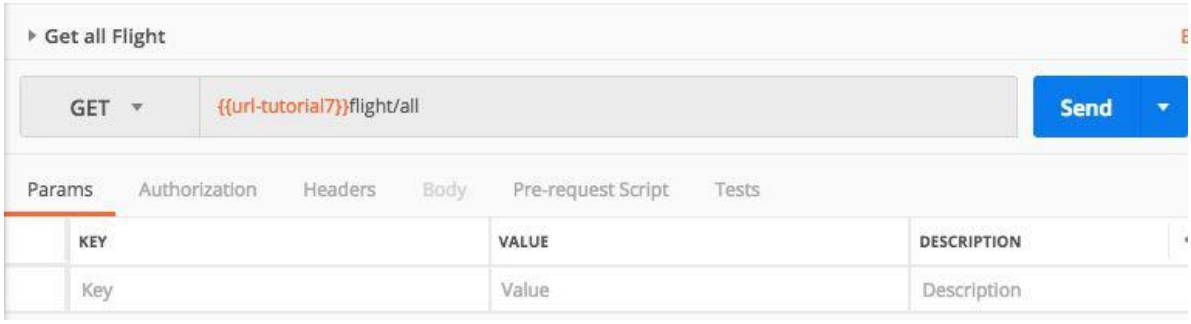
Untuk dapat mengembalikan list flight, implementasikan method getAllFlight pada flightService:

```
@Override
public List<FlightModel> getAllFlight() {
    return flightDb.findAll();
}
```

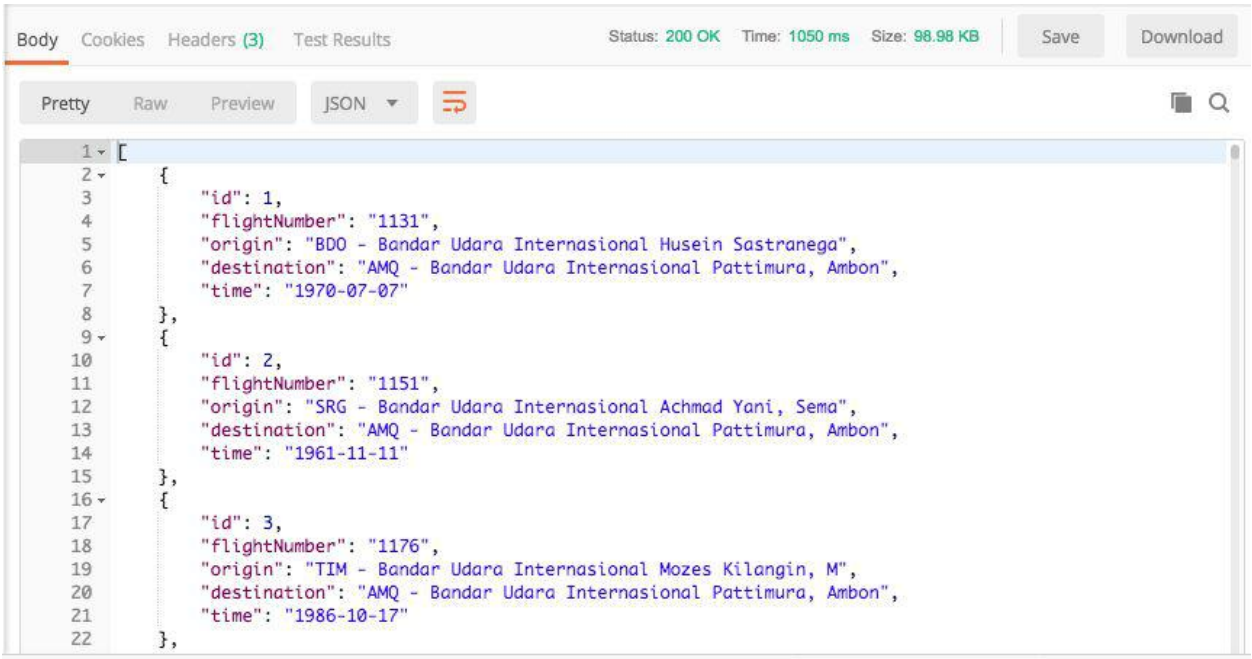
Lalu implementasikan method viewAllFlight pada controller:

```
@GetMapping(value="/view/all")
public List<FlightModel> viewAllFlight() {
    return flightService.getAllFlight();
}
```

Send request pada postman



Maka akan dikembalikan response sebagai berikut:



5. Delete flight

Implementasikan method delete pada flightService

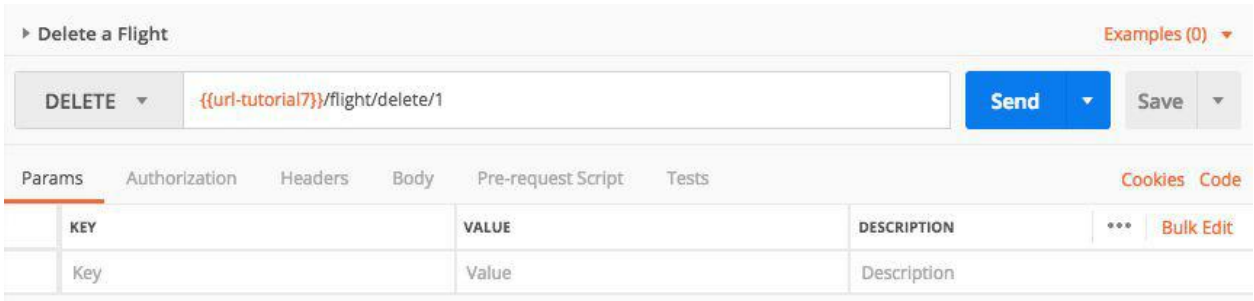
```
@Override
public void delete(FlightModel flight) {
    flightDb.delete(flight);
}
```

Lalu implementasikan method deleteFlight pada controller:

Gunakan anotasi @DeleteMapping, anotasi ini merupakan versi simplifikasi dari @RequestMapping(method = RequestMethod.DELETE).

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

Kirimkan request delete pada postman



Jika berhasil maka akan dikembalikan response:

Stanley Sebastian
APAP C
1606893494

Nomor 2

Pada latihan ini akan dibuat sebuah service producer yang akan mengembalikan daftar airport yang ada di suatu kota di Indonesia. API yang digunakan pada latihan ini merupakan API yang dapat membantu pencarian airport menggunakan autocomplete dengan term parameter.

Misalnya kita ingin mencari airport di Jakarta, kita bisa menggunakan keyword “JAK” atau keyword lain yang merupakan awalan dari jakarta, API akan mengembalikan response berupa value dan label.

value: tiga huruf IATA location code dari airport dan kota.
label: nama dari airport + IATA location code yang dituliskan dalam [...]

1. Pada class Setting masukan url API beserta API key, serta masukan query “country = ID” untuk mem-filterkota yang ada di indonesia.

```
public class Setting {
    final public static String pilotUrl = "https://52143f15-41b3-4031-a360-18fd26289224.mock.pstmn.io";
    final public static String flightUrl = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=hMFmGAaa9WjoB6RMNPGVXD9cLmRnjvBr&country=ID";
}
```

2. Instansiasi sebuah objectRestTemplate dengan menggunakan anotasi @Bean

```
@Autowired
RestTemplate restTemplate;

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

Anotasi @Bean digunakan untuk melakukan proses instansiasi object pada spring. RestTemplate digunakan untuk melakukan simplifikasi terhadap pemanggilan HTTP request dari client. RestTemplate menyediakan methoduntuk membuat HTTP request dan handling response.
Setelah instansiasi, inject dependency dengan menggunakan anotasi @Autowired.

3. Untuk melakukan request GET implementasikan method getAirportDetail

```
@GetMapping(value= "/view")
public String getAirportDetail(@RequestParam(value = "term") String term) throws Exception {
    String path = Setting.flightUrl + "&term=" + term;
    return restTemplate.getForEntity(path, String.class).getBody();
}
```

Anotasi @GetMapping merupakan shortcut dari anotasi @RequestMapping(method = RequestMethod.GET)

Method getAirportDetail() akan menerima parameter term yaitu awalan huruf dari suatu kota misalnya “Jak” untuk “Jakarta”

Ambil url dari Setting dengan menambahkan parameter term

Untuk mengembalikan response berupa raw json gunakan method dari

RestTemplate yaitu getForEntity, getForEntity akan mengembalikan error 404 ketika data tidak ditemukan tidak seperti getForObject yang akan mengembalikan status code 200 jika object null.

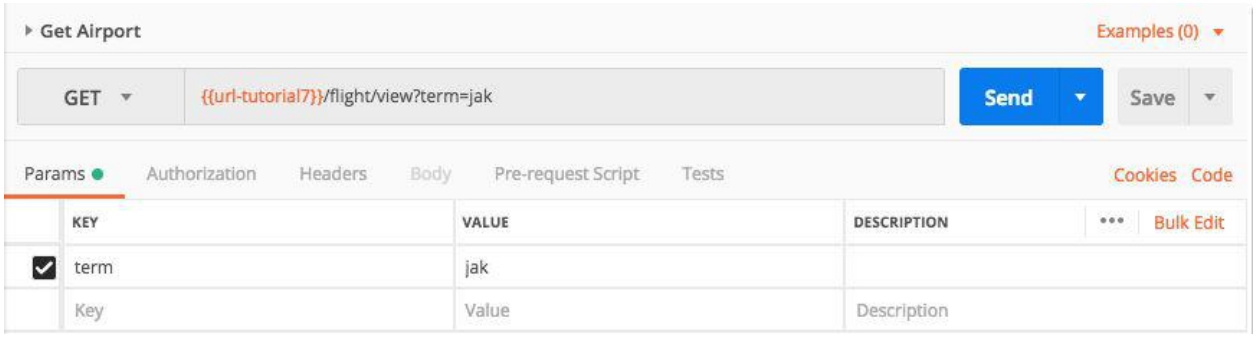
Parameter 1: path - merupakan url dari API

Parameter 2: String.class - merupakan kelas dari string karena parameter yang dibutuhkan adalah kelas bukan object String

method .getBody() mengembalikan response body.

3. Testing pada postman

Masukan request baruGet Airport, dan akses url sebagai berikut:



Send request, maka akan dikembalikan response airportyang berada di Jakarta dan 3 huruf IATA location codeyang berhubungan dengan Jakarta