

Membuat Web Service Menggunakan Spring Boot Framework

Pada tutorial 7 kali ini dijelaskan mengenai pembuatan *web service*. Web Service untuk mengakses data-data Pilot dan Flight pada tutorial sebelumnya. *Web service* merupakan implementasi dari komunikasi antara frontend dan back-end serta komunikasi antar backend dari aplikasi yang berbeda.

1. Pertama diawali dengan membuat RestController Pada tutorial ini anda akan membuat suatu backend sistem. Environment ini akan berguna pada proses debugging dan testing pada development stage sebuah sistem yang biasanya terbagi pada tiga stage yaitu dev, staging, production yang biasanya masing-masing menggunakan request API yang berbeda
2. Mengakses method POST Jika menggunakan field URL pada browser sebenarnya menggunakan method GET, sehingga pada method lainnya, seperti PUT, DELETE, POST, dll tidak akan bisa diakses menggunakan URL browser. Postman bisa mengakses method tanpa membuat terlebih dahulu sistem yang menerima API dari kita.
3. Membuat Mockserver Mockserver adalah server bayangan API yang bisa diakses dengan spesifikasi custom. Mockserver berguna ketika spesifikasi mengharuskan beberapa sistem untuk berkomunikasi satu sama lainnya sedangkan hanya tersedia API documentation dan belum ada implementasinya. Dengan kata lain mockserver mempermudah percepatan development sistem yang sedang dikembangkan secara paralel. Mockserver merupakan service yang ada pada Postman.
4. Terakhir membuat Service Consumer (layer front-end)

Latihan 1

Get flight

```
@GetMapping(value = "/view/{flightId}")  
  
private FlightModel viewFlight(@PathVariable ("flightId") long flightId, Model model)  
{  
    return flightService.findById(flightId);  
}
```

Post add flight

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

Kevin Sutarno
1606895202

Delete flight

```
@DeleteMapping(value = "{flightId}")
private String deleteFlight(@PathVariable("flightId") long flightId, Model model) {
    FlightModel flight = flightService.findById(flightId);
    flightService.deleteByFlightNumber(flight.getFlightNumber());
    return "flight has been delete";
}
```

PUT update flight

```
@PutMapping(value = "/update/{id}")

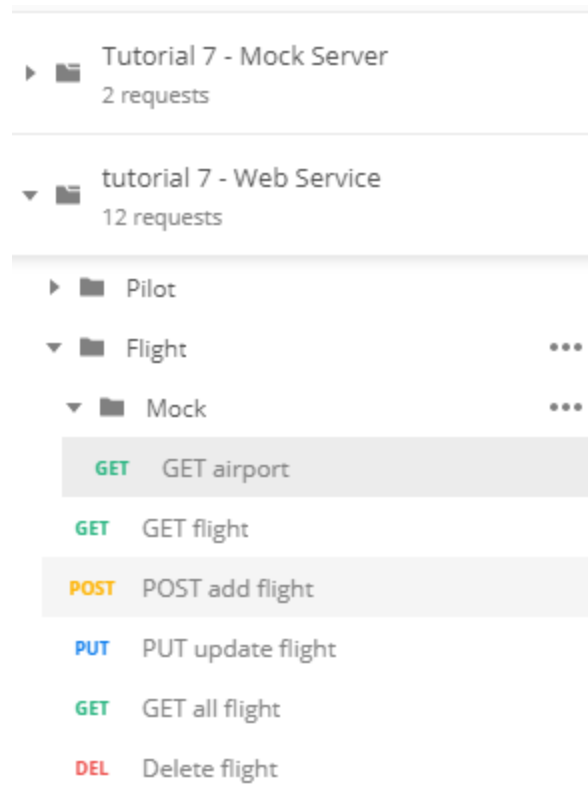
private String updateFlightSubmit(
    @PathVariable (value = "id") long id,
    @RequestParam("flightNumber") String flightNumber,
    @RequestParam("origin") String origin,
    @RequestParam("destination") String destination,
    @RequestParam("time") Date time) {
    FlightModel flight = (FlightModel) flightService.findById(id);
    if(flight.equals(null)) {
        return "Couldn't find flight";
    }
    flight.setFlightNumber(flightNumber);
    flight.setOrigin(origin);
    flight.setDestination(destination);
    flight.setTime(time);
    flightService.addFlight(flight);
    return "flight update success";
}
```

GET all flight

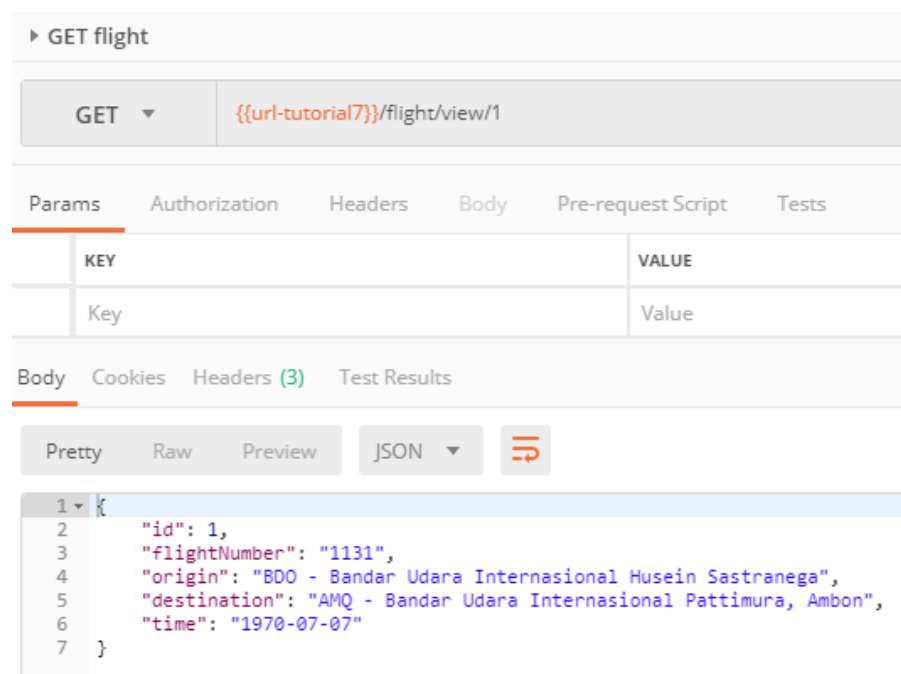
```
@GetMapping("/all")
private List<FlightModel> viewAllFlight(Model model){
    return flightService.viewAllFlight();
}
```

Postman

Kevin Sutarno
1606895202



Postman GET Flight



Postman PUT update flight

Kevin Sutarno
1606895202

PUT `{{url-tutorial7}}/flight/update/2?flightNumber=1512&origin=Jakarta&destination=Medan&time=2018-11-1`

Params Authorization Headers (1) Body Pre-request Script Tests

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> flightNumber	1512	
<input checked="" type="checkbox"/> origin	Jakarta	
<input checked="" type="checkbox"/> destination	Medan	
<input checked="" type="checkbox"/> time	2018-11-1	
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 254 ms

Pretty Raw Preview Auto

1 flight update success

Postman GET all flight

GET all flight

GET `{{url-tutorial7}}/flight/all...`

Params Authorization Headers Body Pre-request Script Tests

KEY	VALUE	D
Key	Value	[

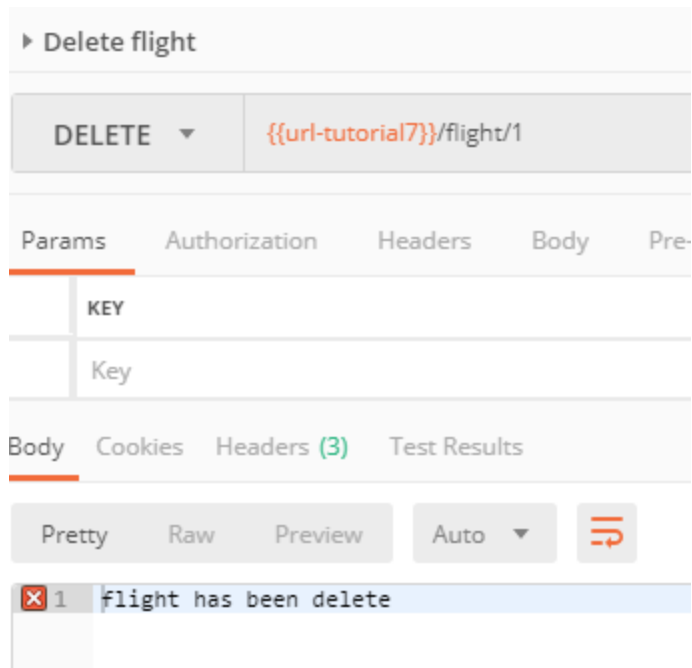
Body Cookies Headers (3) Test Results Status: 200 OK Time:

Pretty Raw Preview JSON

```
1 {
2   {
3     "id": 1,
4     "flightNumber": "1131",
5     "origin": "BDO - Bandar Udara Internasional Husein Sastranegara",
6     "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
7     "time": "1970-07-07"
8   },
9   {
10    "id": 2,
11    "flightNumber": "1151",
12    "origin": "SRG - Bandar Udara Internasional Achmad Yani, Sema",
13    "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
14    "time": "1961-11-10"
15  },
16  {
17    "id": 3,
18    "flightNumber": "1176"
```

Postman Delete Flight

Kevin Sutarno
1606895202



Latihan 2

Buatlah sebuah service producer. Jika dipanggil, akan mengembalikan daftar airport yang ada di suatu kota di Indonesia. Masukan dari API tersebut hanyalah nama kota. Jangan lupa masukkan request untuk mengakses data tersebut pada Postman.

FlightController.java

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

//Pada postman dipanggil dan akan menuju path berisi API Amadeus.

Setting.java

//Menambahkan attribute seperti dibawah yang didapatkan dari sandbox.amadeus.com sebagai path pada

```
final public static String flightUrl =
"https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=A0tTo6YQwwwXmAZCQh
eB2GtClmtnRJZq&country=ID&all_airports=true";
```

Kevin Sutarno

1606895202

POSTMAN


▶ GET airport

GET `{{url-tutorial7}}/flight/airport/jakarta`

Params Authorization Headers Body Pre-request Script Tests

KEY	VALUE
Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Auto 

```
1 [
2   {
3     "value": "JKT",
4     "label": "Jakarta [JKT]"
5   },
6   {
7     "value": "CGK",
8     "label": "Jakarta - Soekarno - Hatta International Airport [CGK]"
9   },
10  {
11    "value": "HLP",
12    "label": "Jakarta - Halim Perdanakusuma Airport [HLP]"
13  }
14 ]
```