

Penjelasan Latihan

Kegunaan dari postman adalah untuk mengetest kodingan yang telah dibuat apakah sesuai keluaran yang diinginkan atau tidak.

1. Langkah – langkah dalam mengerjakan latihan

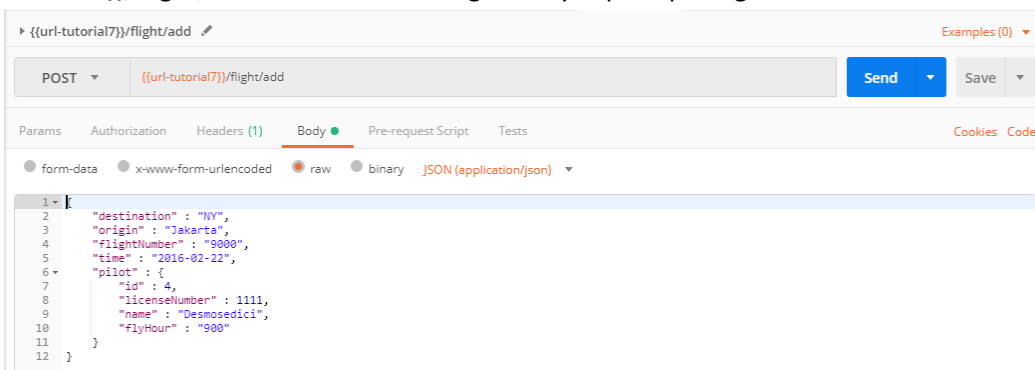
a. Mengubah kodingan yang ada pada Flight controller menjadi seperti di bawah ini.

```

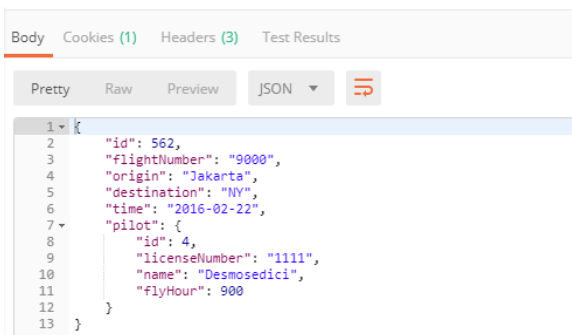
37 @PostMapping(value = "/add")
38 private FlightModel addRow(@RequestBody FlightModel flight) {
39     flightService.addFlight(flight);
40     return flight;
41 }

```

Kemudian setelah itu jalankan test pada postman dengan cara memasukkan “{{url-tutorial7}}/flight/add”. Setelah itu mengisi body seperti pada gambar di bawah ini.



Kemudian klik send akan ada keluaran seperti berikut ini.



b. Mengubah kodingan menjadi seperti di bawah ini

```

72 @PutMapping(value = "/update/{flightId}")
73 public String updateFlightDB(@PathVariable(value = "pilotId") long flightId, @RequestParam(value = "flight
74     PilotModel pilot = pilotService.getPilotDetailById(flightId).get();
75     if (pilot.equals(null)) {
76         return "Couldn't find your pilot";
77     }
78
79     System.out.println(flightId);
80     System.out.println(flightNumber + " " + origin + " " + destination);
81     System.out.println(pilot.getLicenseNumber());
82     flightService.updateFlight(flightId, flightNumber, origin, destination, time);
83     return "flight update success";
84 }

```

Kemudian setelah itu isi url dan variabel seperti di gambar di bawah ini

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> origin	NY	
<input checked="" type="checkbox"/> destination	Jakarta	
<input checked="" type="checkbox"/> flightNumber	9999	
Key	Value	Description

Klik send dan akan ada response seperti pada gambar di bawah ini.

Status: 200 OK Time: 536 ms Size: 137 B

1 flight update success

c. Mengubah kodingan menjadi seperti di bawah ini

```

60 @GetMapping(value = "/view/{flightNumber}")
61 public FlightModel view(@PathVariable(value = "flightNumber") String flightNumber, Model model) {
62     FlightModel archive = flightService.getFlightDetailByFlightNumber(flightNumber).get();
63     return archive;
64 }

```

Kemudian setelah itu isi url dan variabel seperti di gambar di bawah ini

KEY	VALUE	DESCRIPTION
Key	Value	Description

Klik send kemudian akan ada response seperti gambar di bawah ini.

Status: 200 OK Time: 102 ms Size: 387 B

```

1 {
2   "id": 1,
3   "flightNumber": "1131",
4   "origin": "BDO - Bandar Udara Internasional Husein Sastranegara, Bandung",
5   "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
6   "time": "1970-07-07",
7   "pilot": {
8     "id": 1,
9     "licenseNumber": "1234",
10    "name": "Coki",
11    "flyHour": 139
12  }
13 }

```

d. Mengubah kodingan menjadi seperti di bawah ini

```

55 @RequestMapping(value = "/all")
56 public List<FlightModel> viewFlight(Model model) {
57     return flightService.findAllFlight();
58 }

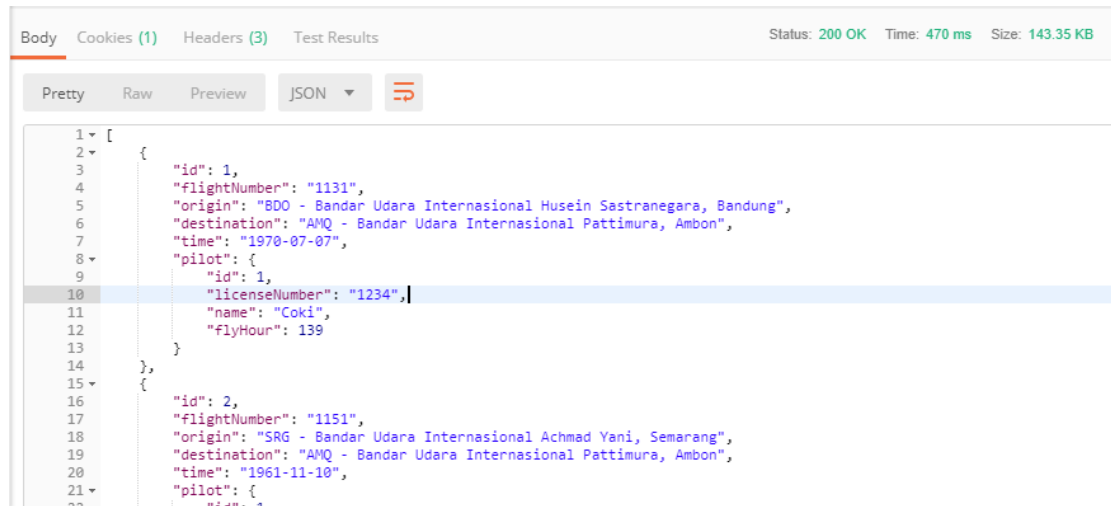
```

Kemudian setelah itu isi url dan variabel seperti di gambar di bawah ini

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 470 ms Size: 143.35 KB

Klik send kemudian akan ada response seperti gambar di bawah ini.



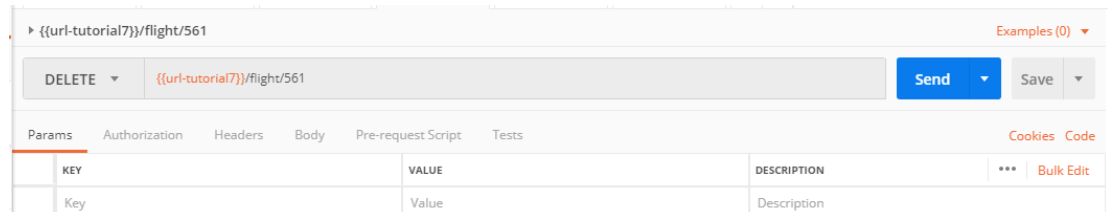
e. Mengubah kodingan menjadi seperti di bawah ini

```

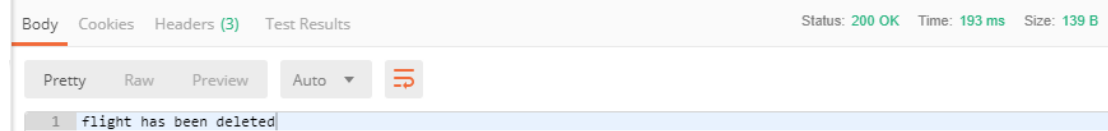
66 @DeleteMapping(value =("/{flightId}")
67 public String delete(@PathVariable(value = "flightId") long flightId) {
68     flightService.deleteFlightById(flightId);
69     return "flight has been deleted";
70 }

```

Kemudian setelah itu isi url dan variabel seperti di gambar di bawah ini



Klik send kemudian akan ada response seperti gambar di bawah ini.



- Untuk menampilkan nama airport dari kota yang diinginkan dengan memasukkan url yang telah di masukkan ke dalam setting pada sts ke dalam postman. Masukkan url dan juga apikey yang diberikan pada pdf ke file setting. Kemudian panggil di kelas Airport Controller. Setelah pada sts selesai semua, kemudian lakukan percobaan dengan menggunakan postman. Berikut adalah gambar-gambar yang diambil dalam pengerjaan nomor 2.

```

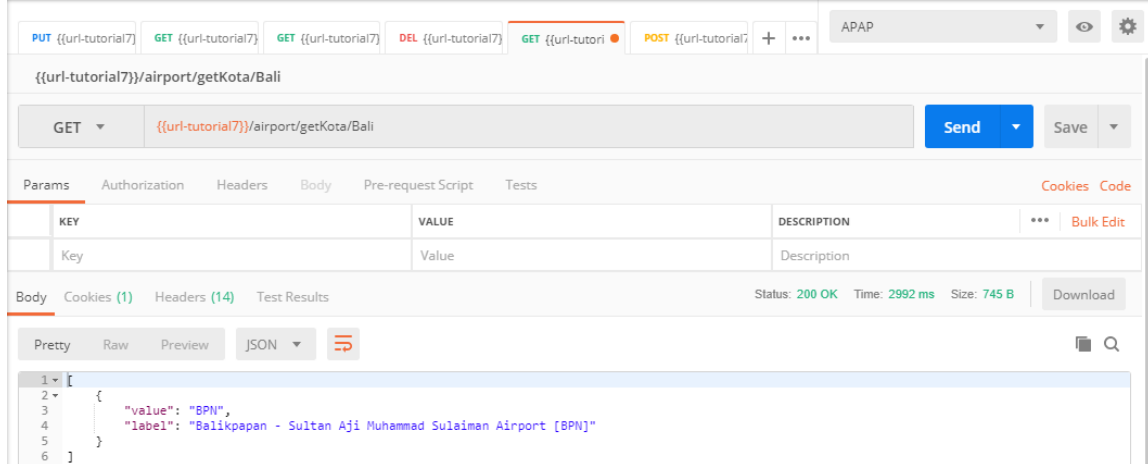
3 public class Setting {
4     final public static String pilotUrl = "https://1d999a83-2247-4d04-a7a1-ad13bfd6d0b2.mock.pstmn.io";
5     final public static String airportUrl = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete";
6     final public static String airportApikey = "08tfbSPFwX9nGzJ1W6mYahePpECpItD5\r\n";
7 }

```

```

16 public class AirportController {
17
18     @Autowired
19     private RestTemplate restTemplate;
20
21     @Bean
22     public RestTemplate restTemplate() {
23         return new RestTemplate();
24     }
25
26     @GetMapping(value = "/getKota/{namaKota}")
27     public ResponseEntity<String> getStatus(@PathVariable("namaKota") String kota) {
28         String path = Setting.airportUrl+"?apikey="+Setting.airportApiKey+"&term="+kota+"&country=ID";
29         System.out.println(path);
30         ResponseEntity<String> response = restTemplate.getForEntity(path, String.class);
31         return response;
32     }
33 }

```



The screenshot shows a REST client interface with the following details:

- Request Method:** GET
- Request URL:** {{url-tutorial7}}/airport/getKota/Bali
- Status:** 200 OK
- Time:** 2992 ms
- Size:** 745 B
- Response Body (JSON):**

```

1 {
2   {
3     "value": "BPN",
4     "label": "Balikpapan - Sultan Aji Muhammad Sulaiman Airport [BPN]"
5   }
6 }

```