

Pada tutorial 7 ini mahasiswa diajarkan mengenai penggunaan Rest API web service, pembuatan mock server pada postman, dan menggunakan service. Web service merupakan software yang dibuat untuk berkomunikasi melalui network, API adalah library yang berfungsi sebagai perantara komunikasi antar-komponen.

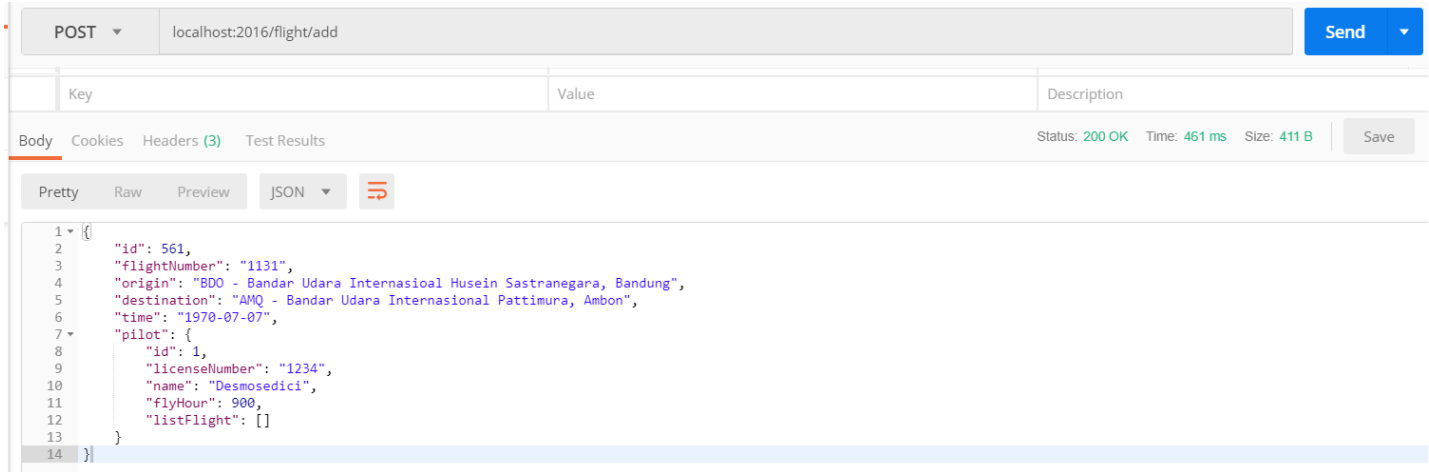
Latihan

1. Pada latihan satu mahasiswa diminta untuk membuat lima API untuk transaksi data flight, yaitu: Menambahkan flight baru, update flight, mencari flight berdasarkan nomor penerbangan, melihat keseluruhan penerbangan, dan menghapus penerbangan berdasarkan flightId.

```
24 @RestController
25 @RequestMapping("flight")
26 public class FlightController {
27     @Autowired
28     private FlightService flightService;
29
30     @GetMapping(value = "/all")
31     public @ResponseBody List<FlightModel> viewAll(){
32         List<FlightModel> flight = flightService.getAll();
33         return flight;
34     }
35
36     @PostMapping(value = "/add")
37     public FlightModel addFlightSubmit(@RequestBody FlightModel flight) {
38         flightService.addFlight(flight);
39         return flight;
40     }
41
42     @GetMapping(value = "/view/{flightNumber}")
43     public FlightModel flightView(@PathVariable("flightNumber") String flightNumber) {
44         FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNumber);
45         return flight;
46     }
47
48     @DeleteMapping(value = "/delete/{flightId}")
49     public String deleteFlight(@PathVariable("flightId") long flightId) {
50         FlightModel flight = flightService.getFlightDetailByFlightId(flightId);
51         flightService.deleteFlight(flight);
52         return "flight has been deleted";
53     }
54
55     @PutMapping(value="/update/{flightId}")
56     public String updateFlightSubmit(@PathVariable("flightId") long flightId,
57                                     @RequestParam("destination") String destination,
58                                     @RequestParam("origin") String origin,
59                                     @RequestParam("time") Date time) {
60         FlightModel flight = flightService.getFlightDetailByFlightId(flightId);
61         if (flight.equals(null)) {
62             return "Couldn't find your flight";
63         }
64
65         flight.setDestination(destination);
66         flight.setOrigin(origin);
67         flight.setTime(time);
68         flightService.addFlight(flight);
69         return "update";
70     }
71 }
```

Untuk melakukan pengecekan tanpa perlu membuat html maka pengecekan dilakukan di postman, dan hasilnya adalah sebagai berikut :

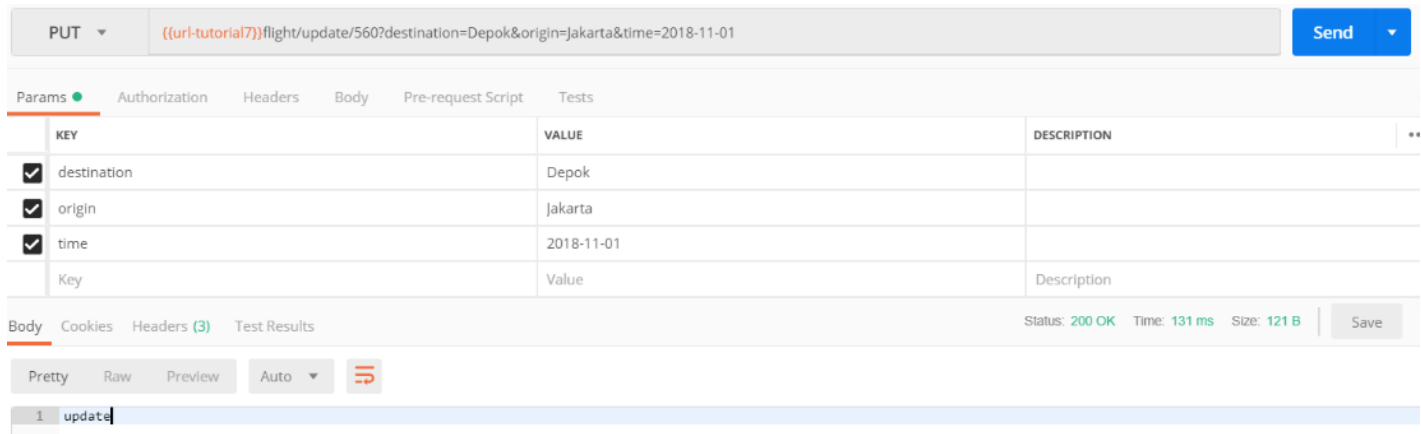
Add Flight :



Postman interface showing a POST request to `localhost:2016/flight/add`. The response is a JSON object with flight details.

```
1 {
2   "id": 561,
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": "Desmosedici",
11    "flyHour": 900,
12    "listFlight": []
13  }
14 }
```

Update Flight :

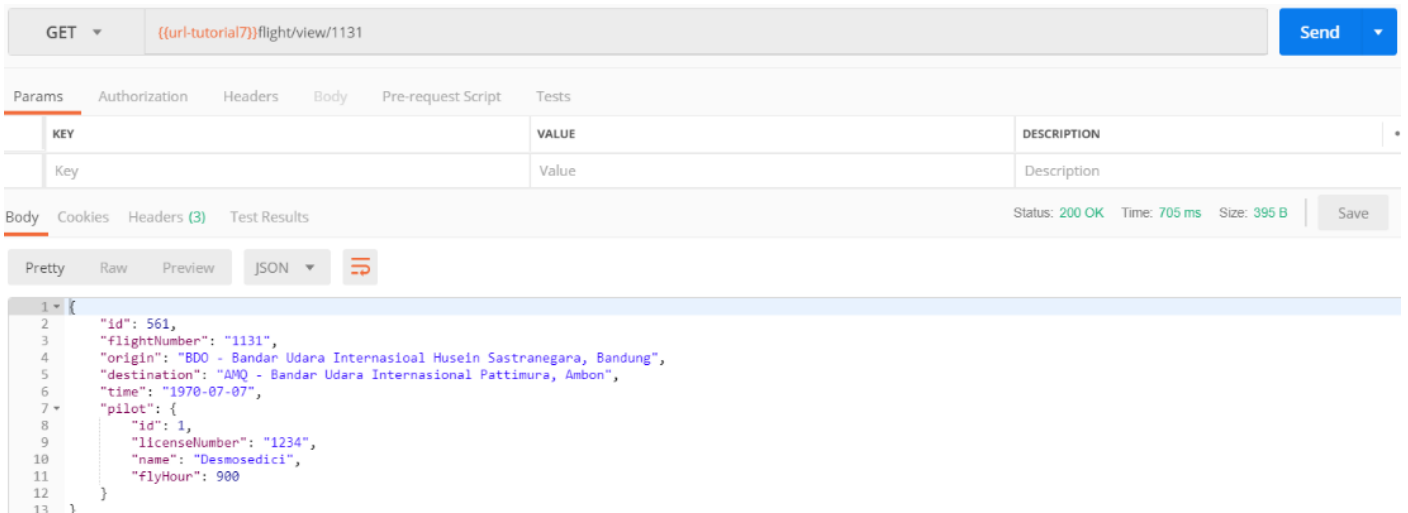


Postman interface showing a PUT request to `{{url-tutorial7}}/flight/update/560?destination=Depok&origin=Jakarta&time=2018-11-01`. The response is a JSON object with flight details.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> destination	Depok	
<input checked="" type="checkbox"/> origin	Jakarta	
<input checked="" type="checkbox"/> time	2018-11-01	
Key	Value	Description

```
1 update
```

Search Flight by flight number :



Postman interface showing a GET request to `{{url-tutorial7}}/flight/view/1131`. The response is a JSON object with flight details.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "id": 561,
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": "Desmosedici",
11    "flyHour": 900
12  }
13 }
```

View all flights

GET

{{url-tutorial7}}flight/all

Send

Params

Authorization

Headers

Body

Pre-request Script

Tests

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (3)

Test Results

Status: 200 OK Time: 65 ms Size: 106.25 KB Save

Pretty

Raw

Preview

JSON

```
1 [
2   {
3     "id": 1,
4     "flightNumber": "1131",
5     "origin": "BDO - Bandar Udara Internasional Husein Sastranegara, Bandung",
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, Semarang",
13    "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
14    "time": "1961-11-10"
15  },
16  {
17    "id": 3,
18    "flightNumber": "1176",
19    "origin": "TIM - Bandar Udara Internasional Mozes Kilangin, Mimika",
20    "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
21    "time": "1986-10-17"
22  },
23  {
24    "id": 4,
25    "flightNumber": "1239",
```

Delete Flights

DELETE

{{url-tutorial7}}flight/delete/2

Send

Params

Authorization

Headers

Body

Pre-request Script

Tests

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (3)

Test Results

Status: 200 OK Time: 505 ms Size: 139 B Save

Pretty

Raw

Preview

Auto

```
1 flight has been deleted
```

- Pada latihan dua mahasiswa diminta untuk menampilkan airport yang berada di kota yang dimasukkan sesuai input di Indonesia, menggunakan service lain yang telah disediakan oleh Amadeus.

```
1 package com.apap.tutorial7.rest;
2
3 public class Setting {
4     final public static String pilotUrl = "https://ac974a33-402c-4316-a599-1dd9738d3bd7.mock.pstmn.io";
5     final public static String airportUrl = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=lhvTZEfaUy1cZAnxJtUXsBWYDuKxwzZR&term=";
6 }
7
```

Setting url API sesuai apps yang didaftarkan pada amadeus.

```

13 @RestController
14 @RequestMapping("/airport")
15 public class AirportController {
16
17     @Autowired
18     RestTemplate restTemplate;
19
20     @Bean
21     public RestTemplate restAirport() {
22         return new RestTemplate();
23     }
24
25     @GetMapping(value = "/{city}")
26     public String getAirport(@PathVariable("city") String city) {
27         String path = Setting.airportUrl + city + "&country=ID&all_airports=true";
28         return restTemplate.getForEntity(path, String.class).getBody();
29     }
30

```

Membuat airport controller untuk menerima input nama kota dari user.
Setelah dicoba dengan GET dari postman, hasilnya sebagai berikut :

The screenshot shows a Postman interface with a GET request to the URL `((url-tutorial7))airport/Bandung`. The request is sent, and the response is displayed in the 'Body' tab. The response status is 200 OK, with a time of 2367 ms and a size of 233 B. The response body is formatted as JSON and shows the following data:

KEY	VALUE	DESCRIPTION
Key	Value	Description

The response body is also shown in a code editor with line numbers 1 through 6. The JSON content is:

```

1 {
2   "value": "BDO",
3   "label": "Bandung - Husein Sastranegara International Airport [BDO]"
4 }
5
6

```