

## TUTORIAL

### 1. Web Service

- *Web service* memfasilitasi komunikasi antar mesin melalui jaringan. Data yang digunakan adalah XML atau JSON.
- *API (Application Programming Interface)* memfasilitasi komunikasi antar komponen.

### 2. Service Producer

- *Service Producer* menyediakan layanan untuk memberikan data kepada *client*.
- Menggunakan *RestController*

### 3. Service Consumer

- *Service consumer* membutuhkan layanan lain untuk mendapatkan data yang dibutuhkan.
- Membuat *mock up* sebagai *service producer*, menggunakan Postman
- *Service consumer* sebagai penerima data dari *producer*, menggunakan *RestTemplate*
- Menambahkan *dependency* di *pom.xml* yaitu *com.fasterxml.jackson.core* untuk melakukan *data bind* ketika kita *post* ke *service* lain

## LATIHAN

### 1. Latihan 1

#### METHOD 1: Menampilkan Flight

Pada *controller*, tambahkan:

```
@GetMapping("/view/{flightNumber}")
public Optional<FlightModel> getFlight(@PathVariable("flightNumber") String flightNumber) {
    return flightService.getFlightDetailByFlightNumber(flightNumber);
}
```

Pada Postman buat *request* baru sebagai berikut:

The screenshot shows the Postman interface for a new request. The request name is 'Get a Flight'. The method is 'GET' and the URL is '{{\$url-tutorial7}}flight/view/1133'. There are 'Send', 'Save', and 'Examples (0)' buttons. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Params' tab is active, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table has one row with 'Key' and 'Value' in the first two columns, and 'Description' in the third. There are also 'Cookies' and 'Code' tabs on the right. At the bottom, there is a 'Response' section.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Kemudian klik *save* dan *send*, maka *response* akan dikembalikan

Postman interface showing a GET request to `{{url-tutorial7}}flight/view/1133`. The response is a JSON object with the following details:

```
{
  "id": 19,
  "FlightNumber": "1133",
  "origin": "UPG - Bandar Udara Internasional Sultan Hasanuddin, Maros",
  "destination": "BDJ - Bandar Udara Internasional Syamsuddin Noor, Banjarbaru",
  "time": "1992-02-01"
}
```

## METHOD 2: Menampilkan Seluruh Flight

Mengimplementasikan *method* pada *service* untuk menampilkan seluruh *flight* dalam bentuk *list*:

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

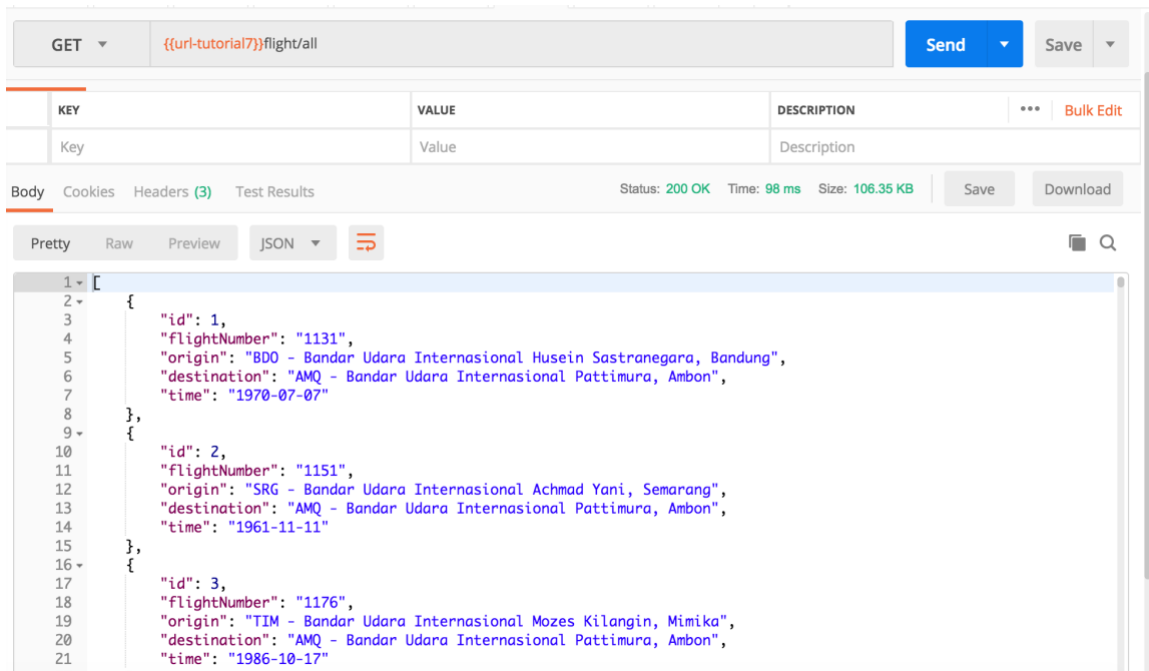
Pada *controller*, tambahkan:

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

Pada Postman buat *request* baru sebagai berikut:

Postman interface showing a GET request to `{{url-tutorial7}}flight/all`. The response status is 200 OK, Time: 98 ms, Size: 106.35 KB.

Kemudian klik *save* dan *send*, maka *response* akan dikembalikan



### METHOD 3: Menghapus Flight

Mengimplementasikan *method* pada *service* untuk menghapus *flight* berdasarkan ID:

```

@Override
public Boolean deleteFlightById(long id) {
    flightDb.deleteById(id);
    return true;
}

```

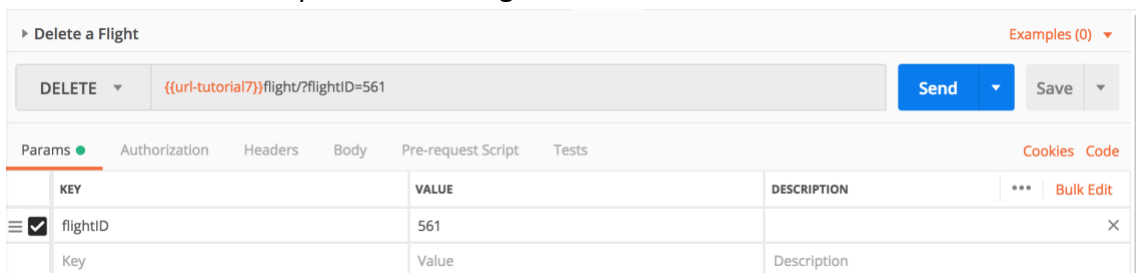
Pada *controller*, tambahkan:

```

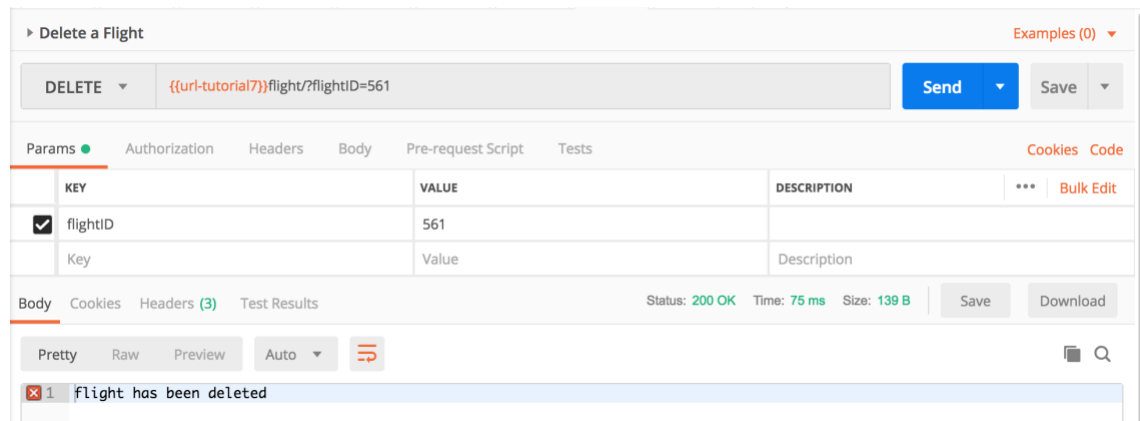
@DeleteMapping(value = "/")
public String deleteFlight(@RequestParam("flightID") long flightID) {
    flightService.deleteFlightById(flightID);
    return "flight has been deleted";
}

```

Pada Postman buat *request* baru sebagai berikut:



Kemudian klik *save* dan *send*, maka *response* akan dikembalikan



## METHOD 4: Mengubah Flight

Mengimplementasikan *method* pada *service* untuk mengubah *flight* berdasarkan ID:

```
@Override
public void updateFlight(long id, FlightModel flight) {
    FlightModel oldFlight = flightDb.getOne(id);
    oldFlight.setDestination(flight.getDestination());
    oldFlight.setOrigin(flight.getOrigin());
    flightDb.save(flight);
}
```

Pada *controller*, tambahkan:

```
@PostMapping("/update/{flightID}")
public String updateFlight(@PathVariable("flightID") long flightID,
    @RequestParam(value = "destination", required = false) String destination,
    @RequestParam(value = "origin", required = false) String origin) {
    FlightModel flight = flightService.getFlightById(flightID);

    if(flight.equals(null)) {
        return "Couldn't find your flight";
    }

    flight.setDestination(destination);
    flight.setOrigin(origin);
    flightService.updateFlight(flightID, flight);
    return "flight update success";
}
```

Pada Postman buat *request* baru sebagai berikut:

The screenshot shows the Postman interface for a PUT request named "Update a Flight". The URL is `{{url-tutorial7}}flight/update/561?destination=Singapura&origin=Surabaya`. The request body is empty. The Params tab is active, showing a table with the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> destination	Singapura	
<input checked="" type="checkbox"/> origin	Surabaya	
Key	Value	Description

Kemudian klik *save* dan *send*, maka *response* akan dikembalikan

The screenshot shows the Postman interface after sending the request. The status is 200 OK, Time is 83 ms, and Size is 137 B. The response body is displayed in the "Body" tab, showing the text "flight update success".

## METHOD 5: Menambahkan Flight

Agar bisa mendapatkan `licenseNumber` dari pilot, maka pada `FlightModel` ubah atribut pilot menjadi sebagai berikut:

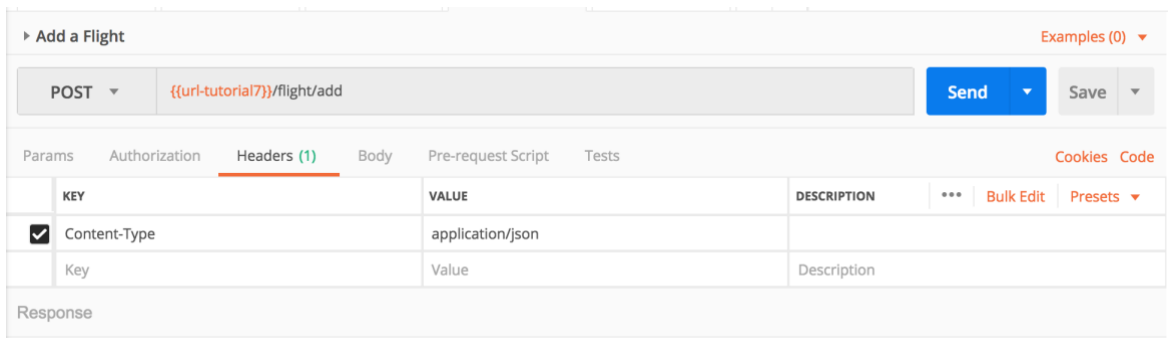
```
@JoinColumn(name = "pilot_licenseNumber", referencedColumnName =
    "license_number", nullable = false)
private String pilotLicenseNumber;
```

dan karena bagian tersebut pada `FlightModel` diubah, untuk sementara hapus atribut `listFlight` pada `PilotModel`.

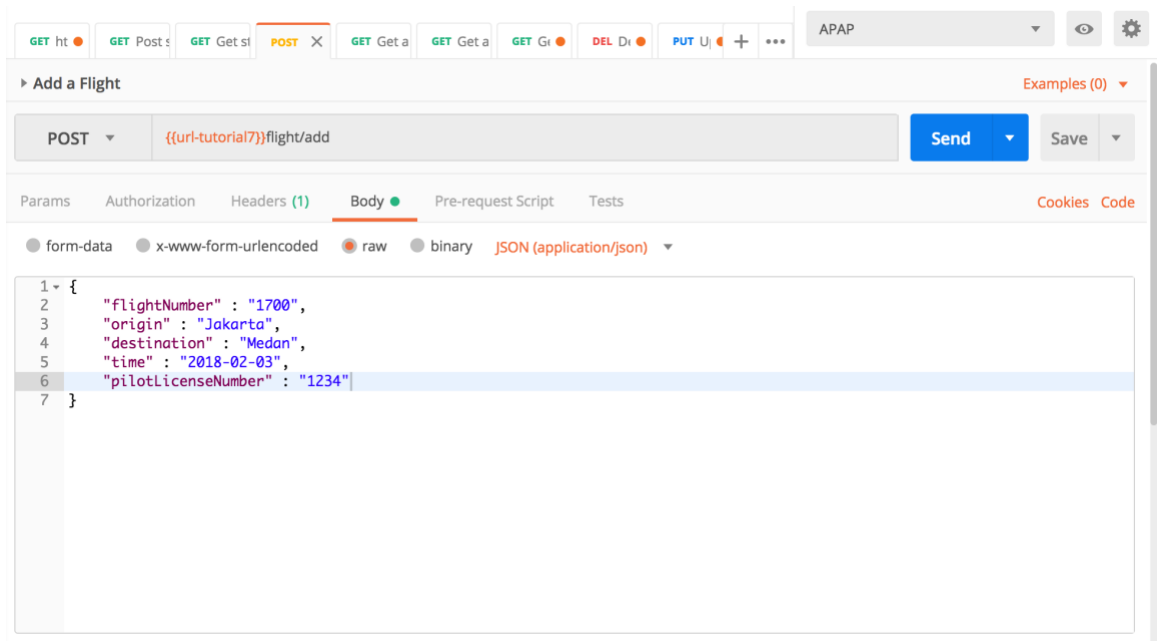
Pada *controller*, tambahkan:

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

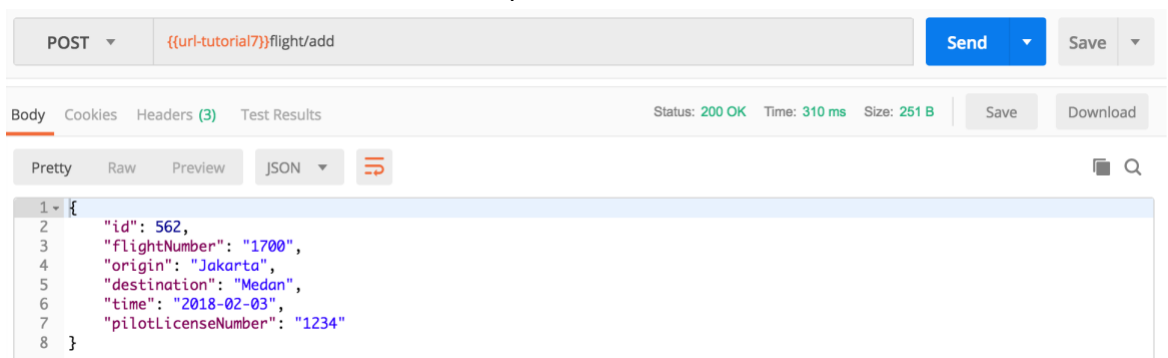
Pada Postman buat *request* baru sebagai berikut:



Lalu pada *body*, tambahkan:



Kemudian klik *save* dan *send*, maka *response* akan dikembalikan



## 2. Latihan 2

Buat akun di Amadeus dan dapatkan Consumer Key. Kemudian dapatkan API dari Sandbox. *Link* yang terbentuk dari keduanya, kemudian dimasukkan ke file Setting, sebagai berikut:

```
final public static String flightUrl =
```

```
"https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=AP2RxtcXs4YKy4jy7JZ79AwUiL  
DuRPym&term=";
```

Kemudian buat sebuah *class* baru bernama *AirportController* dengan isi sebagai berikut:

```
/**  
 * AirportController  
 */  
@RestController  
@RequestMapping("/airport")  
public class AirportController {  
    @Autowired  
    RestTemplate restTemplate;  
  
    @Bean  
    public RestTemplate restAirport() {  
        return new RestTemplate();  
    }  
  
    @GetMapping(value =("/{city}")  
    public String getStatus(@PathVariable("city") String city) throws Exception {  
        String path = Setting.airportUrl + city + "&country=ID";  
        return restTemplate.getForEntity(path, String.class).getBody();  
    }  
}
```

Buat *folder* baru pada Postman bernama *Airport*, buat *request* baru sebagai berikut:

The screenshot shows a Postman request configuration for a GET request. The name of the request is 'Get an Airport'. The URL is set to `{{url-tutorial7}}/airport/Jakarta`. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Params' tab is active, showing a table with one parameter: 'Key' with a value of 'Value'. There are also buttons for 'Send', 'Save', 'Cookies', and 'Code'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Klik *save* dan *send*, kemudian *response* akan dikembalikan.

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

- Request Section:**
  - Method: GET
  - URL: `{{url-tutorial7}}/airport/Jakarta`
  - Buttons: Send, Save
- Params Section:**

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Response Section:**
  - Status: 200 OK, Time: 2366 ms, Size: 400 B
  - Buttons: Save, Download
  - Format: Pretty (selected), Raw, Preview
  - Auto refresh: On
- Response Body (JSON):**

```
[
  {
    "value": "JKT",
    "label": "Jakarta [JKT]"
  },
  {
    "value": "CGK",
    "label": "Jakarta - Soekarno - Hatta International Airport [CGK]"
  },
  {
    "value": "HLP",
    "label": "Jakarta - Halim Perdanakusuma Airport [HLP]"
  }
]
```