Tutorial 7

1. **Web Service**
   - Web Service adalah sebuah software yang menyajikan layanan komunikasi antara dua atau lebih mesin melalui jaringan.
   - Komunikasi dilakukan melalui HTTP menggunakan teknologi seperti XML, SOAP, WSDL, dan UDDI.
   - Perbedaannya dengan API adalah Web Service bukan open source, sedangkan API adalah open source. Selain itu, API melayani komunikasi antar komponen, dan API tidak memerlukan jaringan internet untuk melakukan operasinya
2. **Service Producer**
   - *Service Producer* menyediakan layanan untuk melakukan manipulasi terhadap data (seperti: *update, add, delete)* dan pengambilan data (seperti: *get)* kepada *client*
   - Menggunakan RestController
3. **Service Consumer**
   - *Service Consumer* membutuhkan layanan lain untuk mendapatkan data yang dibutuhkan
   - Membuat *mock up* sebagai *service producer*, dengan menggunakan *Postman*
   - *Service consumer* digunakan sebagai menerima data dari *producer,* menggunakan RestTemplate
   - Menambahkan *dependency* pada pom.xml untuk melakukan *data bind* ketika melakukan operasi *post* ke *service* lain

Latihan

1. **Latihan 1**
   - **Method 1: Menambahkan Flight**
     - Pada FlightModel, sementara hapus bagian

```
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "pilot_licenseNumber", referencedColumnName =
"license_number")
```

```
@OnDelete(action = OnDeleteAction.CASCADE)
@JsonIgnore
private PilotModel pilot;
```

beserta setter dan getternya, dan tambahkan method

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

tambahkan setter dan getternya

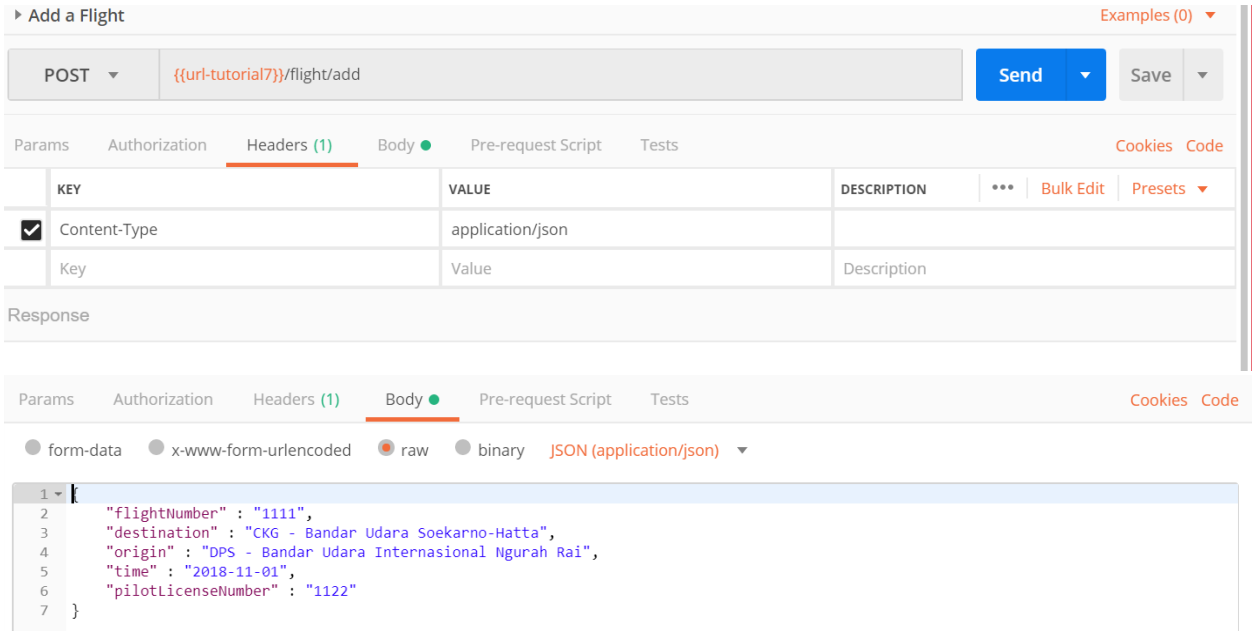
- Pada PilotModel, hapus method

```
@OneToMany(mappedBy = "pilot", fetch = FetchType.LAZY)
private List<FlightModel> listFlight = new ArrayList<FlightModel>();
```

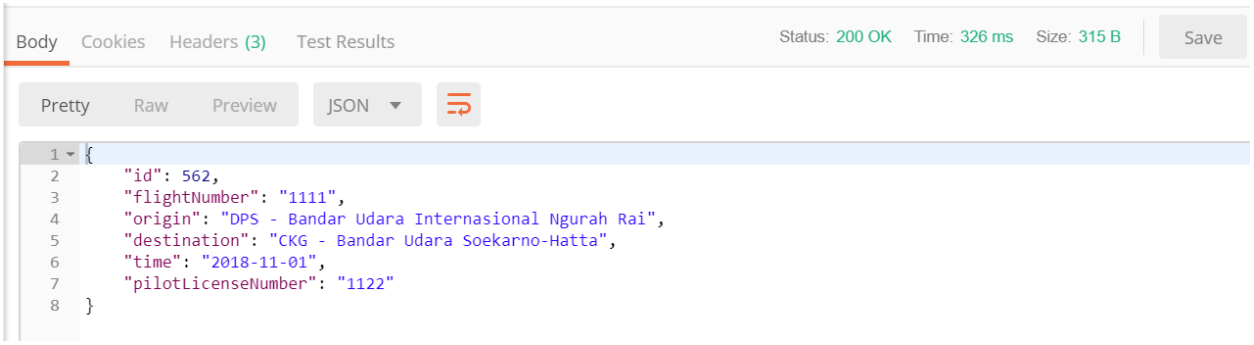
- Pada FlightController tambahkan method

```
@PostMapping(value="/add")
    public PilotModel addPilotSubmit(@RequestBody PilotModel
pilot) {
        return pilotService.addPilot(pilot);
    }
```

- Pada postman, dalam folder “Tutorial 7 – Web Service”, buat folder “flight” dan tambahkan request baru bernama “Add a Flight”, dan lakukan langkah seperti di gambar



- Ketika tombol send ditekan, akan muncul response seperti berikut



Method 2: Memperbarui Flight (*Update Flight*)

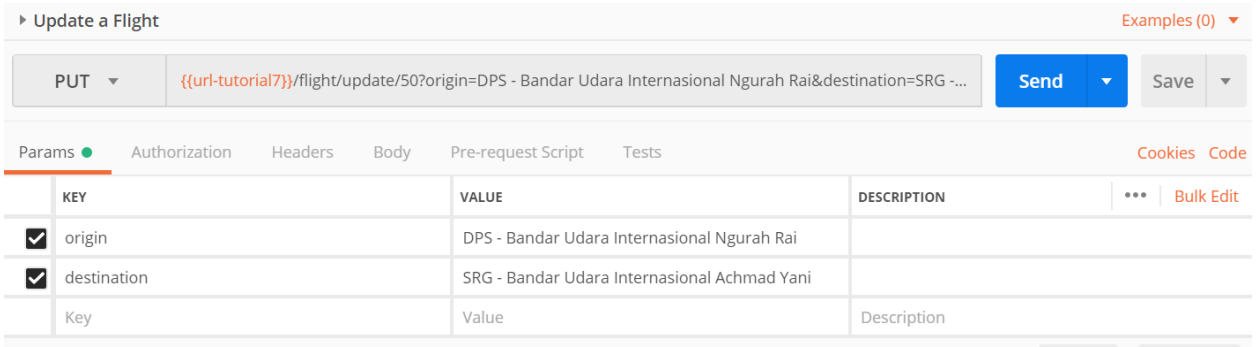
- Tambahkan method berikut pada service untuk mengubah flight berdasarkan flightId:

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

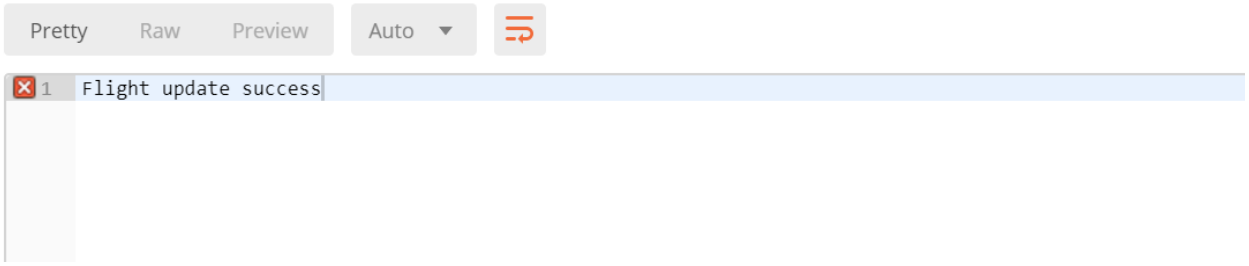
- Pada FlightController, tambahkan method berikut:

```
@PutMapping(value = "/update/{flightId}")
    public String updateFlightSubmit(@PathVariable("flightId")
long flightId,
		@RequestParam(value = "destination", required =
false) String destination,
		@RequestParam(value = "origin", required = false)
String origin) {
	FlightModel flight =
flightService.getFlightDetailById(flightId);
	if(flight.equals(null)) {
		return "Couldn't find your pilot";
	}
	flight.setDestination(destination);
	flight.setOrigin(origin);
	flightService.updateFlight(flightId, flight);
	return "Flight update success";
    }
```

- Pada postman, dalam folder “flight” tambahkan request baru bernama “Update a Flight”, dan lakukan langkah seperti di gambar



- Ketika tombol send ditekan, akan muncul response seperti berikut

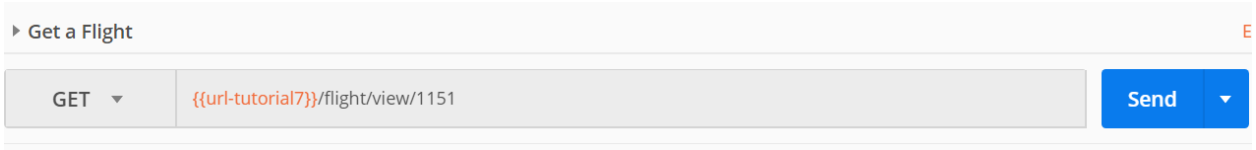


Method 3: Menampilkan Flight

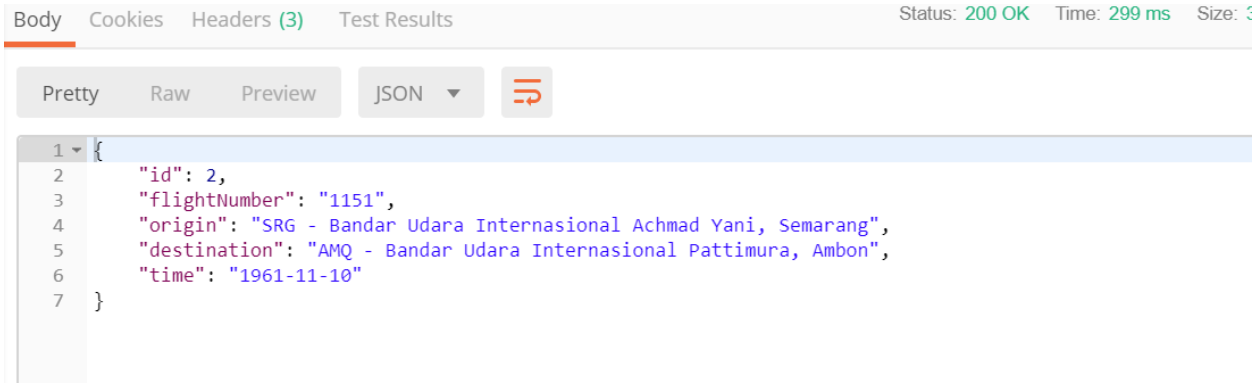
- Pada FlightController, tambahkan method berikut:

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

- Pada postman, dalam folder “flight” tambahkan request baru bernama “Get a Flight”, dan lakukan langkah seperti di gambar



- Ketika tombol send ditekan, akan muncul response seperti berikut



Method 4: Menampilkan Semua Flight

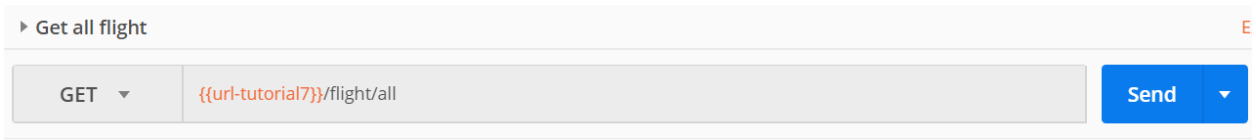
- Tambahkan method berikut pada service untuk melihat seluruh flight dalam bentuk list

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

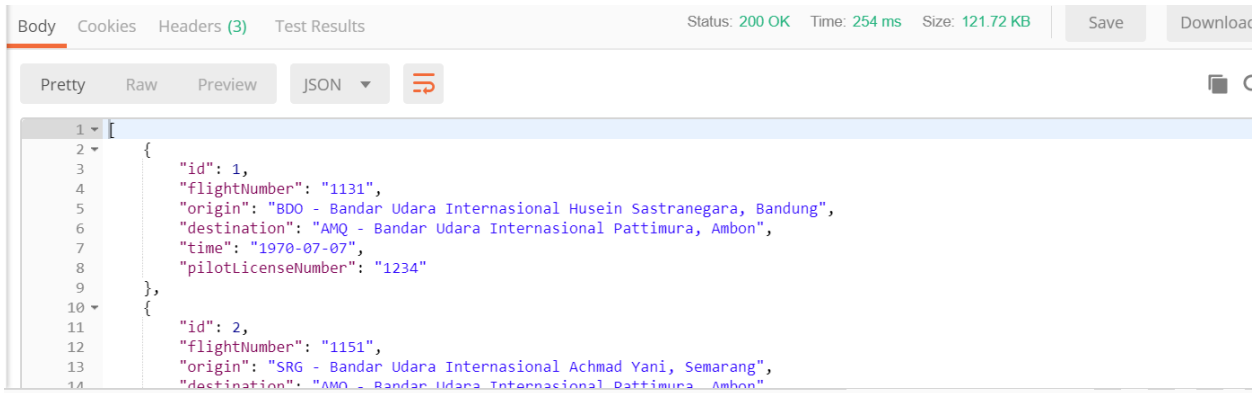
- Pada FlightController, tambahkan method berikut:

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

- Pada postman, dalam folder "flight" tambahkan request baru bernama "Get all Flight", dan lakukan langkah seperti di gambar



- Ketika tombol send ditekan, akan muncul response seperti berikut

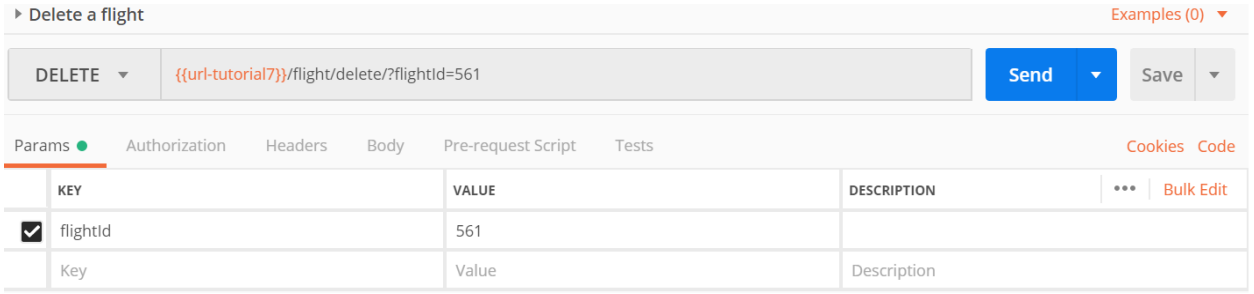


Method 5: Menghapus Flight

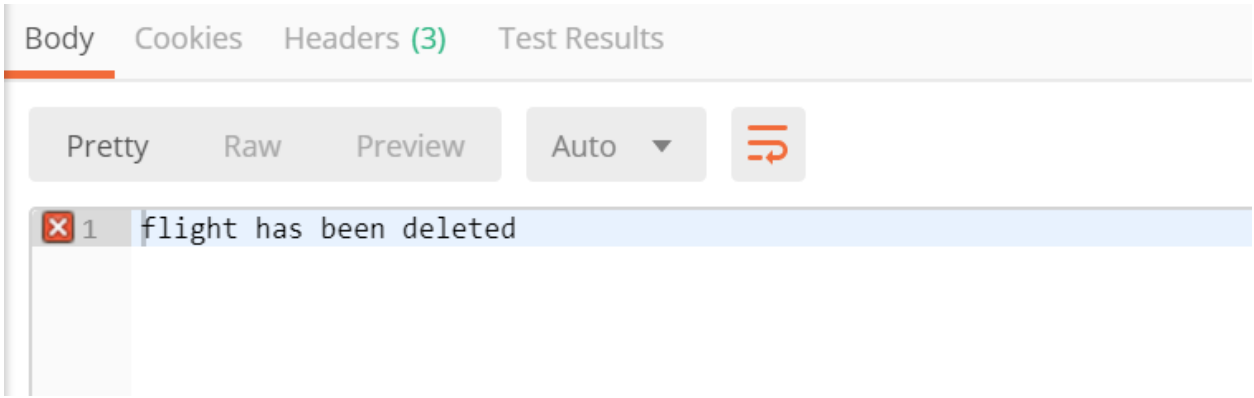
- Pada FlightController, tambahkan method berikut:

```
@DeleteMapping(value="/delete")
    public String deleteFlight(@RequestParam("flightId") long
flightId) {
        FlightModel flight =
flightService.getFlightDetailById(flightId);

        flightService.deleteByFlightNumber(flight.getFlightNumber(
));
        return "flight has been deleted";
    }
```

- Pada postman, dalam folder "flight" tambahkan request baru bernama "Delete a Flight", dan lakukan langkah seperti di gambar



- Ketika tombol send ditekan, akan muncul response seperti berikut



2. Latihan 2

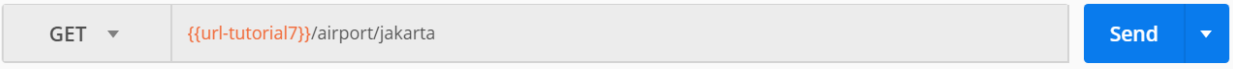
1. Membuat akun di sandbox.amadeus.com
2. Mendapatkan API Key
3. Dapatkan link API sandbox untuk airport ***"https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=Sv4EGPie7hUoeMfPcsHPVmYG3dayVesI&term="***
4. Tambahkan link tersebut di Setting.java sebagai sebuah variable

```
final public static String airportUrl = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=Sv4EGPie7hUoeMfPcsHPVmYG3dayVesI&term=";
```

5. Buat class baru pada controller bernama AirportController.java, yang berisi

```java
package com.apap.tutorial7.controller;

import java.util.Optional;

import com.apap.tutorial7.rest.Setting;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

/**
 * 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();
    }
```

6. Pada postman, buat folder baru bernama “airport” tambahkan request baru bernama “Get Airport”, dan lakukan langkah seperti di gambar



7. Ketika tombol send ditekan, akan muncul response seperti berikut