Tutorial 04 – Database

Tutorial ini mengasumsikan penggunaan pada *Windows*, tanpa melarang penggunaan OS lain, silakan menyesuaikan.

1. Pendahuluan
   - Install XAMPP (unduh di: https://www.apachefriends.org/index.html)
   - Jalankan STS
   - Buat *project* baru:
     - Nama : tutorial-04
     - Package: com.apap.tu04
     - *Dependencies*: DevTools, Thymeleaf, Web, JPA, dan MySQL
     - Pastikan bahwa MySQL sudah terinstall dengan membuka pom.xml

       ```
       <dependency>
           <groupId>mysql</groupId>
           <artifactId>mysql-connector-java</artifactId>
           <scope>runtime</scope>
       </dependency>
       ```

       Jika belum, Anda bisa menambahkan kode diatas ke pom.xml Anda dan Run as >> Maven *Install* untuk mengunduh *dependency* tersebut.
2. Membuat Model
   - Buatlah *package* baru dengan nama **com.apap.tu04.model**
   - Pada *package* model buatlah *class* PilotModel dan FlightModel dengan spesifikasi sbb.:

```
package com.apap.tu04.model;

import java.io.Serializable;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Column;
import javax.persistence.OneToMany;
import javax.persistence.FetchType;
import javax.persistence.CascadeType;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
@Table(name = "pilot")
public class PilotModel implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @NotNull
    @Size(max = 50)
    @Column(name = "license_number", nullable = false, unique = true)
    private String licenseNumber;

    @NotNull
    @Size(max = 50)
    @Column(name = "name", nullable = false)
    private String name;

    @NotNull
    @Column(name = "fly_hour", nullable = false)
    private int flyHour;

    @OneToMany(mappedBy = "pilot", fetch = FetchType.LAZY, cascade = CascadeType.PERSIST)
    private List<FlightModel> pilotFlight;
```

```java
package com.apap.tu04.model;

import java.io.Serializable;
import java.sql.Date;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "flight")
public class FlightModel implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @NotNull
    @Size(max = 50)
    @Column(name = "flight_number", nullable = false)
    private String flightNumber;

    @NotNull
    @Size(max = 50)
    @Column(name = "origin", nullable = false)
    private String origin;

    @NotNull
    @Size(max = 50)
    @Column(name = "destination", nullable = false)
    private String destination;

    @NotNull
    @Column(name = "time")
    private Date time;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "pilot_licenseNumber", referencedColumnName = "license_number", nullable = false)
    @OnDelete(action = OnDeleteAction.NO_ACTION)
    @JsonIgnore
    private PilotModel pilot;
```

- Tambahkan *method setter*, dan *getter*.

3. Membuat Repository

   Repository JPA merupakan interface yang mengandung fitur-fitur dan atribut elemen yang memungkinkan mendefinisikan repository beans.

   - Buatlah package baru dengan nama **com.apap.tu04.repository**
   - Pada *package* repository buatlah *interface* PilotDb dan FlightDb dengan spesifikasi sbb.:

```java
package com.apap.tu04.repository;

import com.apap.tu04.model.PilotModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 * PilotDb
 */
@Repository
public interface PilotDb extends JpaRepository<PilotModel, Long> {
    PilotModel findByLicenseNumber(String licenseNumber);
```

```java
package com.apap.tu04.repository;

import com.apap.tu04.model.FlightModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 * FlightDb
 */
@Repository
public interface FlightDb extends JpaRepository<FlightModel, Long>{
```

4. Membuat Service
   - Buatlah *package* baru dengan nama **com.apap.tu04.service**
   - Pada *package* service buatlah *interface* PilotService dan FlightService dengan spesifikasi sbb.:

```java
package com.apap.tu04.service;

import com.apap.tu04.model.PilotModel;

/**
 * PilotService
 */
public interface PilotService {
    PilotModel getPilotDetailByLicenseNumber(String licenseNumber);
    void addPilot(PilotModel pilot);
```

```java
package com.apap.tu04.service;

import com.apap.tu04.model.FlightModel;

/**
 * FlightService
 */
public interface FlightService {
    void addFlight(FlightModel flight);
```

   - Pada *package* yang sama buatlah *class* PilotServiceImpl dan FlightServiceImpl dengan spesifikasi sbb.:

```java
package com.apap.tu04.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.apap.tu04.model.PilotModel;
import com.apap.tu04.repository.PilotDb;

/**
 * PilotServiceImpl
 */
@Service
@Transactional
public class PilotServiceImpl implements PilotService {
    @Autowired
    private PilotDb pilotDb;

    @Override
    public PilotModel getPilotDetailByLicenseNumber(String licenseNumber) {
        return pilotDb.findByLicenseNumber(licenseNumber);
    }

    @Override
    public void addPilot(PilotModel pilot) {
        pilotDb.save(pilot);
    }
```

```java
package com.apap.tu04.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.apap.tu04.model.FlightModel;
import com.apap.tu04.repository.FlightDb;

/**
 * FlightServiceImpl
 */
@Service
@Transactional
public class FlightServiceImpl implements FlightService {
    @Autowired
    private FlightDb flightDb;

    @Override
    public void addFlight(FlightModel flight) {
        flightDb.save(flight);
    }
```

5. Membuat Controller dan Method Add
   - Buatlah package baru dengan nama **com.apap.tu04.controller**
   - Pada *package* controller buat *class* PilotController & FlightController dengan spesifikasi sbb.:

```java
package com.apap.tu04.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.apap.tu04.model.PilotModel;
import com.apap.tu04.service.PilotService;

/**
 * PilotController
 */
@Controller
public class PilotController {
    @Autowired
    private PilotService pilotService;

    @RequestMapping("/")
    private String home() {
        return "home";
    }

    @RequestMapping(value = "/pilot/add", method = RequestMethod.GET)
    private String add(Model model) {
        model.addAttribute("pilot", new PilotModel());
        return "addPilot";
    }

    @RequestMapping(value = "/pilot/add", method = RequestMethod.POST)
    private String addPilotSubmit(@ModelAttribute PilotModel pilot) {
        pilotService.addPilot(pilot);
        return "add";
    }
```

```java
package com.apap.tu04.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.apap.tu04.model.FlightModel;
import com.apap.tu04.model.PilotModel;
import com.apap.tu04.service.FlightService;
import com.apap.tu04.service.PilotService;

/**
 * FlightController
 */
@Controller
public class FlightController {
    @Autowired
    private FlightService flightService;

    @Autowired
    private PilotService pilotService;

    @RequestMapping(value = "/flight/add/{licenseNumber}", method = RequestMethod.GET)
    private String add(@PathVariable(value = "licenseNumber") String licenseNumber, Model model) {
        FlightModel flight = new FlightModel();
        PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
        flight.setPilot(pilot);
        model.addAttribute("flight", flight);
        return "addFlight";
    }

    @RequestMapping(value = "/flight/add", method = RequestMethod.POST)
    private String addFlightSubmit(@ModelAttribute FlightModel flight) {
        flightService.addFlight(flight);
        return "add";
    }
```

@ModelAttribute merupakan anotasi yang memungkinkan akses ke objek di View.

6. Membuat View
   - Home.html

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Home</title>
    </head>
    <body>
        <h2>Welcome!</h2>

        <h3>Tambah Pilot</h3>
        <td><a th:href="@{/pilot/add}">Tambah</a></td>

        <h3>Cari Pilot</h3>
        <form th:action="@{/pilot/view}" method="GET">
            License Number: <br>
            <input type="text" name="licenseNumber"/>
            <button type="submit">Cari</button>
        </form>
    </body>
</html>
```

- addPilot.hmtl

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Add Pilot</title>
    </head>
    <body>
        <h2>Welcome!</h2>

        <h3>Tambah Pilot</h3>
        <form th:action="@{/pilot/add}" th:object="${pilot}" method="POST">
            License Number: <br>
            <input type="text" name="licenseNumber"/> <br>
            Name: <br>
            <input type="text" name="name"/> <br>
            Fly Hour: <br>
            <input type="number" name="flyHour"/> <br><br>
            <button type="submit">Submit</button>
        </form>
    </body>
</html>
```

- addFlight.html

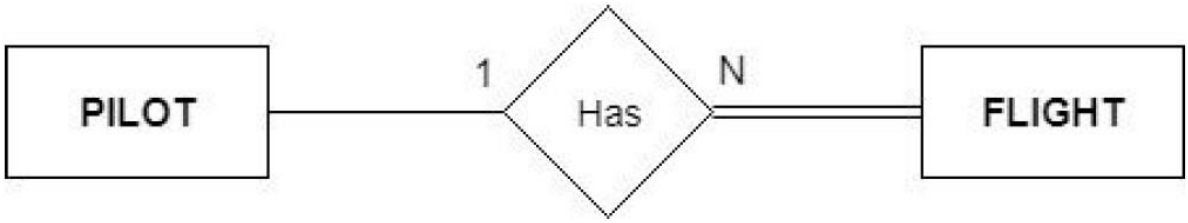
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Add Flight</title>
    </head>
    <body>
        <h2>Welcome!</h2>

        <h3>Tambah Penerbangan</h3>
        <form th:action="@{/flight/add}" th:object="${flight}" method="POST">
            <input type="hidden" th:field="*{pilot}"/>
            Flight Number: <br>
            <input type="text" name="flightNumber"/> <br>
            Origin: <br>
            <input type="text" name="origin"/> <br>
            Destination: <br>
            <input type="text" name="destination"/> <br>
            Time: <br>
            <input type="date" name="time"/> <br><br>
            <button type="submit">Submit</button>
        </form>
    </body>
</html>
```

- add.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Add</title>
    </head>
    <body>
        <h2>Data Berhasil Ditambahkan!</h2>
    </body>
</html>
```

7. Database

ERD untuk hubungan antara Pilot dan Flight adalah sbb.:



PILOT

- id [PRIMARY KEY]
  integer, NOT NULL, AUTO INCREMENT
- name
  varchar(50), NOT NULL
- licenseNumber
  varchar(50), NOT NULL, UNIQUE
- flyhour
  integer, NOT NULL

FLIGHT

- id [PRIMARY KEY]
  integer, NOT NULL, AUTO INCREMENT
- flight_number
  varchar(50), NOT NULL
- origin
  varchar(50), NOT NULL
- destination
  varchar(50), NOT NULL
- time
  date, DEFAULT NULL
- pilot_licenseNumber
  varchar(50), NOT NULL

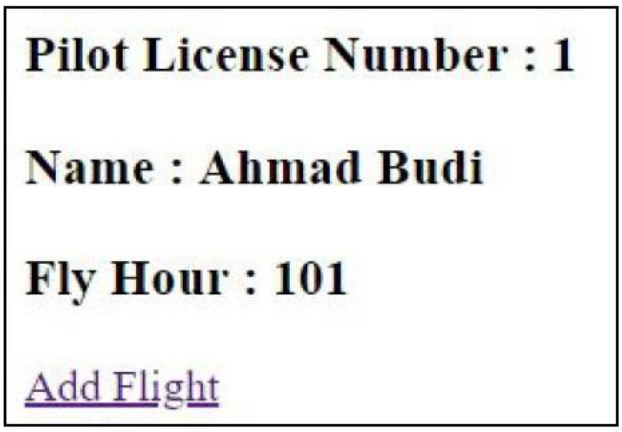
- Pahami ERD dan spesifikasi diatas. Implementasi database menggunakan JPA, tidak perlu membuat skema database secara manual.
- Jalankan XAMPP (MySQL dan Apache).
- Buat database dengan nama tu04 pada MySQL (akses: http://localhost/phpmyadmin/)
- Atur konfigurasi pada **application.properties** (package explorer: **src/main/resource**) sbb.:

```
# Konfigurasi koneksi MySQL
spring.datasource.platform=mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Database Access
spring.datasource.url=jdbc:mysql://localhost:3306/tu04

# Database Account
spring.datasource.username=root
spring.datasource.password=

# Optimize query untuk db MySQL
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect

# Apa ini?
spring.jpa.hibernate.ddl-auto=update
```

  Tidak hanya Spring, Framework lain juga mengenal *configuration file*. Misalkan .env pada Laravel, nodejs, python, dll.
- Jalankan program lalu akses http://localhost:8080/
- Lakukan **tambah Pilot**.

Latihan

1. Implementasikan method di PilotController untuk menampilkan seorang pilot berdasarkan *license number*. Method ini menerima parameter licenseNumber dan mengembalikan view (**view-pilot.html**). Buatlah tampilan view-pilot.html minimal seperti gambar dibawah ini. Tambahkan juga link/tombol untuk menambahkan penerbangan pada bagian bawah view. Lakukan **tambah Flight**.



2. Tambahkan pada view pilot agar menampilkan **daftar flight dari pilot** yang tampil (misalnya pada bagian bawah setelah link/tombol Add Flight).
3. Buatlah fitur **delete** untuk menghapus seorang pilot dan sebuah penerbangan.
4. Buatlah fitur **update** untuk seorang pilot dan sebuah penerbangan. Notes: License Number seorang pilot tidak bisa diubah.
5. Buatlah fitur untuk melihat **daftar flight** termasuk menampilkan nama dan license number dari pilot yang bertugas (misalnya dalam bentuk tabel).

Pengumpulan

1. *Screen capture* pengerjaan latihan yang menunjukkan bahwa fungsi berhasil diimplementasi (Misalnya *output* pada browser, *entry* database, dll. silakan jika perlu lebih dari satu gambar). Pengumpulan dituliskan dalam satu file dengan format **npm_nama-lengkap-anda.pdf** dan unggah ke submission slot yang disediakan di Scele.
2. Folder tutorial-04 dikumpulkan dengan cara push ke GitHub (https://github.com/achmad-f-abka/apap).