https://github.com/lixy1979/java_week_18

https://youtu.be/w3h6myWnb7A

1. Depency

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.7.11</version>
<relativePath /> <!-- lookup parent from repository -->
</parent>
<groupId>com.promineotech</groupId>
<artifactId>person-sighting</artifactId>
<version>1.0.0.1-SNAPSHOT</version>
<name>person-sighting</name>
<description>Person Sighting</description>
<properties>
<java.version>17</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<!-- Bean Validation ========================================== -->
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-validation</artifactId>
</dependency>

<!-- OpenAPI dependency ======================================== -->
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-ui</artifactId>
```

```
<version>1.7.0</version>
</dependency>


<!-- Database Ddependencies ======================================== -->
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-jdbc -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>

<!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
<dependency>
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
</dependency>

<!-- Test dependencies ================================================ -->
<!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
<dependency>
<groupId>com.h2database</groupId>
<artifactId>h2</artifactId>
<scope>test</scope>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<configuration>
<excludes>
<exclude>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
</exclude>
</excludes>
</configuration>
</plugin>
</plugins>
</build>

</project>
```

2. docker-compose yml

```
# This docker-compose file starts MySQL and Flyway in a bridge network. When the Flyway container becomes active, it
# creates the tables in the MySQL database and populates them.
```

```
version: '3.7'

services:
# This section defines the mysql service, named "db". The name can be referenced in things like URLs.
So, we can tell
# Flyway to find the database at jdbc:mysql://db and Docker Compose will fill in the network details.
db:
container_name: mysql
image: mysql:latest
restart: always
environment:
# These environment variables cause MySQL to create the users username/password
# root/root and dev/dev. It also creates a database named person.
MYSQL_ROOT_USER: root
MYSQL_ROOT_PASSWORD: root
MYSQL_DATABASE: person
MYSQL_USER: person
MYSQL_PASSWORD: person
ports:
# Forward host port 8306 to guest port 3306 (MySQL default port). This means that applications
external to the
# container cluster can access MySQL within the container on port 8306 and the requests are
forwarded to port 3306
# within the cluster.
- "8306:3306"
networks:
- person

# Flyway is used to create the tables and populate them with data. The migration files are found in
# src/functional-test/resources. The schema is applied first to create the tables
(V1.0__person_Schema.sql) and
# then the data is applied in V1.1__person_Data.sql. Note that these files are also applied for each
functional
# (integration) test using the @Sql annotation in the functional test classes.
flyway:
container_name: flyway
image: flyway/flyway:latest
command: migrate
# These environment variables are used in ./flyway/conf/flyway.conf to tell Flyway which database to
connect to.
environment:
FLYWAY_URL: jdbc:mysql://db
FLYWAY_SCHEMAS: person
FLYWAY_USER: person
FLYWAY_PASSWORD: person
# Set the retry count to let the database come up before Flyway gives up.
FLYWAY_CONNECT_RETRIES: 60
volumes:
# Create a volume between ./src/functional-test/resources/flyway/migrations in the host and
/flyway/sql in the
# container. This allows Flyway to grab the migration files from the default location.
- ./src/test/resources/flyway/migrations:/flyway/sql
# Create a volume between ./src/functional-test/resources/flyway/conf on the host and /flyway/conf
in the
# container. This allows Flyway to read configuration fron the default configuration location.
- ./src/test/resources/flyway/conf:/flyway/conf
depends_on:
```

```
- db
networks:
- person

# Create a bridge network between the MySQL container and the Flyway container.
networks:
person:
driver: bridge
name: person-to-person
```

3. application.yaml

```
spring:

datasource:

password: person

username: person

url: jdbc:mysql://localhost:3306/person


logging:

level:

root: warn

'[com.promineotech]': debug
```

4. application-test.yaml

```
spring:

datasource:
url: jdbc:h2:mem:person;mode=MYSQL
logging:

level:

root: warn

'[com.promineotech]': debug
```

5. flyway.conf

```
# The ${placeholders} are replaced by Flyway with values from environment variables with the same
# name. The environment variables are set when the container is created by Docker.
flyway.url=${FLYWAY_URL}
flyway.schemas=${FLYWAY_SCHEMAS}
flyway.user=${FLYWAY_USER}
flyway.password=${FLYWAY_PASSWORD}
flyway.connectRetries=${FLYWAY_CONNECT_RETRIES}
```

6. V1.0__Person_Schema.sql

```
DROP TABLE IF EXISTS person_sighting;
DROP TABLE IF EXISTS sighting;
DROP TABLE IF EXISTS person;

CREATE TABLE person(
person_pk int unsigned NOT NULL AUTO_INCREMENT,
person_id VARCHAR(45) NOT NULL,
family_name VARCHAR(45) NOT NULL,
given_name VARCHAR(45) NOT NULL,
birthday DATETIME NOT NULL,
gender VARCHAR(10) NOT NULL,
missing_date DATETIME NOT NULL,
Home_province_id VARCHAR(40) NOT NULL,
PRIMARY KEY (person_pk),
UNIQUE KEY (person_id)
);
CREATE TABLE sighting(
sighting_pk int unsigned NOT NULL AUTO_INCREMENT,
sighting_id VARCHAR(45) NOT NULL,
sighting_date DATETIME NOT NULL,
sighting_province_id VARCHAR(40) NOT NULL,
PRIMARY KEY (sighting_pk),
UNIQUE KEY (sighting_id)
);
CREATE TABLE person_sighting(
person_sighting_pk int unsigned NOT NULL AUTO_INCREMENT,
person_sighting_id VARCHAR(45) NOT NULL,
sighting_fk int unsigned NOT NULL,
person_fk int unsigned NOT NULL,
UNIQUE KEY (person_sighting_id),
PRIMARY KEY (person_sighting_pk),
FOREIGN KEY (sighting_fk) REFERENCES sighting (sighting_pk) ON DELETE CASCADE,
FOREIGN KEY (person_fk) REFERENCES person (person_pk) ON DELETE CASCADE
);
```

7. V1.1__Person_Data.sql

```
-- Person
INSERT INTO person (person_id, family_name, given_name, birthday, gender, missing_date,
Home_province_id) VALUES('YANG_BO', 'Yang',  'Bo', '2018-12-01', 'male', '2022-1-28', 'HENAN' );

INSERT INTO person (person_id, family_name, given_name, birthday, gender,
missing_date,Home_province_id) VALUES('WEIXUE', 'Wei',  'Xue', '2015-8-11', 'female', '2021-5-11',
'YUNNAN');

INSERT INTO person (person_id, family_name, given_name, birthday, gender, missing_date,
Home_province_id) VALUES('WUBIN', 'Wu',  'Bin', '2019-6-22', 'female', '2020-7-3',  'SHANDONG');

-- Sighting
INSERT INTO sighting (sighting_id,sighting_date,  sighting_province_id) VALUES('WEIXUE',   '2021-6-
17', 'FUJIAN' );
```

```
INSERT INTO sighting (sighting_id, sighting_date, sighting_province_id) VALUES('YANG_BO',  '2022-3-
15', 'GUIZHOU');
```

8. Person. entity

```
package com.promineotech.person.entity;


import java.time.LocalDate;
import lombok.Builder;
import lombok.Data;

@Data
@Builder

public class Person {
private Long personPK;
private String personId;
private String familyName;
private String givenName;
private LocalDate birthday;
private String gender;
private LocalDate missingDate;
private String homeProvinceId;

}

package com.promineotech.person.entity;


import java.time.LocalDate;

import lombok.Builder;
import lombok.Data;

@Data
@Builder

public class Sighting {
private Long sightingPK;
private String sightingId;
private LocalDate sightingDate;
private String sightingProvinceId;

}

package com.promineotech.person.entity;


import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.Builder;
import lombok.Data;

@Data
```

```
@Builder
public class PersonSighting {
private Long personSightingPK;
private Person person;
private Sighting sighting;

@JsonIgnore
public Long getPersonSightingPK() {
return personSightingPK;
}
}

package com.promineotech.person.entity;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import org.hibernate.validator.constraints.Length;
import lombok.Data;

@Data
public class PersonSightingRequest {

@NotNull
@Length(max = 30)
@Pattern(regexp = "[\\w\\s]*")
private String person;
// @NotNull
// @Length(max = 30)
// @Pattern(regexp = "[\\w\\s]*")
// private String familyName;
// @NotNull
// @Length(max = 30)
// @Pattern(regexp = "[\\w\\s]*")
// private String givenName;
// @PastOrPresent
// @NotNull
// @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
// private LocalDate birthday;
// @NotNull
// @Length(max = 30)
// @Pattern(regexp = "[\\w\\s]*")
// private String gender;
//
// @PastOrPresent
// @NotNull
// @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
// private LocalDate missingDate;
// @NotNull
// @Length(max = 30)
// @Pattern(regexp = "[\\w\\s]*")
// private String homeProvince;
@NotNull
@Length(max = 30)
@Pattern(regexp = "[\\w\\s]*")
private String sighting;
// @PastOrPresent
// @NotNull
// @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
// private LocalDate sightingDate;
// @NotNull
// @Length(max = 30)
// @Pattern(regexp = "[\\w\\s]*")
// private String sightingProvince;


}
```