ORM

author Kacper Bohaczyk
version 16 -04-2024

Requirements

- Docker
- Clonec Repository
- IDE - InteliJ

Guide

First you need to clone the repository in your desired destination

`git clone [https://github.com/spring-guides/gs-accessing-data-mysql.git] (https://github.com/spring-guides/gs-accessing-data-mysql.git)`

Add dependencies to build.gradle file

```
plugins {
    id 'org.springframework.boot' version '3.2.0'
    id 'io.spring.dependency-management' version '1.1.4'
    id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
```

```
    implementation 'com.mysql:mysql-connector-j:8.3.0'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

test {
    useJUnitPlatform()
}
```

Install docker and setup the database

Create MySQL-Container

```
bash
docker run --name my-mysql-container -e MYSQL_ROOT_PASSWORD=mysql123 -p
3306:3306 -d mysql:latest
```

Enter the Container

```
 mysql -password
```

Create a database

```
create database db_example; -- create database
create user 'springuser'@'%' identified by 'mysql123'; -- create user
grant all on db_example.* to 'springuser'@'%';
```

Bugs

!Did not open specific enough
!Did not mark directory as src
! clean bootRun does not work
! Tried in Windows - Docker Desktop runs on Linux Prozessor
! Tried in VirtualMachine

Questions

- What is ORM and how is JPA used?

        - **ORM (Object-Relational Mapping)** is a programming technique used to convert data between incompatible type systems, such as between object-oriented programming languages and relational databases. It allows developers to interact with databases using an object-oriented paradigm.
    - **JPA (Java Persistence API)** is a Java specification for managing relational data in applications using Java EE and Java SE environments. It provides a set of interfaces and annotations for mapping Java objects to database tables and vice versa. Developers use JPA to simplify database interactions and make them more object-oriented.
- What is the application.properties used for and where must it be stored?
    - The `application.properties` file is commonly used in Spring Boot applications to store configuration properties. These properties can include database connection settings, server port configurations, logging levels, etc.
    - It must be stored in the `src/main/resources` directory of your Spring Boot project. When the application is packaged, these properties are bundled into the resulting JAR or WAR file.
- Which annotations are frequently used for entity types? Which key points must be observed?
    - In JPA, the most frequently used annotations for entity types include `@Entity`, `@Table`, `@Id`, `@GeneratedValue`, `@Column`, `@ManyToOne`, `@OneToMany`, etc.
- What methods do you need for CRUD operations?
    - CREATE, READ, UPDATE, DELETE

Find out which methods are available for the CrudRepository to collect data

`long`	`count()`	Returns the number of entities available.
`Iterable <T>`	`findAll()`	Returns all instances of the type.
`Iterable <T>`	`findAllById(Iterable <ID> ids)`	Returns all instances of the type `T` with the given IDs.
`Optional <T>`	`findById(ID id)`	Retrieves an entity by its id.
`<S extends T> S`	`save(S entity)`	Saves a given entity.

Sources

https://spring.io/guides/gs/accessing-data-mysql
https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/CrudRepository.html

https://stackoverflow.com/

ChatGPT