Exercise 1: Spring Data JPA- A quick Example

Solution:

SQL Queries:

CREATE SCHEMA ormlearn;

USE ormlearn;

CREATE TABLE country(

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

INSERT INTO country VALUES ('UK', 'United Kingdom');

INSERT INTO country VALUES ('CA', 'Canada');

SELECT \* FROM country;

As the IntelliJ community version does not have Spring Initializr,use the web Spring Initializer,input necessities and download the zip file.

Opening the zip file,IntelliJ recognizes it as a maven file and imports it

Pom.xml:

<?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>4.0.0-SNAPSHOT</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>orm-learn</name>  
 <description>Demo project for Spring Data JPA and Hibernate</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</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>  
 </plugin>  
 </plugins>  
 </build>  
 <repositories>  
 <repository>  
 <id>spring-milestones</id>  
 <name>Spring Milestones</name>  
 <url>https://repo.spring.io/milestone</url>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 </repository>  
 <repository>  
 <id>spring-snapshots</id>  
 <name>Spring Snapshots</name>  
 <url>https://repo.spring.io/snapshot</url>  
 <releases>  
 <enabled>false</enabled>  
 </releases>  
 </repository>  
 </repositories>  
 <pluginRepositories>  
 <pluginRepository>  
 <id>spring-milestones</id>  
 <name>Spring Milestones</name>  
 <url>https://repo.spring.io/milestone</url>  
 <snapshots>  
 <enabled>false</enabled>  
 </snapshots>  
 </pluginRepository>  
 <pluginRepository>  
 <id>spring-snapshots</id>  
 <name>Spring Snapshots</name>  
 <url>https://repo.spring.io/snapshot</url>  
 <releases>  
 <enabled>false</enabled>  
 </releases>  
 </pluginRepository>  
 </pluginRepositories>  
  
</project>

Application.properties:

# Spring Framework and application log  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug  
  
# Hibernate logs for displaying executed SQL, input and output  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
  
# Log pattern  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database configuration  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn  
spring.datasource.username=root  
spring.datasource.password=Root@123  
  
# Hibernate configuration  
spring.jpa.hibernate.ddl-auto=validate  
# Remove or update the dialect - Spring Boot will auto-detect  
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

OrmLearnApplication.java:

package com.cognizant.ormlearn;  
  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
  
import java.util.List;  
  
@SpringBootApplication  
public class OrmLearnApplication {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(OrmLearnApplication.class);  
 private static CountryService *countryService*;  
  
 public static void main(String[] args) {  
 *LOGGER*.info("Starting ORM Learn Application");  
  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *countryService* = context.getBean(CountryService.class);  
  
 *LOGGER*.info("Inside main");  
  
 // Test the application  
 *testGetAllCountries*();  
 }  
  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.debug("countries={}", countries);  
 *LOGGER*.info("End");  
 }  
}

Country.java:

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name="country")  
public class Country {  
  
 @Id  
 @Column(name="co\_code")  
 private String code;  
  
 @Column(name="co\_name")  
 private String name;  
  
 // Default constructor  
 public Country() {  
 }  
  
 // Parameterized constructor  
 public Country(String code, String name) {  
 this.code = code;  
 this.name = name;  
 }  
  
 // Getters and Setters  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 // toString method  
 @Override  
 public String toString() {  
 return "Country{" +  
 "code='" + code + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

CountryRepository.java:

package com.cognizant.ormlearn.repository;  
  
import com.cognizant.ormlearn.model.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

CountryService.java:

package com.cognizant.ormlearn.service;  
  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional(readOnly = true)  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

A screen shot of a computer screen

AI-generated content may be incorrect.Output:

A screen shot of a computer screen

AI-generated content may be incorrect.