Spring Data JPA with Hibernate

1) Spring Data JPA - Quick Example

After downloading and installing all the software pre-requisites, went to the link provided to create Eclipse project with Spring Initializr. Changed the build option to Maven. Changed the Group and Artifact Id to com.cognizant and orm-learn respectively. Added the dependencies of MySQL Driver,Spring Data JPA and Spring Boot Devtools. Then downloaded the jar file and imported in to Eclipse as mentioned in the hands on document.

Code:

In MySQL first created a schema, then set it as default schema after that created a table countries, with attributes co\_code and co\_name.

CREATE SCHEMA ormlearn;

After this the sql query for creating the table

create table country(

co\_code varchar(2) primary key,

co\_name varchar(50)

);

Then for inserting data into the table crea the following sql queries were run.

insert into country values ('IN','India');

insert into country values ('US','United States of America');

Now in Eclipse IDE inside the project created, inside src/main/resources there is a file application.properties. In that file added the following code:

spring.application.name=orm-learn

# Spring and Hibernatelogging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

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

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=Dip\_5326

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

Created a package com.cognizant.orm\_learn.model and inside it created Country.java class.

**package** com.cognizant.orm\_learn.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;

**public** String getCode() {

**return** code;

}

**public** String getName() {

**return** name;

}

@Override

**public** String toString() {

**return** "Country [code="+ code+ ", name="+ name+ "]";

}

}

Created another package com.cognizant.orm\_learn.repository and inside it created CountryRepository.java

**package** com.cognizant.orm\_learn.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** com.cognizant.orm\_learn.model.Country;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country,String>{}

Created another package com.cognizant.orm\_learn.service and inside it created CountryService.java

**package** com.cognizant.orm\_learn.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.cognizant.orm\_learn.model.Country;

**import** com.cognizant.orm\_learn.repository.CountryRepository;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

**public** List<Country> getAllCountries(){

**return** countryRepository.findAll();

}

}

After all this changed the code in OrmLearnApplication.java

**package** com.cognizant.orm\_learn;

**import** java.util.List;

**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** com.cognizant.orm\_learn.model.Country;

**import** com.cognizant.orm\_learn.service.CountryService;

@SpringBootApplication

**public** **class** OrmLearnApplication {

**private** **static** **final** Logger ***LOGGER***= LoggerFactory.*getLogger*(OrmLearnApplication.**class**);

**private** **static** CountryService *countryService*;

**public** **static** **void** main(String[] args) {

ApplicationContext context=SpringApplication.*run*(OrmLearnApplication.**class**, args);

*countryService*= context.getBean(CountryService.**class**);

*testGetAllCountries*();

}

**private** **static** **void** testGetAllCountries() {

***LOGGER***.info("Start");

List<Country> countries=*countryService*.getAllCountries();

***LOGGER***.debug("countries={}",countries);

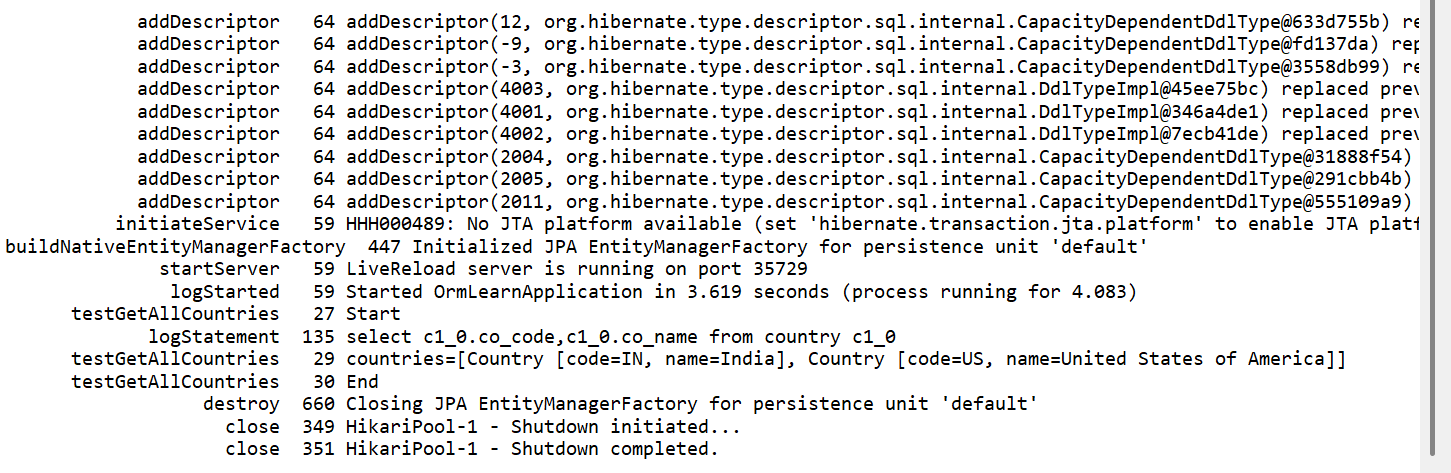
***LOGGER***.info("End");

}

}

The following output was seen in console.

Output:



To add Country code and name and find if it is present in the database or not, the following code was written.

In CountryService.java only two methods are added extra to the already done code as given previously.

**public** Country findCountryByCode(String code) {

**return** countryRepository.findById(code).orElse(**null**);

}

**public** **void** addCountry(Country country) {

countryRepository.save(country);

}

In OrmLearnApplication.java the following method was added

**private** **static** **void** testAddAndFindCountry() {

***LOGGER***.info("Start Add and Find");

//Add

Country country= **new**Country();

country.setCode("EN");

country.setName("England");

*countryService*.addCountry(country);

//Find

Country found=*countryService*.findCountryByCode("EN");

***LOGGER***.debug("Found Country: {}",found);

***LOGGER***.info("End Add and Find");

}

And after adding this fun it is called in main method to see the output produced by it.

**public** **static** **void** main(String[] args) {

ApplicationContext context=SpringApplication.*run*(OrmLearnApplication.**class**, args);

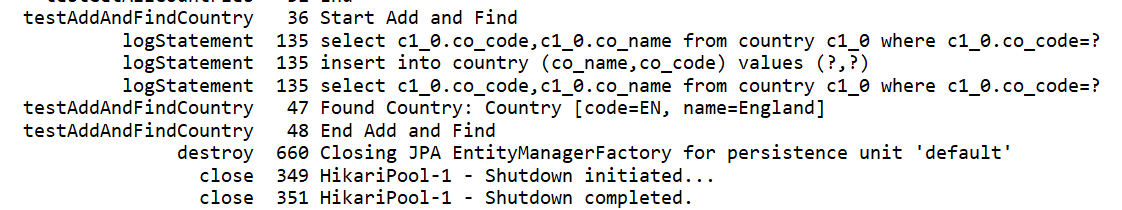
*countryService*= context.getBean(CountryService.**class**);

*testGetAllCountries*();

*testAddAndFindCountry*();

}

Output:



2) Difference between JPA, Hibernate and Spring Data JPA

JPA - JPA means Java Persistence API. It is a API specification for ORM (Object Relational Mapping). JPA is only the specification, no implementations are provided. To use JPA, Hibernate is required.

Hibernate - It is a popular JPA implementation which provides advanced features.

It is an ORM that allows us to map Java classes to database tables, manage sessions, and perform CRUD operations.

Spring Data JPA - Spring Data JPA is a Spring module built on top of JPA and Hibernate. With Spring Data JPA, there is no need to write queries. Boiler plate code is also not required to be written. Spring Data JPA lets us use methods like .save(), .findById(), .deleteById() by just extending JpaRepository.

Code comparison:

Hibernate:

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

Spring Data JPA:  
  
EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

EmployeeService.java

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

Hibernateprovides fine control, but requires boilerplate code for session and transaction management.

Spring Data JPA simplifies persistence by removing that boilerplate.

With @Transactional, Spring handles commits and rollbacks automatically.

Spring Data JPA improves developer productivity, readability, and maintainability.