**Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example**

OrmLearnApplication.java

package com.cognizant.orm\_learn;

import java.util.List;

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

import com.cognizant.orm\_learn.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;

*@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("Inside main()");

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");

}

}

Country.java

package com.cognizant.orm\_learn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.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 void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

*@Override*

public String toString() {

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

}

}

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> {

}

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 org.springframework.transaction.annotation.Transactional;

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

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

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**Country table creation**

Create a new table country with columns code and name.

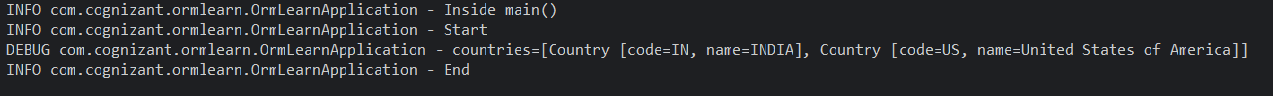
create table country(co\_code varchar(2) primary key, co\_name varchar(50));

Insert couple of records into the table

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

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

Output



**Differences between JPA, Hibernate and Spring Data JPA**

**JPA (Java Persistence API)**

### A standard method for object-relational mapping (ORM) in Java applications is defined by the Java EE (now Jakarta EE) specification known as JPA (Java Persistence API). It provides a collection of interfaces and annotations, including @Entity, @Id, and EntityManager, that specify how Java objects ought to be mapped to relational database tables. JPA only specifies the "what" of ORM, not the "how." As a result, it does not offer any implementation. To carry out the JPA-defined persistence operations, developers require an implementation..

### ****Hibernate****

One of the most widely used implementations of the JPA specification is Hibernate. It offers the underlying engine for persistence operations, such as updating, deleting, and saving database records. Apart from supporting every JPA feature, Hibernate provides a number of sophisticated features that are not included in the standard JPA specification, including batch processing, caching, lazy loading, and custom annotations (@Where, @Cascade, etc.). Hibernate can be used as a JPA provider or as a stand-alone ORM framework.

**Spring Data JPA**

Spring Data JPA is a Spring ecosystem component that takes advantage of JPA and its implementation (typically Hibernate) and provides simplified access to data in Spring applications. It encapsulates a lot of the boilerplate code involved in using JPA through repository interfaces and high-level query derivation features. Using Spring Data JPA, application developers can execute the basic database operations such as CRUD (Create, Read, Update, Delete) without having to write implementation code. It also facilitates the specification of custom queries via JPQL or native SQL, which makes it highly flexible and developer-friendly.

**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**  
EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

*@*Autowire

private EmployeeRepository employeeRepository;

*@*Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Implement Services for managing Country**

**CountryServiceApplication.java**

package com.example.countryservice;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class CountryserviceApplication {

public static void main(String[] args) {

SpringApplication.*run*(CountryserviceApplication.class, args);

}

}

**CountryController.java**

package com.example.countryservice.controller;

import com.example.countryservice.model.Country;

import com.example.countryservice.service.CountryService;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

*@Autowired*

private CountryService service;

*@GetMapping*("/{code}")

public Country getCountry(*@PathVariable* String code) {

return service.getCountryByCode(code);

}

*@PostMapping*

public Country addCountry(*@RequestBody* Country country) {

return service.addCountry(country);

}

*@PutMapping*

public Country updateCountry(*@RequestBody* Country country) {

return service.updateCountry(country);

}

*@DeleteMapping*("/{code}")

public String deleteCountry(*@PathVariable* String code) {

return service.deleteCountry(code) ? "Deleted" : "Not found";

}

*@GetMapping*("/search")

public List<Country> search(*@RequestParam* String name) {

return service.findByPartialName(name);

}

}

**Country.java**

package com.example.countryservice.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@Entity*

*@Table*(name = "country")

public class Country {

*@Id*

private String coCode;

private String coName;

public String getCoCode() { return coCode; }

public void setCoCode(String coCode) { this.coCode = coCode; }

public String getCoName() { return coName; }

public void setCoName(String coName) { this.coName = coName; }

}

**CountryRepository.java**

package com.example.countryservice.repository;

import com.example.countryservice.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByCoNameContainingIgnoreCase(String namePart);

}

**CountryService.java**

package com.example.countryservice.service;

import com.example.countryservice.model.Country;

import com.example.countryservice.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository repository;

public Country getCountryByCode(String code) {

return repository.findById(code).orElse(null);

}

public Country addCountry(Country country) {

return repository.save(country);

}

public Country updateCountry(Country country) {

if (repository.existsById(country.getCoCode())) {

return repository.save(country);

}

return null;

}

public boolean deleteCountry(String code) {

if (repository.existsById(code)) {

repository.deleteById(code);

return true;

}

return false;

}

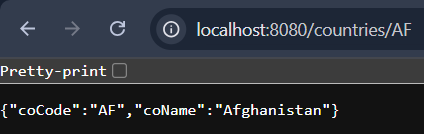
public List<Country> findByPartialName(String namePart) {

return repository.findByCoNameContainingIgnoreCase(namePart);

}

}

**Output**



**Find a country based on country code**

CountryNotFoundException.java

package com.cognizant.spring-learn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

CountryService.java

package com.cognizant.spring-learn.service;

import com.cognizant.spring-learn.model.Country;

import com.cognizant.spring-learn.repository.CountryRepository;

import com.cognizant.spring-learn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.Optional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country with code " + countryCode + " not found");

}

return result.get();

}

}

Country.java

package com.cognizant.spring-learn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Country {

*@Id*

private String code;

private String name;

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;

}

*@Override*

public String toString() {

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

}

}

CountryRepository.java

package com.cognizant.spring-learn.repository;

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

import com.cognizant.spring-learn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

OrmLearnApplication.java

package com.cognizant.spring-learn;

import com.cognizant.spring-learn.model.Country;

import com.cognizant.spring-learn.service.CountryService;

import com.cognizant.spring-learn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static CountryService *countryService*;

public static void main(String[] args) throws CountryNotFoundException {

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

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

*getAllCountriesTest*();

}

private static void getAllCountriesTest() throws CountryNotFoundException {

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

Country country = *countryService*.findCountryByCode("IN");

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

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

}

}

Output

Start

Country: Country [code=IN, name=India]

End

**Add a new country**

Create new method in CountryService.

*@*Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

Create testAddCountry() Method in OrmLearnApplication

private static void testAddCountry() throws CountryNotFoundException {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("JP");

newCountry.setName("Japan");

countryService.addCountry(newCountry);

Country retrievedCountry = countryService.findCountryByCode("JP");

LOGGER.debug("Retrieved Country: {}", retrievedCountry);

LOGGER.info("End");

}

Call testAddCountry() method

public static void main(String[] args) throws CountryNotFoundException {

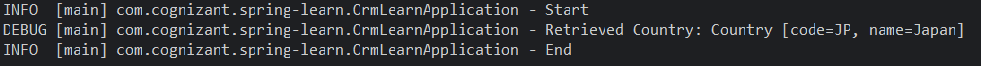
ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

testAddCountry();

}

Output



**Demonstrate implementation of Query methods feature of spring data JPA**

**Country.java**

package com.example.demo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Country {

*@Id*

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

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; }

}

**CountryRepository.java**

package com.example.demo.repository;

import com.example.demo.entity.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

Country findByName(String name);

List<Country> findByNameContaining(String keyword);

List<Country> findByCodeIn(List<String> codes);

}

**DemoRunner.java**

package com.example.demo;

import com.example.demo.entity.Country;

import com.example.demo.repository.CountryRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.Arrays;

*@Component*

public class DemoRunner implements CommandLineRunner {

*@Autowired*

private CountryRepository repo;

*@Override*

public void run(String... args) throws Exception {

repo.save(new Country("IN", "India"));

repo.save(new Country("US", "United States"));

repo.save(new Country("UK", "United Kingdom"));

System.***out***.println(repo.findByName("India"));

System.***out***.println(repo.findByNameContaining("United"));

System.***out***.println(repo.findByCodeIn(Arrays.*asList*("IN", "UK")));

}

}

**CountryController.java**

package com.example.demo.controller;

import com.example.demo.entity.Country;

import com.example.demo.repository.CountryRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

*@RestController*

public class CountryController {

*@Autowired*

private CountryRepository countryRepository;

*@GetMapping*("/countries")

public List<Country> getCountries() {

return countryRepository.findAll();

}

}

Output



**Demonstrate Implementation of O/R Mapping**

CountryRepository.java

package com.example.demo.repository;

import com.example.demo.entity.Country;

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

public interface CountryRepository extends JpaRepository<Country, String> {

}

Country.java

package com.example.demo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Country {

*@Id*

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

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; }

}

CountryController.java

package com.example.demo.controller;

import com.example.demo.entity.Country;

import com.example.demo.repository.CountryRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

*@RestController*

public class CountryController {

*@Autowired*

private CountryRepository repo;

*@GetMapping*("/countries")

public List<Country> getAllCountries() {

return repo.findAll();

}

}

DataLoader.java

package com.example.demo;

import com.example.demo.entity.Country;

import com.example.demo.repository.CountryRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

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

*@Component*

public class DataLoader implements CommandLineRunner {

*@Autowired*

private CountryRepository repo;

*@Override*

public void run(String... args) throws Exception {

repo.save(new Country("IN", "India"));

repo.save(new Country("US", "United States"));

repo.save(new Country("UK", "United Kingdom"));

}

}

Output



**Demonstrate writing Hibernate Query Language and Native Query**

**Country.java**

package com.example.springdataquery.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

*@Entity*

public class Country {

*@Id*

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

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; }

*@Override*

public String toString() {

return "Country{code='" + code + "', name='" + name + "'}";

}

}

**CountryRepository.java**

package com.example.springdataquery.repository;

import com.example.springdataquery.entity.Country;

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

import org.springframework.data.jpa.repository.Query;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

*@Query*("SELECT c FROM Country c WHERE c.name LIKE %:keyword%")

List<Country> findByNameUsingHQL(String keyword);

*@Query*(value = "SELECT \* FROM country WHERE name LIKE %:keyword%", nativeQuery = true)

List<Country> findByNameUsingNative(String keyword);

}

**DataLoader.java**

package com.example.springdataquery;

import com.example.springdataquery.entity.Country;

import com.example.springdataquery.repository.CountryRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.List;

*@Component*

public class DataLoader implements CommandLineRunner {

*@Autowired*

private CountryRepository repo;

*@Override*

public void run(String... args) throws Exception {

// Insert sample data

repo.save(new Country("IN", "India"));

repo.save(new Country("US", "United States"));

repo.save(new Country("UK", "United Kingdom"));

List<Country> hqlResult = repo.findByNameUsingHQL("United");

System.***out***.println("📘 HQL Result:");

hqlResult.forEach(System.***out***::println);

List<Country> nativeResult = repo.findByNameUsingNative("India");

System.***out***.println("💻 Native SQL Result:");

nativeResult.forEach(System.***out***::println);

}

}

**Output**

