SPRING DATA JPA

1. Spring Data JPA - Quick Example

Code:

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** CountryService *countryService*;

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

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

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

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

***LOGGER***.info("Inside main");

*testGetAllCountries*();

}

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

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

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

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

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

}

}

CountryService.java:

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

import java.util.List;

import jakarta.transaction.Transactional;

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;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

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**> {

}

Country.java:

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

**import** jakarta.persistence.\*;

@Entity

@Table(name = "country")

**public** **class** Country {

@Id

@Column(name = "co\_code")

**private** String code;

@Column(name = "co\_name")

**private** String name;

@Override

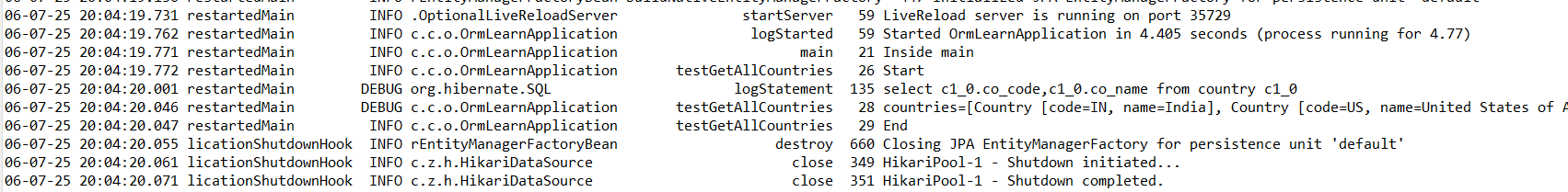
**public** String toString() {

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

}

}

Output:



2) Difference between JPA, Hibernate and Spring Data JPA

**Java Persistence API (JPA)**

* Standard specification (JSR 338) for object-relational mapping (ORM).
* Defines how Java objects are mapped to relational databases.
* Does not provide implementation; it’s just an interface/specification.
* Hibernate is one of the most popular implementations of JPA.

**Hibernate**

* A concrete ORM framework that implements the JPA specification.
* Handles all CRUD operations and database communication.
* Requires boilerplate code for session and transaction management.

**Spring Data JPA**

* Sits on top of JPA providers like Hibernate.
* Reduces boilerplate code using interfaces and annotations.
* Provides powerful features like method-based queries, pagination, and sorting.
* Handles transactions automatically using @Transactional.

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | Implementation of JPA | Abstraction over JPA implementations |
| Provides Implementation | No | Yes | No (Uses Hibernate or others) |
| Boilerplate Code | Absent | Required | Minimizes |
| Transaction Handling | Absent | Manual | Automatic via annotations |