**Week 3: Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example**

**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>3.5.3</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>20</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>

</project>

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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;

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) {

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

LOGGER.info("Inside main");

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");

}

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

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

import com.cognizant.orm\_learn.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

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.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 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 + "]";

}

}

**application.properties**

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

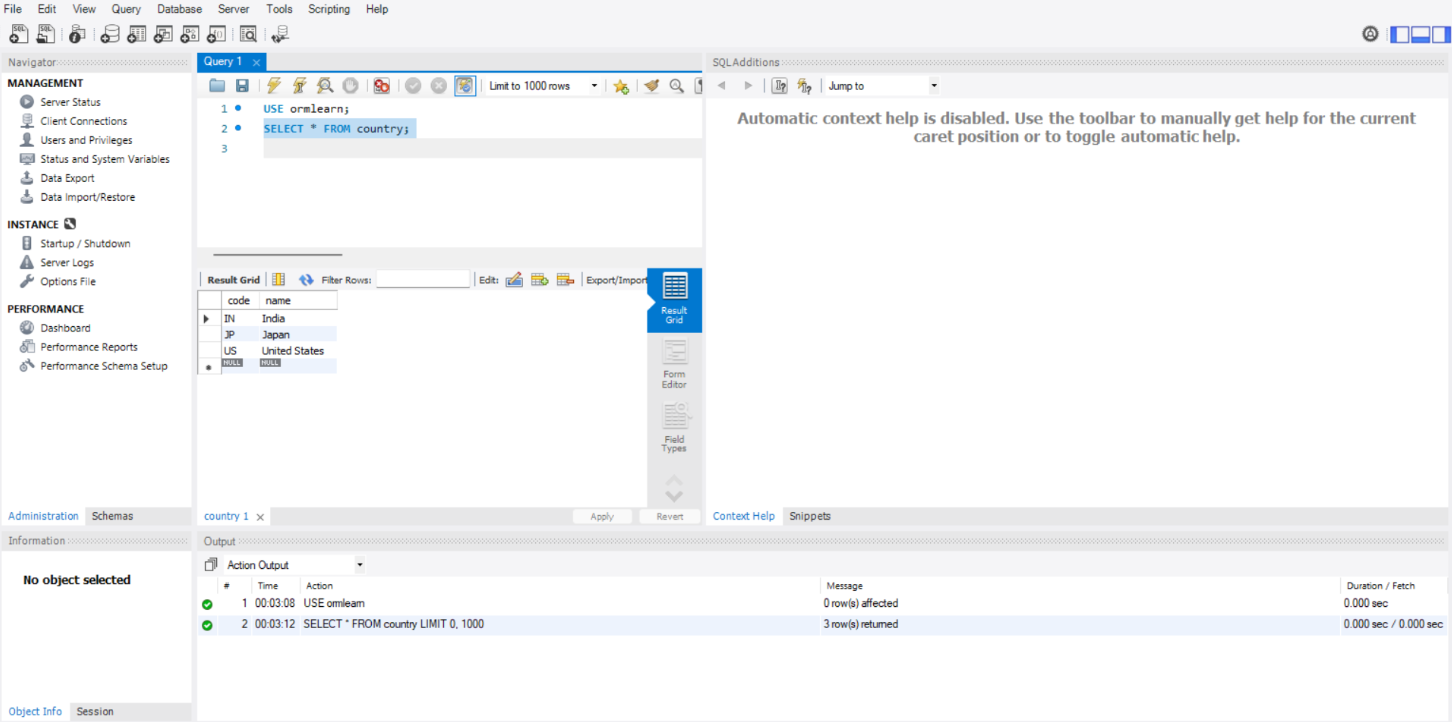
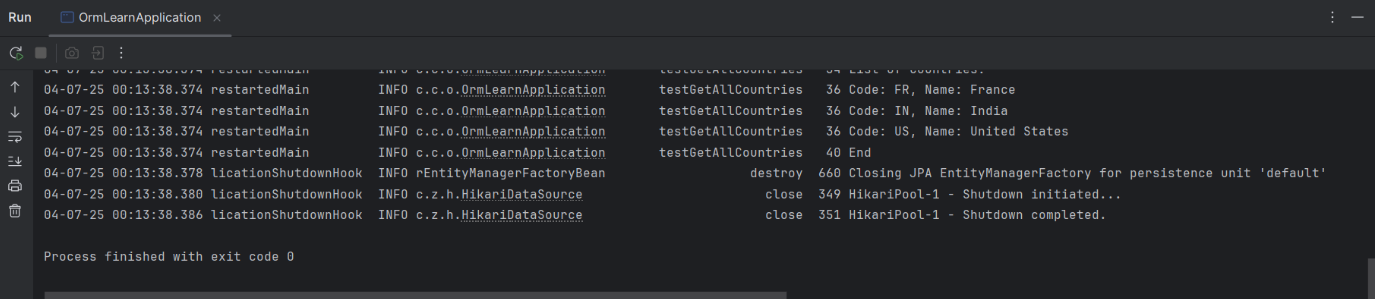
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=12345678

spring.jpa.hibernate.ddl-auto=validate



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

|  |  |  |  |
| --- | --- | --- | --- |
| Feature / Aspect | JPA (Java Persistence API) | Hibernate (ORM Framework) | Spring Data JPA (Spring abstraction) |
| Type | Specification / API (interface-based) | ORM Implementation Framework | Spring-based Data Access Abstraction |
| Defined By | Jakarta EE / Java EE | Red Hat (JBoss) | Spring Team (Pivotal) |
| Purpose | Standard for ORM in Java | Concrete implementation of ORM & JPA | Simplifies JPA usage with powerful repository support |
| Implementation | Requires a provider (e.g., Hibernate, EclipseLink) | Acts as a JPA provider (and adds enhancements) | Works on top of JPA (commonly uses Hibernate underneath) |
| Common Annotations | `@Entity`, `@Id`, `@Table`, `@OneToMany` | Same as JPA, plus specific ones like `@LazyCollection`, `@Fetch` | Uses JPA annotations + `@Repository`, interfaces like `JpaRepository` |
| EntityManager Usage | Must be manually handled | Offers `Session` API (extends EntityManager) | Hidden behind Spring interfaces (`CrudRepository`, `JpaRepository`) |
| Query Language | JPQL (Java Persistence Query Language) | JPQL + HQL (Hibernate Query Language) | JPQL + method names (`findByName`, etc.) + `@Query` annotation |
| Boilerplate Code | More manual coding for queries and entity operations | Less than raw JPA due to helper features | Very minimal — most tasks handled by Spring repositories |
| Ease of Use | Moderate | Easier than plain JPA | Easiest — declarative, convention-over-configuration |
| Advanced Features | Basic ORM support only | Caching, lazy/eager loading, batch fetches | Pagination, Sorting, Derived Queries, Specifications, Projections |
| Best For | When using different ORM providers or custom logic | When using full Hibernate power directly | When rapid development and minimal code is preferred |