Spring Data JPA with Spring Boot, Hibernate

Spring Data JPA - Quick Example

Code:

Country.java:

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

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 + "]";

}

}

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 jakarta.transaction.Transactional;

import jakarta.annotation.PostConstruct;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@PostConstruct

public void init() {

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

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

}

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

CountryRepository.java:

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

pom.xml:

<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

http://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>3.2.0</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<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-starter</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

ormLearnApplication.java:

package com.cognizant.ormlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

@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);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

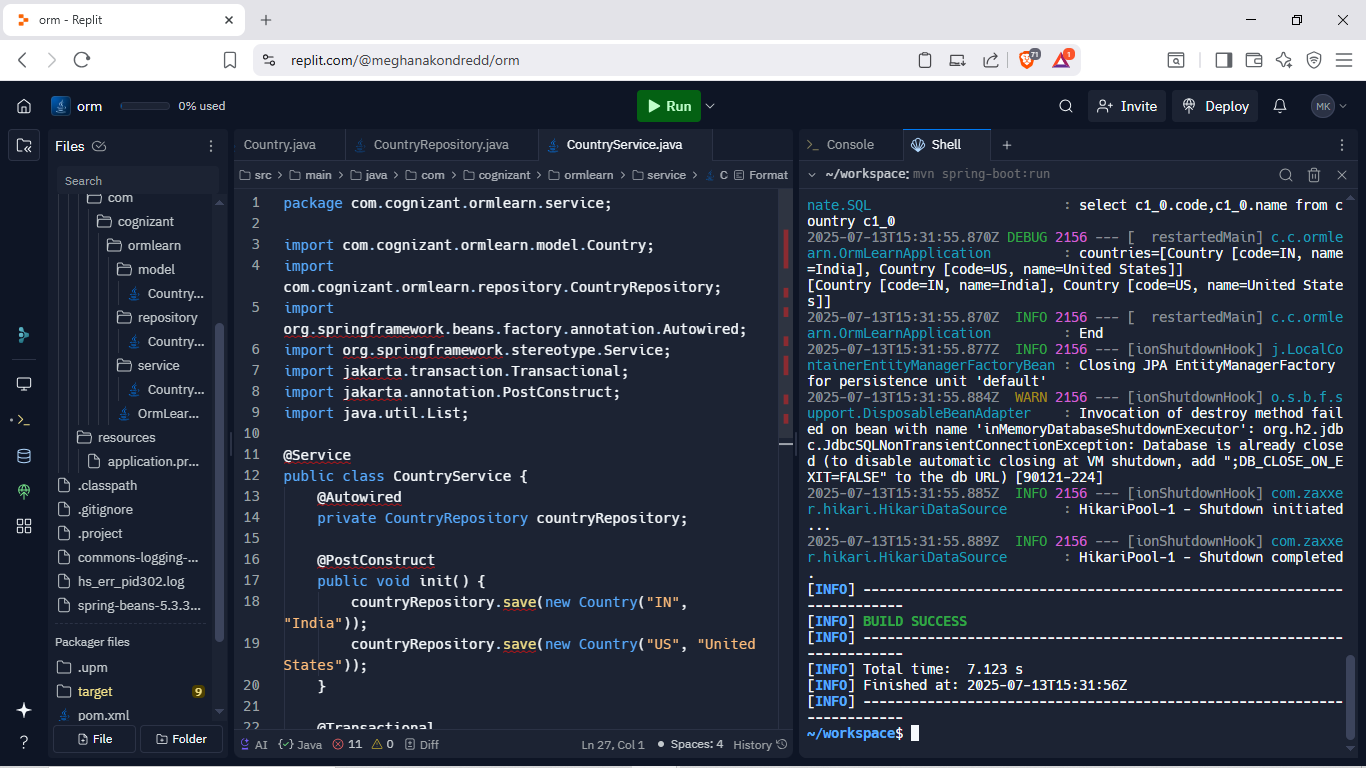
System.out.println(countries);

LOGGER.info("End");

}

}

Output:



Difference between Java Persistence API, Hibernate and Spring Data JPA:

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| Type | JPA is a specification for ORM in Java. | Hibernate is an ORM framework that implements JPA. | Spring Data JPA is a Spring module that builds on JPA and Hibernate. |
| Provided By | Provided by Oracle as part of Java EE/Jakarta EE. | Developed by Red Hat/JBoss. | Provided by the Spring Framework project. |
| Boilerplate Code | Requires writing a lot of standard persistence code. | Reduces some boilerplate through extra features. | Minimizes boilerplate by generating queries automatically. |
| Ease of Use | Less convenient—requires manual entity management. | Easier than JPA with additional tools. | Easiest—Spring handles most configurations and queries. |
| Query Writing | Uses JPQL; queries must be written manually. | Uses HQL (JPQL + extensions); native SQL also supported. | Can avoid writing queries using method names or annotations. |
| Caching Support | No caching support defined in spec. | Supports both first-level and second-level caching. | Uses Hibernate's caching when configured. |
| Integration | Needs manual setup with frameworks. | Can be used standalone or with Spring. | Seamlessly integrates with Spring Boot and other Spring components. |