**Spring Data JPA - Quick Example**   
**Codes:**

**OrmLearnApplication.java:**

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

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

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

**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.\*;

@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);

System.***out***.println(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="code")

**private** String code;

@Column(name="name")

**private** String name;

// getters and setters

**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;}

// toString()

@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 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();

}

}

**application.properties:**

spring.application.name=orm-learn

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

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 configuration

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=Lahari@4404

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**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>2.7.18</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>1.8</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>

**MySQL:**

CREATE DATABASE ormlearn;

use ormlearn;

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

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

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

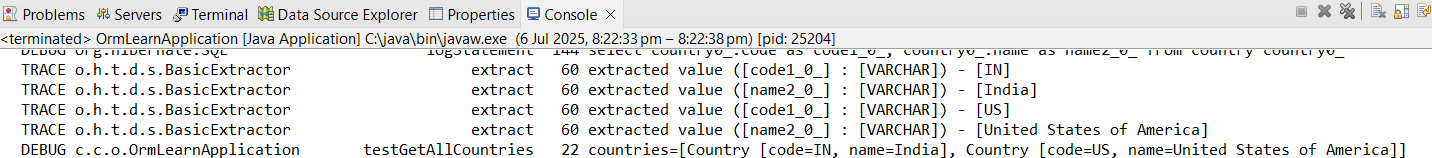
select \* from country;

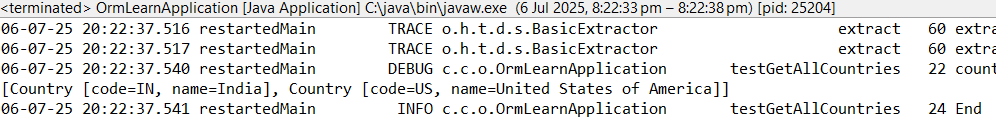
alter table country change co\_code code varchar(2);

alter table country change co\_name name varchar(50);

desc country;

**Output:**

****

****

**Explain the difference between Java Persistence API, Hibernate and Spring Data JPA**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Hibernate** | **JPA** | **Spring Data JPA** |
| **Definition** | ORM (Object Relational Mapping) framework | Specification for ORM in java | Simplified abstraction over JPA |
| **Implementation** | Provides its own implementation | Just a specification, requires an implementation | Builds on JPA, provides additional features |
| **Persistence API** | Hibernate provides its own API | Defines a set of standard APIs for ORM | Builds JPA APIs, adds more functionality |
| **Database Support** | Supports various databases through dialects | Depends on the JPA implementation used | Depends on the JPA implementation used |
| **Transaction Management** | Provides its own transaction management | Depends on the JPA implementation used | Depends on the JPA implementation used |
| **Query Language** | Hibernate Query Language (HQL) | JPQL (Java Persistence Query Language) | JPQL (Java Persistence Query Language) |
| **Caching** | Provides first-level and second-level caching | Depends on the JPA implementation used | Depends on the JPA implementation used |
| **Configuration** | XML, annotations, or Java-based configuration | XML, annotations, or Java-based configuration | XML, annotations, or Java-based configuration |
| **Integration** | Can be used independently or with JPA | Works with any JPA-compliant implementation | Works with any JPA-compliant implementation |
| **Spring Integration** | Can be integrated with Spring | Can be integrated with Spring | Part of the Spring Data family, integrates with Spring |

**Java Persistence API (JPA)** is a specification provided by Java to define how objects in Java are mapped to relational database tables. It only provides guidelines and interfaces, but no actual implementation. Developers can use any JPA-compliant implementation like Hibernate or EclipseLink to persist data.

**Hibernate** is a popular implementation in JPA. It is an Object-Relational Mapping (ORM) tool that actually performs the database operations based on JPA rules. Hibernate adds many advanced features beyond the JPA standard, such as caching, lazy loading, and its own query language (HQL).

**Spring Data JPA** is a framework built on top of JPA and Hibernate that simplifies database access in Spring applications. It helps remove boilerplate code by providing ready-to-use repository interfaces, allowing developers to perform common operations like save, update, delete, and query with minimal code.