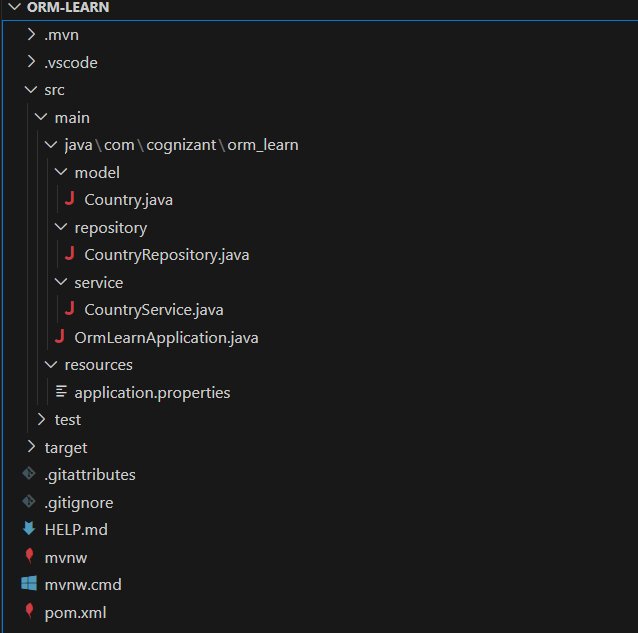
**Exercise 1: Spring Data JPA - Quick Example** 

**Solution:**

Project Structure-



Country.java –

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Column;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

    @Id

    @Column(name = "code")

    private String code;

    @Column(name = "name")

    private String name;

}

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> {

}

OrmLearningApplication.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) {

        try {

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

            System.out.println("!!!!! Spring Boot context started !!!!!");

            countryService = context.getBean(CountryService.class);

            System.out.println("!!!!! Got CountryService bean !!!!!");

            testGetAllCountries();

        } catch (Exception e) {

            System.out.println("Caught exception: " + e.getClass().getName() + " - " + e.getMessage());

            e.printStackTrace();

        }

    }

    private static void testGetAllCountries() {

        System.out.println("Runnin ------------------------------------------------!!");

        LOGGER.info("Start");

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

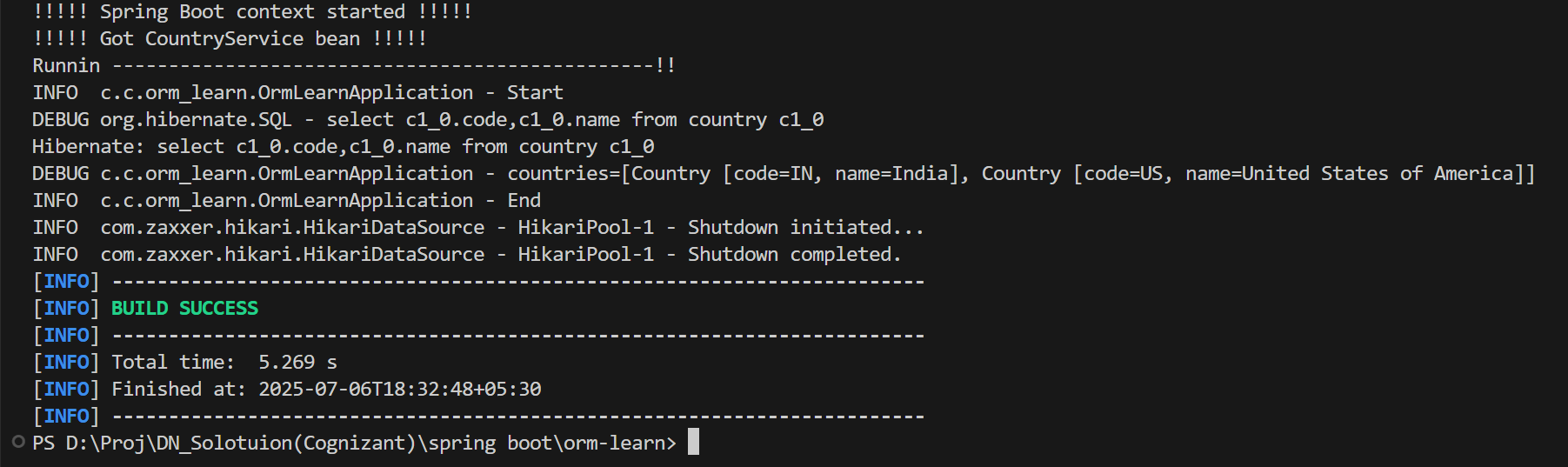
        LOGGER.debug("countries={}", countries);

        LOGGER.info("End");

    }

}

**Output:**



**Exercise 4 -Difference between JPA, Hibernate and Spring Data JPA**

**Solution:**

**Java Persistence API (JPA):**

* It is a specification (or standard), which sets some rules and structure so that each developer doesn't have to rely on a specific tool.
* It’s like a Blueprint.
* It is not an implementation, so to use it we have to use other tools like Hibernate.
* And most importantly, it is portable — so we can switch from tool to tool as per our need (provides flexibility).

**Hibernate**:

* It is an ORM tool that implements the JPA specifications and also provides extra features.
* With its help, we can map Java objects to tables, so we don’t have to write SQL directly in our Java files.

**Spring Data JPA**:

* It is an abstraction layer over JPA.
* It is a Spring project that builds on top of JPA.
* It is better, as it provides automation (writing queries, etc.).
* It provides many interfaces which reduce repetitive code.
* And behind the scenes, it uses JPA with Hibernate (or any other) provider.

**Code Difference:**

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);

    }