**SPRING DATA JPA WITH SPRING BOOT, HIBERNATE**

**Hands on 1**

**Spring Data JPA - Quick Example**

**CODE:**

**application.properties.xml**

# Logging config

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

# Database connection

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=toor

# Hibernate config

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

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 + "]";

}

}

**CountryRepository.java**

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.ormlearn.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.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);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

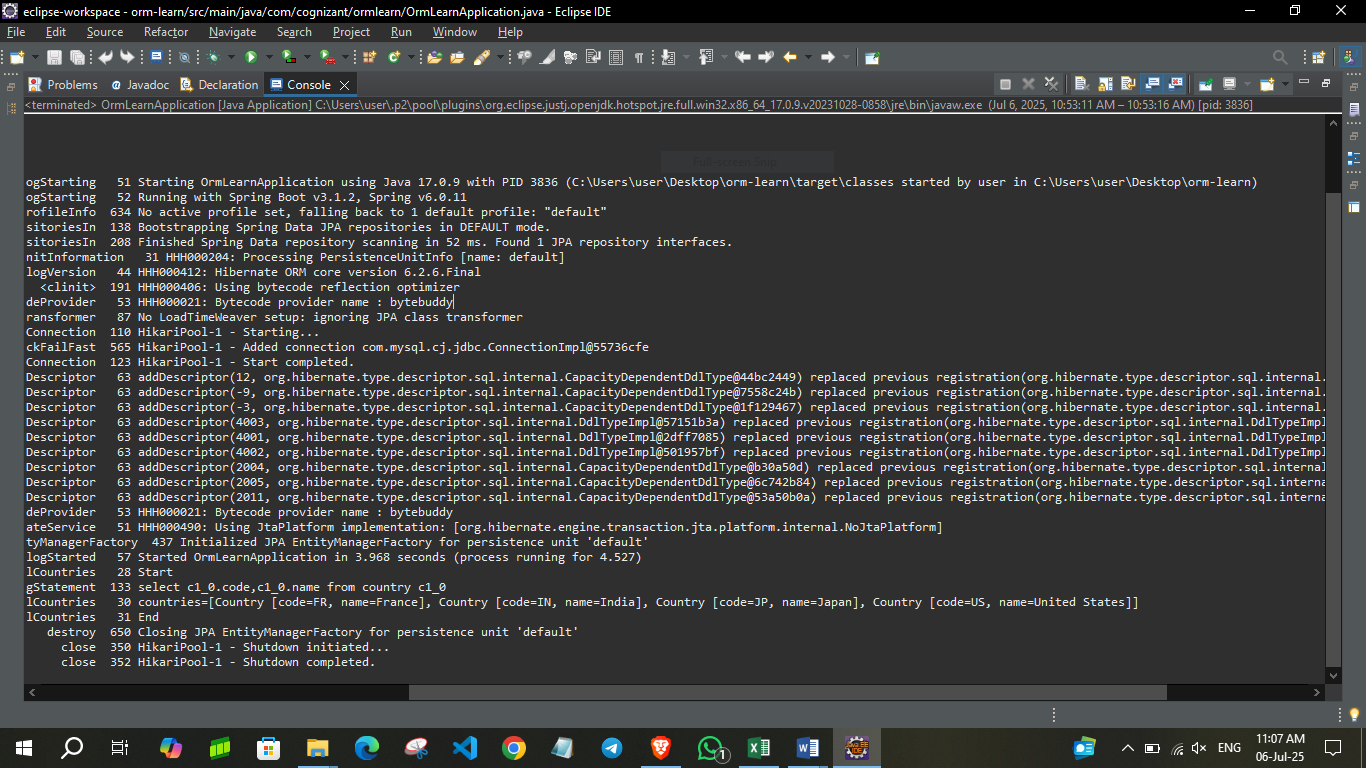
LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

**OUTPUT:**



**Hands on 4**

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

**CODE:**

**Hibernate**

   /\* Method to CREATE an employee in the database \*/

   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);

  }

**Difference:**

* The Hibernate code involves manual handling of sessions and transactions using **Session**, **Transaction**, and **try-catch** blocks, making the code more verbose and error-prone.
* In contrast, Spring Data JPA simplifies this by providing built-in repository interfaces like **JpaRepository**, which automatically handles CRUD operations and transaction management.
* This eliminates boilerplate code, improves readability, and speeds up development. While Hibernate requires explicit persistence logic, Spring Data JPA leverages annotations and dependency injection to achieve the same functionality with minimal effort, making it more convenient for modern Spring Boot applications**.**
* Spring Data JPA allows you to define custom queries just by naming methods in a specific way (e.g., **findByName**), whereas in Hibernate, you typically need to write HQL (Hibernate Query Language) or SQL queries manually.
* Spring Data JPA is tightly integrated with the Spring Framework, which provides out-of-the-box support for dependency injection, transaction management, and REST controllers, making it easier to build full-stack applications compared to plain Hibernate.
* Spring Data JPA eliminates boilerplate code by providing built-in repository interfaces, while Hibernate requires manual handling of sessions, transactions, and CRUD operations.
* Spring Data JPA provides built-in support for pagination and sorting through **Pageable** and **Sort** interfaces, whereas in Hibernate, developers need to manually implement pagination logic.
* Applications using Spring Data JPA are easier to test, thanks to Spring's testing utilities and embedded database support, whereas setting up and testing with plain Hibernate typically involves more configuration.