**Hands on 1**

**Spring Data JPA - Quick Example**   
**Software Pre-requisites**

* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

**Topic**: Spring Data JPA – Quick Example

**Description**:This Spring Boot application demonstrates a simple setup of Spring Data JPA and Hibernate using MySQL. The project includes the creation of a Country entity, the use of JpaRepository for data access, and a service to fetch data from the database.

**Country.java**

**package** com.cognizant.ormlearn.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 = "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 + "]";

}

}

**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();

}

**public** **void** addCountry(Country country) {

countryRepository.save(country);

}

}

**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** **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**);

***LOGGER***.info("Inside main");

*testInsertCountries*(); // Insert sample data

*testGetAllCountries*(); // Fetch all data

}

**private** **static** **void** testInsertCountries() {

***LOGGER***.info("Start insert");

*countryService*.addCountry(**new** Country("IN", "India"));

*countryService*.addCountry(**new** Country("US", "United States"));

*countryService*.addCountry(**new** Country("JP", "Japan"));

***LOGGER***.info("End insert");

}

**private** **static** **void** testGetAllCountries() {

***LOGGER***.info("Start fetch");

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

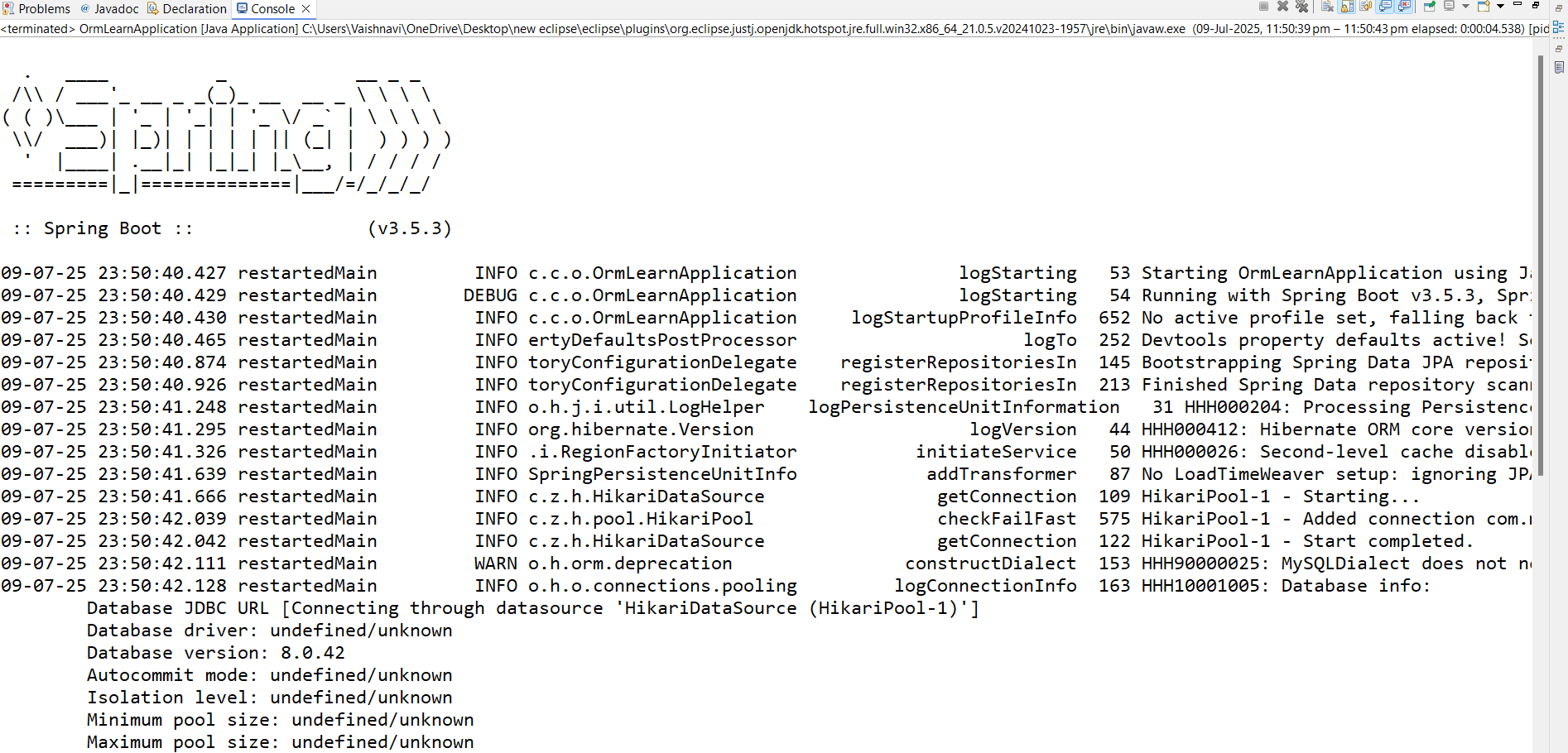
***LOGGER***.debug("countries={}", countries);

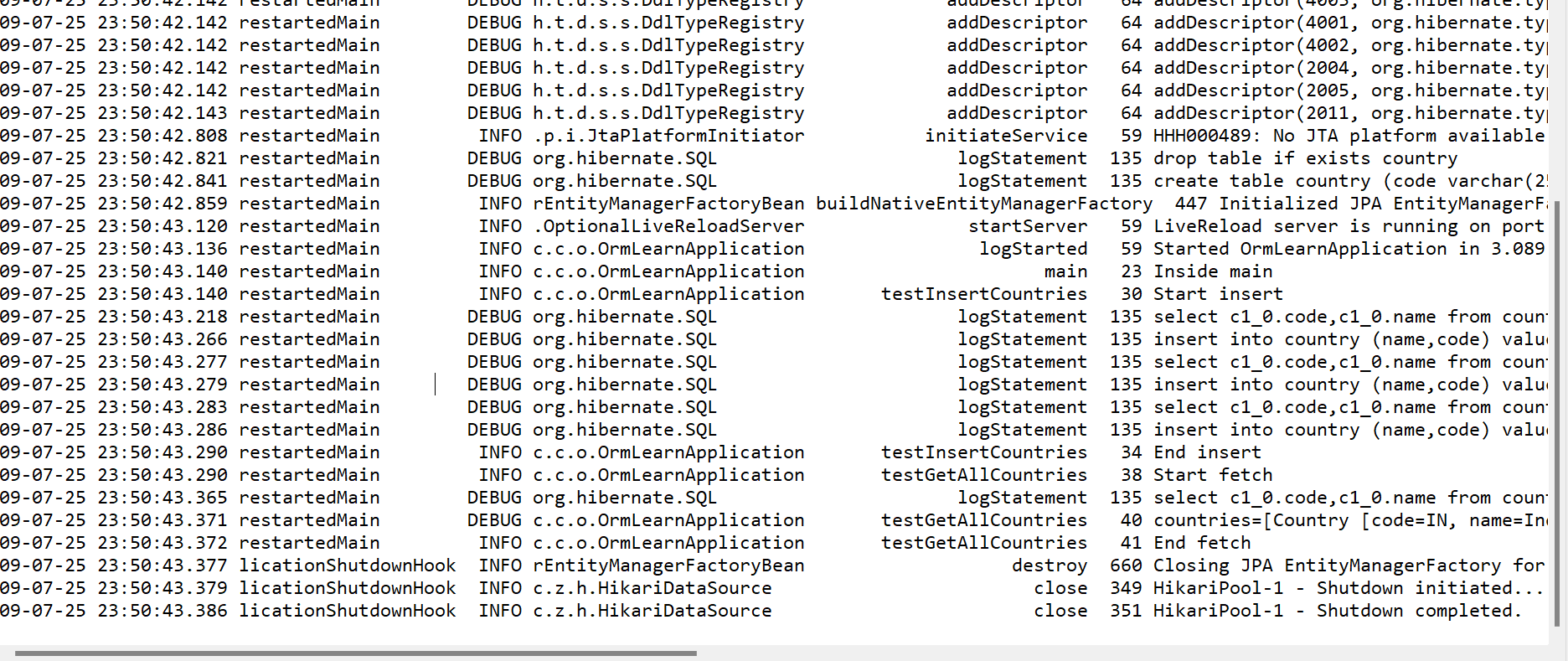
***LOGGER***.info("End fetch");

}

}

**OUTPUT:**

****

****

**NOTE:**

* MySQL version: 8.0
* Spring Boot version: 3.5.3
* Database schema: ormlearn
* Table: country(code, name)

**Hands on 4**

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

**Java Persistence API (JPA)**

* JPA is a specification (JSR 338) for managing relational data in Java applications.
* It provides annotations and APIs to define how objects are persisted.
* No implementation itself – requires a provider like Hibernate.
* Example providers: Hibernate, EclipseLink, OpenJPA.

**Hibernate**

* Concrete implementation of JPA.
* Popular ORM (Object Relational Mapping) framework.
* Provides extra features beyond JPA such as caching, performance tuning, etc.

**Spring Data JPA**

* Abstraction over JPA & Hibernate.
* Helps reduce boilerplate code by providing ready-to-use repository interfaces.
* Manages transactions, CRUD operations, and queries easily.
* Uses JpaRepository, CrudRepository, etc.

**Code Comparison:**

**Hibernate Example**

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 Example**

**EmployeeRepository.java**

Public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Conclusion**:

* JPA defines *what to do* (specification),
* Hibernate shows *how to do it* (implementation),
* Spring Data JPA makes it *easy to do* (abstraction + automation).