**Spring Data JPA - Quick Example**

**Application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1

spring.datasource.username=root

spring.datasource.password=Gayatri@81$

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

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

server.port=8086

**ormLearnApplication.java:**

**package** com.cognizant.ormlearn;

**import** com.cognizant.ormlearn.model.Country;

**import** com.cognizant.ormlearn.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) {

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

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

*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");

}

}

**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 = "co\_code", nullable = **false**, length = 2)

**private** String code;

@Column(name = "co\_name", nullable = **false**, length = 50)

**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** org.springframework.stereotype.Repository;

**import** com.cognizant.ormlearn.model.Country;

@Repository

**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** org.springframework.transaction.annotation.Transactional;

**import** com.cognizant.ormlearn.model.Country;

**import** com.cognizant.ormlearn.repository.CountryRepository;

@Service

**public** **class** CountryService {

**private** **final** CountryRepository countryRepository;

@Autowired

**public** CountryService(CountryRepository countryRepository) {

**this**.countryRepository = countryRepository;

}

@Transactional(readOnly = **true**)

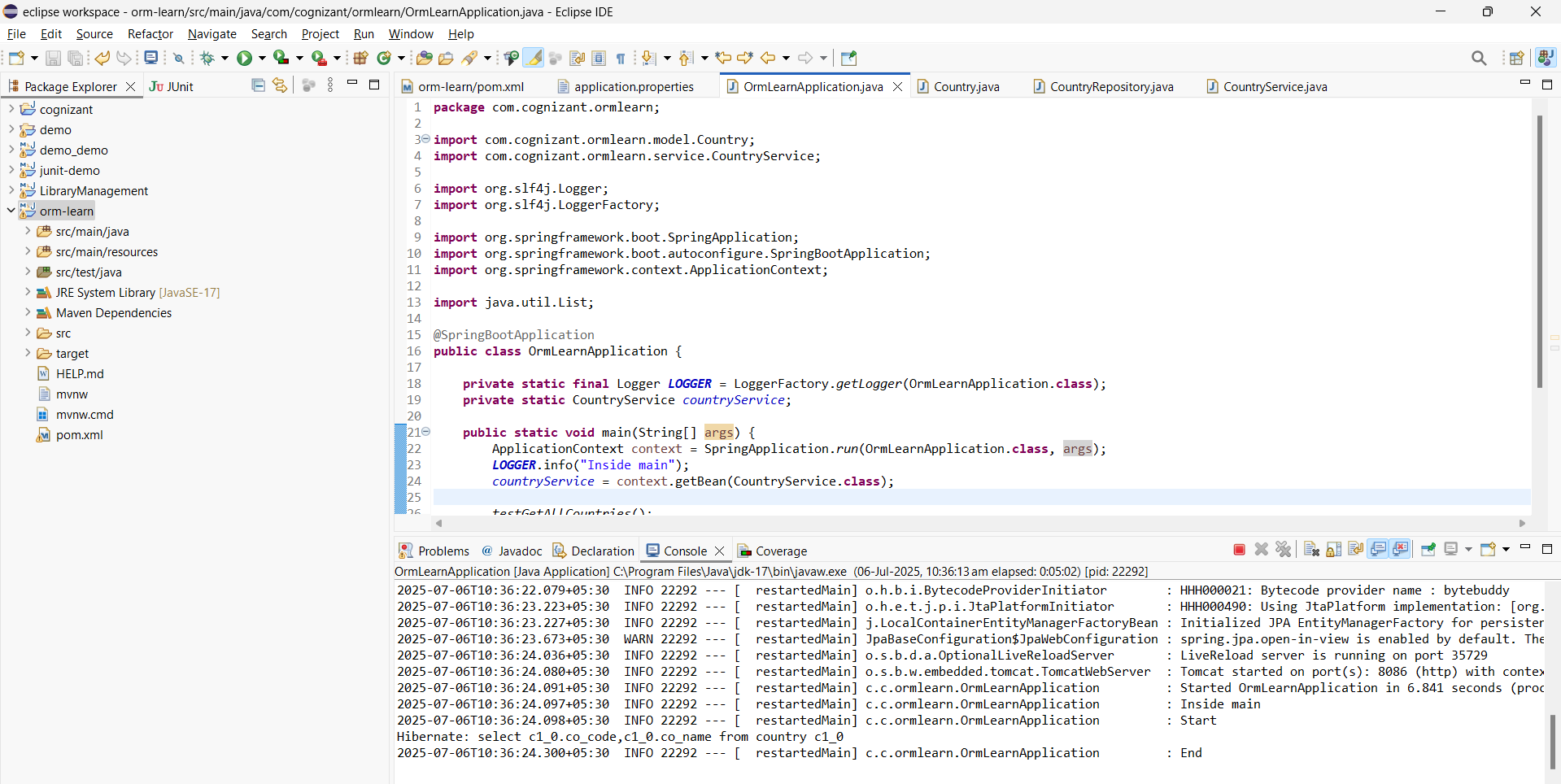
**public** List<Country> getAllCountries() {

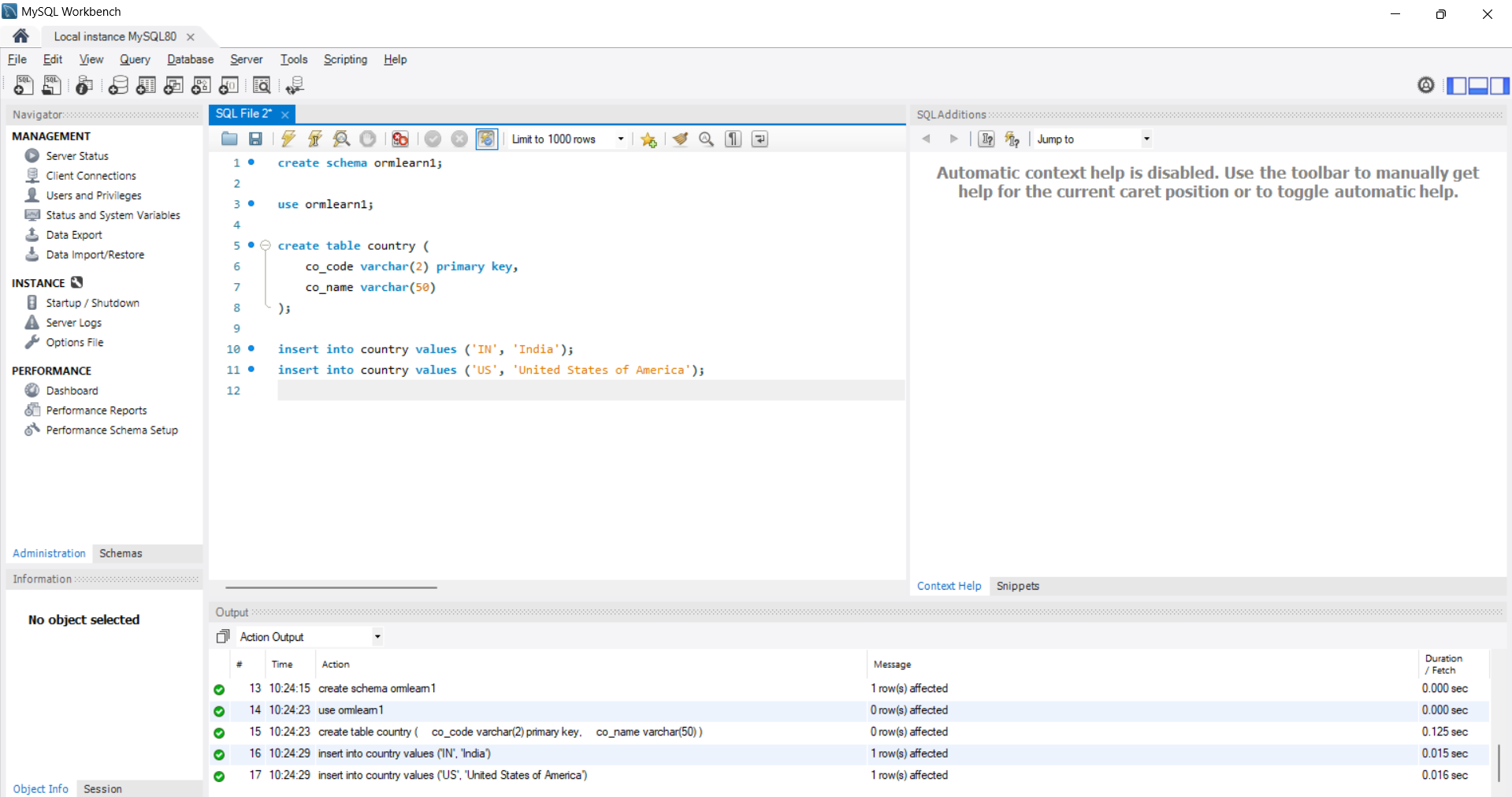
**return** countryRepository.findAll();

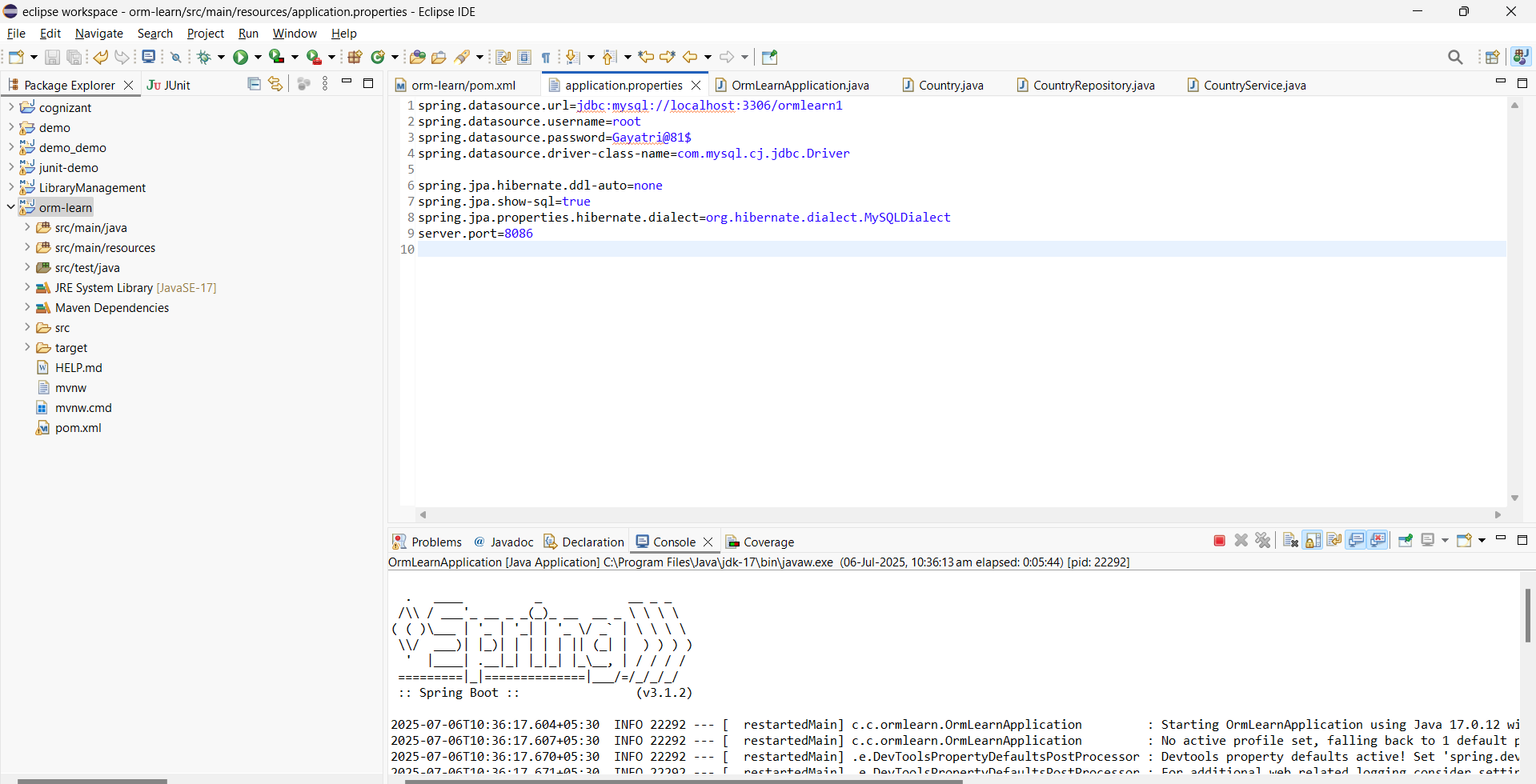
}

}

**OUTPUT:**







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

1. JPA (Java Persistence API) – The Specification

* JPA is just a set of guidelines (like an interface) for how Java should interact with databases.
* It provides annotations like @Entity, @Id, and defines how entities should be mapped.
* Doesn’t do anything by itself — you need a provider (like Hibernate) to make it work.

2. Hibernate – The Implementation

* Hibernate is the most popular implementation of JPA — it actually handles database operations.
* It provides extra features beyond JPA, like caching, lazy loading, and HQL (Hibernate Query Language).
* You can use it directly, or let it work behind the scenes when you use JPA or Spring Data JPA.

3. Spring Data JPA – The Simplifier

* Spring Data JPA is a framework built on top of JPA and Hibernate to reduce boilerplate code.
* It provides interfaces like JpaRepository so you don’t have to write your own DAO logic.
* With method naming (e.g., findByName()), it can auto-generate queries, saving tons of time.