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

**CODE:**

**Country.java**

package com.cognizant.ormlearn;

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")

private String code;

@Column(name="co\_name")

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;

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

import org.springframework.stereotype.Repository;

*@Repository*

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**  
package com.cognizant.ormlearn;

import java.util.List;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public void testMethod() {

System.***out***.println("CountryService method called");

}  
}  
**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;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static com.cognizant.ormlearn.CountryService countryService;

public static void main(String[] args) {

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

LOGGER.info("Inside main");

countryService = context.getBean(com.cognizant.ormlearn.CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

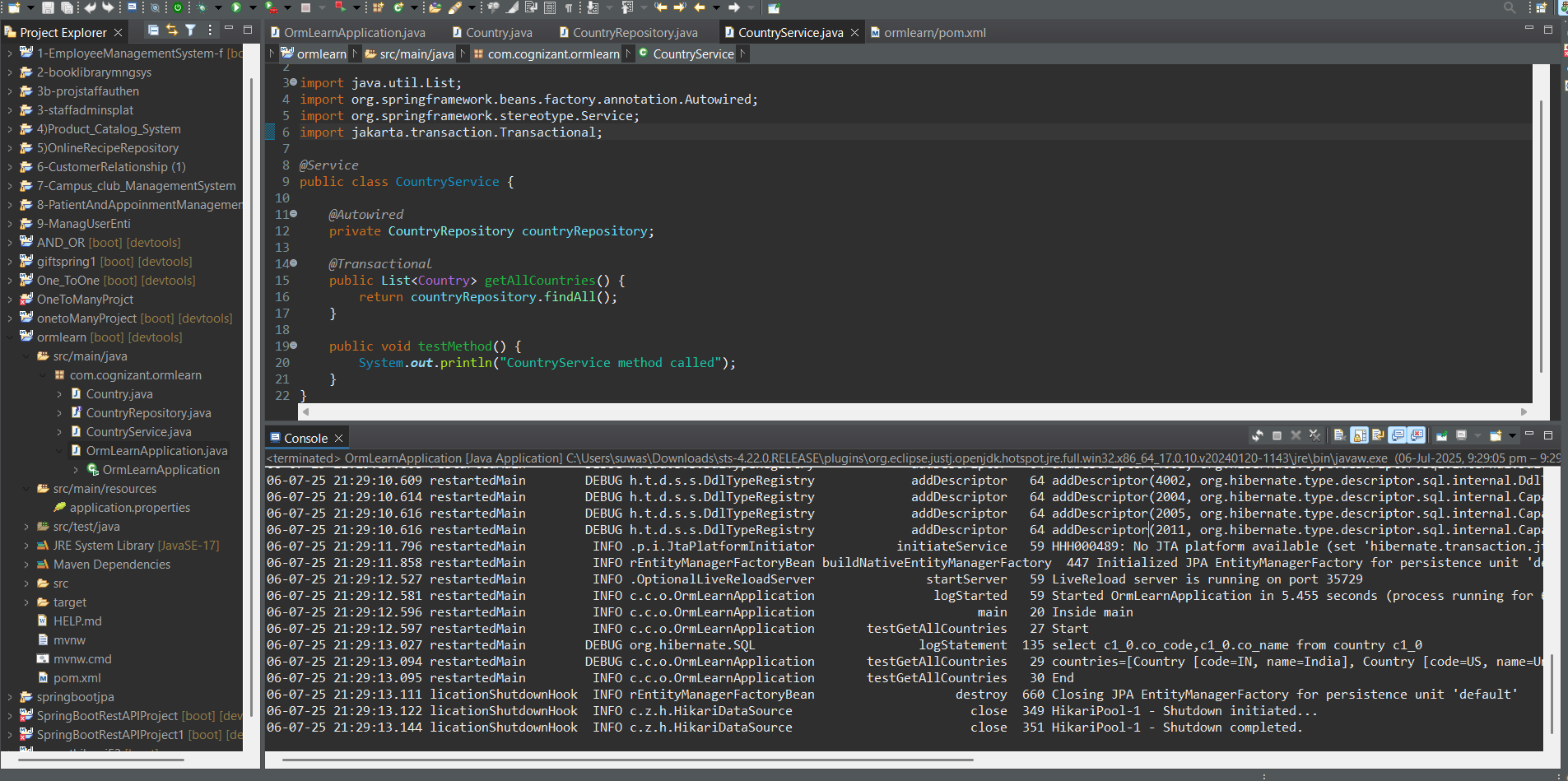
LOGGER.debug("countries={}", countries);

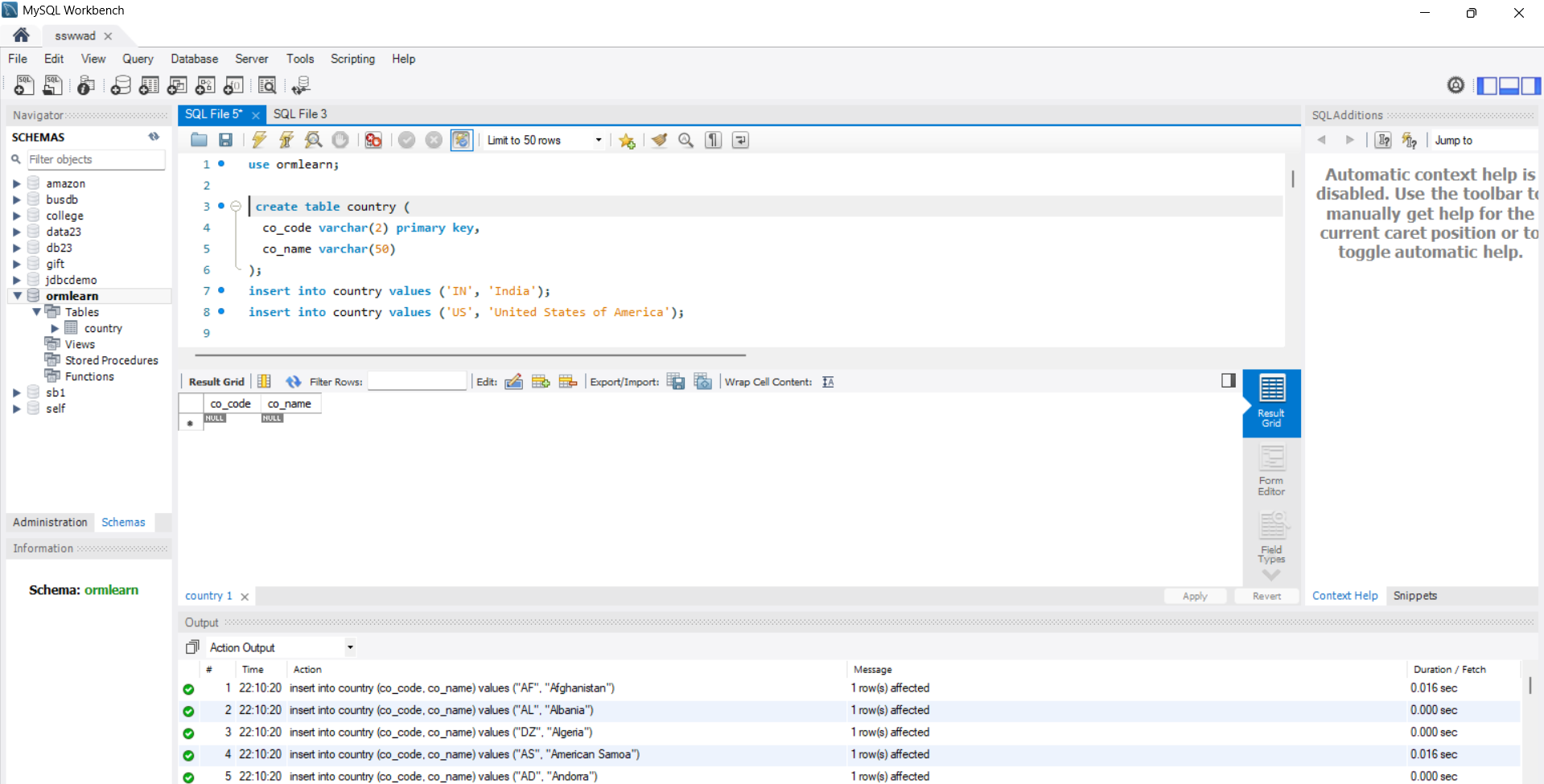
LOGGER.info("End");

}

}

**OUTPUT:**





**Exercise 5: Implement services for managing Country**

**Country.java**  
package com.cognizant.ormlearn;

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")

private String code;

@Column(name="co\_name")

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 + "]";

}

}

**CountryNotFoundException.java**  
package com.cognizant.ormlearn;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**CountryService.java**  
package com.cognizant.ormlearn;

import java.util.List;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Transactional*

public Country findCountryByCode(String code) {

return countryRepository.findByCode(code);

}

*@Transactional*

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByNameContaining(partialName);

}

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void updateCountry(Country country) {

countryRepository.save(country);

}

*@Transactional*

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

public void testMethod() {

System.***out***.println("CountryService method called");

}

}  
**CountryRepository.java**package com.cognizant.ormlearn;

import java.util.List;

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

import org.springframework.stereotype.Repository;

*@Repository*

public interface CountryRepository extends JpaRepository<Country, String> {

Country findByCode(String code);

List<Country> findByNameContaining(String partialName);

}  
**OrnLearnApplication.java**  
package com.cognizant.ormlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class OrmLearnApplication {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(OrmLearnApplication.class);

private static CountryService *countryService*;

public static void main(String[] args) throws CountryNotFoundException {

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

*countryService* = context.getBean(CountryService.class);

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

*testGetAllCountries*();

*testFindCountryByCode*();

*testAddCountry*();

*testUpdateCountry*();

*testDeleteCountry*();

*testFindCountriesByPartialName*();

}

private static void testGetAllCountries() {

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

***LOGGER***.debug("Countries={}", *countryService*.getAllCountries());

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

}

private static void testFindCountryByCode() throws CountryNotFoundException {

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

***LOGGER***.debug("Country={}", *countryService*.findCountryByCode("IN"));

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

}

private static void testAddCountry() {

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

Country newCountry = new Country();

newCountry.setCode("XY");

newCountry.setName("Testland");

*countryService*.addCountry(newCountry);

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

}

private static void testUpdateCountry() throws CountryNotFoundException {

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

Country updateCountry = new Country();

updateCountry.setCode("XY");

updateCountry.setName("Testlandia");

*countryService*.updateCountry(updateCountry);

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

}

private static void testDeleteCountry() throws CountryNotFoundException {

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

*countryService*.deleteCountry("XY");

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

}

private static void testFindCountriesByPartialName() {

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

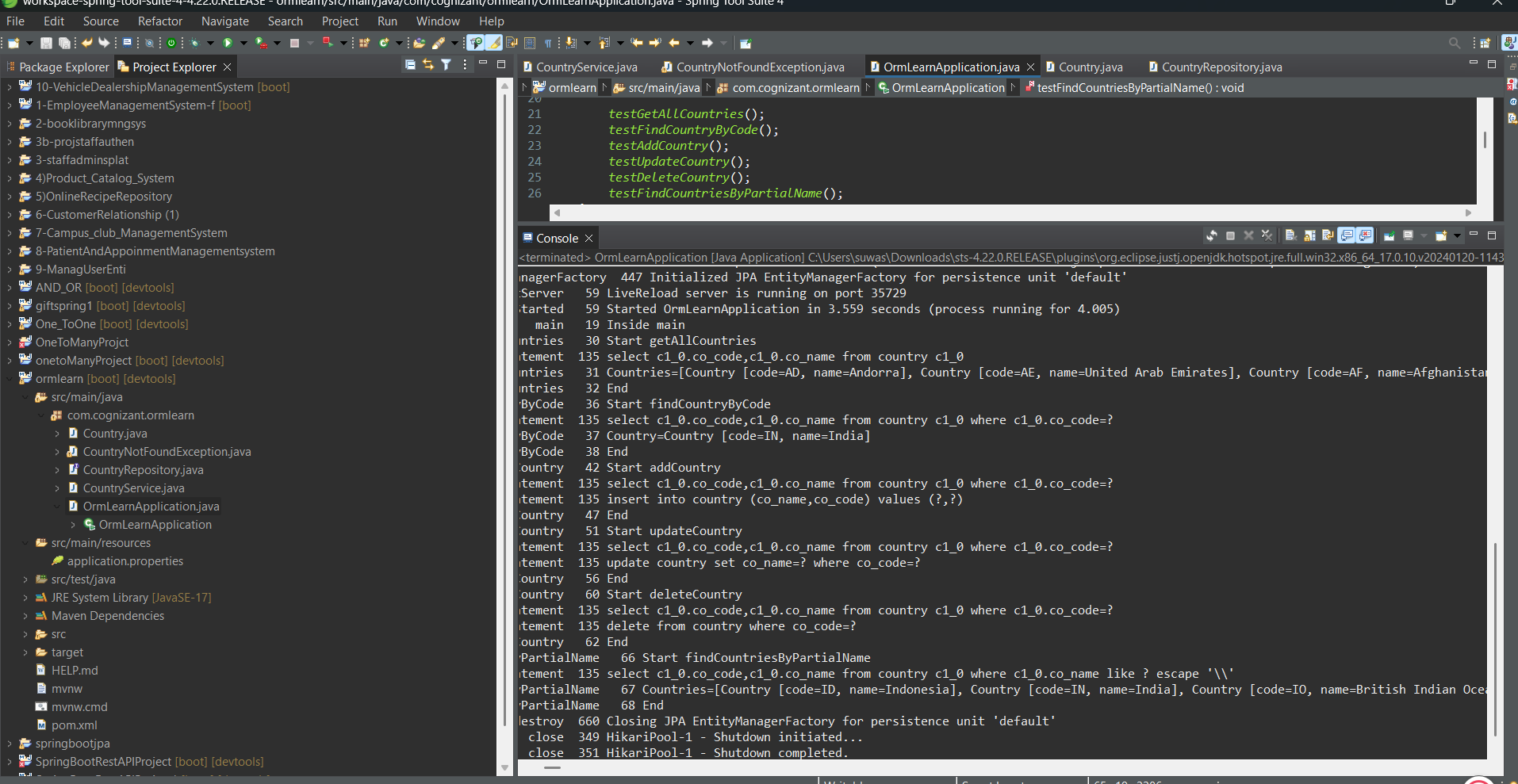
***LOGGER***.debug("Countries={}", *countryService*.findCountriesByPartialName("Ind"));

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

}

}

**OUTPUT:**



**Difference between Spring Data JPA and Hibernate**

What Is Java Persistence API?

The Java Persistence API provides a specification for persisting, reading, and managing data from your Java object to relational tables in the database.

What Is Hibernate Framework?

Hibernate is an object-relational mapping solution for Java environments. Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables.

Hibernate provides a reference implementation of the Java Persistence API that makes it a great choice as an ORM tool with the benefits of loose coupling.

Example: Below diagram shows an Object Relational Mapping between the Student Java class and the students table in the database.

What Is Spring Data JPA?

Spring Data is a part of the Spring Framework. The goal of Spring Data repository abstraction is to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores.

Spring Data JPA is not a JPA provider. It is a library/framework that adds an extra layer of abstraction on the top of our JPA provider (like Hibernate).

What Is the Difference Between Hibernate and Spring Data JPA?

Hibernate is a JPA implementation, while Spring Data JPA is a JPA Data Access Abstraction.

Spring Data offers a solution to GenericDao custom implementations. It can also generate JPA queries on your behalf through method name conventions.

With Spring Data, you may use Hibernate, Eclipse Link, or any other JPA provider. A very interesting benefit is that you can control transaction boundaries declaratively using the @Transactional annotation.

Spring Data JPA is not an implementation or JPA provider, it's just an abstraction used to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores.

Hibernate provides a reference implementation of the Java Persistence API that makes it a great choice as an ORM tool with the benefits of loose coupling.