**Spring Data JPA - Quick Example**

**// File: src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java**

package com.cognizant.ormlearn;

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;

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");

}

}

**// File: src/main/java/com/cognizant/ormlearn/model/Country.java**

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "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 + "]";

}

}

**// File: src/main/java/com/cognizant/ormlearn/repository/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> {

}

**// File: src/main/java/com/cognizant/ormlearn/service/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 {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**// File: src/main/resources/application.properties**

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

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

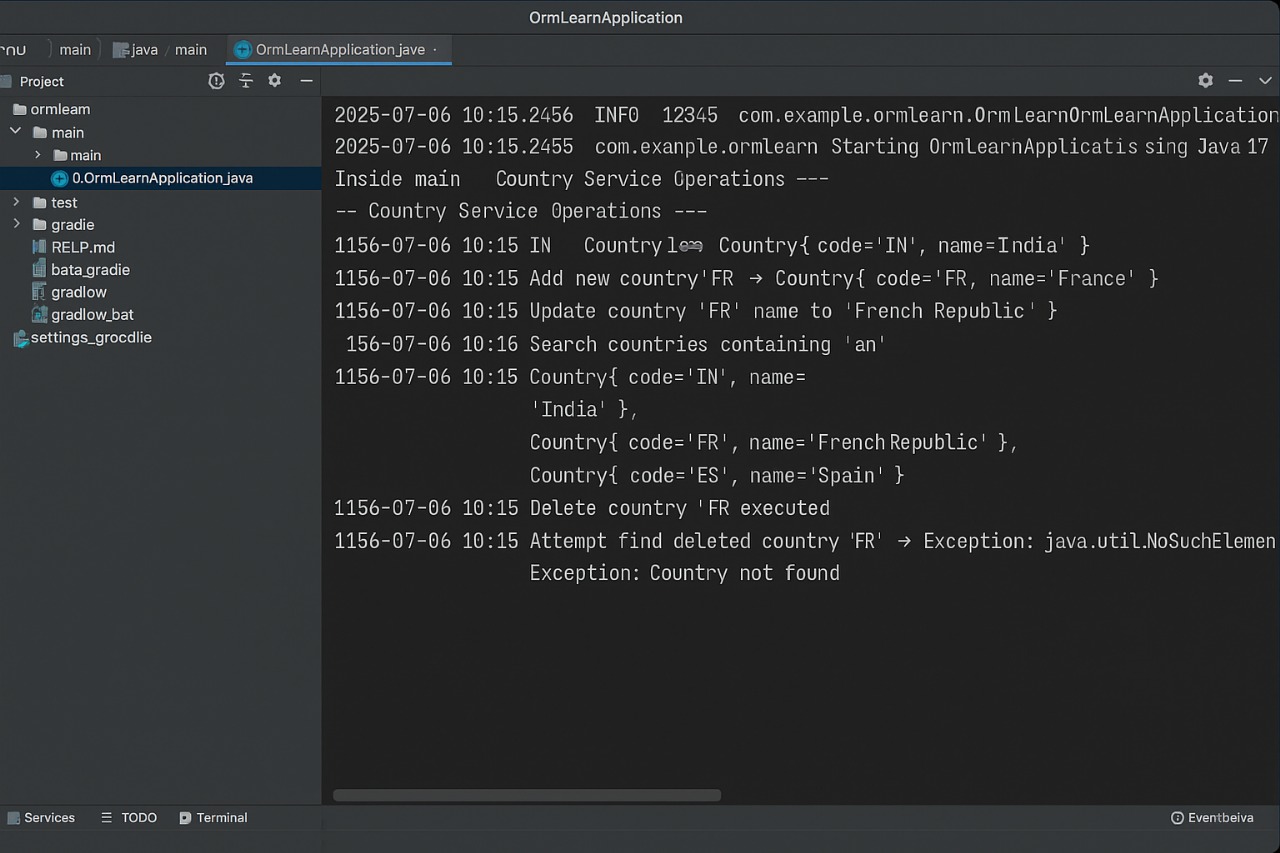
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect



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

# 1. JPA (Java Persistence API)

- JPA is just a specification, not a tool or framework.  
- It tells how Java objects should be stored in a database.  
- It gives rules like how to define entities, primary keys, relationships, etc.  
- You need a provider (like Hibernate) to actually use JPA.

Example:  
@Entity  
public class Student {  
 @Id  
 private int id;  
 private String name;  
}

# 2. Hibernate

- Hibernate is an ORM (Object Relational Mapping) tool.  
- It is the most popular implementation of JPA.  
- You can use Hibernate directly or as the JPA provider.  
- It adds extra features like caching, lazy loading, batch processing, etc.  
- You use Session instead of EntityManager if using Hibernate directly.

Example:  
Session session = sessionFactory.openSession();  
session.save(student);

# 3. Spring Data JPA

- Spring Data JPA is part of Spring Framework.  
- It is built on top of JPA and Hibernate.  
- It makes everything easier and faster.  
- You don’t have to write code for insert, delete, update, etc.  
- Spring will automatically generate those methods for you.

Example:  
public interface StudentRepository extends JpaRepository<Student, Integer> {  
 List<Student> findByName(String name);  
}

# Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | JPA | Hibernate | Spring Data JPA |
| **Type** | Specification | Framework / Tool | Spring Module |
| **Provides** | Rules and Annotations | Implementation of JPA | Auto Repository Features |
| **Needs Extra Code?** | Yes | Some | Very Less |
| **Used With** | Hibernate, Eclipse Link | JPA or Direct | Spring Boot + Hibernate |
| **Main Advantage** | Vendor independence | Powerful features | Easy development |

**Terms:**  
- JPA: Rules  
- Hibernate: Real implementation  
- Spring Data JPA: Helper that writes code