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

**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

**Create a Eclipse Project using Spring Initializr**

* Go to <https://start.spring.io/>
* Change Group as “com.cognizant”
* Change Artifact Id as “orm-learn”
* In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
* Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
* Click Generate and download the project as zip
* Extract the zip in root folder to Eclipse Workspace
* Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
* Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema.

> mysql -u root -p

mysql> create schema ormlearn;

* In orm-learn project, open src/main/resources/application.properties and include the below database and log configuration.

**application.properties**

spring.application.name=orm-learn  
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=SpringUser  
spring.datasource.password=Spring@123.  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

* Include logs for verifying if main() method is called.
* Execute the OrmLearnApplication and check in log if main method is called.  
  SME to walk through the following aspects related to the project created:

1. **src/main/java** - Folder with application code
2. **src/main/resources** - Folder for application configuration
3. **src/test/java** - Folder with code for testing the application
4. **OrmLearnApplication.java** - Walkthrough the main() method.
5. **pom.xml**
   1. Walkthrough all the configuration defined in XML file
   2. Open 'Dependency Hierarchy' and show the dependency tree.

**Country table creation**

* Create a new table country with columns for code and name.

create table country(code varchar(2) primary key, name varchar(50));

* Insert couple of records into the table

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

**Persistence Class - com.cognizant.orm-learn.model.Country**

* Open Eclipse with orm-learn project
* Create new package com.cognizant.orm-learn.model
* Create Country.java, then generate getters, setters and toString() methods.

**Country.java**

package com.cognizant.orm\_learn.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 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 + '\'' +  
 '}';  
 }  
}

**Repository Class - com.cognizant.orm-learn.CountryRepository**

* Create new package com.cognizant.orm-learn.repository
* Create new interface named CountryRepository that extends JpaRepository<Country, String>

**CountryRepository.java**

package com.cognizant.orm\_learn;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import com.cognizant.orm\_learn.model.Country;  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
  
}

**Service Class - com.cognizant.orm-learn.service.CountryService**

* Create new package com.cognizant.orm-learn.service
* Create new class CountryService
* Include @Service annotation at class level
* Autowire CountryRepository in CountryService
* Include new method getAllCountries() method that returns a list of countries.
* Include @Transactional annotation for this method
* In getAllCountries() method invoke countryRepository.findAll() method and return the result

**CountryService.java**

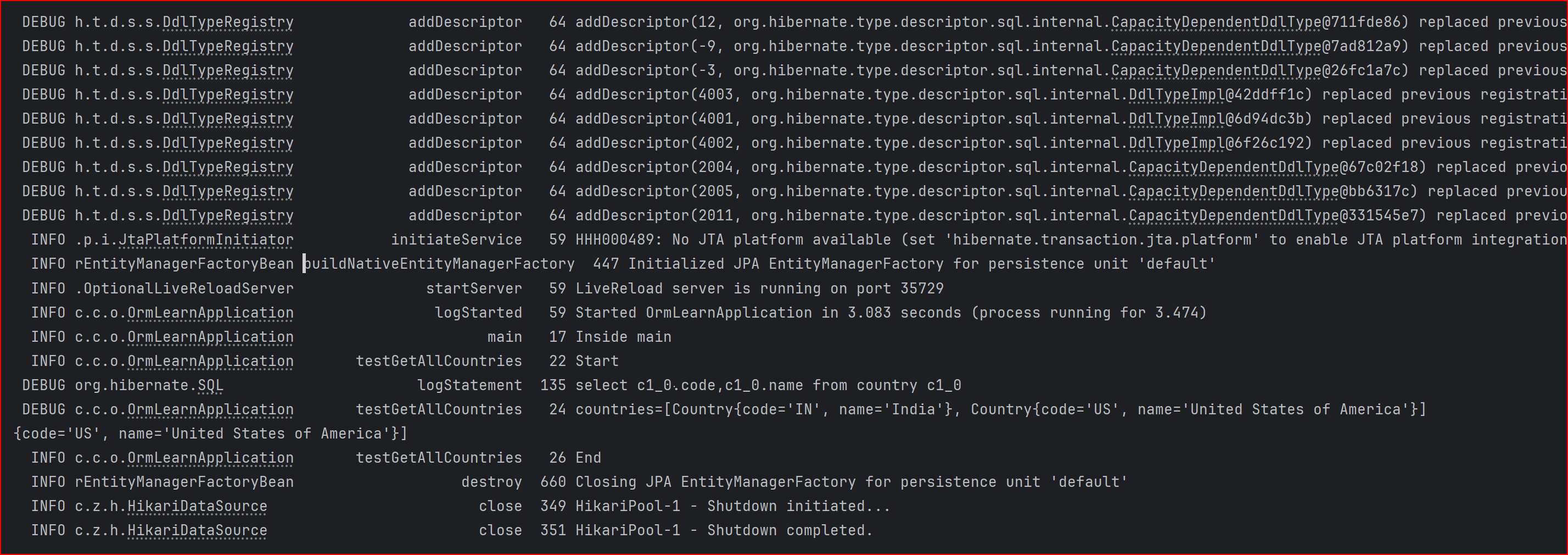
package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.CountryRepository;  
import com.cognizant.orm\_learn.model.Country;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
import java.util.List;  
  
@Service  
public class CountryService {  
 @Autowired  
 private CountryRepository countryRepository;  
 @Transactional  
 public List<Country> getAllCountries(){  
 return countryRepository.findAll();  
 }  
}

**Testing in OrmLearnApplication.java**

* Include a static reference to CountryService in OrmLearnApplication class
* Define a test method to get all countries from service.
* Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.
* Execute main method to check if data from ormlearn database is retrieved.

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.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.\*;  
@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);  
 System.*out*.println(countries.toString());  
 *LOGGER*.info("End");  
 }  
}

**OUTPUT:**

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

JPA (Java Persistence API) is a specification that defines how Java objects are mapped to relational database tables and how to manage data persistence. It provides a set of interfaces but does not contain any implementation. Hibernate is one of the most widely used implementations of JPA, offering powerful ORM features along with enhancements like caching, lazy loading, and advanced querying with HQL. Spring Data JPA is a layer built on top of JPA and typically uses Hibernate under the hood. It simplifies JPA-based data access in Spring applications by providing repository interfaces (like JpaRepository) that automatically generate queries based on method names, reducing boilerplate code and making database interactions more concise and easier to maintain. In essence, JPA is the standard, Hibernate is a JPA provider, and Spring Data JPA is a convenience layer that makes working with JPA in Spring faster and more efficient.