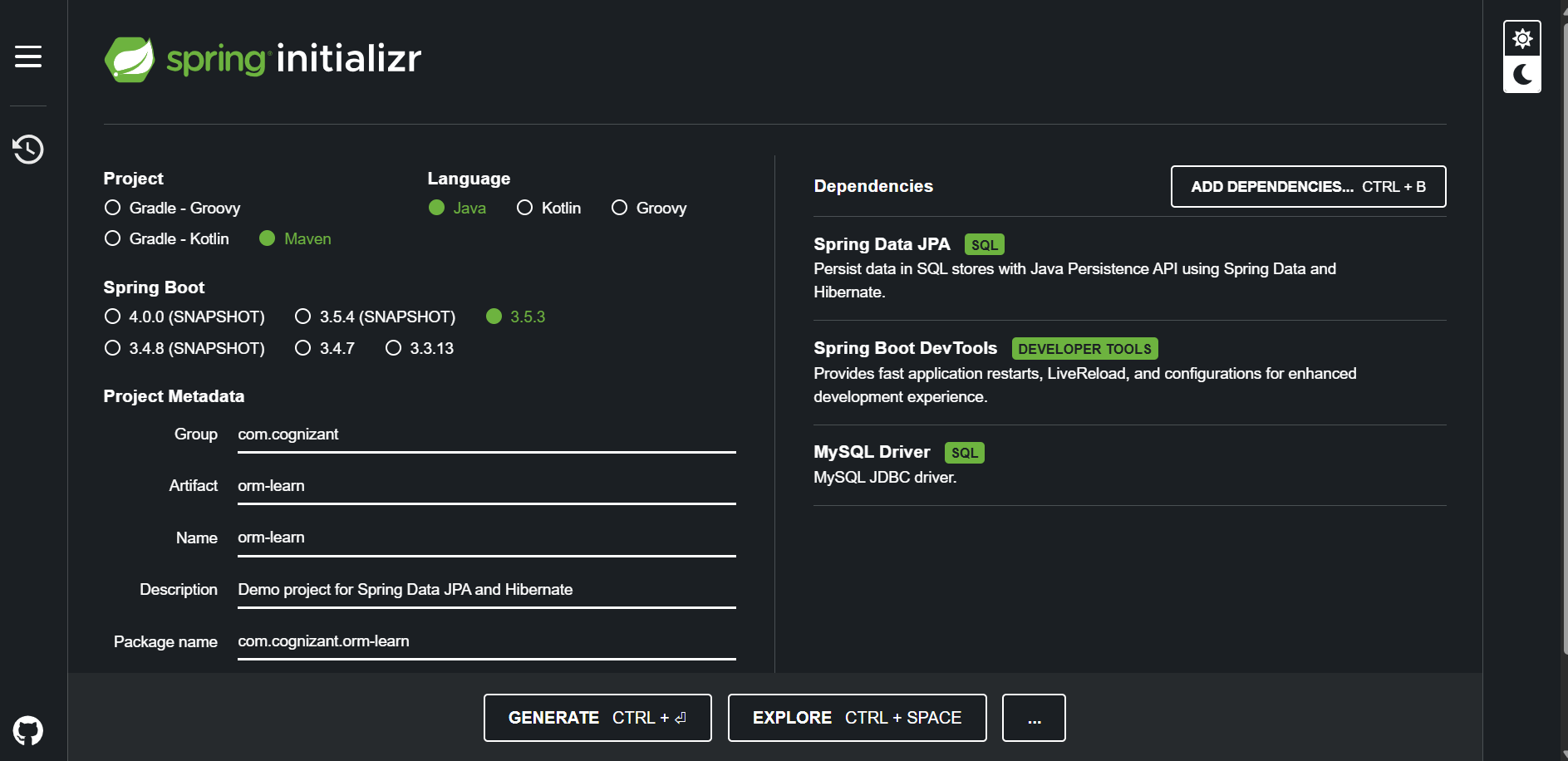
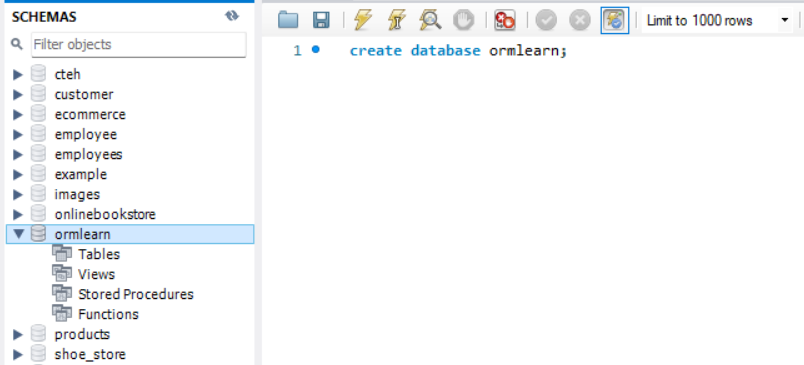
Week 3 – Hands-on : Spring Data JPA hands-on

Hands-on 1 : Configuring a basic Spring Application



**·** Creating new MySQL schema “ormlearn” in MySQL Workbench



Database and log configuration:

application.poperties:

spring.application.name=orm-learn

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# Log pattern

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# Database configuration

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=karthik2004

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

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

· Executing the OrmLearnApplication and checking in log if main method is called.

File name: OrmApplication.java

package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

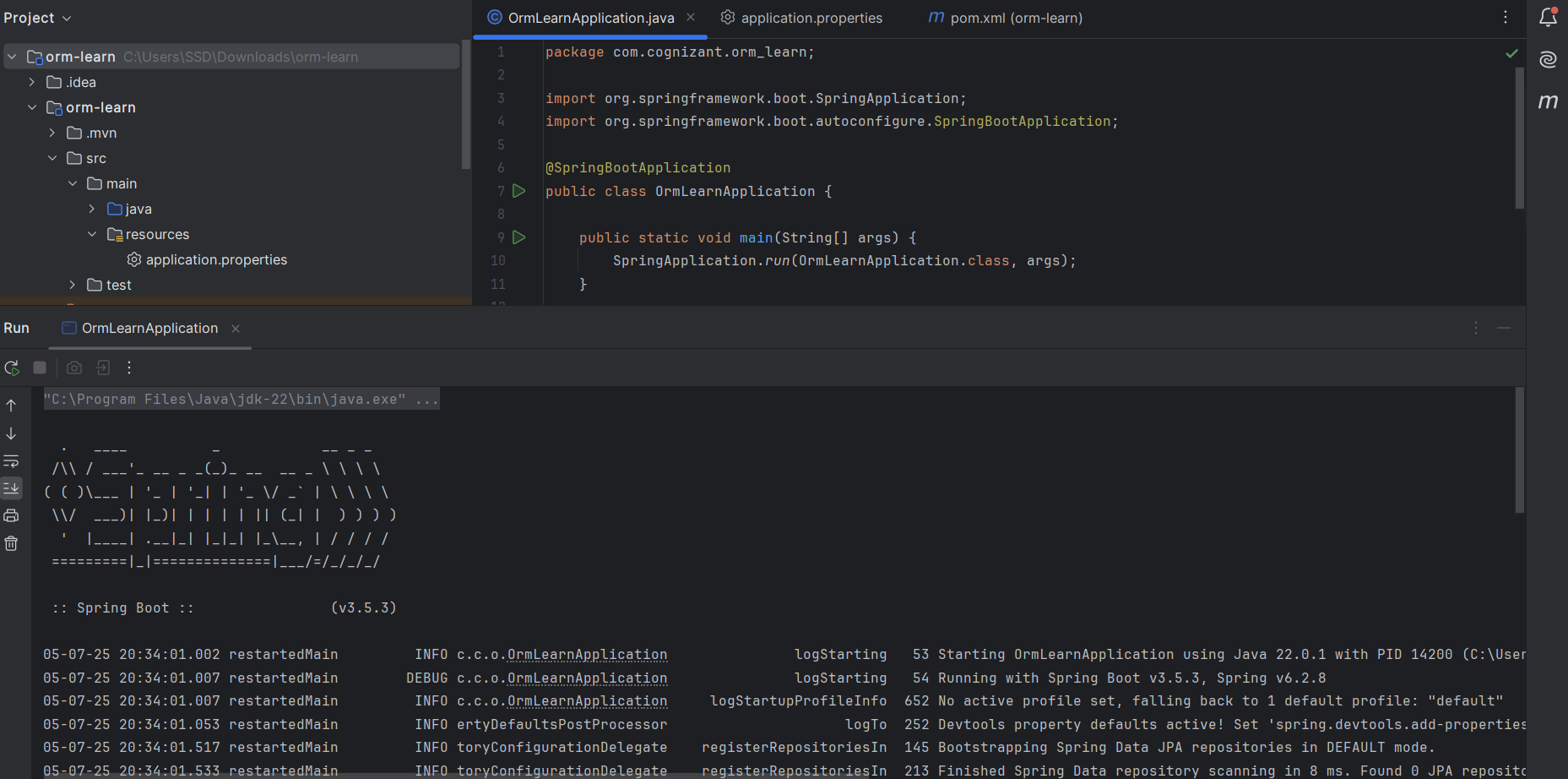
public class OrmLearnApplication {

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

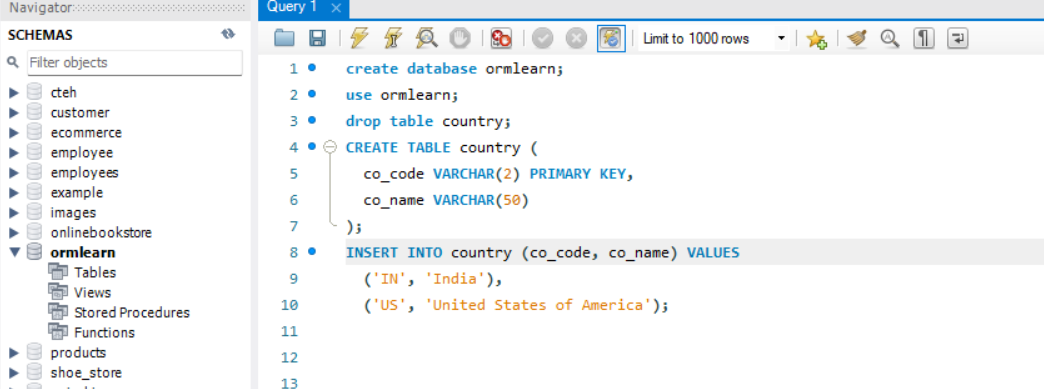
}

}

OUTPUT: 

Country table creation:

**·** Creating a new table country with columns for code and name. For sample, let us insert one country with values 'IN' and 'India' in this table.



Create Country.java, then generating getters, setters and toString() methods:

File name: Country.java

package com.cognizant.orm\_learn.Model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Table;

import org.springframework.data.annotation.Id;

@Entity

@Table(name="country")

public class Country{

@Id

@Column(name="code")

private String code;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

@Column(name="name")

private String name;

}

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

· Create new interface named CountryRepository that extends JpaRepository<Country, String>

File name: CountryRepository.java

package com.cognizant.orm\_learn.Repository;

import com.cognizant.orm\_learn.Model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country , String > {

}

Creating the Service class:

File name: CountryService.java

package com.cognizant.orm\_learn.Service;

import com.cognizant.orm\_learn.Model.Country;

import com.cognizant.orm\_learn.Repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.stream.Collectors;

@Service

public class CountryService {

private final CountryRepository countryRepository;

@Autowired

public CountryService(CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

public List<String> getAllCountryNames() {

List<Country> countries = countryRepository.findAll();

return countries.stream()

.map(Country::getName)

.collect(Collectors.toList());

}

}

Testing in 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.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

testAllCountries();

}

private void testAllCountries() {

LOGGER.info("Start");

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

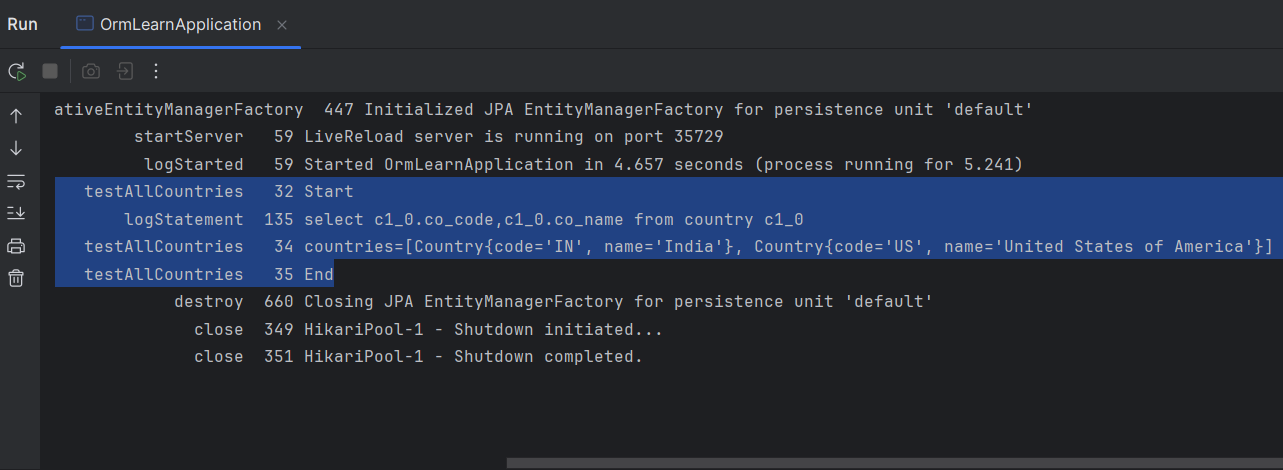
LOGGER.debug("countries={}", countries.toString());

LOGGER.info("End");

}

}

OUTPUT:



Difference between JPA, Hibernate and Spring Data JPA:

These are the main differences I’ve understood between JPA, Hibernate, and Spring Data JPA.

| **Point** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| 1 | It is just a specification | It is a concrete implementation of JPA | It is a Spring-based framework over JPA |
| 2 | Requires manual query writing | Supports HQL and criteria API | Auto-generates queries using method names |
| 3 | Needs an implementation | Provides full ORM features | Reduces boilerplate with repository layer |
| 4 | Cannot run by itself | Can work standalone | Integrates well with Spring Boot |
| 5 | No default provider | Acts as the provider | Uses JPA + Hibernate internally |