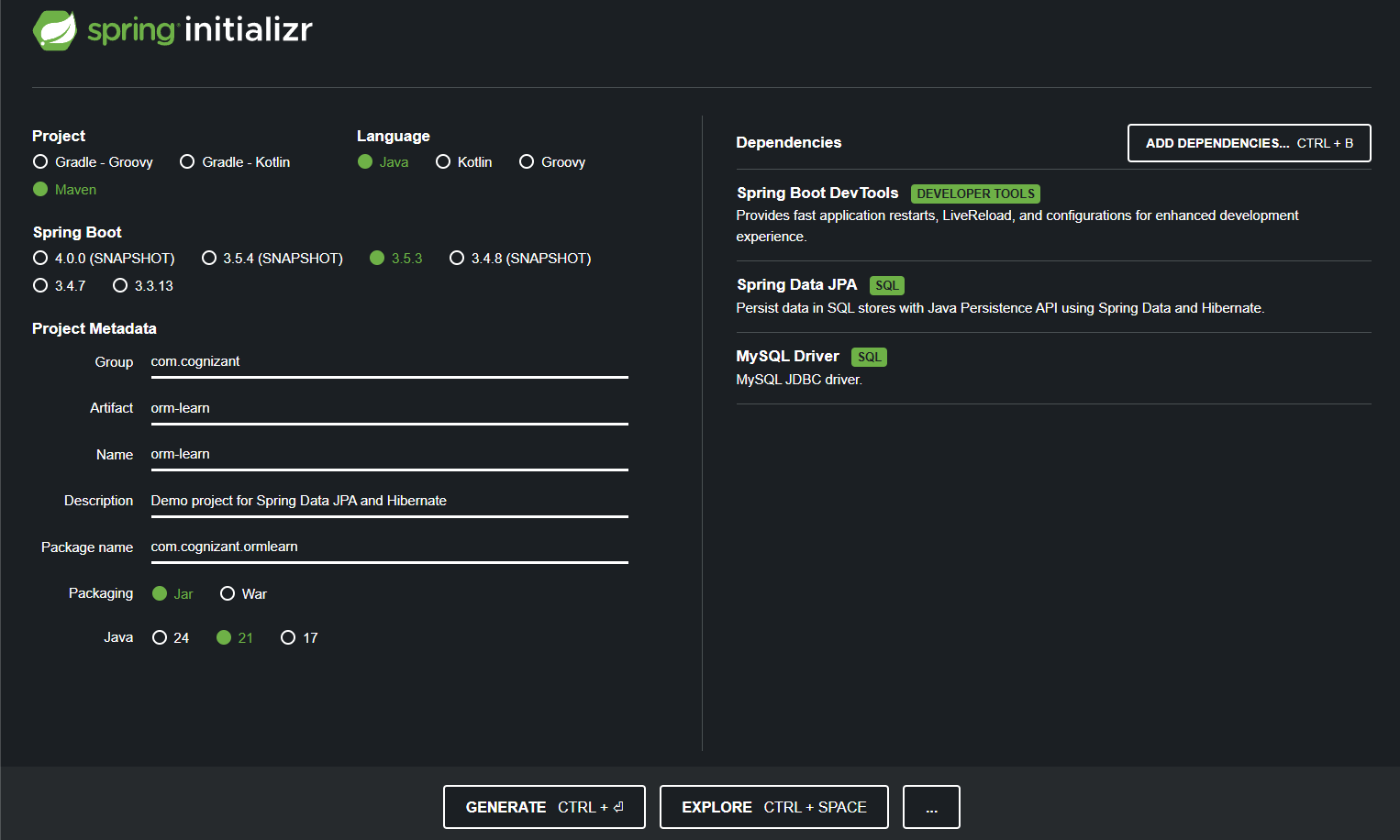
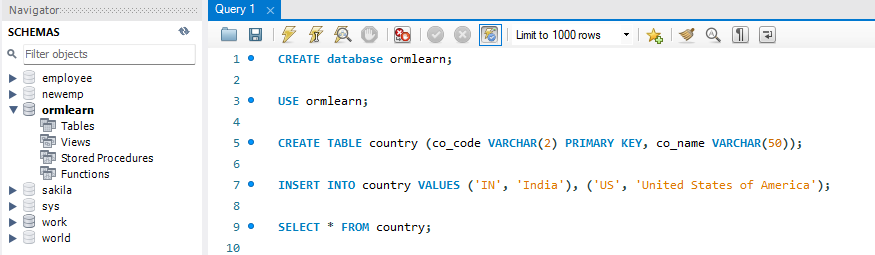
**Week 3 – Hands-on : Spring Data JPA hands-on**

Hands-on 1 : Configuring a basic Spring Application

* First we go to Spring Initializr, add the required dependencies, then generate and download the Spring Boot starter project.



* Then we create new MySQL schema “ormlearn” in MySQL Workbench



* Then we configure the “application.properties”

spring.application.name=orm-learn  
# Logging only necessary details  
logging.level.com.cognizant=debug  
logging.level.root=WARN  
  
# Logging formatting  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database Config  
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=\*\*\*\*  
  
# Hibernate  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

* Now we proceed to create “Country” entity

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
  
@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 + "]";  
 }  
}

* Now we create “CountryRepository”

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

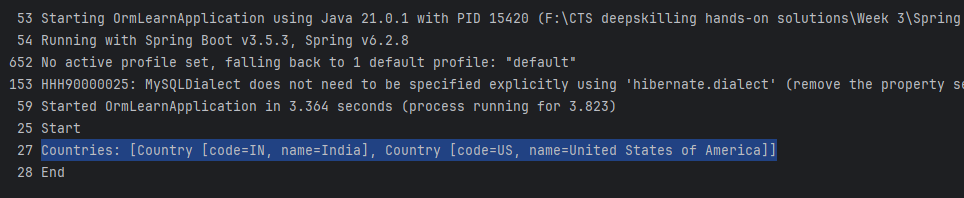
* And we proceed with creating “CountryService class”

package com.cognizant.ormlearn.service;  
  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
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();  
 }  
}

* As final step we update our main class that is “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);  
 *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");  
 }  
}

OUTPUT



Difference between JPA, Hibernate and Spring Data JPA

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| WHAT IS : | A **specification** | An **implementation** of JPA | A **wrapper abstraction** over JPA |
| PROVIDES | Interfaces & annotations | Core ORM features & JPA implementation | Auto-implements common CRUD patterns |
| REQUIRES BOILERPLATE? | Yes | Some | No (very little) |
| EXAMPLE | @Entity, EntityManager | SessionFactory, @Entity | JpaRepository, @Repository |
| USE CASES | For standard compliance | For full control | For fast development |

Here is my understanding of the key differences between JPA, Hibernate   
and Spring Data JPA.