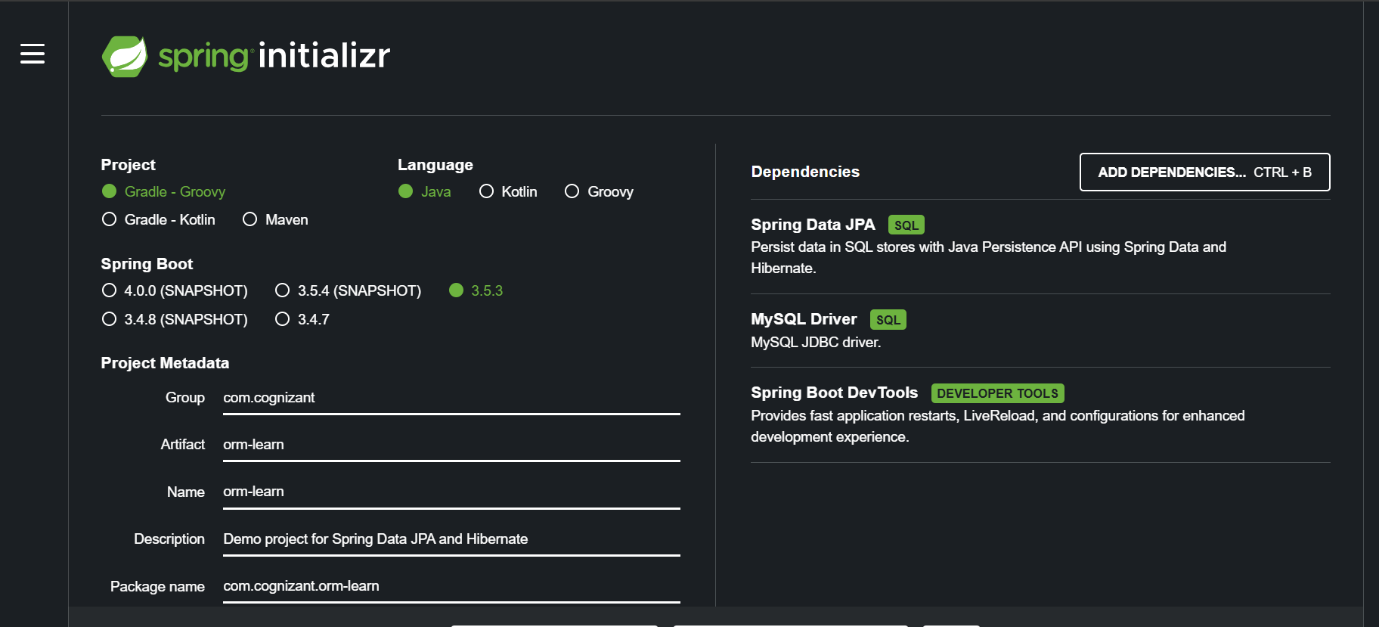
**Spring Data JPA with Spring Boot, Hibernate**

1. **spring-data-jpa-handson**

**Spring Data JPA ]**   
  
**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**

****

**SOLUTION:**

**1.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 = "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 + "]";

}

}

**2.CountryService.java**

package com.cognizant.orm\_learn.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.orm\_learn.model.Country;

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

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**3.CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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> {

}

**4.OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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;

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

import com.cognizant.orm\_learn.service.CountryService;

@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");

}

}

**5.application.properties**

# 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

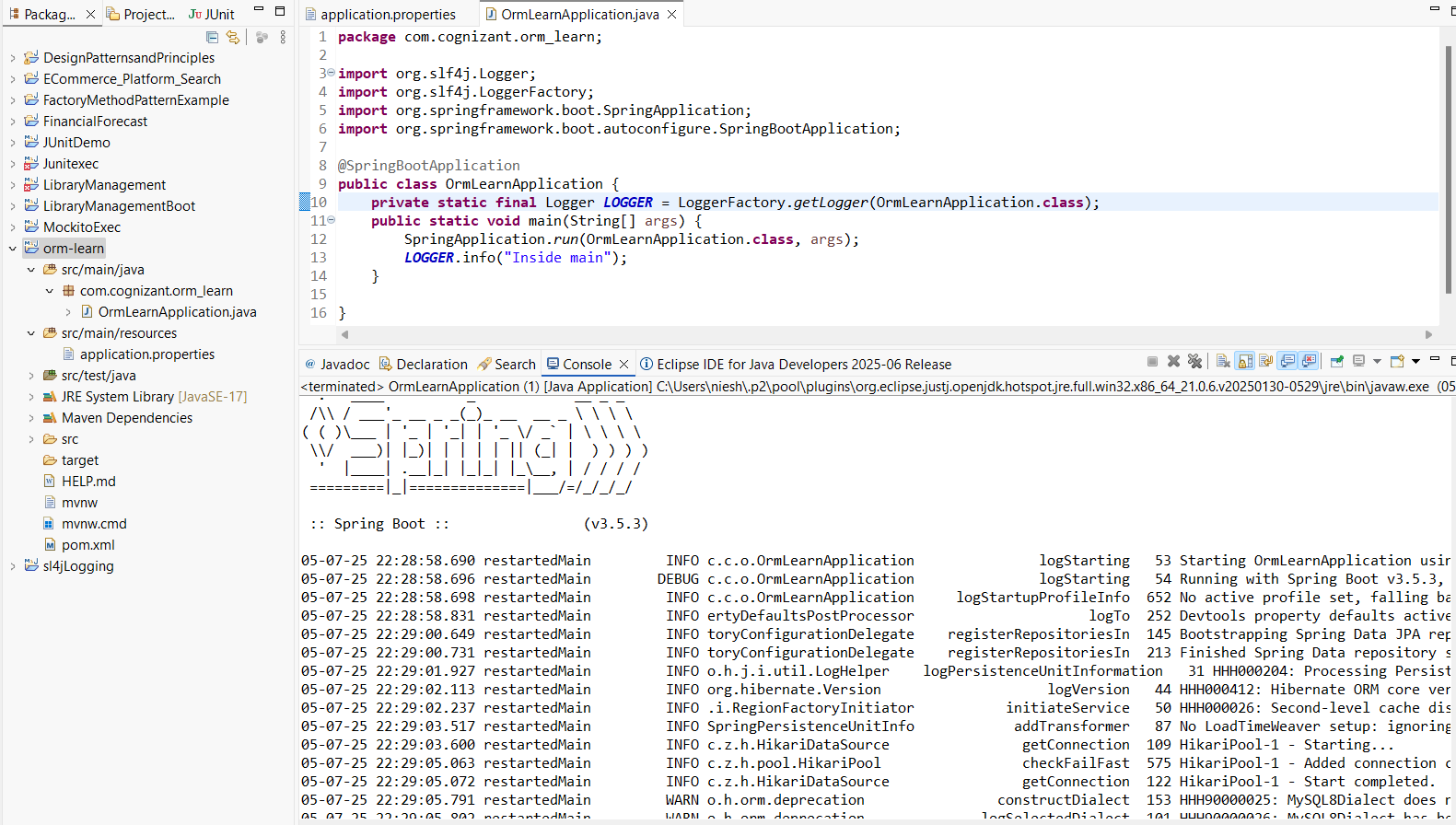
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=Gun@Strong#DB84

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**OUTPUT:**



**Hands-on 4: Difference between JPA, Hibernate and Spring Data JP**

**1. Java Persistence API (JPA)**

**What is JPA?**

JPA is a specification defined in JSR 338.

It provides standard interfaces and annotations to manage relational data using Java objects.

JPA itself does not provide any implementation. It just defines how data should be persisted.

**Example:**

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

}

**2. Hibernate**

**What is Hibernate?**

Hibernate is a complete ORM tool and the most widely used implementation of JPA.

It supports all JPA features and adds additional capabilities like:

* Lazy loading
* Caching (first-level and second-level)
* HQL (Hibernate Query Language)
* Batch processing

**Example using Hibernate:**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

1. **Spring Data JPA**

**What is Spring Data JPA?**

Spring Data JPA is a Spring module built on top of JPA.

* It simplifies the development of data access layers by:
* Reducing boilerplate code
* Automatically implementing repository methods
* Managing transactions automatically

**Example using Spring Data JPA:**

Employee.java

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

}

EmployeeRepository.java

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// No need to write save(), findAll(), etc.

}

EmployeeService.java

java

Copy code

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

Controller (Optional for REST APIs):

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping("/employees")

public ResponseEntity<?> add(@RequestBody Employee emp) {

employeeService.addEmployee(emp);

return ResponseEntity.ok("Employee added");

}

}