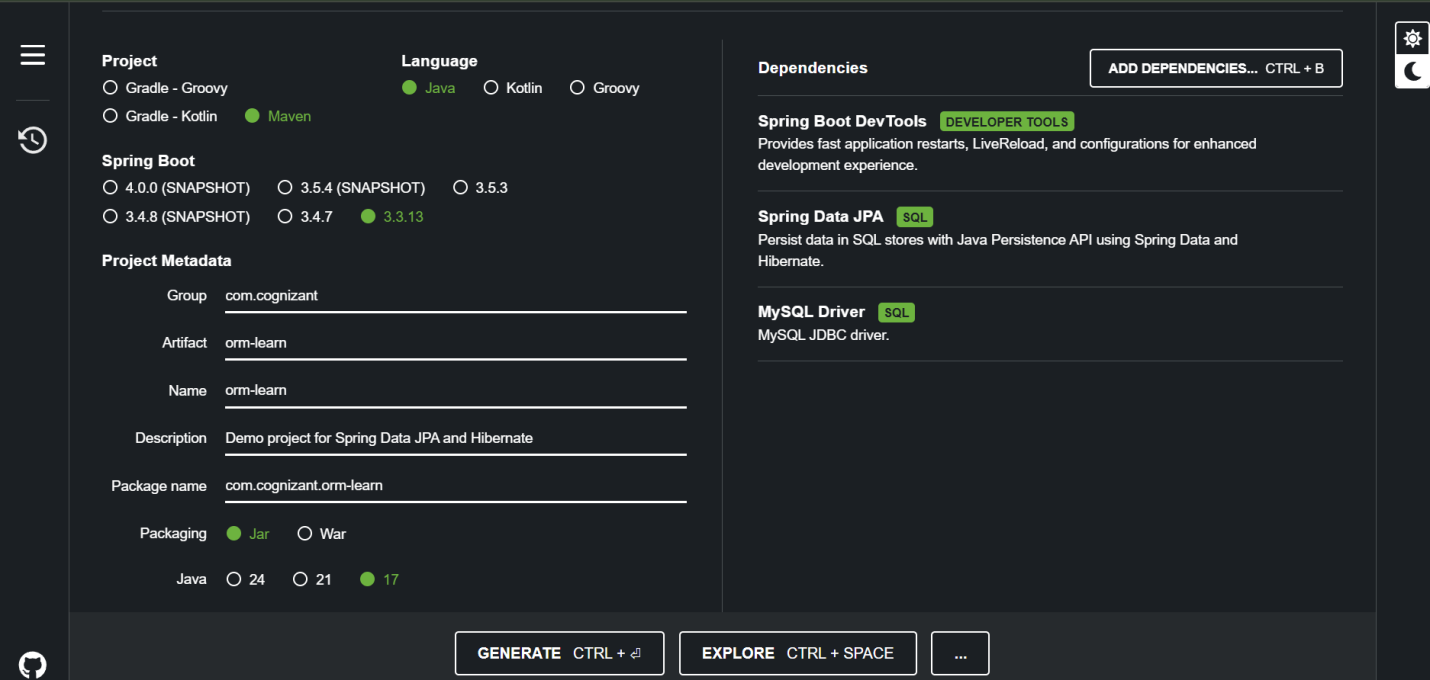
Spring Data JPA with Spring Boot, Hibernate

# Exercise: 1: Spring Data JPA - Quick Example

**Hands on 1**

**Spring Data JPA - Quick Example**



Application.properties

spring.application.name=orm-learn

# Logging

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

# MySQL DB 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=root

# JPA

spring.jpa.hibernate.ddl-auto=validate

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

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.3.13</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter Web (Optional if you use REST controllers) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- MySQL Connector -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Hibernate Core (for MySQL dialects) -->

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Ormlearn.sql

CREATE SCHEMA ormlearn;

USE ormlearn;

CREATE TABLE country (

  code VARCHAR(2) PRIMARY KEY,

  name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

Country.java

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "country")

public class Country {

*@Id*

*@Column*(name = "code")

private String code;

*@Column*(name = "name")

private String name;

// Getters and Setters

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 + "]";

}

}

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> {

}

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();

}

}

OrmApplication.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);

***LOGGER***.info("Inside main");

*countryService* = context.getBean(CountryService.class);

*testGetAllCountries*();

}

private static void testGetAllCountries() {

***LOGGER***.info("Start");

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

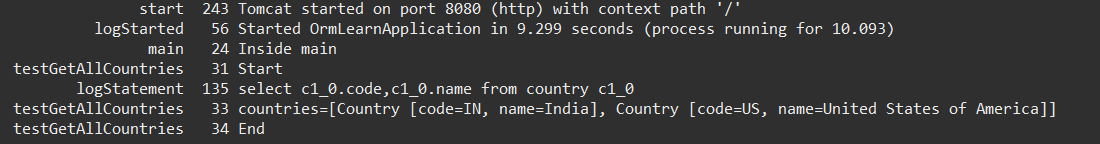
***LOGGER***.debug("countries={}", countries);

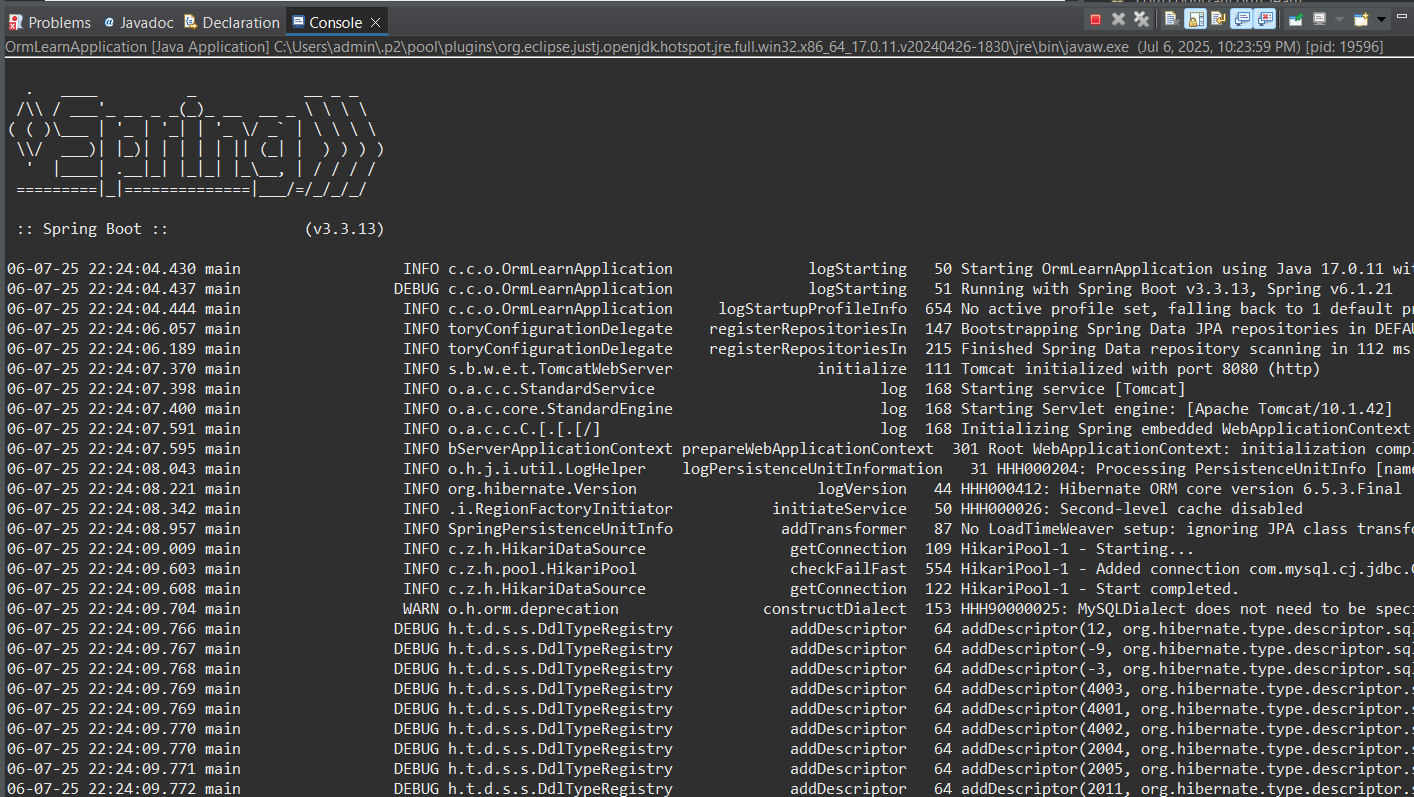
***LOGGER***.info("End");

}

}

Output





# Exercise: 2: Difference between JPA, Hibernate and Spring Data JPA

Title: Difference Between JPA, Hibernate, and Spring Data JPA

1. Java Persistence API (JPA)

* A specification (JSR 338) for object-relational mapping in Java.
* Provides a standard set of annotations and interfaces to persist Java objects to relational databases.
* Does not provide any implementation.
* Requires a provider (like Hibernate) to work.

1. Hibernate

* A popular ORM (Object-Relational Mapping) tool in Java.
* Implements the JPA specification.
* Offers additional features beyond JPA, such as caching, lazy loading, and custom queries.
* Requires manual configuration of sessions and transactions.

1. Spring Data JPA

* A Spring-based abstraction built on top of JPA.
* Simplifies data access by reducing boilerplate code.
* Manages transactions automatically and provides repository interfaces.
* Does not implement JPA directly, but relies on a provider like Hibernate underneath.

Comparison Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification | Implementation of JPA | Abstraction over JPA/Hibernate |
| Provides Implementation | No | Yes | No |
| Requires Configuration | Yes | Yes | Minimal |
| Boilerplate Code | Moderate | High | Low |
| Transactions | Manual | Manual | Automatic |

Code Comparison:

Hibernate Example:

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;

}

Spring Data JPA Example:

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

@Service

public class EmployeeService {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Transactional

    public void addEmployee(Employee employee) {

        employeeRepository.save(employee);

    }

}

Output

2025-07-06 22:15:02  INFO 1234 --- [           main] c.e.EmployeeService         : Added employee: John Doe (ID: 101)

2025-07-06 22:15:03  INFO 1234 --- [           main] o.s.web.servlet.DispatcherServlet : Completed initialization