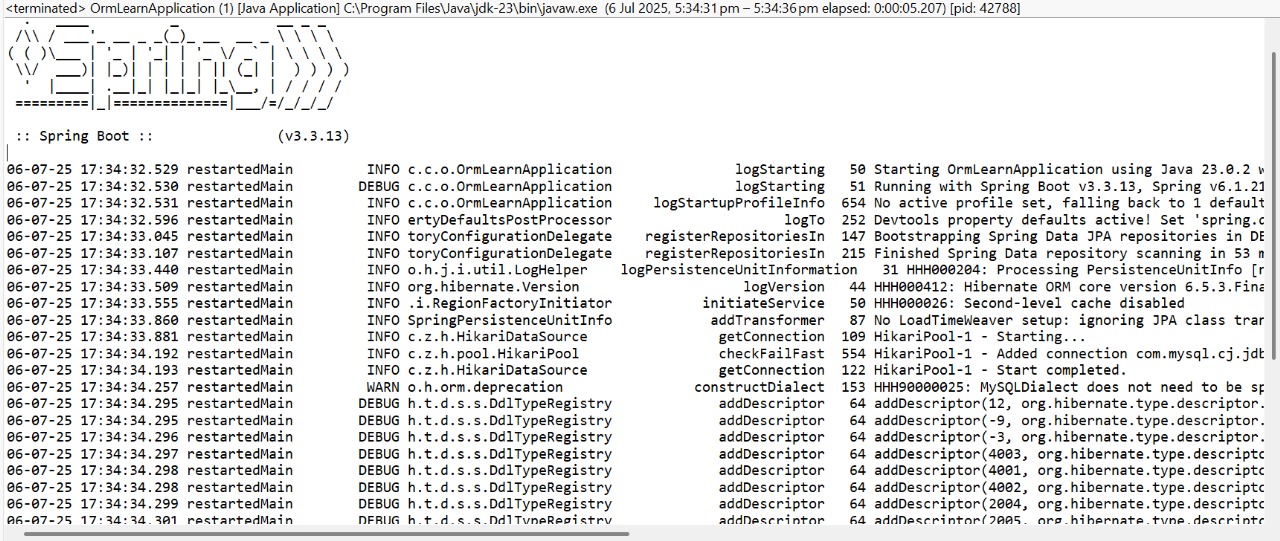
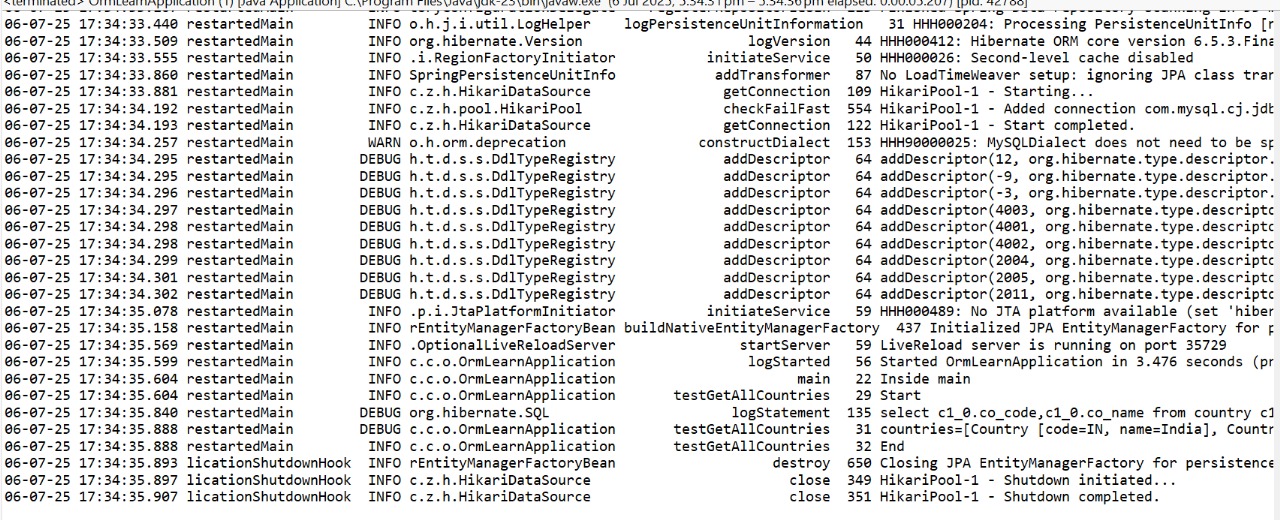
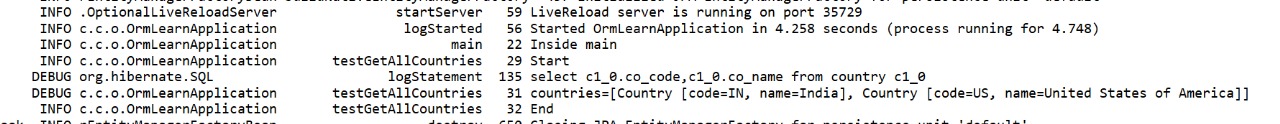
**Hands on 1**

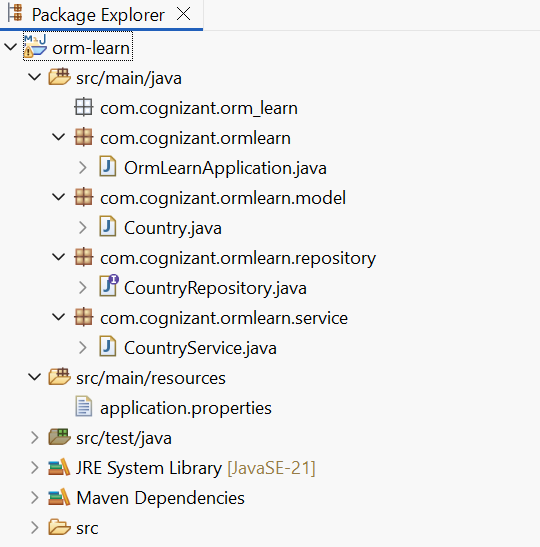
**Spring Data JPA - Quick Example** 

Output:-









**com.cognizant.ormlearn -> OrmLearnApplication.java**

**package** com.cognizant.ormlearn;

**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.ormlearn.model.Country;

**import** com.cognizant.ormlearn.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");

}

}

**com.cognizant.ormlearn.model -> Country.java**

**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 + "]";

}

}

**com.cognizant.ormlearn.repository -> CountryRepository.java**

**package** com.cognizant.ormlearn.repository;

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

**import** org.springframework.stereotype.Repository;

**import** com.cognizant.ormlearn.model.Country;

@Repository

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

}

**com.cognizant.ormlearn.service -> CountryService.java**

**package** com.cognizant.ormlearn.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** com.cognizant.ormlearn.model.Country;

**import** com.cognizant.ormlearn.repository.CountryRepository;

**import** jakarta.transaction.Transactional;

@Service

**public** **class** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Transactional

**public** List<Country> getAllCountries() {

**return** countryRepository.findAll();

}

}

**src/main/resources -> application.properties**

spring.application.name=orm-learn

# Logging Configuration

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

# Database Connection

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 Settings

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/>

</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>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.3.0</version> <!-- Ensure version is compatible -->

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>21</source>

<target>21</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Comparison of Hibernate and Spring Data JPA**

1. **Setup and Boilerplate Code**

**Hibernate**

* Requires us to manually manage the SessionFactory, Session, and Transaction classes
* **Example:**

Session session = factory.openSession();  
Transaction tx = session.beginTransaction();

* Need to close sessions in finally blocks.

**Spring Data JPA**

* No boiler plate; it uses the JpaRepository and the methods are already implemented.
* **Example:**

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

1. **Transaction Management**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Hibernate** | **Spring Data JPA** |
| **Control** | Manual (begin, commit, rollback) | Automatic (@Transactional) |
| **Rollback** | Manual (tx.rollback()) | Auto-rollback on exceptions |

**3. Saving an Entity**

**Hibernate:**

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

* Uses session.save(employee) to save an entity.
* After saving, it will return an ID (if one was generated). You need to manage the session.

**Spring Data JPA:**

employeeRepository.save(employee);

* Uses employeeRepository.save(employee).
* The save() method is provided by JpaRepository.
* No need to manage the session or check manually for exceptions.

1. **Exception Handling**

* **Hibernate:** Catch HibernateException and rollback manually.

**Example:**

catch (HibernateException e) {

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

e.printStackTrace();

}

* **Spring Data JPA:** Exceptions auto-translated to DataAccessException. No need for manual rollback if @Transactional is used.

**5. Code Readability**

**Hibernate:**

* More wordy, because of doing session and transaction management manually.
* More risk for error if something isn't closed properly.

**Spring Data JPA:**

* Clean and concise due to abstraction.
* Less boilerplate code, easier to maintain.

**6. Dependency Injection**

**Hibernate:** None (manual SessionFactory setup).

**Spring Data JPA:**  Built-in (@Autowired repositories).

**7. Default Methods**

* **Hibernate:** Basic CRUD via Session.
* **Spring Data JPA:** Advanced methods (findAll(), deleteById(), etc.).