**Mandatory hands-on**

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

**Exercise 1: Spring Data JPA - Quick Example**

**Code:**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

}

**Pom.xml**

<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*

*http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<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>

<packaging>jar</packaging>

<parent>

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

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

<version>2.7.18</version>

<relativePath/>

</parent>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.8.1</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

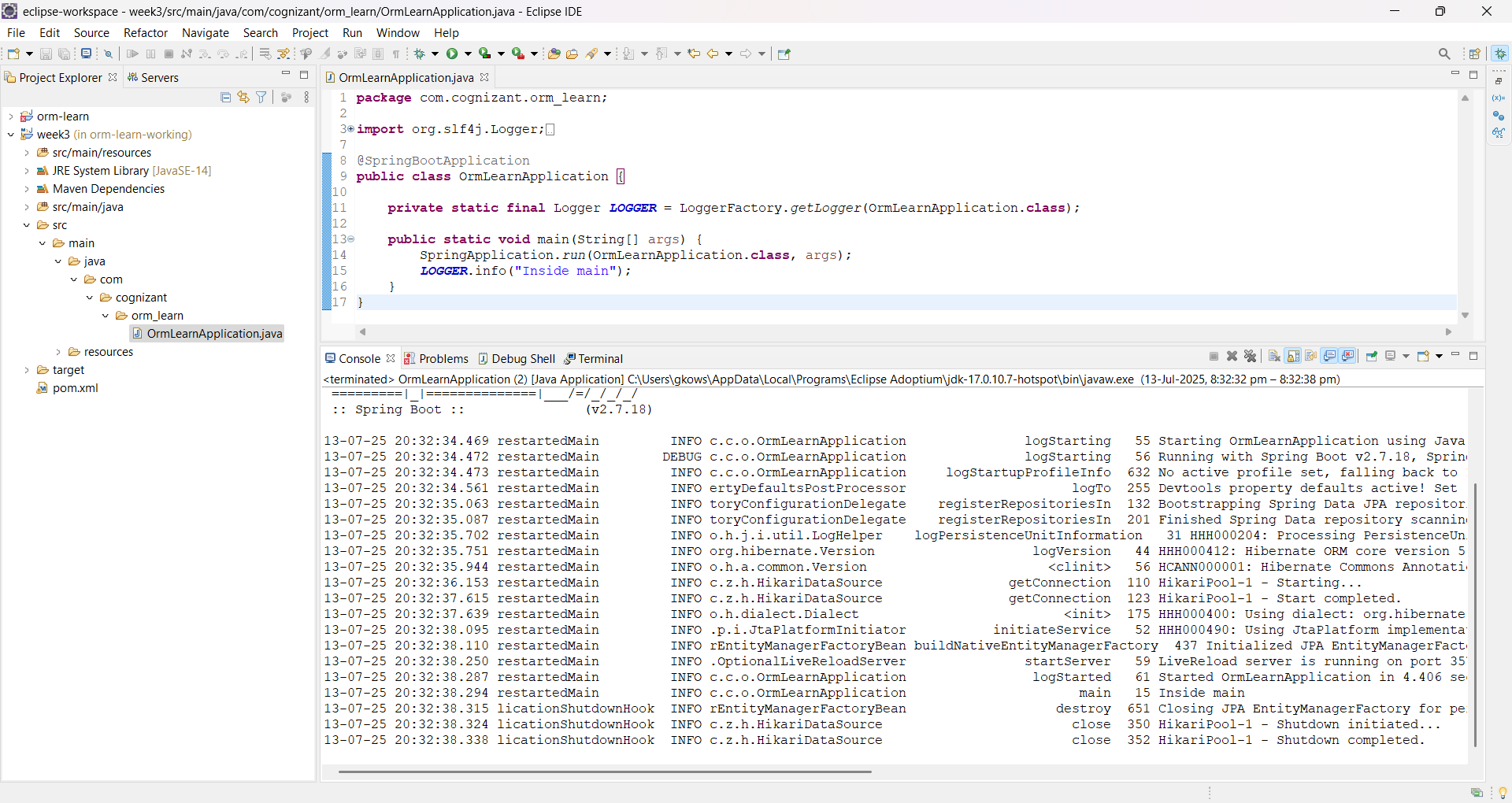
</plugin>

</plugins>

</build>

</project>

**Output:**

****

**Exercise 4: Difference between JPA, Hibernate and Spring Data JPA**

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

* **JPA** is a **Java specification** (JSR 338) for accessing, persisting, and managing data between Java objects and a relational database.
* It is **not an implementation** itself, but defines a set of interfaces and annotations for ORM (Object Relational Mapping).
* Example: @Entity, @Id, @OneToMany are JPA annotations.
* **Hibernate** is the most commonly used implementation of the JPA specification.

**2. Hibernate**

* **Hibernate** is a **popular ORM tool** that implements the JPA specification.
* It provides a full set of features like lazy loading, caching, HQL (Hibernate Query Language), etc.
* Need to handle **session, transaction, exception** management manually.
* Hibernate is the **real working engine** behind most JPA-based applications.

**3. Spring Data JPA**

* Spring Data JPA is a **Spring framework abstraction layer** on top of JPA/Hibernate.
* It simplifies data access using **interfaces** and **custom query methods**.
* It **reduces boilerplate** code (no need to write implementation for basic CRUD).
* Spring handles **transactions**, **EntityManager**, and other infrastructure.
* Spring Data JPA is a **productivity booster** that uses Hibernate or other JPA providers underneath.

**Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| Type | Specification | Implementation | Abstraction |
| Boilerplate Code | Moderate | High | Very Low |
| Requires Manual Transaction Handling | N/A | Yes | No |
| Uses Annotations | Yes | Yes | Yes |
| Provided By | Oracle (JSR 338) | Red Hat | Spring Framework |
| Dependency on Provider | No | Yes (e.g., Hibernate) | Yes (e.g., Hibernate) |

**Code Comparison**

**Hibernate Code**

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 Code**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}