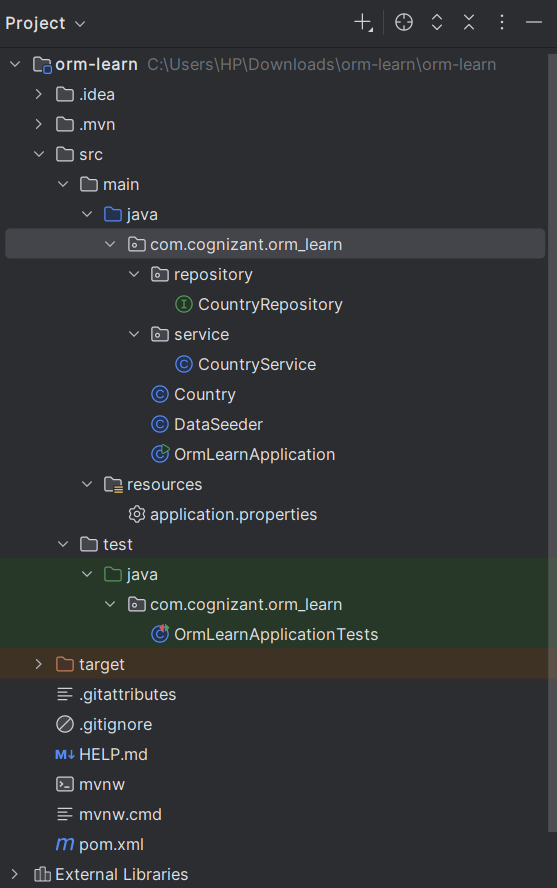
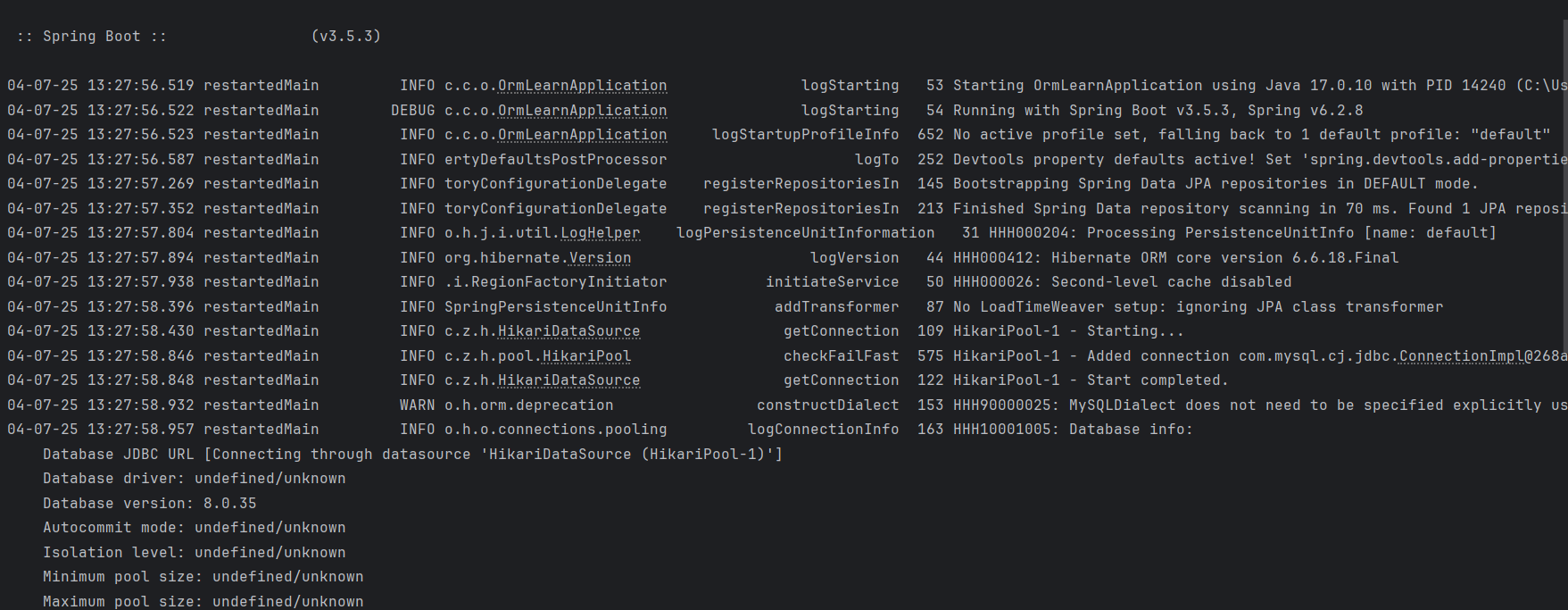
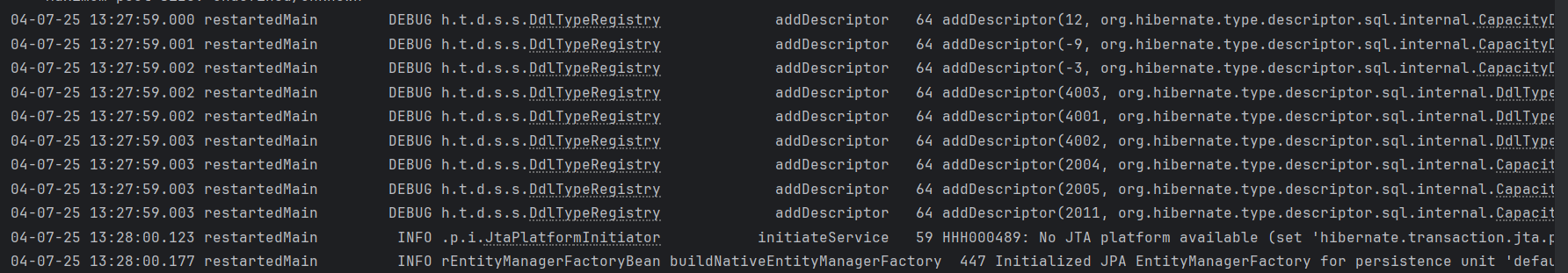
**Spring Data JPA - Quick Example**

****

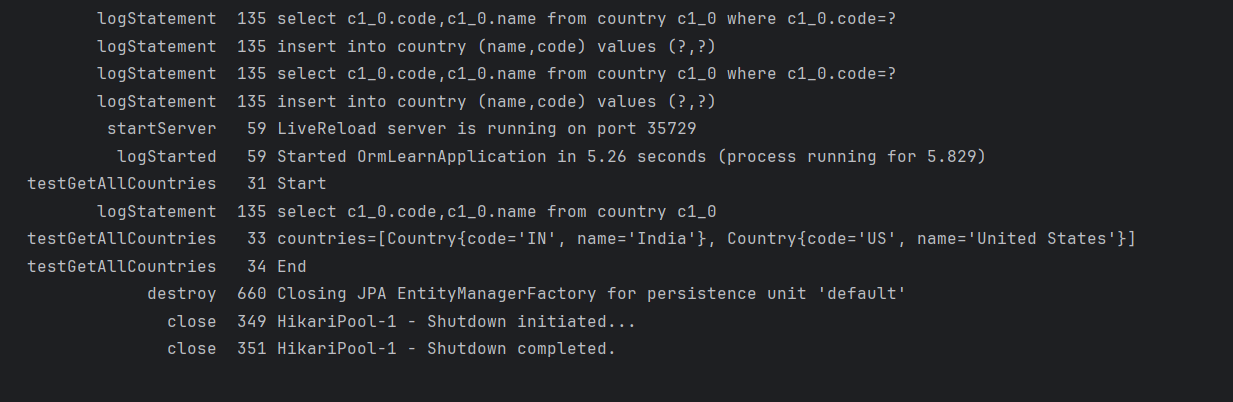
|  |
| --- |
| **Country.java**  **package com.cognizant.orm\_learn;  import jakarta.persistence.Entity; import jakarta.persistence.Id; import jakarta.persistence.Table;  @Entity @Table(name = "country") public class Country {   @Id  private String code;   private String name;   public Country() {  }   public Country(String code, String name) {  this.code = code;  this.name = 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 + '\'' + '}';  } }**  **CountryRepository.java**  **package com.cognizant.orm\_learn.repository;  import com.cognizant.orm\_learn.Country; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  @Repository public interface CountryRepository extends JpaRepository<Country, String> { }**  **CountryService.java**  **package com.cognizant.orm\_learn.repository;  import com.cognizant.orm\_learn.Country; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;  @Repository public interface CountryRepository extends JpaRepository<Country, String> { }**  **OrmLearnApplication.java**  **package com.cognizant.orm\_learn;  import com.cognizant.orm\_learn.service.CountryService; import com.cognizant.orm\_learn.Country;  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");  } }**  **DataSeeder.java**  **package com.cognizant.orm\_learn;  import com.cognizant.orm\_learn.repository.CountryRepository; import jakarta.annotation.PostConstruct; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;  @Component public class DataSeeder {   @Autowired  private CountryRepository countryRepository;   @PostConstruct  public void init() {  System.*out*.println("DataSeeder: @PostConstruct called");  countryRepository.save(new Country("IN", "India"));  countryRepository.save(new Country("US", "United States"));  } }**  **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>3.5.3</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</description>  <url/>  <licenses>  <license/>  </licenses>  <developers>  <developer/>  </developers>  <scm>  <connection/>  <developerConnection/>  <tag/>  <url/>  </scm>  <properties>  <java.version>17</java.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>   <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <dependency>  <groupId>com.mysql</groupId>  <artifactId>mysql-connector-j</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>   <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>**  **application.properties**  **spring.application.name=orm-learn  # Logging configuration logging.level.org.springframework=info logging.level.com.cognizant=debug logging.level.org.hibernate.SQL=debug logging.level.org.hibernate.type.descriptor.sql=debug   logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  # MySQL DB 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=triptisql@  # Hibernate config spring.jpa.hibernate.ddl-auto=update spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect** |
|  |
|  |

**OUTPUT -:**

****

****

****

****

**Difference between JPA, Hibernate and Spring Data JPA**

|  |  |  |
| --- | --- | --- |
| **JPA** | **HIBERNATE** | **SPRING DATA JPA** |
| 1. A specification (not an implementation) provided by Java for ORM | 1. A JPA implementation (also an ORM framework in its own right). | 1. A Spring project that makes it easier to use JPA in Spring applications. |
| 2. Part of Java EE | 2. Provides the actual logic to persist and manage data using the specifications of JPA | 2. It uses JPA under the hood and commonly Hibernate as the default provider. |
| 3. It defines standard APIs and annotations for managing relational data in Java applications. | 3. Hibernate offers additional capabilities beyond JPA LIKE HQL (hibernate query language) | 3. Automatically implements repositories (CRUD, paging, sorting) |
| 4. JPA needs a provider/implementation like Hibernate, EclipseLink, or OpenJPA to work. | 4. Hibernate can be used as a standalone ORM, even without JPA. | 4. Repository interfaces , Query annotation |