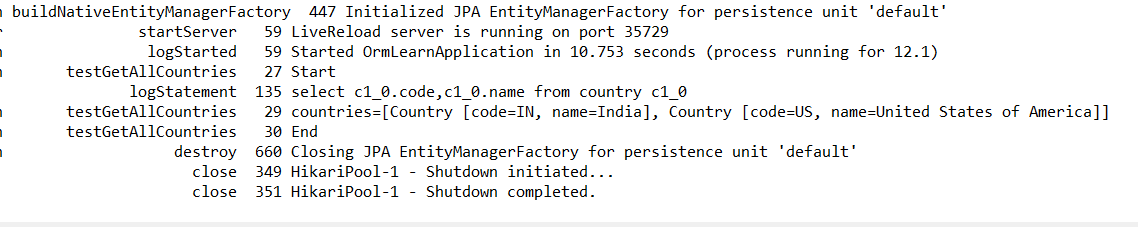
**Spring Data JPA with Spring Boot, Hibernate**

**Spring Data JPA - Quick Example**

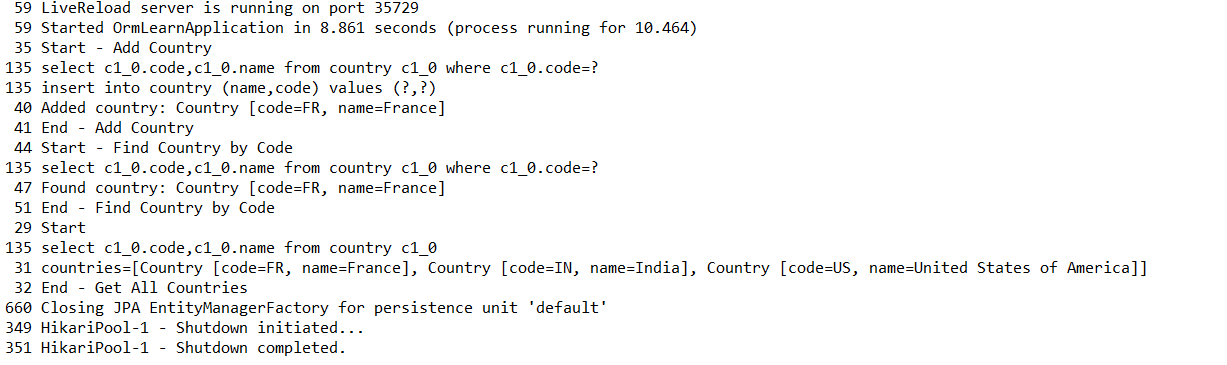
OUTPUT:



**Additional Exercise:**

1. Implement services for managing Country
2. Find a country based on country code
3. Add a new country

OUTPUT:



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

**JPA (Java Persistence API)**

JPA is a Java specification that defines a standard for object-relational mapping (ORM) in Java.

It provides interfaces and annotations but does not implement any logic to interact with databases.

It uses JPA annotations like @Entity, @Table, @Id, etc to map Java objects to database tables.

It can be easily be switched between different JPA providers without changing code structure.

**Hibernate**

Hibernate is the most popular implementation of JPA, providing the actual logic for persistence.

It offers many advanced features beyond the JPA specification, such as first-level and second-level caching, lazy/eager fetching, etc.

Hibernate provides its own powerful Session API in addition to JPA's EntityManager.

It generates and optimizes SQL queries internally, reducing manual query writing.

Hibernate is a mature and robust framework used in many enterprise applications globally.

**Spring Data JPA**

Spring Data JPA is a higher-level abstraction built on top of JPA and Hibernate to simplify data access layers.

It minimizes boilerplate code by letting developers define only interfaces, Spring auto-generates implementations.

Developers can define custom queries just by naming methods, Spring handles query generation.

Works seamlessly with Spring Boot to provide out-of-the-box configuration and starter dependencies.

Greatly improves development speed and code readability, making it ideal for modern enterprise applications.